Pontifícia Universidade Católica
do Rio de Janeiro

Loïck Geoffrey Hodonou

Geração e Detecção de Objetos em Documentos por Modelos de Redes Neurais de Aprendizagem Profunda (DeepDocGen)

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Eléctrica, do Departamento de Engenharia Eléctrica da PUC-Rio
.

Orientador : Prof. Marco Aurelio Cavalcanti Pacheco
Co-orientador: Evelyn Conceição Santos Batista

Rio de Janeiro
Outubro de 2024

Pontifícia Universidade Católica
do Rio de Janeiro

Loïck Geoffrey Hodonou

Geração e Detecção de Objetos em Documentos por Modelos de Redes Neurais de Aprendizagem Profunda (DeepDocGen)

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Eléctrica da PUC-Rio . Aprovada pela Comissão Examinadora abaixo:

Prof. Marco Aurelio Cavalcanti Pacheco
Orientador
Departamento de Engenharia Eléctrica – PUC-Rio

Evelyn Conceição Santos Batista
Co-orientador
Departamento de Engenharia Eléctrica – PUC-Rio

Prof. Karla Figueiredo
UERJ

Prof. Leonardo Alfredo Forero Mendoza
UERJ

Diogo da Silva Margalhaes Gomes
PETROBRAS

Jose David Bermudez Castro
McMaster University

Rio de Janeiro, 04 de Outubro de 2024

Loïck Geoffrey Hodonou

Graduado em Engenharia Eletrotécnica, Sistemas Industriais (SI), pela pela Universidade Católica da África Ocidental; Unidade Universitária de Cotonou (UCAO-UUC) no Benim.

Ficha Catalográfica

Hodonou,Loïck Geoffrey

Geração e Detecção de Objetos em Documentos por Modelos de Redes Neurais de Aprendizagem Profunda (DeepDocGen) / Loïck Geoffrey Hodonou; orientador: Marco Aurelio Cavalcanti Pacheco; co-orientador: Evelyn Conceição Santos Batista. – 2024.

131 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Eléctrica, 2024.

Inclui bibliografia

1. – Teses. 2. Aprendizado de máquina. 3. Aprendizado profundo. 4. Detecção de objetos. 5. Active learning. 6. Geração de dados. 7. Dados sintéticos. 8. Análise do layout do documento. 9. Processamento automático de documentos. 10. Extração de conteúdo. I. Cavalcanti Pacheco, Marco Aurelio. II. Batista, Evelyn. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Eléctrica. IV. Título.

CDD: 004

Aos meus pais, por seu apoio
e incentivo.

Agradecimentos

Antes de mais, gostaria de expressar a minha sincera gratidão a Deus pelas inúmeras bênçãos que Ele graciosamente me concedeu ao longo da minha vida. Sem a Sua orientação e apoio, nada disto seria possível.

Aos meus queridos pais, Fortuné Thierry Hodonou e Noelli Victorine Padonou, estou grato pela sólida educação que me proporcionaram, bem como pelo seu apoio inabalável e conselhos inestimáveis que me permitiram aproveitar a oportunidade de viajar e prosseguir os meus estudos no estrangeiro.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Além disso, gostaria de estender meus sinceros agradecimentos a PUC-Rio, ao Laboratório de Inteligência Artificial Aplicada (ICA) e a Petrobras pelo apoio financeiro e institucional, sem os quais a conclusão deste trabalho teria sido impossível.

Ao meu orientador de dissertação, o professor Marco Aurélio Cavalcanti Pacheco, e à minha co-orientadora, a Evelyn Batista, assim como a todo o pessoal do laboratório ICA, gostaria de expressar minha mais profunda gratidão pelos seus preciosos conselhos e pelo apoio incondicional durante todo este projeto. Coisas boas ainda estão por vir.

Resumo

Hodonou,Loïck Geoffrey; Cavalcanti Pacheco, Marco Aurelio; Batista, Evelyn. **Geração e Detecção de Objetos em Documentos por Modelos de Redes Neurais de Aprendizagem Profunda (DeepDocGen)**. Rio de Janeiro, 2024. 131p. Dissertação de Mestrado – Departamento de Engenharia Eléctrica, Pontifícia Universidade Católica do Rio de Janeiro.

A eficácia dos sistemas de conversação homem-máquina, como chatbots e assistentes virtuais, está diretamente relacionada à quantidade e qualidade do conhecimento disponível para eles. Na era digital, a diversidade e a qualidade dos dados aumentaram significativamente, estando disponíveis em diversos formatos. Entre esses, o PDF (Portable Document Format) se destaca como um dos mais conhecidos e amplamente utilizados, adaptando-se a variados setores, como empresarial, educacional e de pesquisa. Esses arquivos contêm uma quantidade considerável de dados estruturados, como textos, títulos, listas, tabelas, imagens, etc.

O conteúdo dos arquivos PDF pode ser extraído utilizando ferramentas dedicadas, como o OCR (Reconhecimento Ótico de Caracteres), o PdfMiner, Tabula e outras, que provaram ser adequadas para esta tarefa. No entanto, estas ferramentas podem deparar-se com dificuldades quando lidam com a apresentação complexa e variada dos documentos PDF. A exatidão da extração pode ser comprometida pela diversidade de esquemas, formatos não normalizados e elementos gráficos incorporados nos documentos, o que frequentemente leva a um pós-processamento manual.

A visão computacional e, mais especificamente, a detecção de objetos, é um ramo do aprendizado de máquina que visa localizar e classificar instâncias em imagens utilizando modelos de detecção dedicados à tarefa, e está provando ser uma abordagem viável para acelerar o trabalho realizado por algoritmos como OCR, PdfMiner, Tabula, além de melhorar sua precisão.

Os modelos de detecção de objetos, por serem baseados em aprendizagem profunda, exigem não apenas uma quantidade substancial de dados para treinamento, mas, acima de tudo, anotações de alta qualidade pois elas têm um impacto direto na obtenção de altos níveis de precisão e robustez. A diversidade de layouts e elementos gráficos em documentos PDF acrescenta uma camada

adicional de complexidade, exigindo dados anotados de forma representativa para que os modelos possam aprender a lidar com todas as variações possíveis.

Considerando o aspecto volumoso dos dados necessários para o treinamento dos modelos, percebemos rapidamente que o processo de anotação dos dados se torna uma tarefa tediosa e demorada que requer intervenção humana para identificar e etiquetar manualmente cada elemento relevante. Essa tarefa não é apenas demorada, mas também sujeita a erros humanos, o que muitas vezes exige verificações e correções adicionais.

A fim de encontrar um meio-termo entre a quantidade de dados, a minimização do tempo de anotação e anotações de alta qualidade, neste trabalho propusemos um pipeline que, a partir de um número limitado de documentos PDF anotados com as categorias *texto, título, lista, tabela* e *imagem* recebidas como entrada, é capaz de criar novas layouts de documentos semelhantes com base no número desejado pelo usuário. Este pipeline vai mais longe em preenchendo com o conteúdo as novas layouts criadas, a fim de fornecer imagens de documentos sintéticos e suas respectivas anotações. Com sua estrutura simples, intuitiva e escalável, este pipeline pode contribuir para o active learning, permitindo assim aos modelos de detecção serem treinados continuamente, os tornando mais eficazes e robustos diante de documentos reais.

Em nossas experiências, ao avaliar e comparar três modelos de detecção, observamos que o RT-DETR (Real-Time DEtection TRansformer) obteve os melhores resultados, atingindo uma precisão média (mean Average Precision, mAP) de 96,30%, superando os resultados do Mask R-CNN (Region-based Convolutional Neural Networks) e Mask DINO (Mask DETR with Improved Denoising Anchor Boxes). A superioridade do RT-DETR indica seu potencial para se tornar uma solução de referência na detecção de características em documentos PDF. Esses resultados promissores abrem caminho para aplicações mais eficientes e confiáveis no processamento automático de documentos.

Palavras-chave

Aprendizado de máquina; Aprendizado profundo; Detecção de objetos; Active learning; Geração de dados; Dados sintéticos; Análise do layout do documento; Processamento automático de documentos; Extração de conteúdo.

Abstract

Hodonou,Loïck Geoffrey; Cavalcanti Pacheco, Marco Aurelio (Advisor); Batista, Evelyn (Co-Advisor). **Generation and Detection of Objects in Documents by Deep Learning Neural Network Models (DeepDocGen)**. Rio de Janeiro, 2024. 131p. Dissertação de Mestrado – Departamento de Engenharia Eléctrica, Pontifícia Universidade Católica do Rio de Janeiro.

The effectiveness of human-machine conversation systems, such as chatbots and virtual assistants, is directly related to the amount and quality of knowledge available to them. In the digital age, the diversity and quality of data have increased significantly, being available in various formats. Among these, the PDF (Portable Document Format) stands out as one of the most well-known and widely used, adapting to various sectors, such as business, education, and research. These files contain a considerable amount of structured data, such as text, headings, lists, tables, images, etc.

The content of PDF files can be extracted using dedicated tools, such as OCR (Optical Character Recognition), PdfMiner, Tabula and others, which have proven to be suitable for this task. However, these tools may encounter difficulties when dealing with the complex and varied presentation of PDF documents. The accuracy of extraction can be compromised by the diversity of layouts, non-standardized formats, and embedded graphic elements in the documents, often leading to manual post-processing.

Computer vision, and more specifically, object detection, is a branch of machine learning that aims to locate and classify instances in images using models dedicated to the task. It is proving to be a viable approach to accelerating the work performed by algorithms like OCR, PdfMiner, Tabula and improving their accuracy.

Object detection models, being based on deep learning, require not only a substantial amount of data for training but, above all, high-quality annotations, as they have a direct impact on achieving high levels of accuracy and robustness. The diversity of layouts and graphic elements in PDF documents adds an additional layer of complexity, requiring representatively annotated data so that the models can learn to handle all possible variations.

Considering the voluminous aspect of the data needed for training the models, we quickly realize that the data annotation process becomes a tedious

and time-consuming task requiring human intervention to manually identify and label each relevant element. This task is not only time-consuming but also subject to human error, often requiring additional checks and corrections.

To find a middle ground between the amount of data, minimizing annotation time, and high-quality annotations, in this work, we proposed a pipeline that, from a limited number of annotated PDF documents with the categories text, title, list, table, and image as input, can create new document layouts similar to the desired number by the user. This pipeline goes further by filling the new created layouts with content to provide synthetic document images and their respective annotations. With its simple, intuitive, and scalable structure, this pipeline can contribute to active learning, allowing detection models to be continuously trained, making them more effective and robust in the face of real documents.

In our experiments, when evaluating and comparing three detection models, we observed that the RT-DETR (Real-Time Detection Transformer) achieved the best results, reaching a mean Average Precision (mAP) of 96.30%, surpassing the results of Mask R-CNN (Region-based Convolutional Neural Networks) and Mask DINO (Mask DETR with Improved Denoising Anchor Boxes). The superiority of RT-DETR indicates its potential to become a reference solution in detecting features in PDF documents. These promising results pave the way for more efficient and reliable applications in the automatic processing of documents.

Keywords

Machine learning; Deep learning; Object detection; Active learning; Data generation; Synthetic data; Document layout analysis; Automatic document processing; Content extraction.

Sumário

1 Introdução **19**
1.0.1 Objetivos 20
1.0.2 Conteúdo da dissertação 21

2 Background **22**
2.1 Rede neural artificial 22
2.1.1 Rede neural artificial e transformação funcional 23
2.1.2 Funções de ativação 23
2.1.3 Tipos de funções de ativação 24
2.1.4 Aprendizagem de uma rede neural 25
2.2 Rede neural Convolucional 29
2.2.1 Arquitetura de uma Rede Neural Convolucional 29
2.2.2 Camada de Convolução 30
2.2.3 Camada de Pooling 31
2.2.4 Camada Totalmente Conectada 32
2.2.5 ResNet: Redes Neurais Residuais 32
2.3 Transformers 33
2.3.1 Arquitetura do modelo Transformer 34
2.4 Detecção de objetos 40
2.4.1 Detecção de objetos em documentos digitais 40
2.5 Mask R-CNN 41
2.5.1 Rede de Proposição de Regiões (RPN) 42
2.5.2 Alinhamento da Região de Interesse (RoIAlign) 45
2.5.3 Geração de Máscara 46
2.6 Mask DINO 46
2.7 RT-DETR 47
2.7.1 Encoder híbrido 48
2.7.2 Mecanismo de seleção de consultas de incerteza mínima 49
2.8 K-means 49
2.9 t-SNE 50
2.9.1 Perplexidade 51

3 Metodologia **52**
3.1 Metodologia de criação de conjunto de dados 52
3.1.1 Pipeline de anotação e processamento de PDFs 52
3.1.2 Pipeline de geração e processamento de layout de documento 60
3.2 Metodologia de treinamento, validação, teste dos modelos e seleção de modelo 74
3.2.1 Separação em dados de treinamento, validação e teste 75
3.2.2 Processamento de treinamento e validação dos modelos 77
3.2.3 Processamento de teste dos modelos 77
3.2.4 Seleção de Modelo 78

4 Resultados **81**

4.1 Pipeline de Anotação e Processamento de PDFs 81
4.1.1 Agrupamento com o Algoritmo K-means 81
4.1.2 Redução da dimensionalidade 82
4.1.3 Seleção Manual de PDFs por grupo 82
4.1.4 Anotação de PDFs selectionados com o software VoTT (Visual Object Tagging Tool) 83
4.2 Pipeline de geração e processamento de layout de document 84
4.2.1 Gerar layout de documento usando o Gerador de Layouts Sintéticos 85
4.2.2 Pós-Processamento para remoção de sobreposições 87
4.2.3 Preencher conteúdo usando o Gerador de Imagens de Documentos Sintéticos 88
4.3 Separação em dados de treinamento, validação e teste 90
4.4 Processamento de treinamento e validação dos modelos 91
4.4.1 Separação Totalmente Sintética 92
4.4.2 Teste com Dados Reais Parcial 94
4.4.3 Combinação Híbrida 97
4.5 Processamento de teste dos modelos 100
4.6 Seleção de Modelo 101

5 Discussão 102
5.1 Comparação do pipeline com outros trabalhos 102
5.1.1 Vantagens do nosso pipeline 104
5.1.2 Limitações do pipeline 105
5.2 Interpretação dos resultados 105
5.2.1 Análise focada no conjunto de dados 106
5.2.2 Análise focada nos modelos 107
5.3 Trabalhos futuros 108

6 Conclusão 110

7 Apêndice 112

8 Referências bibliográficas 124

Lista de figuras

Figura 2.1 Ilustração da arquitetura de uma rede neural composta por três camadas: uma camada de entrada com m entradas, uma camada oculta formada por três subcamadas contendo, respectivamente, 7, 7 e 4 neurônios, e uma camada de saída com 1 neurônio, que gera a saída $y_{predict}$. 23
Figura 2.2 Ilustração do cálculo realizado em um neurônio. 24
Figura 2.3 Ilustração do funcionamento da função softmax. 25
Figura 2.4 Ilustração do mecanismo de atualização dos pesos usando o algoritmo de descida de gradiente. 28
Figura 2.5 Ilustração dos processos de propagação para frente, cálculo do erro e retropropagação em uma rede neural. 28
Figura 2.6 Ilustração da arquitetura de uma rede neural convolucional. 29
Figura 2.7 Ilustração do kernel K convoluindo sobre a imagem I. 31
Figura 2.8 Ilustração da operação de max pooling. 32
Figura 2.9 Ilustração de um bloco residual em uma ResNet [38]. 33
Figura 2.10 Ilustração da arquitetura do modelo Transformer [39]. 34
Figura 2.11 Ilustração do mecanismos de autoatenção e de autoatenção multi-cabeça [39]. 37
Figura 2.12 Ilustração de detecção de objetos em documentos digitais [61]. 41
Figura 2.13 Detecção de objetos, segmentação semântica e segmentação de instâncias. 42
Figura 2.14 Arquitetura do modelo Mask R-CNN [60] 42
Figura 2.15 Ilustração de 9 caixas de ancoragem de diferentes tamanhos e proporções geradas em um ponto do mapa de características. 43
Figura 2.16 Ilustração do mecanismo da rede de propostas de região (RPN) [50]. 44
Figura 2.17 Ilustração do mecanismo do NMS na eliminação das regiões propostas redundantes. 45
Figura 2.18 Ilustração do mecanismo de RoIAlign [60]. 46
Figura 2.19 Arquitetura do modelo Mask DINO [**?**]. 47
Figura 2.20 Arquitetura do modelo RT-DETR [51]. 48

Figura 3.1 Imagem ilustrativa mostrando o nosso pipeline para criar conjuntos de dados de documentos sintéticos. 52
Figura 3.2 Imagem ilustrativa mostrando o processo de extração dos embeddings com a rede ResNet-152. A dimensão de cada embedding extraído é de (1 x 2048 ou seja cada imagem dos 12.995 tem 1 x 2048 como dimensão), o que corresponde aos resultados esperados da camada de agrupamento médio (average pooling) da rede ResNet-152. 55
Figura 3.3 Imagem ilustrativa mostrando o fluxo de criação do arquivo JSON no software VoTT. 59
Figura 3.4 Imagem ilustrativa mostrando o esquema de nosso código para pós-processamento do arquivo JSON (saída do software VoTT) no formato PubLayNet. 60
Figura 3.5 Distribuição dos rótulos nos conjuntos de treinamento e validação do conjunto de dados PubLayNet. 62

Figura 3.6 Imagem ilustrativa mostrando dois layouts de documentos gerados: um com sobreposição de caixas e outro sem sobreposição. 63
Figura 3.7 Imagem ilustrativa mostrando a exclusão da caixa delimitadora: Retângulo roxo nesse caso. 64
Figura 3.8 Imagem ilustrativa mostrando b_1 e b_2 no espaço 2D. Vale lembrar que, no caso desta Figura, o termo 'b_1 ou b_2' é utilizado apenas para indicar que b_1 ou b_2 sempre terão o mesmo posicionamento das coordenadas num espaço 2D. Além disso, quando se trata de imagens, as coordenadas começam da esquerda para a direita e de cima para baixo. 65
Figura 3.9 Imagem ilustrativa mostrando a caixa delimitadora b_1 sobrepondo a caixa delimitadora b_2 no lado superior esquerdo. 65
Figura 3.10 Imagem ilustrativa mostrando a subtração na abscissa da área da superfície de interseção $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$ da caixa delimitadora $\mathcal{A}_{b_1}$. 67
Figura 3.11 Imagem ilustrativa mostrando a remoção da distância Δ_x. Isso equivale a retirar a área da superfície de interseção $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$ de uma distância $\mathcal{W}_{\mathcal{I}} + \Delta_x$ da área do retângulo $\mathcal{A}_{b_1}$. 67
Figura 3.12 Imagem ilustrativa mostrando a caixa delimitadora b_1 completamente sobrepondo b_2. 68
Figura 3.13 Imagem ilustrativa mostrando o esquema do nosso código de pós-processamento para remover a sobreposição. 69
Figura 3.14 Imagem ilustrativa mostrando o esquema do Gerador de Imagens de Documentos Sintéticos. 71
Figura 3.15 Imagem illustrativa mostrando a esquema do nosso código de pós-processamento para o formato COCO. 73
Figura 3.16 Imagem ilustrativa mostrando o nosso pipeline de treinamento, validação, teste dos modelos e seleção de modelo. 75

Figura 4.1 Imagem ilustrativa mostrando o resultado da escolha do número de grupos K para o algoritmo K-means pelo método do cotovelo. 82
Figura 4.2 Imagem ilustrativa mostrando a distribuição das páginas dos nossos documentos PDF após o agrupamento pelo algoritmo e uma redução de dimensionalidade com o algoritmo t-SNE. As cores representam os grupos. 82
Figura 4.3 Interface de anotação do software VoTT. À esquerda, na barra lateral, estão as imagens que serão anotadas. No centro, a imagem sendo anotada. À direita, na barra lateral, estão os rótulos utilizados: texto, título, lista, tabela e imagem. 84
Figura 4.4 Imagem ilustrativa mostrando a curva de perda de treinamento do Gérador de Layout sintéticos. 85
Figura 4.5 Imagem ilustrativa mostrando a curva de perda de validação do Gérador de Layout sintéticos. O menor erro de validação ocorreu na iteração 83.920, ou seja, na época 16. 85
Figura 4.6 Imagem ilustrativa mostrando alguns exemplos de layouts de documentos gerados pelo Gerador de Layouts Sintéticos. 86
Figura 4.7 Exemplo de correção de sobreposição pelo estagio de pós-processamento para remoção de sobreposições 87
Figura 4.8 Exemplo de imagem de documento gerada pelo Gerador de Imagens de Documentos Sintéticos. 89

Figura 4.9 Anotação em formato XML correspondente à imagem de documento na Figura 4.8, gerada pelo Gerador de Imagens de Documentos Sintéticos. 90
Figura 4.10 Distribuição dos rótulos nos conjuntos de dados de treinamento, validação e teste em cada experimento. O símbolo "$*$" na figura se refere ao envolvimento dos dados reais. 91
Figura 4.11 Curva de perda durante o treinamento do modelo Mask R-CNN no experimento de Separação Totalmente Sintética. 92
Figura 4.12 Curva de perda durante a validação do modelo Mask R-CNN no experimento de Separação Totalmente Sintética. 92
Figura 4.13 Curva de perda durante o treinamento do modelo Mask DINO no experimento de Separação Totalmente Sintética. 93
Figura 4.14 Curva de perda durante a validação do modelo Mask DINO no experimento de Separação Totalmente Sintética. 93
Figura 4.15 Curva de perda durante o treinamento do modelo RT-DETR no experimento de Separação Totalmente Sintética. 94
Figura 4.16 Curva de perda durante a validação do modelo Mask R-CNN no experimento de Separação Totalmente Sintética. 94
Figura 4.17 Curva de perda durante o treinamento do modelo Mask R-CNN no experimento de Teste com Dados Reais Parcial. 95
Figura 4.18 Curva de perda durante a validação do modelo Mask R-CNN no experimento de Teste com Dados Reais Parcial. 95
Figura 4.19 Curva de perda durante o treinamento do modelo Mask DINO no experimento de Teste com Dados Reais Parcial. 95
Figura 4.20 Curva de perda durante a validação do modelo Mask DINO no experimento de Teste com Dados Reais Parcial. 96
Figura 4.21 Curva de perda durante o treinamento do modelo RT-DETR no experimento de Teste com Dados Reais Parcial. 96
Figura 4.22 Curva de perda durante a validação do modelo RT-DETR no Experimento de Teste com Dados Reais Parcial. O menor erro de validação foi registrado na época 99. 97
Figura 4.23 Curva de perda durante o treinamento do modelo Mask R-CNN no experimento de Combinação Híbrida. 98
Figura 4.24 Curva de perda durante a validação do modelo Mask R-CNN no experimento de Combinação Híbrida. 98
Figura 4.25 Curva de perda durante o treinamento do modelo Mask DINO no experimento de Combinação Híbrida. 98
Figura 4.26 Curva de perda durante a validação do modelo Mask DINO no experimento de Combinação Híbrida. 99
Figura 4.27 Curva de perda durante o treinamento do modelo RT-DETR no experimento de Combinação Híbrida. 99
Figura 4.28 Curva de validação do modelo RT-DETR no experimento de Combinação Híbrida. O menor erro de validação ocorreu na época 99. 100

Figura 7.1 Exemplo de exclusão manual de layout de doucmento com estrutura inconsistente 1. 112
Figura 7.2 Exemplo de exclusão manual de layout de doucmento com estrutura inconsistente 2. 113

Figura 7.3 Exemplo de exclusão manual de layout de doucmento com estrutura inconsistente 3. 114
Figura 7.4 Exemplo de exclusão manual de layout de doucmento com estrutura inconsistente 4. 115
Figura 7.5 Exemplo de correção de sobreposição pelo estagio de pós-processamento para remoção de sobreposições 116
Figura 7.6 Conjunto de oito imagens de documentos gerados pelo Gerador de Imagens de Documentos Sintéticos. 117
Figura 7.7 Exemplo de visualização das Delimitações em Documentos Sintéticos Usando Detectron2 (1). 118
Figura 7.8 Exemplo de visualização das delimitações em documentos dintéticos usando Detectron2 (2). 119
Figura 7.9 Exemplo de visualização das delimitações em documentos dintéticos usando Detectron2 (3). 120
Figura 7.10 Exemplo de visualização das delimitações em documentos dintéticos usando Detectron2 (4). 121

Lista de tabelas

Tabela 3.1 Estatísticas dos 1.980 documentos PDF convertidos em imagens. No caso da linha 5, uma imagem representa uma (1) página do documento PDF se este tiver uma única página, ou várias páginas se o documento PDF tratado tiver várias. 54
Tabela 3.2 Valores dos parâmetros usados em nosso experimento com o algoritmo K-means. Vale lembrar que usamos o algoritmo K-means da biblioteca Sklearn assim que o visualizador KElbowVisusalizer da biblioteca Yellowbrick, portanto, alguns valores de parâmetros são mantidos por padrão. 56
Tabela 3.3 Valores dos diferentes parâmetros utilizados na nossa experiência com o algoritmo t-SNE. Vale lembrar que usamos o t-SNE da biblioteca Sklearn, portanto, alguns valores de parâmetros são mantidos por padrão. 57
Tabela 3.4 Hiperparâmetro utilizado durante o treinamento do Gerador de Layouts Sintéticos 62
Tabela 3.5 Métricas padrão do COCO para detecção de objetos 80

Tabela 4.1 Distribuição dos documentos por grupo 83
Tabela 4.2 Distribuição dos documentos PDF representativos selecionados manualmente por grupo. 83
Tabela 4.3 Número de imagens anotadas por grupo utilizando o software VoTT. 84
Tabela 4.4 Número de layouts de documentos gerados por grupo. Para o grupo 4, nenhum layout foi gerado, pois não houve anotações disponíveis para este grupo. 86
Tabela 4.5 Número de layouts de documentos por grupo, resultante da remoção de sobreposições. 88
Tabela 4.6 Distribuição dos dados de treinamento, validação e teste utilizados nas experiências. Cabe ressaltar que o símbolo " $*$ " presente na tabela indica o envolvimento de dados reais. 91
Tabela 4.7 Resultados dos modelos em cada experimento nos dados de teste envolvidos. 100

Tabela 7.1 Hiperparâmetros ulizados pelo modelo Mask R-CNN. 122
Tabela 7.2 Hiperparâmetros ulizados pelo modelo Mask DINO. 122
Tabela 7.3 Hiperparâmetros ulizados pelo modelo RT-DETR. 123

Lista de Abreviaturas

AIFI – Attention-based Intra-scale Feature Interaction
BCE – Binary Cross-Entropy
CCFF – CNN-based Cross-scale Feature Fusion
CNN – Convolutional Neural Network
COCO – Common Objects in Context
DETR – DEtection TRansformer
DINO – DETR with Improved deNoising anchOr boxes
FFN – Feed Forward Network
GPU – Graphics Processing Unit
IA – Inteligência Artificial
IoU – Interseção sobre União
JPEG – Joint Photographic Experts Group
JSON – JavaScript Object Notation
MSE – Mean Squared Error
mAP – mean Average Precision
NLP – Processamento de Linguagem Natural
NMS – Non-Maximum Suppression
OCR – Optical Character Recognition
PDF – Portable Document Format
R-CNN – Region-based Convolutional Neural Network
ROI – Region Of Interest
RPN – Regional Proposal Network
RT-DETR – Real-Time DEtection TRansformer
t-SNE – t-distributed Stochastic Neighbor Embedding
ViT – Vision Transformer
VoTT –Visual object Tagging Tool
XML – Extensible Markup Language

Loïck Geoffrey Hodonou, *Everything happens in the mind. Either you work hard without cheating, or you don't work hard.*

1
Introdução

Os avanços da inteligência artificial (IA) e do aprendizado de máquina têm criado inúmeras oportunidades para aprimorar a automação das tarefas cotidianas. Dentre esses avanços, o Processamento de Linguagem Natural (NLP) transformou significativamente a maneira como interagimos com as máquinas, favorecendo o surgimento de sistemas de conversação homem-máquina, como chatbots e assistentes virtuais.

Para que esses sistemas alcancem um desempenho ideal e atendam às necessidades dos usuários, dois elementos são cruciais: uma compreensão aprofundada do domínio de aplicação e a qualidade dos dados utilizados para treinar os modelos. Na era digital, uma ampla variedade de dados está disponível, incluindo dados estruturados (bancos de dados SQL, arquivos CSV, planilhas Excel), dados não estruturados (e-mails, documentos Word, documentos PDF, imagens JPEG, vídeos MP4), etc.

Entre esses dados, os documentos PDF (Portable Document Format) ocupam uma posição de destaque devido à sua utilização generalizada em diversos setores, sejam eles profissionais, educacionais, comerciais ou outros. Esse formato se destaca por sua flexibilidade de layout, que pode variar conforme o setor, e pela riqueza das informações que pode conter. Por exemplo, a estrutura dos documentos PDF de uma empresa como a Petrobras difere significativamente da estrutura dos documentos utilizados no meio educacional, refletindo a disposição de elementos estruturados como textos, títulos, listas, tabelas, imagens, gráficos, equações, etc.

Para extrair informações desses documentos, é comum utilizar ferramentas especializadas, como OCR[1] (Reconhecimento Óptico de Caracteres), PdfMiner[2], Tabula[3], etc. No entanto, embora essas ferramentas sejam eficazes, elas enfrentam dificuldades diante de layouts mais complexos, o que pode resultar em perdas de dados e exigir um pós-processamento manual.

A visão computacional, e em particular a detecção de objetos, é uma subdisciplina do aprendizado de máquina baseada em aprendizado profundo. Para que essa detecção seja eficaz, é essencial dispor não apenas de uma grande quantidade de dados, mas também de anotações detalhadas desses dados, para que os modelos possam aprender a identificar com precisão os objetos de interesse. Aplicada a documentos PDF, essa abordagem permite identificar

[1]https://github.com/tesseract-ocr/tesseract
[2]https://github.com/pdfminer/pdfminer.six/tree/master?tab=readme-ov-file
[3]https://github.com/tabulapdf/tabula

áreas de interesse e melhorar a precisão da extração de informações com o auxílio das ferramentas mencionadas anteriormente.

Pesquisas recentes exploraram a aplicação da detecção de objetos em documentos PDF de duas maneiras principais. Por um lado, alguns trabalhos propõem novos modelos de detecção [1, 2, 3, 4] ou utilizam modelos existentes, baseando-se em conjuntos de dados públicos como PubLayNet [5], DocBank [6], DocLayNet [7], ICDAR-13 [8], ICDAR-17 POD [9], etc. Esses grandes conjuntos de dados anotados fornecem uma base sólida para treinar modelos capazes de identificar com precisão os elementos estruturais dos documentos, como títulos, parágrafos, tabelas, figuras, equações, etc.

Por outro lado, outras pesquisas se concentram na criação de conjuntos de dados *sintéticos* por meio de modelos generativos [10, 11, 12, 13, 14], que têm ganhado interesse recentemente na comunidade de visão computacional. Aproveitando essa capacidade de geração, trabalhos como os de [15, 16, 17] criam conjuntos de dados gerados artificialmente, não apenas a partir da criação de layouts específicos de documentos, mas também preenchendo esses layouts com conteúdo realista. Ou seja, além de criar a estrutura do documento, esses modelos também geram o conteúdo dentro de elementos como textos, tabelas e imagens, tornando os documentos sintéticos completos. Esses documentos gerados são então automaticamente anotados para treinar modelos de detecção de objetos. Essa abordagem permite produzir conjuntos de dados adaptados a necessidades específicas, especialmente em contextos onde os dados reais anotados são limitados ou inexistentes.

1.0.1 Objetivos

Neste estudo, foi adotada a abordagem de detecção de objetos aplicada especificamente aos documentos da Petrobras, como, por exemplo, relatórios. No entanto, essa aplicação encontrou vários desafios, dentre eles:

1. A quantidade limitada de documentos PDF disponíveis, o que dificulta o treinamento dos modelos de detecção de objetos baseados em aprendizado profundo.

2. A impossibilidade de utilizar conjuntos de dados volumosos, como PubLayNet, DocBank, DoclayNet, ICDAR-13, ICDAR-17 POD, devido ao fato de que os layouts desses datasets diferem significativamente dos layouts dos documentos da Petrobras.

Para superar esses obstáculos, foi desenvolvido um pipeline capaz de gerar layouts de documentos semelhantes aos originais a partir de anotações manu-

ais realizadas em uma pequena quantidade de documentos PDF da Petrobras. Esse pipeline não apenas cria novos layouts, mas também os preenche automaticamente com conteúdo estruturado (textos, títulos, listas, tabelas, imagens), gerando assim imagens de documentos sintéticos.

As anotações são essenciais para o treinamento dos modelos de detecção de objetos, mas o processo manual é frequentemente longo, custoso em termos de mão de obra e sujeito a erros, especialmente quando o número de imagens a serem anotadas é elevado. Para remediar isso, o pipeline criado automatiza a geração das anotações correspondentes às imagens de documentos criadas, reduzindo assim os riscos de erro e o tempo de processamento.

Para ser pronto ao uso pelos modelos de detecção de objetos, o pipeline formata esses dados de forma a torná-los diretamente utilizáveis . Esses conjuntos de dados sintéticos incluem imagens de documentos e suas respectivas anotações.

Esse pipeline não se limita aos documentos da Petrobras; ele pode ser adaptado a diversos tipos de documentos. Além disso, facilita o processo de aprendizado ativo, permitindo que os modelos de detecção sejam treinados continuamente, melhorando assim sua eficácia ao longo do tempo.

1.0.2 Conteúdo da dissertação

No Capítulo 2, é apresentado o contexto teórico necessário para entender o conteúdo da dissertação. O Capítulo 3 descreve a metodologia utilizada para a criação do nosso pipeline, seguido pela apresentação dos resultados no Capítulo 4. Por fim, as conclusões são apresentadas no Capítulo 5, onde também são sugeridas direções para trabalhos futuros.

2
Background

Este capítulo começa explicando algumas das teorias básicas necessárias para entender o conteúdo da dissertação, incluindo conceitos fundamentais sobre aprendizagem automática e aprendizagem profunda.

Assim como os seres humanos se adaptam a novos eventos com base em experiências passadas, a aprendizagem automática e profunda seguem um princípio semelhante. Por exemplo, estudantes que recebem aulas e explicações de professores conseguem, posteriormente, realizar testes e exames com sucesso. Da mesma forma, essas tecnologias utilizam dados históricos, conhecidos como *dados de treinamento*, para desenvolver algoritmos que preveem eventos futuros a partir de novos dados, denominados *dados de teste*. Essa capacidade de prever eventos futuros é possibilitada pela aprendizagem profunda, que se baseia no uso de redes neurais artificiais para capturar padrões complexos nos dados [18].

2.1
Rede neural artificial

Uma rede neural artificial é um modelo de aprendizado de máquina projetado para tomar decisões de maneira semelhante ao cérebro humano. Ela imita o funcionamento dos neurônios biológicos, permitindo detectar padrões, avaliar opções e chegar a conclusões [19].

Uma rede neural artificial é composta por várias unidades de processamento ou unidades de cálculo chamadas *neurônios*. Esses neurônios estão conectados entre si através de *sinapses*, e as conexões são ponderadas por *pesos*, que determinam a importância relativa de cada entrada [20]. A Figura 2.1 mostra a arquitetura de uma rede neural artificial.

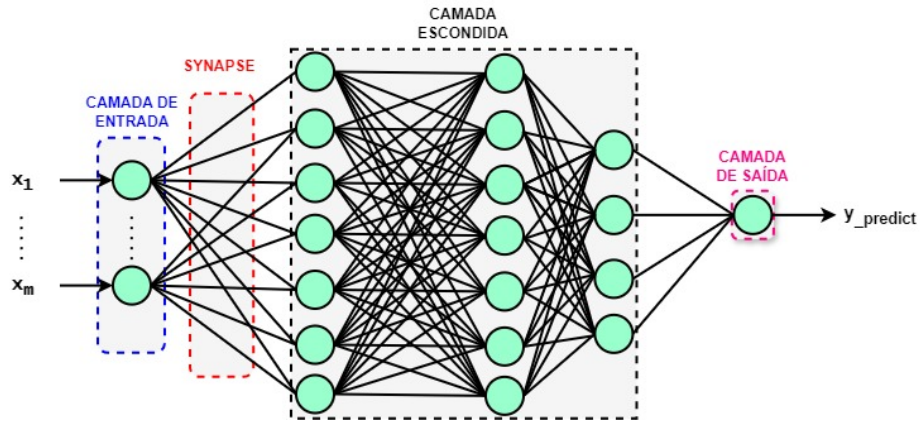

Figura 2.1: Ilustração da arquitetura de uma rede neural composta por três camadas: uma camada de entrada com m entradas, uma camada oculta formada por três subcamadas contendo, respectivamente, 7, 7 e 4 neurônios, e uma camada de saída com 1 neurônio, que gera a saída $y_{predict}$.

2.1.1 Rede neural artificial e transformação funcional

Matematicamente, uma rede neural artificial pode ser descrita como uma série de transformações funcionais. Para cada neurônio na primeira camada, construímos uma combinação linear das variáveis de entrada $x_1, ..., x_m$, como segue:

$$a_j^{(1)} = \sum_{i=1}^{m} \left(w_{ij}^{(1)} x_i \right) + b_j^{(1)}, \tag{2-1}$$

onde $j = 1, \ldots, M$, e o índice (1) indica que os parâmetros correspondentes estão na primeira camada da rede. $w_{ij}^{(1)}$ e $b_j^{(1)}$ representam, respectivamente, o peso da conexão entre a entrada x_i e o neurônio j, e o viés associado a esse neurônio [21]. O termo a_j representa a *função de agregação* do neurônio j, que é simplesmente uma função linear.

2.1.2 Funções de ativação

Posteriormente, a função de agregação a_j é passada por uma função *diferenciável não linear* conhecida como *função de ativação*, denotada por f, para produzir a saída do neurônio j.

$$y_j = f(a_j). \tag{2-2}$$

Essa prática tem como objetivo introduzir não-linearidade na rede neural, permitindo que ela aprenda relações complexas entre as entradas e as saídas, e, especialmente, decida se um neurônio deve ser ativado ou não. Quando um neurônio é ativado, isso significa que sua entrada é considerada importante

para a previsão da saída. A Figura 2.2 mostra o cálculo realizado dentro do neurônio.

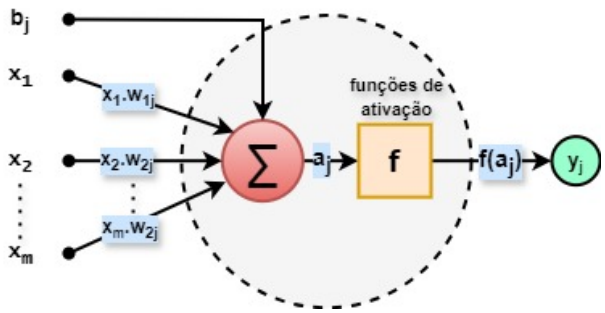

Figura 2.2: Ilustração do cálculo realizado em um neurônio.

2.1.3 Tipos de funções de ativação

Existem diversas funções de ativação utilizadas em redes neurais, cada uma com características e aplicações específicas:

1. **Função Sigmoide**: Esta função transforma entradas do domínio dos números reais $\mathbb{R}$ em saídas que pertencem ao intervalo $[0, 1]$. Ela é especialmente útil em problemas onde a saída precisa ser interpretada como uma probabilidade, como em modelos de classificação binária [22]. Sua expressão matemática é:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2-3}$$

2. **Função ReLU (Rectified Linear Unit)**: A função ReLU é amplamente utilizada em redes profundas devido à sua simplicidade e eficácia. Ela retém apenas os valores positivos e atribui o valor zero para todas as entradas negativas, promovendo uma ativação esparsa na rede [22]. Sua expressão é:

$$\text{ReLU}(x) = \max(x, 0). \tag{2-4}$$

3. **Função Tanh (Tangente Hiperbólica)**: Semelhante à função sigmoide, a função Tanh mapeia as entradas para o intervalo $[-1, 1]$. Ela é frequentemente utilizada em situações onde se deseja uma saída centrada em torno de zero [22]. Sua expressão matemática é:

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2-5}$$

4. **Função Softmax**: Utilizada em problemas de classificação múltipla, a função softmax converte um vetor de K valores reais em uma distribuição

de probabilidade sobre K classes, garantindo que cada probabilidade seja positiva e que a soma de todas as probabilidades seja igual a 1. Isso é especialmente útil para determinar a classe mais provável em um conjunto de possíveis categorias [22]. A expressão matemática é:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2-6}$$

onde z_j é um vetor de valores reais $(z_1, \ldots, z_K)$ para $j = 1, \ldots, K$, e $\sigma(z)_j$ representa o vetor resultante de probabilidades. A Figura 2.3 ilustra o funcionamento da função softmax.

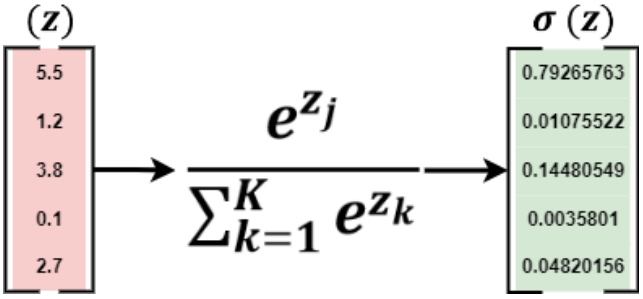

Figura 2.3: Ilustração do funcionamento da função softmax.

2.1.4 Aprendizagem de uma rede neural

Existem quatro tipos mais comuns de aprendizagem em aprendizado de máquina :

- **Aprendizado supervisionado**: Neste tipo de aprendizado, o modelo é treinado com base em pares de dados rotulados (entrada, alvo). O objetivo é ensinar o modelo a reconhecer padrões ou esquemas nos dados, permitindo que ele faça previsões precisas sobre novos conjuntos de dados [23].
- **Aprendizado não supervisionado**: Ao contrário do aprendizado supervisionado, este método lida com dados não rotulados, ou seja, sem respostas predefinidas. O aprendizado não supervisionado busca identificar estruturas ou padrões intrínsecos nos próprios dados, descobrindo agrupamentos naturais ou reduzindo a dimensionalidade dos dados [23].
- **Aprendizado semi-supervisionado**: Este tipo de aprendizado combina uma pequena quantidade de dados rotulados com uma grande quantidade de dados não rotulados. O objetivo é melhorar o desempenho do

modelo e sua capacidade de generalização, explorando a estrutura subjacente dos dados [24]. Ele se posiciona entre o aprendizado supervisionado e o não supervisionado.

– **Aprendizado por reforço**: Nesse caso, o modelo é projetado para tomar decisões e interagir com um ambiente, aprendendo a partir das consequências de suas ações. O modelo recebe recompensas por ações corretas e punições por ações incorretas. Através desse processo contínuo de tentativa e erro, o modelo ajusta seu comportamento para maximizar as recompensas ao longo do tempo [23].

O processo de aprendizado de uma rede neural pode ser dividido em três etapas principais: *propagação para frente*, *cálculo do erro* e *retropropagação*.

2.1.4.1 Propagação para frente

A propagação para frente (*forward propagation*) é o processo pelo qual os dados de entrada são transmitidos através das camadas de uma rede neural para gerar a saída final. Matematicamente, esse processo pode ser expresso da seguinte forma:

– Para a primeira camada escondida:

$$\mathbf{h}^{(1)} = f^{(1)}(\mathbf{W}^{(1)T}\mathbf{x} + \mathbf{b}^{(1)}). \tag{2-7}$$

– Para a segunda camada escondida:

$$\mathbf{h}^{(2)} = f^{(2)}(\mathbf{W}^{(2)T}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}). \tag{2-8}$$

De maneira geral, para a l-ésima camada (com l variando de 1 a L):

$$\mathbf{h}^{(l)} = f^{(l)}(\mathbf{W}^{(l)T}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \tag{2-9}$$

onde $\mathbf{x}$ é o vetor de entrada, $\mathbf{W}^{(l)}$ é a matriz de pesos da l-ésima camada, $\mathbf{b}^{(l)}$ é o vetor de vieses dessa camada, $f^{(l)}$ é a função de ativação aplicada na camada l, e $\mathbf{h}^{(l)}$ é o vetor de ativações resultante.

Esse processo é repetido para todas as camadas da rede até que a saída final seja gerada na última camada [25].

2.1.4.2 Cálculo do erro

Após a propagação para frente, a saída gerada pela rede neural é comparada com o valor alvo real, especialmente no contexto do aprendizado supervisionado. Essa comparação é feita por meio de uma *função de custo* ou *função de perda*, que mede a diferença entre a saída prevista (y_{predict}) e a saída real (y_{true}). A escolha da função de custo depende do tipo de problema:

- **Regressão**: O erro quadrático médio (*Mean Squared Error*, MSE) é comumente utilizado para problemas de regressão, onde a saída é um valor contínuo [26].

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{true},i} - y_{\text{predict},i})^2. \tag{2-10}$$

- **Classificação Binária**: A entropia cruzada binária (*Binary Cross-Entropy Loss*, BCE) é frequentemente empregada em problemas de classificação binária, onde a saída é uma probabilidade [26].

$$\text{BCE} = -[y_{\text{true,i}}\log(y_{\text{predict,i}}) + (1 - y_{true,i})\log(1 - y_{predict,i})]. \tag{2-11}$$

- **Classificação Múltipla** A entropia cruzada (*Cross-Entropy Loss*) é utilizada em problemas de classificação múltipla, onde as saídas são distribuídas entre várias classes [26].

$$\text{Cross-Entropy Loss} = -\sum_{i=1}^{n} y_{\text{true},i}\log(y_{\text{predict},i}). \tag{2-12}$$

2.1.4.3 Retropropagação

O objetivo do treinamento de uma rede neural é minimizar a função de custo, também conhecida como função de perda. Após a propagação para frente e o cálculo do erro, o próximo passo é a retropropagação (*backpropagation*), onde o erro calculado é propagado de volta pela rede, permitindo o ajuste dos parâmetros (pesos e vieses). Esse processo utiliza um otimizador, como o algoritmo de descida de gradiente [27], para atualizar os pesos, seguindo a fórmula:

$$\theta = \theta - \eta\nabla J(\theta), \tag{2-13}$$

onde θ representa o parâmetro a ser atualizado, η é a taxa de aprendizado, que controla o tamanho do passo de atualização, e $\nabla J(\theta)$ é o gradiente da função

de custo em relação ao parâmetro θ, indicando a direção e magnitude do ajuste necessário.

Existem vários tipos de otimizadores além da descida de gradiente, como Adam [28], RMSprop [29], e Adagrad [30], cada um com suas próprias características e vantagens, que podem ser utilizados para melhorar a eficiência do treinamento e a convergência da rede neural.

A Figura 2.4 ilustra o mecanismo de atualização dos pesos utilizando o algoritmo de descida de gradiente.

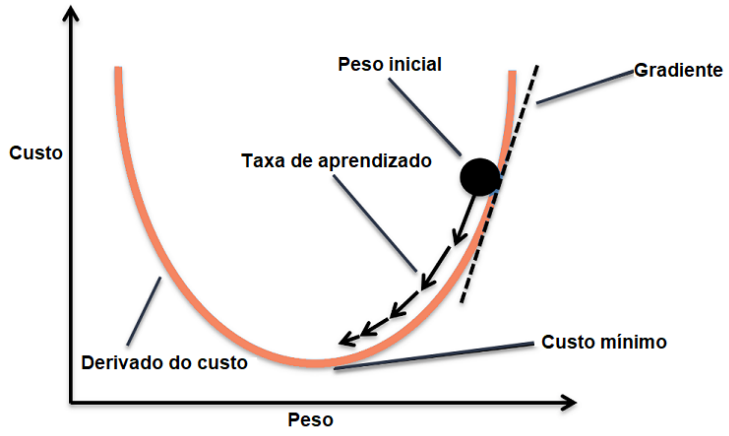

Figura 2.4: Ilustração do mecanismo de atualização dos pesos usando o algoritmo de descida de gradiente.

Por fim, a Figura 2.5 resume os processos de propagação para frente, cálculo do erro, e retropropagação em uma rede neural.

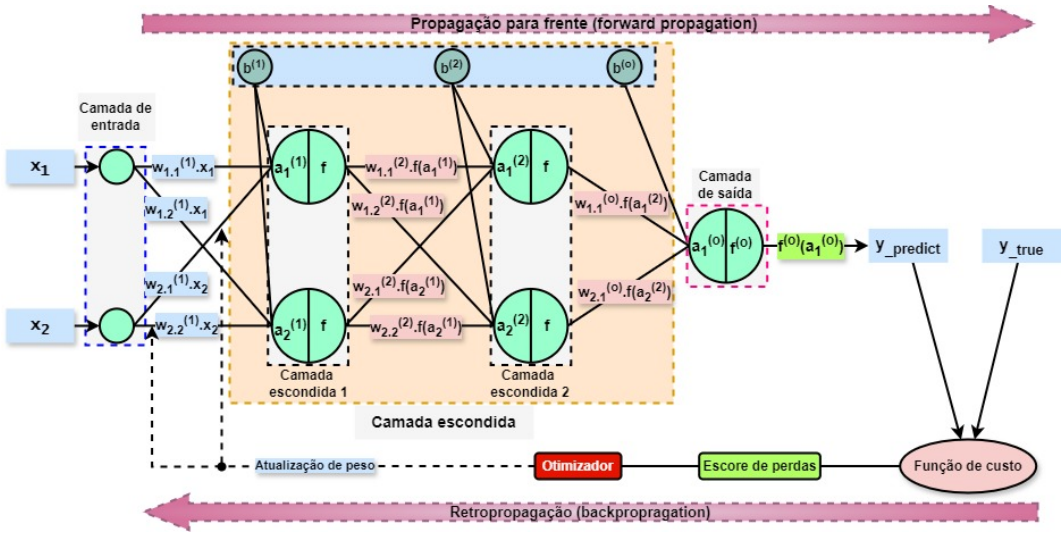

Figura 2.5: Ilustração dos processos de propagação para frente, cálculo do erro e retropropagação em uma rede neural.

2.2 Rede neural Convolucional

As redes neurais tradicionais, embora amplamente utilizadas e com bons resultados, podem enfrentar limitações no processamento de imagens digitais devido ao crescimento exponencial do número de parâmetros, aumentando a complexidade computacional.

Uma imagem digital pode ser representada como uma matriz tridimensional $H \times W \times C$, onde H é a altura, W a largura, e C o número de canais de cor. Por exemplo, para uma imagem de 300x300x3, um único neurônio exigiria 270.000 pesos, e para 5 neurônios (só numa camada), isso representaria 1.350.000 parâmetros a serem treinados, tornando difícil o aprendizado automático das características hierárquicas e a preservação das informações espaciais [31].

As redes neurais convolucionais (Convolutional Neural Networks, CNN) foram introduzidas para resolver esses problemas. Especializadas no processamento de dados estruturados em grade, como imagens, as CNNs reduzem o número de parâmetros ao mesmo tempo em que preservam as informações espaciais cruciais. Isso permite uma extração precisa das características e oferece um desempenho superior em tarefas como reconhecimento de objetos [32] e detecção de padrões [33].

2.2.1 Arquitetura de uma Rede Neural Convolucional

Uma CNN é composta principalmente por três tipos de camadas:

1. Camada de convolução;
2. Camada de agrupamento (pooling) e
3. Camada totalmente conectada.

A Figura 2.6 mostra a arquitetura de uma rede neural convolucional.

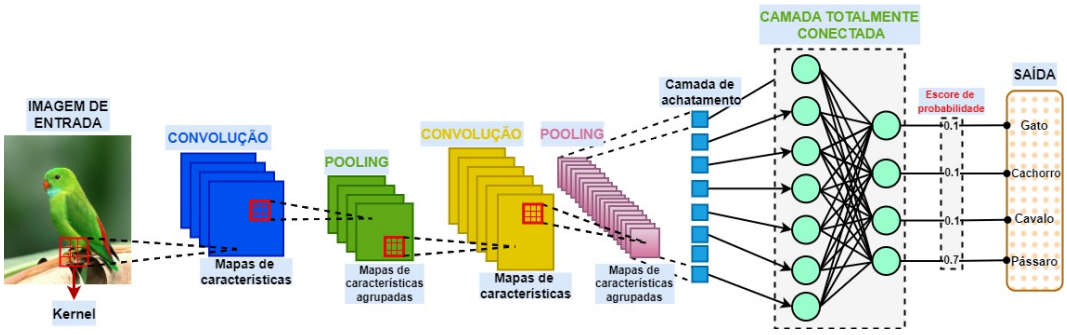

Figura 2.6: Ilustração da arquitetura de uma rede neural convolucional.

Ao receber uma imagem I como entrada, a camada de convolução aplica filtros para extrair características importantes. Cada filtro produz um mapa de características que destaca certas informações na imagem, como bordas, texturas ou outros padrões. Em seguida, a camada de pooling é utilizada para reduzir a dimensão espacial dos mapas de características, preservando as informações mais significativas. Após várias fases de convolução e pooling, as características são achatadas e transmitidas para a camada totalmente conectada, que gera finalmente uma etiqueta de classe prevista ou uma pontuação de probabilidade para cada classe, dependendo da tarefa.

2.2.2 Camada de Convolução

A camada de convolução é o bloco básico das redes neurais convolucionais. Ela gera uma nova representação da imagem de entrada ou das características intermediárias por meio da aplicação de um filtro chamado *Kernel*.

Uma imagem digital I pode ser representada por $I \in \mathbb{R}^{3\times H\times W}$, onde 3 indica o número de canais de cor, H a altura e W a largura da imagem. A operação de convolução consiste em processar a imagem I através do kernel K, descrita pela seguinte fórmula:

$$S_{(i,j)} = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n), \qquad (2\text{-}14)$$

onde :

- $S_{(i,j)}$ representa o mapa de características resultante da operação de convolução;
- $(I * K)(i,j)$ descreve a operação de convolução entre a imagem I e o kernel K na posição (i,j);
- $\sum_m \sum_n I(i+m, j+n)K(m,n)$ é a fórmula da convolução, onde m e n percorrem as dimensões do kernel K, e $i+m$, $j+n$ são os índices na imagem I sobre os quais o kernel está sendo aplicado [34].

Simplificadamente, a convolução consiste em deslizar uma janela (Kernel) da esquerda para a direita sobre uma imagem para obter o mapa de características resultante.

A Figura 2.7 ilustra a operação de convolução entre a imagem I e o kernel K.

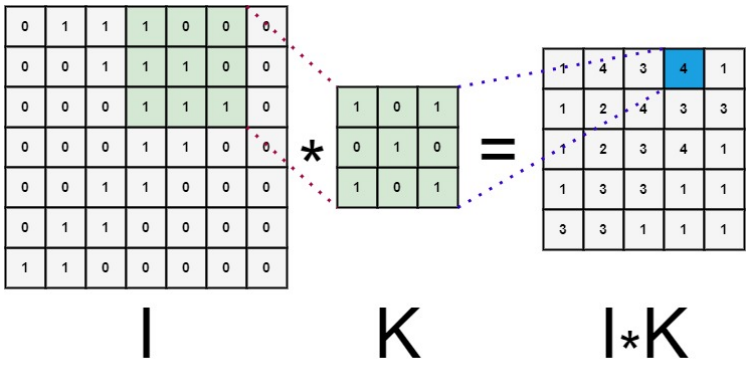

Figura 2.7: Ilustração do kernel K convoluindo sobre a imagem I.

A convolução também desempenha um papel crucial na redução da dimensão espacial da imagem de entrada. Dependendo do objetivo, pode-se desejar manter a dimensão espacial do mapa de características resultante igual à da imagem de entrada, ou reduzi-la. É nesse contexto que o parâmetro *padding* se torna essencial.

Existem vários tipos de padding (preenchimento) [35], sendo os mais utilizados:

- **Preenchimento com Zeros (Zero padding)**: Consiste em adicionar zeros ao redor das bordas da imagem ou do mapa de características, permitindo um controle preciso sobre a dimensão do mapa de características de saída.
- **Preenchimento Válido (Valid padding)**: Consiste em não adicionar nenhum elemento adicional à imagem ou ao mapa de características, o que pode resultar em uma saída com dimensão menor em comparação com a entrada.

Outro parâmetro muito importante, que afeta não apenas a dimensão de saída, mas também o recobrimento do campo receptivo [35], é o *stride* de convolução. O stride de convolução determina de quantos pixels o kernel se move horizontalmente e verticalmente durante sua convolução na imagem ou no mapa de características. Geralmente, o stride de 1 é utilizado, o que significa que o kernel se move um pixel de cada vez.

2.2.3 Camada de Pooling

Após a extração dos mapas de características $S_{(i,j)}$ pela camada de convolução, aplica-se uma camada de *pooling* para reduzir gradualmente a dimensão espacial dos mapas de características, resultando na compressão dos

dados de entrada. Para isso, a camada de pooling divide os dados de entrada em pequenas regiões, chamadas de *janelas de pooling* ou campos receptivos, e realiza uma operação de agregação, como a tomada do valor máximo (Max Pooling) ou médio (Average Pooling), em cada janela [36]. A Figura 2.8 ilustra a operação de max pooling.

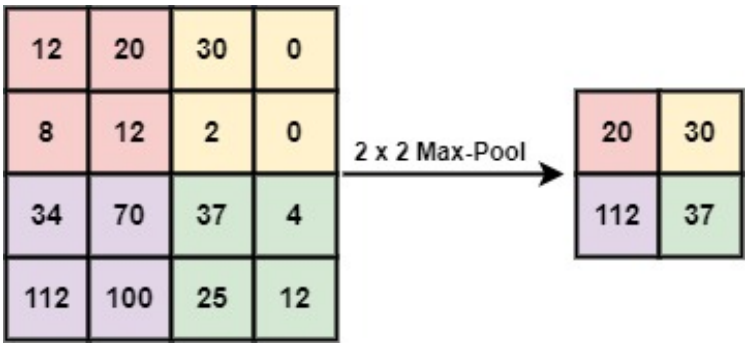

Figura 2.8: Ilustração da operação de max pooling.

2.2.4 Camada Totalmente Conectada

A camada totalmente conectada desempenha um papel crucial na classificação e na tomada de decisão final da rede neural convolucional. Após uma série de camadas de convolução e pooling, os mapas de características são achatados pela camada de achatamento (flatten layer) [37], transformando-os em um vetor unidimensional $\mathbf{V}$, que é então transmitido para a camada totalmente conectada. Nesta etapa, o processo de aprendizado descrito na seção Rede Neural artificial é retomado para ajustar os pesos e os vieses através da retropropagação do erro.

2.2.5 ResNet: Redes Neurais Residuais

Embora as arquiteturas baseadas em aprendizado profundo (como CNNs e redes neurais) tenham demonstrado grande capacidade de aprender funcionalidades complexas, elas enfrentam problemas à medida que se tornam mais profundas (com mais camadas). Um dos principais desafios é o *gradiente desaparecendo (vanishing gradients)*, onde os gradientes se tornam muito pequenos durante a Retropropagação do erro, dificultando o treinamento das camadas iniciais da rede [38].

As redes neurais residuais, ou ResNets, foram introduzidas para resolver esse problema. A principal inovação da ResNet é a introdução de blocos residuais, que permitem a criação de redes extremamente profundas sem sofrer degradação de desempenho [38].

2.2.5.1 Blocos Residuais

A chave para o sucesso da ResNet é a adição de conexões residuais ([38] chamou de *shortcut connections*) que *saltam* uma ou mais camadas. Em vez de aprender diretamente a função de mapeamento desejada, os blocos residuais aprendem a diferença (ou resíduo) entre a entrada e a saída esperada. Um bloco residual é definido como:

$$y = F(x, W_i) + x \tag{2-15}$$

Onde y é a saída do bloco residual, $F(x, W_i)$ representa a transformação aprendida pelas camadas internas do bloco residual (com os pesos W_i), e x é a entrada original do bloco [38]. A Figura 2.9 ilustra um bloco residual.

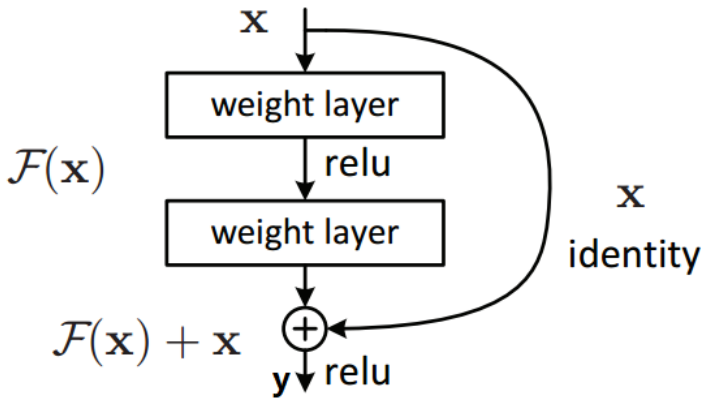

Figura 2.9: Ilustração de um bloco residual em uma ResNet [38].

A introdução da ResNet levou ao desenvolvimento de várias variantes, como ResNet-50, ResNet-101 e ResNet-152, onde o número se refere à profundidade da rede em camadas [38]. Essas variantes utilizam blocos residuais para aprender características complexas de forma mais eficiente, superando significativamente as arquiteturas tradicionais de redes neurais convolucionais sem blocos residuais.

2.3 Transformers

Apresentado pela primeira vez no artigo *Attention Is All You Need* [39], o modelo Transformer é uma arquitetura de rede neural que revolucionou o campo da aprendizagem profunda, na área do processamento de linguagem natural (NLP). É um modelo de transdução de sequência que se baseia inteiramente em mecanismos de autoatenção, capaz de modelar dependências de longa distância sem a necessidade de processamento sequencial [40].

O NLP é uma área de pesquisa e aplicação que explora como os computadores podem ser usados para entender e manipular textos ou falas em linguagem natural [41], abrindo caminho para uma ampla gama de aplicações, como extração de informações [42], tradução automática [39], reconhecimento de voz [43], etc.

Nos últimos anos, os Transformers também têm se mostrado eficazes em tarefas de visão computacional devido à sua arquitetura única e adaptável. Por exemplo, modelos como Vision Transformer (ViT) [44] têm demonstrado desempenho superior em tarefas como classificação de imagens [45] e detecção de objetos [46].

2.3.1 Arquitetura do modelo Transformer

O modelo Transformer é composto por duas partes principais: o *codificador (encoder)* e o *decodificador (decoder)*, cada um formado por uma pilha de camadas com estruturas idênticas, mas com pesos diferentes [39]. A Figura 2.10 mostra a arquitetura do modelo Transformer onde ① e ② representam o codificador e o decodificador, respectivamente.

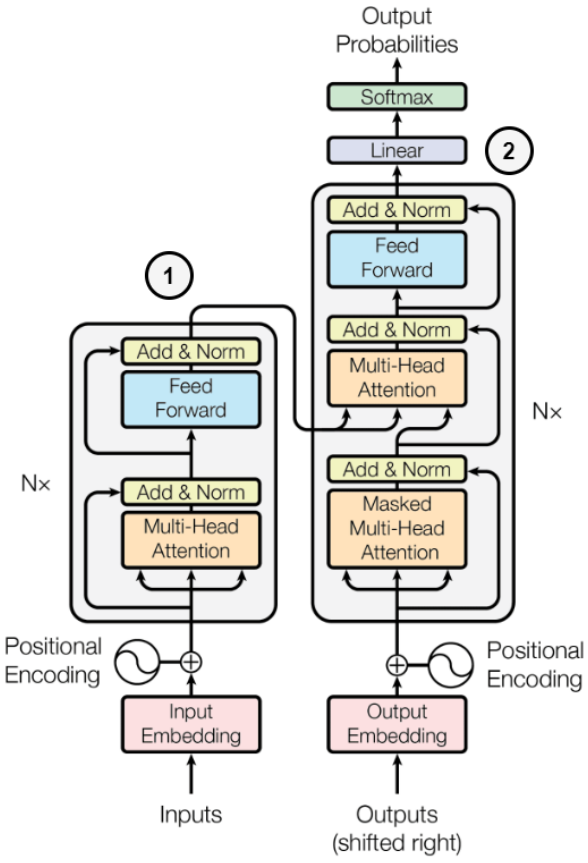

Figura 2.10: Ilustração da arquitetura do modelo Transformer [39].

Ao receber uma entrada sequencial (como uma palavra ou frase) previamente transformada em vetores através de um algoritmo de incorporação (embedding) [47], juntamente com as posições de cada palavra na sequência (Positional Encoding), o codificador gera uma representação matricial. Esta representação matricial é então passada para o decodificador, que iterativamente gera a saída.

2.3.1.1 Codificador

O codificador é composto por N camadas idênticas ($N = 6$ foram usadas em [39]). Cada camada é composta por dois subcomponentes principais: o mecanismo de autoatenção multi-cabeça (multi-head self-attention mechanism) e a rede propagação para frente totalmente conectada position-wise (position-wise fully connected feed-forward network) [39].

1. **Camada de autoatenção multi-cabeça**:

   Ela permite que o modelo pondere a importância de cada elemento na sequência de entrada em relação a todos os outros, através do mecanismo de *autoatenção*, facilitando assim a captura de dependências de longo alcance [39].

   O cálculo da autoatenção, chamado de *Scaled Dot-Product Attention* em [39], pode ser decomposto em cinco etapas:

   a. **Criação dos vetores de consulta, chave e valor (1)**:

      Três vetores são gerados a partir de cada vetor de entrada X_i (ou seja, as palavras previamente transformadas pelo algoritmo de incorporação (embedding) [47]) multiplicando-os por matrizes de pesos específicas: W_q para as consultas (queries), W_k para as chaves (keys) e W_v para os valores (values). Essas matrizes de pesos são treináveis durante o treinamento [39].

      $$q_i = W_q \cdot X_i, \quad k_i = W_k \cdot X_i, \quad v_i = W_v \cdot X_i, \tag{2-16}$$

      onde q_i, k_i e v_i são os vetores de consulta (query), chave (key) e valor (value), respectivamente, para a palavra X_i.

   b. **Cálculo dos scores de atenção (2)**:

      Para cada par de palavras na sequência de entrada, calcula-se o produto escalar entre o vetor de consulta q_i da palavra atual e o vetor de chave k_j de cada palavra na sequência resultando nos scores de atenção.

$$score_{ij} = q_i \cdot k_j, \tag{2-17}$$

onde i é o índice da palavra atual e j é o índice de cada palavra na sequência.

c. **Normalização dos scores (3)**:

Os scores de atenção são multiplicados por um fator de $\frac{1}{\sqrt{d_k}}$ (onde d_k é a dimensão dos vetores de chave) e normalizados com uma função softmax para obter uma distribuição de probabilidade. Isso ajuda a estabilizar o gradiente durante o treinamento. Como o score de atenção é obtido pelo produto escalar entre o vetor de consulta q_i e o vetor de chave k_i, quando a dimensão d_k é muito grande, o score resultante tende a ser muito alto, empurrando a função softmax para regiões onde seus gradientes são muito pequenos [39].

$$\text{softmax}\left(\frac{score_{ij}}{\sqrt{d_k}}\right). \tag{2-18}$$

d. **Ponderação dos vetores de valor (4)**:

Os scores de atenção normalizados são usados para ponderar os vetores de valor correspondentes, resultando em uma representação que enfatiza as palavras mais relevantes. A intuição por trás desse processo é manter intactos os valores das palavras em que queremos focar e atenuar os valores das outras palavras [39].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2-19}$$

onde Q é a matriz de consultas, K a matriz de chaves, e V a matriz de valores.

e. **Cálculo do vetor de saída (5)**:

A última etapa consiste em somar os vetores de valor ponderados para cada palavra. Isso resulta no vetor de saída final para a palavra atual. Esse vetor de saída contém a informação contextual ponderada por relevância, capturada através dos scores de atenção, permitindo que o modelo considere as dependências de longo alcance e a importância relativa das palavras na sequência de entrada [39].

A fim de melhorar o desempenho, ao invés de utilizar uma única camada de autoatenção (ou seja, realizar as etapas 1, 2, 3, 4, 5 apenas uma vez), [39] propõe o uso de várias cabeças de autoatenção em paralelo, resultando no mecanismo com o nome *autoatenção multi-cabeça* (multi-head self-attention). Esse mecanismo permite ao modelo focar em diferentes partes da sequência de entrada de forma simultânea,

capturando múltiplas dependências contextuais e, além disso, permitindo um alto grau de paralelismo..

Cada cabeça de autoatenção realiza as etapas de 1 a 5 de forma independente, e os vetores de saída ponderados de todas as cabeças são então concatenados e projetados através da rede propagação para frente totalmente conectada position-wise (position-wise fully connected feed-forward network) para obter a saída final da camada de autoatenção multi-cabeça.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W_O, \quad \text{(2-20)}$$

onde $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

Aqui, W_i^Q, W_i^K, W_i^V são matrizes de pesos para a i-ésima cabeça de atenção, e W_O é a matriz de pesos usada para projetar as cabeças concatenadas de volta ao espaço de dimensões originais [39].

A Figura 2.11 mostra o mecanismo de autoatenção chamado Scaled Dot-Product Attention ① e a camada de autoatenção multi-cabeça (Multi-Head Attention) ②.

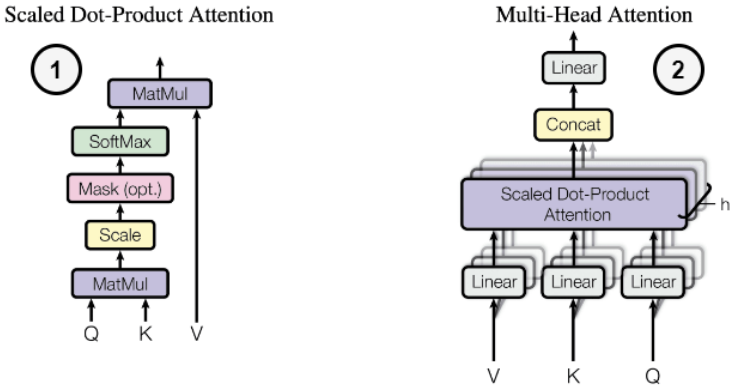

Figura 2.11: Ilustração do mecanismos de autoatenção e de autoatenção multi-cabeça [39].

f. **Codificação Posicional (Positional Encoding)**:

Dado que a entrada do codificador é uma sequência ordenada (por exemplo, uma sucessão de palavras numa frase) e o mecanismo de autoatenção visa aprender as relações entre os diferentes elementos da sequência afim de criar um contexto apropriado, é crucial de conhecer a posição de cada elemento na sequência em relação aos demais. Para conseguir isso, [39] propôs adicionar codificação posicional às sequências de entrada para preservar a dependência temporal entre elas. Dois tipos

de codificação posicional foram usados em [39], eles são codificação posicional *seno* e *cosseno*.

A codificação posicional é definida pelas seguintes fórmulas:

$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), \tag{2-21}$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), \tag{2-22}$$

onde $\mathrm{PE}_{(pos,2i)}$ e $\mathrm{PE}_{(pos,2i+1)}$ são os componentes *senoidal* e *cossenoidal* da codificação posicional para uma posição *pos* e dimensão i, *pos* é a posição na sequência, d é a dimensão do vetor de embedding.

2. **Rede propagação para frente totalmente conectada position-wise (position-wise fully connected feed-forward network)**:

   Como mostrado na Figura 2.10 além do mecanismo de autoatenção multi-cabeça, cada camada do codificador (lembrando que $N = 6$ no [39]) possui uma rede feed-forward totalmente conectada, que é aplicada de maneira independente a cada posição. Esta rede feed-forward é composta por duas camadas lineares com uma ativação não linear no meio. A principal função desta rede é transformar a representação de cada posição de forma não linear e agregar mais complexidade ao modelo [39].

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2-23}$$

   Onde W_1 e W_2 são matrizes de pesos, e b_1 e b_2 são os vieses. A função de ativação utilizada é a ReLU (Rectified Linear Unit), indicada pela função $\max(0, x)$.

2.3.1.2 Decodificador

O decodificador do modelo Transformer (② na Figura 2.10) é responsável por gerar a saída final com base na representação matricial fornecida pelo codificador e nas entradas anteriores do decodificador. Ele também é composto por N camadas idênticas ($N = 6$ foram usadas em [39]), cada uma contendo três subcomponentes principais: o mecanismo de autoatenção mascarada, o mecanismo de atenção multi-cabeça encodificador-decodificador e a rede feed-forward totalmente conectada.

1. **Camada de Autoatenção Mascarada**:

A camada de autoatenção mascarada no decodificador é semelhante à camada de autoatenção no codificador, com a diferença crucial de que utiliza uma máscara para impedir que cada posição atenda a posições futuras. Isso é importante para garantir que as previsões feitas pelo modelo dependam apenas de entradas passadas e presentes, mantendo a causalidade da sequência [39]. A fórmula para o cálculo é similar ao codificador:

$$\text{MaskedAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V, \tag{2-24}$$

onde M é a máscara que atribui um valor muito negativo (-∞) a posições não permitidas, assegurando que elas não influenciem a atenção.

2. **Mecanismo de Atenção Multi-Cabeça Encodificador-Decodificador**:

   Após a camada de autoatenção mascarada, o decodificador aplica um mecanismo de atenção multi-cabeça que utiliza as representações do codificador. Esse mecanismo permite que o decodificador foque em diferentes partes da sequência de entrada ao mesmo tempo, combinando informações contextuais de forma mais rica e diversificada.

3. **Rede Feed-Forward Totalmente Conectada**:

   Cada camada do decodificador inclui também uma rede feed-forward totalmente conectada, aplicada de forma independente a cada posição da sequência. Essa rede é composta por duas camadas lineares com uma ativação ReLU entre elas, o que permite a modelagem de relações não-lineares complexas entre as representações.

4. **Conexões Residuais e Normalização de Camada**:

   Assim como no codificador, cada subcomponente do decodificador é seguido por conexões residuais e normalização de camada. Isso ajuda a estabilizar o treinamento e permite que o gradiente flua melhor através das camadas profundas:

$$\text{SublayerOutput} = \text{LayerNorm}(x + \text{Sublayer}(x)). \tag{2-25}$$

2.4 Detecção de objetos

A detecção de objetos [48, 49, 50, 51] é uma tarefa de visão computacional na qual o objetivo é detectar e localizar objetos de interesse em uma imagem ou vídeo. Ela consiste em identificar a posição e a delimitação dos objetos em uma imagem por meio de uma caixa delimitadora retangular (bounding box) e, em seguida, classificá-los de acordo com as categorias às quais pertencem. É um elemento crucial no reconhecimento visual, juntamente com a classificação [52] e a recuperação de imagens [53, 54].

Quando se trata de detecção de objetos, geralmente são usadas duas abordagens: a abordagem de uma etapa, em que a prioridade é dada à velocidade de detecção e classificação de objetos [55, 56] pelo modelo, e a abordagem de duas etapas, em que a prioridade é dada à precisão da detecção e classificação realizada pelo modelo [**?**].

2.4.1 Detecção de objetos em documentos digitais

A detecção de objetos em documentos digitais, como ilustrado na Figura 2.12, consiste em identificar e localizar entidades estruturais e lógicas em documentos digitalizados, como títulos, textos, parágrafos, tabelas, gráficos, equações, listas, figuras, etc. Diferentemente da detecção de objetos padrão, que usa conjuntos de dados como COCO (Common Objects in Context) [57] ou PASCAL VOC, onde as entidades (como animais, veículos, pessoas) não são necessariamente de natureza estrutural, a detecção em documentos foca em características estruturais. Um documento digital (ou físico) é caracterizado por uma apresentação estruturada de seu conteúdo, em que a fonte, o estilo, a orientação, a posição e o tamanho podem variar de acordo com a localização relativa das entidades no documento e com as informações que ele deve transmitir.

Embora ambos os tipos de detecção tenham como objetivo identificar e localizar objetos, a detecção em documentos é particularmente útil em aplicativos de processamento de documentos. Graças à sua capacidade de localizar precisamente as entidades, essa detecção permite agilizar o processo de extração automática de informações utilizando ferramentas como OCR (Optical Character Recognition). Além disso, organiza o conteúdo de forma estruturada, abrindo caminho para outros aplicativos relacionados, como o enriquecimento de bancos de dados e a criação de chatbots [58, 59].

Dada a complexidade da tarefa, o uso de modelos de detecção sofisticados, como Mask R-CNN [60], Mask DINO [**?**] e RT-DETR [51], é uma alternativa

muito eficaz. Esses modelos aproveitam técnicas avançadas de aprendizado profundo para detectar, segmentar e classificar com acurácia as diferentes entidades dentro dos documentos.

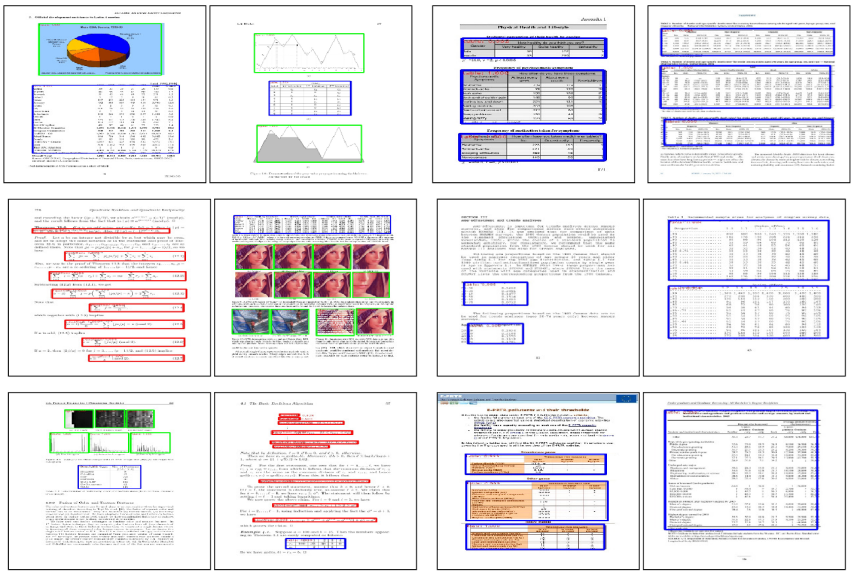

Figura 2.12: Ilustração de detecção de objetos em documentos digitais [61].

2.5 Mask R-CNN

Mask R-CNN (Region-based Convolutional Neural Network) [60], criado em 2017, é um modelo de segmentação de instâncias de ponta (estado da arte) baseado em aprendizado profundo. Trata-se de uma extensão do modelo Faster R-CNN [50], com a adição de um ramo de predição de máscaras de objetos paralelamente ao ramo existente de detecção de caixas delimitadoras.

De fato, a segmentação de instâncias é uma tarefa de visão computacional que combina:

- **A detecção de objetos**: cujo objetivo é, como especificado na seção Detecção de objetos, localizar e delimitar objetos individuais em uma imagem por meio de retângulos ou, mais precisamente, caixas delimitadoras (bounding box) e
- **A segmentação semântica**: cujo objetivo é classificar cada pixel da imagem em uma classe específica sem distinguir as diferentes instâncias de uma mesma classe. Em outras palavras, trata-se de atribuir um mesmo rótulo de pixel aos elementos de mesma natureza em uma imagem.

A Figura 2.13 mostra respectivamente, detecção de objetos (a), a segmentação semântica (b) e a segmentação de instâncias (c).

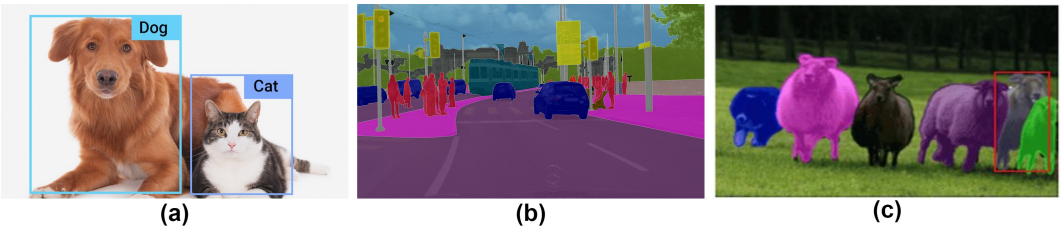

Figura 2.13: Detecção de objetos, segmentação semântica e segmentação de instâncias.

Mask R-CNN, ao combinar essas duas tarefas, permite fornecer uma máscara detalhada para cada instância de objeto. A Figura 2.14 mostra a arquitetura do modelo Mask R-CNN.

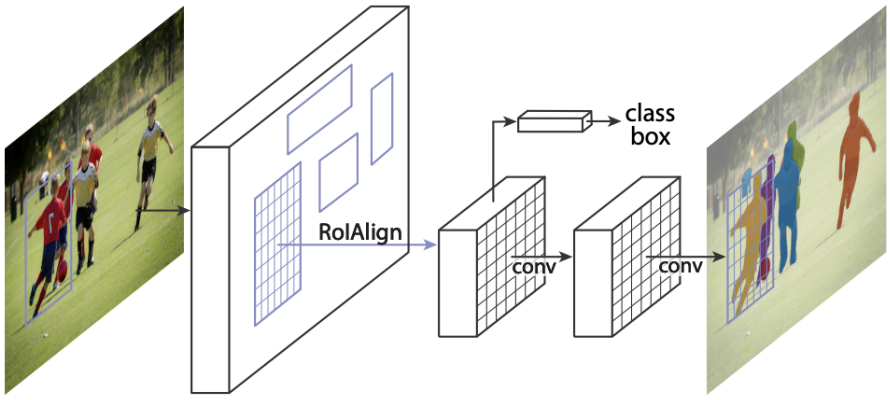

Figura 2.14: Arquitetura do modelo Mask R-CNN [60]

O modelo possui três etapas, que são:

1. Rede de Proposição de Regiões (Region Proposal Network, RPN);
2. Alinhamento da Região de Interesse (Region of Interest Alignment, RoIAlign);
3. Geração de Máscara

2.5.1 Rede de Proposição de Regiões (RPN)

A RPN foi proposta em [50] para solucionar as limitações (como custo computacional e lentidão) relacionadas aos métodos anteriores de proposição de regiões de interesse [62, 63].

Como o nome sugere, a rede, em seu funcionamento, toma uma imagem (de qualquer tamanho) como entrada e produz um conjunto de objetos retangulares (coordenadas do centro, altura e largura), cada um com uma pontuação de objetividade [50]. O RPN funciona da seguinte maneira:

2.5.1.1 Geração de mapas de características

A primeira etapa consiste em usar uma rede neural convolutiva (CNN) ([64, 65] foram usadas em [50]) para extrair mapas de características (feature maps) da imagem de entrada. Esses mapas de características, graças ao processo de convolução, capturam as informações bidimensionais essenciais necessárias (semânticas e espaciais) para identificar os objetos.

2.5.1.2 Colocação das caixas de ancoragem (anchor boxes)

Em cada ponto do mapa de características, por meio de uma janela deslizante [50], várias caixas de ancoragem [1] (9 foram usadas em [50]) de diferentes tamanhos (size) e proporções (aspect ratio) são colocadas. Essas caixas de ancoragem servem como pontos de partida para a geração das propostas de regiões. Cada caixa de ancoragem é avaliada para determinar se contém um objeto ou não [50]. A Figura 2.15 mostra um exemplo de caixas de ancoragem num ponto do mapa de caractéristica.

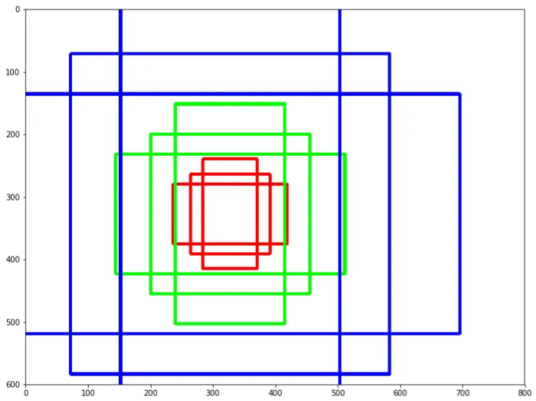

Figura 2.15: Ilustração de 9 caixas de ancoragem de diferentes tamanhos e proporções geradas em um ponto do mapa de características.

2.5.1.3 Classificação e regressão

Para cada caixa de ancoragem, o RPN realiza duas tarefas:

- **Classificação**: consiste em determinar se a caixa contém um objeto (classe foreground) ou não (classe background),
- **Regressão das caixas**: consiste em ajustar as dimensões da caixa de ancoragem para melhor corresponder ao objeto detectado. Essa regressão

[1] Caixas de ancoragem são candidatas para prever caixas delimitadoras.

prevê os deslocamentos relativos em termos de centro (em abscissa e ordenada), altura e largura em relação à caixa de ancoragem inicial.

Essas duas etapas são realizadas por meio da função de perda indicada pela fórmula [50]:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{\text{cls}}}\sum_i L_{\text{cls}}(p_i, p_i^*) + \lambda\frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{\text{reg}}(t_i, t_i^*), \tag{2-26}$$

onde L_{cls} é a perda de classificação, L_{reg} é a perda de regressão, p_i é a probabilidade prevista de a âncora i conter um objeto, p_i^* é o Ground Truth (1 se a âncora contém um objeto, 0 caso contrário), t_i é o vetor de deslocamento previsto para a âncora i, e t_i^* é o vetor de deslocamento do Ground Truth [50]. A Figura 2.16 mostra o funcionamento da RPN:

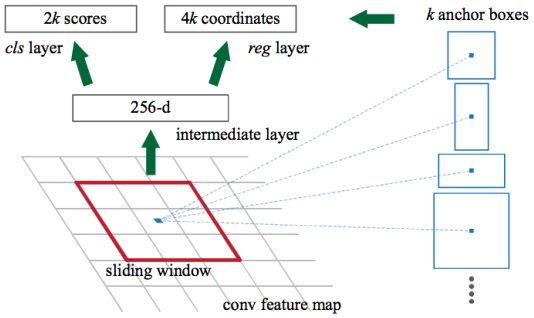

Figura 2.16: Ilustração do mecanismo da rede de propostas de região (RPN) [50].

2.5.1.4 Supressão Não-Maximal (NMS)

Após a geração das propostas de regiões, uma supressão não-maximal (Non-Maximum Suppression, NMS) é aplicada para eliminar as propostas redundantes e manter apenas as caixas com a maior *pontuação de objetividade (objectness Score)*. Esta etapa reduz o número de propostas e permite focar nas regiões mais prováveis de conter objetos. A NMS utiliza a pontuação de objetividade e um limiar de Intersection over Union (IoU) para decidir quais caixas manter. O IoU [66] mede a proporção entre a interseção e a união de duas caixas delimitadoras e permite eliminar as caixas que têm uma alta sobreposição com a caixa que possui a maior pontuação de objetividade. A Figura 2.17 mostra a aplicação do NMS na eliminação das regiões propostas redundantes.

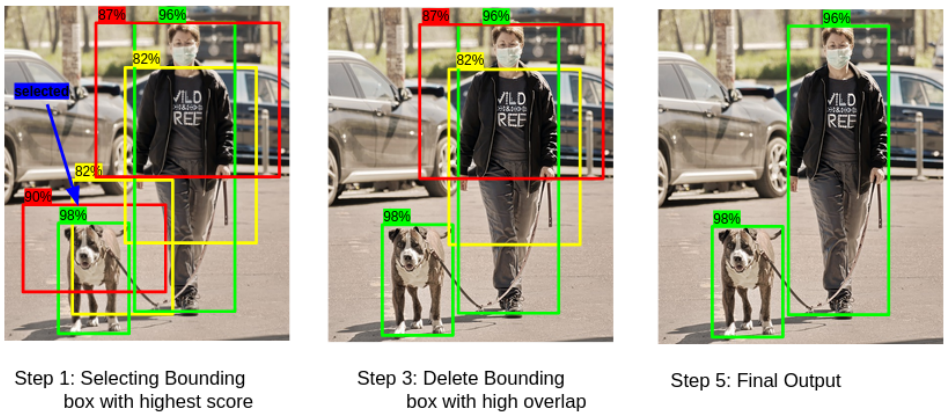

Figura 2.17: Ilustração do mecanismo do NMS na eliminação das regiões propostas redundantes.

2.5.2 Alinhamento da Região de Interesse (RoIAlign)

Uma vez que as regiões propostas são geradas pelo RPN e refinadas pelo processo de NMS, elas se tornam regiões de interesse (RoIs). As RoIs devem ser normalizadas para um tamanho fixo antes de serem processadas pelas camadas seguintes da rede. Diferentemente do método anterior baseado em quantificação usado em [50], que é o RoI Pooling e que mostrou induzir perda de informação, o RoIAlign utiliza o método de interpolação bilinear.

Para isso, é realizada uma projeção da região de interesse no mapa de características. Em seguida, a região de interesse é dividida em $N \times N$ regiões iguais. Em cada quadrado resultante da divisão, é realizada a interpolação bilinear, começando com a criação de 4 pontos de amostragem uniformemente distribuídos. Cada ponto de amostragem é então interpolado bilinearmente a partir dos pixels adjacentes/próximos do mapa de características.

A interpolação bilinear é realizada da seguinte maneira: Para cada ponto de amostragem, os valores dos pixels vizinhos são usados para calcular um valor interpolado. A fórmula de interpolação bilinear é:

$$P(x,y) = \frac{y_2 - y}{y_2 - y_1} \cdot \left(\frac{x_2 - x}{x_2 - x_1} Q_{11} + \frac{x - x_1}{x_2 - x_1} Q_{21} \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} Q_{12} + \frac{x - x_1}{x_2 - x_1} Q_{22} \right), \tag{2-27}$$

onde $Q_{11}, Q_{21}, Q_{12}, Q_{22}$ são os valores dos pixels vizinhos, e x_1, x_2, y_1, y_2 são as coordenadas desses pixels. $P(x,y)$ é o valor interpolado no ponto de amostragem (x,y).

Os valores interpolados para os 4 pontos de amostragem são então agregados para obter um único valor representativo da região correspondente no mapa de características.

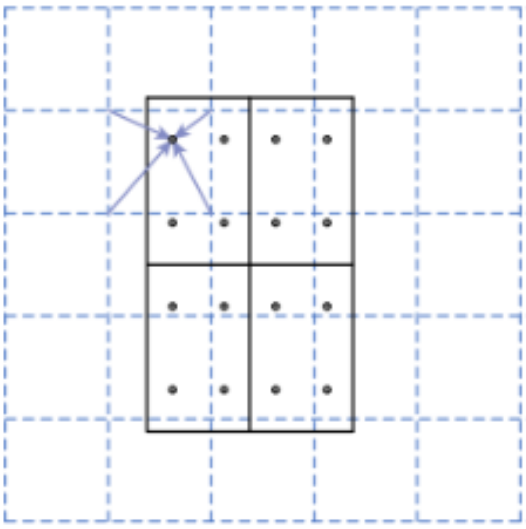

Figura 2.18: Ilustração do mecanismo de RoIAlign [60].

2.5.3 Geração de Máscara

Uma vez que as regiões de interesse (RoIs) foram identificadas e alinhadas pelo RoIAlign, o modelo Mask R-CNN passa para a geração de máscaras para cada instância detectada. O ramo de geração de máscaras adiciona uma convolução totalmente conectada para prever uma máscara binária para cada RoI. Este ramo funciona paralelamente aos ramos de classificação e regressão das caixas delimitadoras, permitindo obter uma segmentação precisa ao nível dos pixels para cada instância de objeto [60].

2.6 Mask DINO

Mask DINO (DEtection TRansformer (DETR) with Improved deNoising anchOr boxes) [**?**], assim como o Mask R-CNN, é um modelo de segmentação de instâncias. A principal diferença reside no fato de que Mask DINO é baseado na arquitetura Transformer. Este modelo é uma extensão do modelo DINO(DEtection TRansformer with Improved deNoising anchOr boxes) [67], sendo ele próprio uma melhoria de [46, 68, 69].

A Figura 2.19 mostra a arquitetura do modelo Mask DINO.

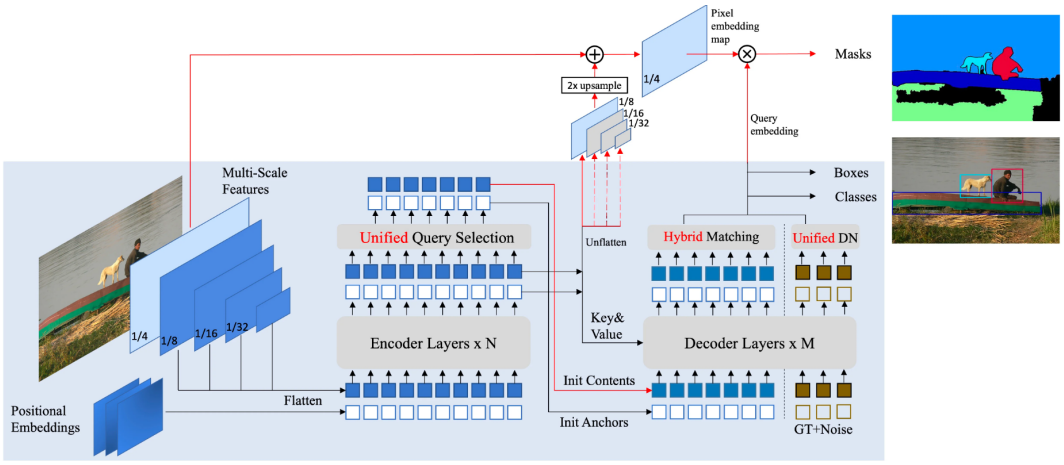

Figura 2.19: Arquitetura do modelo Mask DINO [?].

A área sombreada em azul representa a arquitetura original do modelo DINO [67], enquanto as linhas vermelhas indicam o design adicional para a segmentação.

Recebendo uma imagem I como entrada, os backbones, como ResNet [38] ou Swin Transformer [70], extraem mapas de características em diferentes escalas (1/4, 1/8, 1/16 e 1/32, como utilizado em [67, ?]). Esses mapas de características, juntamente com seus embeddings posicionais, são então introduzidos no codificador do Transformer. O codificador do Transformer utiliza o mecanismo de auto-atenção, explicado na seção Transformer, para criar um contexto global, gerando representações mais densas dos mapas de características de entrada.

Na saída do codificador, diferentemente de [67], que usa a abordagem de seleção mista de consultas para inicializar as âncoras como consultas posicionais para o decodificador e deixa o conteúdo das consultas como um parâmetro aprendível, o Mask DINO utiliza a abordagem de seleção unificada. Esta abordagem inicializa as âncoras como consultas posicionais para o decodificador, como em [67], mas também inicializa o conteúdo das consultas a partir das melhores consultas provenientes do codificador.

2.7 RT-DETR

RT-DETR (Real Time Detection Transformer) [51], comparado aos modelos Mask R-CNN e Mask DINO, é um modelo dedicado exclusivamente à detecção de objetos. Assim como Mask DINO, ele se beneficia da estrutura adaptável da arquitetura Transformer. Como o nome indica, é um modelo de detecção em tempo real que maximiza tanto a velocidade quanto a precisão da detecção sem a necessidade de Supressão Não-Maximal (NMS) [51].

A Figura 2.20 apresenta a arquitetura do modelo RT-DETR.

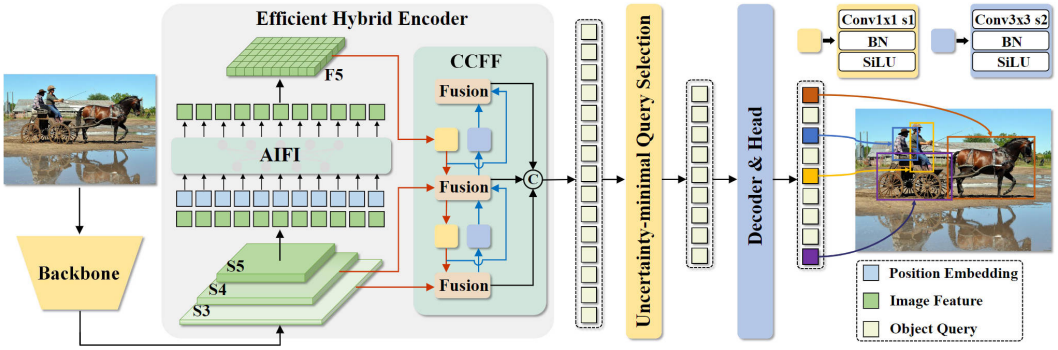

Figura 2.20: Arquitetura do modelo RT-DETR [51].

RT-DETR é composto por três componentes principais: um backbone, um encoder híbrido eficiente (Efficient Hybrid Encoder) e um decoder Transformer com cabeças de predição auxiliares. O backbone extrai características multiescalares da imagem de entrada, que são então processadas pelo encoder híbrido. Este encoder transforma as características multiescalares em uma sequência de características de imagem por meio da interação de características intra-escala baseada em atenção (Attention-based Intra-scale Feature Interaction, AIFI) e da fusão de características inter-escalas baseada em CNN (CNN-based Cross-scale Feature Fusion, CCFF). Em seguida, o mecanismo de seleção de consultas de incerteza mínima (uncertainty-minimal query selection) seleciona um número fixo de características do encoder para servir como consultas iniciais de objetos para o decoder. Por fim, o decoder com cabeças de predição auxiliares otimiza iterativamente as consultas de objetos para gerar categorias e caixas delimitadoras [51].

2.7.1 Encoder híbrido

O encoder híbrido do modelo RT-DETR melhora significativamente a velocidade de processamento. O encoder é projetado para desacoplar a interação de características intra-escala da fusão de características inter-escala. Essa abordagem atenua o gargalo (bottleneck) computacional tipicamente associado aos encoders Transformer multi-escala [51].

O encoder híbrido utiliza duas estratégias principais:

2.7.1.1 Interação de Características Intra-escala Baseada em Atenção (AIFI)

Este componente se concentra nas características de alto nível que encapsulam uma rica informação semântica, permitindo um processamento

eficiente sem interações desnecessárias entre todas as escalas [51].

2.7.1.2 Fusão de Características Inter-escala Baseada em CNN (CCFF)

Este módulo funde eficientemente as características de diferentes escalas, otimizando a representação da informação multi-escala enquanto mantém baixos custos computacionais [51].

2.7.2 Mecanismo de seleção de consultas de incerteza mínima

A seleção de consultas minimizando a incerteza, de acordo com [51], visa melhorar a inicialização das consultas de objeto, reduzindo a incerteza das características selecionadas. Este método otimiza explicitamente a incerteza epistêmica para fornecer consultas de alta qualidade ao decodificador, melhorando assim a precisão geral do modelo em classificação e localização de objetos [51].

2.8 K-means

O algoritmo K-means [71] é um dos algoritmos de agrupamento especializados em aprendizado não supervisionado. Ele é usado para particionar um conjunto de dados em um número predefinido de K clusters. O objetivo do agrupamento é reunir pontos de dados semelhantes e descobrir padrões ou estruturas subjacentes dentro dos dados. Para isso, o K-means visa minimizar a distância intra-cluster (a distância entre os pontos de um mesmo grupo) e maximizar a distância inter-cluster (a distância entre os pontos de diferentes grupos).

Cada cluster é representado por um centróide, que é o ponto central desse cluster. Inicialmente, K centróides são escolhidos (de maneira aleatória ou segundo um método determinado [72]), e o algoritmo funciona recalculando continuamente os centróides dos clusters e reatribuindo os pontos de dados até que a convergência seja atingida.

A função de perda usada pelo algoritmo K-means é chamada soma dos quadrados dos erros (Sum of Squared Errors ou SSE) [73]. Essa função mede a variância dentro dos clusters calculando a soma das distâncias ao quadrado entre cada ponto de dados e o centróide de seu cluster. Formalmente, a função de perda J para K-means pode ser expressa da seguinte forma:

$$J = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2, \tag{2-28}$$

onde:

- K é o número de clusters;
- C_i representa o i-ésimo cluster;
- x é um ponto de dados pertencente ao cluster C_i;
- μ_i é o centróide do cluster C_i;
- $\|x - \mu_i\|^2$ é a distância euclidiana ao quadrado entre o ponto de dados x e o centróide μ_i.

O objetivo do algoritmo K-means é minimizar essa função de perda para produzir clusters compactos e bem separados.

2.9 t-SNE

O t-SNE (t-distributed Stochastic Neighbor Embedding) é uma técnica de redução de dimensionalidade não linear e não supervisionada para a exploração e visualização de dados de alta dimensão. A redução de dimensionalidade não linear significa que o algoritmo nos permite separar dados que não podem ser separados por uma linha reta.

Ele é principalmente empregado para visualizar dados de alta dimensão, projetando-os em um espaço de dimensão mais baixa (*geralmente em duas* ou *três dimensões*) enquanto preserva as estruturas *locais* e *globais* dos dados.

O algoritmo t-SNE, proposto em [74], converte as distâncias entre os pontos de dados no espaço de alta dimensão em probabilidades de similaridade. Ele define as probabilidades de maneira que os pontos próximos tenham altas probabilidades de similaridade, enquanto os pontos distantes tenham baixas probabilidades de similaridade. Em seguida, ele tenta minimizar a divergência de Kullback-Leibler (KL) entre essas distribuições de probabilidades de alta dimensão e aquelas no espaço de baixa dimensão [74].

O t-SNE funciona em duas etapas principais:

1. **Cálculo das probabilidades de similaridade no espaço de alta dimensão**: Para cada ponto de dados no espaço de alta dimensão, as distâncias euclidianas para todos os outros pontos são calculadas e convertidas em probabilidades condicionais. Essas probabilidades são simetrizadas para obter as probabilidades conjuntas [74].

2. **Otimização dos pontos no espaço de baixa dimensão**: Usando um algoritmo de descida de gradiente, o t-SNE otimiza as posições dos pontos no espaço de baixa dimensão para minimizar a divergência KL entre as distribuições de probabilidades de alta e baixa dimensões [74].

A função de custo usada pelo t-SNE para avaliar a qualidade da projeção é dada pela divergência de Kullback-Leibler:

$$C = KL(P\|Q) = \sum_i \sum_j P_{ij} \log \left(\frac{P_{ij}}{Q_{ij}} \right), \tag{2-29}$$

onde:

- C é a função de custo a ser minimizada.
- $KL(P\|Q)$ é a divergência de Kullback-Leibler entre as distribuições de probabilidade conjuntas no espaço de alta dimensão (P) e no espaço de baixa dimensão (Q).
- P_{ij} é a probabilidade conjunta de similaridade entre os pontos i e j no espaço de alta dimensão.
- Q_{ij} é a probabilidade conjunta de similaridade entre os pontos i e j no espaço de baixa dimensão.

O objetivo do t-SNE é minimizar essa função de custo para obter uma representação em baixa dimensão onde os pontos próximos no espaço de alta dimensão permanecem próximos e os pontos distantes permanecem distantes.

2.9.1 Perplexidade

Um parâmetro chave no algoritmo t-SNE é a perplexidade. Ela controla o número efetivo de vizinhos que cada ponto considera durante o processo de redução de dimensionalidade. A perplexidade pode ser vista como uma medida de equilíbrio entre a preservação das estruturas *locais* e *globais* dos dados. Valores baixos de perplexidade enfatizam a preservação das relações locais, enquanto valores altos tentam preservar as estruturas globais. Escolher a perplexidade correta é crítico e geralmente requer experimentação, variando tipicamente entre 5 e 50 [74].

3
Metodologia

O objetivo principal deste trabalho, como especificado na Introdução desta dissertação, é criar um pipeline capaz não apenas de gerar imagens de documentos sintéticos, mas também de disponibilizar suas respectivas anotações, criando assim um conjunto de dados de documentos sintéticos. Esses dados serão utilizados para treinar modelos de detecção de objetos capazes de identificar elementos estruturados como textos, títulos, listas, Tabelas e imagens. Essa iniciativa foi tomada devido às diferenças significativas nos layouts entre os documentos PDF da empresa Petrobras e aqueles dos conjuntos de dados públicos, como PubLayNet [5], DocBank [6], e DoclayNet [7], etc. Esta seção tem como objetivo explicar em detalhes os métodos e procedimentos que usamos para atingir esse objetivo.

3.1
Metodologia de criação de conjunto de dados

A Figura 3.1 apresenta o pipeline para a criação de conjuntos de dados de documentos sintéticos. Ele é composto por duas partes principais: o *Pipeline de anotação e processamento de PDFs* e o *Pipeline de geração e processamento de layout de documentos*. A seguir, cada etapa do processo será descrita.

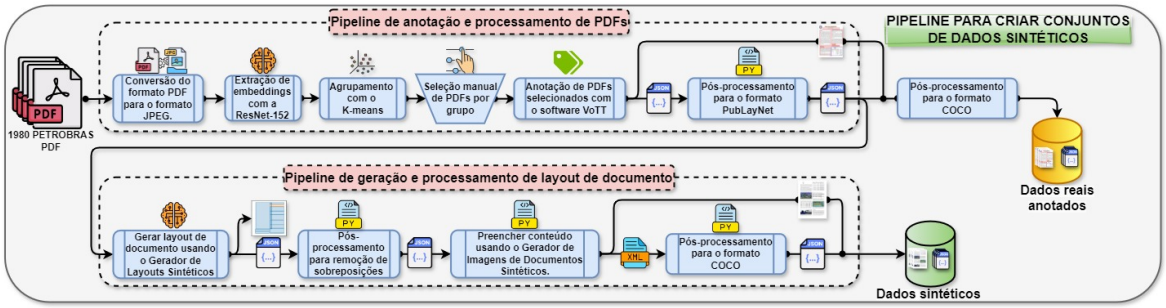

Figura 3.1: Imagem ilustrativa mostrando o nosso pipeline para criar conjuntos de dados de documentos sintéticos.

3.1.1
Pipeline de anotação e processamento de PDFs

O pipeline de anotação e processamento de PDFs é um processo integrado projetado para converter documentos PDF em um formato padronizado de anotações. O processo inicia com a conversão dos documentos PDF e avança por vários estágios de processamento, incluindo a seleção de documentos

representativos, para produzir anotações precisas que atendem aos padrões do domínio. As linhas a seguir descrevem cada parte do pipeline.

3.1.1.1 Coleta de dados

Primeiramente, é importante ressaltar que, nesta seção, por questões de confidencialidade dos dados da empresa Petrobras, nenhuma imagem de documento PDF será divulgada.

A Petrobras, sendo uma empresa estatal brasileira, é especializada na pesquisa, extração, refino, transporte e comercialização de petróleo. Devido à sua posição na indústria petrolífera, desenvolveu formatos de documentação próprios para apresentar os resultados das suas expedições em alto-mar, bem como os seus relatórios de investigação e outros documentos. É, portanto, adequado que estes formatos (dos documentos) sejam concebidos de forma a satisfazer os requisitos específicos da empresa, garantindo precisão, consistência e conformidade com os padrões da indústria petrolífera e regulamentos vigentes.

Através da nossa parceria com a Petrobras, tivemos acesso a um acervo de documentos, totalizando *1.980 documentos PDF*. Esses documentos incluem diversos tipos, como *relatórios operacionais*, *e-mails*, *apresentações técnicas* e outros tipos relevantes. Esses 1.980 documentos PDF constituem o banco de dados e servem como ponto de partida para a criação do conjunto de dados.

3.1.1.2 Conversão de PDF em imagens

Quando se trata de manipular documentos PDF para edição de imagens, integração em software de design, ou melhoria da qualidade e acessibilidade, a conversão para imagens se torna importante. No caso em que uma imagem é tratada como um vetor de números por modelos de inteligência artificial, a conversão dos documentos PDF em imagens, especificamente no formato JPEG (Joint Photographic Experts Group), é uma etapa apropriada e essencial.

Portanto, os 1.980 documentos PDF foram convertidos em *12.995 imagens*. A Tabela 3.1 fornece uma estatística dos dados, incluindo o *número mínimo* de páginas entre os 1.980 documentos PDF, o *número máximo* de

páginas entre esses mesmos documentos PDF, a *média* e o *desvio padrão* do número de páginas.

ID	Descrição	Valores
1	Número mínimo de páginas em 1.980 PDF	1
2	Número máximo de páginas em 1.980 PDF	201
3	Média do número de páginas em 1.980 PDF	≈ 7
4	Desvio padrão do número de páginas em 1.980 PDF	13
5	Total de imagens geradas apois conversão dos 1.980 PDF	12.995

Tabela 3.1: Estatísticas dos 1.980 documentos PDF convertidos em imagens. No caso da linha 5, uma imagem representa uma (1) página do documento PDF se este tiver uma única página, ou várias páginas se o documento PDF tratado tiver várias.

3.1.1.3 Extração de embeddings com a rede Resnet152

Após converter os 1.980 documentos PDF em 12.995 imagens (onde uma imagem representa a página de PDF), nossa próxima etapa foi extrair os embeddings dessas imagens. Em aprendizado de máquina, um embedding é um vetor numérico que representa um objeto (como uma imagem, palavra, usuário, etc.) em um espaço de dimensões reduzidas. Essa transformação permite que modelos de aprendizado automático analisem e interpretem o objeto de maneira eficiente.

Para extrair os embeddings das 12.995 imagens, foi utilizada a ResNet-152 [38], uma rede neural convolucional pré-treinada no conjunto de dados ImageNet [75].

Essa abordagem utiliza a saída da camada de *agrupamento médio (average pooling)* [38] para extrair os embeddings das 12.995 imagens, respectivamente. A Figura 3.2 ilustra o processo de extração de embeddings das 12.995 imagens resultantes da conversão dos 1.980 documentos PDF usando a rede ResNet-152.

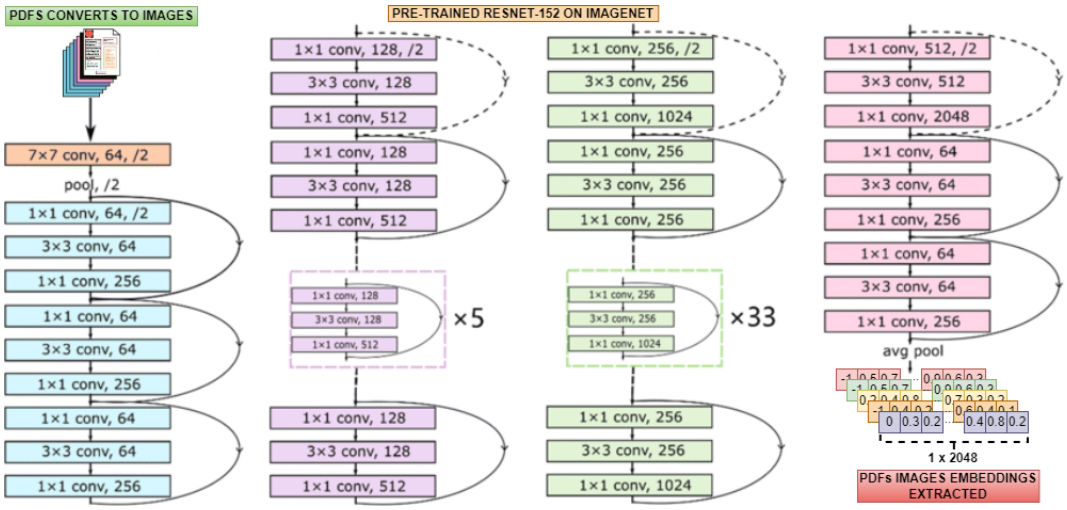

Figura 3.2: Imagem ilustrativa mostrando o processo de extração dos embeddings com a rede ResNet-152. A dimensão de cada embedding extraído é de (1 x 2048 ou seja cada imagem dos 12.995 tem 1 x 2048 como dimensão), o que corresponde aos resultados esperados da camada de agrupamento médio (average pooling) da rede ResNet-152.

3.1.1.4 Agrupamento com o algoritmo K-means

O treinamento de modelos baseados em aprendizado profundo depende diretamente da quantidade e da qualidade dos dados ingeridos, assim como do formato adequado desses dados. No contexto da detecção de objetos, é importante que o conjunto de dados contenha não apenas imagens, mas também as posições precisas dos objetos (anotações) nelas, para garantir um treinamento eficaz. Tradicionalmente, essa anotação manual é um processo trabalhoso e demorado, especialmente quando se trata de milhares de imagens, como as 12.995 imagens resultantes da conversão dos 1.980 documentos PDF.

Para evitar esse trabalho manual extensivo, foi adotada a abordagem baseada no algoritmo de agrupamento K-means [71].

A premissa que adotamos é que como *temos documentos PDF relacionados a um domínio específico (documentos da Petrobras nesse caso), deve haver um grau de similaridade entre suas páginas.*

O K-means [71], como introduzido na seção Background, é um algoritmo de aprendizado não supervisionado que agrupa dados semelhantes. Com o uso dos embeddings das imagens gerados pela ResNet-152 [38], é possível agrupar essas imagens de forma eficiente. Com um número fixo de clusters (K) passado como parâmetro para o algoritmo, podemos selecionar documentos

representativos de cada grupo para anotação, minimizando assim o esforço manual necessário.

a. **Escolha do número fixo de Clusters (K)**:

A escolha do número de grupos K (um número inteiro positivo diferente de zero) é uma etapa crucial no processo de agrupamento com K-means [71], pois impacta diretamente a qualidade e a utilidade dos clusters formados. Determinar o valor ideal de K pode ser um desafio, mas existem várias abordagens e métodos que podem nos ajudar.

Uma das abordagens mais comumente usadas é o método do cotovelo (Elbow method) [72], cujo objetivo é encontrar o K ideal usando uma representação gráfica (basicamente é de testar o algoritmo com diferentes valores de K). O valor K ideal é o ponto em que o gráfico forma um cotovelo.

Neste caso, fizemos um experimento com um $K \in [2, 20]$ grupos. A Tabela 3.2 fornece os parâmetros usados pelo algoritmo K-means durante o experimento.

ID	**Parâmetros**	**Valores**
1	Número do cluster (K)	[2, 20]
2	Random_state	0
3	n_init	auto

Tabela 3.2: Valores dos parâmetros usados em nosso experimento com o algoritmo K-means. Vale lembrar que usamos o algoritmo K-means da biblioteca Sklearn assim que o visualizador KElbowVisusalizer da biblioteca Yellowbrick, portanto, alguns valores de parâmetros são mantidos por padrão.

b. **Redução da dimensionalidade**:

Foi utilizado o algoritmo t-SNE [74] para visualizar e explorar os agrupamentos dos embeddings de dimensão 1 x 2048 (ou seja, das páginas dos documentos PDF).Ajustar os parâmetros como o número de iterações, a perplexidade, a taxa de aprendizado e o número de componentes é essencial para obter uma representação clara dos dados com o t-SNE.

A Tabela 3.3 fornece os valores dos parâmetros utilizados em nosso experimento.

ID	Parâmetros	Valores
1	Número de iterações	1000
2	Perplexidade	5 a 80, com incremento de 5
3	Taxa de aprendizado	auto
4	n_component	2

Tabela 3.3: Valores dos diferentes parâmetros utilizados na nossa experiência com o algoritmo t-SNE. Vale lembrar que usamos o t-SNE da biblioteca Sklearn, portanto, alguns valores de parâmetros são mantidos por padrão.

3.1.1.5 Seleção manual de PDF por grupo

Uma vez que as páginas dos 1.980 documentos PDF já foram agrupadas utilizando o algoritmo K-means, é possível agora selecionar os próprios PDFs.

A premissa adotada é que, *como as páginas dos documentos PDF (referindo-se às 12.995 imagens) já foram agrupadas, para determinar o grupo de um documento PDF completo, basta atribuí-lo ao grupo ao qual pertence a maioria de suas páginas, independentemente do número de páginas que ele contém.*

Uma vez obtidos os documentos PDF por grupo, é possível prosseguir com a sua visualização com o objetivo de *selecionar os documentos PDF mais representativos de cada grupo*. Para isso, é necessário abrir os documentos com um software como o Adobe Acrobat para verificar, por um lado, se os elementos de um grupo fazem sentido juntos (agrupamento intra-grupo) e, por outro lado, se eles diferem dos elementos dos outros grupos (agrupamento inter-grupo).

Essa etapa de visualização antes da seleção é fundamental para:

1. **Verificação da consistência intra-grupo**: garantir que os documentos em cada grupo compartilhem características semelhantes e que tenham sido agrupados corretamente.

2. **Analisar as diferenças inter-grupos**: identificar as características distintas entre os diferentes grupos para entender os critérios de diferenciação usados, que vão desde a extração dos embeddings com o ResNet-152 até o agrupamento usando o algoritmo K-means. Esses critérios podem incluir aspectos como o conteúdo textual, o layout das páginas e outros metadados dos documentos.

Em outras palavras, essa análise garante que os grupos formados sejam internamente consistentes e distintos uns dos outros em termos de conteúdo e formato.

3.1.1.6 Anotação de PDFs selecionados com o software VoTT (Visual Object Tagging Tool)

Após selecionar os documentos PDF mais representativos de cada grupo, foi utilizado o software VoTT (Visual Object Tagging Tool) para anotar seus conteúdos, ou seja, as páginas desses documentos PDF, que são imagens devido à sua conversão prévia.

O VoTT[1] é uma ferramenta de anotação open source popular publicada pela Microsoft. Ele permite rotular (com rótulos pré-definidos pelo usuário) facilmente imagens para uso com modelos de detecção de objetos. Em nosso caso, os rótulos (classes ou categorias) que usamos são: *texto, título, lista, Tabela e imagem.* A saída dele (VoTT) é composta de:

- Um arquivo JSON (JavaScript Object Notation): que contém anotações para todas as imagens anotadas e
- As imagens correspondentes

a. **Conteúdo do arquivo JSON**:

   Considerando uma imagem I_i (referente à página do documento PDF que está sendo anotado), com $i \in [1, \ldots, \text{imagem\_total\_para\_anotar} - 1]$, e conforme mencionado, usamos o VoTT para criar 5 rótulos: *texto, título, lista, Tabela e imagem.*

   Dependendo da combinação (referente ao layout da página do documento PDF que está sendo anotado) de elementos presentes em cada página, delimitamos as regiões correspondentes. Essas delimitações são mais conhecidas como *caixas delimitadoras (bounding boxes)*. À medida que essas delimitações são feitas, o arquivo JSON é atualizado automaticamente com as chaves *rótulos (tags)* e *activos (assets)*.

   - **rótulos**: Essa chave refere-se os rótulos criados para as várias classes de anotação.
   - **activos**: Essa chave fornece acesso a todas as informações relacionadas às anotações das imagens.

   A Figura 3.3 mostra o fluxo de criação do arquivo JSON no software VoTT.

[1] `https://github.com/microsoft/VoTT`

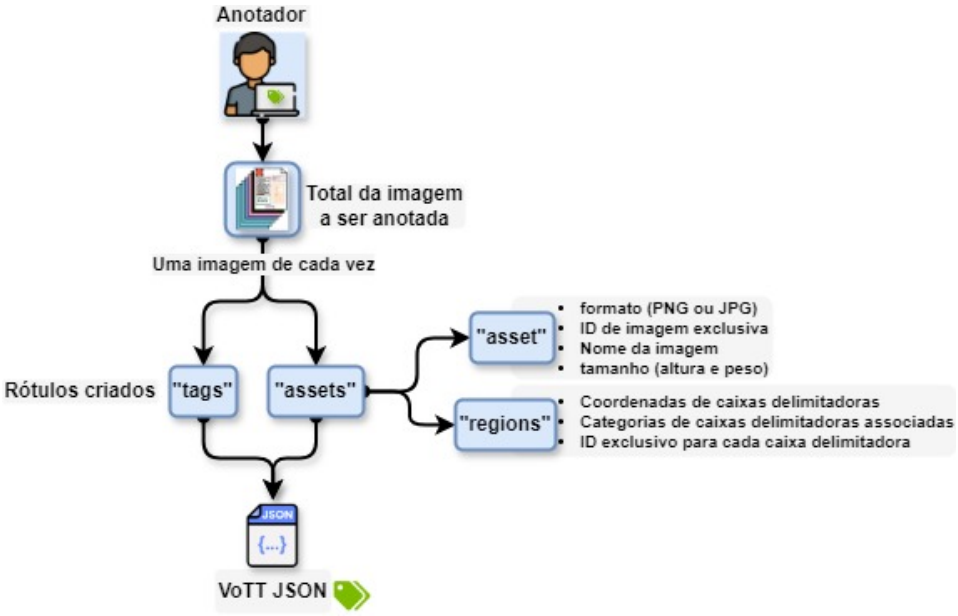

Figura 3.3: Imagem ilustrativa mostrando o fluxo de criação do arquivo JSON no software VoTT.

Na Figura 3.3, podemos observar a presença de duas outras chaves na chave *activos (assets)*: *activo (asset)* e *regiões (regions)*. Conforme mostrado na Figura 3.3, elas contêm, respectivamente, os metadados da imagem I_i e os metadados dos elementos anotados na imagem I_i.

3.1.1.7 Pós-processamento para o formato PubLayNet

Para garantir que as anotações geradas pelo VoTT (o arquivo JSON) sejam utilizáveis por um modelo de detecção (ou outro modelo), é essencial padronizá-las em um formato compatível com o modelo. Para isso, foi escolhido o formato do conjunto de dados PubLayNet [5]. O PubLayNet é um vasto conjunto de dados de imagens de artigos e documentos de pesquisa, incluindo os rótulos (classes ou categorias) *texto, título, lista, Tabela e Figura*. A Figura 3.4 ilustra o esquema do nosso código de pós-processamento.

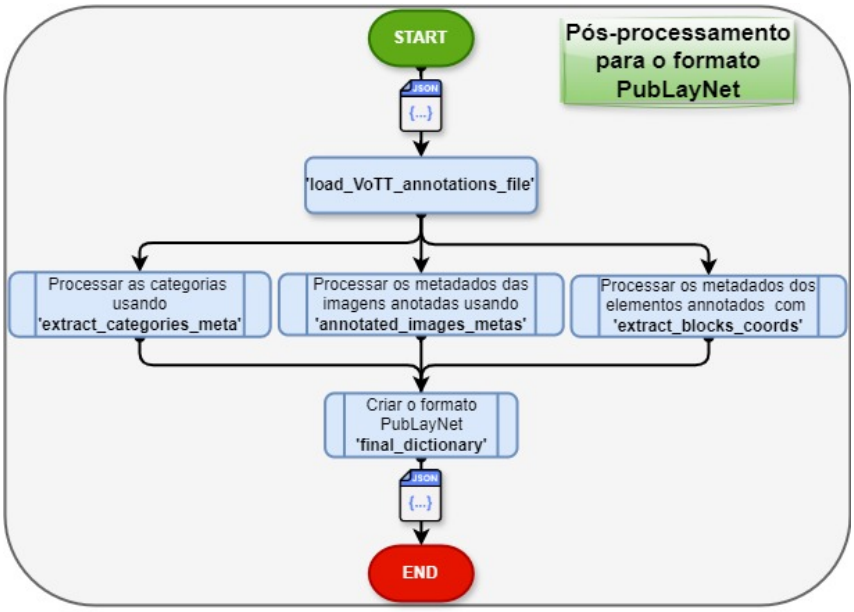

Figura 3.4: Imagem ilustrativa mostrando o esquema de nosso código para pós-processamento do arquivo JSON (saída do software VoTT) no formato PubLayNet.

O processo começa com o recebimento de um arquivo JSON contendo as anotações feitas no software VoTT. Primeiramente, o sistema carrega esse arquivo JSON utilizando a função *load_VoTT_annotations_file*. Em seguida, a função *extract_categories_meta* extrai os rótulos (classes ou categorias) definidos no arquivo. A função *annotated_images_metas* é então utilizada para obter metadados das imagens anotadas, como o *nome do arquivo*, *altura* e *largura*. Posteriormente, a função *extract_block_coords* é chamada para extrair as coordenadas das regiões anotadas em cada imagem. Por fim, o processo gera um novo arquivo JSON contendo os resultados de cada uma dessas funções, estruturando as anotações no formato PubLayNet.

3.1.2 Pipeline de geração e processamento de layout de documento

O pipeline de geração e processamento do layout do documento é um processo que começa com a criação do layout inicial usando um gerador especializado chamado de *Gerador de Layouts Sintéticos*. Em seguida, esse layout é preenchido com conteúdo usando um gerador apropriado chamado de *Gerador de Imagens de Documentos Sintéticos* [†] . Por meio de várias operações de pós-processamento, esse pipeline resulta na produção de imagens de documentos realistas e suas anotações correspondentes em um formato específico. As linhas a seguir descrevem cada parte do pipeline.

[1†]Uma contribuição dos autores.

3.1.2.1 Gerar layout de documento usando o Gerador de Layouts Sintéticos

A intuição por trás desse processo é a seguinte: *dado que a Petrobras nos forneceu apenas 1.980 documentos PDF e dada a premissa mencionada na seção de Agrupamento com o algoritmo K-means*, poderíamos usar um modelo de inteligência artificial para gerar outros tipos de formatos de documento (diferentes layouts) a partir dos documentos PDF existentes. Essa abordagem permitiria, por um lado, enriquecer a base de dados e, por outro, simular e criar formatos de documentos que não estão incluídos nos 1.980 PDFs iniciais, embora isso envolva um certo grau de incerteza que não pode ser totalmente controlado. Finalmente, essa abordagem contribuiria para tornar o modelo de detecção mais genérico e robusto.

O Gerador de Layouts Sintéticos (um nome que foi escolhido deliberadamente) foi apresentado no artigo *LayoutTransformer: Layout Generation and Completion with Self-attention* [12]. Trata-se de um modelo de machine learning baseado na arquitetura Transformer, mais especificamente na parte decodificadora (② na Figura 2.10) dessa arquitetura, como discutido na seção Background. Este modelo é capaz de gerar novos layouts a partir do zero em diversos domínios, como imagens, aplicativos móveis, documentos e objetos 3D. O modelo pode compreender as relações entre os elementos do layout e criar arranjos significativos.

O layout de um documento, também conhecido como organização do conteúdo, refere-se à estruturação e disposição dos elementos visuais (título, texto, parágrafo, tabela, etc.) em uma página. Considerando o layout de um documento em um espaço bidimensional $\mathbb{R}^2$, cada um desses elementos visuais pode ser delimitado por meio de uma caixa delimitadora.

Partindo desse princípio, [12] considera que a apresentação de um documento (seu layout) pode ser representada na forma de um grafo $\mathcal{G}$ totalmente conectado com n nós. Cada nó $i \in \{1, \ldots, n\}$ corresponde a uma primitiva gráfica, como texto, imagens ou legendas, e contém as informações geométricas $g_i = [x_i, y_i, h_i, w_i]$, onde (x, y) são as *coordenadas do centro* e (h, w) representam a altura e a *largura*. Essas primitivas podem ser aprendidas pelo modelo. Devido a esse aspecto de grafo totalmente conectado, o mecanismo de atenção tem a capacidade de aprender as relações existentes entre os nós para produzir novos layouts de documentos.

Seguindo [12], o Gerador de Layouts Sintéticos foi treinado com o conjunto de dados PubLayNet [5], visto que o modelo pré-treinado não estava

disponível.

Foi utilizado os conjuntos de treinamento, com 335.703 imagens, e os de validação, com 11.245 imagens, do conjunto de dados PubLayNet. A Figura 3.5 mostra a distribuição dos rótulos (classes ou categorias) nos conjuntos de treinamento (a) e validação (b), respectivamente.

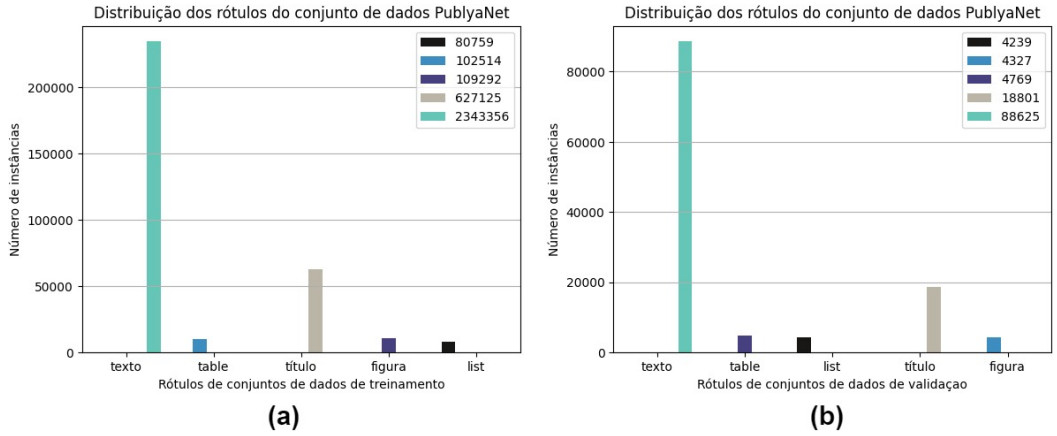

Figura 3.5: Distribuição dos rótulos nos conjuntos de treinamento e validação do conjunto de dados PubLayNet.

A Tabela 3.4 apresenta os hiperparâmetros do modelo utilizados em nossa experiência com o conjunto de dados PubLayNet, seguindo [12].

ID	Parâmetros	Valores	
1	Épocas	20	
2	Tamanho do lote	64	
3	Taxa de aprendizado	4.5×10^{-6}	
4	Otimizador	Adam	$\beta\_1 = 0.9$
			$\beta\_2 = 0.99$
5	Número autoatenção multi-cabeça	8	
6	Dimensão dos vetores de entrada (d)	512	
7	Número de camadas no decodificador do Transformer (L)	6	
8	Número de GPUs	8 Tesla V100-SXM2-32G	

Tabela 3.4: Hiperparâmetro utilizado durante o treinamento do Gerador de Layouts Sintéticos

O modelo pré-treinado que apresentou o menor erro de validação durante o treinamento no conjunto de dados PubLayNet, juntamente com as anotações dos documentos da Petrobras convertidas para o formato PubLayNet por meio do pós-processamento (Pós-processamento para o formato PubLayNet), foi utilizado para gerar a quantidade desejada de novos layouts de documentos.

Esses novos layouts de documentos foram transformados em dicionários e, em seguida, convertidos em arquivos JSON. Cada dicionário contém o nome

do layout de documento criado, bem como as posições relativas (coordenadas) de cada primitiva no espaço bidimensional $\mathbb{R}^2$ e os rótulos associados.

3.1.2.2 Pós-Processamento para remoção de sobreposições

Olhando mais de perto e de forma resumida, o Gerador de Layouts Sintéticos é um modelo de machine learning que prevê novos formatos de layout de documentos, utilizando caixas delimitadoras para propor o posicionamento de elementos como texto, título, lista, tabela e imagem. No entanto, em algumas ocasiões, ele pode gerar layouts onde as caixas delimitadoras se sobrepõem. A Figura 3.6 mostra duas amostras de layouts de documentos gerados: (a) um layout com caixas sobrepostas e (b) um layout sem sobreposição de caixas.

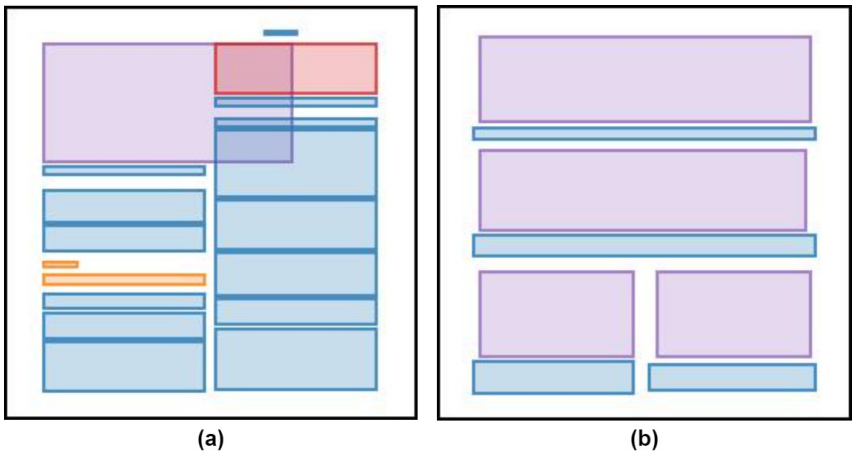

Figura 3.6: Imagem ilustrativa mostrando dois layouts de documentos gerados: um com sobreposição de caixas e outro sem sobreposição.

Para resolver os casos em que há sobreposições nos layouts, é necessário aplicar um pós-processamento adequado. Para isso, foram testadas duas abordagens diferentes:

1. Exclusão de caixas delimitadoras sobrepostas: essa abordagem não se mostrou vantajosa, pois como a ideia do Gerador de Layouts Sintéticos é criar novos layouts de documentos, a exclusão de caixas delimitadoras sobrepostas levaria a uma perda de informações. A Figura 3.7 mostra o resultado dessa abordagem.

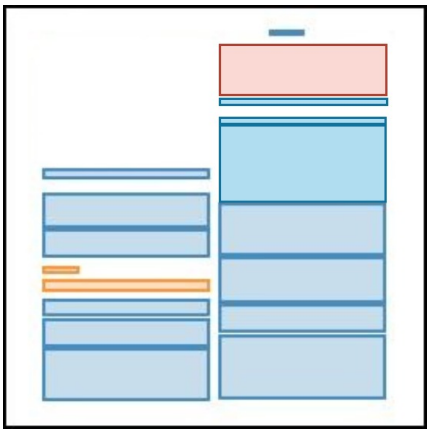

Figura 3.7: Imagem ilustrativa mostrando a exclusão da caixa delimitadora: Retângulo roxo nesse caso.

2. Verificação do tipo de sobreposição entre as caixas delimitadoras sobrepostas e eliminação da informação da sobreposição entre elas.

Abordagens de resolução de Sobreposição de Caixas delimitadoras sobrepostas :

Considerando um layout L no espaço bidimensional $\mathbb{R}^2$, gerado pelo Gerador de Layouts Sintéticos e composto por B caixas delimitadoras, é possível distinguir diferentes tipos de sobreposições entre duas caixas delimitadoras b_1 e $b_2 \in B$, como segue:

- b_1 parcialmente sobrepondo b_2:
  - no lado superior esquerdo (1);
  - no lado inferior esquerdo (2);
  - no lado superior direito (3);
  - no lado inferior direito (4);
  - no lado superior (5);
  - no lado inferior (6);
  - no lado esquerdo (7);
  - no lado direito (8);
- b_1 completamente sobrepondo b_2 (9);
- b_2 completamente sobrepondo b_1 (10).

b_1 e b_2 são caracterizadas respectivamente por suas coordenadas (x_0, y_0) representando as coordenadas do canto inferior esquerdo, e (x_1, y_1) representando as coordenadas do canto superior direito, como ilustrado na Figura 3.8.

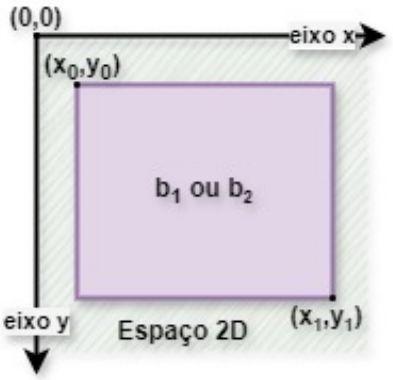

Figura 3.8: Imagem ilustrativa mostrando b_1 e b_2 no espaço 2D. Vale lembrar que, no caso desta Figura, o termo 'b_1 ou b_2' é utilizado apenas para indicar que b_1 ou b_2 sempre terão o mesmo posicionamento das coordenadas num espaço 2D. Além disso, quando se trata de imagens, as coordenadas começam da esquerda para a direita e de cima para baixo.

Eliminação da informação de sobreposição:

Consideremos a Figura 3.9 (a título explicativo) com a caixa delimitadora b_1 parcialmente sobrepondo a caixa delimitadora b_2 no lado superior esquerdo (1).

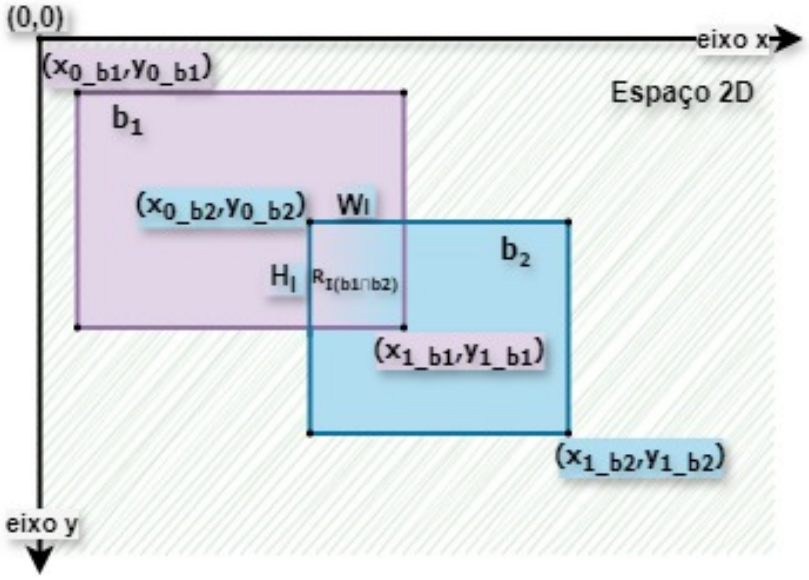

Figura 3.9: Imagem ilustrativa mostrando a caixa delimitadora b_1 sobrepondo a caixa delimitadora b_2 no lado superior esquerdo.

Denotemos $\mathcal{R}_{\mathcal{I}(b_1,b_2)}$ a região de interseção entre as duas caixas delimitadoras b_1 e b_2, e $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$ a área da superfície dessa região.
Para eliminar a sobreposição entre as duas caixas delimitadoras, ou seja, a região de interseção $\mathcal{R}_{\mathcal{I}(b_1,b_2)}$, nossa primeira etapa foi calcular a área da superfície, $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$.

Isso foi feito determinando primeiro as coordenadas da região de interseção, $\mathcal{R}_{\mathcal{I}(b_1,b_2)}$, ou seja, aquelas do canto inferior esquerdo e aquelas do canto superior direito, representadas respectivamente pelos pares $(x_{0\mathcal{R}_\mathcal{I}}, y_{0\mathcal{R}_\mathcal{I}})$ e $(x_{1\mathcal{R}_\mathcal{I}}, y_{1\mathcal{R}_\mathcal{I}})$. Essas coordenadas são dadas pelas seguintes fórmulas:

$$(x_{0\mathcal{R}_\mathcal{I}}, y_{0\mathcal{R}_\mathcal{I}}) = (\max(x_{0b_1}, x_{0b_2}), \max(y_{0b_1}, y_{0b_2})); \tag{3-1}$$

$$(x_{1\mathcal{R}_\mathcal{I}}, y_{1\mathcal{R}_\mathcal{I}}) = (\min(x_{1b_1}, x_{1b_2}), \min(y_{1b_1}, y_{1b_2})), \tag{3-2}$$

onde os pares (x_{0b_1}, y_{0b_1}), (x_{1b_1}, y_{1b_1}), (x_{0b_2}, y_{0b_2}) e (x_{1b_2}, y_{1b_2}) representam respectivamente as coordenadas dos cantos inferior esquerdo e superior direito das caixas delimitadoras b_1 e b_2.

Uma vez obtidas essas coordenadas, o cálculo da área da superfície de interseção $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$ é feito pela seguinte fórmula:

$$\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)} = \mathcal{H}_\mathcal{I} \times \mathcal{W}_\mathcal{I} = (y_{1\mathcal{R}_\mathcal{I}} - y_{0\mathcal{R}_\mathcal{I}}) \times (x_{1\mathcal{R}_\mathcal{I}} - x_{0\mathcal{R}_\mathcal{I}}), \tag{3-3}$$

onde $\mathcal{H}_\mathcal{I}$ e $\mathcal{W}_\mathcal{I}$ representam respectivamente a altura e a largura da região de interseção $\mathcal{R}_{\mathcal{I}(b_1,b_2)}$.

Com a área da superfície de interseção $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$ obtida, nossa segunda etapa foi calcular respectivamente as áreas das superfícies das caixas delimitadoras b_1 e b_2, denotadas $\mathcal{A}_{b_1}$ e $\mathcal{A}_{b_2}$. Elas são dadas pelas seguintes fórmulas:

$$\mathcal{A}_{b_1} = (x_{1b_1} - x_{0b_1}) \times (y_{1b_1} - y_{0b_1}) \tag{3-4}$$

$$\mathcal{A}_{b_2} = (x_{1b_2} - x_{0b_2}) \times (y_{1b_2} - y_{0b_2}) \tag{3-5}$$

$\mathcal{A}_{b_1}$ e $\mathcal{A}_{b_2}$ sendo obtidas, decidimos subtrair a área da superfície de interseção $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$ da caixa delimitadora que tem a maior área entre as duas. Em outras palavras, subtraímos a região de interseção $\mathcal{R}_{\mathcal{I}(b_1,b_2)}$ da caixa delimitadora com a maior área. Essa subtração foi feita comparando primeiro a altura $\mathcal{H}_\mathcal{I}$ e a largura $\mathcal{W}_\mathcal{I}$ da região de interseção $\mathcal{R}_{\mathcal{I}(b_1,b_2)}$ para tomar o menor valor entre os dois. A seleção do menor valor nos permite, em primeiro lugar, minimizar a perda de informação e, em seguida, determinar em que direção realizar a subtração de $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$ da maior área entre $\mathcal{A}_{b_1}$ e $\mathcal{A}_{b_2}$ (ou seja, subtração em abscissa *eixo x* ou em ordenada *eixo* y).

No caso da Figura 3.8, $\mathcal{A}_{b_1}$ é a caixa delimitadora com a maior área, e $\mathcal{W}_\mathcal{I}$ é menor que $\mathcal{H}_\mathcal{I}$, o que resulta em uma subtração na abscissa da distância $\mathcal{W}_\mathcal{I}$. A Figura 3.10 apresenta o resultado obtido após essa subtração.

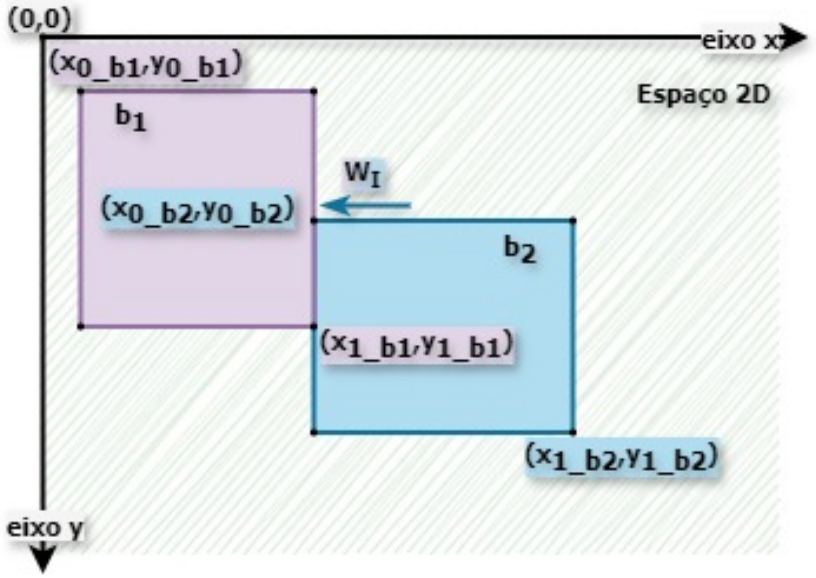

Figura 3.10: Imagem ilustrativa mostrando a subtração na abscissa da área da superfície de interseção $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$ da caixa delimitadora $\mathcal{A}_{b_1}$.

Como podemos ver na Figura 3.10, mesmo após a remoção da sobreposição entre as duas caixas b_1 e b_2, ainda resta uma sobreposição parcial. Para uma separação definitiva, subtraímos uma pequena distância aleatória denotada por Δ, tomando os valores Δ_x e Δ_y quando a subtração da região de interseção $\mathcal{R}_{\mathcal{I}(b_1,b_2)}$ é feita respectivamente em abscissa (*eixo x*) ou em ordenada (*eixo y*).

A Figura 3.11 apresenta o resultado após a subtração da distância aleatória em abscissa, ou seja, Δ_x.

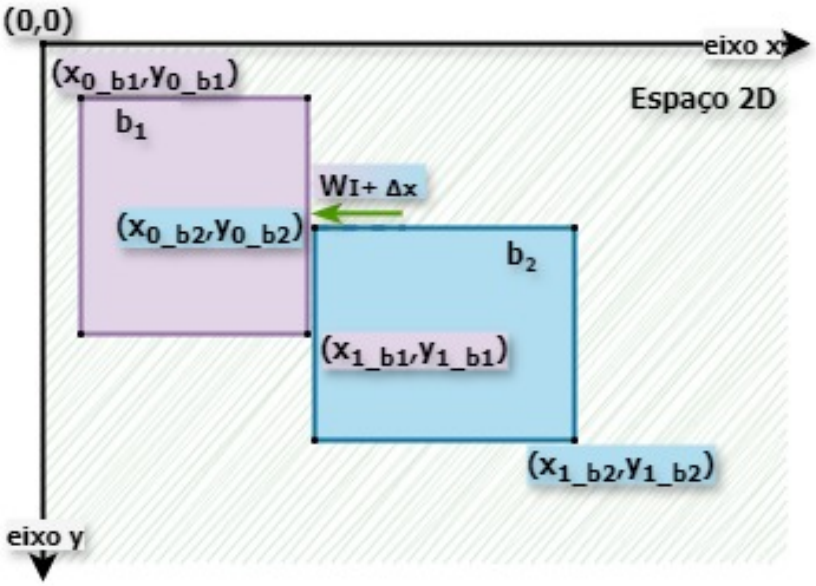

Figura 3.11: Imagem ilustrativa mostrando a remoção da distância Δ_x. Isso equivale a retirar a área da superfície de interseção $\mathcal{A}_{\mathcal{I}(b_1 \cap b_2)}$ de uma distância $\mathcal{W}_{\mathcal{I}} + \Delta_x$ da área do retângulo $\mathcal{A}_{b_1}$.

Em alguns casos, quando a altura $\mathcal{H}_{\mathcal{I}}$ e a largura $\mathcal{W}_{\mathcal{I}}$ da região de interseção $\mathcal{R}_{\mathcal{I}(b_1,b_2)}$ são iguais, removemos a informação de sobreposição, seja em abscissa ou em ordenada, dependendo do tipo de sobreposição entre as duas caixas, da caixa delimitadora com a área maior.

Eliminação da informação de sobreposição nos casos (9, 10) :

Considere a Figura 3.12 (para fins explicativos), onde a caixa delimitadora b_1 sobrepõe completamente a caixa delimitadora b_2 (8).

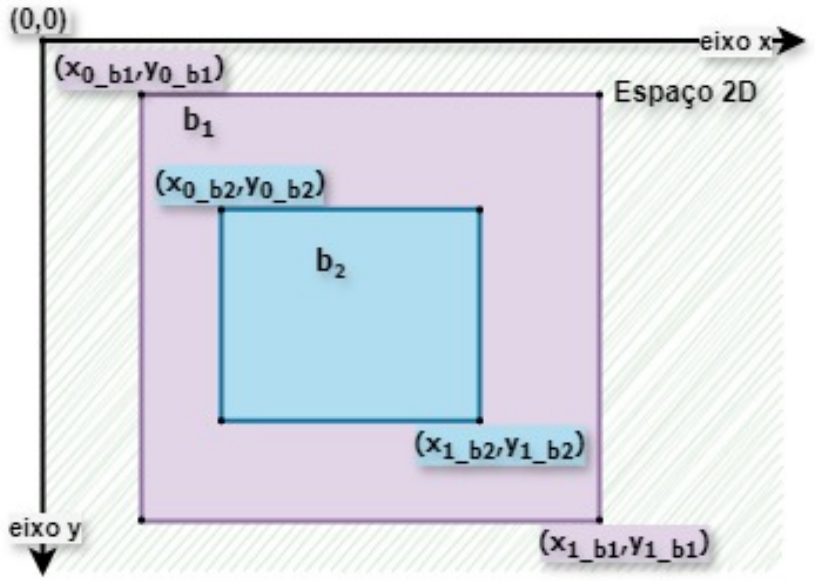

Figura 3.12: Imagem ilustrativa mostrando a caixa delimitadora b_1 completamente sobrepondo b_2.

Para eliminar a informação de sobreposição neste caso, primeiro foi verificada qual superfície tinha a maior área entre $\mathcal{A}_{b_1}$ e $\mathcal{A}_{b_2}$. Em seguida, simplesmente foi removida a caixa delimitadora com a menor área (no caso da Figura 3.12, trata-se de b_2).

A Figura 3.13 apresenta o esquema do nosso código de pós-processamento para remoção de sobreposição. Este código é capaz de lidar com todos os tipos de sobreposições que listamos (de 1 a 10).

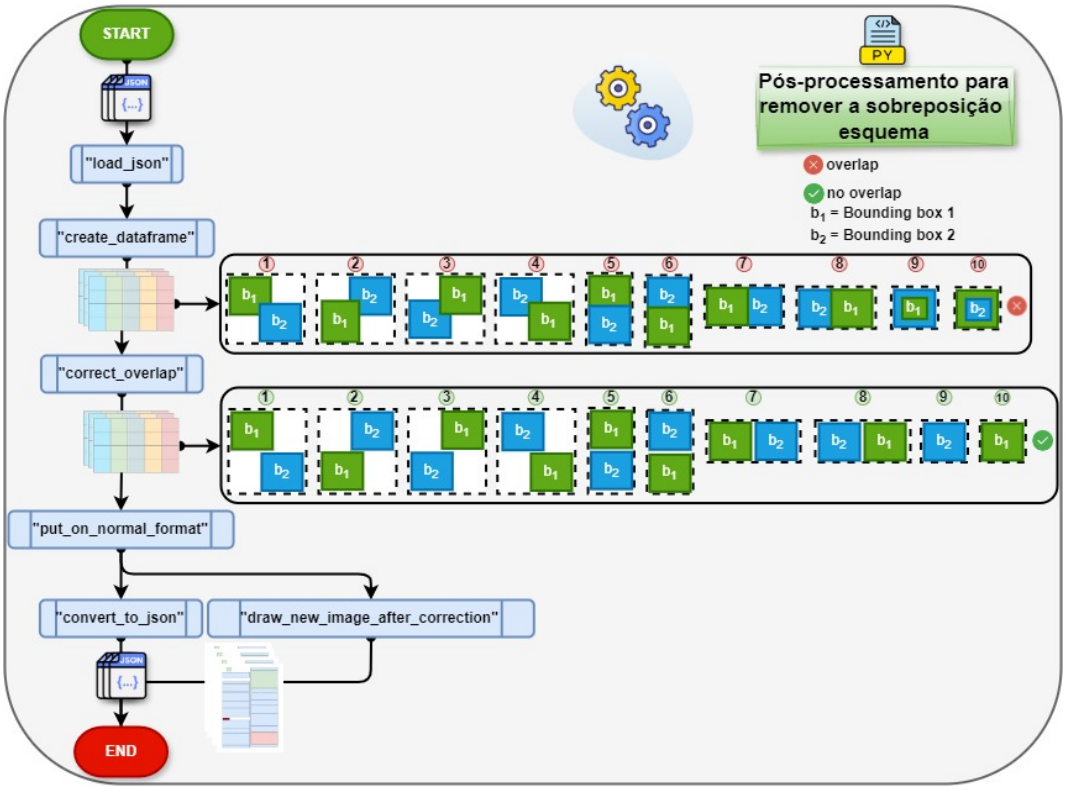

Figura 3.13: Imagem ilustrativa mostrando o esquema do nosso código de pós-processamento para remover a sobreposição.

O processo começa com a recepção de uma lista de arquivos JSON representando as disposições de documentos geradas pelo Gerador de Layout Sintético. Cada arquivo JSON contém o nome do arquivo, as coordenadas das caixas delimitadoras associadas, bem como seus respectivos rótulos. A função *load_json* carrega esses arquivos e retorna uma lista de dicionários, onde cada dicionário contém o nome do arquivo, as coordenadas das caixas delimitadoras associadas, bem como seus respectivos rótulos.

A função *create_dataframe* transforma essa lista de dicionários em uma lista de DataFrames, onde cada DataFrame representa uma tabela que contém o nome do arquivo em processamento, as coordenadas das caixas delimitadoras associadas, bem como seus respectivos rótulos.

A função *correct_overlap* analisa os DataFrames para identificar e corrigir as sobreposições entre as caixas delimitadoras. A função retorna uma lista de DataFrames, onde cada DataFrame contém o nome do arquivo, as coordenadas das caixas delimitadoras associadas corrigidas, bem como seus respectivos rótulos.

A função *put_on_normal_format* formata esses DataFrames corrigidos para que possam ser convertidos em arquivos JSON pela função *convert_to_json*. Finalmente, a função *draw_new_image_after_correction* de-

senha as caixas delimitadoras para verificar se as correções foram aplicadas corretamente.

3.1.2.3 Preencher conteúdo usando o Gerador de Imagens de Documentos Sintéticos

Após obter os layouts de documentos sem sobreposição de caixas delimitadoras, é possível iniciar o preenchimento dessas caixas com conteúdo, criando, assim, conjuntos de dados sintéticos compostos por imagens de documentos. A natureza do conteúdo a ser preenchido é definida pelo rótulo associado a cada caixa delimitadora no layout do documento (como texto, título, lista, tabela, imagem). Para esse fim, foi desenvolvido o Gerador de Imagens de Documentos Sintéticos.

O Gerador de Imagens de Documentos Sintéticos, como o próprio nome sugere, é um projeto concebido para criar imagens de documentos e suas respectivas anotações, com base nos layouts de documentos e outros recursos que recebe. A saída do gerador inclui uma imagem no formato PNG (Portable Network Graphics) representando o documento criado e um arquivo XML (Extensible Markup Language) detalhando suas anotações. Isso permite gerar simultaneamente milhares de imagens de documentos, que podem ser utilizadas como dados de treinamento e validação para um modelo de detecção de objetos.

A Figura 3.14 mostra a esquema do nosso Gerador de Imagens de Documentos Sintéticos.

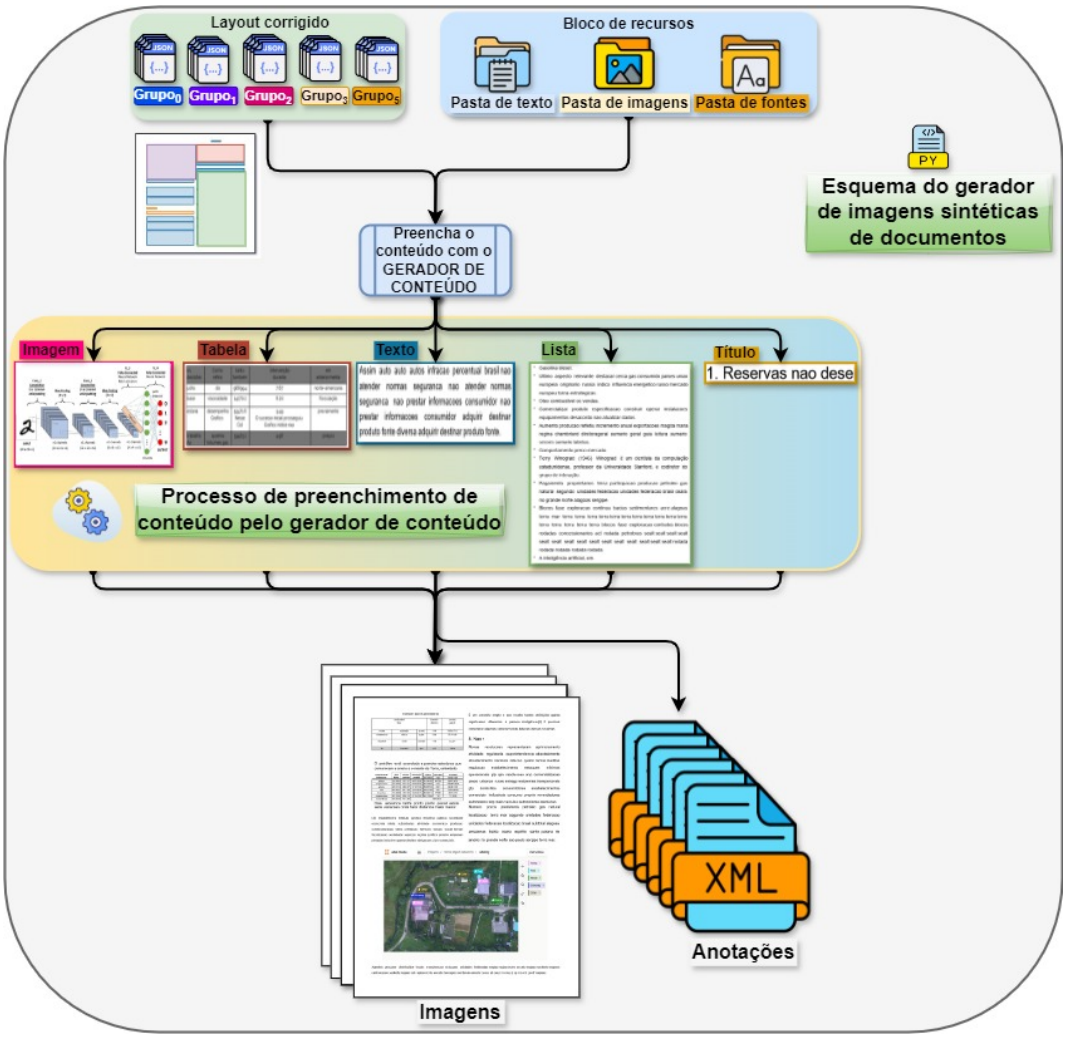

Figura 3.14: Imagem ilustrativa mostrando o esquema do Gerador de Imagens de Documentos Sintéticos.

Conforme mostrado na Figura 3.14, o Gerador de Imagens de Documentos Sintéticos recebe dois blocos como entrada. Eles são, respectivamente, o *bloco de layout do documento* corrigido pelo Pós-processamento para remoção de sobreposições e o *bloco de recursos*.

O bloco de layout de documento corrigido contém os layouts de documentos resultantes do pós-processamento para a remoção de sobreposições (arquivos JSON). O bloco de recursos, por sua vez, contém três tipos de pastas:

– Pasta de texto: Contém textos em português ou inglês, armazenados em arquivos .txt. Esses textos podem ter natureza informativa ou não informativa, ou seja, sua fonte pode variar amplamente,

– Pasta de imagens: Contém imagens nos formatos PNG ou JPEG e de diferentes tamanhos,

– Pasta de fontes: Contém diferentes tipos de fontes, como Arial (itálico, negrito, regular, etc.), Times (itálico, negrito, regular, etc.), etc

Ao receber um layout de documento caracterizada por um nome, as coordenadas das caixas delimitadoras associadas, bem como os rótulos correspondentes e o bloco de recursos, o Gerador de Imagens de Documentos Sintéticos posiciona o conteúdo adequado conforme o rótulo da caixa delimitadora.

– Para um rótulo de caixa delimitadora do tipo texto: ele pode alinhar o texto de forma justificada, à direita, à esquerda ou centralizada, respeitando certas regras como: começar o texto com uma letra maiúscula, manter o espaçamento entre as linhas e terminar cada frase com um ponto final.
– Para um rótulo de caixa delimitadora do tipo título: ele pode iniciar o título com um número, seguido do título (pode ser totalmente em maiúsculas ou começar com uma letra maiúscula).
– Para um rótulo de caixa delimitadora do tipo lista: ele pode iniciar a lista com marcadores do tipo "$*$", $-$" , "$\bullet$", "[1]" ou "{1}".
– Para um rótulo de caixa delimitadora do tipo tabela: ele pode desenhar a tabela e adicionar conteúdo em cada coluna.
– Para um rótulo de caixa delimitadora do tipo imagem: ele pode selecionar aleatoriamente uma imagem no formato JPG ou PNG de qualquer tamanho na pasta (bloco de recursos) de imagens e redimensionar essa imagem para que ela se ajuste à caixa delimitadora.

Considerando outros hiperparâmetros, como o tamanho da imagem de documento a ser gerado (por exemplo, 1200×1600), a cor de fundo da página, o tamanho das letras, etc., obtemos imagens de documentos sinéticos.

3.1.2.4 Pós-processamento para o formato COCO

Como destacamos anteriormente o Gerador de Imagens de Documentos Sintéticos não apenas cria imagens de documentos, mas também suas anotações no formato XML respectivas.

A utilização de um arquivo XML oferece uma solução estruturada e flexível para a gestão e o tratamento da informações do documento. Isso facilita não apenas a leitura e a interpretação das anotações, mas também a sua integração com outros sistemas e ferramentas de processamento de dados. Além disso, considerando que a principal função das imagens de documentos sintéticos criados é serem usados como conjuntos de treinamento e validação

para um modelo de detecção de objetos, é crucial que as anotações sigam um formato padronizado. Esse formato deve ser compatível com os requisitos de entrada do modelo, garantindo assim uma utilização eficaz dos dados anotados para o treinamento e a validação do modelo.

É aqui que entra o formato COCO (Common Objects in Context) [57]. COCO, sendo um conjunto de dados especializado na detecção, segmentação e legendagem de objetos, possui seu próprio formato (basicamente um arquivo JSON que fornece informações sobre as posições relativas dos elementos presentes nas imagens) amplamente utilizado e reconhecido. Portanto, as anotações de nossos documentos sintéticos (arquivos XML, neste caso) devem ser convertidas para esse formato para garantir compatibilidade ideal com as ferramentas e modelos de detecção de objetos comumente utilizados. Assim, a boa estruturação inicial de nossas anotações em XML permite uma conversão fácil para o formato COCO.

Na Figura 3.15, é apresentado o esquema do nosso codigo do pós-processamento para o formato COCO. Sua principal função é preparar nossas anotações para que possam ser usadas por um modelo de detecção.

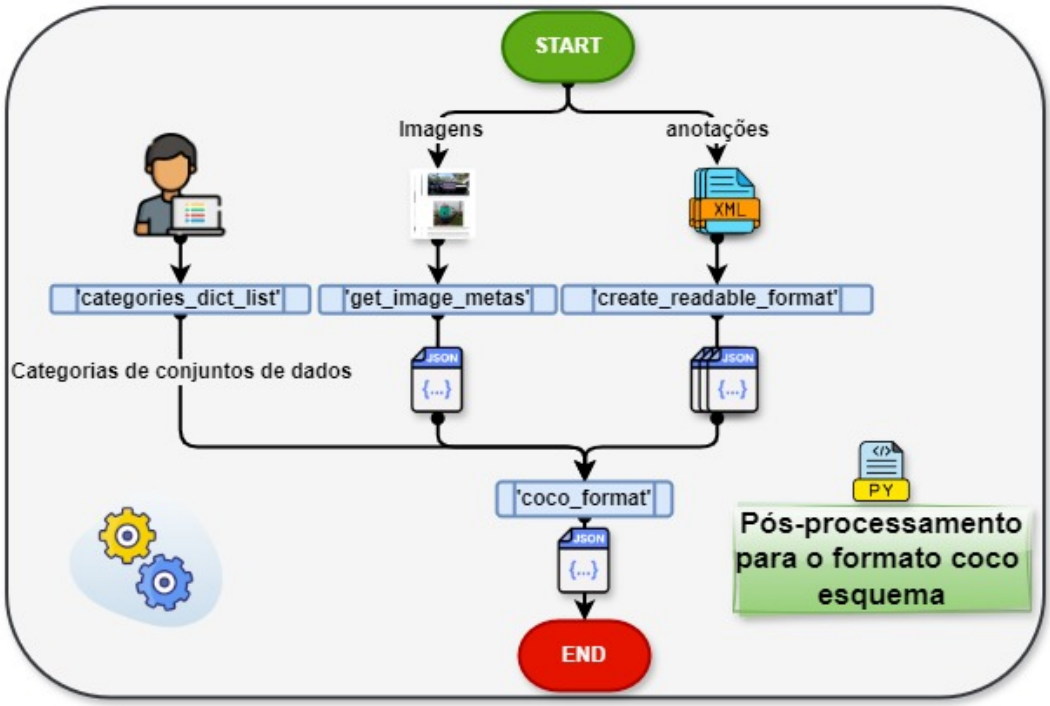

Figura 3.15: Imagem illustrativa mostrando a esquema do nosso código de pós-processamento para o formato COCO.

O processo começa com o recebimento das imagens e suas respectivas anotações no formato XML.

A função *get_image_metas* é usada para extrair os metadados da imagem de documento, como o nome, a altura, a largura e o identificador único.

A função *create_readable_format* converte anotações do formato XML para o formato JSON, extraindo fielmente os metadados que elas contêm, ou seja, as coordenadas precisas de elementos como texto, título, lista, tabela e imagem, sua área de superfície, bem como as coordenadas de segmentação.

Em seguida, um dicionário de categorias *categories_dict_list* e os resultados dos dois processos anteriores são passados para a função *coco_format*, que os converte para o formato COCO. Ao final do processo, obtemos um arquivo JSON que representa as anotações das imagens no formato COCO [57].

3.2 Metodologia de treinamento, validação, teste dos modelos e seleção de modelo

Introdução parcial

Nesta seção, abordamos a metodologia utilizada para o treinamento, validação e teste de modelos, bem como o processo de seleção do modelo mais adequado. O objetivo principal é garantir que os modelos usados apresentem um desempenho robusto e confiável ao serem aplicados a novos dados. Inicialmente, discutiremos as estratégias de divisão dos conjuntos de dados em subconjuntos de treinamento, validação e teste. Em seguida, exploraremos as técnicas de otimização de hiperparâmetros, que são cruciais para maximizar a precisão e a eficiência dos modelos. Por fim, apresentaremos os critérios de avaliação utilizados para comparar diferentes modelos e justificar a escolha do modelo final. A Figura 3.16 mostra o nosso pipeline para o treinamento, validação e teste de modelos, bem como o processo de seleção do modelo. Nas linhas a seguir descreveremos cada etapa deste último.

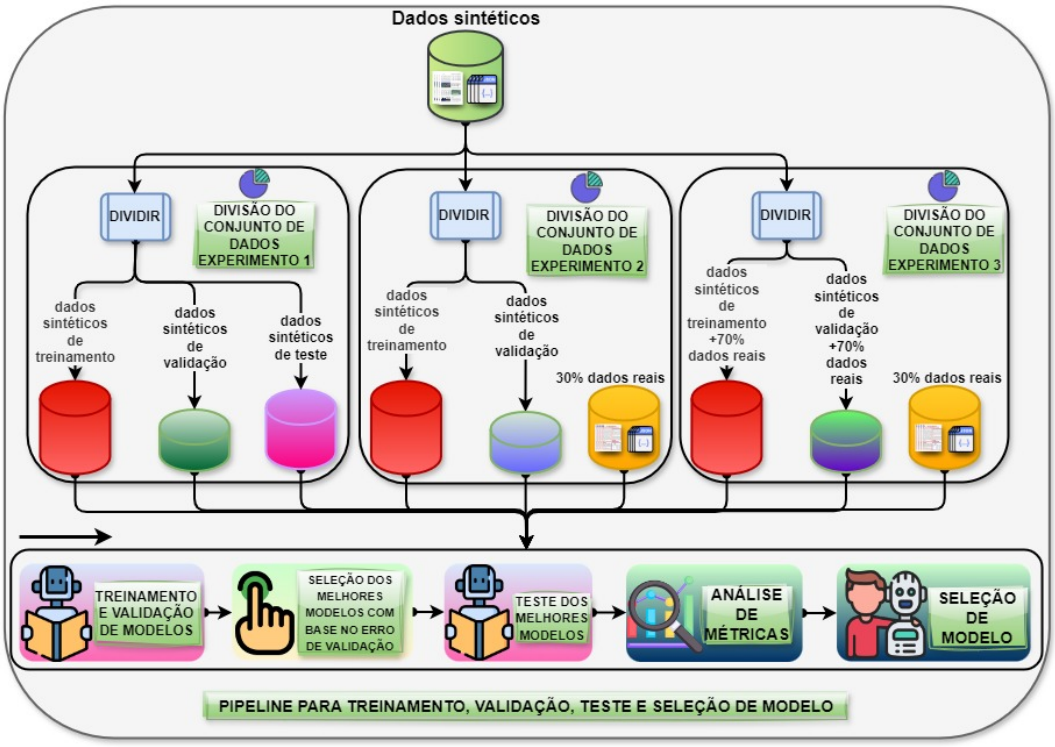

Figura 3.16: Imagem ilustrativa mostrando o nosso pipeline de treinamento, validação, teste dos modelos e seleção de modelo.

Os dados sintéticos gerados pelo pipeline de criação de conjuntos de dados de documentos sintéticos são, em seguida, divididos de diferentes maneiras utilizando o bloco *Dividir*, conforme as necessidades específicas de cada experimento. O termo *dados reais*, na Figura 3.16, refere-se aos documentos reais que foram anotados. Graças ao Pós-processamento para o formato COCO nos conseguimos esse tipo de dado inclui tanto as imagens reais anotadas quanto suas respectivas anotações no formato COCO (consulte a Figura 3.1).

3.2.1 Separação em dados de treinamento, validação e teste

Para treinar um modelo de inteligência artificial de maneira eficaz (neste caso, um modelo de detecção de objetos), é prática comum dividir o conjunto de dados em subconjuntos de treinamento, validação e teste. O objetivo do conjunto de treinamento é instruir o modelo, permitindo que ele aprenda os padrões e características dos dados. O conjunto de validação é usado para ajustar os hiperparâmetros e evitar o sobreajuste, garantindo que o modelo generalize bem para novos dados. Por último, o conjunto de teste é utilizado para avaliar o desempenho final do modelo, fornecendo uma estimativa precisa de como ele se comportará em situações reais.

Com base nesse princípio e seguindo essa lógica, lembrando que a saída do pipeline de criação de conjuntos de dados de documentos sintéticos (Figura 3.1) consiste em imagens de documentos e suas respectivas anotações no formato COCO, realizamos três tipos de experiências de separação de dados:

- **Separação Totalmente Sintética:** Separação do conjunto de dados sintético em dados de treinamento, validação e teste.
- **Teste com Dados Reais Parcial:** Separação do conjunto de dados sintético em dados de treinamento e validação, e utilização de 30% do conjunto de dados real como conjunto de teste.
- **Combinação Híbrida:** Adição de 70% do conjunto de dados real no conjunto de dados sintético e separação em dados de treinamento e validação, com utilização de 30% do conjunto de dados real como conjunto de teste (refere-se aos mesmos 30% do Teste de Dados Reais Parcial).

3.2.1.1 Separação Totalmente Sintética

O objetivo principal deste experimento foi avaliar a capacidade do modelo de aprender a partir de dados inteiramente sintéticos. Esta abordagem visava entender como o modelo se comportava sem a influência dos dados reais, permitindo assim avaliar a eficácia do conjunto de dados gerado pelo nosso pipeline.

Para isso, dividimos nosso conjunto de dados sintético (obtido do pipeline de criação de conjuntos de dados de documentos sintéticos) em 67% de dados de treinamento, 21% de validação e 12% de teste[2]. Este tipo de separação, totalizando 100%, visa maximizar o aprendizado, fornecendo dados suficientes para validação e teste.

3.2.1.2 Teste com Dados Reais Parcial

O objetivo principal desta experiência é avaliar como o modelo, treinado exclusivamente com dados sintéticos, se comporta quando aplicado a dados reais, a fim de estimar a similaridade entre os dois conjuntos. Para isso, inicialmente separamos nosso conjunto de dados sintéticos em 70% para treinamento e 30% para validação. Em seguida, utilizamos 30% dos dados reais como conjunto de teste, com o objetivo de obter uma estimativa mais precisa do desempenho do modelo em condições reais.

[2]Não confundir com uma separação padrão; este tipo de distribuição foi escolhido especificamente para este experimento.

3.2.1.3 Combinação Híbrida

O objetivo aqui foi entender o impacto da integração de dados reais no conjunto de dados sintéticos sobre a avaliação do modelo. Para isso, adicionamos 70% dos dados reais aos dados sintéticos. Em seguida, realizamos uma divisão dessas dados em 70% para treinamento e 30% para validação. Na fase de teste, assim como na experiência 2, utilizamos os 30% dos dados reais. Outro aspecto desta experiência é avaliar o nosso pipeline de uma perspectiva operacional, simulando um contexto de produção.

3.2.2 Processamento de treinamento e validação dos modelos

Como se trata de uma tarefa de detecção de objetos, foram selecionados três modelos de ponta (State-of-the-Art), cada um dedicado a essa tarefa específica. Os modelos escolhidos são: Mask R-CNN [60], que é baseado em redes convolutivas; Mask DINO [**?**], que utiliza uma arquitetura do tipo Transformer; e RT-DETR [51], que também se baseia na arquitetura Transformer. Cada um desses modelos foi utilizado nos diferentes experimentos mencionados anteriormente e seguindo a Figura 3.16.

Uma abordagem que usamos para treinar os modelos selecionados com nosso conjunto de dados de documentos, e que é familiar a todos, é a abordagem de *ajuste fino (fine tuning)*. Essa técnica, comum no aprendizado de máquina, consiste em adaptar um modelo pré-treinado a uma tarefa específica, ajustando seus parâmetros com base em um novo conjunto de dados.

As Tabelas 7.1, 7.2 e 7.3, disponíveis na seção Apêndice, apresentam os principais hiperparâmetros utilizados pelos modelos em nossos experimentos.

3.2.3 Processamento de teste dos modelos

Após a fase de treinamento e validação, vem a fase de teste, onde o objetivo é utilizar o conhecimento adquirido pelo modelo para avaliá-lo nos dados de teste. Para isso, em cada caso de experiência, selecionamos os melhores modelos com base no menor erro de validação. Em outras palavras, para cada experiência envolvendo três modelos, escolhemos para cada um aquele que apresenta o menor erro de validação para realizar o teste nos dados de teste.

3.2.4 Seleção de Modelo

A seleção de modelo consistiu, para cada experiência, em escolher, entre os três modelos testados nos dados de teste, aquele que apresentou os melhores resultados. Isso foi feito através da avaliação das métricas de desempenho.

Os modelos de detecção de objetos são geralmente avaliados usando a métrica de *Precisão Média (mean Average precisão, mAP)*. Esta métrica é baseada em cálculos de *precisão* e *recall*, proporcionando uma compreensão abrangente do desempenho do modelo, especialmente em conjuntos de dados altamente assimétricos [76].

A precisão refere-se à proporção de verdadeiros positivos (VP) em relação ao número total de predições positivas feitas pelo modelo. Uma alta precisão indica que a maioria das detecções positivas feitas pelo modelo são corretas [77]. A fórmula da precisão é dada por:

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{3-6}$$

O recall, por outro lado, refere-se à proporção de verdadeiros positivos previstos pelo modelo em relação ao número total de elementos positivos reais no conjunto de dados. Um alto recall indica que o modelo pode detectar a maioria dos verdadeiros positivos [77]. A fórmula do recall é dada por:

$$\text{Recall} = \frac{VP}{VP + FN} \tag{3-7}$$

Onde VP, FP, FN significam, respectivamente, *Verdadeiro Positivo*, *Falso Positivo* e *Falso Negativo*.

Determinar se um objeto é considerado um verdadeiro positivo ou não pelo modelo depende de um parâmetro definido como *limiar de Interseção sobre a União (Intersection Over Union Threshold)* $\in [0, 1]$, que é crucial na métrica Interseção sobre a União (Intersection over Union, IoU). A IoU é uma métrica utilizada para calcular o grau de sobreposição entre a caixa delimitadora do objeto prevista pelo modelo e a caixa delimitadora do objeto verdadeiro (anotação feita). Ela é dada pela fórmula:

$$IoU = \frac{A \cap B}{A \cup B} \tag{3-8}$$

Onde A e B representam, respectivamente, a caixa delimitadora prevista e a verdadeira do objeto. Isso significa que, se a previsão de um determinado objeto feita pelo modelo tem uma IoU acima do limiar de Intersection sobre Union, ela é considerada um Verdadeiro Positivo caso contrario, ela é considera com um Falso positivo.

Com base nesse princípio, é possível calcular facilmente as equações 3-6 e 3-7 para cada rótulo dentro do conjunto de dados. Os valores de precisão e recall podem ser usados para criar a *Curva de Precisão-Recall (PR-Curve).* Essa curva é utilizada para apresentar resultados de problemas de decisão binária, com precisão no eixo y e recall no eixo x [76]. Basicamente, é possível plotar essa curva para diferentes limiares de IoU.

No entanto, é mais vantajoso ter uma estimativa numérica que indique o quão bem o modelo desempenha. A Precisão Média (AP) é essa estimativa, onde o objetivo é encontrar a área sob a PR-Curve. Ela é dada pela fórmula:

$$AP = \int_0^1 p(r)dr \tag{3-9}$$

Onde $p(r)$ é a medição da precisão p no recall r.

O termo mAP, por sua vez, refere-se a uma estimativa global do desempenho do modelo sobre todos os rótulos existentes no conjunto de dados (no nosso caso: texto, título, lista, tabela, imagem). Ele é dado pela fórmula (onde N representa o número de rótulos):

$$mAP = \frac{1}{N} \sum_0^1 AP_i \tag{3-10}$$

Uma prática comum para reportar o mAP em tarefas de detecção de objetos é utilizar as métricas padrão do conjunto de dados COCO [57], que incluem mAP, mAP50, mAP75, mAPS, mAPM e mAPL. Essa abordagem foi adotada neste estudo, e a Tabela 3.5 apresenta uma visão geral de cada uma dessas métricas.

Métricas	Significado
mAP	mAP para limiar de IoU = [0.5:0.95] com passo de 0.05
$mAP^{IoU=.50}$	mAP para limiar de IoU de 0.50
$mAP^{IoU=.75}$	mAP para limiar de IoU de 0.75
mAP^{small}	mAP para objetos pequenos: área $< 32^2$
mAP^{medium}	mAP para objetos médios: $32^2 <$ área $< 96^2$
mAP^{large}	mAP para objetos grandes: área $\geq 96^2$

Tabela 3.5: Métricas padrão do COCO para detecção de objetos

Conclusão parcial

Nesta seção, foi discutida a metodologia adotada para o treinamento, validação e teste dos modelos, bem como o processo final de seleção do modelo, conforme ilustrado na Figura 3.16. Enfatizamos a importância da divisão dos dados em conjuntos de treinamento, validação e teste. Além disso, apresentamos os critérios de avaliação utilizados para comparar diferentes modelos e justificar a escolha do modelo final.

4
Resultados

Nesta seção, são apresentados os resultados obtidos nas diferentes etapas do estudo, desde a criação do conjunto de dados sintético, utilizando o pipeline de geração de dados, até os resultados provenientes do treinamento, validação, teste dos modelos e seleção do modelo.

4.1
Pipeline de Anotação e Processamento de PDFs

Conforme apresentado na seção Metodologia de criação de conjunto de dados, a primeira parte do nosso pipeline, ilustrada na Figura 3.1, é o Pipeline de Anotação e Processamento de PDFs, cujo objetivo é padronizar a conversão dos documentos PDF em um formato de anotações que atenda aos padrões do domínio. Envolvendo vários estágios de processamento descritos na seção Pipeline de Anotação e Processamento de PDFs, nas linhas a seguir apresentamos os resultados desses estágios.

4.1.1
Agrupamento com o Algoritmo K-means

Para determinar o valor ideal do número de grupos (clusters) K a ser utilizado pelo algoritmo K-means, o algoritmo foi testado com diferentes valores de $K \in [2, 20]$. A escolha do melhor valor foi baseada no método do cotovelo (Elbow method). O valor ideal encontrado foi $K = 6$, conforme mostrado na Figura 4.1.

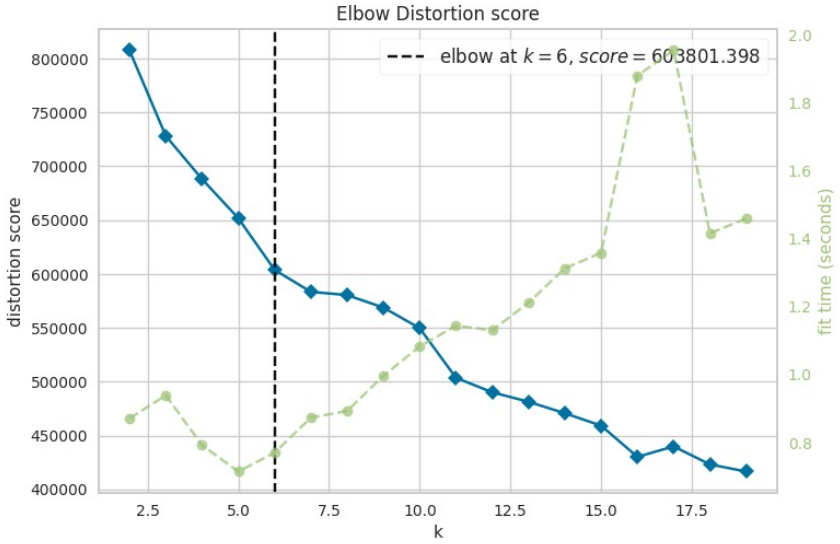

Figura 4.1: Imagem ilustrativa mostrando o resultado da escolha do número de grupos K para o algoritmo K-means pelo método do cotovelo.

4.1.2 Redução da dimensionalidade

A Figura 4.2 apresenta a distribuição das páginas dos documentos PDF (totalizando 12.995 imagens) após o agrupamento pelo algoritmo K-means e uma redução de dimensionalidade com o algoritmo t-SNE.

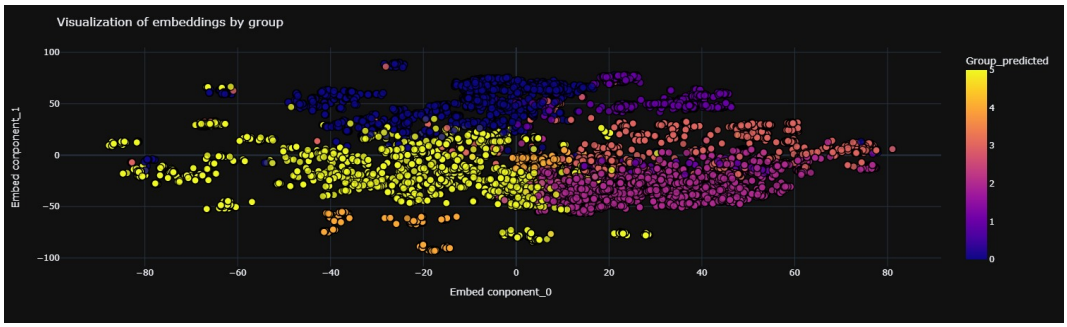

Figura 4.2: Imagem ilustrativa mostrando a distribuição das páginas dos nossos documentos PDF após o agrupamento pelo algoritmo e uma redução de dimensionalidade com o algoritmo t-SNE. As cores representam os grupos.

4.1.3 Seleção Manual de PDFs por grupo

A Tabela 4.1 apresenta a distribuição dos 1.980 documentos PDF iniciais por grupo após a fase de agrupamento com o algoritmo K-means. Uma explicação detalhada foi apresentada na seção Seleção manual de PDFs por grupo.

Grupos	**Número de documentos por grupo**
0	678
1	237
2	677
3	71
4	2
5	315
Total	**1980**

Tabela 4.1: Distribuição dos documentos por grupo

A Tabela 4.2 apresenta a distribuição dos documentos PDF representativos resultantes da seleção manual que realizamos. Ela inclui o número mínimo e máximo de páginas, a média de páginas por documento, e o número total de imagens em cada grupo.

Grupos	**Número de documentos representativos**	**Número mínimo de páginas**	**Número Máximo de páginas**	**Média de página**	**Número total de Imagem**
0	22	1	58	15	335
1	13	1	42	7	87
2	70	1	110	8	582
3	14	1	97	11	160
4	0	0	0	0	0
5	54	1	67	13	706
Total	173	-	-	-	1.870

Tabela 4.2: Distribuição dos documentos PDF representativos selecionados manualmente por grupo.

4.1.4 Anotação de PDFs selectionados com o software VoTT (Visual Object Tagging Tool)

A Figura 4.3 mostra a interface de anotação do software VoTT. É importante destacar que as imagens de documentos apresentadas na interface não são da Petrobras, mas são imagens de documentos sintéticos que nós criamos. A Tabela 4.3 apresenta o total de imagens que foram anotados com o software VoTT.

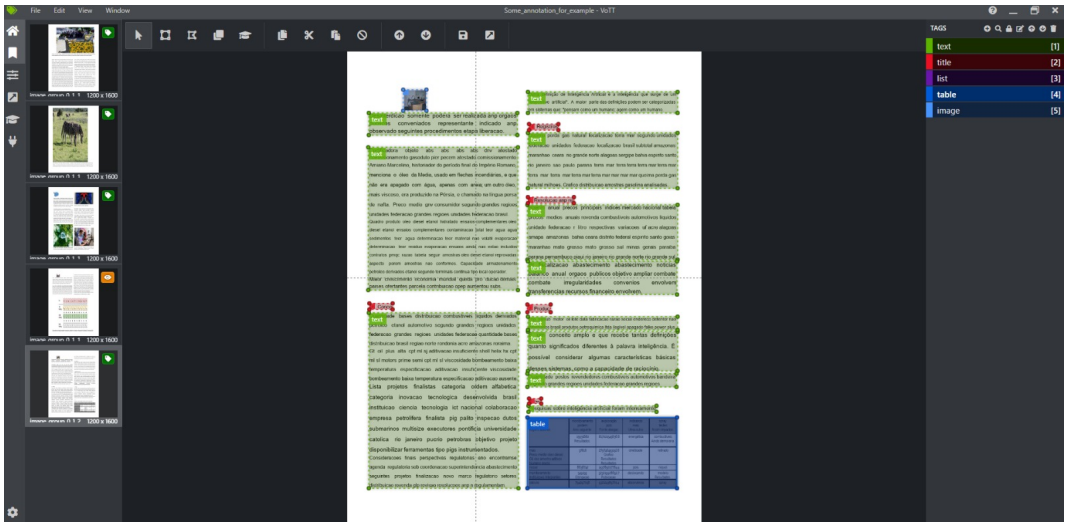

Figura 4.3: Interface de anotação do software VoTT. À esquerda, na barra lateral, estão as imagens que serão anotadas. No centro, a imagem sendo anotada. À direita, na barra lateral, estão os rótulos utilizados: texto, título, lista, tabela e imagem.

Grupos	**Número de imagens anotadas com o VoTT**
0	214
1	40
2	358
3	76
4	0
5	398
Total	**1086**

Tabela 4.3: Número de imagens anotadas por grupo utilizando o software VoTT.

4.2 Pipeline de geração e processamento de layout de document

A segunda parte do nosso pipeline, conforme ilustrada na Figura 3.1, é o Pipeline de Processamento e Gerenciamento de Layout de Documentos, cujo objetivo é criar imagens realistas de documentos e suas respectivas anotações. Envolvendo vários estágios de processamento descritos na seção Pipeline de Geração e Processamento de Layout de Documentos, nas linhas a seguir apresentamos os resultados desses estágios.

4.2.1 Gerar layout de documento usando o Gerador de Layouts Sintéticos

Conforme mencionado na seção Gerar layout de documento usando o Gerador de Layouts Sintéticos da metodologia, o Gerador de Layouts Sintéticos [12] foi treinado com o conjunto de dados PubLayNet, utilizando os hiperparâmetros especificados na Tabela 3.4. As Figuras 4.4 e 4.5 apresentam, respectivamente, as curvas de perda de treinamento e validação do Gerador de Layouts Sintéticos.

Figura 4.4: Imagem ilustrativa mostrando a curva de perda de treinamento do Gérador de Layout sintéticos.

Figura 4.5: Imagem ilustrativa mostrando a curva de perda de validação do Gérador de Layout sintéticos. O menor erro de validação ocorreu na iteração 83.920, ou seja, na época 16.

Para cada imagem anotada em um grupo de documentos da Petrobras (ou seja, para cada layout de documento de entrada), foram gerados 10 novos

layouts de documentos. A Figura 4.6 apresenta exemplos de novos layouts de documentos gerados pelo Gerador de Layouts Sintéticos. À esquerda, está o layout de entrada do documento, que corresponde à anotação manual realizada na etapa *pipeline de anotação e processamento de PDFs* do *pipeline de criação de conjuntos de dados sintéticos*. À direita, estão quatro layouts de documentos gerados pelo Gerador de Layouts Sintéticos.

Figura 4.6: Imagem ilustrativa mostrando alguns exemplos de layouts de documentos gerados pelo Gerador de Layouts Sintéticos.

A Tabela 4.4 apresenta o número total de layouts de documentos gerados por grupo.

Grupos	**Número de layouts de documentos gerados**
0	2140
1	400
2	3580
3	760
4	0
5	3980
Total	**10860**

Tabela 4.4: Número de layouts de documentos gerados por grupo. Para o grupo 4, nenhum layout foi gerado, pois não houve anotações disponíveis para este grupo.

4.2.2 Pós-Processamento para remoção de sobreposições

Conforme mencionado na seção Pós-Processamento para Remoção de Sobreposições da metodologia, o Gerador de Layouts Sintéticos pode, em algumas ocasiões, gerar layouts de documentos onde as caixas delimitadoras se sobrepõem. Para resolver esses casos, testamos duas abordagens, cujos resultados são apresentados nas linhas a seguir.

4.2.2.1 Abordagens de resolução de Sobreposição de Caixas delimitadoras sobrepostas

A Figura 4.7 apresenta um exemplo de correção de sobreposição de caixas delimitadoras. À esquerda, vemos o layout com as caixas delimitadoras sobrepostas, enquanto à direita, temos o layout corrigido, sem sobreposições.

Figura 4.7: Exemplo de correção de sobreposição pelo estagio de pós-processamento para remoção de sobreposições

A Tabela 4.5 apresenta o número total de layouts de documentos por grupo, resultante da remoção de sobreposições.

Grupos	**Número de layouts de documentos resultante**
0	1965
1	366
2	2919
3	667
4	0
5	3810
Total	**9727**

Tabela 4.5: Número de layouts de documentos por grupo, resultante da remoção de sobreposições.

A diferença nos valores em relação à Tabela 4.4 resulta da exclusão manual de certos layouts de documentos devido à sua formato. Consulte as Figuras 7.1, 7.2, 7.3, 7.4 na seção Apêndice para visualizar alguns exemplos.

4.2.3 Preencher conteúdo usando o Gerador de Imagens de Documentos Sintéticos

A Figura 4.8 apresenta um exemplo de uma imagem de documento gerada pelo Gerador de Imagens de Documentos Sintéticos. À esquerda, é mostrado o layout do documento criado pelo Gerador de Layouts Sintéticos, e à direita, a imagem do documento correspondente preenchida com conteúdo. No apêndice, na Figura 7.6 é apresentado um conjunto de oito imagens de documentos gerados.

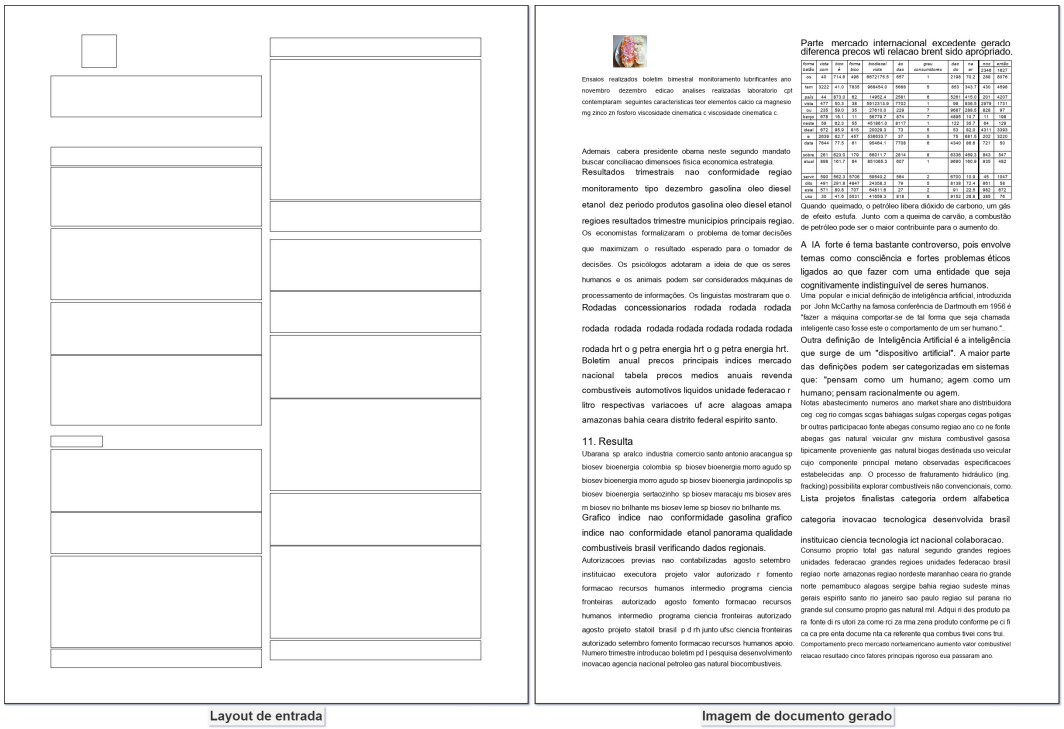

Figura 4.8: Exemplo de imagem de documento gerada pelo Gerador de Imagens de Documentos Sintéticos.

A Figura 4.9 apresenta a anotação no formato XML correspondente à imagem do documento na Figura 4.8.

Figura 4.9: Anotação em formato XML correspondente à imagem de documento na Figura 4.8, gerada pelo Gerador de Imagens de Documentos Sintéticos.

Conforme apresentado na Tabela 4.5, foram criadas *9.725 imagens de documentos sintéticos*, juntamente com suas respectivas anotações no formato XML. Seguindo nosso pipeline na Figura 3.1, essas anotações foram convertidas para o formato COCO, preparando-as para serem utilizadas nos modelos de detecção de objetos. Com as anotações no formato COCO, utilizamos o visualizador da biblioteca Detectron2 [78] para observar as delimitações dos diferentes elementos nas imagens de documentos gerados, confirmando a precisão dessas delimitações. As Figuras 7.7, 7.8, 7.9 e 7.10 no apêndice apresentam alguns exemplos ilustrativos.

4.3
Separação em dados de treinamento, validação e teste

A Tabela 4.6 apresenta a distribuição dos dados de treinamento, validação e teste utilizados em nossas experiências, conforme explicado na seção

Separação em dados de treinamento, validação e teste da metodologia.

Conjunto de dados	**Separação Totalmente Sintética**	**Teste com Dados Reais Parcial**	**Combinação Híbrida**
Treinamento	5251	6807	7339 *
Validação	3501	2918	3146 *
Teste	973	326 *	326 *

Tabela 4.6: Distribuição dos dados de treinamento, validação e teste utilizados nas experiências. Cabe ressaltar que o símbolo " $*$ " presente na tabela indica o envolvimento de dados reais.

A Figura 4.10 mostra a distribuição dos rótulos (texto, título, lista, tabela, imagem) nos dados de treinamento, validação e teste em cada experimento.

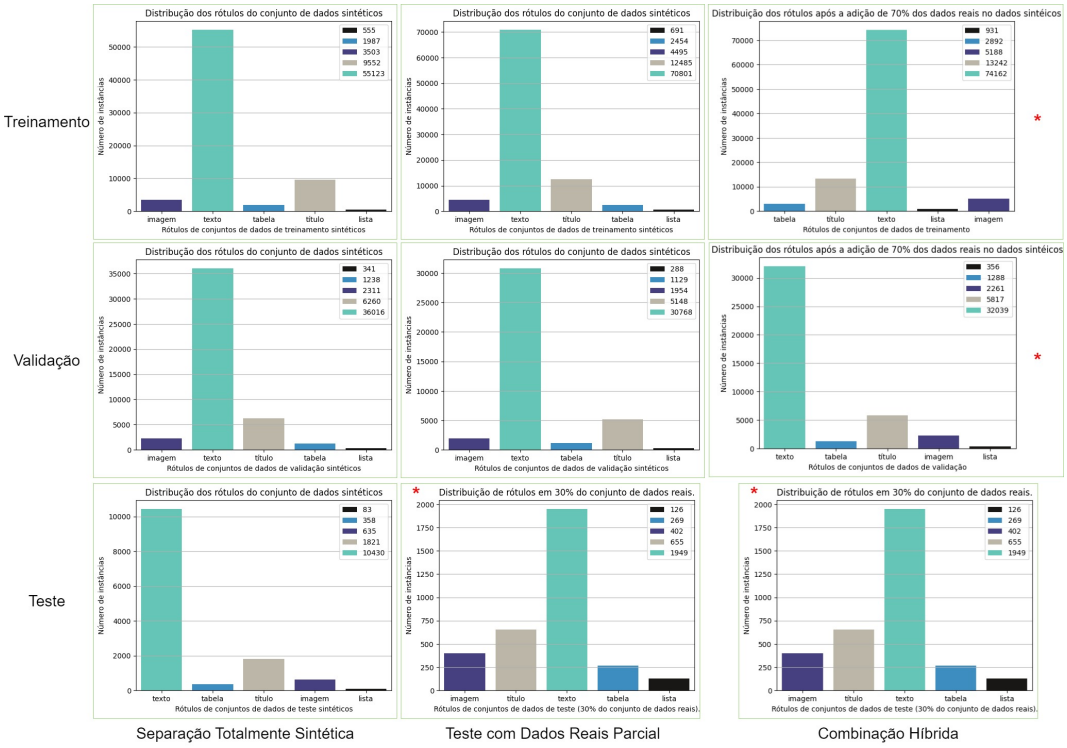

Figura 4.10: Distribuição dos rótulos nos conjuntos de dados de treinamento, validação e teste em cada experimento. O símbolo " $*$ " na figura se refere ao envolvimento dos dados reais.

4.4 Processamento de treinamento e validação dos modelos

Conforme discutido na seção Processamento de treinamento e validação dos modelos da metodologia, a abordagem que utilizamos para o treinamento dos modelos foi o ajuste fino, cujo objetivo, lembramos, é utilizar o conhecimento adquirido pelo modelo como ponto de partida para o novo treinamento.

Para os três modelos, a saber, Mask R-CNN, Mask DINO e RT-DETR, utilizamos os modelos pré-treinados no dataset COCO. Nas seções a seguir, apresentaremos os resultados obtidos para cada um desses modelos nas diferentes experiências.

4.4.1 Separação Totalmente Sintética

Este experimento, conforme discutido na seção Separação em dados de treinamento, validação e teste da metodologia, consiste na utilização de conjuntos de dados totalmente sintéticos.

4.4.1.1 Treinamento e validação do modelo Mask R-CNN

A Figura 4.11 mostra a curva de perda do modelo Mask R-CNN durante o treinamento, enquanto a Figura 4.12 mostra a curva de perda durante a validação. Observamos uma tendência de redução na perda ao longo do treinamento, indicando que o modelo estava aprendendo as características do conjunto de dados sintético. Durante a validação, o menor erro foi alcançado na iteração 15.251, correspondendo à época 93, sugerindo que o modelo atingiu seu desempenho máximo nesse ponto.

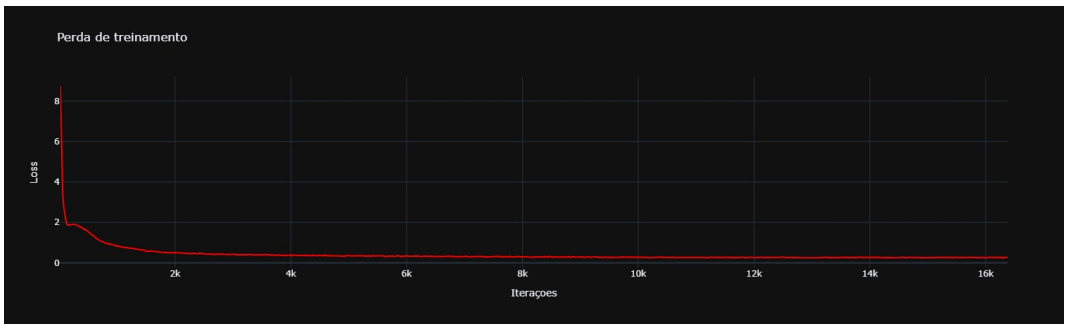

Figura 4.11: Curva de perda durante o treinamento do modelo Mask R-CNN no experimento de Separação Totalmente Sintética.

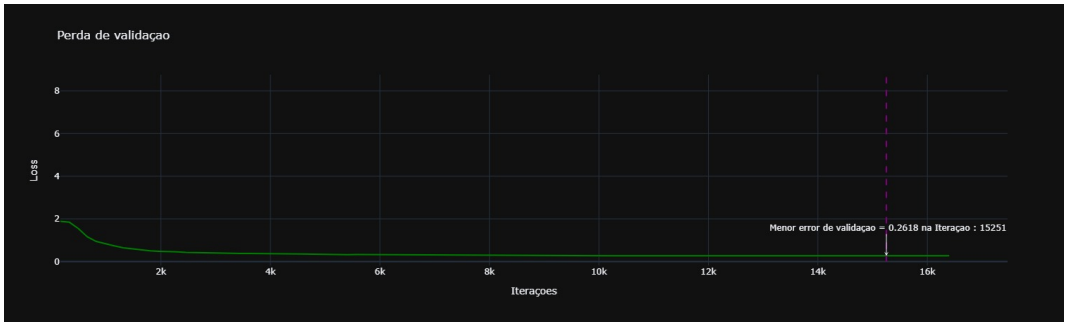

Figura 4.12: Curva de perda durante a validação do modelo Mask R-CNN no experimento de Separação Totalmente Sintética.

4.4.1.2 Treinamento e validação do modelo Mask DINO

As curvas de perda durante o treinamento e validação do modelo Mask DINO são apresentadas, respectivamente, nas Figuras 4.13 e 4.14. Observa-se uma redução na perda de treinamento ao longo das iterações. Na curva de perda durante a validação, o menor erro foi atingido na iteração 1.967 correspondendo à época 3, indicando que o modelo alcançou seu desempenho máximo.

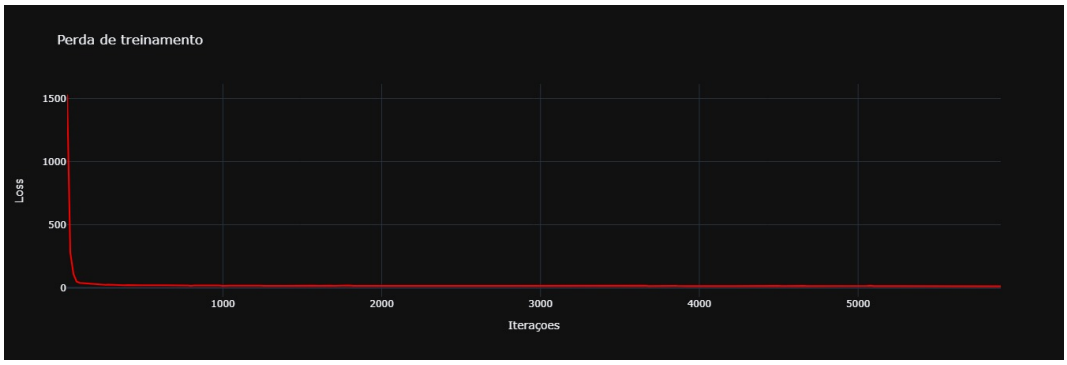

Figura 4.13: Curva de perda durante o treinamento do modelo Mask DINO no experimento de Separação Totalmente Sintética.

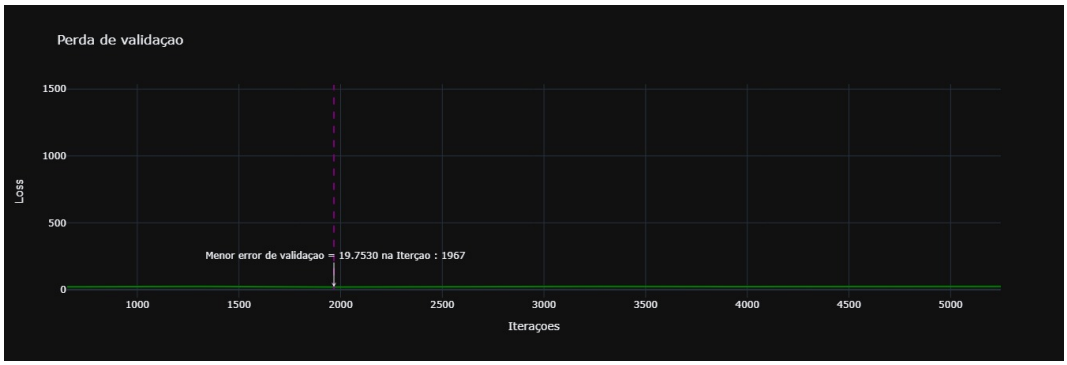

Figura 4.14: Curva de perda durante a validação do modelo Mask DINO no experimento de Separação Totalmente Sintética.

4.4.1.3 Treinamento e validação do modelo RT-DETR

As Figuras 4.15 e 4.16 mostram, respectivamente, as curvas de perda durante o treinamento e a validação do modelo RT-DETR. Como mostrado nas figuras, observamos na curva de perda durante o treinamento uma tendência de redução, indicando que o modelo está aprendendo as características do conjunto de dados sintéticos. Para a curva de perda na validação, o menor erro foi alcançado na época 99.

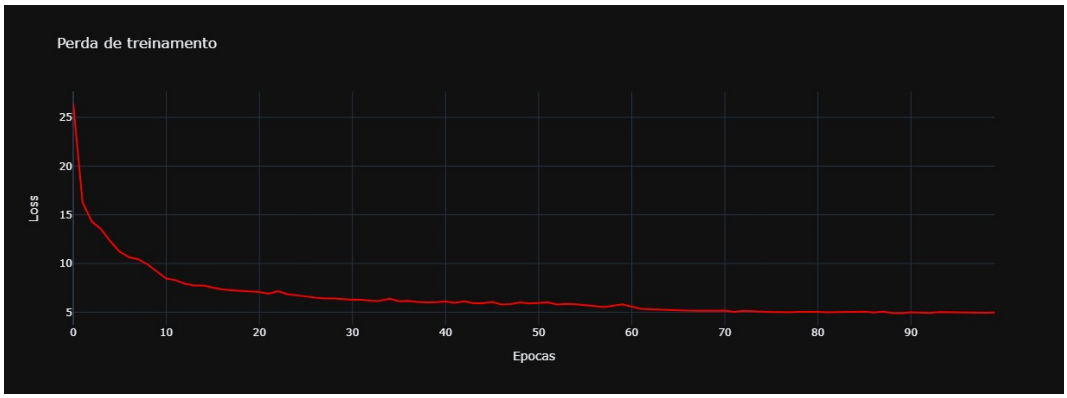

Figura 4.15: Curva de perda durante o treinamento do modelo RT-DETR no experimento de Separação Totalmente Sintética.

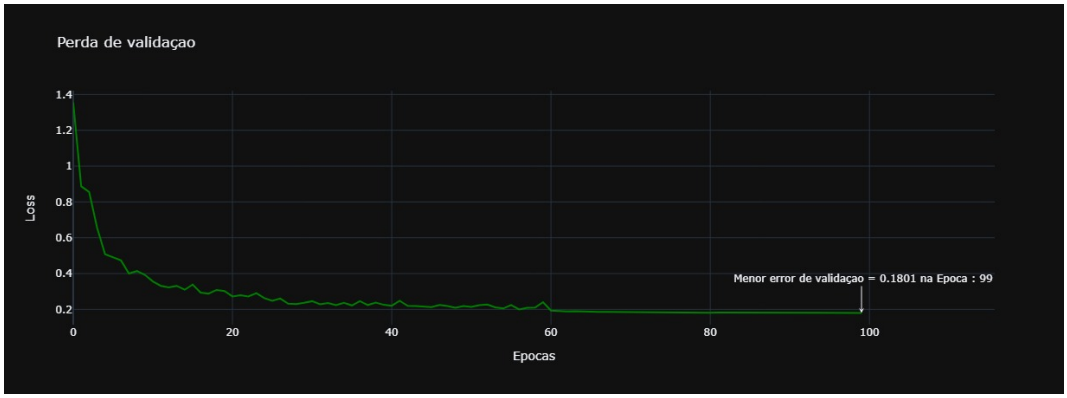

Figura 4.16: Curva de perda durante a validação do modelo Mask R-CNN no experimento de Separação Totalmente Sintética.

4.4.2 Teste com Dados Reais Parcial

Como discutido na seção Separação em dados de treinamento, validação e teste da metodologia, este experimento consiste na utilização de conjuntos de dados sintéticos para o treinamento e validação, enquanto 30% dos dados reais são utilizados como conjunto de teste.

4.4.2.1 Treinamento e validação do modelo Mask R-CNN

As Figuras 4.17 e 4.18 mostram, respectivamente, as curvas de perda durante o treinamento e a validação do modelo Mask R-CNN. Observa-se uma redução do erro de treinamento na curva de perda, refletida pela queda constante da curva. Já na curva de perda de validação, o menor erro foi alcançado na iteração 18.956, correspondendo à época 89, o que sugere que o modelo atingiu seu desempenho máximo nesse ponto.

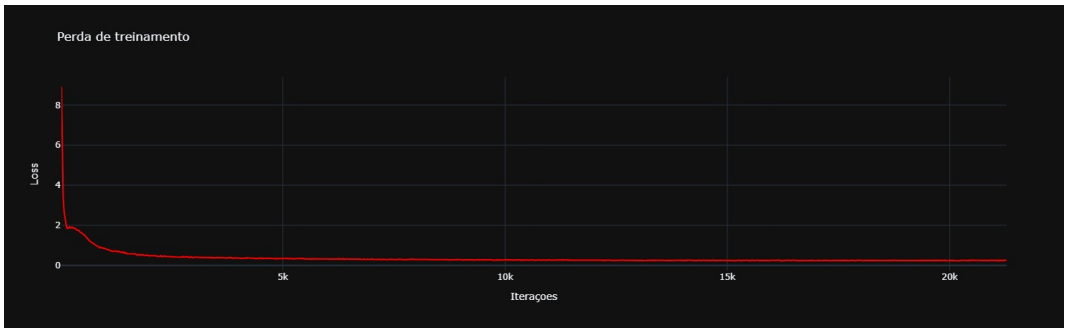

Figura 4.17: Curva de perda durante o treinamento do modelo Mask R-CNN no experimento de Teste com Dados Reais Parcial.

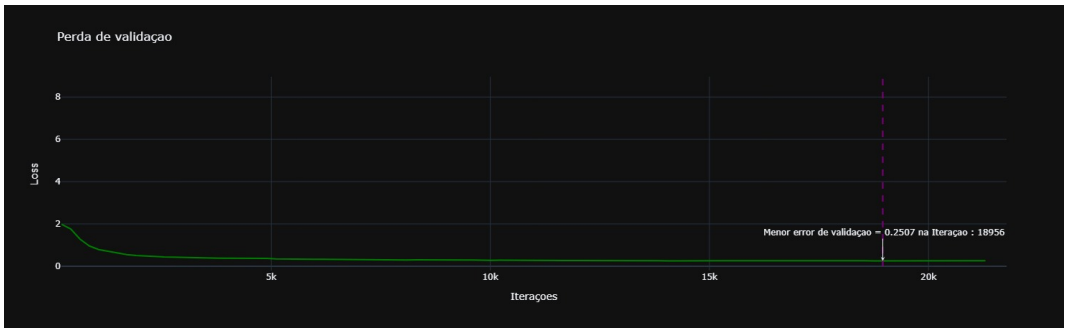

Figura 4.18: Curva de perda durante a validação do modelo Mask R-CNN no experimento de Teste com Dados Reais Parcial.

4.4.2.2 Treinamento e validação do modelo Mask DINO

As Figuras 4.19 e 4.20 apresentam as curvas de perda do modelo Mask DINO durante o experimento de Teste com Dados Reais Parcial, tanto no treinamento quanto na validação, respectivamente.

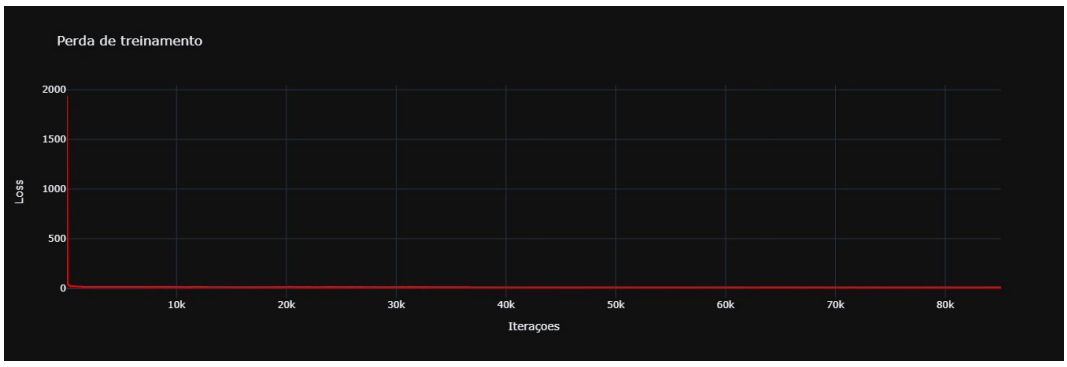

Figura 4.19: Curva de perda durante o treinamento do modelo Mask DINO no experimento de Teste com Dados Reais Parcial.

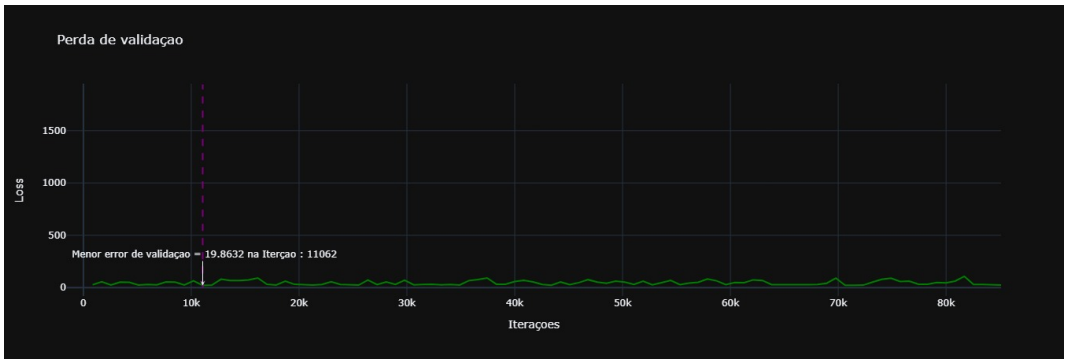

Figura 4.20: Curva de perda durante a validação do modelo Mask DINO no experimento de Teste com Dados Reais Parcial.

A curva de perda na validação mostra a evolução da perda ao longo das iterações, com o menor erro de validação sendo atingido na iteração 11.062, correspondendo à época 13, sugerindo que o modelo atingiu seu desempenho máximo nesse ponto.

4.4.2.3 Treinamento e validação do modelo RT-DETR

As Figuras 4.21 e 4.22 apresentam as curvas de perda do modelo RT-DETR durante o treinamento e a validação, respectivamente, no contexto do experimento de Teste com Dados Reais Parcial.

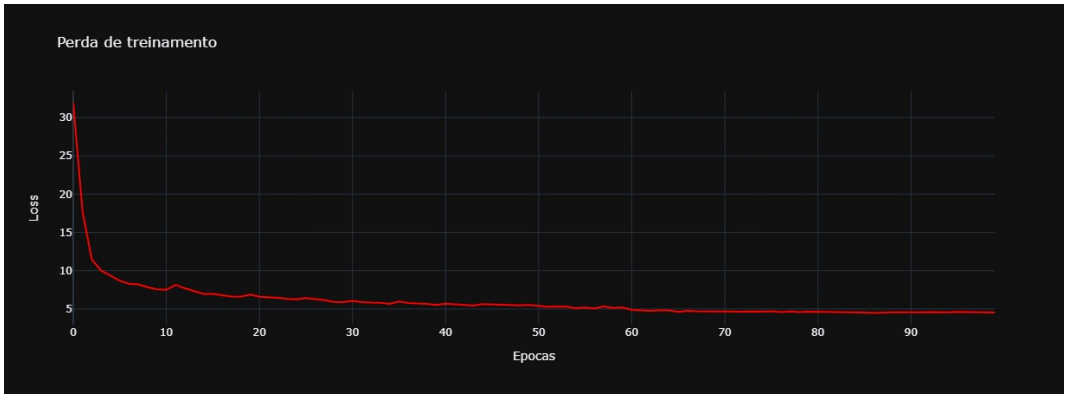

Figura 4.21: Curva de perda durante o treinamento do modelo RT-DETR no experimento de Teste com Dados Reais Parcial.

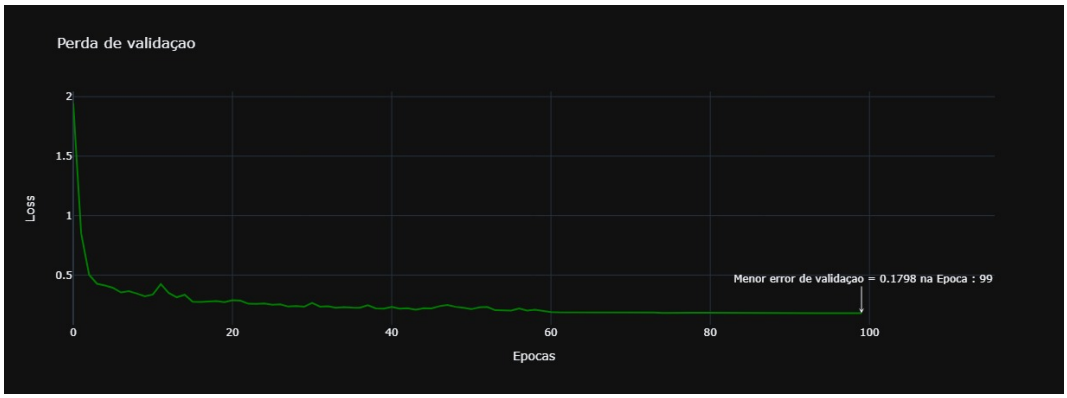

Figura 4.22: Curva de perda durante a validação do modelo RT-DETR no Experimento de Teste com Dados Reais Parcial. O menor erro de validação foi registrado na época 99.

A curva de validação demonstra a evolução da perda ao longo das épocas, destacando que o menor erro foi atingido na época 99, indicando que o modelo alcançou seu desempenho ideal nesse momento.

4.4.3 Combinação Híbrida

Por fim, este experimento consiste em treinar os modelos após a adição de 70% dos dados reais aos dados sintéticos, conforme destacado na seção Separação em dados de treinamento, validação e teste.

4.4.3.1 Treinamento e validação do modelo Mask R-CNN

As Figuras 4.23 e 4.24 mostram, respectivamente, as curvas de perda durante o processo de treinamento e validação do modelo Mask R-CNN no experimento de Combinação Híbrida.

A curva de validação demonstra uma redução na perda ao longo das iterações, com o menor erro de validação observado na iteração 19.006, correspondente à época 83, indicando que o modelo alcançou seu desempenho máximo nesse ponto.

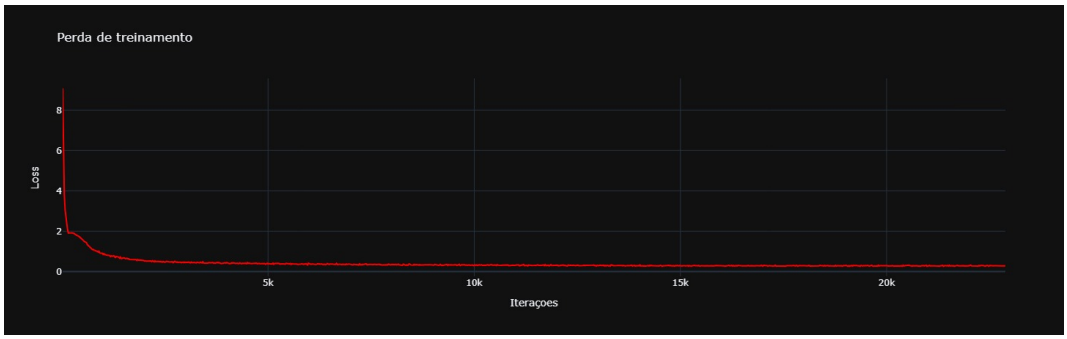

Figura 4.23: Curva de perda durante o treinamento do modelo Mask R-CNN no experimento de Combinação Híbrida.

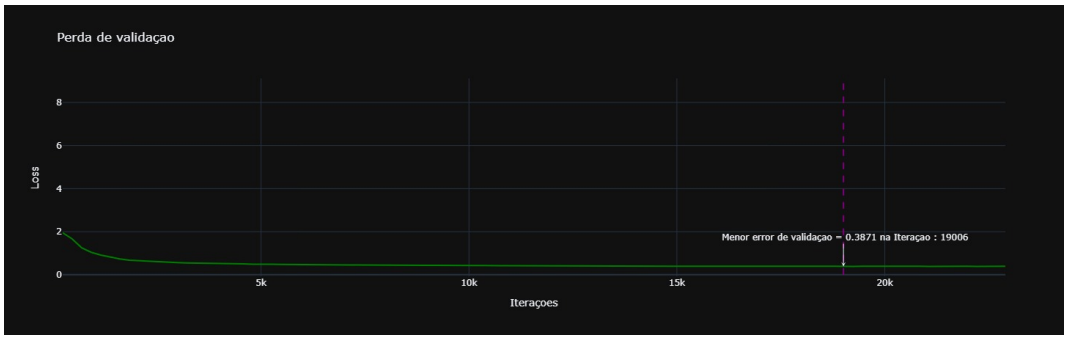

Figura 4.24: Curva de perda durante a validação do modelo Mask R-CNN no experimento de Combinação Híbrida.

4.4.3.2 Treinamento e validação do modelo Mask DINO

A Figura 4.25 mostra a curva de perda durante o treinamento, enquanto a Figura 4.26 mostra a curva de perda durante a validação do modelo Mask DINO.

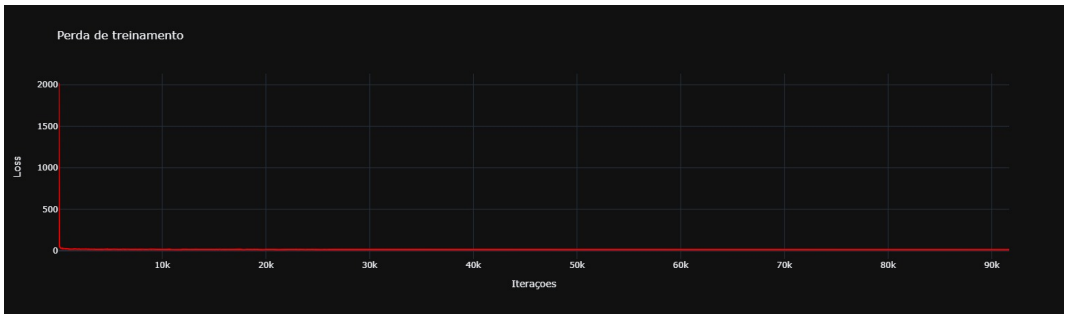

Figura 4.25: Curva de perda durante o treinamento do modelo Mask DINO no experimento de Combinação Híbrida.

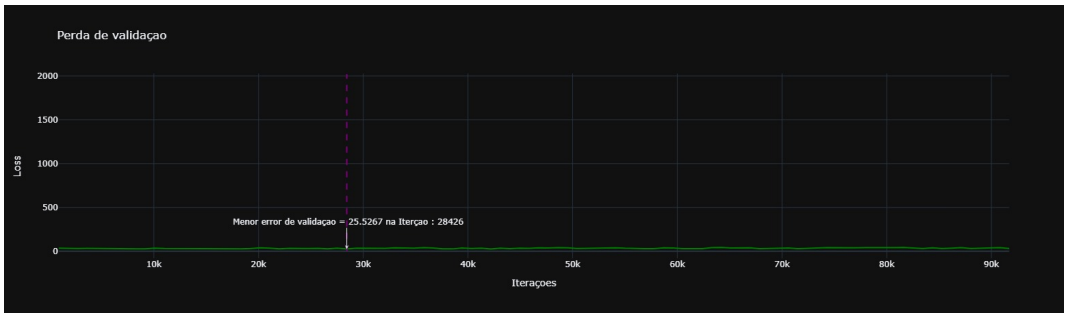

Figura 4.26: Curva de perda durante a validação do modelo Mask DINO no experimento de Combinação Híbrida.

Na curva de perda durante a validação, observa-se a evolução da perda ao longo das iterações, com o menor erro de validação sendo registrado na iteração 28.426, correspondente à época 31. Isso indica que o modelo atingiu seu desempenho máximo nesse ponto.

4.4.3.3 Treinamento e validação do modelo RT-DETR

As Figuras 4.27 e 4.28 mostram, respectivamente, as curvas de perda durante o treinamento e a validação do modelo RT-DETR. A curva de validação ilustra a evolução da perda ao longo das épocas, com o menor erro de validação sendo registrado na época 99, indicando que o modelo atingiu seu desempenho máximo nesse ponto.

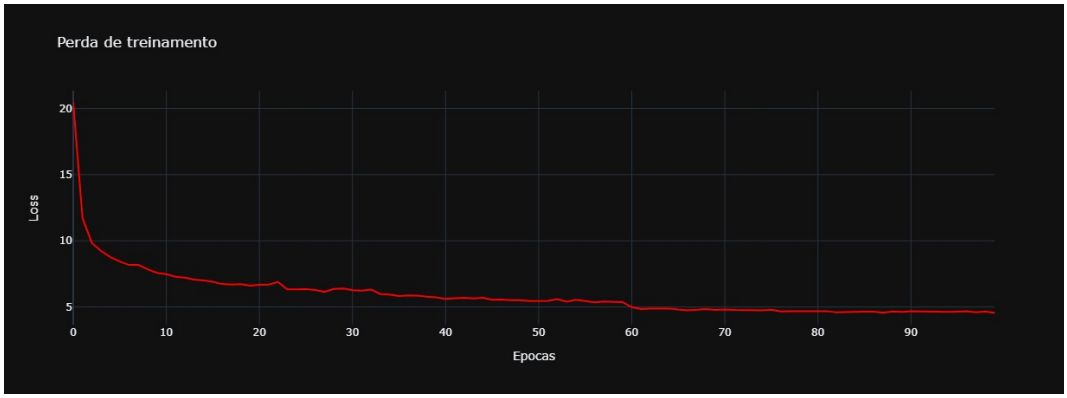

Figura 4.27: Curva de perda durante o treinamento do modelo RT-DETR no experimento de Combinação Híbrida.

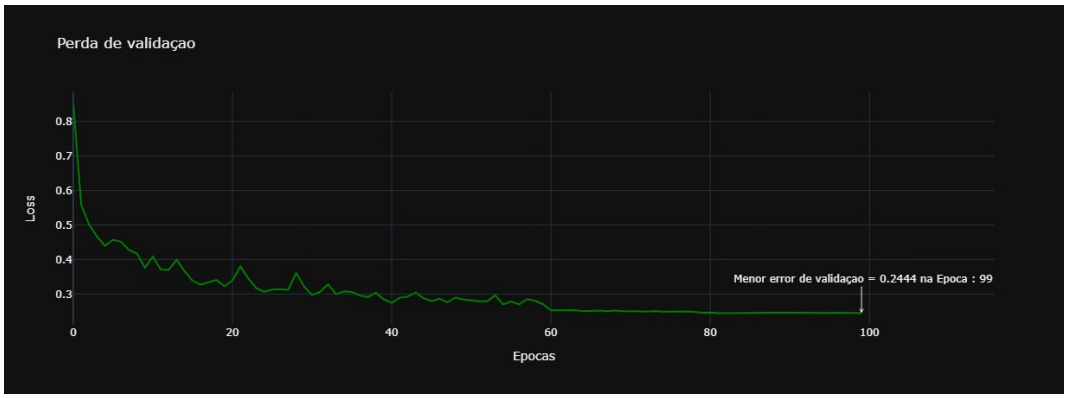

Figura 4.28: Curva de validação do modelo RT-DETR no experimento de Combinação Híbrida. O menor erro de validação ocorreu na época 99.

4.5 Processamento de teste dos modelos

Como mostrado na Figura 3.16, que ilustra nosso pipeline de treinamento, validação e teste dos modelos, assim como a seleção do melhor modelo, é evidente que, após as fases de treinamento e validação abordadas nas seções anteriores e cujos resultados foram apresentados, segue a fase de teste, conforme explicado na seção Processamento de teste dos modelos. A Tabela 4.7 apresenta os resultados de detecção e segmentação dos diferentes modelos em cada experimento. As colunas marcadas com − indicam que o modelo RT-DETR não realiza segmentação, conforme explicado na seção RT-DETR do Background.

Separação Totalmente Sintética												
Tarefas	**Detecção**						**Segmentação**					
Modelos	**mAP**	**mAP50**	**mAP75**	**mAPs**	**mAPm**	**mAPl**	**mAP**	**mAP50**	**mAP75**	**mAPs**	**mAPm**	**mAPl**
Mask R-CNN	86.13	94.38	92.07	32.89	51.93	88.14	85.10	94.42	91.83	36.27	57.31	87.32
Mask DINO	91.48	97.64	94.39	27.37	50.50	94.15	91.23	98.09	96.15	33.54	60.54	93.62
RT-DETR	96.30	99	98.40	39.00	76.50	97.80	-	-	-	-	-	-

Teste com Dados Reais Parcial												
Modelos	**mAP**	**mAP50**	**mAP75**	**mAPs**	**mAPm**	**mAPl**	**mAP**	**mAP50**	**mAP75**	**mAPs**	**APm**	**mAPl**
Mask R-CNN	19.80	29.94	20.09	0	5.71	22.33	20.30	29.81	21.30	0	5.99	22.90
Mask DINO	22.81	31.96	22.92	0	8.72	25.39	22.80	32.26	23.00	0	6.76	25.50
RT-DETR	23.90	33.20	23.40	0	4.30	26.8	-	-	-	-	-	-

Combinação Híbrida												
Modelos	**mAP**	**mAP50**	**mAP75**	**mAPs**	**mAPm**	**mAPl**	**mAP**	**mAP50**	**mAP75**	**mAPs**	**mAPm**	**mAPl**
Mask R-CNN	43.62	58.16	49	0	27.33	44.78	43.55	58.41	49.30	0	26.67	45.06
Mask DINO	65.78	78.26	73.37	2.11	39.40	67.73	65.43	78.29	72.79	2.23	38.95	67.41
RT-DETR	69.60	81.00	77.00	14.80	38.80	71.19	-	-	-	-	-	-

Tabela 4.7: Resultados dos modelos em cada experimento nos dados de teste envolvidos.

4.6 Seleção de Modelo

Conforme discutido na seção Seleção de Modelo da metodologia, a seleção final do modelo consiste em escolher aquele que apresentou o melhor desempenho nos dados de teste, considerando a métrica de Precisão Média (mAP).

Conforme reportado na Tabela 4.7, podemos notar que o modelo RT-DETR foi o que apresentou o melhor desempenho em todos os experimentos. Destacado pela região em verde na tabela, para o experimento envolvendo a Separação Totalmente Sintética, o modelo alcançou um score de mAP de 96,30%. No experimento envolvendo o Teste com Dados Reais Parcial, o modelo atingiu um score de mAP de 23,90%, e, por fim, no experimento de Combinação Híbrida, o modelo obteve um score de mAP de 69,60%.

5
Discussão

Neste trabalho, que adotou a abordagem de detecção de objetos em documentos, especificamente nos documentos da Petrobras, foi necessário criar um conjunto de dados sintéticos para superar algumas limitações, propondo um pipeline para esse fim. Entre essas limitações, podemos citar a quantidade insuficiente de documentos PDF, que não permitia o uso eficaz de modelos de detecção de objetos baseados em aprendizado profundo, além da impossibilidade de utilizar conjuntos de dados públicos volumosos devido às diferenças de layout em relação aos documentos da Petrobras.

Esta seção tem como objetivo discutir os resultados obtidos, desde a criação do nosso conjunto de dados sintéticos até os diferentes experimentos que realizamos com os modelos de detecção de objetos. Realizaremos uma análise qualitativa do conjunto de dados sintético que criamos, usando a métrica mAP dos modelos de detecção de objetos aplicados. No entanto, antes de proceder a essa análise, é essencial avaliar a eficiência do pipeline desenvolvido em comparação com outros trabalhos relevantes. Ao final, discutiremos os trabalhos futuros.

5.1
Comparação do pipeline com outros trabalhos

Nosso pipeline de criação de conjunto de dados de documentos sintéticos, conforme mostrado na Figura 3.1, consiste em duas partes que são: o Pipeline de anotação e processamento de PDFs e o Pipeline de geração e processamento de layout de documentos.

1. Pipeline de anotação e processamento de PDFs: Foca-se principalmente na seleção de documentos representativos dentro de um conjunto de dados para anotação manual, com base na hipótese de que deve haver um grau de similaridade entre as páginas dos documentos de um mesmo domínio. Neste caso, foram considerados os documentos da Petrobras, compostos principalmente por relatórios,

2. Pipeline de geração e processamento de layout de documentos: Concentra-se na geração de layouts de documentos e na criação de documentos realistas com suas respectivas anotações, a fim de serem utilizados por modelos de detecção de objetos.

É importante reconhecer que já existem trabalhos que, assim como o nosso, criam layouts de documentos e os preenchem com conteúdo, gerando documentos realista sintéticos completos para o treinamento de modelos de detecção de objetos, contribuindo significativamente para o avanço na automatização do processamento de documentos.

Um exemplo disso é o trabalho de [15], que, em uma primeira fase, utiliza um modelo de difusão para gerar layouts de documentos e, em uma segunda fase, realiza um processo de correspondência com dados reais (PubLayNet foi utilizado). Esse processo envolve a aproximação entre as caixas delimitadoras dos dados reais e dos layouts de documentos gerados, baseando-se em categorias, aspecto de proporção (aspect ratio), tamanho, etc. Após encontrar as caixas delimitadoras mais semelhantes, o conteúdo dos pixels correspondentes nos dados reais é recortado e inserido nos layouts de documentos gerados.

Ao comparar este trabalho com o Pipeline de Geração e Processamento de Layout de Documentos, observam-se diferenças significativas na forma de preenchimento do conteúdo nas caixas delimitadoras dos layouts gerados. Neste caso, o conteúdo para categorias como texto, lista, título e tabela é gerado diretamente em formato escrito, o que permite um controle espacial preciso sobre cada elemento adicionado, incluindo as coordenadas exatas das linhas de texto. Isso pode ser particularmente benéfico para aplicações que requerem a segmentação precisa das linhas nos documentos, como a extração de informações estruturadas em sistemas de análise automática de documentos por meio de ferramentas como OCR, ou para ensinar a um modelo de detecção de objetos a identificar as linhas de um texto. Para as categorias de tipo imagem, diferentes imagens podem ser inseridas, proporcionando uma diversificação adicional.

Por outro lado, o trabalho realizado por [16] adota uma abordagem utilizando uma rede bayesiana para gerar documentos realistas com layouts variados e anotações associadas, abordando o grande desafio da escassez de dados anotados para o treinamento de modelos de detecção de objetos em documentos. Uma diferença fundamental entre o trabalho deles e o nosso é que, enquanto a geração de layouts em nossa abordagem utiliza um gerador de conteúdo previamente treinado com o dataset PubLayNet, o que o torna altamente adaptado ao domínio específico dos documentos da Petrobras, a abordagem de [16] gera os layouts diretamente a partir de uma distribuição de probabilidade, permitindo uma maior flexibilidade em termos de variedade de layouts.

Além disso, o método de [16] introduz uma diversidade maior nas

categorias de documentos, incluindo elementos como cabeçalhos e rodapés, o que contribui para aumentar o realismo dos documentos gerados. No entanto, o processo de preenchimento dos layouts gerados com conteúdo não é claramente detalhado no trabalho deles, o que sugere que o procedimento pode ser semelhante ao nosso.

5.1.1 Vantagens do nosso pipeline

Embora nosso trabalho compartilhe semelhanças com outros estudos, nosso pipeline se destaca por sua modularidade, tornando-o altamente escalável. Graças à sua estrutura flexível, diferentes componentes podem ser facilmente substituídos ou ajustados para se adaptar a novas necessidades ou contextos específicos. Essa capacidade de adaptação permite que o pipeline seja personalizado para diferentes tipos de documentos ou domínios, garantindo sua relevância em uma ampla gama de aplicações.

Além disso, a modularidade do pipeline facilita a integração de novas tecnologias ou algoritmos à medida que surgem, sem a necessidade de redesenhar completamente o sistema. Por exemplo, na primeira parte do pipeline (Pipeline de anotação e processamento de PDFs), o algoritmo de agrupamento utilizado para classificar as páginas dos documentos pode ser facilmente substituído ou complementado por outros algoritmos. Isso permite uma avaliação comparativa de desempenho, facilitando a seleção do algoritmo que oferece os melhores resultados para o conjunto de dados específico, assegurando uma maior precisão na classificação das páginas.

Na segunda parte do pipeline (Pipeline de geração e processamento de layout de documentos), a modularidade permite a integração de diferentes geradores de layout, o que maximiza a diversidade dos layouts gerados, resultando em documentos sintéticos mais variados e realistas. Ao contrário do que pode ter sido sugerido ao longo do nosso trabalho, nosso gerador de conteúdo não se limita apenas ao preenchimento de caixas delimitadoras para tipos como texto, título, lista, tabela ou imagem. Suas capacidades vão além, incluindo o preenchimento de caixas para tipos como parágrafo, subtítulo, subsubtítulo, número de página e equação. Essa flexibilidade é especialmente relevante ao analisarmos documentos reais, onde a profundidade e a hierarquia dentro das seções podem desempenhar um papel significativo na estrutura do documento.

Em resumo, a flexibilidade do nosso pipeline garante que ele não fique restrito a um único domínio, ampliando sua aplicabilidade em diferentes contextos e tipos de documentos. Isso não só aumenta a longevidade do

pipeline, mas também permite que ele evolua ao longo do tempo, incorporando avanços técnicos que podem melhorar ainda mais o desempenho na geração de documentos mais realistas e na detecção de objetos.

5.1.2 Limitações do pipeline

Embora o nosso pipeline tenha mostrado eficácia na geração de documentos sintéticos e na facilitação do treinamento de modelos de detecção de objetos, ele não está isento de limitações. Primeiramente, a dependência do dataset PubLayNet para o treinamento inicial do gerador de layouts pode restringir a diversidade dos layouts gerados, uma vez que eles tendem a refletir as características predominantes desse conjunto de dados. Isso pode resultar em uma falta de generalização quando aplicados a domínios que apresentam layouts significativamente diferentes.

Além disso, a primeira parte do nosso pipeline, que envolve a seleção manual de documentos PDF representativos, está sujeita a desafios adicionais. Embora seja essencial que esses documentos reflitam a diversidade dentro de um domínio específico, a natureza manual desse processo pode introduzir viés e limitar a variedade dos layouts selecionados. Portanto, é crucial que os documentos processados, mesmo pertencendo a um domínio específico, apresentem uma variedade suficiente para garantir que o pipeline seja capaz de lidar de maneira eficaz com diferentes estruturas e complexidades dos documentos reais.

Essas limitações devem ser consideradas ao avaliar a aplicabilidade e a eficácia do nosso pipeline em diferentes contextos e domínios de documentos. A superação dessas limitações pode exigir ajustes adicionais ou complementos à abordagem atual.

5.2 Interpretação dos resultados

Após estabelecermos comparações com outros trabalhos, assim como as vantagens e limitações do nosso pipeline, passamos agora a apresentar e discutir os resultados dos experimentos realizados com os modelos de detecção de objetos (Mask R-CNN, Mask DINO, RT-DETR), conforme discutido na seção Resultados. Como destacado no início desta seção de discussão, utilizaremos o mAP como métrica de avaliação principal.

5.2.1 Análise focada no conjunto de dados

Começamos com o primeiro experimento: **Separação Totalmente Sintética**, que consideramos como a nossa linha de base (baseline). Ao analisar os resultados na Tabela 4.7, observamos que a métrica mAP para os três modelos apresenta valores elevados: 86,13% para o Mask R-CNN, 91,48% para o Mask DINO, e 96,30% para o RT-DETR. Sob a perspectiva do conjunto de dados, esses resultados sugerem que os documentos sintéticos que criamos são eficazes para o treinamento dos modelos de detecção de objetos, permitindo uma boa performance em um cenário totalmente sintético.

Seguindo para o segundo experimento: **Teste com Dados Reais Parcial**, ao analisar os resultados na Tabela 4.7, observamos que a métrica mAP para os três modelos diminuiu consideravelmente, apresentando valores de 19,80% para o Mask R-CNN, 22,81% para o Mask DINO e 23,90% para o RT-DETR, em comparação ao experimento Separação Totalmente Sintética. Sob a perspectiva do conjunto de dados, esses resultados sugerem que, embora os dados sintéticos que criamos sejam eficazes entre si, eles ainda não capturam adequadamente o comportamento dos dados reais, neste caso, os documentos da Petrobras.

Para finalizar, no terceiro experimento: **Combinação Híbrida**, ao analisar os resultados na Tabela 4.7, observamos que a métrica mAP para os três modelos, em comparação ao experimento Separação Totalmente Sintética, permanece relativamente baixa. No entanto, quando comparada ao experimento Teste com Dados Reais Parcial, há uma melhoria significativa, com 43,62% para o Mask R-CNN, 65,78% para o Mask DINO e 63,60% para o RT-DETR. Esses resultados permitem deduzir duas conclusões principais: a primeira confirma a afirmação discutida no segundo experimento de que os dados sintéticos ainda não capturam adequadamente o comportamento dos dados reais; a segunda é que a combinação de dados sintéticos e reais pode ajudar a melhorar o desempenho dos modelos, indicando que uma abordagem híbrida pode ser uma estratégia promissora para lidar com a complexidade dos documentos reais.

Além disso, considerando nosso pipeline de um ponto de vista de produção, a adição de conjuntos de dados reais aos conjuntos de dados sintéticos pode tornar o modelo mais robusto e adaptável a diferentes cenários do mundo real. Essa combinação não só melhora a capacidade do modelo de generalizar para novos dados, como também pode aumentar a precisão em ambientes de produção onde os documentos apresentam uma maior variabilidade. Isso torna ainda mais eficiente o uso de dados sintéticos, cujo objetivo principal é aumentar o volume de dados disponíveis para o treinamento de modelos baseados

em aprendizado profundo, especialmente em situações onde os dados reais são escassos ou estão disponíveis em quantidades limitadas.

5.2.2 Análise focada nos modelos

Nesta subseção, discutimos o desempenho específico de cada modelo de detecção de objetos utilizado em nossos experimentos: Mask R-CNN, Mask DINO, e RT-DETR. A análise se baseia nas métricas obtidas em diferentes cenários experimentais, permitindo-nos identificar as forças e fraquezas de cada modelo.

1. **Mask R-CNN**: O Mask R-CNN, conhecido por sua robustez e precisão em tarefas de segmentação de objetos, apresentou resultados promissores em nosso experimento de **Separação Totalmente Sintética**, com uma métrica mAP de 86,13%. Este resultado indica que o modelo é eficaz quando treinado exclusivamente com dados sintéticos, capturando bem os padrões presentes em nossos documentos gerados.

   Entretanto, ao ser testado com dados reais no experimento **Teste com Dados Reais Parcial**, o desempenho do Mask R-CNN caiu drasticamente, atingindo um mAP de apenas 19,80%. Isso sugere que, apesar de sua alta precisão com dados sintéticos, o modelo enfrenta dificuldades em generalizar para dados reais, refletindo as diferenças entre os layouts sintéticos e os reais. No experimento **Combinação Híbrida**, o Mask R-CNN apresentou uma melhora significativa, alcançando um mAP de 43,62%, o que indica que a introdução de dados reais no treinamento ajudou o modelo a melhorar sua capacidade de generalização, embora ainda longe de um desempenho ideal.

2. **Mask DINO**: O Mask DINO destacou-se pelo seu bom desempenho, especialmente no experimento de **Separação Totalmente Sintética**, onde alcançou um mAP de 91,48%. Isso demonstra sua capacidade superior de adaptação a dados gerados artificialmente. Quando testado com dados reais, o desempenho do Mask DINO também sofreu uma queda, atingindo um mAP de 22,81%, mas ainda assim apresentou resultados ligeiramente melhores que o Mask R-CNN, indicando uma leve vantagem na generalização para dados reais.

   No experimento **Combinação Híbrida**, o Mask DINO mostrou-se particularmente eficaz, com um mAP de 65,78%, o que representa a maior recuperação de desempenho entre os modelos testados. Esse resultado sugere que o Mask DINO é capaz de aproveitar melhor a combinação

de dados sintéticos e reais, potencialmente devido à sua arquitetura e estratégias de treinamento mais avançadas.

3. O RT-DETR, por sua vez, foi o modelo com o melhor desempenho geral, especialmente em cenários sintéticos. No experimento de **Separação Totalmente Sintética**, ele atingiu um mAP impressionante de 96,30%, superando os demais modelos. Esse resultado pode ser atribuído à capacidade do RT-DETR de lidar eficazmente com tarefas de detecção e segmentação em ambientes controlados, onde os dados sintéticos oferecem condições ideais para seu treinamento.

   No entanto, ao ser testado com dados reais, o RT-DETR, assim como os outros modelos, sofreu uma queda de desempenho, obtendo um mAP de 23,90%. Embora essa queda seja significativa, o RT-DETR ainda conseguiu manter um desempenho relativamente alto em comparação com os outros modelos.

   No experimento **Combinação Híbrida**, o RT-DETR continuou a demonstrar sua robustez, alcançando um mAP de 63,60%. Apesar da queda em relação ao cenário sintético, o modelo mostrou uma boa capacidade de adaptação quando exposto a uma combinação de dados sintéticos e reais, destacando-se como uma opção promissora para aplicações que envolvem variabilidade nos dados.

5.3 Trabalhos futuros

Como foi especificado nas vantagens do nosso pipeline, planejamos explorar extensivamente a estrutura modular para criar conjuntos de dados sintéticos de documentos que sejam altamente representativos.

Primeiramente, pretendemos integrar o modelo proposto por [16] em paralelo ao nosso gerador de layout de documentos atual. Isso nos permitirá tirar proveito das vantagens desse modelo, que não exige uma entrada restrita a um domínio específico, aumentando a diversidade e a flexibilidade dos layouts gerados.

Além dessas melhorias estruturais, estamos focados em aumentar o realismo dos documentos gerados. Planejamos investigar outros modelos de geração de layout para incorporar as melhorias que eles possam oferecer. Também queremos realizar experimentos adicionais, como o ajuste fino dos modelos de geração de layout de documentos utilizando

dados reais, para avaliar como os layouts gerados se comportam e se refletem de maneira mais precisa as características dos documentos reais.

Outro aspecto importante do nosso plano futuro é a adaptação do gerador de conteúdo. Além de continuar preenchendo os layouts com conteúdo, pretendemos introduzir a capacidade de adicionar ruído, como desfoque (blur) e outras distorções visuais. Isso permitirá que os documentos sintéticos sejam ainda mais realistas, simulando as imperfeições comuns em documentos escaneados ou fotografados, o que pode melhorar a robustez dos modelos de detecção de objetos.

Um aspecto crucial que ainda não abordamos é a avaliação da similaridade entre os dados sintéticos e os dados reais. Para isso, planejamos utilizar métricas avançadas, como o Earth Mover's Distance (EMD) [79], Doc-EMD [15], e a Wasserstein Distance [80, 81], além do Fréchet Inception Distance (FID) [82]. Essas métricas nos permitirão quantificar o grau de similaridade entre os dados sintéticos e reais, garantindo que os dados sintéticos sejam suficientemente realistas para o treinamento de modelos de detecção de objetos.

Também, planejamos testar o aproveitamento das anotações feitas pelos modelos nos conjuntos de dados reais. Essencialmente, a ideia é selecionar as **top-k** melhores anotações feitas pelos modelos e, a partir delas, gerar novos layouts de documentos baseados nessas anotações. Em termos simplificados, isso significa utilizar o modelo como um anotador automático, facilitando a criação de novos layouts com base nas previsões mais confiáveis dos modelos. Esse processo pode servir como ponto de partida para o aprendizado ativo (Active Learning), permitindo que os modelos se tornem mais genéricos e adaptáveis à aparição de novos layouts de documentos.

6
Conclusão

Neste estudo, exploramos o potencial da detecção de objetos aplicada a documentos PDF, com foco específico nos documentos da Petrobras. A necessidade de criar um pipeline robusto e adaptável surgiu das limitações impostas pela quantidade limitada de dados disponíveis e pela inadequação de conjuntos de dados públicos amplamente utilizados. Em resposta a esses desafios, desenvolvemos um pipeline capaz de gerar documentos sintéticos com layouts realistas e anotações automatizadas, proporcionando uma solução eficaz para o treinamento de modelos de detecção de objetos.

Testando as imagens de documentos geradas com três modelos de detecção de objetos de ponta, constatamos que o RT-DETR (Real Time Detection Transformer) alcançou uma precisão média de 96,30%, superando os resultados de modelos como Mask R-CNN (Region-based Convolutional Neural Networks) e Mask DINO (Mask DETR with Improved Denoising Anchor Boxes). Esses resultados destacam a eficácia do RT-DETR em cenários sintéticos, confirmando a capacidade do nosso pipeline de produzir dados de alta qualidade para o treinamento de modelos avançados.

Nosso pipeline demonstrou ser uma ferramenta versátil, não apenas para a criação de layouts de documentos específicos, mas também para o preenchimento desses layouts com conteúdo estruturado, permitindo a geração de grandes volumes de dados sintéticos de alta qualidade. Essas características tornam o pipeline adequado para uma ampla gama de aplicações, além de oferecer suporte ao aprendizado ativo, permitindo que os modelos sejam continuamente refinados e aprimorados.

Além de atender às necessidades específicas dos documentos da Petrobras, nossa abordagem pode ser facilmente adaptada a outras áreas, ampliando consideravelmente o escopo de aplicação. Por exemplo, no setor de educação, o pipeline pode ser usado para gerar testes, exames, trabalhos de pesquisa e materiais didáticos personalizados, o que pode facilitar a extração de conteúdo para ajudar a criar um chatbot bem-sucedido em um ambiente escolar e também criar conteúdo adaptado a diferentes níveis de ensino. No setor de saúde, ele pode ser usado para criar registros médicos sintéticos e relatórios de testes, contribuindo para o treinamento de modelos de extração de informações clínicas e para o desenvolvimento de sistemas de gerenciamento hospitalar.

Em suma, nosso trabalho contribui para o avanço das técnicas de processamento de documentos através da integração de visão computacional

e aprendizado profundo, destacando-se pela sua aplicabilidade em ambientes industriais e pela sua adaptabilidade a diferentes cenários. Futuras pesquisas poderão expandir essa abordagem, integrando novos modelos e tecnologias para aumentar ainda mais a precisão e a eficiência na geração de documentos complexos e representativos.

7
Apêndice

As Figuras 7.1, 7.2, 7.3, 7.4 apresentam respectivemente alguns exemplo de layout de documento com estrutura inconsistente.

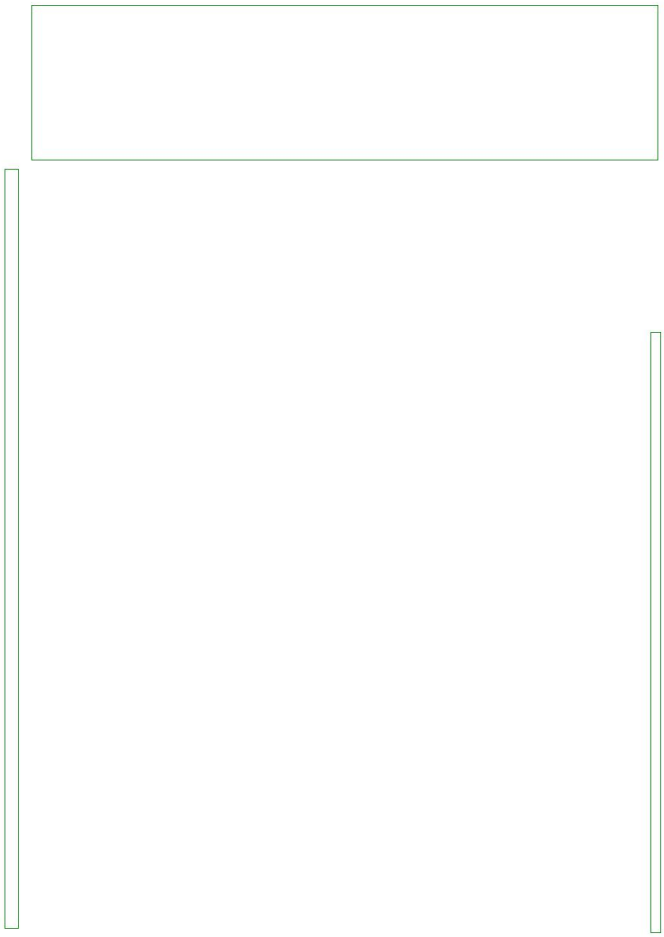

Figura 7.1: Exemplo de exclusão manual de layout de doucmento com estrutura inconsistente 1.

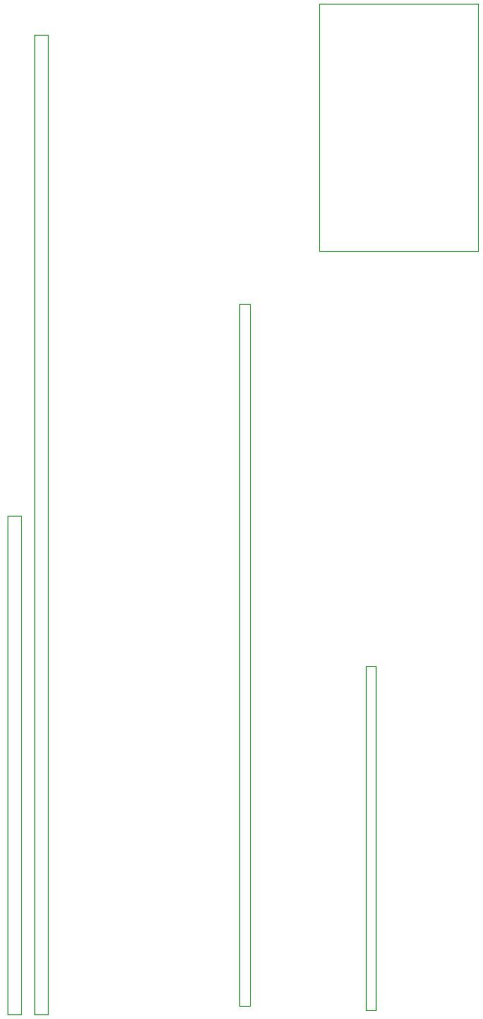

Figura 7.2: Exemplo de exclusão manual de layout de doucmento com estrutura inconsistente 2.

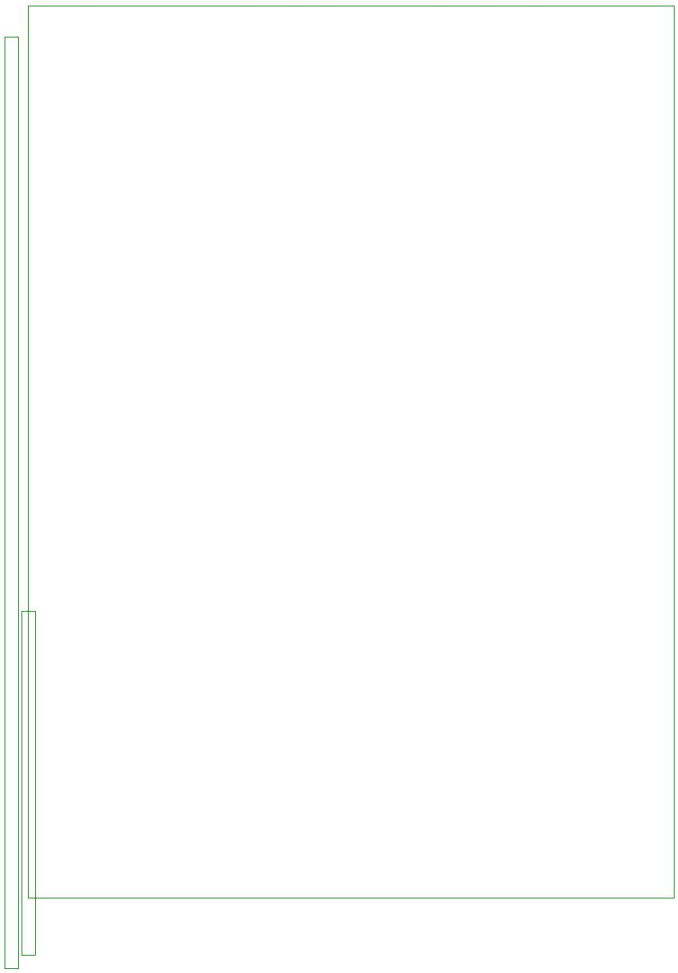

Figura 7.3: Exemplo de exclusão manual de layout de doucmento com estrutura inconsistente 3.

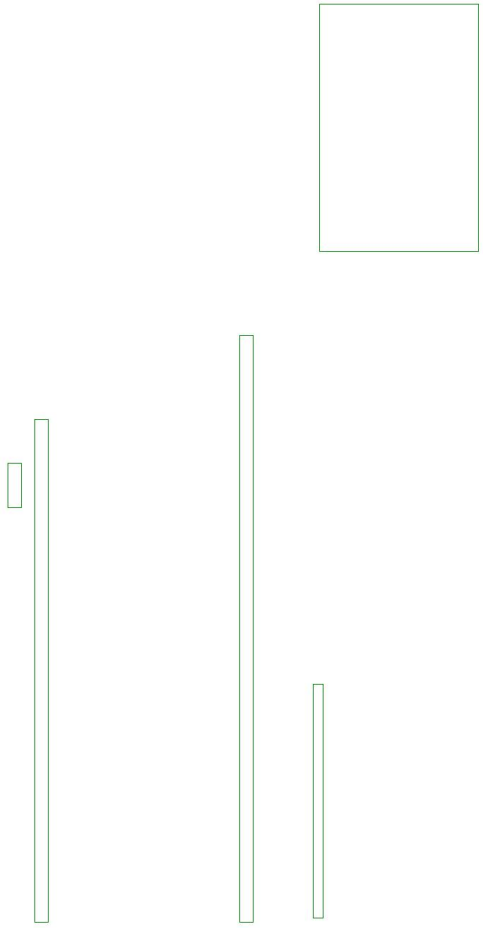

Figura 7.4: Exemplo de exclusão manual de layout de doucmento com estrutura inconsistente 4.

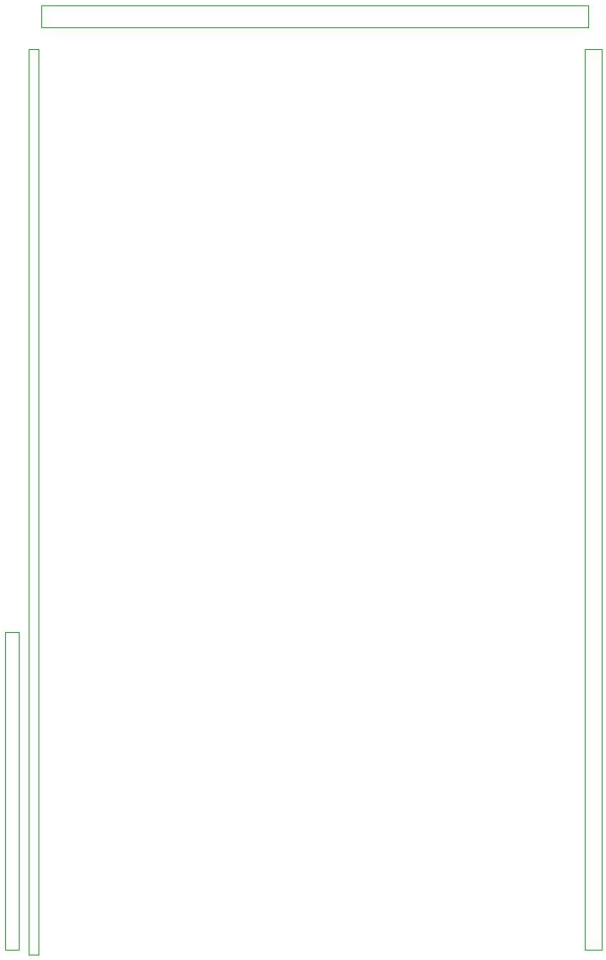

Figura 7.5: Exemplo de correção de sobreposição pelo estagio de pós-processamento para remoção de sobreposições

A Figura 7.6 apresenta um conjunto de oito imagens de documentos gerados pelo Gerador de Imagens de Documentos Sintéticos.

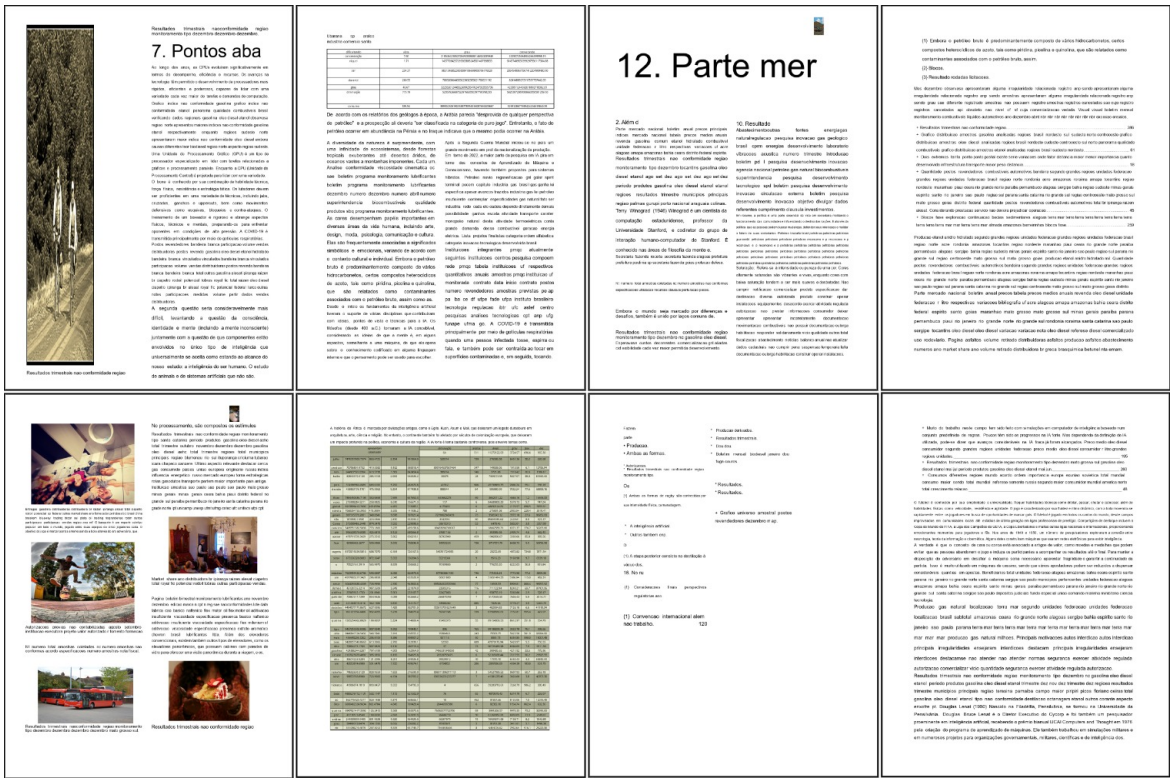

Figura 7.6: Conjunto de oito imagens de documentos gerados pelo Gerador de Imagens de Documentos Sintéticos.

As Figuras 7.7, 7.8, 7.9, 7.10 apresentam respectivemente alguns exemplo de visualização das delimitações em documentos dintéticos usando Detectron2.

Figura 7.7: Exemplo de visualização das Delimitações em Documentos Sintéticos Usando Detectron2 (1).

Figura 7.8: Exemplo de visualização das delimitações em documentos dintéticos usando Detectron2 (2).

Figura 7.9: Exemplo de visualização das delimitações em documentos dintéticos usando Detectron2 (3).

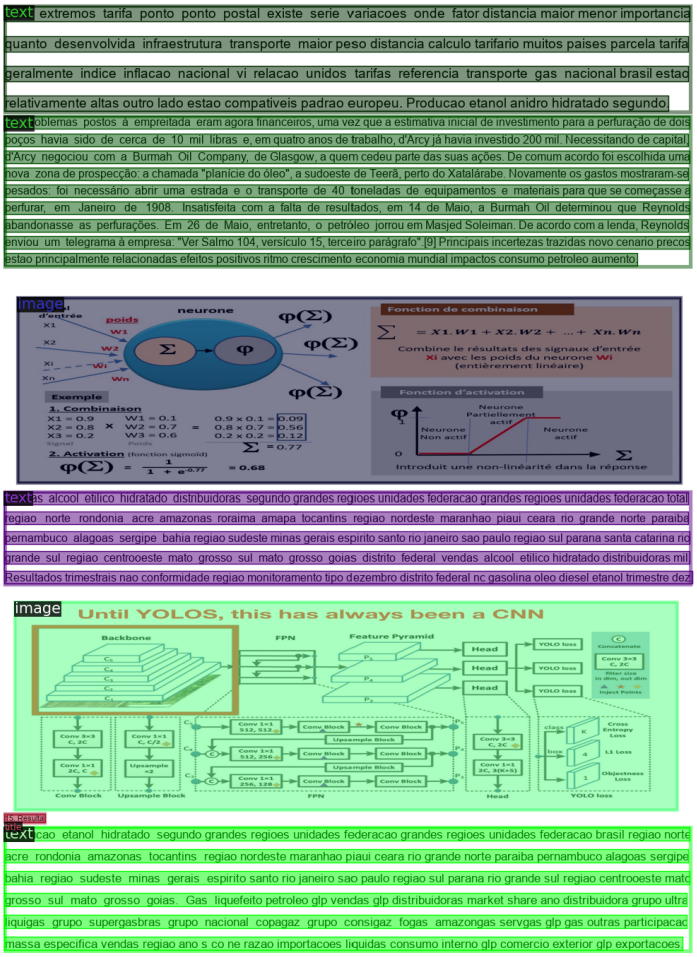

Figura 7.10: Exemplo de visualização das delimitações em documentos dintéticos usando Detectron2 (4).

As Tabelas 7.1, 7.2 e 7.3 apresentam, respectivamente, os principais parâmetros utilizados no treinamento dos modelos Mask R-CNN, Mask DINO e RT-DETR em nossos experimentos.

	Modelo : Mask R-CNN
Hiperparâmetros	**Valores**
Épocas	100
Tamanho do lote	32
RoI por imagem	512
Taxa de aprendizado base	0.0001
Otimizador	SGD
Momentum	0.9
Parada anticipada	Após 10 épocas
Número de GPU	8 Tesla V100-SXM2-32G

Tabela 7.1: Hiperparâmetros ulizados pelo modelo Mask R-CNN.

	Modelo : Mask DINO	
Hiperparâmetros	**Valores**	
Épocas	100	
Tamanho do lote	8	
Taxa de aprendizado base	0.0001	
Número de consultas	300	
Número de camada no codificador	6	
Número de camadas no decodificado	9	
Peso de Perda de classe	4	
Peso de Perda de Bbox	Peso de Perda de GIoU	2
	Peso de Perda L1	5
Peso de Perda de mascara	Peso de Perda de Entropia cruzada	5
	Peso de Perda Dice (Dice loss)	5
Número de GPUs	8 Tesla V100-SXM2-32G	

Tabela 7.2: Hiperparâmetros ulizados pelo modelo Mask DINO.

	Modelo : RT-DETR
Hiperparâmetros	**Valores**
Épocas	100
Otimizador	AdamW
Taxa de aprendizado base	0.0001
Taxa de aprendizado do backbone	0.00001
nheads	8
Número de camadas no decodificado	6
Número de consultas	300
Tamanho do lote	8
Dimensão do vetore de entrada (embedding)	256
Dimensão da rede Feedforward	1024
Peso de Custo do Bbox	5
Peso de Perda do Bbox	5
Peso de Custo GIoU	2
Peso de perda GIoU	2
Peso de Custo da Classe	2
Peso de Perda da Classe	1
Decaimento do Peso (weight decay)	0.0001
Parada antecipada (Early stopping)	Após 10 épocas
Número de GPUs	8 Tesla V100-SXM2-32G

Tabela 7.3: Hiperparâmetros ulizados pelo modelo RT-DETR.

8
Referências bibliográficas

[1] Shoaib Ahmed Siddiqui, Muhammad Imran Malik, Stefan Agne, Andreas Dengel, and Sheraz Ahmed. Decnt: Deep deformable cnn for table detection. *IEEE Access*, 6:74151–74161, 2018.

[2] Junaid Younas, Shoaib Ahmed Siddiqui, Mohsin Munir, Muhammad Imran Malik, Faisal Shafait, Paul Lukowicz, and Sheraz Ahmed. Fi-fo detector: Figure and formula detection using deformable networks. *Applied Sciences*, 10(18), 2020.

[3] Madhav Agarwal, Ajoy Mondal, and C. V. Jawahar. CDeC-Net: Composite Deformable Cascade Network for Table Detection in Document Images, August 2020. arXiv:2008.10831 [cs].

[4] Honghong Zhang, Canhui Xu, and Yuanzhi Cheng. Document image object detection algorithm based on transformer and mixed-mlp network. In *2023 IEEE 6th International Conference on Electronic Information and Communication Technology (ICEICT)*, pages 45–50, 2023.

[5] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. PubLayNet: Largest Dataset Ever for Document Layout Analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022, September 2019. ISSN: 2379-2140.

[6] Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. DocBank: A Benchmark Dataset for Document Layout Analysis, November 2020. arXiv:2006.01038 [cs].

[7] Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter W. J. Staar. DocLayNet: A Large Human-Annotated Dataset for Document-Layout Analysis. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3743–3751, August 2022. arXiv:2206.01062 [cs].

[8] Max C. Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. Icdar 2013 table competition. *2013 12th International Conference on Document Analysis and Recognition*, pages 1449–1453, 2013.

[9] Liangcai Gao, Xiaohan Yi, Zhuoren Jiang, Leipeng Hao, and Zhi Tang. Icdar2017 competition on page object detection. *2017 14th IAPR International*

Conference on Document Analysis and Recognition (ICDAR), 01:1417–1422, 2017.

[10] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, and Hadar Averbuch-Elor. READ: Recursive Autoencoders for Document Layout Generation, April 2020. arXiv:1909.00302 [cs].

[11] Diego Martin Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation, 2021.

[12] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry Davis, Vijay Mahadevan, and Abhinav Shrivastava. LayoutTransformer: Layout Generation and Completion with Self-attention. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 984–994, Montreal, QC, Canada, October 2021. IEEE.

[13] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Generating graphic layouts with wireframe discriminators, 2019.

[14] Sanket Biswas, Rajiv Jain, Vlad I. Morariu, Jiuxiang Gu, Puneet Mathur, Curtis Wigington, Tong Sun, and Josep Lladós. DocSynthv2: A Practical Autoregressive Modeling for Document Generation, June 2024. arXiv:2406.08354 [cs].

[15] Liu He, Yijuan Lu, John Corring, Dinei A. F. Florêncio, and Cha Zhang. Diffusion-based document layout generation. In *IEEE International Conference on Document Analysis and Recognition*, 2023.

[16] Natraj Raman, Sameena Shah, and Manuela Veloso. Synthetic document generator for annotation-free layout recognition. *Pattern Recognition*, 128:108660, August 2022.

[17] Sanket Biswas, Pau Riba, Josep Lladós, and Umapada Pal. Docsynth: A layout guided approach for controllable document image synthesis. In *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part III*, page 555–568, Berlin, Heidelberg, 2021. Springer-Verlag.

[18] B. Mehlig. *Machine learning with neural networks*. October 2021. arXiv:1901.05639 [cond-mat, stat].

[19] What is a Neural Network? | IBM, October 2021.

[20] Ivan Liljeqvist. The Essence of Artificial Neural Networks, March 2016.

[21] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006.

[22] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation Functions: Comparison of trends in Practice and Research for Deep Learning, November 2018. arXiv:1811.03378 [cs].

[23] Mohammed Amine El Mrabet, Khalid El Makkaoui, and Ahmed Faize. Supervised Machine Learning: A Survey. In *2021 4th International Conference on Advanced Communication Technologies and Networking (CommNet)*, pages 1–10, Rabat, Morocco, December 2021. IEEE.

[24] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-supervised learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. OCLC: ocm64898359.

[25] Geoffrey Hinton. The Forward-Forward Algorithm: Some Preliminary Investigations, December 2022. arXiv:2212.13345 [cs].

[26] avcontentteam. Regression vs Classification in Machine Learning Explained!, May 2023.

[27] Sebastian Ruder. An overview of gradient descent optimization algorithms, June 2017. arXiv:1609.04747 [cs].

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[29] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. On Empirical Comparisons of Optimizers for Deep Learning, June 2020. arXiv:1910.05446 [cs, stat].

[30] Li Shen, Congliang Chen, Fangyu Zou, Zequn Jie, Ju Sun, and Wei Liu. A unified analysis of adagrad with weighted aggregation and momentum acceleration, 2023.

[31] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks, December 2015. arXiv:1511.08458 [cs].

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[34] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[35] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, January 2018. arXiv:1603.07285 [cs, stat].

[36] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. pages 92–101, January 2010.

[37] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. FLATTENED CONVOLUTIONAL NEURAL NETWORKS FOR FEEDFORWARD ACCELERATION. 2015.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. arXiv:1512.03385 [cs].

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[40] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach*. Pearson, 3 edition, 2009.

[41] Gobinda G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aris.1440370103.

[42] Xiaoyan Zhao, Yang Deng, Min Yang, Lingzhi Wang, Rui Zhang, Hong Cheng, Wai Lam, Ying Shen, and Ruifeng Xu. A Comprehensive Survey on Relation Extraction: Recent Advances and New Frontiers. *ACM Computing Surveys*, 56(11):1–39, November 2024.

[43] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition, 2014.

[44] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer,

Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. arXiv:2010.11929 [cs].

[45] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, January 2021. arXiv:2012.12877 [cs].

[46] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers, May 2020. arXiv:2005.12872 [cs].

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013. arXiv:1301.3781 [cs].

[48] Ross Girshick. Fast R-CNN, September 2015. arXiv:1504.08083 [cs].

[49] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection, May 2016. arXiv:1506.02640 [cs].

[50] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, January 2016. arXiv:1506.01497 [cs].

[51] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. DETRs Beat YOLOs on Real-time Object Detection, April 2024. arXiv:2304.08069 [cs].

[52] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions, September 2014. arXiv:1409.4842 [cs].

[53] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep Metric Learning via Lifted Structured Feature Embedding, November 2015. arXiv:1511.06452 [cs].

[54] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization, December 2015. arXiv:1512.04150 [cs].

[55] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. volume 9905, pages 21–37. 2016. arXiv:1512.02325 [cs].

[56] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection, February 2018. arXiv:1708.02002 [cs].

[57] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context, February 2015. arXiv:1405.0312 [cs].

[58] Sophia Keyner, Vadim Savenkov, and Svitlana Vakulenko. Open Data Chatbot, September 2019. arXiv:1909.03653 [cs].

[59] Tobias Bauer, Emre Devrim, Misha Glazunov, William Lopez Jaramillo, Balaganesh Mohan, and Gerasimos Spanakis. #MeTooMaastricht: Building a chatbot to assist survivors of sexual harassment, September 2019. arXiv:1909.02809 [cs].

[60] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN, January 2018. arXiv:1703.06870 [cs].

[61] Ranajit Saha, Ajoy Mondal, and C. V. Jawahar. Graphical object detection in document images. *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 51–58, 2019.

[62] C. Lawrence Zitnick and Piotr Dollár. Edge Boxes: Locating Object Proposals from Edges. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8693, pages 391–405. Springer International Publishing, Cham, 2014. Series Title: Lecture Notes in Computer Science.

[63] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171, September 2013.

[64] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, April 2015. arXiv:1409.1556 [cs].

[65] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks, November 2013. arXiv:1311.2901 [cs].

[66] Sven Kosub. A note on the triangle inequality for the jaccard distance, 2016.

[67] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection.

[68] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: DYNAMIC ANCHOR BOXES ARE BETTER QUERIES FOR DETR. 2022.

[69] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. DN-DETR: Accelerate DETR Training by Introducing Query DeNoising.

[70] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, August 2021. arXiv:2103.14030 [cs].

[71] J Macqueen. SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS. *MULTIVARIATE OBSERVATIONS*.

[72] Danny Matthew SAPUTRA, Daniel SAPUTRA, and Liniyanti D. OSWARI. Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method. In *Proceedings of the Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019)*, pages 341–346. Atlantis Press, 2020.

[73] Rena Nainggolan, Resianta Perangin-angin, Emma Simarmata, and Astuti Feriani Tarigan. Improved the Performance of the K-Means Cluster Using the Sum of Squared Error (SSE) optimized by using the Elbow Method. *Journal of Physics: Conference Series*, 1361(1):012015, November 2019.

[74] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[75] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, January 2015. arXiv:1409.0575 [cs].

[76] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. volume 06, 06 2006.

[77] David M. W. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation, October 2020. arXiv:2010.16061 [cs, stat].

[78] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[79] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The Earth Mover's Distance as a Metric for Image Retrieval.

[80] Arthur Stéphanovitch, Ugo Tanielian, Benoît Cadre, Nicolas Klutchnikoff, and Gérard Biau. Optimal 1-wasserstein distance for wgans, 2023.

[81] Jannis Chemseddine, Paul Hagemann, Gabriele Steidl, and Christian Wald. Conditional wasserstein distances with applications in bayesian ot flow matching, 2024.

[82] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.