ÁLAN GONZALEZ MEGER ZANGRANDI

IDENTIFICAÇÃO DE REGIÕES DE TEXTO EM JORNAIS HISTÓRICOS GERMANO-BRASILEIROS UTILIZANDO REDE NEURAL YOLO

(*versão pré-defesa, compilada em 11 de julho de 2019*)

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Daniel Weingaertner.

CURITIBA PR

2019

RESUMO

O objetivo desse trabalho é analisar a viabilidade do YOLO na identificação de regiões de texto em jornais Germano-Brasileiros históricos. Usualmente esse tipo de problema envolve técnicas baseadas em redes neurais e deep learning como CNNs, uma categoria na qual a *darknet* pertence. Diferentes abordagens para a categorização de objetos foram consideradas, e a que mostrou mais sucesso, blocos de texto, foi aprofundada com técnicas de *data augmentation* para gerar melhores resultados. O resultado foi uma rede que apresenta uma *precision* de no máximo 0,91 e um *recall* de até 0,62. O objetivos foram atingidos apenas parcialmente, visto que a rede demonstrou resultados que não superam outros algorítimos modernos de identificação de layout.

Palavras-chave: YOLO.

ABSTRACT

The objective of this work is to analize the viability of YOLO on the identification of text regions in historical journals. Usually those types of problems involve techniques based around neural networks and deep learning, such as CNNs, in which YOLO makes use of. Different approches for the text detection were considered, and the with the biggest success, bounding boxes on text blocs, were further explored with data augmentation techniques to generate better results. The results were a network that show a precision value of 0.91 and a recall value of 0.62. The network only partially acomplished the intended objective, since it didn't show results that surpassed current state of art model.

Keywords: YOLO. Deeplearning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

CNN	Redes Neurais Convolucionais
YOLO	You only look once
FCN	Fully Convolutional Network

SUMÁRIO

1 INTRODUÇÃO

Jornais históricos sempre apresentaram dificuldade para analise de layout. Isso se deve a vários fatores, como a qualidade das folhas escaneadas e fontes não padronizadas. Com o advento da era digital, a capacidade da humanidade de guardar e replicar informação nunca foi tão grande. Para preservar e catalogar o conhecimento humano, é necessário não somente olhar para o presente, mas também o passado que possibilitou que tudo isso ocorra. Apenas digitalizar jornais históricos não é suficiente, devido à grande quantidade de material disponível, linguagem arcaica e um layout e fonte não padronizado. Uma rede capaz de identificar regiões de texto em folha seria um primeiro passo para permitir que o conhecimento existente nesses jornais seja padronizado e processado para o mundo moderno.

1.1 OBJETIVOS GERAIS

O objetivo desse trabalho é desenvolver um programa computacional que identifique regiões de texto em documentos históricos utilizando a rede neural YOLO.

1.2 OBJETIVOS ESPECÍFICOS

- Identificar e extrair as características das imagens para o treinamento da rede
- Realizar *data agumentation* para expandir o dataset existente
- Treinar a rede *darknet* com as características extraídas
- Dado a imagem digital de uma folha de jornal, classificar regiões de texto existentes na página.

2 FUNDAMENTOS TEÓRICOS

2.1 INTRODUÇÃO

Nesse capitulo será introduzido a base teórica do trabalho proposto, em particular, o funcionamento da rede YOLO e os componentes. Esse conhecimento é de vital importância para entender as vantagens e desvantagens de usar algoritmos de aprendizagem.

2.2 REDES NEURAIS CONVOLUCIONAIS

Um dos primeiros passos para compreender o framework YOLO e o motivo para o seu uso nesse trabalho é entender o que exatamente é uma rede neural convolucional (CNN).

Uma CNN é um algoritmo de classificação que busca imitar a forma pela qual seres humanos reconhecem objetos no meio ambiente, atribuindo a um objeto características que o compõem. As CNNs se mostraram uma ferramenta extremamente poderosa na área de reconhecimento de imagens. Por exemplo, CNNs podem ser utilizadas para identificar faces em fotografias ou placas de carro em uma via movimentada. LeCun et al. (1999).

Uma *pipeline* de um CNN começa com uma imagem de entrada, e seu objetivo é extrair as características representativas de uma imagem ao longo da rede, e para isso, a rede irá reduzir a imagem de entrada para uma forma mais simples de processar, sem perder suas características críticas. Essa imagem irá passar por diversos filtros, também chamados de camadas convolucionais. Cada filtro consiste de uma matriz quadrada com tamanho predeterminado pelo algoritmo (filtros de menor tamanho consideram uma maior quantidade de características por área da imagem). Esses filtros irão deslizar pela imagem, aplicando o produto escalar entre o filtro e o pedaço da imagem em que se encontra, repetindo essa ação até o filtro passar por todos os pontos da imagem de entrada.A imagem 3.1 demonstra como a camada convolucional funciona. O resultado é uma imagem que contêm apenas as características extraídas.

As camadas convolucionais geralmente são seguidas por camadas de agrupamento(*pooling*). A função dessas camadas é diminuir o tamanho da imagem de entrada com o objetivo de reduzir o uso de memória e parâmetros utilizados e controlar *overfitting*. A técnica de *pooling* mais comum é o *max pooling*, onde dada uma janela deslizante sobre um pedaço da imagem, o maior valor é escolhido, diminuindo o tamanho da imagem, porém mantendo as características mais relevantes. Por exemplo, dado um filtro $2x2$ com um passo de tamanho 2, dado uma imagem de tamanho nxn, o resultado será uma imagem de tamanho $\frac{n}{2}x\frac{n}{2}$, contendo apenas os máximos de cada filtro. A figura 2.2 demonstra uma versão mais completa do processo de uma CNN.

A CNN pode passar por múltiplas camadas convolucionais e *pooling* para melhor filtrar as características escolhidas e por fim, as imagens resultantes, que deverão representar apenas

Figura 2.1: Demonstração de como os filtros funcionam em camadas convolucionais

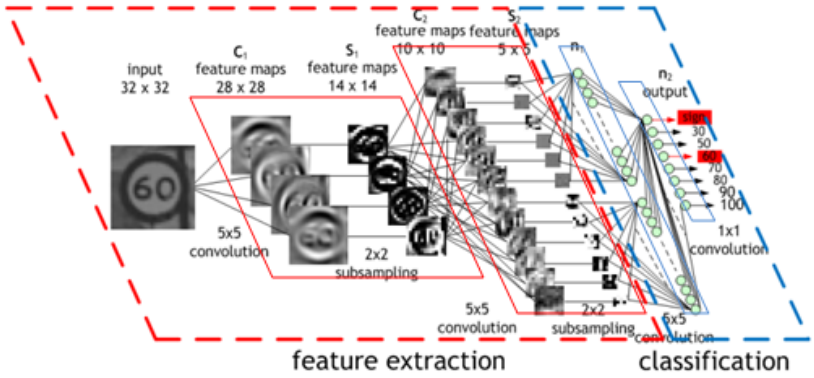


Figura 2.2: Demonstração do que ocorre com uma imagem que entra em uma CNN

uma única característica, são achatadas em um vetor $nx1$ e interpretados por camadas totalmente conectadas, que trabalham como uma rede neural tradicional. O treinamento de uma CNN funciona como uma rede neural regular, através de *backpropagration* de sua camada totalmente conectada. Após uma primeira rodada, os pesos são reajustados e o processo se inicia novamente. O tempo de treinamento de uma CNN varia de acordo com a complexidade do problema proposto, o número de classes, o hardware utilizado no treino e a avaliabilidade de pesos já ajustados para o problema proposto para aumentar a velocidade. A figura 2.3 demonstra um exemplo da uma CNN *AlexNet*, composta por 5 camadas convolucionais e 3 camadas totalmente conectada

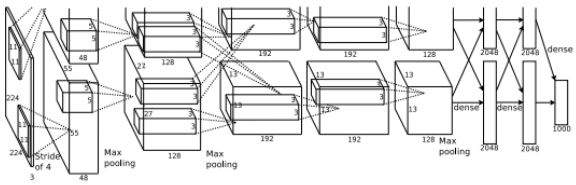


Figura 2.3: Exemplo da CNN *AlexNet*, composta por 5 camadas convolucionais com camadas de agrupamento intercaladas e 3 camadas totalmente conectada

Existem vários *frameworks* disponíveis para se trabalhar com CNNs. Um dos mais comuns, o *TensorFlow* desenvolvido pela *Google*, é reconhecido pelas sua habilidades no reconhecimento de linguagens e classificação/sumarização de textos, o *Caffe*, excede na velocidade de treinamento, o CNTK, desenvolvido pela Microsoft, um framework opensource com capacidades similares ao *TensorFlow*, mas com uma melhor escalabilidade e performance e sem um suporte para arquitetura ARM, limitando o uso em aparelhos móveis. O *pyTorch* é usado principalmente pelas grandes plataformas de mídias sociais, é considerado um forte competidor do *TensorFlow*, com um alto nível de adoção na comunidade de *deep learning*. O *MXNet*, usado pela *Amazon*, aceita um grande número de linguagens, permitindo ao programador trabalhar com o que lhe é de maior familiaridade.

2.3 YOLO

You only look once (YOLO) é um framework para a detecção de objetos, que usa a rede *darknet*, uma CNN, para predizer simultaneamente a bounding box de um objeto e a probabilidade de sua classe. Uma das suas principais características é a sua velocidade na predição de classes, devido a sua simplicidade, permitindo identificação de objetos em tempo real. Redmon et al. (2016)

A *darknet* é composto por 24 camadas convolucionais com camadas de agrupamento intercaladas, seguida por 2 camadas totalmente conectadas. Diferente de métodos como janela deslizantes ou métodos baseados em regiões, o YOLO considera a imagem inteira durante a fase de treinamento e teste, codificando implicitamente informações contextuais sobre as classes observadas.

Para observar a imagem inteira, o algoritmo divide a imagem em uma grade de *SxS* pedaços idênticos, e cada célula é responsável pela identificação de objetos cujo centro se encontram nela. Cada célula da grade prediz *N bounding boxes* e a pontuação de confiança de cada bounding boxes(caso nenhum objeto se encontre na *bounding boxes*, a pontuação é 0), a imagem 2.4 melhor demonstra esse processo. A pontuação de confiança é definida por $Pr(objeto) * IoU^p red$, sendo IoU a intersecção sobre união, uma métrica de avaliação comumente utilizada para determinar a acurácia de um detector de objetos. Ela é obtida através da formula $IoU = \frac{AreadeIntersecao}{readeUnio}$ Cada *bounding box* consiste de 3 predições: as coordenadas da *bounding box*, seu tamanho e a pontuação de confiança. Cada célula também prediz *C* classes condicionais, ou seja, a probabilidade de uma determinada classe é condicionada à sua posição na imagem.

Geralmente o treinamento no YOLO é feito com um peso já pré treinado para reduzir o tempo necessário, no entanto esse peso foi gerado utilizando o conjunto de imagens do *ImageNet* (**?**), e sua aplicação é limitada para casos não genéricos. Para se evitar *overfitting*, uma camada de dropout com uma taxa de 0,5 é introduzida no final. O treinamento também naturalmente realiza *data agumentation*, através de um processo de escala e translação aleatória de até 20%

da imagem original, e modificações na saturação da imagem em um fator de até 1,5 no espaço HSV. Apesar da utilidade do YOLO na identificação de objetos em imagens em um tempo rápido, algumas limitações podem ser observadas no algoritmo, devido as fortes restrições no número de *bounding boxes* e a quantidade de classes permitidas por quadrante o YOLO tem dificuldades de identificar objetos pequenos que se encontram agrupados. Dificuldades também surgem em objetos com proporções ou configurações anômalas. Essas deficiências foram aliviadas com a mais recente versão do YOLOv3, que reduziu o tamanho da grade e aumentou o número de camadas de convolução para melhor identificar objetos pequenos. Redmon e Farhadi (2018)

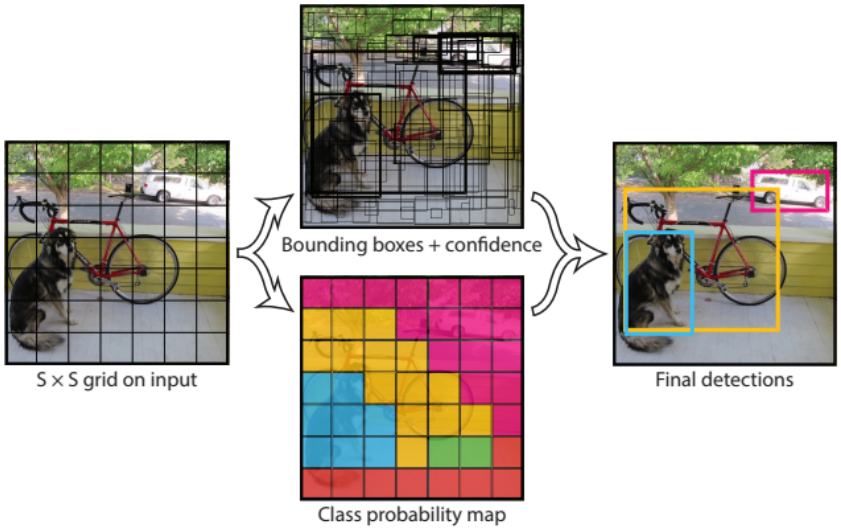


Figura 2.4: Demonstração de como o YOLO divide a imagem em uma grade 7x7 e determina as classes baseado em um mapa de probabilidades e nas bouding boxes geradas

Para começar o treinamento de rede YOLO, primeiro é necessário preparar as imagens de treino. A rede interpreta a *bounding box* de forma retangular com dois valores, a posição do centro do objeto e sua largura e altura, ambos são descritos como um valor entre [0..1], ou seja, os valores são normalizados. Um arquivo simples de texto deve detalhar o diretório onde as imagens de treino e teste se encontram. No arquivo detalhando as configurações do YOLO o número de filtros na ultima camada deve ser reajustado de acordo com o número total de classes. Esse número varia dependendo da versão em uso, no caso do YOLOv2, esse número seria $f = (n * classes) + 5$. A quantidade de classes também deve ser indicada no arquivo de configurações. É recomendado (apesar de opcional) também modificar o valor das ancoras das *bounding boxes*, valores que permitem o YOLO estimar a proporção das *bounding boxes* presididas, aumentando o IoU final. Essas ancoras podem ser obtidas pelo mesmo framework na qual a rede YOLO é rodada, e são calculadas usando usando os centroides dos clusters obtidos em uma KNN. É possível controlar a quantidade máxima de iterações e taxa de treinamento possui uma curva logarítmica, ou seja, será observada um grande avanço na conversão no começo, e quando mais próximo da conversão, menos avanços serão observados por iteração. O recomendado é um mínimo de 2000 iterações por classe no treinamento da *darknet*.

O framework YOLO recomendado para usar a *darknet* seria o *fork* da implementação original `https://github.com/AlexeyAB/darknet`. Essa implementação adiciona diversas ferramentas que facilitam o acompanhamento do treinamento e analise dos resultados finais.

3 REVISÃO DE LITERATURA

Com o advento dos computadores pessoais e o aumento da dependência das pessoas no meio digital, a necessidade de digitalizar documentos em papel surgiu naturalmente. No início as técnicas mais usadas requeriam uma fonte e *layout* específicos para digitalização, ou áreas demarcadas manualmente com canetas magnéticas para demarcar áreas de processamento. Esses métodos naturalmente limitavam documentos já existentes ou exigiam um grande esforço manual que muitas vezes seriam inviável para a quantidade de documentos existentes. Os primeiros métodos desenvolvidos se baseavam na observação que regiões de caracteres eram distinguidos do resto da página pela densidade linhas e os espaços em branco entre cada linha.

Esse método é conhecido como *Optical Masking* (Nagy (1968)), e o objetivo era preparar uma máscara de transparência onde as áreas transparentes correspondem a texto. Essa máscara consiste de uma imagem negativa desfocada da página de interesse com uma magnificação 1:1. Ela é criada fisicamente a partir de uma foto da imagem, e para evitar perder caracteres isolados, a máscara é movida lentamente na horizontal e vertical a uma distância equivalente a cinco e meio caracteres respectivamente, resultando em uma máscara que enfatiza o texto às custas da parte gráfica. A folha em questão é digitalizada com a máscara, em tiras de tamanho fixo, e cada tira passa por uma sub-rotina para determinar se as condições que formam áreas de texto são cumpridas. Para isso, são observados a densidade de pontos pretos, e se existe um espaço branco suficiente entre as linhas. Essas regiões eram demarcadas como regiões de texto e passavam por uma segunda rodada escaneamento, com um foco exclusivo em áreas de texto. Esse método, desenvolvido na década de 60, levava em conta a baixa memória e capacidade de processamento de computadores, necessitando de um pré-processamento manual e hardware especializado para a digitalização de regiões de texto, algo que hoje me dia pode ser automatizado. O ideal é um algoritmo que não necessite um trabalho na pré-digitalização.

Métodos mais avançados também levam em conta certos atributos que se esperam de documentos. Por exemplo linhas sempre se encontram na horizontal, a base de caracteres sempre se encontra alinhada e espaço entre caracteres é sempre uniforme. O algoritmo relevante (Nagy et al. (1992)) busca separar pedaços do texto em blocos gramaticais baseados em informações individuais dos píxeis. Um bloco gramatical pode ser subdividido em blocos horizontais e verticais e representado por uma estrutura de árvore. Esses blocos são compostos por estruturas chamadas de *átomos*, compostas por *strings* de píxeis de valores binários (0 para preto e 1 para branco). Esse átomos são divididos em categorias baseados no tamanho de suas strings, e uma cadeia de átomos pretos e brancos interlaçados é chamada de *molécula*. Essas *moléculas* também são dividas em diferentes classes baseadas nos números e tipos de átomos que o compõem. Uma tabela com todos esses valores determina o tipo de estrutura que se encontra em um determinado bloco gramatical, como demonstrado na tabela 3.1.

Tabela 3.1: Exemplos de um perfil sintático

	Titulo	Espaço entre titulo-subtítulo	Subtítulo
Tamanho de átomos pretos	100-160	—	35-52
Tamanho de átomos brancos	4-20	10-40	2-14
Número de átomos	1-4	—	1-6
Número de entidades	1	1	1
Precedência	Título		

Esse método não requer um pré-processamento como o primeiro método demonstrado, no entanto, ainda é necessário determinar as convenções nas quais as páginas se encontram, e padrões de texto não convencionais como fórmulas matemáticas podem atrapalhar na identificação de *layout*.

Até o momento, as técnicas observadas identificam *layouts* de texto com base em regras que assumem como o texto irá se encontrar em uma página. Esse método é simples e um dos mais populares para a identificação de *layout*, no entanto, ele possui alguns fatores limitantes, principalmente com relação ao tipo de páginas que ele consegue identificar. Mudanças no tipo de fonte, qualidade da imagem, tamanho da fonte e *layout*s especiais requerem uma recalibração do algoritmo, impedindo seu uso em contexto mais generalizado. Apesar de convenções modernas na impressão de documentos permitir que esse algoritmos em contexto mais generalizado, o problema ainda existe em publicações históricas, onde a falta de convenções, fontes mais exóticas e baixa qualidade de digitalização atrapalham na identificação de regiões de texto. Para isso, algumas soluções que incorporam *deep learning* serão analisadas. A grande vantagem dessas soluções é que elas permitem ao computador aprender sozinho as convenções de um documento, sem a necessidade de intervenção humana.

Em outro jornal observado, (Wick e Puppe (2018)) uma *Fully Convolutional Network* (FCN), é proposta para identificar regiões de texto em documentos históricos. Uma FCN é uma variante da CNN, cujas camadas convolucionais requerem uma entrada de tamanho fixo, e não possuem as camadas totalmente conectada encontradas normalmente no fim das CNNs, ou seja, os filtros de aprendizado são espalhados por toda a rede neural.

O *dataset* utilizado é uma série de livros históricos (4 no total) com um livro sendo representado por 3 diferentes edições (apesar de conterem o mesmo assunto, a fonte, o *layout* e imagens são todos diferentes). Um total de 860 páginas foram utilizadas no treinamento da FCN.

O treinamento do modelo requer que todas as paginas tenham a mesma dimensão. Todas as imagens são encaixadas em um molde com a proporção fixa de 2/3. Essa página reduzida para o tamanho fixo de 260x390 e binarizada, reduzindo a profundidade das imagens, diminuindo o tamanho da rede e reduzindo o tempo computacional. A arquitetura proposta pela equipe consiste de duas estruturas: uma codificador e um decodificador. A estrutura de codificação consiste de camadas convolucionais seguidas por camadas de ReLU e *maxpooling*. Ela é seguida pela estrutura de decodificação, que consiste de camadas de deconvolução e ReLU. Essas camadas

efetuam o processo inverso de uma camada de convolução. A imagem 3.1 demonstra a FCN usada.

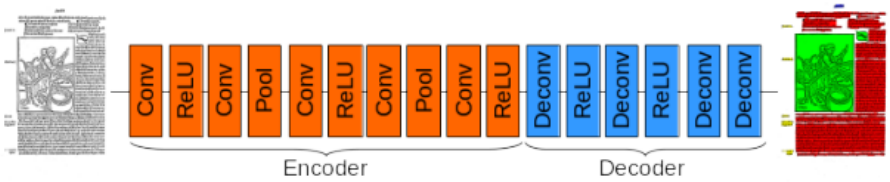


Figura 3.1: Descrição da FCN utilizada

O resultado é uma máscara de rótulos que deve ser multiplicada com a imagem binarizada para se obter o resultado final. A acurácia do trabalho varia entre 98,4% e 92,1%, dependendo do *dataset*. Livros com *layout* mais simples obtiveram uma melhor acurácia. Aqueles com piores resultados não demonstraram uma melhora mesmo com o aumento do tamanho de *dataset*, devido a erros no próprio ground truth. Por fim uma etapa de pós processamento reduz a quantidade de rótulos em uma página usando componentes conectados para unir rótulos similares.

O foco do trabalho Chen e Seuret (2017) é utilizar uma CNN para segmentação de páginas históricas manuscritas. A rotulação é considerada um problema de pixel, ou seja, píxeis são classificados individualmente. A CNN desenvolvida é uma arquitetura simples com uma única camada de convolução.

O pré-processamento do algoritmo começa com um algoritmo para geração de superpixels. Um superpixel é um grupo de píxeis conectados que representam um único objeto. Isso permite classificar apenas o centro de um superpixel, com todos os outros píxeis representados por esse pixel, reduzindo o custo computacional e melhorando os resultados obtidos. Um algoritmo linear simples de *clustering* iterativo(SLIC)Achanta et al. (2010) é utilizado para determinar esses superpíxeis.

A arquitetura CNN utilizada consiste de uma única camada de convolução com 4 *kernels* de tamanho 3x3. A entrada é uma imagem em tons de cinza de tamanho 28x28. Essa camada se conecta diretamente com a camada totalmente conectada, ou seja, camadas de *pooling* não são utilizadas. No treinamento da CNN, pedaços da imagem são extraídos com base nos centros dos superpíxeis, no tamanho determinado pela arquitetura da CNN.

Um total de 6 fontes diferentes foram utilizadas para formar o *dataset*, cada fonte distinta uma da outra, totalizando 110 páginas. Os papéis escolhidos consistem de manuscritos escritos em diferentes línguas. Os resultados obtidos variam entre 86% e 91% dependendo do *dataset*, uma performance comparável à outros métodos, apesar da simplicidade da CNN empregada.

Nesse capítulo foram analisados diferentes formas de solucionar o problema de identificação de regiões de texto em documentos. Pode-se observar um avanço natural na área, com a constante redução da intervenção humana, onde os métodos mais antigos utilizam pressuposição de como um texto deve ser formatado, enquanto os métodos mais novos automatizam esse processo com redes de *deep learning*, tais como CNNs e suas variantes, ferramentas poderosas para a solução desse problema. A rede *darknet* proposta nesse trabalho ainda não foi utilizada

nesse problema específico, mas ela pode ser observada como uma possível tangente a ser explorada.

4 IDENTIFICAÇÃO DE REGIÕES DE TEXTO EM JORNAIS HISTÓRICOS GERMANO-BRASILEIROS UTILIZANDO REDE NEURAL YOLO

4.1 INTRODUÇÃO

O YOLO é um *framework* recente que tem mostrado grande potencial na identificação de objetos, no entanto, grande parte de seu uso ocorre em aplicações em tempo real e em cenários não controlados. O uso do YOLO na identificação de regiões de texto ainda não foi documentado. O objetivo desse trabalho é analisar a viabilidade do YOLO nesse caso específico, identificando quais métodos são os mais eficazes e com se comparam com o estado da arte, e para isso será definido qual o tipo de objeto que a *darknet* quer identificar cujo resultado melhor se aproxima do objetivo proposto. O fluxograma 4.1 reflete o fluxo do projeto.

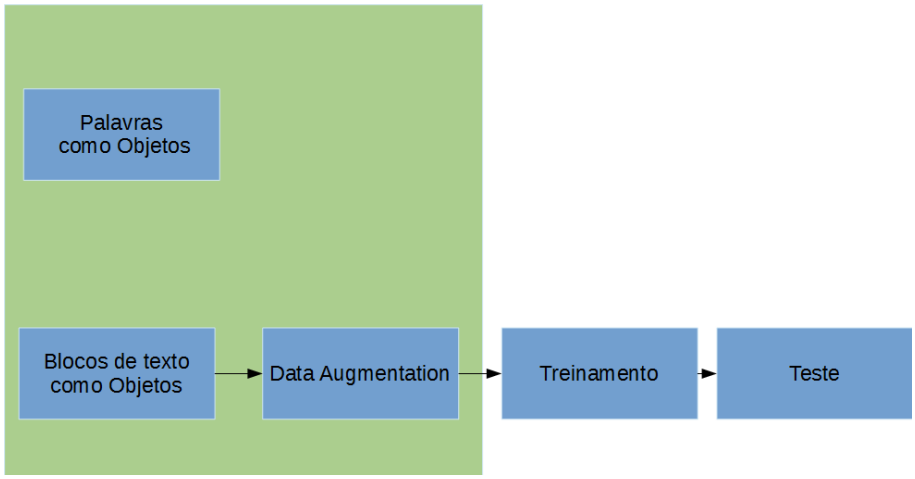


Figura 4.1: Fluxograma simples do processo completo

4.2 EXTRAÇÃO DE CARACTERÍSTICAS

Para se alcançar o objetivo de identificar regiões de texto, o YOLO precisa de informações sobre as regiões de texto que se deseja extrair, ou seja, é necessário definir qual objeto que o *framework* deve identificar. Para se alcançar o objetivo, as regiões de texto devem sem repartidas em objetos que serão detectados pelo programa. Dois métodos para a identificação de regiões de texto foram considerados, palavras individuais serão consideradas como objetos e todas as regiões e o resultado final é a concatenação de todas as palavras encontradas pela rede para se determinar as regiões de texto, e no segundo método blocos de texto, geralmente separados por parágrafos, são os objetos a serem identificados. Ambos os métodos considerados apresentam vantagens e desvantagens que serão explicados mais adiante.

4.2.1 Palavras como objetos

Primeiro foram testados o caso de palavras individuais serem consideradas objetos. O principal motivo para essa decisão é baixo número de páginas disponíveis no começo do projeto (apenas 8 páginas do total de 101). A vantagem desse esquema seria que mesmo um baixo número de imagens geraria um alto número de objetos para o algoritmo de aprendizado,e blocos de texto com formatos exóticos não afetariam o resultado final. Apesar do YOLO possuir dificuldades em identificar objetos pequenos reunidos em clusters, o algoritmo proposto possui apenas uma única classe(texto), amenizando essa limitação do YOLO.

O algoritmo 1 demonstra como as *bounding boxes* foram extraídas dado uma imagem e seu arquivo xml equivalente. O kernel utilizado no código apenas dilata os pixeis na horizontal, de forma a evitar que palavras se conectem na vertical. A figura 4.2 demonstra as *bounding boxes* extraídas.

Figura 4.2: Bouding boxes resultantes

Algoritmo 1 Algoritmo de extração de palavras

```
for all i ← points do
  newimage ← Boudning boxes[i]
  newimage ← dilate[newimage]
  labels ← connectedComponents[newimage]
  for all j ← labels do
    if boudningbox for um valor possível then
      converte coordenadas para o valor padrão
      out ← coordenadas
    end if
  end for
end for
```

Os arquivos resultantes são arquivos texto que denotam as *bounding boxes* da imagem no formato aceito pelo YOLO.

4.2.2 Blocos de texto como objetos

As *bounding boxes* foram determinadas pelo arquivo xml, proveniente do próprio banco de imagens disponíveis, esse arquivo separa as regiões de texto em parágrafos e demarca não só o texto, como também as regiões gráficas e os títulos dos jornais separadamente, ele foi gerado manualmente para o aprendizado da rede *darknet*. A demarcação se encontra na forma de *bounding boxes* com o mesmo sistema de coordenadas do *OpenCV*. A imagem 4.3 demonstra como as *bounding boxes* são geradas nesse método.

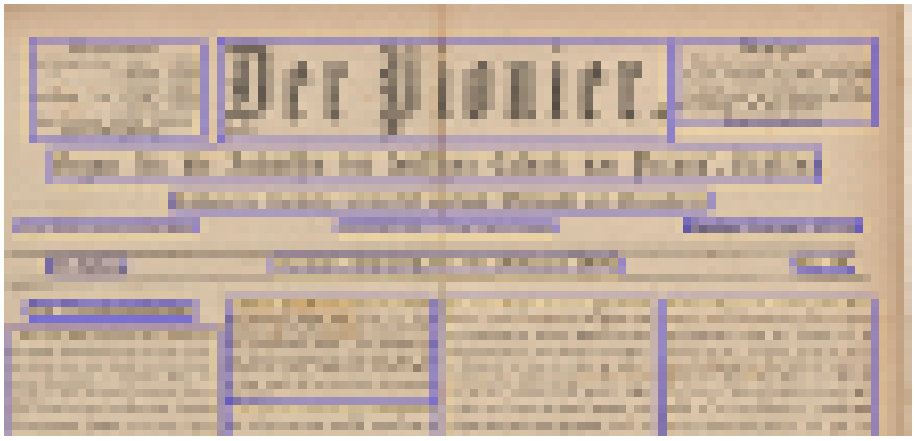


Figura 4.3: *bounding boxes* resultantes

4.2.2.1 Data augmentation

O resultado é uma média de 2-6 objetos por imagem, um número baixo de objetos considerando o total de imagens disponíveis. Para garantir um melhor resultado, técnicas de *data augmentation* são empregadas para aumentar o número de A técnica usada é composto de duas operações básicas: a inversão ou espelhamento de um bloco e a sua troca com outro bloco. O objetivo é gerar uma nova imagem semelhante a original, que não quebre o layout da página. Para que duas regiões de texto possam ser consideradas intercambiáveis, a diferença de sua altura e largura deve ser menor que 10% de seu tamanho total e as transformações são realizadas de maneira aleatória, ou seja, para cada bloco, existe uma chance de 30% que um operação de inversão ou espelhamento será realizada, sendo possível realizar ambas operações no mesmo bloco.

Para cada imagem original, 30 páginas extras foram geradas usando o algoritmo de data agumentation, como pode ser visto na figura 4.4.

4.3 MATERIAIS E MÉTODOS

Um total de 101 páginas de jornais do German-Brazilian Newspapers dataset(GBN), originadas de 8 jornais diferentes. Esse jornais foram introduzidos no Brasil no século XIX e mantidos em circulação até 1940, interrompidos devido a segunda guerra mundial. Eles

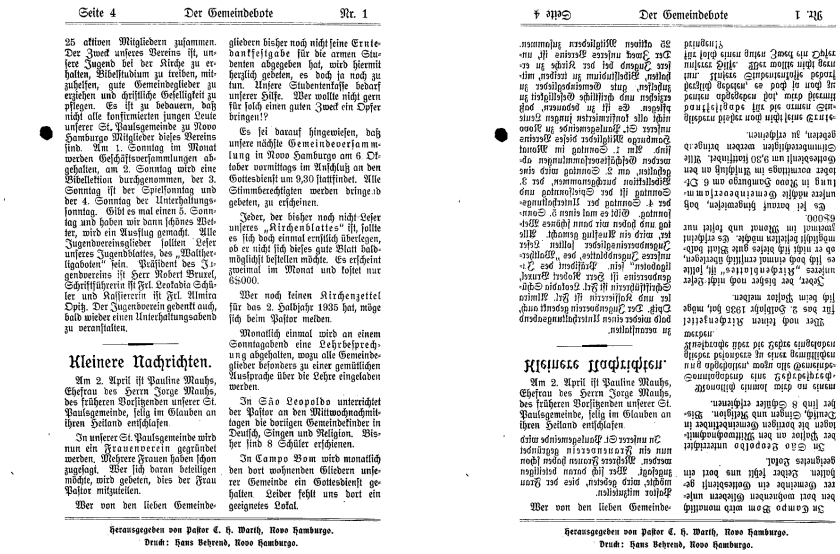


Figura 4.4: Imagem original à direita, e uma versão com caixas invertidas resultante

eram distribuídos em comunidades locais, geralmente de mão em mão, e eram considerados importantes fontes de informação para a comunidade imigrante alemã, no entanto, devido a natureza de sua circulação, muitas poucas cópias foram preservadas. Todos esse jornais se encontram em alta resolução(entre 2590x3690 e 7050x9300) e foram disponibilizados tanto na versão original quanto uma versão binária. Cada folha também é acompanhada de um arquivo xml com as *bounding boxes* de todas as regiões de interesse da folha, incluindo regiões de texto, imagens e separadores.

Para o teste da rede uma total de 15 páginas de 4 jornais diferentes foram usadas. Essas imagens não pertenciam ao arquivo de treino para um resultado que não possua nenhum viés.

Tabela 4.1: PTR indica páginas de treinamento e PTS indica as páginas de teste

Jornal	Tamanho da imagem (em pixels)	PTR	PTS
Der Gemeindebote	3850 x 5480	19	5
Der Jugendfreund	2650 x 3950 à 5320 x 8050	15	3
Der Pioner	7100 x 10590	17	5
Der Sandwirt	4250 x 6020	17	5
Evangelissch-Lutherisches Kirchenblatt	2590 x 3690	17	5
Kolonie Zeitung	5470 x 7010 6700 x 8400 7050 x 9300	17	5
Gemeindeblatt	3850 x 5870	0	8
Heimatbote	3850 x 5480	0	14

4.3.1 YOLO

Todos os experimentos foram efetuados no YOLOv2 e YOLOv3. Foi definido uma única classe(texto) e as âncoras foram calculadas usando k-means para determinar os centroides dos clusters de dados, usando o própio comando do *framework* YOLO. Para os parâmetros do YOLOv2 foram usados:

- *batch* = 64
- *subdivisions* = 64
- *width* e *height* = 608
- *decay* = 0.005
- *max batches* = 20000
- *classes* = 1
- *filters*(linha 237) = 30

Para os parâmetros do YOLOv3 foram usados:

- *batch* = 128
- *subdivisions* = 128
- *width* e *height* = 608
- *decay* = 0.005
- *max batches* = 20000
- *classes* = 1
- *filters*(linha 776) = 18

Para todos os outros parâmetros não mencionados foram usados os padrões. O treinamento foi realizado em uma gpu nvidia GeForce GTX Titan X, com um mínimo de 8000 iterações e um máximo de 20000.

4.3.2 Métricas

Quatro métricas foram utilizadas para se analisar o resultados do trabalho: $Precision = \frac{TP}{TP + FP}$, onde TP são os *True Positives* e FP os *False Positives*, $recall = \frac{TP}{TP + FN}$, onde FN são os *False Negatives* e $F1 - Score = \frac{Precision * Recall}{Precision + Recall}$, a divisão harmônica entre o precision e o recall. O *mean average precision*(mAP) é uma métrica usada em detecção de objetos que representa a média da *area precision*(AP), a área da curva gerado por um gráfico gerada pelo precision-recall.

Essas métricas foram extraídas pelo próprio *framework* YOLO.

4.3.3 Palavras como objetos

Na primeira rodada de teste foram utilizados um total de 8 imagens, cada imagem com uma média de 290 à 500 objetos. A tabela 4.6 demonstra as métricas resultantes para um mínimo de 8000 iterações.

Tabela 4.2: Os valores das métricas em diferentes thresholds no YOLOv3 para palavras como objetos. NaN significa *Not a number*

Thresholding	precision	recall	F1-Score	mAP
`0.10`	NaN	NaN	NaN	0.54
`0.25`	NaN	NaN	NaN	0.74
`0.50`	NaN	NaN	NaN	0.74

Devido a falta de resultados nesse método, ele não foi levado adiante. Os *NaN* demonstram que nenhum resultado foi obtido na rede gerada, e tanto o alto mAP quanto os resultados NaN são um artefato resultante de uma divisão por 0(nesse caso o número de TP e FPs obtidos).

4.3.4 Blocos de texto como objetos

O treinamento no YOLO foi realizado com um total de 3060 imagens, por mais de 50 mil iterações. Os resultados na tabela 4.3 e 4.4. Foram testados tanto o YOLOv2 quanto o YOLOv3, o gráfico 4.5 demonstra a curva de aprendizado no YOLOv2 e o gráfico 4.5 a do YOLOv3, a versão que sacrifica um pouco da velocidade do YOLOv2 para aumentar a quantidade de camadas de convolução e diminuir o tamanho das grades usadas na detecção, aumentando a acurácia em casos com múltiplas *bounding boxes* pequenas agrupadas.

Tabela 4.3: Os valores das métricas em diferentes thresholds no YOLOv3

Thresholding	precision	recall	F1-Score	mAP
`0.10`	0.66	0.58	0.62	0.56
`0.25`	0.81	0.38	0.62	0.56
`0.50`	0.51	0.81	0.58	0.56

Tabela 4.4: Os valores das métricas em diferentes thresholds no YOLOv2

Thresholding	precision	recall	F1-Score	mAP
0.10	0.45	0.70	0.55	0.52
0.25	0.72	0.51	0.60	0.53
0.50	0.82	0.37	0.51	0.52

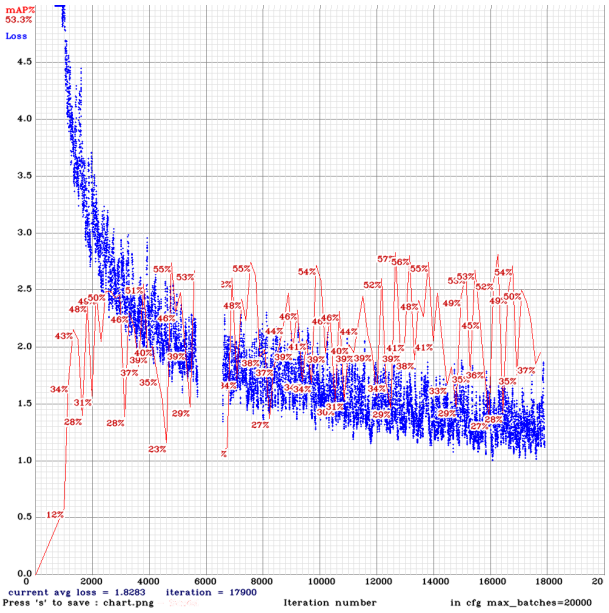


Figura 4.5: Gráfico de aprendizado do YOLOv2, mostrando a queda do avg. loss e a percentagem do mAP com o aumento de iterações

4.4 RESULTADOS

Um total de 14 imagens foram usadas para analisar os resultados. Essas imagens não estavam presentes no arquivo de treino, para evitar qualquer forma de bias

Tabela 4.5: Os valores das métricas em diferentes thresholds no YOLOv3

Thresholding	precision	recall	F1-Score	mAP
0.10	0.79	0.49	0.61	0.53
0.25	0.87	0.45	0.60	0.53
0.50	0.91	0.40	0.55	0.54

Tabela 4.6: Os valores das métricas em diferentes thresholds no YOLOv2

Thresholding	precision	recall	F1-Score	mAP
0.10	0.65	0.62	0.64	0.56
0.25	0.75	0.56	0.64	0.56
0.50	0.86	0.42	0.56	0.56

Apesar das métricas serem um indicativo para a qualidade do programa de *deep learning*, elas não apresentam uma visão completa.Para melhor entender os resultados, é necessário observar alguns casos individuais com o objetivo de analisar quais imagens o programa melhor identifica

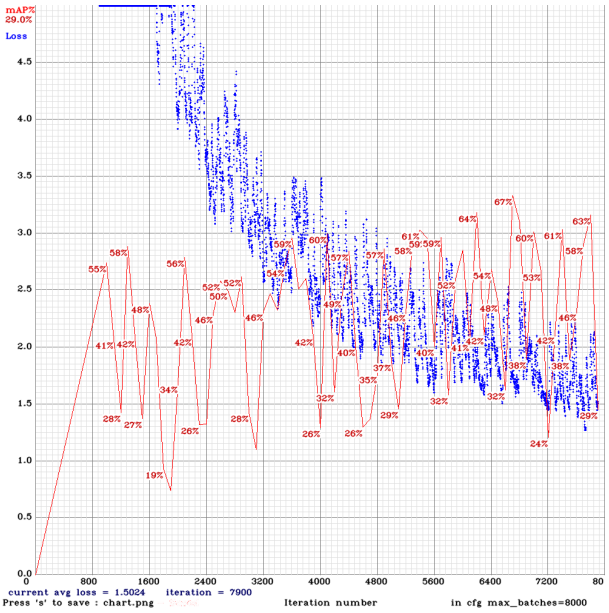


Figura 4.6: Gráfico de aprendizado do YOLOv3, mostrando a queda do avg. loss e a percentagem do mAP com o aumento de iterações

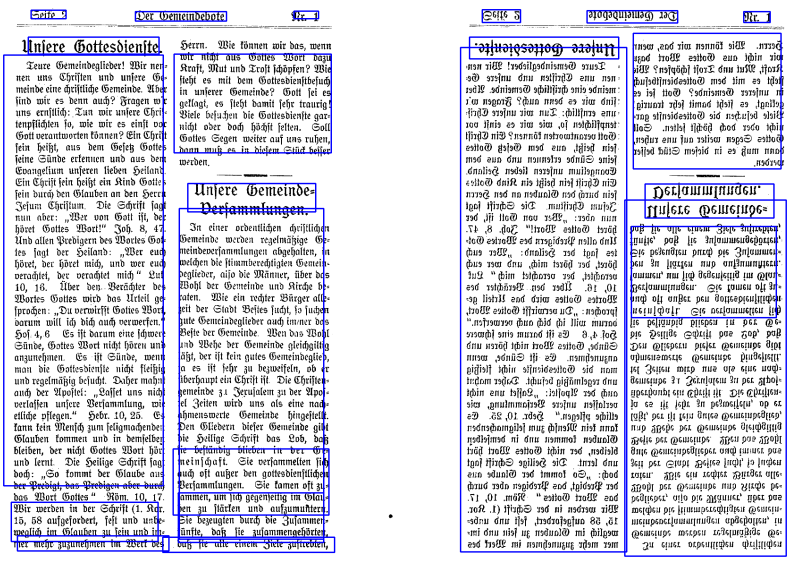
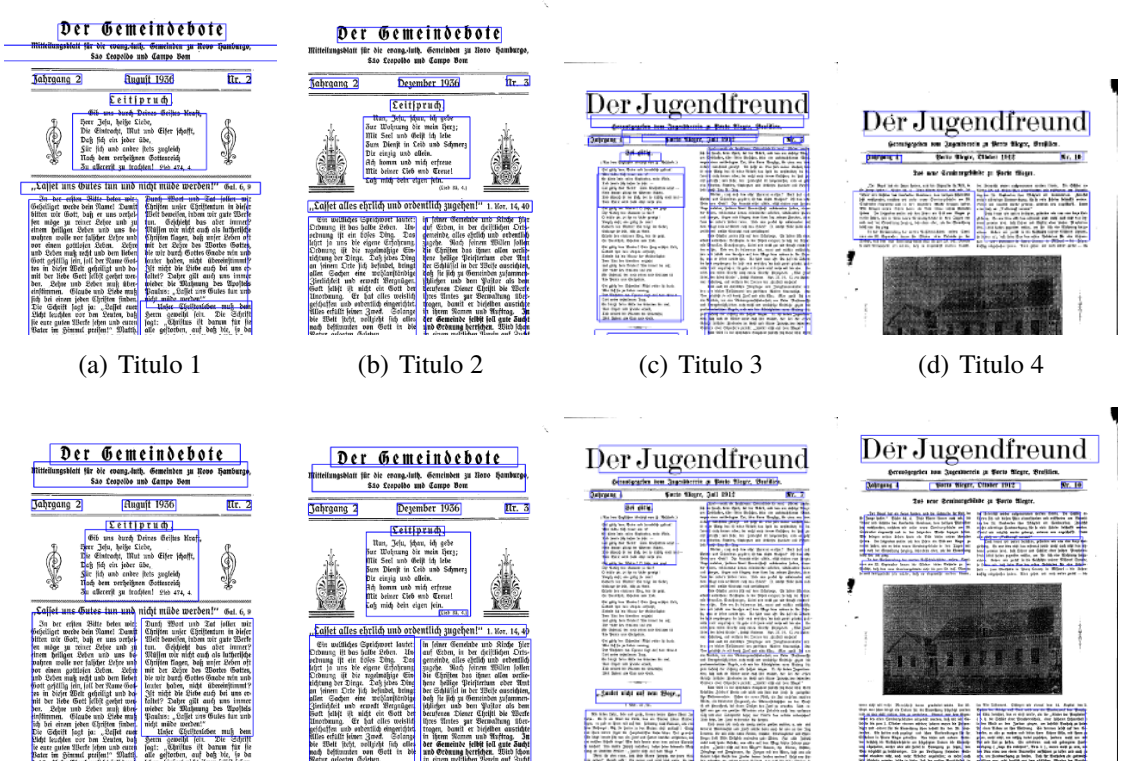


Figura 4.7: Resultados no YOLOv2 à esquerda e no YOLOv3 à direita. Observe que a figura a direita melhor representa as regiões de texto com um menor número de *bounding boxes*

e quais apresentam dificuldades para o YOLO. Na imagem 4.7, praticamente todas as áreas de texto foram identificadas com um alto nível de precisão. A figura referente é uma folha simples que possui apenas áreas de texto, e a grande maioria das páginas do *dataset* possuem um layout similar a esse. Olhando para pedaços específicos de um jornal, podemos notar diferentes níveis de sucesso dependendo do estilo da página e fonte observados. Conforme a figura 4.8, pode-se observar que a rede consegue identificar títulos e fontes grandes com um alto grau de confiança.



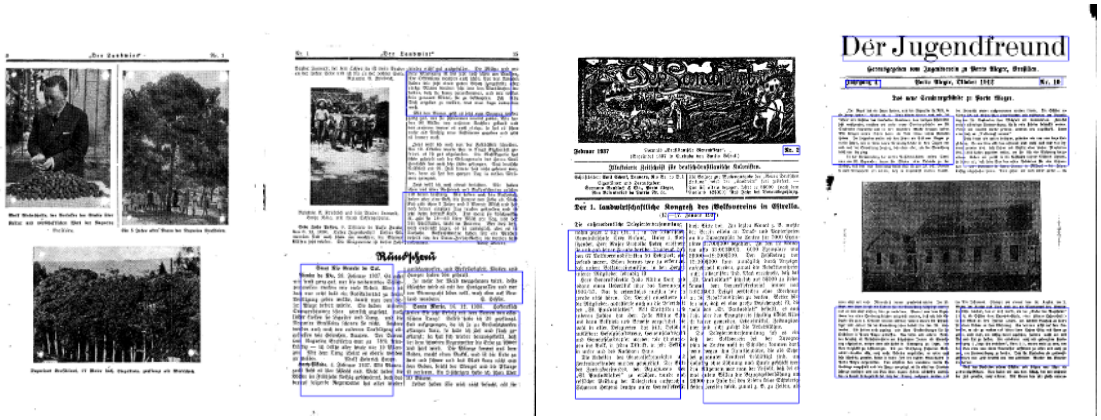
(a) Titulo 1 (b) Titulo 2 (c) Titulo 3 (d) Titulo 4

(e) Titulo 1 (f) Titulo 2 (g) Titulo 3 (h) Titulo 4

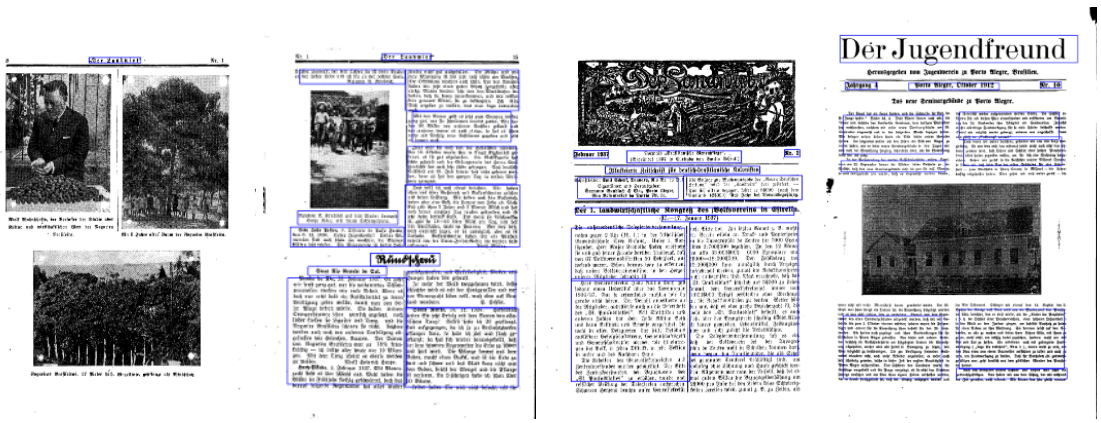
Figura 4.8: As imagens de cima demonstram o sucesso da rede em identificar títulos ou texto com uma maior fonte nas imagens observadas(YOLOV2 na linha superior e YOLOv3 na linha inferior)

A figura 4.9 também demonstra outro ponto importante da rede *darknet* gerada, a quantidade de FPs observados é negligível, ou seja, a rede consegue determinar com grande habilidade quando uma região de imagem não é texto, apesar da rede demonstrar um *recall* de 0.64, o que se deve ao fato da rede ter dificuldades em capturar todas as regiões de texto de uma imagem.

Na figura 4.10, é observado imagens que devolvem piores resultados, e nota-se que quando uma imagem se encontra posicionada dentro de blocos de texto, um exemplo que raramente aparece no *dataset*,a rede tem dificuldade de separar um bloco de textos de sua parte gráfica e blocos de texto que não seguem o padrão normal. Padrões exóticos com por exemplo texto na vertical não são bem representados no *dataset*, e mesmo com *data augmentation* a rede não consegue identificar essas regiões de texto. Essas fraquezas servirão de identificação para possíveis melhorias em trabalhos futuro.

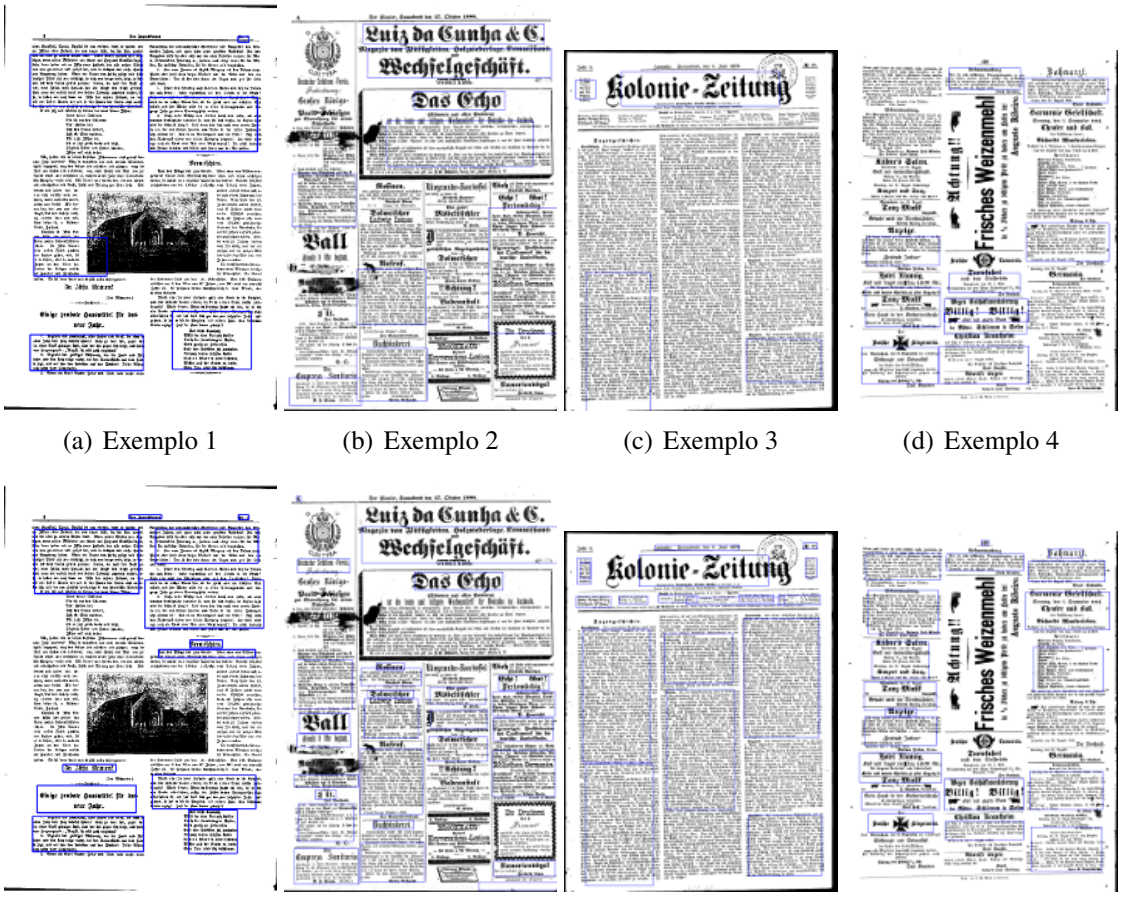


(a) Exemplo 1 (b) Exemplo 2 (c) Exemplo 3 (d) Exemplo 4



(e) Exemplo 1 (f) Exemplo 2 (g) Exemplo 3 (h) Exemplo 4

Figura 4.9: As imagens de cima demonstram que rede não identifica imagens como regiões de texto(YOLOV2 na linha superior e YOLOv3 na linha inferior)



(a) Exemplo 1 (b) Exemplo 2 (c) Exemplo 3 (d) Exemplo 4

(e) Exemplo 1 (f) Exemplo 2 (g) Exemplo 3 (h) Exemplo 4

Figura 4.10: As imagens de cima demonstram situações onde a rede tem dificuldade em identificar regiões de texto devido a vários motivos(YOLOV2 na linha superior e YOLOv3 na linha inferior)

5 CONCLUSÃO

O objetivo desse trabalho foi analisar a viabilidade da rede *darknet* para a localização de regiões de texto em jornais históricos. No início do trabalho foi discutido as ferramentas usadas nesse projeto e a história da identificação de layouts em documentos digitalizados, observado-se diferentes métodos de análise e como as limitações de hardware da época influenciavam os algoritmos propostos. Por fim optou-se pelo uso inédito do *framework* YOLO, mantendo a tendência de se usar métodos de *machine learning* para extrair o layout de jornais.

Foi determinado usar parágrafos completos como objetos para o YOLO, visto que essas abordagem apresentou os melhores resultados, e o treinamento foi realizado em ambos o YOLOv2 quanto no v3, com o YOLOv3 apresentando melhores resultados consistentemente, com uma maior área de identificação e uma menor quantidade de sobreposições.

Apesar da taxa de identificação não superar outros métodos apresentados nesse trabalho, o programa apresentou um nível de sucesso que pode ser trabalhado em cima e o alto nível de precisão na identificação de regiões de texto demonstra o futuro potencial dessa rede. O método demonstrado identifica com sucesso blocos de texto e apresenta dificuldades quando elementos gráficos se encontram dentro de regiões de texto, blocos de texto com formatos atípicos e fontes de baixa resolução.

5.1 TRABALHOS FUTUROS

A rede darknet se mostrou capaz de reconhecer layouts de documentos históricos, apesar de necessitar de alguns ajustes para melhor competir com outros métodos. Para possíveis trabalhos futuros, seria interessante incorporar todas os componentes de uma página e não se limitar a reconhecer regiões de texto e melhorar as técnicas de data agumentation para páginas exóticas. Uma possibilidade seria reduzir o tamanho dos blocos de texto para melhor localizar blocos de diferentes formatos. Dado mais tempo, uma análise aprofundada dos casos em que a rede tem dificuldade irá demonstrar como melhorar o algoritmo.

REFERÊNCIAS

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P. e Süsstrunk, S. (2010). Slic superpixels. página 15.

Chen, K. e Seuret, M. (2017). Convolutional neural networks for page segmentation of historical document images. *CoRR*, abs/1704.01474.

LeCun, Y., Haffner, P., Bottou, L. e Bengio, Y. (1999). Object recognition with gradient-based learning. Em *Shape, Contour and Grouping in Computer Vision*, páginas 319–, London, UK, UK. Springer-Verlag.

Nagy, G. (1968). Preliminary investigation of techniques for automated reading of unformatted text. *Commun. ACM*, 11(7):480–487.

Nagy, G., Seth, S. C. e Viswanathan, M. (1992). A prototype document image analysis system for technical journals. *Computer*, 25:10–22.

Redmon, J., Divvala, S. K., Girshick, R. B. e Farhadi, A. (2016). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Redmon, J. e Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

Wick, C. e Puppe, F. (2018). Fully convolutional neural networks for page segmentation of historical document images. Em *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, páginas 287–292.