FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Ensembles de OCRs para aplicações médicas

João Matos Silva

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Rui Camacho

29 de outubro de 2021

Abstract

With the increasing use of new technologies in research, there is an enormous advantage in extracting data and knowledge stored in traditional media such as books and written records into databases. This transition is necessary because it facilitates all processes that involve the handling and processing of data on a large scale.

One of the cases where this transition is necessary is the case of the "Child and Youth Health Bolentins", which is the case that this dissertation will focus on. These documents contain the information of users from birth to 20 years old. The information is stored in a table per document, and in the following pages the same information is shown in graphs.

There is a great deal of information contained in these bulletins that is of interest to the scientific community to be transposed to digital media, so that it can be used in pediatric studies.

What was aimed in the dissertation is to achieve an automated process through an Optical Character Recognition (OCR) system, associated with machine learning, data mining and also using Ensembles methods, in order to collect the data contained in the bulletins, obtaining the best possible predictive performance of the algorithms used.

Keywords: Optical Character Recognition, Machine Learning, Ensemble methods

Resumo

Com a crescente utilização de novas tecnologias na investigação, torna-se extremamente vantajoso a extracção automática de dados e conhecimento disponível (explícita ou implícita) na informação guardada em meios tradicionais, como livros e registos escritos, para bases de dados. Esta transição é necessária pois facilita todos os processos que impliquem o tratamento e processamento de dados à grande escala.

Um dos casos em que essa transição é necessária, é o caso dos "Bolentins de Saúde Infantil e Juvenil", sendo este o caso que a dissertação vai incidir. Estes documentos contêm a informação de utentes desde que nascem até aos 20 anos. A informação é guardada numa tabela por documento e, nas páginas seguintes, a mesma informação é demonstrada em gráficos.

Existe uma grande quantidade de informação contida nestes boletins que é do interesse da comunidade científica que seja transposta para o meio digital, de forma a poder ser utilizada em estudos pediátricos.

O que foi almejado na dissertação é conseguir um processo automatizado através de um sistema de reconhecimento de caracteres óptico (Optical Character Recognition - OCR), associado a aprendizagem computacional (Machine Learning), mineração de dados e também utilizando métodos de Ensembles, de forma a recolher os dados contidos nos boletins, obtendo a melhor performance preditiva dos algoritmos utilizados possível.

Keywords: Optical Character Recognition, Machine Learning, Ensemble methods

Agradecimentos

Em primeiro lugar, gostava de agradecer ao meu orientador, o professor Rui Camacho, pela disponibilidade e pela boa vontade que teve comigo durante este percurso.

Começo por agradecer às mulheres da minha vida. Quero agradecer à minha mãe, por me ter dado a oportunidade de viver a vida académica, por ter sempre acreditado em mim, por me fazer acreditar em mim, por se sacrificar diariamente para eu ter esta oportunidade e por ser um exemplo do que eu almejo um dia ser. Quero agradecer à minha avó (Mãe), por ter vivido comigo e por me aturar durante os anos iniciais desta jornada. Quero agradecer à minha tia (Titi) por todos os conselhos e por me ter ajudado, sempre que precisei, durante o meu percurso. Quero agradecer às minhas pequenas, Mafalda, Marina e Maria Inês, por me quererem ser alguém melhor, de quem elas se possam orgulhar um dia. Por último, mas não menos importante, agradecer à minha Maria, por me meter juízo e por me manter no caminho certo, pelas discussões, por todos os sorrisos, por se preocupar comigo, por querer que eu dê o melhor de mim e por ser a minha companheira em tudo. O melhor que a FEUP me deu. A todas elas agradeço pelo amor, carinho e paciência que por vezes exijo.

Quero também agradecer ao Aires, pelos conselhos, ensinamentos, e tempo disponibilizado, quando eu necessitei. Agradeço ao Guilherme por me ensinar o que é a FEUP, e por me ensinar o que é um engenheiro. Agradecer ao meu pai, pelo que me ensinou durante o meu percurso até chegar à faculdade. Quero também agradecer ao meu avô, pelas histórias e momentos partilhados nesta fase da minha vida. Quero agradecer ao meu padrinho, à Ariana e ao Martim, por serem sempre uma alegria de convívio e pelos bons tempos. Uma palavra de agradecimento à dona Paula, ao sr Joca e à D.Júlia, pela hospitalidade e por serem sempre uma boa companhia.

Quero também agradecer à Turma 6 + extras: Ruizão, Pedrocas, Sarzedas, Cadavez, e outros que eu eventualmente me esqueci, visto que escrevo isto com meras horas da entrega (como foi costume), agradeço por todos os momentos de risota que passámos e pela ajuda disponibilizada, ao longo do percurso.

Quero também agradecer às pessoas que fizeram parte daquilo que ninguém conhece. Quero agradecer a todos aos que deram do tempo das suas vidas para me tentar ensinar alguma coisa. Garanto que não foi tempo perdido. Quero também agradecer àqueles a quem eu dei tempo da minha vida para lhes tentar ensinar alguma coisa. Espero que não tenha sido tempo perdido.

Por último quero agradecer aos que entraram e saíram comigo: Coentrão, Aquafresh, Suma, Barbot, Jynx, Quiche, Impressora, TAC, Eclair, Sirenes, Badass, Pro-V, Sequoia, Desaparecida, Capucho e Fiception (Lamento se me enganei na ordem, mas da Barbot para frente, nunca foi comigo). Quero agradecer por terem vivido intensamente, durante o tempo que nos deixaram, a vida académica, por nunca me terem deixado na mão e principalmente, por terem acreditado em mim, nos momentos em que foi mesmo necessário. O nosso tempo passou, mas viverá eternamente connosco.

João de Matos Silva

"Started making it.
Had a breakdown.
Bon Appétit."

James Acaster

Conteúdo

Lista de Figuras

Lista de Tabelas

Abbreviations

ANN	Artificial Neural Network
ASCII	American Standard Code for Information Interchange
BSIJ	Boletins de Saúde Infantil e Juvenil
CRIP-DM	CRoss Industry Standard Process for Data Mining
DM	Data Mining
HSV	Hue, Saturation, Value
ML	Machine Learning
OCR	Optical Character
PCA	Principal Component Analysis
PDF	Portable Document Format
SEMMA	Sample, Explore, Modify, Model, and Access

Capítulo 1

Introdução

Este capitulo tem como propósito introduzir o tema abordado nesta dissertação. O foco da dissertação assenta nos Boletins de Saúde Infantil e Juvenil (BSIJ) e a importância que podem vir a ter, se a informação que contêm for propriamente extraditada. Este capitulo introduz tanto a importância dos BSIJ, assim como o trabalho a ser desenvolvido, de forma a viabilizar a informatização destes.

1.1 Contexto

Com os avanços tecnológicos da era moderna, com a explosão da internet, e com o mundo cada vez mais digitalizado, a informação é dada como adquirida. Mas para chegarmos à maior informatização possível ainda é necessário um esforço de modo a converter informação literária física em informação digital.

Em Portugal, existe o BSIJ que tem como propósito registar e acompanhar o desenvolvimento de todas as crianças no país, até ao momento que se tornam adultos. A utilidade deste registo baseia-se nos dados previamente recolhidos e que dão uma indicação do desenvolvimento chamado "regular"de uma criança em função da idade. São feitas medições de peso, altura, perímetro cefálico, índice de massa corporal, bem como relações entre estes. Além disso, os grafismos, onde se registam os valores, fazem-se acompanhar de diversas linhas que indicam uma distribuição normal da população, isto é, o percentil 90 no gráfico indica que apenas 10% da população tem essa medida ou superior, enquanto que o percentil 10 indica que 90% da população atinge essa marca.

É também de salientar que existem BSIJ masculino e feminino, que não apresentam diferenças significativas entre eles, exceptuando nas curvas dos grafismos (que se devem às diferenças biológicas de desenvolvimento entre crianças masculinas e crianças femininas).

Para a informação contida nestes documentos continuar relevante e para que possa ser utilizada pela área da investigação da saúde, no futuro, é necessário transpor essa informação para o mundo digital, não só para ser mais fácil de agrupar dados, mas para que possibilite todo o tipo de documentação estatística. Nos dias de hoje, recorrendo aos meios que temos ao nosso alcance, já

Figura 1.1: Exemplos de BSIJ masculino e feminino, respectivamente.

é possível fazê-lo de forma eficiente, utilizando múltiplos dispositivos e com custo reduzido, algo que não era possível anteriormente.

Hoje, temos ao nosso alcance técnicas como processamento de imagem, *Optical Character Recognition* (OCR) e *Machine Learning* (ML), que tornam um processo, previamente dispendioso, tanto a nível financeiro como a nível horário, num processo relativamente rápido e barato.

O trabalho a ser desenvolvido nesta dissertação, não só se focará no desenvolvimento uma plataforma de OCR, que automaticamente recolha a informação contida nos BSIJ, como na *Ensemble* de diversas técnicas de OCR, de forma a obter a maior taxa de acerto possível.

1.2 Motivação

Até ao momento foram feitos alguns avanços nesta área, mas ainda existe uma larga margem de progresso. A motivação por detrás desta dissertação, entre outras, é o uso que pode ser feito por parte da investigação na área da pediatria e que pode ser usado para investigar os dados que estão escondidos atrás da dificuldade que é analisar dados não informatizados.

1.3 Definição do problema

Como mencionado previamente, o tópico principal deste projeto é a utilização das tecnologias disponíveis para transitar dados registados manualmente em meios tradicionais para bases de dados, amplamente disponíveis para a pesquisa científica, nomeadamente no mundo da pediatria.

Neste momento existe uma grande quantidade de boletins, com dados relevantes para a pesquisa científica, que contêm a informação de desenvolvimento jovem português ao longo de várias décadas. Assim sendo, esta informação é de relevância para estudos no que toca à pediatria em Portugal. No entanto esta informação é de difícil acesso, fazendo com que, de forma a se estudar, seja necessário um processo extenuado e moroso de agrupamento de informação e de dados. Tendo isto em conta, o processo manual de extração de dados pode-se considerar um processo

dispendioso pois pode consumir muito tempo ou então pode exigir que seja necessária mão de obra em demasia. Caso o dados não sejam extraídos, o preço a pagar será a falta de informação que mais tarde poderá a vir ser útil na saúde infantil de Portugal.

Apesar da realidade deste problema, ainda não foi desenvolvida nenhuma solução que consiga resolver o problema de forma eficiente, no entanto, já existem ferramentas que possibilitam o seu desenvolvimento.

1.4 Objetivos

O objetivo principal desta dissertação é desenvolver um sistema que automatize o processo de digitalização, interpretação e armazenamento de dados captados a partir dos Boletins de Saúde Infantil e Juvenil. Para tal ser possível, objetivos mais específicos foram definidos:

- Automatizar o processo de extração e conversão de informação de dados de BSIJs.
- Armazenar os dados convertidos a partir do ponto anterior.
- Concatenar vários métodos de OCR num único sistema, aliados cada um deles a um classificador de base, que permitam avaliar a confiança na conversão do caractere.
- Desenvolver um "meta-decisor", que tenha em conta todos os classificadores e escolha a opção com menor probabilidade de erro.

1.5 Proposta de Solução

A solução proposta incide sobre uma aplicação Python, utilizando os seus recursos, para receber como entrada, boletins digitalizados, em formato Portable Document Format (PDF). O que se propõe é um sistema de machine learning com supervisão, que automatize o processo de transcrição de Boletins de Saúde Infantil e Juvenil manuscritos para informação digital, utilizando processamento de imagem, com recurso à segmentação dos elementos de imagem, interpretando o significado do que está escrito, tendo em conta as imagens passadas, utilizando vários algoritmos diferentes. No final do processo, tem-se em conta as diversas previsões em que os diferentes algoritmos resultaram e tendo em conta um sistema decisor, finalizar com a resposta mais provável.

Em última instância aplicar este sistema, em larga escala, de forma a poder obter a informação disponível, para que possa ser estudada na área da pediatria.

1.6 Estrutura do Documento

Este documento está dividido em 5 capítulos distintos e um capítulo de anexos.

O Capítulo 1 apresenta o contexto, motivação, objetivos e a proposta de solução desta dissertação. É neste capítulo que é realizada uma abordagem introdutória sobre o tema da dissertação

assim como apresentados os objetivos e a solução. O Capítulo 2 apresenta a revisão bibliográfica realizada no contexto desta dissertação. O Capítulo 3 contextualiza algumas das ferramentas cruciais na arquitectura da solução. O Capítulo 4 apresenta todos os desenvolvimentos feitos no âmbito do *Ensemble Learning* realizados detalhadamente. Por último, o Capítulo 5 apresenta as conclusões retiradas com a realização desta dissertação e tópicos de trabalho futuro.

Capítulo 2

Estado da Arte

Este capítulo tem como principal objetivo apresentar o levantamento do estado da arte realizado no âmbito desta dissertação. Como tal, além de se abordarem alguns conceitos relevantes ao tema também se apresentam algumas soluções previamente desenvolvidas que poderão facilitar a caracterização do problema e da sua solução. Os conceitos a ser introduzidos são: processamento digital de imagem abordado na Secção 2.1.2, reconhecimento de caracteres na Secção 2.1.3, data mining e text mining nas Secções 2.1.4 e 2.1.5 respectivamente, Por último, o conceito de ensemble é abordado na Secção 2.1.6.

2.1 Caracterização do Problema

2.1.1 Ambiente de trabalho

O ambiente em que será construída a solução para o problema apresentado é uma aplicação Python 3 com uma pipeline bem definida. Este ambiente oferece uma larga quantidade de ferramentas e recursos para cada um dos desafios que vai apresentando.

2.1.2 Processamento de Imagem

Um dos principais desafios a combater nesta dissertação é o processamento de imagem digital. Uma imagem digital é uma representação em duas dimensões de uma imagem, constituídos por pixeis. Pode-se olhar para uma imagem como um capa de coordenadas cartesiano, em que cada pixel corresponde a uma coordenada de um função f(x,y) e o seu valor, numa escala de 0 a 1 pode ser chamado de intensidade ou nível de cinzento. [4]

O processamento de imagem pode ser considerado como uma série de técnicas de tratamento dos pixeis da imagem de forma a retirar informação desta. Este processamento pode ser divido em três níveis principais, tal como identificado em [16]:

- **Baixo nível** - Procedimentos básicos ao nível dos pixeis. Envolvem o processamento de imagem propriamente dito. Começa com uma imagem e produz uma versão modificada ou

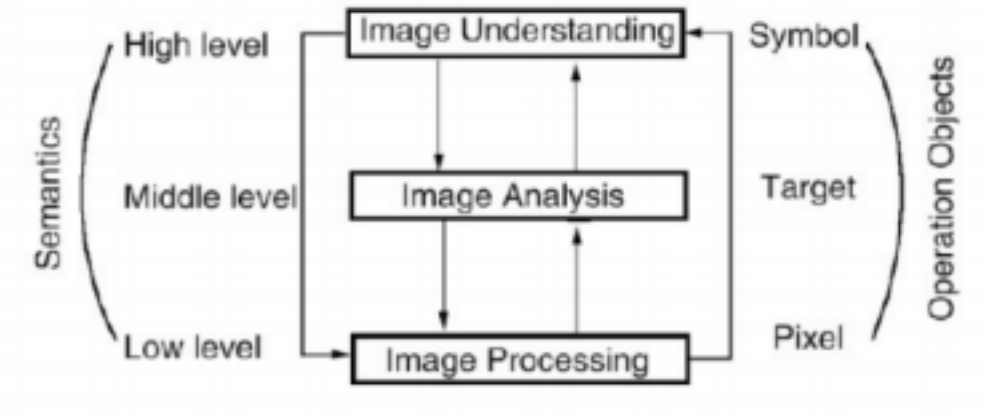

Figura 2.1: Níveis de processamento. De [16].

a transformação entre imagens, melhorando os efeitos visuais. A unidade básica de operação é o pixel.

- **Médio nível** - Foca-se na análise de imagem, em medições. Principal Component Analysis (PCA) produz um novo conjunto de imagens a partir de um dado conjunto. Além disso, este andar é onde ocorre a segmentação. A unidade básica neste nível é o target (alvo).
- **Alto nível** - Interpretação da imagem em que as operações se focam no estudo de cada target e as relações entre eles, assim como o seu possível significado, e em última análise, o significado da imagem original.

A Figura 2.1, tem como propósito esclarecer as relações entre os níveis - como é expectável, não nos é possível passar de baixo nível para alto nível directamente, nem vice-versa.

Os restantes conceitos que são de relevância para um estudo focado no processamento de imagem são: adjacência, conectividade, distância e vizinhança [5].

- **Adjacência** - dizem-se adjacentes dois pixeis que partilham uma borda ou um vértice, tendo em conta que um pixel é um quadrado. A Figura 2.2 mostra-nos 3 pixeis: dois cinzentos e um preto. Os pixeis cinzentos são adjacentes ao pixel preto, mas não são adjacentes um ao outro. Por exemplo, a figura 2.3 mostra um caso de dois pixéis adjacentes.

Figura 2.2: Pixeis adjacentes por borda

- **Conectividade** - Para dois pixeis se dizerem conectados, eles devem ser adjacentes e além disso obedecer a um critério de proximidade escalar a nível de cor.

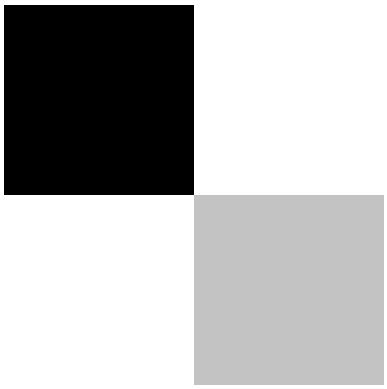

Figura 2.3: 2 Pixeis adjacentes por vértice

- **Distância** - Distância em linha reta entre dois pixeis, tendo em conta a aritmética conhecida dentro de referenciais cartesianos. Valor muitas vezes necessário para a aplicação de algoritmos no processamento de imagem.

- **Vizinhança** - A vizinhança de um pixel é o conjunto de pixeis que lhe são adjacentes.

2.1.2.1 Processamento de Baixo Nível

Como citado em [16], o processamento de baixo nível pode ser dividido em 3 estágios, cada um dividido em várias categorias. Os estágios são: reconstrução, transformação e classificação.

2.1.2.1.1 Reconstrução

- **Restauro** - Minimização de gradientes, de forma a sobressair as fronteiras entre vizinhanças de pixeis. Este processe serve para melhorar a condição da imagem para o restante processamento. Pode-se fazer um restauro geométrico ou radiométrico.

- **Reconstrução** - São feitas projecções de imagens em 2D ou superior, a partir de várias projeções unidimensionais.

- **Mosaico** - Várias imagens em trechos são combinadas de forma a perfazer uma única imagem. Isto serve para obter uma visão de toda a área da imagem.

2.1.2.1.2 Transformação

- **Alargamento do contraste** - Usado para tornar mais evidentes pequenas diferenças de cor entre vizinhanças de pixeis. Essencial quando obtemos uma imagem muito homogénea.

- **Filtragem de ruído** - Remoção de informação desnecessária da imagem. São usados filtros como passa-alto, passa-baixo, passa-banda ou remove-banda para escolher determinados intervalos de informação.

- **Modificação em histograma** - Utilizada para enaltecer uma imagem, fazendo sobressair características da imagem. Isto possível reduzindo a quantidade de níveis diferentes de cinzento existentes numa imagem, forçando valor intermédios a elevar-se ao nível superior ou ao nível inferior.

- **Compressão de dados** - Usada para conservar espaço, minimizando o tamanho da imagem sem se perderem as propriedades radiométricas.

- **Rotação** - Usada para fazer correspondência de imagens, de forma a fazer com que as relações entre pixeis sejam as mesmas, em séries de imagens.

2.1.2.1.3 Classificação

- **Segmentação** - Divide a imagem nos seus objetos, dependendo do problema.

- **Classificação** - Os pixeis são classificados em relação ao seu nível de cinzento. De forma a treinar um classificador, são necessárias várias imagens do mesmo objecto.

2.1.2.2 Processamento de Médio Nível

Tendo em conta o contexto desta dissertação, é relevante referir que existem dois tipos de imagens principais a ser interpretados: gráficos com texto impresso e texto manuscrito. Para esta parte parte do processo, é importante distinguir com clareza os diferentes objetos contidos dentro de uma imagem ao mesmo tempo que identificamos que tipo de objeto se trata. Deste modo, conseguimos facilitar a sua classificação no passo seguinte.

É relevante referir que existe uma vasta variedade de técnicas de segmentação, sendo que apenas se escolheram as técnicas de maior interesse para o contexto desta dissertação.

2.1.2.2.1 Segmentação de gráficos

- **Métodos baseados em região** - Estes são métodos em que o processo de divisão da imagem se baseia na criação de subregiões dentro da imagem. O objetivo é que as regiões criadas tenham algumas semelhanças, definidas por uma série de regras a ser determinadas em função do problema em questão. Podemos ter em conta, por exemplo, o valor do nível de cinzento de uma série de pixeis em que este seja o mesmo e criar uma região, baseada nesse valor.

- **Métodos baseados em fronteiras** - Por oposição aos métodos anteriores, podemos em vez de criar regiões baseadas em semelhanças, é possível dividir a imagem baseada nas diferenças acentuadas, i.e., observar uma queda ou uma elevação de um valor na adjacência de um pixel.

2.1.2.2.2 Segmentação de texto

Como identificado em [10], é possível atingir a segmentação de texto em três níveis sequenciais. À medida que se avança nos níveis, obtemos cada vez mais detalhe, apesar de não ser necessário utilizar todos os três níveis.

- **Segmentação de linha** - É o primeiro passo para a segmentação de texto. Faz-se uma análise linha a linha e coluna a coluna, da esquerda para a direita e de cima a baixo, verificando-se a intensidade de cada pixel. Dependendo da intensidade, agrupam-se os pixeis em regiões, indicando onde está o conteúdo relevante. Devido a imprecisões no scanning e na escrita, pode acontecer que o texto esteja torto, o que pode implicar complicações no sucesso dos algoritmos. Por esse motivo, os erros devem ser detetados e corrigidos. Quando se deteta um caso destes, a palavra ou caractere deve ser rodada e alargada de forma até que o ângulo e formato estejam de acordo com o padrão.

- **Segmentação de palavra** - Define com clareza a distinção entre palavras distintas, através da identificação dos espaços entre elas. Da mesma forma que o passo anterior, esta análise é feita através de scanning vertical e horizontal, da esquerda para a direita e de cima para baixo.

- **Segmentação de caractere** - Esta fase tem como propósito dividir, com clareza, a palavra em caracteres individuais.

2.1.2.3 Processamento de Alto Nível

O último nível, o alto nível, é o nível em que é possível interpretar a imagem e extrair desta, a informação contida. No contexto da dissertação, reconhecimento de caracteres, pode-se considerar que existem quatro técnicas de reconhecimento de padrões: correspondência de templates, técnicas estatísticas, técnicas estruturais e redes neuronais.

No caso da dissertação, o objetivo é reconhecer caracteres, por isso será aplicada uma forte componente de treino de redes, que será explicada na Secção 2.1.3.

2.1.2.4 Ferramentas consideradas

Visto que a solução será desenvolvida em Python 3, a ferramenta a utilizar será a biblioteca OpenCV para Python [1], que oferece um conjunto de funções com uma elevada utilidade neste contexto.

2.1.3 Reconhecimento de Caracteres

O reconhecimento ótico de caracteres é um tema que tem vindo a ganhar notoriedade por diversas razões. À medida que o recurso a meios digitais tem vindo a crescer, a necessidade de

passar caracteres visíveis e reconhecíveis no mundo real para o mundo digital tem vindo a crescer. Para esse efeito, existe a tecnologia de reconhecimento ótico de caracteres, ou OCR.

A tecnologia mencionada utiliza imagens de texto, como entrada, e devolve o carácter codificado como American Standard Code for Information Interchange (ASCII), de forma a ser lido computacionalmente. Segundo [2], existe uma linha definida que todos os sistemas de OCR seguem, tipicamente. Esta linha é definida pela Figura 2.4.

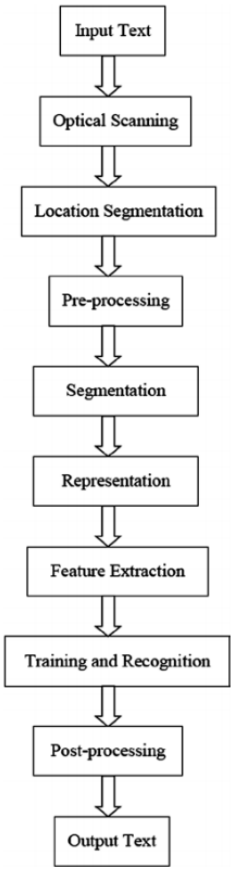

Figura 2.4: Componentes de um sistema OCR. De [2].

Nesta linha, podemos observar que é introduzido o texto de entrada (Input Text), que é seguido de um Optical Scanning, que analisa a imagem e converte a intensidade da luz em nível de cinzento. De seguida, tem se a segmentação de localizações (Location Segmentation), que, por sua vez, tem como objetivo determinar os constituintes da imagem. Após este processo, tem-se o pré-processamento, que é um conjunto de processos que tornam a imagem usável para os passos seguintes. De seguida, executa-se a segmentação, que divide a imagem nos seus constituintes. A fase seguinte é das mais relevantes e essenciais do processo, a representação. Esta utiliza imagens em níveis de cinzento ou em binário são fornecidas para um reconhecedor. Existem 3 grupos de métodos que se utilizam nesta fase: transformação global e expansão de série, representação estatística e representação geométrica e topológica. A fase seguinte é a extração de características,

que tal como o nome indica o objetivo é extrair características dos símbolos. O passo seguinte é o treino e reconhecimento, que se pode considerar uma das fases mais cruciais que esta tese planeia desenvolver, sendo que é nesta fase que se utilizam as redes neuronais. Nesta fase é feita a classificação de um *sample* desconhecido para uma classe predefinida. O uso de redes neuronais deve-se ao facto de serem capazes de se adaptar a mudanças nos dados e de conseguirem aprender novas características que possam aparecer. A fase final é o pós-processamento, que consiste, principalmente, no agrupamento e na deteção e correção de erros.

2.1.4 Data Mining

O conceito de data mining é analíse de dados, em larga escala, de forma a identificar padrões, anomalias e correlações, de forma a conseguir fazer previsões, dadas determinadas entradas.

De acordo com [7], existem uma linha de processos a seguir, quando se trabalha com mineração de dados:

- Data cleaning - remoção de ruído;
- Data integration - agregação de dados de fontes diversas;
- Data selection - Seleção de dados relevantes para a análise a ser feita;
- Data transformation - Operações a ser feitas nos dados para ser apropriado para mineração;
- Data mining - Mineração de dados propriamente dita, em que processos inteligentes são aplicados de forma a extrair padrões dos dados;
- Pattern evaluation - Avaliação e distinção entre padrões encontrados anteriormente, de forma a identificar os verdadeiramente importantes para a análise;
- Knowledge presentation - Representação das descobertas encontradas ao utilizador.

  Os primeiros quatro passos são passos de preparação de dados para serem minerados. É de referir também que as descobertas podem ser apresentadas a um utilizador ou registadas numa base de dados.

Os objetivos do data mining dividem-se em duas categorias [5]: Previsão e Descrição. Dentro da previsão, temos dois objetivos concretos:

- Classificação
- Regressão

No lado da Descrição, os objetivos são:

- Associação
- Segmentação

- Sumarização
- Visualização

Objetivos Gerais	**Objetivos Concretos**
Previsão	Classificação e Regressão
Descrição	Associação, Segmentação, Sumarização e Visualização

Tabela 2.1: Objetivos do Data Mining.

No contexto da dissertação, o objetivo principal irá incidir sobre a previsão e mais concretamente na classificação, pois é aqui que estão englobados os algoritmos de reconhecimento de caracteres manuscritos.

2.1.4.1 Classificação

A classificação, sendo um dos tópicos mais importantes, senão o mais importante da dissertação, consiste em definir uma estrutura que possa ser aplicada a objetos não classificados e consequentemente categorizá-los em classes. Existem diversos métodos de categorização e alguns deles podem ser verificados nos tópicos que se seguem. [4]

- Artificial Neural Network (ANN), ou redes neuronais artificias, como se falou previamente.
- Classificador de Bayes.
- Máquina de vetores de suporte.
- Modelos Ocultos de Markov.
- K-ésimo vizinho mais próximo.
- Deep learning.

2.1.4.2 Metodologias de Data Mining

O processo de Data Mining (DM) é complexo e por isso é necessário recorrer a metodologias já existentes e comprovadas de forma a tirar o máximo proveito desta tecnologia. Existem dois métodos que se destacam para esta categoria: CRoss Industry Standard Process for Data Mining (CRIP-DM) e Sample, Explore, Modify, Model, and Access (SEMMA).

As principais diferenças entre estes dois métodos são a definição de Estudo de Negócio e Estudo de dados e Implementação que são incorporados pelo CRISP-DM e não pelo SEMMA. Por esse motivo, considerou-se o CRISP-DM como sendo o método mais vantajoso.

2.1.4.2.1 CRISP-DM

O CRISP-DM é um modelo de data mining open-source, desenvolvido pelos líderes na área. É um modelo desenvolvido com o intuito de melhorar as práticas e providenciar uma estrutura mais eficiente e com melhores resultados de data mining. Isto é observável na Figura 2.5.

A metodologia inclui os seguintes passos:

- **Estudo do negócio**: Estudo realizado com o intuito de entender de uma perspectiva financeira o objetivo do projeto.
- **Estudo dos dados**: Recolha de dados e interpretação dos mesmos.
- **Preparação dos dados**: Limpeza, formatação e integração dos dados de forma a estarem prontos a ser trabalhados.
- **Modelação**: Otimização do sistema, através de técnicas de calibragem e modelação.
- **Avaliação**: Avaliação dos resultados, tendo em conta determinados parâmetros, decisão dos passos seguintes.
- **Implementação**: Utilização do conhecimento que se adquiriu pela utilização do modelo.

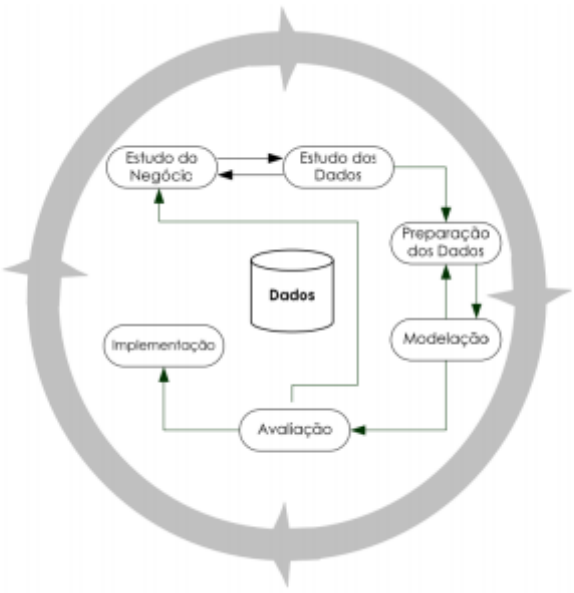

Figura 2.5: Metodologia CRISP-DM. De [11].

2.1.5 Text Mining

Mineração de texto, apesar de ser uma variante da Secção 2.1.4, lida com dados de texto não estruturados. Isto implica que estes dados não são organizados nem predefinidos, o que dificulta

o processo de identificação de padrões. Este processo permite-nos automatizar a extração de informação a partir de texto, e é, na generalidade da dissertação, o que é pretendido fazer.

A mineração de texto, tal como em 2.1.4, tem melhor performance com a maior quantidade de dados que lhe sejam dados, por isso faz sentido que o primeiro passo a ser tido em conta é a agregação de dados. De seguida estes textos são processados e destes são extraídos os padrões, através de uma análise textual. Estes padrões são comparados aos existentes e é feita um ajuste na informação já conhecida de forma a manter o sistema inteligente e atual. Numa fase final é reconhecida.

Existem diversas técnicas de text mining [12]. Um dos passos mais importantes desta dissertação é a segmentação de palavras de forma a identificar unicamente cada uma destas, tal como mencionado em [4].

Uma das formas de ajudar na segmentação de palavras é a remoção de caracteres que possam confundir o sistema, isto pode incluir elementos como vírgulas, pontos, parênteses, etc. Estes itens removidos são chamados *tokens*.

A remoção de palavras que não interferem no significado da frase também utilizada de forma a reduzir o ruído na interpretação da informação. Preposições, artigos ou conjunções são exemplos de palavras que podem ser removidas pois não interferem no sentido geral da frase.

É chamado *stemming* ao processo de reduzir uma família de palavras à palavra base com o mesmo significado, de forma a reduzir a variedade de formas em que um elemento pode aparecer na interpretação de dados.

O reconhecimento de nomes é um processo que permite identificar nomes que se refiram a pessoas, empresas, países, etc. e que permitem ao sistema ignorar esta palavra na interpretação.

2.1.6 Ensemble

A utilização de *ensemble learning* é algo que se provou útil em sistemas de classificação de imagens [15] [6], que é o caso desta dissertação.

O conceito de ensembles consiste na utilização de diferentes algoritmos para a obtenção de informação de forma a poder comparar resultados utilizando os diferentes métodos. A decisão final pode ser tomada, tendo em conta diferentes métodos: por maioria, tendo em conta pesos de votos de cada um dos métodos, tendo em conta a imagem considerar um dos algoritmos como o mais provável de acertar, etc.

Nesta dissertação o tipo de ensembles que se pretende implementar é *stacked generalization* [14]. Este conceito consiste na utilização das previsões feitas anteriormente como entradas para o treino das próximas redes neuronais. Desta forma, a cada camada de previsão, a precisão relativa à previsão aumenta.

2.1.6.1 Ferramentas consideradas

A ferramenta considerada para a construção de algoritmos preditivos foi a biblioteca Keras[3] para Python, pois é a biblioteca mais completa e com mais recursos no que toca ao *text mining*. A

ferramenta a ser utilizada para a gestão de dados foi o Pickle[13] para Python, pois essa ferramenta oferece as utilidades necessárias a um baixo custo de armazenamento.

2.2 Conclusões

Tendo em conta o que foi referido neste Capítulo, o pretendido com a dissertação é utilizar as ferramentas de pré-processamento e processamento de imagem, de modo a manipular a imagem a otimizar a entrada nas redes neuronais a ser desenvolvidas. Também se utilizam os conceitos compreendidos de *data mining*, *text mining* e *ensemble learning*, que em primeira instância, ajudam na criação e configuração de algoritmos preditivos, e numa fase posterior oferece recursos de modo a criar um algoritmo meta-decisor que escolha qual a previsão com maior probabilidade de acerto.

De modo geral, o Capítulo serviu como base teórica aos conceitos desenvolvidos nos Capítulos 3 e 4 e além disso apresenta algumas das ferramentas que foram utilizadas nesses mesmo capítulos

Capítulo 3

Metodologia e Implementação

O principal propósito deste capítulo é explicar, em grande detalhe, cada camada da arquitectura da solução desenvolvida. É de realçar que este processo tem como entrada o conjunto de livros BSIJ em formato PDF e como saída a informação médica contida nos livros de forma agrupada.

A Secção 3.1 apresenta as preparações tomadas na antemão do desenvolvimento da aplicação, enquanto a Secção 3.2 apresenta a estrutura e arquitectura da *pipeline* desenvolvida para a aplicação. A Secção 3.3 apresenta os métodos de pré-processamento tomados na aplicação e a Secção 3.4 apresenta o processamento de todas as imagens, incluindo tabelas e gráficos. A última Secção 3.5 apresenta o modo como se tratou os dados depois de cada previsão.

3.1 Preparação

Como forma de preparação para iniciar a pipeline são feitas diversas preparações, nomeadamente: a criação de um diretório dos livros correspondentes às crianças do género feminino, criação de um diretório dos livros correspondentes às crianças do género masculino e fora dos directórios são depositados os livros de forma a poderem ser acedidos pela aplicação e para que esta consiga depositar os resultados nos respetivos diretórios. Um esquema representante da estrutura dos diretórios pode ser encontrada na Figura 3.1. Para todas as implementações foram utilizados 10 exemplos de boletins de saúde.

3.2 Pipeline

A aplicação construída em Python 3, assenta numa pipeline que assume como entrada um BSIJ, digitalizado em formato PDF, e que tem como saída os valores propriamente identificados para cada uma das idades marcadas no documento. Para isto acontecer, o documento PDF é submetido a uma pipeline que leva a cabo, passo-a-passo cada processo necessário para chegar ao resultado final. Em primeira instância, é necessário mudar o formato do documento e dividi-lo em páginas, de forma a transformar processos que outrora seriam morosos e complexos, em processos mais simples e rápidos. De seguida, é iniciado o pré-processamento, que tem como propósito

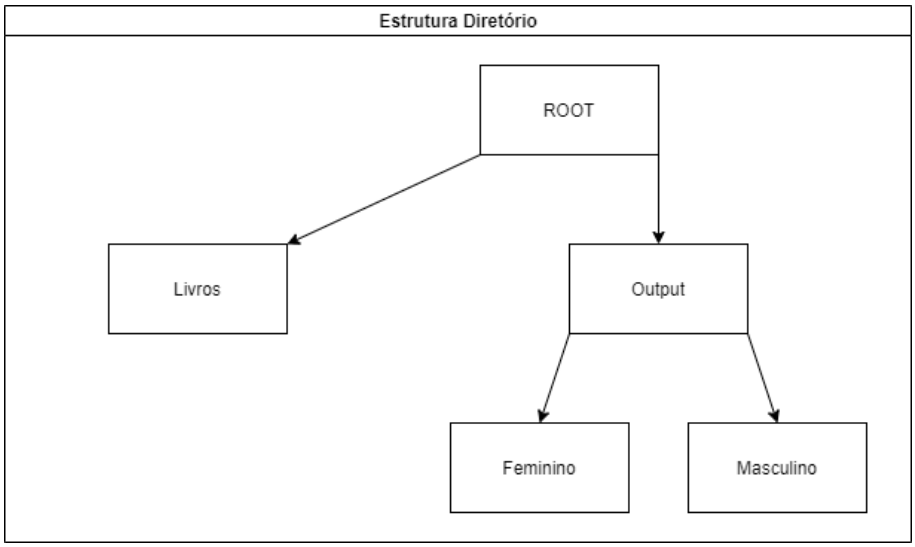

Figura 3.1: Estrutura dos diretórios

principal, editar a imagem de forma a que o processamento seja o mais simples possível, isto é, afinamentos e binarização. Tendo este passo concluído, dá-se início ao processamento da imagem.

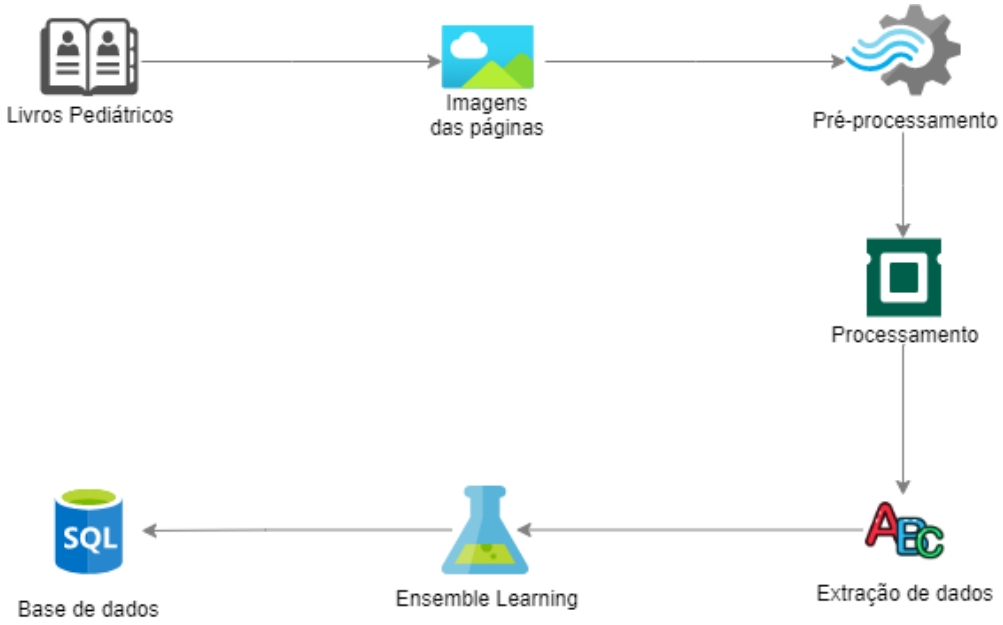

Figura 3.2: Pipeline simplificada.

Nesta fase, o objetivo principal é identificar unicamente cada caractere, envolvendo portanto todas os processos desde a binarização até lá chegar, assim como proceder à sua previsão. Tendo chegado a este patamar, segue-se o tratamento de dados, que pode ser feito de dois modos: um modo manual, em que é o utilizador que avalia a previsão, de forma a dar maior controlo sobre todo o processo e outro modo automático em que se compara os valores obtidos nos gráficos com os valores obtidos do OCR, sendo então necessário que haja duplo acerto, em termos preditivos.

Ambos os modos, terminam com a adição do caractere escrito a ser guardado no *dataset*, em caso de acerto ou a ser descartado em caso de erro. Isto leva a uma melhoria à medida que se prevêem mais e mais boletins, pois a cada boletim adicionado se incrementa o *dataset*. Além disso, existe a possibilidade de os valores adicionados melhorarem ainda mais a previsão de dados, pois os valores guardados são especificamente retirados do próprio caso de estudo, sendo valores especializados no que toca à previsão de caracteres.

```
tabela = identificarTabela(pagina)

celulas[] = identificarCelulas(tabela)

for celula in celulas:

    chars[] = identificarChars(celula)

    i=0
    for char in chars:

        prevs[i] = previsaoChar(char)
        i++

    numero = fazerNumero(prevs)

    graf = associarGrafico(celula.coluna)

    acerto = verificarPonto(numero, graf)

    if acerto{

        guardarDados("Diretório de Acertos")

    }else{

        guardarDados("Diretório de Erros")

    }
```

Figura 3.3: Pseudocódigo da pipeline para o caso de uma tabela com o processo automatizado de seleção de dados.

A Figura 3.3, apresenta o pseudocódigo relativo ao caso de ser de forma totalmente automatizada. De forma simplificada, temos na primeira linha a obtenção da tabela como um todo. Na segunda linha obtemos as um vetor com todas as células da tabela. De seguida iniciamos o processo para cada uma das células de identificar dentro da célula quantos caracteres há e em que área se encontram. Do mesmo modo para cada um dos caracteres inicia-se o processo de prever cada um deles, através da função *previsaoChar()*.Dessa previsão sai um algarismo de 0 a 9 e no final das previsões, é feito um numero composto por todos os algarismos das células, através da função *fazerNumero()*. A essa célula é associado um gráfico, através da coluna da célula e no final é feita uma verificação através da comparação de dados. Em caso de acerto, é guardada num diretório, que será utilizado para treinar a rede seguinte, em caso de erro é guardado num diretório, que poderá ser utilizado futuramente para a deteção de erros e eventuais melhorias de performance.

3.3 Pré-Processamento

Ao nível do pré-processamento, é utilizado um *wrapper* de *pdftoppm* e *pdftocairo*, chamado *pdf2image*, que requer também a instalação do *poppler for Windows*, além deste é também utilizado a biblioteca *OpenCV*, que é uma biblioteca com o propósito de resolução de problemas de visão por computador.

O *pdf2image* é utilizado de forma inicial apenas para converter o ficheiro PDF em diversas imagens, isto é, converte cada uma das páginas do documento PDF numa imagem com o nome definido pelo código.

O *OpenCV* tem ferramentas que permitem identificar linhas e pontos, que por sua vez permitem resolver problemas como orientação das páginas e o seu respectivo recorte.

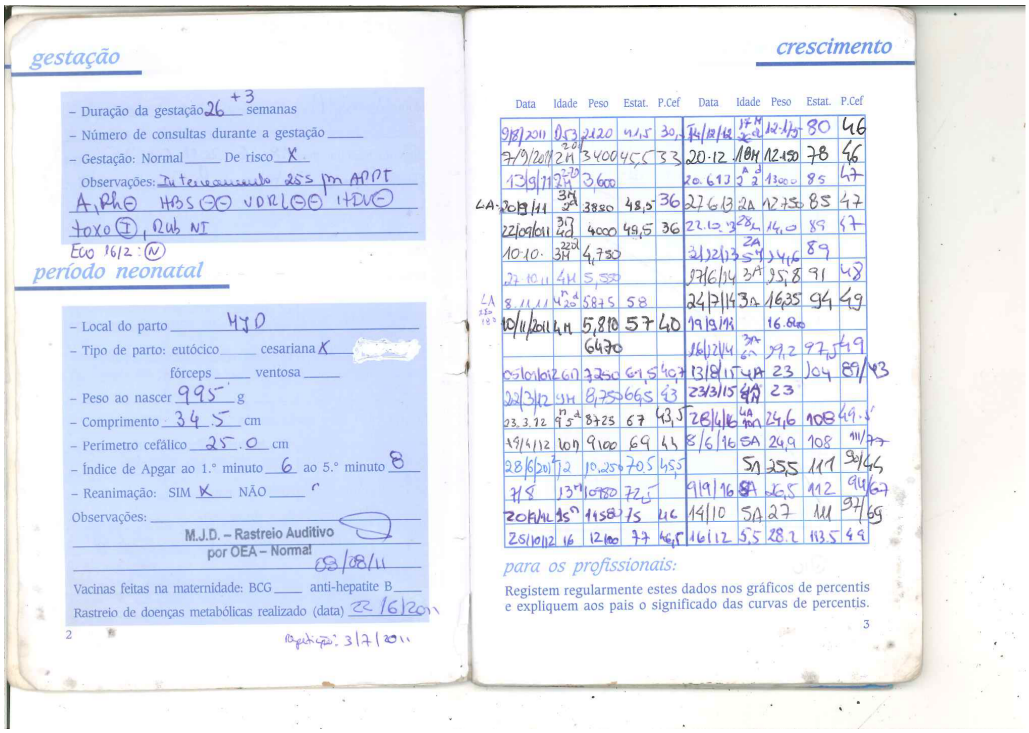

Figura 3.4: Exemplo de uma imagem após *pdf2image*.

Visto que cada imagem contémpla duas páginas de BSIJ, como é possível observar na Figura 3.4, é necessário identificar o ponto de largura em que é necessário fazer um corte transversal. Além disto, é necessário verificar que imagem está no modo paisagem. Esta verificação é feita facilmente, fazendo uma comparação de largura com comprimento da imagem. Caso não esteja, nos moldes necessários é feita uma rotação de modo a cumprir o requisito.

De forma a efectuar o corte transversal, é necessário aplicar um filtro de cinzentos, utilizando a biblioteca previamente mencionada, e também é aplicado um filtro de Gauss, para aumentar a névoa da imagem, isto prova-se útil pois reduz as imperfeições e eventuais erros de digitalização. De forma a aumentar a detecção de bordas com o algoritmo *Canny*, a imagem é reduzida, o que aumenta a sua eficácia. Deste modo, a imagem resultante pode ser observada na Figura 3.5

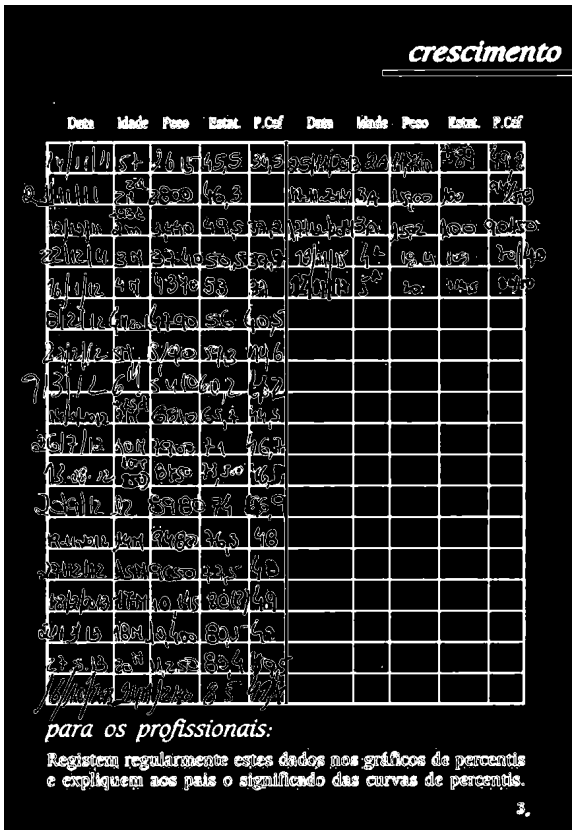

Figura 3.5: Exemplo de uma tabela após *Canny*.

O último passo antes do processamento, assenta na correção da verticalidade da página. Em primeira instância é necessário percorrer a imagem, do exterior para o interior, de forma a detetar oscilações abruptas na imagem, que correspondem às bordas da página. Dentro dessas bordas verifica-se se está contido 60% do total da imagem no interior das bordas. Em caso positivo, divide-se a aresta maior por 2 e ocorre um corte transversal nesse local. Em caso negativo, significa que existe apenas uma página na imagem e é feito apenas um recorte à volta da única página do documento correspondente.

De seguida, é utilizada a função *HoughLines*, que aplica a transformada de Hough à imagem, que é uma técnica matemática que identifica formas geométricas, para identificar linhas retas. Identificando as linhas retas verticais das tabelas ou o eixo das ordenadas no caso dos gráficos. Tendo estas retas é possível identificar qualquer pixel incorporado por estas, e sendo assim é possível calcular o declive. Tendo o declive é possível calcular o desvio na orientação das páginas. Sendo assim, é necessário aplicar uma rotação à página, com origem no centro da página.

Sendo assim, no final da camada do pré-processamento, o produto obtido são as páginas recortadas prontas a entrar individualmente na camada do processamento.

3.4 Processamento

Como mencionado previamente, o foco principal do processamento passa por isolar o conteúdo que se pretende prever e proceder à sua previsão.

Tendo em conta que nesta fase, o que é recebido como entrada é uma imagem propriamente orientada e que contém apenas uma imagem de conteúdo que pode tanto ser uma tabela como um gráfico, tanto pode ser uma imagem cor-de-rosa, correspondente a uma criança do género feminino, como uma imagem azul, correspondente a uma criança do género masculino.

Assim sendo o primeiro passo do processo, passa por determinar se a imagem recebida é correspondente ao género feminino ou masculino. Tendo em conta que as imagens tão agrupadas por boletins, é possível fazer uma verificação para a primeira imagem e podemos concluir que todas as imagens deste boletim são do mesmo género. De forma a detetar o género é feita uma conversão da imagem para o modo Hue, Saturation, Value (HSV). Deste modo é possível estabelecer um intervalo de cores no HSV e definir esse intervalo para o cor-de-rosa e fazer o mesmo para o azul. Este intervalo é de 0 a 360 e cada 60 pontos são correspondentes às seguintes cores: vermelho, amarelo, verde, ciano, azul e magenta, como é possível observar na Figura 3.6.

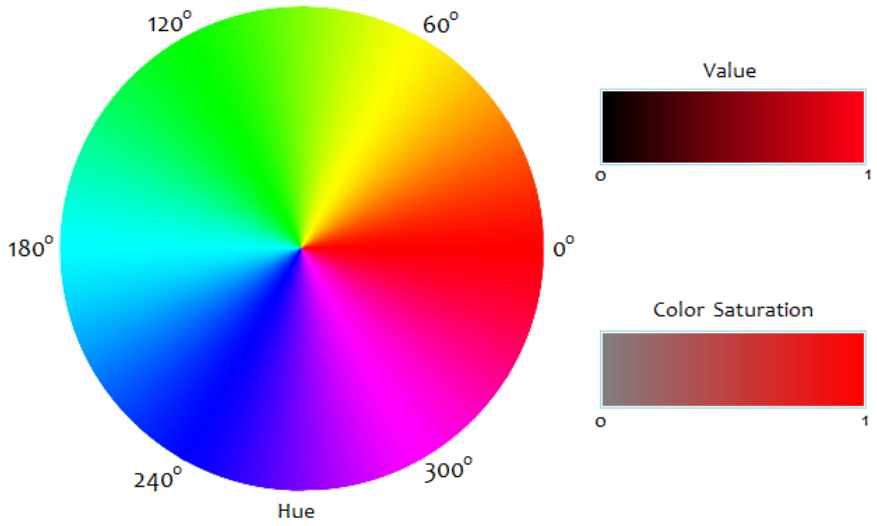

Figura 3.6: Representação das cores em HSV .

Sendo assim e tendo em conta que apenas existem duas opções, são feitos dois intervalos com 180 pontos de *value* cada um. É aplicada uma máscara relativa a cada intervalo e de seguida é feita uma contagem de pixeis não nulos. É feita uma comparação e a imagem que tiver o maior número de pixeis não nulos é a imagem correspondente a esse género.

O passo seguinte corresponde ao isolamento do conteúdo, que é feito numa primeira instância de forma similar nos gráficos e nas tabelas. Este passo tem como propósito remover toda a informação não essencial para se dar início à previsão de caracteres

De forma a isolar o conteúdo de cada uma das páginas, e da mesma forma que já ocorreu anteriormente, é feita uma redução do tamanho da imagem, de forma a reduzir as imperfeições.

É removido o fundo branco através da aplicação de uma máscara, que é necessário para realçar e contrastar o conteúdo. Da mesma forma que ocorreu anteriormente, é aplicado um filtro Gaussiano de forma a aumentar a névoa, que também tem como propósito reduzir as imperfeições. Este processo é necessário pois por vezes, no processo de digitalização, as linhas retas ficam cortadas e aumentando a névoa, permite a estas linhas a voltar a ficar conectadas.

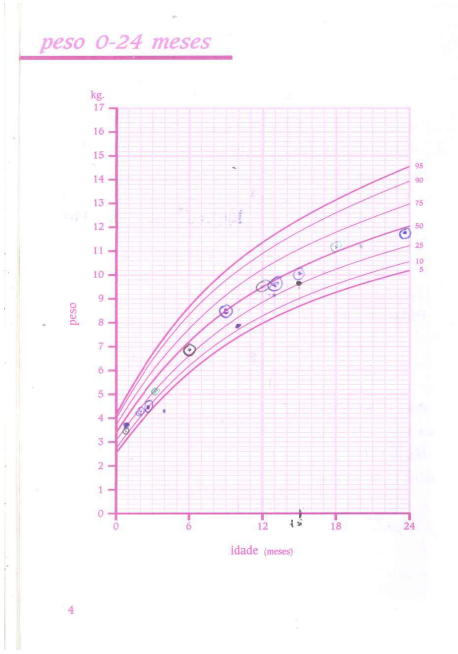

Figura 3.7: Página cortada do gráfico com as cores originais.

Tendo os filtros aplicados, o passo seguinte é a execução da função *Canny*, da mesma forma que aconteceu na divisão por páginas, de forma a encontrar os rebordos das tabelas, assim como os eixos dos gráficos. A essas coordenadas é associado um intervalo de extra de forma a incluir valores de legenda, números associados a gráficos, títulos, etc. Além disto é aplicada uma dilatação da imagem de forma a aumentar o espaçamento em pontos importantes da imagem. Isto representa uma melhoria no comportamento do passo seguinte.

O próximo passo, correspondente à utilização da função *findContours*, também do *OpenCV*, que tem como utilidade, como o nome indica, identificar *contours*. Define-se contours, como "uma curva que une todos os pontos contínuos (ao longo do limite), com a mesma cor ou intensidade."Os contornos são uma ferramenta útil para a análise da forma e detecção e reconhecimento de objectos.

Além disto é imposto à função que a área que encontra com a função seja pelo menos 40% da área total da página. Isto restringe as *contours* que encontra sejam apenas as da tabela, no caso da página ser uma tabela ou do gráfico como um todo caso as imagens sejam dos gráficos. É neste ponto retirado o valor das coordenadas do canto inferior esquerdo assim como o canto superior direito.

A partir deste momento, o processamento dos gráficos e das tabelas é distinto, procedendo-se a uma bifurcação.

3.4.1 Gráficos

Relativamente aos gráficos, a imagem chega processada como visível na Figura 3.8 e o primeiro passo passa por identificar os eixos dos gráficos, sendo que o eixo das ordenadas é correspondente ao valor medido nessa página e o eixo das abcissas corresponde à idade do utente.

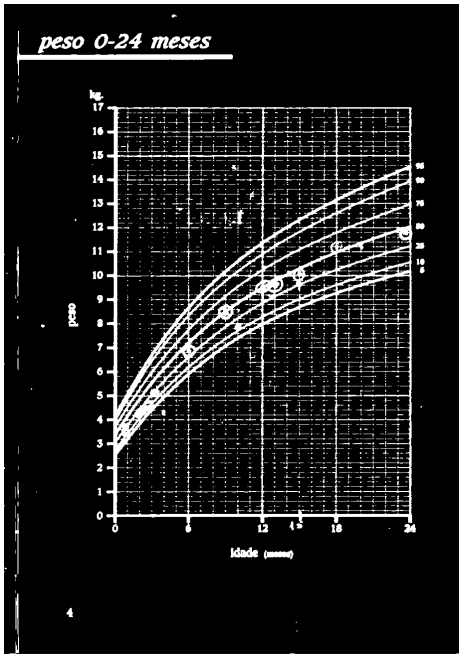

Figura 3.8: Gráfico inicial.

3.4.1.1 Identificação dos limites do gráfico e da escala

Tendo identificado a origem e os eixos, é então necessário identificar a escala do gráfico, isto é os limites do gráfico onde é necessário identificar os valores. Estes valores são os pontos das extremidades dos eixos, tanto das ordenadas como das abcissas. Com este passo concluído, temos

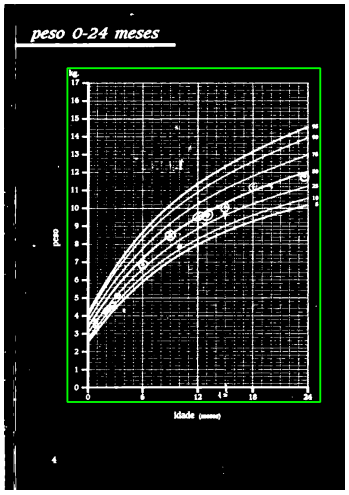

Figura 3.9: Limites do gráfico.

então definidos os limites do gráfico. Os limites podem ser identificados na Figura 3.9

Estando os limites do gráfico bem definidos, é necessário saber a que corresponde cada incremento do gráfico. Para o efeito é utilizado o *pytesseract* com recurso ao *Google Tesseract OCR*, de forma a identificar os caracteres dos gráficos. Sendo que os caracteres estão escritos à máquina, a precisão é muito superior ao que seria caso fosse escrito manualmente, no entanto, são feitas várias leituras dos valores de forma a aumentar a certeza da leitura.

3.4.1.2 Identificação de linhas

É importante realçar que todos os itens correspondentes ao gráfico são da cor correspondente ao género, isto é, os eixos, as linhas divisórias, as linhas correspondentes aos percentis e os valores dos gráficos são da cor correspondente ao género do utente (azul para masculino e rosa para feminino). Isto é visível na Figura 3.4 e na Figura 3.7 .

Da mesma forma que é feito anteriormente, é detetado o género utilizando os intervalos HSV, e de seguida é aplicada uma máscara com a cor respectiva, isolando as linhas.O que se obtém pode ser observável na Figura 3.8 3.8.

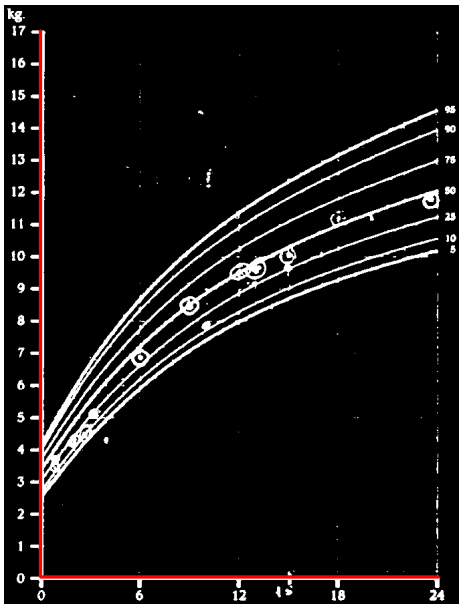

Figura 3.10: Identificação dos eixos correspondente aos gráficos.

Com o intuito de melhorar a clareza da imagem obtida, é aplicado uma dilatação. Aplicado esta transformação, é aplicada a função *HoughLines*, da mesma forma que anteriormente com o propósito de identificar os eixos, contudo todas as linhas de auxílio também são identificadas no processo, verticais e horizontais. De modo a se chegar ao eixo das ordenadas, são feitas restrições que correspondam a um único eixo. Estas restrições são as seguintes: inclinação seja igual a 90º ± 5º e tem de estar, no máximo, a 10% da largura total da página. De forma a identificar o eixo das abcissas, as restrições são: inclinação seja igual a 0º ± 5º e tem de estar, no máximo, a 10%

da altura total da página. Além destas restrições, é feita uma verificação do ângulo que estes dois eixos fazem, tendo necessariamente de perfazer 90º.

Tendo calculado o eixo das abcissas e o eixo das ordenadas, é calculado a sua intersecção,completando o gráfico com a sua origem. Isto é visível em 3.10.

3.4.1.3 Identificação de pontos

De forma a identificar os pontos do gráfico, depois de aplicada a máscara correspondente ao género, é aplicado uma máscara que identifique as cores mais escuras da imagem. Sendo que por norma a tinta da caneta que escreve nos boletins é azul-escura ou preta. A imagem resultante pode ser encontrada na Figura 3.11

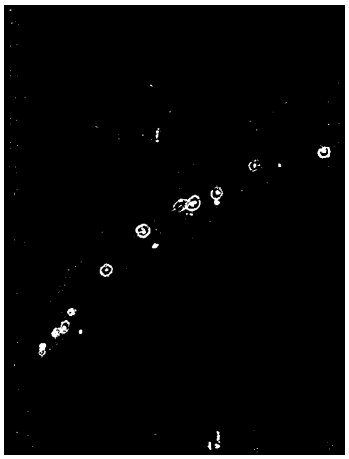

Figura 3.11: Marcas escuras dos gráficos isoladas.

Depois de aplicadas as máscaras é aplicada a função *findContours*, também já usada anteriormente, sendo que a condição é que o tamanho da área dos *contours* seja superior a 10 píxeis e inferior a 100 píxeis. O resultado das detecções pode ser encontrado na Figura 3.12. Como se pode observar com este processo são eliminadas pequenas marcas que possam ser ter passado por defeitos na digitalização.

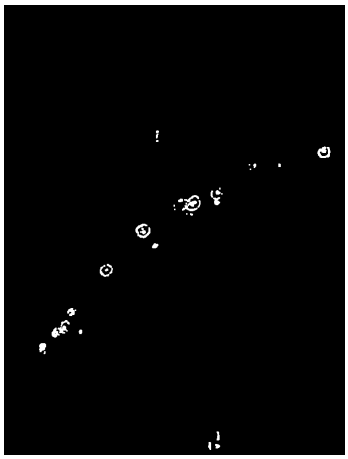

Figura 3.12: Pontos isolados dos gráficos.

3.4.1.4 Cálculo dos valores correspondentes aos pontos

De forma a conseguir identificar os pontos com os eixos, e chegar a valores concretos correspondentes aos pontos, é necessário utilizar as coordenadas dos pontos, que já se relacionam diretamente com os eixos, e relacionando a distância da coordenada das ordenadas à coordenada origem e relacionando a distância da coordenada abcissa também à origem, chegamos a um valor para cada uma das coordenadas que, multiplicado pela escala obtida anteriormente corresponderá ao valor do ponto no gráfico.

Depois de obtidos os valores, é necessário eliminar os outliers. Estes pontos podem surgir por pequenas inscrições acidentais com a caneta, assim como sombras que acontecem durante o processo de digitalização.

Essa eliminação é feita tendo em conta dois fatores: primariamente, é necessário verificar que apenas haja um valor de ordenada por cada abcissa, e em seguida verificar se o valor de ordenada é inferior a pelo menos um dos dois valores seguintes. Por norma, os valores de ordenada não devem decrescer, por isso, a regra geral é que os gráficos são sempre crescentes. A tabela final resulta no que está demonstrado na Figura 3.13.

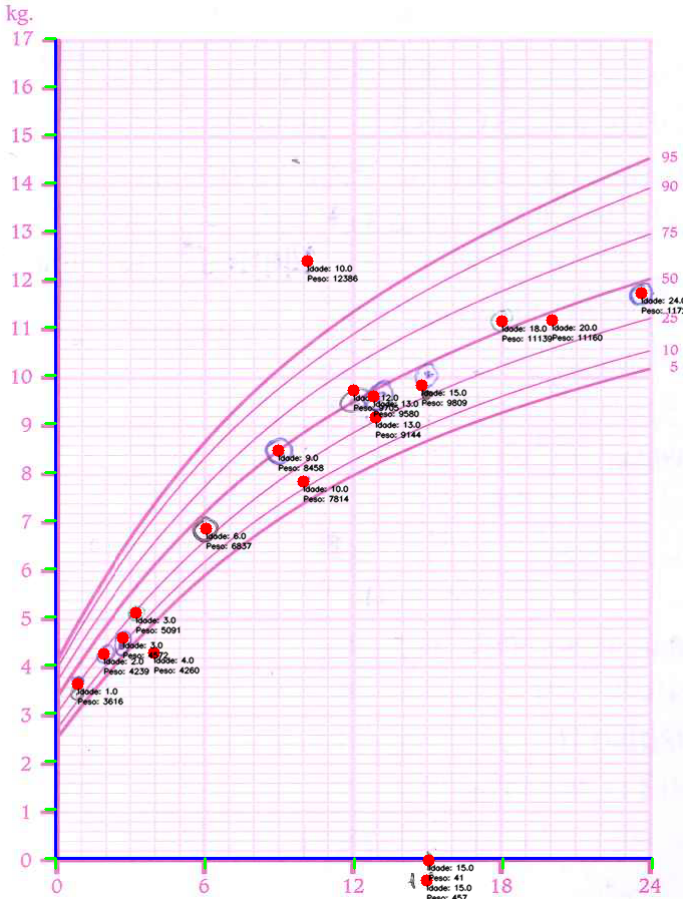

Figura 3.13: Gráfico com os pontos finais devidamente identificados.

3.4.2 Tabelas

Relativamente às tabelas, a imagem chega processada como visível na Figura 3.14 e o primeiro passo passa por identificar a grelha da tabela, isolar cada uma das células da tabela, identificação da posição de cada um dos caracteres da célula, assim como o número de caracteres, o procedimento em casos de sucesso, o procedimento em casos de insucesso, e a previsão base, utilizando o dataset open-source **MNIST** [9].

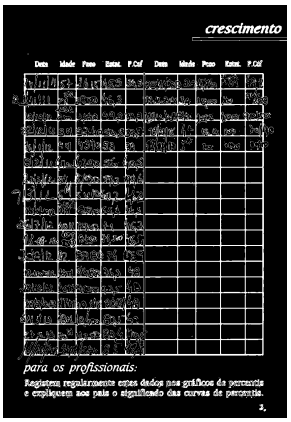

Figura 3.14: Tabela inicial.

3.4.2.1 Deteção da grelha da tabela

Do mesmo modo que demonstrado na Secção 3.4.1, é aplicada uma máscara correspondente ao género do utente indicado no boletim, com o mesmo propósito que anteriormente, isto é, o isolamento das linhas. É aplicada também uma dilatação de forma a realçar elementos, e por último é utilizada a função *HoughLines*, também utilizada anteriormente.

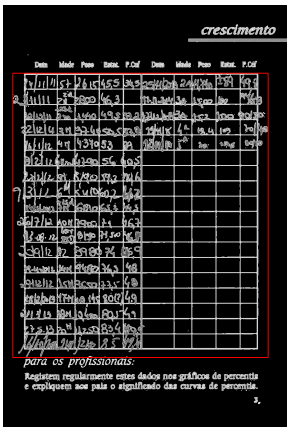

Figura 3.15: Limites da tabela.

O fator decisivo no processamento de tabelas passa por interpretar o nível de preenchimento das tabelas, isto é, existem células em que o seu conteúdo ultrapassa os limites da própria célula e por vezes até fora da própria tabela. Nos casos em que isto acontece, é necessário que o algoritmo contenha os conteúdos da tabela que ultrapassem os limites da tabela. Assim sendo, nos casos em que não é encontrado o limite da tabela devido aos conteúdos que excedem o limite, a tabela é encontrada utilizando as linhas horizontais, sendo que estas tem um espaço igual entre elas desde a primeira à última. Tendo isto em conta, basta que duas linhas sejam encontradas para que seja facilmente identificável onde se localizam as outras, desde que a última e a primeira linha horizontal sejam encontradas. Tendo uma previsão de onde se localizam as outras linhas apenas

é necessário verificar que no fim de contas sejam encontradas as linhas referentes a 18 linhas da tabela, isto é, 19 linhas geométricas. Um caso disto é observável em 3.15.

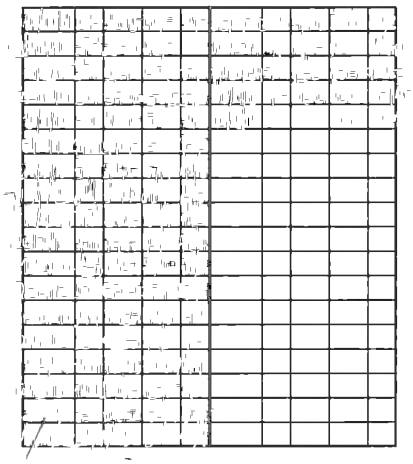

Figura 3.16: Grelha isolada.

Sendo que nas linhas verticais o espaçamento é irregular, é utilizada a linha central que tem uma largura superior às intermédias e além disso localiza-se precisamente no meio da tabela, o que facilita o cálculo sua posição. Tendo este processo em conta, as linhas horizontais serão as que obtenham melhor correlação com a posição prevista da tabela, assim como as que obtenham a melhor correlação de espaçamentos entre as duas extremidades da tabela. Isto pode ser confirmado no exemplo dado na Figura 3.16.

3.4.2.2 Isolamento de células

Cruzando as linhas horizontais e verticais da tabela obtidas na Secção 3.4.2.1, obtém-se a grelha. Se o mesmo processo for feito com duas linhas horizontais consecutivas com duas linhas verticais consecutivas, obtêm-se quatro pontos que correspondem aos quatros vértices da célula. Cada célula é vísivel em 3.17.

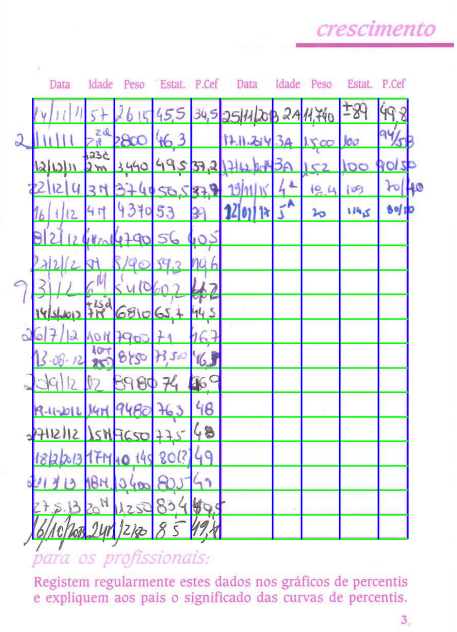

Figura 3.17: Células da tabela.

3.4.2.3 Identificação de caracteres

O primeiro passo na identificação dos caracteres é isolar os caracteres existentes na célula. O processamento necessário para cada célula é composto por: é aplicada uma escala de cinzentos, um filtro gaussiano de forma a remover imperfeições, a mesma restrição, de forma a encontrar a tinta da caneta por ser mais escura que o resto da imagem, que em 3.4.1.3, isto é, uma máscara que identifique as zonas mais escuras da imagem. Além disto é feita uma dilatação de forma a realçar as fronteiras entre os caracteres. Um exemplo de célula isolada é vísivel em 3.19.

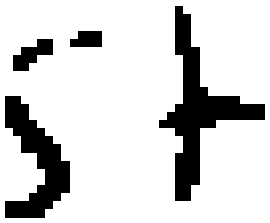

Figura 3.18: Exemplo de célula individual.

Os limites dos caracteres são calculados utilizando a função *findContours*. Considera-se válido um caractere que no máximo ocupe 35% da célula e no mínimo 5%

Em casos que a área encontrada pelo *findContours* seja superior a 35%, assume-se que mais que um caractere está presente na área e por isso, é necessário determinar quantos caracteres estão presentes nessa área. Para determinar essa área uma solução implementada foi associar a cada percentagem um número possível de caracteres nessa área e de seguida fazer uma contagem

de pixeis escuros e caso sejam mais escuros que claros, então assume-se que está no intervalo superior. Uma tabela foi desenvolvida para a decisão a ser tomada nesse instante, visível em 3.1.

Percentagem	Nº de caracteres
5% - 35%	1
35% - 45%	1/2
45% - 60%	2/3
60% - 80%	3/4
80% - 100%	4/5

Tabela 3.1: Tabela da percentagem associada ao nº de caracteres.

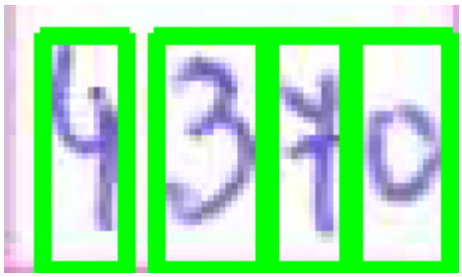

Figura 3.19: Exemplo de célula com caracteres individualizados.

No final do processo de identificação dos caracteres e em caso de sucesso, na identificação do número de caracteres, a imagem é guardada num diretório que posteriormente é utilizada para fazer a previsão. O nome do ficheiro contém a coluna, a linha e a posição do caractere na célula. Tendo como exemplo a imagem da Figura 3.19, que representa os caracteres identificados na tabela mostrada na Figura 3.17 na TERCEIRA coluna e quinta linha, obtemso as imagens mostradas nas figuras 3.20, 3.21, 3.22 e 3.23.

Figura 3.20: 2_5_1.png.

Figura 3.21: 2_5_2.png.

Figura 3.22: 2_5_3.png.

Figura 3.23: 2_5_4.png.

3.4.2.4 Previsão de caracteres usando *Keras MNIST*

O passo seguinte corresponde à previsão individual de cada caractere. Para isso é feita uma primeira primeira previsão utilizando o dataset *MNIST*, que consiste em 60000 exemplos de treino, conjuntamente com 10000 exemplos de teste. Estes exemplos são caracteres manuscritos para cada um dos algarismos arábicos que servem para treino e teste de redes neuronais. O maior contratempo da utilização deste dataset é a necessidade de formatação das entradas. As entradas são formatadas para terem 28x28 pixeis, além disso, o algarismo tem de ser branco em fundo preto.

De forma a modelar e compor as redes neuronais foi utilizada a ferramenta Keras [3], uma ferramenta proveniente do *engine* Tesseract [8], disponível em Python 3 que é a linguagem principal de desenvolvimento nesta dissertação

O modelo de rede neuronal base é **sequencial**, isto é, um *stack* de *layers*, com exactamente uma saída e uma entrada. Este modelo é apropriado ao problema pois apenas existe uma entrada e uma saída.

No entanto, ao longo do processo de composto pelas seguintes *layers*: *Conv2D*, *MaxPooling2D*, *Flatten*, *Dense* e *Dropout*.

- **Conv2D**: Esta camada cria um núcleo de convolução que se convolve com a camada de entrada para produzir um tensor de saídas.
- **MaxPooling2D**: Reduz a entrada ao longo das suas dimensões espaciais (altura e largura), tomando o valor máximo sobre uma janela de entrada (de tamanho definido pelo tamanho da piscina) para cada canal da entrada. A janela é deslocada por passos ao longo de cada dimensão.
- **Flatten**: Aplana a entrada. Não afecta o tamanho do lote.Por exemplo a entrada **(None, 1, 10, 64)** compõe a saída **(None, 640)**.
- **Dense**: Uma camada regular de rede neuronal composta densamente.
- **Dropout**: A camada Dropout define aleatoriamente unidades de entrada a 0 com uma frequência de taxa em cada etapa durante o tempo de treino, o que ajuda a evitar o excesso de equipamento. As entradas não ajustadas a 0 são aumentadas em 1/(1 - taxa) de modo a que a soma sobre todas as entradas permaneça inalterada.

Além disto, é importante mencionar que o *dataset* juntamente com a rede neuronal resultam num nível de certeza para cada algarismo arábico, ou seja, cada caractere de "0"a "9"vai ter associado um nível de certeza de 0% a 100%. Contudo, este processo é explicado com maior rigor e detalhe no Capítulo 4.

3.5 Tratamento dos Dados

Um passo importante que compõe a dissertação é o tratamento da informação. Este passo impacta tanto as saídas das previsões das redes neuronais como as entradas da rede neuronal a ser criada na iteração seguinte.

De forma a guardar as variáveis, recorre-se à biblioteca *pickle*[13], cuja função é a serialização e desserialização de estruturas de objetos *python*, isto é, converte variáveis *python*, neste caso *lists*, em *stream* de *bytes* a ser guardadas em ficheiros.

Tendo isto em conta, no final de cada previsão, as variáveis a ter em conta são as seguintes:

- **número**: o algarismo a que está associado o nível de certeza.

- **percentagem**: a percentagem correspondente à certeza da rede que o **caractere** a ser previsto é o **número**.
- **linha**: a linha correspondente à célula, onde está o caractere que está a ser previsto.
- **coluna**: a coluna correspondente à célula, onde está o caractere que está a ser previsto.
- **caractere**: a posição do caractere na célula, de forma sequencial, isto é, o número "1"corresponde ao primeiro caractere da célula, o número 2 corresponde ao segundo caractere da célula, o número "n"corresponde ao n-ésimo caractere da célula, etc.
- **max**: o **número** que obtém a maior **percentagem** de certeza relativa ao **caractere** a ser previsto.

Depois de feita a previsão, é feita uma verificação de qual a coluna correspondente à célula, de maneira a separar as previsões por tipo de dados. Essa separação pode ser vista na Tabela 3.2.

coluna	Dados
1	Idade
2	Peso
3	Estatura
4	Perímetro Cefálico

Tabela 3.2: Valor da variável **coluna** associada ao tipo de dados.

É de realçar que é necessário fazer uma reordenação, devido ao facto de que o nome do ficheiro, em ordem alfabética vai priorizar os caracteres que têm os algarismos mais reduzidos, ao invés de priorizar por ordem numérica.

Tendo este facto em conta, é utilizado um *zipper* de forma a guardar como *tuple* as previsões, juntamente com o valor correspondente à linha. De seguida, ordena-se o tuple como um todo, tendo em conta o valor da linha. Finalmente o tuple é desfeito e é guardado apenas o valor do peso, de forma ordenada num ficheiro chamado "**X**Predictions.txt", sendo que o valor de **X** é substituído pelo tipo de dados correspondente à coluna, usando a Tabela 3.2.

A interpretação de dados passa por determinar se a previsão está correta ou errada e para o efeito foram desenvolvidas duas soluções: uma solução manual e uma solução automatizada, utilizando os dados retirados dos gráficos cruzados com os dados retirados das tabelas.

3.5.1 Tratamento manual

Para o tratamento manual dos dados, foi desenvolvida uma interface que apresenta ao utilizador a imagem do caractere seguida de uma janela pop-up que pergunta ao utilizador se o número apresentado na imagem corresponde ao que foi determinado na Secção 3.4.2.4, isto é, a variável **max**. Esta interface é visível na Figura 3.24.

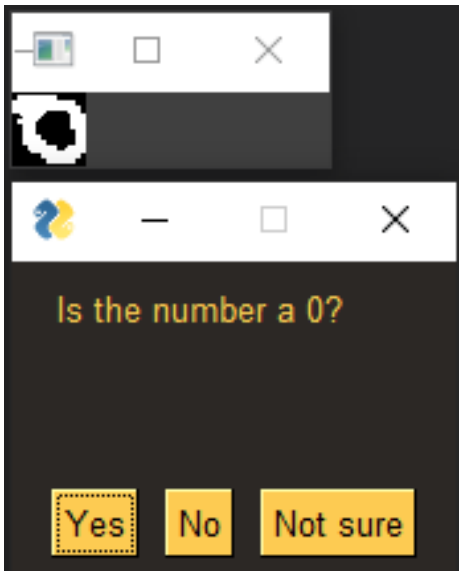

Figura 3.24: Exemplo da interface desenvolvida

Em caso positivo, a imagem é guardada num diretório chamado **Manual Dataset**, e no interior desse dataset fica num diretório com o nome **charX**, sendo que X é substituído pelo algarismo a que a imagem está associada.

Em caso negativo, um segundo pop-up surge ao utilizador de forma a questionar qual o algarismo que está presente na imagem. Caso o utilizador consiga identificar um algarismo, terá de introduzir um algarismo de 0-9, caso não consiga identificar ou haja um erro de processamento de imagem, o utilizador tem uma opção para não adicionar a imagem ao dataset manual. Caso seja dado um algarismo, a imagem segue o mesmo caminho que no passo anterior com a excepção de que ao invés de ser introduzido no diretório em que X corresponde ao valor com maior previsão no 3.4.2.4, é introduzido no diretório em que X corresponde ao valor introduzido pelo utilizador.

Em casos de erro de processamento, isto é, em casos que mais que um caractere está presente, o utilizador tem acesso a um botão "Mais do que um dígito", que no pop-up seguinte pergunta ao utilizador quantos caracteres estão presentes na imagem. Dada a resposta, a imagem é redirecionada para um diretório onde estão todas as imagens com esse número de caracteres.

Cada imagem nesse diretório contém no nome informação relativa ao boletim a que pertence, coluna, linha e posição dentro da célula da imagem. Cada item deste diretório pode ser depois reprocessado de forma a melhor consistentemente o processamento da aplicação.

3.5.2 Tratamento por cruzamento de dados

O tratamento automatizado dos dados é o processo sem intervenção humana que trata as imagens e as previsões. No caso ideal, este é o processo a ser utilizado pelo produto final. Este processo utiliza os dados retirados do gráfico como base de forma a precisar se os valores retirados das tabelas são os correctos.

O primeiro passo passa por transitar os valores do formato string para o formato de inteiros. Por exemplo, os valores retirados dos gráficos do Peso dos 0 aos 2 anos estão em "gramas". Sendo assim, independentemente da formatação definida pelo número de caracteres, é necessário transitar o valor para "gramas".

Para o processo de tratamento de informação proveniente das tabelas, é feito o seguinte processo:

- Verifica todos os dígitos mais significativos de cada item da lista, até se chegar ao primeiro "9". Caso o próximo dígito seja um "1", então considera-se que a primeira lista é a lista de números com 4 algarismos (lista A) e a segunda lista terá 5 algarismos (lista B).

- É então feita uma verificação de quantos *chars* tem cada item da lista. No caso da lista A, são adicionado *chars* com o valor "0"ate chegar ao tamanho 4 *chars*. No caso da lista B, são adicionado *chars* com o valor "0"ate chegar ao tamanho 5 *chars*.

- As listas são então juntas numa só, tendo a lista A os primeiros itens, seguidos pelos itens da lista B.

- É então percorrido um ciclo que altera cada um dos valores de *string* para *ints*.

- São agora removidos os outliers, valores que não fazem sentido no contexto actual, caso existam. São estes, valores que são mais altos do que os próximos dois valores, ou mais altos que o próximo valor em 200%.

Dado que os valores dos gráficos vão ter consistentemente um erro associado à precisão humana de definição do ponto, aliada a eventuais deformações de imagem e de processamento, é necessário atribuir um erro de precisão aos gráficos, o nome atribuído a esse erro será **erro de precisão**.

Tendo isto em conta, cada item da lista unificada é comparada com cada item da lista das previsões do gráfico no intervalo do erro de precisão. Isto é, cada item da lista da previsão de caracteres da tabela é verificado se está contida no intervalo de cada item da lista de previsões do gráfico subtraído pelo erro de precisão até a esse mesmo item somado ao erro de previsão. Sendo a lista A a das tabelas e a lista B a dos gráficos, o pseudocódigo em seguinte, decompõe o procedimento na prática.

```
1 for aItem in ListaA:
2     for bItem in ListaB:
3         if aItem > bItem - erroPrecisao and aItem < bItem + erroPrecisao:
4             listaCorretos.adicionar(aItem)
```

Listing 3.1: Algoritmo de seleção de dados.

No final, os valores que tiverem uma associação, são considerados correctos e por isso os seus caracteres são considerados para acrescentar ao dataset inicial, da mesma forma que acontece em 3.5.1.

A saída, em consola, do algoritmo automático pode ser observada na figura 3.25.

```
Dados retirados das tabelas
[4400, 4770, 5140, 8800, 9400, 10145, 10400, 11200, 15000, 15200]
Dados retirados dos gráficos
[[1.0, 3616], [2.0, 4239], [4.0, 4260], [6.0, 6837], [9.0, 8458], [12.0, 9705], [15.0, 9809], [18.0, 11139], [20.0, 11160], [24.0, 11721]]
Pares escolhidos
['11200', '11200']
Correspondem a(s) linha(s)
[19]
Da coluna do Peso
```

Figura 3.25: Exemplo de saída do algoritmo automático para o utilizador.

Uma versão do código referente ao peso pode ser encontrada no anexo A.1.

3.5.3 Comparação entre soluções

De forma a avaliar as duas soluções decidiu-se que os tópicos seriam os seguintes:

- **Velocidade de execução**: Tempo que demora em média a processamento de cada caractere, a partir do momento em que é previsto, até ao momento em que é seleccionado para se adicionar ao dataset.
- **Propensão a erros**: Probabilidade de cada caractere ser adicionado erradamente à lista de seleção de caracteres a ser adicionados ao dataset.
- **Volume de dados adicionados**: Número de caracteres selecionados para ser adicionados ao dataset por boletim.

Como mencionado na Secção 3.5.2, o tratamento automático de dados é o que é pretendido de ser a solução definitiva da aplicação, no entanto, o tratamento manual é útil para desenvolvimento da aplicação e para utilizar, caso não haja cruzamento de dados com gráficos, nos casos em que estes não estejam preenchidos.

O tratamento automático de dados é o mais rápido em termos de velocidade de execução, sendo capaz de processar todos os dados de uma tabela em alguns segundos, enquanto que o tratamento de dados demora algumas dezenas de minutos, dependendo da velocidade do utilizador.

O tratamento manual apresenta menor propensão a erros, pois o utilizador consegue definir (por norma) qual o caractere certo, enquanto que o automático por vezes pode cometer enganos. Além disso o utilizador consegue identificar quando houve erros de processamento e aparece mais do que um caractere na imagem, o processamento automático não tem essa capacidade.

Em termos de volume de dados, o utilizador consegue extrair o maior número de caracteres de um dado boletim, pois consegue definir com maior certeza cada um dos caracteres. O tratamento automático em caso de dúvida exclui esses caracteres.

Tendo estes fatores em conta, pode-se afirmar que dependendo a tarefa em mão, é possível utilizar cada solução independentemente. Se a tarefa requerer o processamento de todos os boletins, a solução mais viável seria a automática, em casos de melhoria de precisão da aplicação ou se for necessário retirar toda a informação de um boletim em específico, a solução a ser utilizada seria a a manual. Em casos de resolução de bugs, a melhor solução é a manual e em casos de melhoria de performance, a solução seria a automática.

Capítulo 4

Métodos de Ensembles

Este Capítulo tem como propósito utilizar *ensemble learning* de forma a melhorar a performance da rede neuronal a cada boletim analisado. Além disso, este Capítulo tem como objetivo apresentar os progressos implementados no que toca ao *ensemble learning*. A Secção 4.1 explica o conceito geral que se implementou como base de metodologia. A Secção 4.1.1 apresenta a rede de base a ser utilizada de forma a inicial, enquanto as Secções 4.1.2 e 4.1.3 apresentam as redes alternativas de redes neuronais necessárias para a implementação de *ensemble learning*. A Secção 4.1.4 apresenta o comportamento de um meta-decisor que utiliza critérios de forma a tomar uma decisão acerca do problema em questão.

Além disto, é de importante menção referir que todas redes mencionadas neste Capítulo são construídas utilizando a ferramenta Keras[3], que oferece os recursos necessários para a sua configuração, compilação, treino e teste.

4.1 Stacked Generalization

Os métodos de **Stacked Generalization** utilizam as saídas de determinados algoritmos de redes neuronais como entradas para o treino e teste das redes neuronais seguintes. Fazendo este processo de forma iterativa e à medida que se adicionam cada vez mais *samples* no treino e teste de redes neuronais, o objetivo é aumentar a performance da aplicação na previsão de caracteres manuscritos.

Em específico, na aplicação, o objetivo é utilizar as filtragens feitas no tratamento de dados em 3.5, assumindo que a filtragem remove todas as previsões erradas, adicionar as imagens a um dataset manual, adaptar o algoritmo de forma adaptar-se ao novo dataset, sendo este o dataset composto pelo Keras MNIST e pelo dataset manual, composto por previsões corretas de boletins previamente analisados.

Sendo assim, de modo inicial, é feita a filtragem manual 3.5.1 até ser possível obter 50 imagens de cada caractere, retirados diretamente dos boletins. E a partir desse momento, utilizar a filtragem automática 3.5.2.

A partir desse momento, introduz-se a utilização de *Ensemble Learning*, em que são utilizados diversos modelos de redes neuronais, treinados com o dataset MNIST, os seguintes com o dataset MNIST+Manual. Além da diferença no teste e treino, também é alterado o modelo da rede neuronal, apesar de todos os utilizados serem sequenciais.

A figura 4.1 mostra o processo utilizado na previsão dos caracteres.

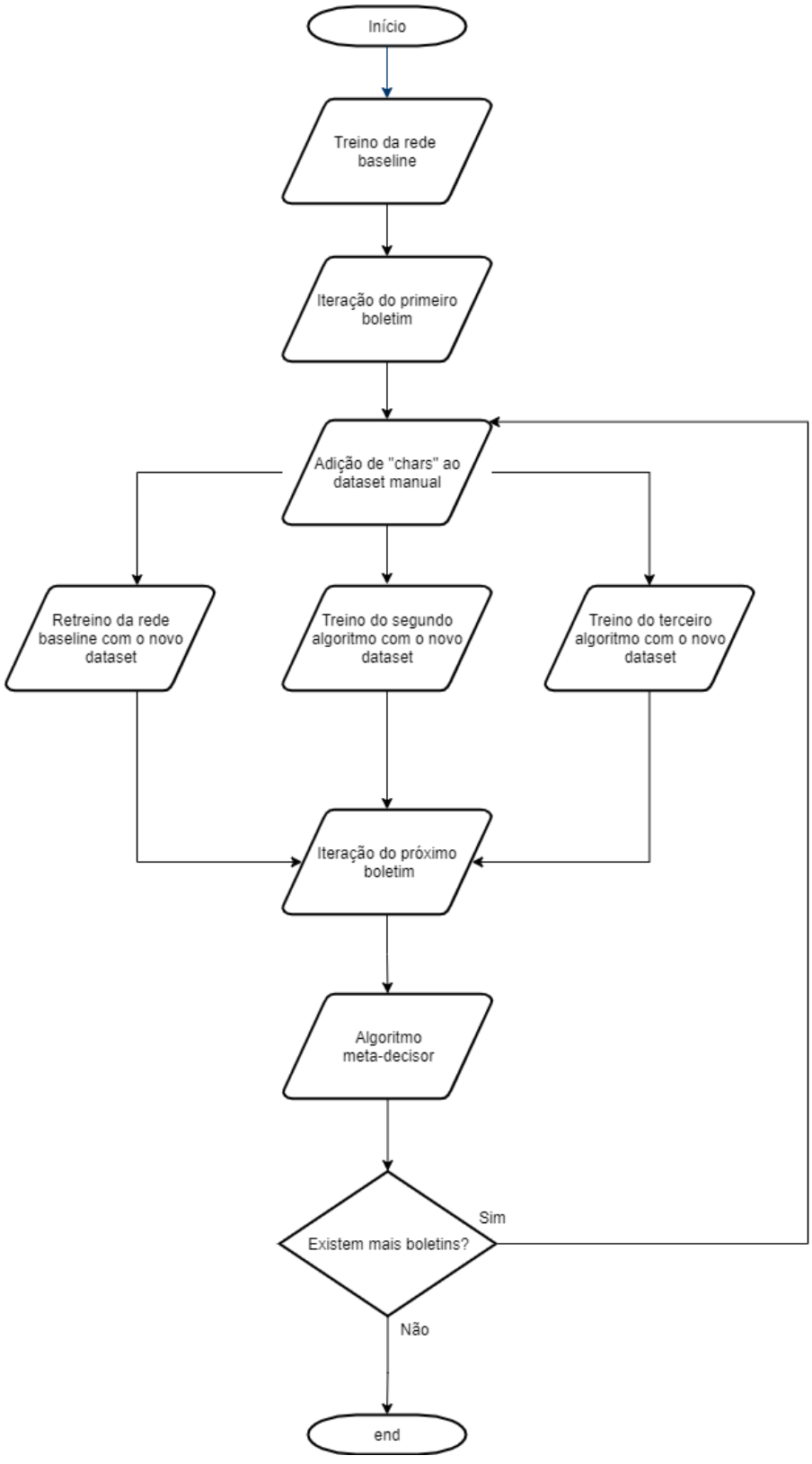

Figura 4.1: Processo de previsão de caracteres.

4.1.1 Baseline

Conforme mencionado na Secção 3.4.2.4, o modelo utilizado pela *baseline* é **Sequencial**, com apenas uma entrada e uma saída, com as *layers* ordenadas da seguinte forma: uma layer densa com 512 saídas, com a ativação em *ReLu*, seguida de uma layer de dropout com rate igual a "0,2", seguida de outra layer densa similar à anterior, com uma layer de dropout também similar à layer de dropout anterior. De forma a terminar a rede com 10 saídas referentes à percentagem de cada item, é sempre necessário que a última layer seja uma densa com 10 saídas (uma para cada algarismo), sendo que a soma da percentagem dada a cada algarismo tem de dar 100%. Tendo isto em conta, utiliza-se a função ativada "softmax"que executa exatamente esse processo.

```
1  def baseline_model():
2      model = Sequential()
3      model.add(Dense(512, activation='relu', input_shape=(input_size,),
           kernel_initializer='normal'))
4      model.add(Dropout(0.2))
5      model.add(Dense(512, activation='relu'))
6      model.add(Dropout(0.2))
7      model.add(Dense(10, kernel_initializer='normal', activation='softmax'))
8
9      model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['
           accuracy'])
10     return model
11
12 model = baseline_model()
13
14 model.fit(x=x_train, y=y_train, validation_data=(x_test, y_test), shuffle=False,
       epochs=100, batch_size=20, verbose=1)
15
16 model.save("baseline_model.h5", overwrite=True)
```

Listing 4.1: Algoritmo baseline para o primeiro modelo treinado pelo dataset MNIST.

O modelo é então compilado, isto é, fica configurado e guardado na variável definida, com o a *loss function* definida *categorical_crossetropy*, que é indicada quando existem 2 ou mais *labels*, que é o caso pois as *labels* correspondem aos algarismos "0"a "9". O otimizador utilizado é o *adam* e a métrica escolhida para se avaliar a rede é a precisão da rede.

A este modelo de base é alimentado o dataset MNIST, que tem composto em si 60000 entradas de treino e 10000 entradas de teste (que no *listing* 4.1 correspondem ao x_train e x_test), assim como as respetivas samples (dadas por y_train e y_test no *listing* 4.1). O modelo é então treinado e testado utilizando o método *fit*, com 100 *epochs*. *Epochs* definem o número de iterações para o qual o modelo vai ser treinado.

Depois de treinada e testada a rede é guardada no diretório de forma a poder ser utilizada mais tarde.

4.1.2 Segundo algoritmo

Conforme mencionado anteriormente, de forma a ser possível por em prática o método de *ensemble learning: Stacked Generalization*[14], é necessário oferecer variabilidade ao meta-decisor.

O primeiro passo deste processo incide em criar uma rede que apesar de poder ter algumas semelhanças à rede de *baseline*, altera as suas camadas de forma a poder chegar a resultados diferentes.

Esta rede é baseada no modelo apresentado em [5]

```
1  model = Sequential()
2
3  model.add(Conv2D(28, kernel_size=(3, 3), input_shape=input_shape))
4  model.add(MaxPooling2D(pool_size=(2, 2)))
5  model.add(Flatten())
6  model.add(Dense(128, activation=tf.nn.relu))
7  model.add(Dropout(0.2))
8  model.add(Dense(10, activation=tf.nn.softmax))
9
10 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy
       '])
11 model.fit(x=x_train, y=y_train, epochs=100)
12
13 model.evaluate(x_test, y_test)
14
15 model.save('cnn_mnist_1.model')
```

Listing 4.2: Segundo modelo treinado pelo dataset MNIST+manual.

Neste algoritmo, a camada única é composta por: uma layer convolucional 2D (**Conv2D**), seguida de uma layer de Max Pooling 2D (**MaxPooling2D**), seguida de uma layer de aplanamento (**Flatten**), uma layer densa com a função *relu* activa (**Dense**), seguida de uma layer de dropout (**Dropout**) e a última layer é densa também mas desta vez com a função *softmax* ativa. A entrada da rede é uma matriz 28 por 28, ou seja, a imagem formatada em que cada valor da matriz corresponde a um pixel. As saídas são os algarismos "0-9".

Sendo assim, a primeira layer convoluciona o input de forma a produzir um tensor de inputs, utilizando um *kernel* 3 por 3. Ao que resultar disto é aplicado um max pooling 2d, com uma pool size 2 por 2, isto resulta numa redução do tamanho da matriz pois apenas o máximo de cada matriz 2 por 2 dentro da matriz permanece na matriz final. De seguida, o output é aplanado ou alisado com a layer **Flatten**, que tem como propósito reduzir a dimensão sem reduzir o tamanho do lote. A camada seguida é uma camada densa da qual resultam 128 saídas, com a ativação *ReLu* que consiste em aplicar a função de activação da unidade linear rectificada. Com valores por defeito, isto devolve a activação padrão *ReLU*: max(x, 0), o máximo de 0 no sentido do elemento e o tensor de entrada. De seguida, contém uma camada *dropout* que põe um valor a 0 com chance de acontecer 20%, os que não têm *drop*, sofrem um incremento de 125%. Isto acontece de forma a

evitar *overfitting*. A última camada é uma camada densa com a função softmax ativada que atua da seguinte forma: converte um vector de valores para uma distribuição de probabilidade e os elementos do vector de saída estão no intervalo (0, 1) e soma a 1. O *softmax* é frequentemente utilizado como activação para a última camada de uma rede de classificação porque o resultado poderia ser interpretado como uma distribuição de probabilidade.

4.1.3 Terceiro algoritmo.

No momento de elaboração da dissertação, o último algoritmo desenvolvido incide sobre uma rede neuronal derivada da anterior, mas com algumas diferenças cruciais nas camadas envolventes. Este fator é necessário, mais uma vez, para gerar variabilidade nas respostas dadas ao meta-decisor.

```
model = Sequential()

model.add(Conv2D(30, (3, 3), input_shape=input_shape, activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(15, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(10, activation='softmax'))

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

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_split=(5000.0/60000.0),
          shuffle = True)

# Save model
model.save("cnn_mnist_2.h5", overwrite=True)
```

Listing 4.3: Terceiro modelo treinado pelo dataset MNIST+manual.

As principais diferenças entre esta rede e a anterior, incidem na dupla convolução, seguida de *max pooling*. Além desta diferença também existe uma tripla camada densa, que vai diminuindo o número de saídas desde a primeira camada densa até à última.

Além disto, existem diferenças no treino da rede, esta rede tem o *Shuffle* ativado assim como um *split* de validação diferente ao utilizado na Secção 4.1.2. O *Shuffle* é um item aleatório que avalia se existe uma alteração na ordem, antes de cada *epoch*. Estes dois fatores contribuem significativamente para a alteração das saídas da rede.

4.1.4 Meta-decisor

Dada arquitectura definida na Figura 4.1, são usados os 3 algoritmos na iteração de todos os boletins excepto o primeiro. Nesses casos, é necessário definir um meta-decisor, dado por um pequeno algoritmo, que chegue a uma decisão acerca de qual o algarismo correspondente à imagem. Para o efeito é necessário utilizar as saídas das redes neuronais.

No momento da escrita da dissertação, a hipótese a ser implementada no momento da decisão consiste num algoritmo que tem em conta as percentagens nas saídas das redes neuronais. De forma a mais facilmente identificar cada uma das saídas, a saída do algoritmo original denomina-se como "SAÍDA 1", a saída do segundo como "SAÍDA 2"e a saída do terceiro como "SAÍDA 3". Cada uma destas saídas vai ter 10 valores, cuja soma é 100%. Um exemplo da saída de cada um dos algoritmos seria o apresentado na imagem da Figura 4.2.

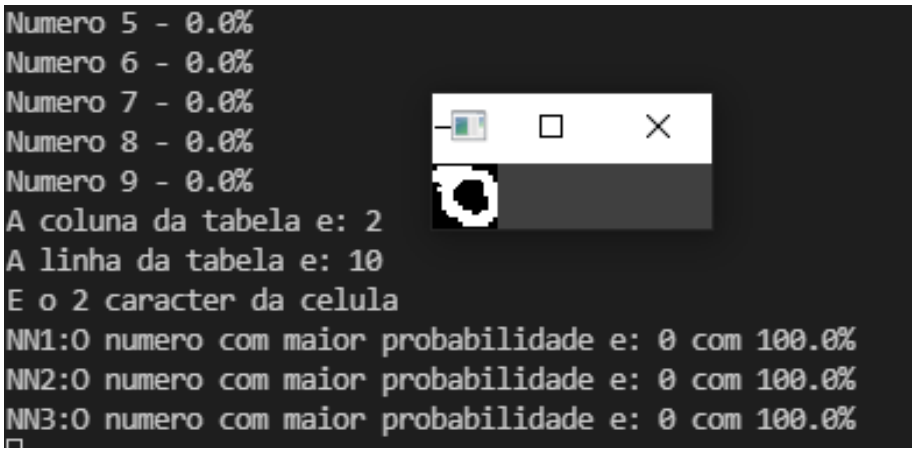

Figura 4.2: Exemplo da saída para uma imagem em que todos as redes têm 100% de certeza.

Em casos em que nem todos os algoritmos estão de acordo na decisão de qual o caractere correspondente à imagem, e visto que cada algoritmo tem uma saída relativa a cada um dos caracteres, é feita uma soma da percentagem de cada um dos caracteres para cada um dos algoritmos. No final, divide-se por 3 o valor relativo a cada um dos caracteres e esse valor é reflexão da média dos 3 algoritmos para esse caractere. Tomando agora como exemplo a previsão visível na Tabela 4.1, a tabela na primeira coluna, tem o caractere a ser comparado com a imagem, na segunda, a previsão do algoritmo original, na terceira coluna, tem a previsão do segundo algoritmo e na quarta coluna, tem a previsão para a dada imagem do terceiro algoritmo. Além disto, pode-se verificar que todas as redes têm a soma de todas as previsões dos caracters igual a 100%, é possível então verificar a utilidade da função *softmax*, referida nas subsecções 4.1.1, 4.1.2 e 4.1.3.

Tendo em conta o exemplo das previsões, e atribuindo o mesmo peso a todos os algoritmos, chega-se à Tabela 4.2. Assim sendo, a imagem será associada ao caractere "7", pois este valor é o máximo, fazendo a média entre todos os algoritmos.

Em casos que se verifique que um certo algoritmo tem mais chance de acerto, é possível atribuir mais peso a esse algoritmo no cálculo da média, calculando assim uma média ponderada.

Caractere	NN1	NN2	NN3
0	12%	0%	10%
1	50%	35%	25%
2	0%	10%	0%
3	0%	0%	0%
4	0%	0%	0%
5	0%	0%	0%
6	0%	0%	0%
7	38%	20%	65%
8	0%	0%	0%
9	0%	35%	0%

Tabela 4.1: Exemplo de previsões para uma dada imagem.

Caractere	Soma	Média
0	22	7,33%
1	110	36,67%
2	10	3,33%
3	0	0%
4	0	0%
5	0	0%
6	0	0%
7	123	41%
8	0	0%
9	35	11,67%

Tabela 4.2: Exemplo das médias das previsões para uma dada imagem.

Para o mesmo exemplo dado em 4.1, com um peso de 0,5 para o primeiro algoritmo, um peso de 1,75 para o segundo e 0,75 para o terceiro algoritmo, resultam na Tabela 4.3.

Caractere	Soma	Média
0	13,5	4,5%
1	105	35%
2	17,5	5,83%
3	0	0%
4	0	0%
5	0	0%
6	0	0%
7	102,5	34,25%
8	0	0%
9	61,25	20,42%

Tabela 4.3: Exemplo das médias ponderadas das previsões para uma dada imagem.

Tendo a Tabela 4.3 em conta, o algarismo escolhido passará a ser o "1". Isto acontece devido

ao nível de confiança atribuído ao segundo algoritmo, que considera o caractere "1"como sendo o mais provável, com 15 pontos percentuais de diferença. À partida, estes 15 pontos podem não fazer a diferença, no entanto, dado o nível de confiança ser 1,75 neste algoritmo, e feitos os cálculos, a média ponderada aponta ao resultado obtido.

4.2 Resumo e conclusões

Este capítulo compila os conceitos de principal importância no contexto da dissertação e providenciam uma análise cujas conclusões retiradas são:

- Consolidação a utilização de métodos de *ensemble* para maior consistência na obtenção de resultados no que toca à previsão;
- Reforço da ideia de que utilizando um dataset mutável e incremental, de forma a obter sempre os dados mais atualizados no treino das redes neuronais;
- Abertura da possibilidade da atribuição de pesos a cada algoritmo, tendo em conta a confiança que o utilizador terá em cada um deles.

Capítulo 5

Conclusões e trabalho futuro

É de realçar a importância que estes dados podem ter para a pediatria em Portugal. São dados extremamente importantes pois são específicos ao país e aos seus habitantes. Devem ser realizados esforços para extrair esta informação e é isso que esta dissertação almeja alcançar.

Este Capítulo tem como objetivo apresentar as conclusões retiradas após a concretização desta dissertação. Para isso, a primeira Secção 5.1 apresenta as principais contribuições do trabalho realizado considerando que o trabalho desenvolvido foi um passo dado na direcção certa estabelece alicerces para que mais progresso possa ser feito no futuro. A Secção 5.2 expõe as principais limitações que existem ainda na aplicação e por último a Secção 5.3 apresenta alguns tópicos de trabalho futuro que consigam ultrapassar essas limitações e que permitam a melhoria do objeto de estudo desta dissertação.

Além disso, é de realçar que as tecnologias de **Ensemble Learning** estão a ser desenvolvidas de uma forma exponencial. As implementações realizadas no âmbito desta dissertação apresentam uma forte contribuição para o progresso deste tema.

5.1 Contribuições

Tendo como ponto de partida as contribuições teóricas de [4], assim como as contribuições teóricas e práticas de [5], foi possível conceber uma aplicação com uma *pipeline* bem definida assim como um processo bem definido de processamento de imagem, assim como métodos de ML para previsão de caracteres.

Relativamente às contribuições inteiramente novas feitas por esta dissertação, existe o facto de a aplicação ter sido inteiramente desenvolvida em *Python 3*, algo que não era possível anteriormente. Foram atualizados métodos e dados para estarem a par das novas bibliotecas, como o *OpenCV* e o *Keras* do *Tensorflow*. Esta dissertação também contribuiu com uma interface de forma a ser possível fazer-se tratamento de dados manualmente.

É também de crucial importância mencionar que foi desenvolvido um algoritmo de cruzamento de dados das tabelas com os gráficos, que é extremamente importante na seleção de dados automática.

Foi implementado pela primeira vez neste projeto um método de ensemble, que apesar de ainda estar numa fase inicial, será certamente de utilidade essencial no produto final.

5.2 Limitações

No momento atual, as limitações da aplicação podem ser identificadas pelos tópicos que se seguem.

- **Erros de digitalização** - Neste momento, existe a probabilidade de haver erros de digitalização ou de escrita nos livros que são impossíveis de prever com precisão.
- **Baixa eficiência dos modelos** - Os modelos de redes neuronais desenvolvidos são limitados a um caractere por rede o que torna o processo preditivo mais lento.
- **Validação da aplicação** - Devido ao rápido desenvolvimento de algumas tecnologias usadas nesta aplicação, existe uma alta probabilidade de depreciação de software.

5.3 Trabalho Futuro

Futuramente, à aplicação desenvolvida podem ser adicionados os seguintes tópicos.

- Melhorias concretas no processamento de imagem que envolve a segmentação e a separação dos caracteres dentro das células dos caracteres. Este tópico pode ser melhorado, caso haja melhorias no software das bibliotecas utilizadas, nomeadamente o *OpenCV*.
- Adição de maior número de classificadores, isto é, modelos de redes neuronais, que podem ser utilizadas no processo de *Ensemble Learning*. Este fator é escalável, por isso, qualquer adição de algoritmos é uma melhoria ao que foi desenvolvido.
- Aumentar o número de exemplos de forma a obter uma aplicação mais adaptativa. Além de diminuir a chance de erro, é também uma maneira de acrescentar *templates* aos datasets desenvolvidos.

Anexo A

Anexos

A.1 Código referente ao cruzamento de dados entre gráfico e tabela

```
import cv2
import os
import re
import pickle
import keras
from utils import *
from shutil import copyfile


dir = "characters/"
coluna = "2_"
dest = "Recortes/"
os.chdir('1179_final')

#Peso (0-2) - page3.png
#Comprimento (0-2) - page4.png
#Peso (2-20) - page5.png
#Estatura (2-20) - page6.png
#Per.Cefalico (0-3) - page7.png
#IMC (2-20) - page8.png

#fileName = input("filename: ")
fileName = "page3.txt"
file1 = open(fileName, 'rb')
chartDataPeso = pickle.load(file1)
file1.close()

fileName = "PesoPredictions.txt"
file2 = open(fileName, 'rb')
tableDataPeso = pickle.load(file2)
file2.close()
```

```
32
33 fileName = "PesoOrdem.txt"
34 file3 = open(fileName, 'rb')
35 tableDataOrdem = pickle.load(file3)
36 file2.close()
37
38
39
40
41 sortedDatedChart = []
42 auxList = []
43
44 for item in chartDataPeso:
45     auxItem = [item[1],item[2]]
46     auxList.append(auxItem)
47
48
49 sortedDatedChart = sorted(auxList)
50
51 #REMOCAO DE REPETIDOS
52 i=0
53 while(i<len(sortedDatedChart)-1):
54
55     if(sortedDatedChart[i][0]==sortedDatedChart[i+1][0]):
56         sortedDatedChart.remove(sortedDatedChart[i])
57         i=i-1
58
59     i=i+1
60
61 #REMOCAO DE OUTLIERS
62 i=1
63 while(i<len(sortedDatedChart)-1):
64     if(sortedDatedChart[i][1]>sortedDatedChart[i+1][1] or sortedDatedChart[i][1]<
           sortedDatedChart[i-1][1]):
65         sortedDatedChart.remove(sortedDatedChart[i])
66     elif(sortedDatedChart[i][1]>sortedDatedChart[i+1][1] and sortedDatedChart[i
           ][1]>sortedDatedChart[i+2][1] ):
67         sortedDatedChart.remove(sortedDatedChart[i])
68     i=i+1
69
70
71 # print(tableDataPeso)
72 adjustedTableData = []
73
74 for item in tableDataPeso:
75     while len(item) < 4:
76         item = item + "0"
77     adjustedTableData.append(item)
78
```

```
79
80 adjustedTableData.reverse()
81 for index,item in enumerate(adjustedTableData)  :
82     if item[0]=='9':
83         break
84     while len(item) < 5:
85         item = item + "0"
86     adjustedTableData[index] = item
87 adjustedTableData.reverse()
88
89 for index,item in enumerate(adjustedTableData):
90     adjustedTableData[index]=int(item)
91
92 i=1
93 while (i<len(adjustedTableData)-1):
94     if(adjustedTableData[i]>adjustedTableData[i+1] or adjustedTableData[i]<
           adjustedTableData[i-1]):
95         adjustedTableData.remove(adjustedTableData[i])
96         tableDataOrdem.remove(tableDataOrdem[i])
97         i=i-1
98     elif(adjustedTableData[i]>adjustedTableData[i+1] and adjustedTableData[i]>
           adjustedTableData[i+2]):
99         adjustedTableData.remove(adjustedTableData[i])
100        tableDataOrdem.remove(tableDataOrdem[i])
101        i=i-1
102
103    i=i+1
104
105 print("Dados retirados das tabelas")
106 print(adjustedTableData)
107 print("Dados retirados dos graficos")
108 print(sortedDatedChart)
109 pairsList = []
110 lineList =[]
111 for i,item in enumerate(adjustedTableData):
112     for point in sortedDatedChart:
113         if (item < point[1] + 100 and item > point[1] - 100):
114             pair = (item)
115             pairsList.append(pair)
116             lineList.append(tableDataOrdem[i])
117
118 for index,item in enumerate(pairsList):
119     pairsList[index]= str(item)
120
121 #REMOVER DUPLICADOS
122 for i,item in enumerate(lineList):
123     if (lineList[i] == lineList[i+1]):
124         lineList.remove(lineList[i+1])
125
```

```
126 print("Pares escolhidos")
127 print(pairsList)
128 print(lineList)
129
130
131
132 filelist=os.listdir(dir)
133 for file in filelist[:]: # filelist[:] makes a copy of filelist.
134     if not(file.endswith(".png")):
135         filelist.remove(file)
136
137
138
139 for index,line in enumerate(lineList):
140     i=0
141     for file in filelist:
142         if file.startswith(coluna+str(line)+"_"):
143             print(file)
144             char=pairsList[index][i]
145             n=len(os.listdir(dest+"char"+char))
146             src = dir+file
147             destiny = dest+"char"+str(char)+"/"+str(n+1)+".png"
148             copyfile(src, destiny)
149             i=i+1
```

Referências

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Arindam Chaudhuri, Krupa Mandaviya, Pratixa Badelia, e Soumya Ghosh. Optical character recognition systems for different languages with soft computing. 352, 01 2017.

[3] Francois Chollet et al. Keras, 2015.

[4] Fábio André da Silva Amarante. Extracting medical information from personal child health records, 2018.

[5] David Rafael Silva Falcão. Extracção de informação dos boletins de saúde infantil e juvenil. 2020.

[6] Giorgio Giacinto e Fabio Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9-10):699–707, 2001.

[7] Jiawei Han, Micheline Kamber, e Jian Pei. Data mining: concepts and techniques, waltham, ma. *Morgan Kaufman Publishers*, 10:978–1, 2012.

[8] Anthony Kay. Tesseract: An open-source optical character recognition engine. *Linux J.*, 2007(159):2, julho 2007.

[9] Yann LeCun e Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[10] Gupta Mehul, Patel Ankita, Dave Namrata, Goradia Rahul, e Saurin Sheth. Text-based image segmentation methodology. *Procedia Technology*, 14:465–472, 2014. 2nd International Conference on Innovations in Automation and Mechatronics Engineering, ICIAME 2014.

[11] João Rodrigues Pereira. Modelos de data mining para multi-previsão : aplicação à medicina intensiva, 2005.

[12] SP Ruba Rani, B Ramesh, e JGR Sathiaseelan. Evaluation of stemming techniques for text classification. *International Journal of Computer Science and Mobile Computing*, 4(3):165–171, 2015.

[13] Guido Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[14] D. H. Wolpert. Stacked generalization. Relatório técnico LA-UR-90-3460, Los Alamos, NM, 1990. He proposes a more sophisticated voting scheme where the a second level function output performs the final estimation of the true class. I don't think this is going to referenced in the final thesis.

[15] L. Xu, A. Krzyzak, e C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[16] Nida M. Zaitoun e Musbah J. Aqel. Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806, 2015. International Conference on Communications, management, and Information technology (ICCMIT'2015).