Universidade da Beira Interior
Departamento de Informática

Nº 131 - 2022: ***[SUDOKU: Gerador + Solucionador Automático de Puzzles]***

Elaborado por:

Francisco Manuel Marques Pereira

Orientador:

Professor Doutor Hugo Pedro Proença

19 de junho de 2023

Agradecimentos

Queria deixar um agradecimento, ás pessoas que me ajudaram e motivaram na resolução deste projeto.

Em primeiro lugar, ao Professor Hugo Proença, pela ajuda que me deu e o conhecimento partilhado ao longo do tempo.

Em segundo lugar, aos meus pais Alice e José, que ajudaram me em mais do que um maneira no acabamento deste projeto.

E em ultimo lugar, a todos os outras pessoas, familiares e amigos que me ajudaram e motivaram de qualquer maneira possível.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

IBM International Business Machines

OCR Optical Character Recognition

PDF Portable Document Format

IA Inteligência Artificial

ICR Intelligent Character Recognition

Capítulo

1

Introdução

1.1 Enquadramento

No nosso tempo, nós como seres humanos observamos o progresso da ciência, várias tecnologias vão surgindo que são revolucionárias no seu campo, mas a que diz respeito a este projeto seria o campo de Inteligência Artificial e a tecnologia de "Machine Learning".

Este campo tenta simular o cérebro humano, e mais especificamente a capacidade do cérebro humano de detetar padrões, processá los e a capacidade de aprendizagem com esses padrões.

Logo se conseguíssemos programar computadores, para aprender a identificar e aprender com os padrões como nós fazemos, eles seriam capazes de resolver problemas demasiado difíceis para o cérebro humano.

E assim se espera que com estes avanços tecnológicos, seja possível resolver problemas, não só no campo de inteligência artificial, mas também em medicina, engenharia,etc, que até á atualidade seriam julgados impossíveis.

1.2 Motivação e Objetivos

Hoje em dia, vivemos num mundo que funciona cada vez mais num meio digital, logo seria necessário uma forma de processar grandes quantidades de dados, e os computadores nem sempre conseguem processá-los, especialmente se esses dados são números ou letras escritos manualmente por pessoas, então programas como este são necessários para processar rapidamente e eficazmente. Foi escolhido o sudoku para este programa, por ser um puzzle popular e relativamente simples de fazer e entender.

O objetivo principal deste projeto é desenvolver um programa capaz de criar puzzles sudoku, envia-los ao utilizador para resolução e depois correção do mesmo, e também a resolução dum puzzle sudoku dado pelo utilizador.

1.3 Organização do Documento

O relatório deste projeto está organizado da seguinte forma:

1. O primeiro capítulo – **Introdução** – aqui será apresentado o projeto, o enquadramento para o mesmo, a sua motivação e os seus objetivos, e a respetiva organização do relatório.

2. O segundo capítulo – **Estado da arte** – irá descrever os conceitos mais importantes no âmbito deste projeto, bem como a sua história e o funcionamento do Sudoku e um sumário das tecnologias de *Machine Learning* e *OCR* utilizadas.

3. O terceiro capítulo – **Método Proposto** – neste capítulo será descrito o método utilizado no desenvolvimento deste projeto. Conterá uma explicação detalhada do código e uma listagem e explicação de módulos, tecnologias e bibliotecas utilizadas na sua criação.

4. O quarto capítulo – **Testes** – demonstração do projeto, incluindo testes e os seus respetivos resultados.

5. O quinto capítulo – **Conclusão** – Por ultimo, neste capítulo iremos concluir o relatório do projeto e enunciar ideias para um trabalho futuro.

Capítulo

2

Estado da Arte

2.1 Introdução

Neste capítulo, iremos mostrar os modelos do estado da arte que fazem parte do funcionamento deste projeto.

Irá existir duas secções: a primeira secção que diz respeito ao **Sudoku** está dividida em duas subsecções, em que na primeira irá ser explicadas as origens e historia deste jogo e na segunda será explicado como o jogo funciona, as suas regras, etc... .

A segunda secção, que irá explicar os conceitos das técnicas de reconhecimento utilizadas, também está dividida em duas subsecções, a primeira sobre *Machine Learning* e a segunda sobre *Pattern Matching*.

2.2 Sudoku

2.2.1 Sudoku - História

O jogo que é conhecido hoje em dia como **Sudoku**, tem as suas origens no século 18, num jogo matemático que na altura era conhecido como *Latin Square*, que em Combinatória e Design Experimental um *Latin Square* é um array m por m constituído com n símbolos diferentes, em que cada um deles ocorre exatamente uma vez em cada fila e coluna.
O nome do jogo foi inspirado pelos artigos do matemático suíço Leonard Euler (1707 - 1783), que utilizava caracteres latinos como símbolos. Os primeiros puzzles utilizando números aparecem em jornais que foram publicados em França em 1895.

Mas o jogo moderno que reconhecemos hoje em dia foi inventado por Howard Garns um inventor de puzzles *freenlance* de Connersvile, Indiana, USA em 1979, onde foi publicado na revista **"Dell Pencil Puzzles and Word Games"**, onde era conhecido como **"Number Place"**, pois envolve colocar números em espaços vazios numa grelha 9 por 9.

O jogo ficou conhecido como **Sudoku** no Japão em 1984, em que o seu nome vem de da expressão japonesa *"Sūji wa dokushin ni kagiru"* que significa "os dígitos estão limitados em uma ocorrência.".

O homem que reintroduziu Sudoku de volta ao Ocidente foi um Juiz Neozelandês chamado Wayne Gould, que estava de férias em Tokyo em Março de 1997, e descobriu o Sudoky numa libraria. Rapidamente, ele tornou-se um grande fanático pelo jogo e passou os próximos 6 anos a desenvolver um jogo de computador que gerava puzzles de Sudoku.

O *"New York Times"* começou a publicar puzzles de Sudoku em 2004, e o primeiro jornal dos U.S.A a publicar Sudoku foi o *"The Conway Daily Sun"* de New Hampshire em 2004. Nos passados 20 anos , Sudoku transformou-se num fenómeno global, em que o primeiro torneio mundial de Sudoku foi sediado em Itália em 2006.

Enquanto as pessoas continuem a adorar testar as suas capacidades mentais com estes puzzles de lógica, Sudoku irá ser uma parte popular e adorada na vida de milhões de pessoas em todo o mundo.

2.2.2 Sudoku - Funcionamento

E como o jogo funciona ?, bem para começar, um tabuleiro de sudoku é normalmente constituído por uma grelha 9 por 9, com 9 sub-grelhas 3 por 3.

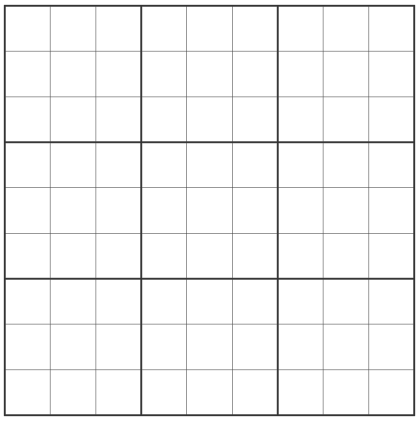

Figura 2.1: Grelha 9x9

E dentro dessa grelha cada "casa", ou seja, cada quadrado dos 81, presentes na grelha, ou está vazio ou contém um número de 1 a 9. O número espaços vazios e de números já na grelha, é determinado pelo criador do puzzle, em que essa quantidade de números e a respetiva posição no tabuleiro irá definir o nível de dificuldade no tabuleiro, embora a quantidade de números tenha uma pequena relevância na dificuldade do jogo, o que torna o puzzle desafiador será o posicionamento e relevância de cada "pista"(número já colocado pelo criador do puzzle.

O objetivo do jogo seria completar o tabuleiro, ou seja, preencher os espaços vazios com números de 1-9 de modo a não ficar nenhum espaço vazio no final. Mas claro, existem regras que é necessário ser cumpridas na sua resolução.

Essas regras consistem em que, em cada fila, ou seja ,cada conjunto de 9 casas que se transpõem ao longo do tabuleiro numa linha horizontal, sejam constituídas por números diferentes, como por exemplo:

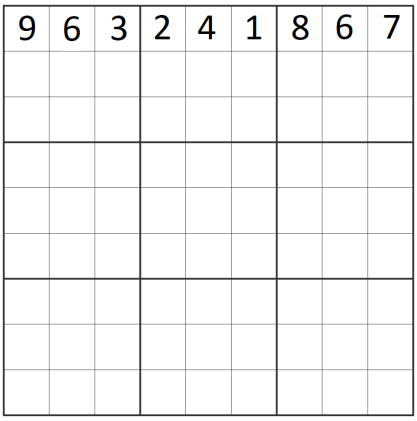

Figura 2.2: Fila

Outra regra diz que, da mesma maneira da fila, cada coluna(conjunto de 9 casas que que se transpõem ao longo do tabuleiro numa linha vertical), sejam também constituídas de números diferente.

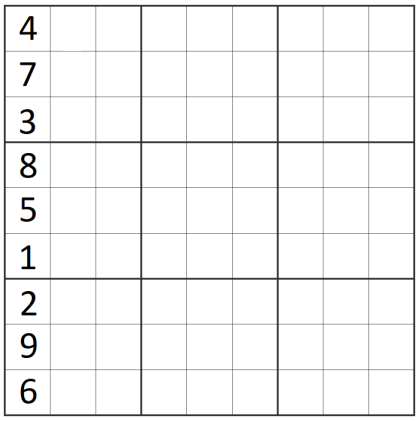

Figura 2.3: Coluna

A próxima regra dita que, numa caixa (sub-grelhas 3 por 3 que juntas constituem a grelha 9 por 9), também seja impossível colocar o mesmo número mais que uma vez em cada casa da respetiva caixa.

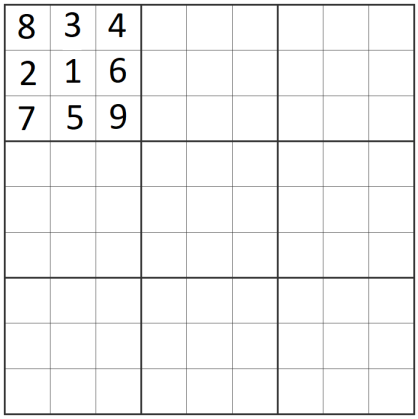

Figura 2.4: Caixa

Estas são as regras básicas, que constituem num moderno jogo de sudoku, e são as que foram usadas na criação, deste projeto. Mas existe uma variedade de outras regras que constituem variações do sudoku moderno original, como por exemplo:

1. **XSudoku** – variante exatamente igual á original com a regra extra de que as linhas diagonais da casa superior esquerda á inferior direita e casa superior direita á casa inferior esquerda, também precisam conter números diferentes de 1-9

4	6	2	3	9	8	1	7	5
8	5	3	2	1	7	4	6	9
7	1	9	4	6	5	2	3	8
9	4	8	6	2	3	7	5	1
2	7	1	5	8	9	6	4	3
6	3	5	7	4	1	9	8	2
5	8	4	9	7	2	3	1	6
3	9	7	1	5	6	8	2	4
1	2	6	8	3	4	5	9	7

Figura 2.5: X-Sudoku

2. **Sudoku Samurai** – consiste em cinco quebra-cabeças normais de Sudoku de quadrados 9x9, sobrepondo-se nos cantos com quatro caixas de 3x3. Cada um dos cinco Sudoku tem 9 linhas e 9 colunas para resolver, mas em vez de 45, existem 41 caixas para resolver devido à sobreposição.

2	1	7	4	5	8	9	6	3				8	4	6	5	7	2	1	3	9
9	4	8	1	6	3	7	5	2				3	5	9	8	6	1	7	2	4
3	5	6	9	7	2	1	4	8				2	1	7	3	4	9	8	5	6
7	3	4	5	1	6	2	8	9				4	9	5	2	8	7	6	1	3
6	8	2	7	3	9	4	1	5				7	8	2	6	1	3	9	4	5
5	9	1	8	2	4	3	7	6				1	6	3	9	5	4	2	8	7
1	6	5	2	9	7	8	3	4	5	2	6	9	7	1	4	3	8	5	6	2
4	2	3	6	8	1	5	9	7	8	3	1	6	2	4	1	9	5	3	7	8
8	7	9	3	4	5	6	2	1	9	4	7	5	3	8	7	2	6	4	9	1
						1	5	6	7	8	4	2	9	3						
						2	4	9	6	5	3	8	1	7						
						3	7	8	2	1	9	4	5	6						
4	3	8	9	2	1	7	6	5	1	9	8	3	4	2	7	5	1	8	9	6
1	5	9	6	3	7	4	8	2	3	7	5	1	6	9	2	3	8	5	4	7
6	7	2	8	5	4	9	1	3	4	6	2	7	8	5	6	4	9	1	3	2
8	2	4	5	7	3	6	9	1				9	5	4	3	6	7	2	8	1
5	9	6	1	8	2	3	4	7				2	1	6	9	8	5	4	7	3
3	1	7	4	9	6	2	5	8				8	3	7	4	1	2	6	5	9
2	6	1	3	4	8	5	7	9				6	9	8	1	7	4	3	2	5
7	8	5	2	6	9	1	3	4				4	2	3	5	9	6	7	1	8
9	4	3	7	1	5	8	2	6				5	7	1	8	2	3	9	6	4

Figura 2.6: Samurai Sudoku

2.3 Reconhecimento em Imagens

2.3.1 Machine Learning

2.3.1.1 História e Definição

Em 1943, lógico e cientista cognitivo Walter Pitts e neuro-cientista Warren McCulloch criaram o primeiro modelo matemático de uma rede neural do mundo, para a criação de algoritmos que pudessem imitar o processo de pensamento humano.

Em 1950, Alan Turing criou o **"Turing Test"**, que definia que se uma maquina pudesse convencer um ser humano , de que também era um ser humano, seria designada de "Inteligente".

Mas foi em 1952, que o termo **"Machine Learning"**, foi cunhado por Arthur Samuel, um pioneiro na área de Inteligência Artificial, começado a sua pesquisa em 1949, o seu ***Checkers Player***, um programa para um computador International Business Machines (IBM) que tinha a capacidade de jogar damas e melhorar o desempenho com cada jogo.

Iremos também enumerar outras varias datas de eventos importantes na história de Machine Learning:

- 1956 - ***Dartmouth Summer Research*** - John McCarthy, convidou vários cientistas e matematicos para a Faculdade de **Darthmouth**(Hanover,New Hampshire, U.S.A) onde, durante 6-8 semanas pensaram em ideias para máquinas que "pensam". Este evento é considerado por muitos pelo berço da Inteligência Artificial.

- 1963 - **MENACE** - Pesquisador Donald Michie desenvolveu um programa batizado de MENACE (*Matchbox Educable Noughts and Crosses Engine*, que tinha a capacidade de aprender como jogar um ronda perfeita do jogo do galo.

- 1979 - **Stanford Cart** - Um grupo de investigadores na **Universidade de Stanford**(Palo Alto, Santa Clara, Califórnia, U.S.A), criaram um robot conhecido como ***the Cart*** que seria capaz de navegar por obstáculos num certo espaço.

- 1996 - **Deep Blue** - programa DeepBlue desenvolvido por IBM, derrotou o campeão do mundo, na altura, Garry Kasparov.

- 1998 - **MNIST** - Uma equipa liderada por **Yann LeCun** lançaram um dataset conhecido como MNIST (Modified National Institute of Standards and Technology), que ficou reconhecido como um marco na evaluação de reconhecimento de escrita manual.

Machine Learning é um ramo da **Inteligência Artificial** que utiliza modelos estatísticos para fazer previsões.
Normalmente é descrita como uma forma de analise preditiva e é definida como a habilidade de um computador aprender sem ser explicitamente programado para tal.

Em termos mais básicos, ***Machine Learning*** usa algoritmos que utilizam dados empíricos, analisa-os, e gera outputs baseados nessa análise.
Os algoritmos usam os "dados de treino", e depois aprendem, preveem e encontram maneiras de melhorar o seu desempenho ao longo do tempo.

2.3.1.2 Tipos e Exemplos de Machine Learning

Existe 3 abordagens que são normalmente e regularmente usadas em ***Machine Learning***: **Aprendizagem Supervisionada**, **Aprendizagem sem Supervisão** e **Aprendizagem Reforçada**, em que cada abordagem tem as suas vantagens e desvantagens, logo são usadas cada uma para situações diferentes.

- **Aprendizagem Supervisionada** - nesta abordagem, o computador é treinado num grupo de dados (inputs e outputs), com a finalidade de aprender uma regra geral que liga os inputs aos outputs.
  Os dois tipos de Aprendizagem Supervisionada incluem:
  - **Classificação** - irá prever a que categoria pertence os dados e coloca-os num grupo com a sua respetiva etiqueta.
    Tem como exemplos: detenção de spam e detenção de raça de cão.
  - **Regressão** - prevê um valor numérico baseado em dados previamente observados.
    Tem como exemplos: Previsão do preço de casas e Previsão de Peso-Altura.
- **Aprendizagem sem Supervisão** - o algoritmo não ira receber este tipo de guias como no anterior, em vez disso irá tentar descobrir um padrão ou estrutura no input sozinho.
  Os dois tipos de Aprendizagem sem Supervisão incluem:
  - **Agrupamento** - implica a divisão dos pontos dos dados num numero de grupos para que esses pontos que estejam no mesmo grupo sejam mais similares a outros pontos no mesmo grupo e diferentes de pontos de outros grupos.
    Tem como exemplos: Identificar espécies de plantas ou animais diferentes e agrupar livros em bibliotecas por tópico e ou informação.

  - **Estimação de Densidade** - é o ato de estimar um continua espaço de dados de uma certo grupo de pontos retirados desse espaço. Tem como exemplos: O tempo de vida médio das pessoas num certo país pode ser usado para estimar a distribuição da idade de morte da população.

- **Aprendizagem Reforçada** - o computador e algoritmos irão encontrar o problema num ambiente dinâmico para completar um dado objetivo, e irá receber *feedback*, que irá reforçar a sua aprendizagem e "desejo"de chegar ao objetivo final.
  Tem como exemplos:

  - **QLearning** - é um algoritmo de aprendizagem que utiliza um sistema baseados em valores. Este tipo de algoritmo irá atualizar a função do valor baseado numa equação (normalmente a equação Bellman). Exemplos: Carros auto-guiados e robôs.
  - **Aprendizagem Profundamente Reforçada** - é um algoritmo que combina is a Aprendizagem Reforçada com Aprendizagem Profunda, considera um problema de um agente computacional que aprende a fazer decisões utilizando tentativa e erro. Exemplos: AlphaGo

2.3.2 Optical Character Recognition

2.3.2.1 Definição

Optical Character Recognition (OCR) é um programa de reconhecimento de texto. Um programa OCR extrai e reutiliza dados de documentos digitalizados, fotografias, ou Portable Document Format (PDF)s de imagens. *Software* OCR identifica e destaca letras numa imagem, transforma letras em palavras e palavras em frases, para a finalidade de aceder e editar o conteúdo original dessas imagens.

Sistemas OCR usam uma combinação de *hardware* e *software* para converter documentos físicos em texto que possa ser lido por computadores.
Hardware como um scanner óptico copia ou lê texto, a seguir **Software** lida com o processamento.

Software OCR utiliza **Inteligência Artificial (IA)** para implementar métodos mais avançados de **Intelligent Character Recognition (ICR)**, como identificação de linguagens ou estilos de escrita manual.
O processo de OCR é normalmente mais usado para legalizar cópias impressas ou documentos históricos em PDF para serem editados, formatos e pesquisados como se fossem criados por um processador de texto.

2.3.2.2 História

A tecnologia conhecida como OCR tem a sua origem no telegrafo e na criação de dispositivos de leitura para cegos.
Em 1914, inventor e físico israelense, **Emanuel Goldberg** desenvolveu um máquina capaz de ler caracteres e converte-los em código telegrafo. Simultaneamente, astrofísico e químico irlandês, **Edmund Fournier d'Albe** desenvolveu o *Optophone*, um scanner manual que quando movido sobre uma página imprimida, produzia sons que correspondiam a certas letras ou caracteres.

Nos final da década de 1920 e inicio da de 1930, **Emanuel Goldberg** desenvolveu o chamado *Statistical Machine* para procurar arquivos de microfilme usando um sistema de reconhecimento óptico de código. Em 1931, foi lhe dada a patente dos E.U.A, número 1,838,389 pela invenção, que por sua vez foi adquirida pela IBM.

Em 1974, futurista e inventor americano, **Ray Kurzweil** fundou a *Kurzweil Computer Products, Inc.*, cujo produto omni-fonte OCR podia reconhecer texto escrito em quase qualquer fonte.
Em 1980, ele vendou a empresa á **Xerox**, que estava interessada em comercializar conversão de texto papel-computador.

Mas a tecnologia OCR só ficou popular no inicio da década de 90 na digitalização de jornais históricos.
Hoje em dia, serviços OCR estão acessíveis ao publico, como por exemplo, o Google Cloud Vision OCR é usado para examinar e guardar documentos no smartphone.

2.3.2.3 Técnicas

Software OCR possui várias técnicas para a leitura dos caracteres, as mais usadas incluem:

- **Pré-Processamento** - manipulação e remoção de dados antes do uso, para melhorar o desempenho. Exemplos de Pré-Processamento incluem:

    1. *De-skew* - se o documento não está alinhado com o scanner na sua examinação, irá ser um pouco inclinado para puder criar linhas de texto.
    2. *Despeckle* - remove pontos e alisa as arestas.
    3. *Binarization* - converte uma imagem em preto e branco, para ser mais fácil distinguir texto da imagem em si.
    4. Remoção de linhas - remove linhas e do tipo, desde que não sejam considerados caracteres.

5. *Zoning* - Usada na identificação de parágrafos, colunas, legendas.
6. Deteção de linhas e palavras - estabelece a base para a divisão de letras e palavras, e separa palavras se necessário.
7. Reconhecimento de script - em documentos com várias linguagens, o *script* pode se transformar ao nível de palavra ,logo a identificação do script é vital.
8. *Segmentation* - vários caracteres unidos por "artefactos"de imagem devem ser divididos, caracteres únicos divididos em várias peças baseadas em artefactos devem ser unidos.
9. *Normalization* - afeta o *aspect ratio* e a escala

- ***Feature Extratment*** - é um tipo de redução dimensional, onde um grande número de pixeis numa imagem são representados de numa maneira especifica, para que as partes importantes da imagem sejam capturadas efetivamente.

- **Pós-Processamento** - envolve etapas de "limpeza"de dados para documentos que foram digitalizados, por exemplo na identificação e correção de ortografia e gramática, gerados por falhas no sistema de OCR.

- **Correção de erro** -*Near Neighbor Analysis* usa a frequência de ocorrência para corrigir erros, por exemplo algumas palavras que nunca foram vistas juntas.

2.4 Base de Dados

Para o treinamento, avaliação e teste do modelo de OCR, foi criada uma base de dados constituída por 100 imagens, cada imagem consiste numa fotografia de um puzzle de sudoku gerado por um programa e completado á mão.

Para atingir uma maior variação nas imagens para treinamento, os puzzles foram completos e fotografados de maneiras diferentes. Estas modificações resumem-se a:

1. **Cor do número** - Como se pode ver nas imagens, foram usados canetas de cores diferentes e lápis.

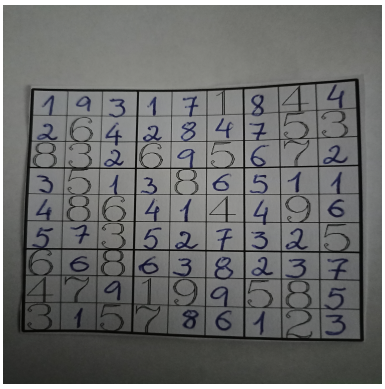

Figura 2.7: Puzzle feito com caneta azul

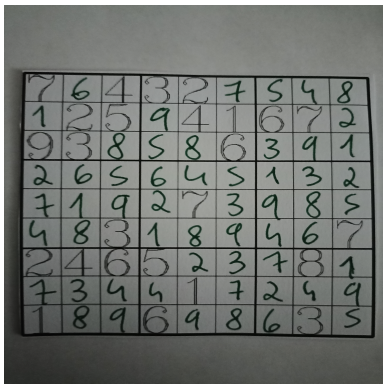

Figura 2.8: Puzzle feito com caneta verde

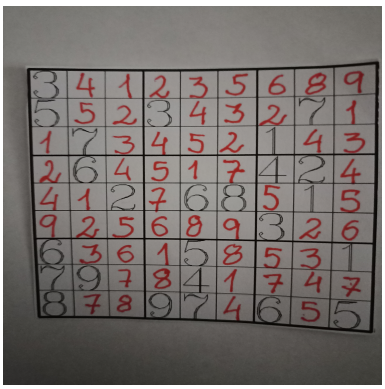

Figura 2.9: Puzzle feito com caneta vermelha

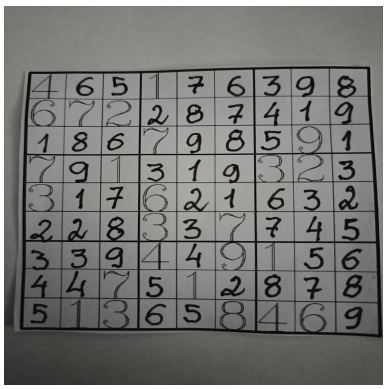

Figura 2.10: Puzzle feito com caneta preta

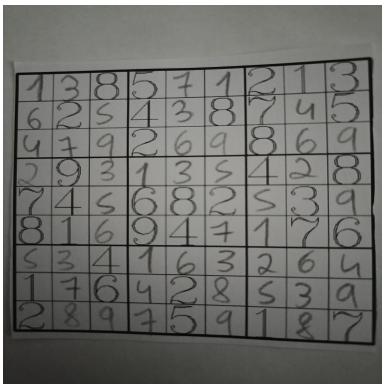

Figura 2.11: Puzzle feito com lápis

2. **Caligrafia** - Cada pessoa têm a sua maneira de escrever, logo os puzzles foram feitos por várias pessoas.

Figura 2.12: Caligrafia de pessoa nº1

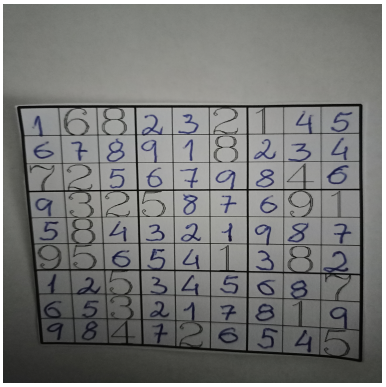

Figura 2.13: Caligrafia de pessoa nº2

Figura 2.14: Caligrafia de pessoa nº3

3. **Iluminação** - Fotos foram tiradas em vários ambientes de iluminação diferentes.

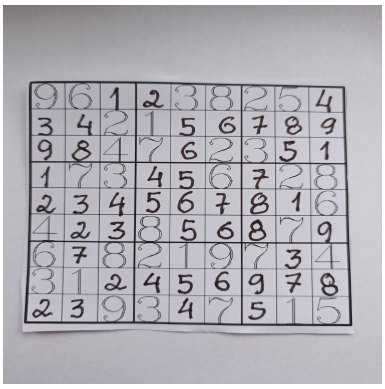

Figura 2.15: Ambiente de Iluminação nº1

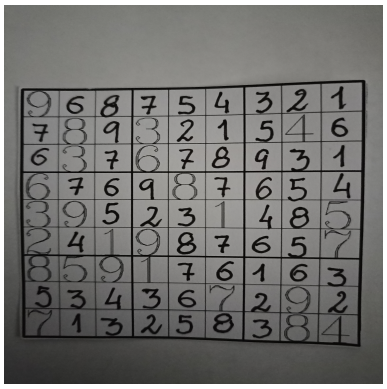

Figura 2.16: Ambiente de Iluminação nº2

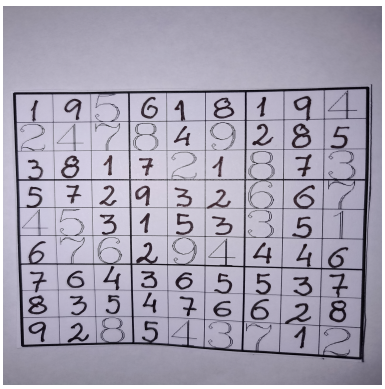

Figura 2.17: Ambiente de Iluminação nº3

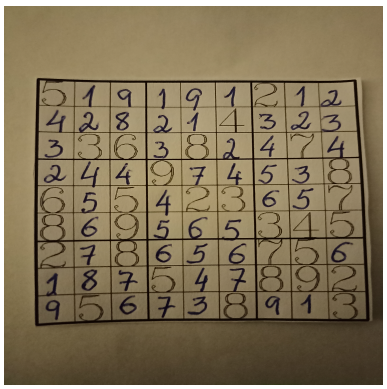

Figura 2.18: Ambiente de Iluminação nº4

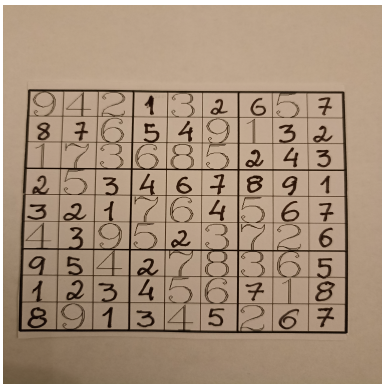

Figura 2.19: Ambiente de Iluminação nº5

4. **Perspetiva** - Alguns fotos foram tiradas de várias posições e perspetivas.

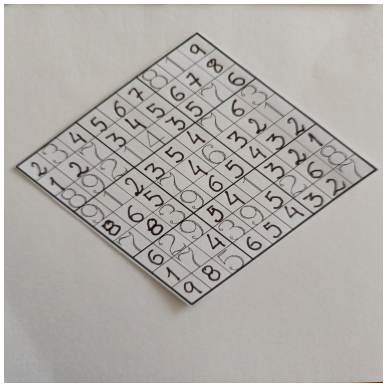

Figura 2.20: Imagem com Rotação de 135º

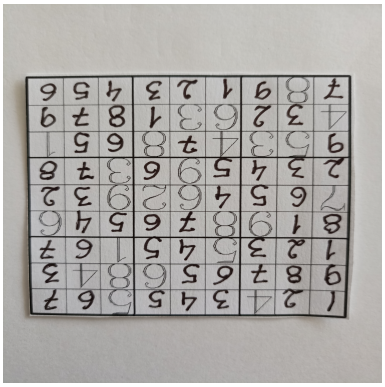

Figura 2.21: Imagem com Rotação de 180º

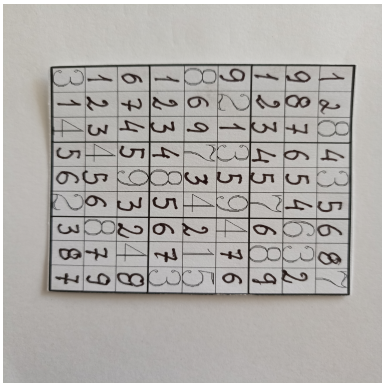

Figura 2.22: Imagem com Rotação de 90º

Figura 2.23: Perspetiva lateral direita

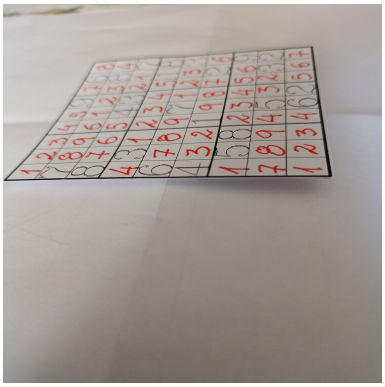

Figura 2.24: Perspetiva lateral esquerda

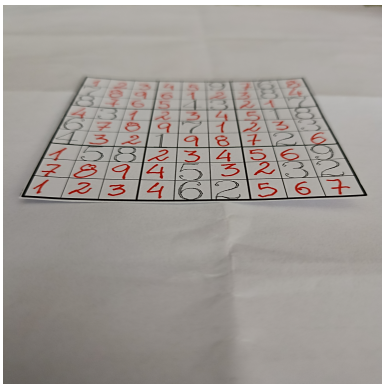

Figura 2.25: Perspetiva Frontal

Capítulo

3

Método Proposto

3.1 Introdução

Neste capitulo, irá ser demonstrado o método utilizado para a resolução do projeto, todos os módulos irão ser expostos e o código feito explicado.

3.2 Gerador

Nesta secção, vai-se demonstrar o primeiro passo na criação um puzzle sudoku, que é a criação de uma matriz 9x9, completamente feita de números de 1 a 9 aplicando as regras básicas enunciadas na secção 2.2.2.

3.2.1 Aplicar regras

As seguintes funções irão verificar se o número a ser colocado na matriz de acordo com as regras.

```
def UsadoCaixa(start_row, start_col, n):
```

Excerto de Código 3.1: Verificar se o número já foi utilizado na caixa 3x3 correspondente.

```
def UsadoColuna (j, n):
```

Excerto de Código 3.2: Verificar se o número já foi utilizado na coluna correspondente.

```
def UsadoFila (i, n):
```

Excerto de Código 3.3: Verificar se o número já foi utilizado na fila correspondente.

```
def Verificar(i, j, n):
```

Excerto de Código 3.4: Verificar se o número pode ser colocado no tabuleiro utilizando os 3 excertos anteriores.

3.2.2 Preenchimento do Tabuleiro

Aqui irá ser explicado a função usada para o preenchimento do tabuleiro.

```
def preencher(i,j,x,a,arr):
```

Excerto de Código 3.5: Cabeçalho da função.

- i - posição x do tabuleiro
- j - posição y do tabuleiro
- x - 1 a 81, representa cada casa
- a - Array de arrays onde iram ser colocados os números possíveis para cada das 81 casas do tabuleiro
- arr - Array [1,2,3,4,5,6,7,8,9] constante

Primeiro inicia a variável g a -1 para verificar se na posição (i,j) ainda é possível colocar algum dos 9 números, que estão presentes no seu respetivo array a, exemplo se na posição (0,0) possa ser colocado um número que exista no array a[0].

Se todas as posições do array de arrays correspondente á posição estiver cheio de -1 (não poderá ser colocado nenhum número), reseta o array correspondente usando o array constante e da reset o número da posição para 0, e volta atrás uma casa para tentar outro número diferente do colocado.

Quando já não houver 0s no tabuleiro o programa termina.

Se for possível colocar o número na casa, ele irá transformar esse número que esta presente no array de arrays correspondente a -1, para identificar que foi usado, coloca o no tabuleiro, e continua para a próxima casa.

Se ainda não chegou ao final da fila continua uma casa para a frente.

Se chegou ao final da fila, passa para a fila de baixo na posição j = 0.

Se o número não pode ser colocado, mas ainda pode ser tentado outro número, nessa posição, coloca o número a -1 para não tentar outra vez, e continua na mesma posição.

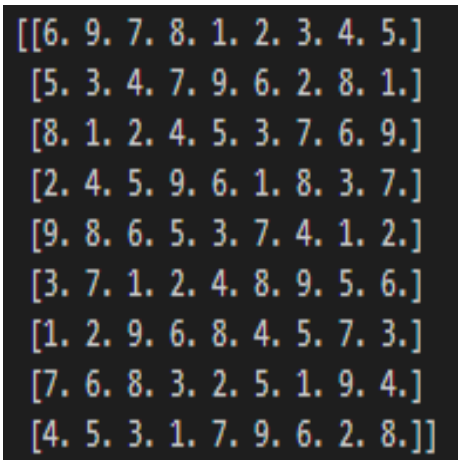

Figura 3.1: Exemplo da matriz criada

3.3 Dificuldades

Para um sudoku ter uma única solução, é necessário no mínimo 17 números no tabuleiro, mas seria bastante difícil para uma pessoa resolver, logo foram criados 3 níveis de dificuldade para a criação da imagem do sudoku para resolver(3.4). Os três níveis de dificuldade são compostos por : Dificuldade Difícil : 25 números

```
def Dificil(m):
```

Excerto de Código 3.6: Dificuldade Difícil

Nesta dificuldade, depois de receber uma matriz 9x9(m), preenchida de números de 1 a 9, ele irá transformar aleatoriamente números de 1 a 9 para 0 que vai simular a inexistência de um número nessa casa até só restarem 25 números de 1 a 9, quando só esses números ficarem o programa irá parar. Dificuldade Média : 30 números

```
def Medio(m):
def ExistemUmCaixa(start_row, start_col,m):
```

Excerto de Código 3.7: Dificuldade Média

Nesta dificuldade acontece o mesmo que na dificuldade anterior, em que as únicas diferenças são que em vez de só restarem 25 números de 1 a 9, neste têem de restar 30 e em cada caixa 3x3 têm estar sempre no mínimo 1 número que não é 0, isto é feito na função "ExistemUmCaixa". Dificuldade Fácil : 35 números

```
def Facil(m):
def ExistemDoisCaixa(start_row, start_col,m,count,pos_i,pos_j):
```

Excerto de Código 3.8: Dificuldade Fácil

Nesta dificuldade acontece o mesmo que na dificuldade anterior, em que as únicas diferenças são que em vez de só restarem 30 números de 1 a 9, neste têem de restar 35 e em cada caixa 3x3 em vez de estar sempre no mínimo 1 número que não é 0 tem de existir 2 números, isto é feito na função "Existem-DoisCaixa".

3.4 Criação da imagem do puzzle

Nesta secção irá ser exposta a criação de uma imagem com um puzzle sudoku a partir da matriz (m) gerada na secção 3.2.

```
def CriaImagem(m):
```

Excerto de Código 3.9: Cabeçalho da função

Em primeiro lugar irá criar uma imagem completamente branca que será o "canvas"para desenhar o tabuleiro.

Seguidamente irá escrever 4 linhas que quando combinadas formam um quadrado que seriam as bordas do tabuleiro, para haver um maior contraste com as linhas interiores do tabuleiro, estas linhas de borda têm uma grossura de 3.

Como dito anteriormente, irão ser desenhadas também linhas de marcação de casas e de caixas 3x3 que irão conter 9 casas cada uma,e como nas linhas da borda, as linhas que delimitam as caixas teriam grossura de 2 para melhor contraste e da mesma maneira as linhas que delimitam as casas teriam grossura de 1 pelo mesmo motivo.

Sucessivamente a função irá retirar o número de cada casa da matriz, para expo-lo na imagem criada, em que se o número for 0 seria o equivalente a um espaço em branco

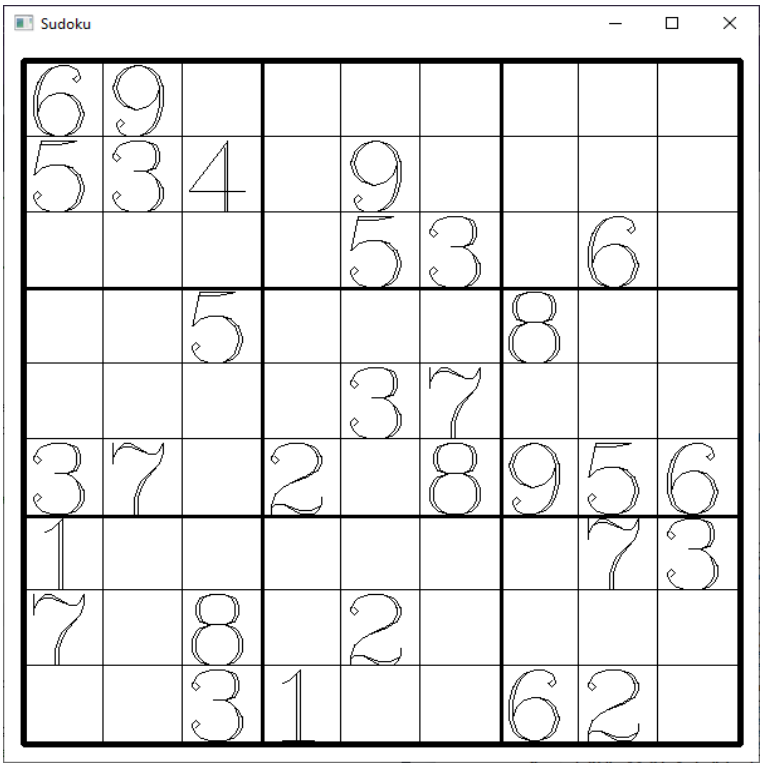

Figura 3.2: Exemplo do sudoku criado

3.5 Base de Dados

Nesta secção, vai ser explicado a criação de varias imagens de 1 a 9, que serviram de base de dados para o treinamento e teste do modelo de reconhecimento.

3.5.1 Tratamento de imagem

Aqui será demonstrado, o tratamento da imagem criada, como enunciado em 3.4, e depois preenchida pelo utilizador, como exibida na figura 3.3. O tratamento da imagem têm como objetivo, preparar a imagem para poder ser possível retirar cada número manuscrito presente para a criação da base de dados.

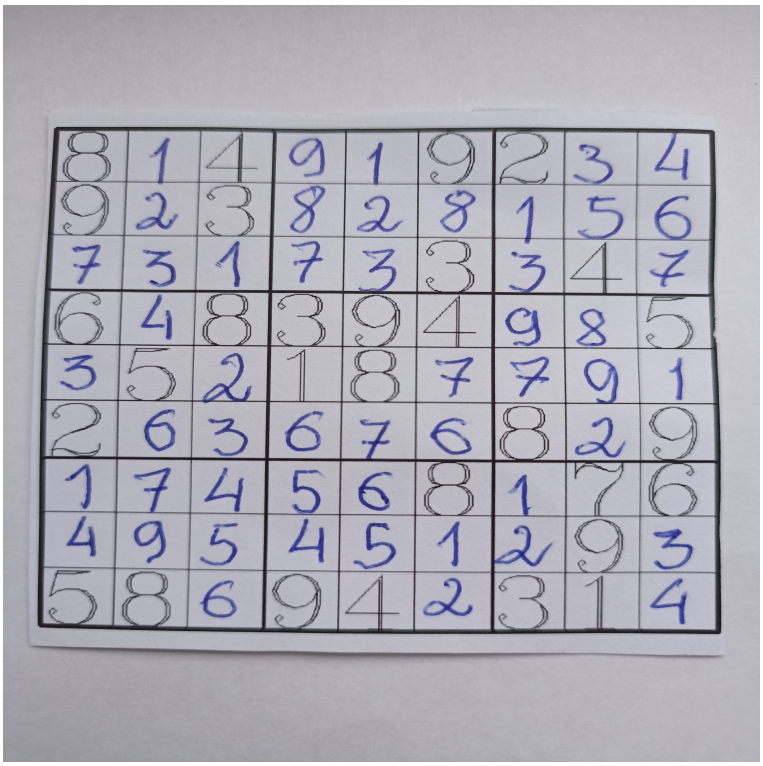

Figura 3.3: Exemplo de imagem preenchida

Em primeiro lugar, irá começar por receber uma imagem, neste exemplo a imagem exposta em 3.3, depois irá reduzi-la em tamanho para ser mais fácil examiná-la, seguido por transforma-la numa imagem grayscale e depois aplica-lhe um blur para alisar os cantos da imagem para ser mais fácil identificar os 4 cantos do tabuleiro e também os cantos de cada "casa".

A seguir, iremos utilizar a função "Corner Harris"para identificar os cantos do tabuleiro, irá ser aplicada também uma mask para podermos visualizar os pontos que representam os cantos e coloco-los no array "coordinates".

Como existem muitos pontos, e só precisamos de 4 que seriam os cantos do tabuleiro, é necessário extrair esses 4 dos demais. Para isso transformamos a variável "coordinates"de um array de arrays, numa lista de arrays e depois numa lista de tuplos. Extraímos esses cantos para as variáveis, "top-left","bottom-right","top-right","bottom-left", que seriam os cantos superior esquerdo,inferior direito, superior direito e inferior esquerdo respetivamente.

Depois iremos recortar, ou "crop"da imagem utilizando esses cantos adicionando alguns pixeis a mais para não cortar nada do tabuleiro. Foram feitos dois "crops"pois era necessário ainda aplicar um "adaptive threshold"a imagem para ser extraídos os cantos das "casas", e outro utilizando a imagem

original irá ser utilizada no final para extrair os números como estavam originalmente na imagem.

Irá ser aplicado um threshold para ser possível retirar as linhas horizontais e verticais, que compõem o tabuleiro.
O resultado será exposto na figura 3.4

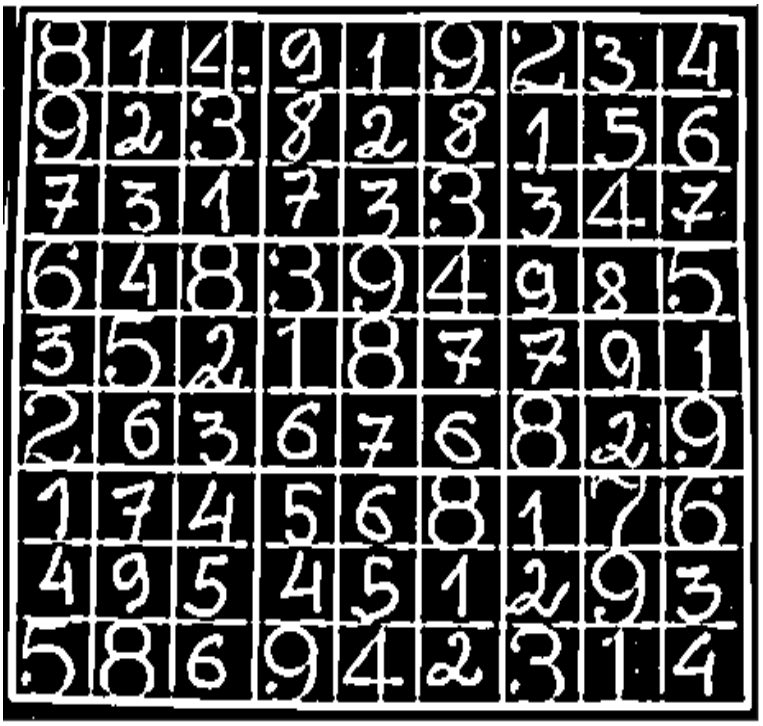

Figura 3.4: Exemplo de imagem com o threshold

Seguidamente, foi utilizada a função "HoughLine", que é utilizada para detetar linhas direitas.

E também é utilizado um "loop"for para desenhar as linhas no tabuleiro da imagem original como exposto na figura 3.5.

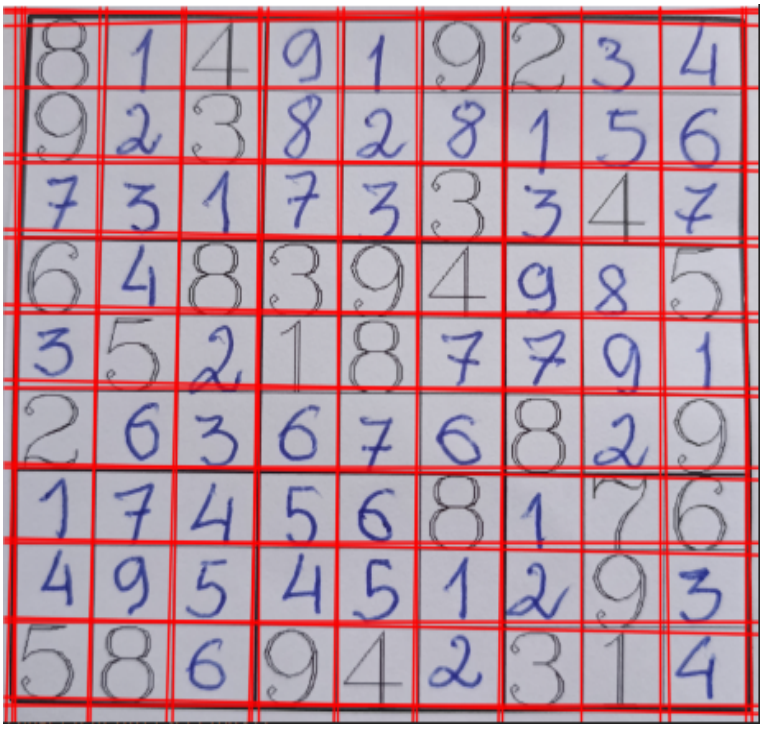

Figura 3.5: Exemplo de imagem com as linhas desenhadas

Nas seguintes funções, iremos enumerar por todas as linhas, e colocar num array o ponto de intersecção dessas linhas desde que as linhas sejam perpendiculares.

```
def segmentar_por_angulo(lines, k=2, **kwargs):
def interseccao(line1, line2):
def interseccoes_segmentadas(lines):
```

Nesta função, serão removidos pontos extra, criados por linhas extras no tabuleiro, utilizando esta função os pontos têm que estar espaçados em x e y, utilizando o threshold como o numero de pixeis que tem de existir entre dois pontos.

```
def check(array1,array2):
```

Depois, iremos utilizar as funções enunciadas anteriormente para a criação dos pontos de interseção de linhas que separam as "casas"do tabuleiro, e coloca-los uma lista denominada "intersections-simplified".

Em algumas situações os pontos mais extremos no tabuleiro podem não aparecer na totalidade da imagem, logo iriam aparecer com valores negativos

ou no valor x ou y, colocamos esses valores a 0 para ser necessário a extração dos números.

Para melhorar o visualizamento dos pontos no terminal, serão ordenados de menor para maior em relação ao primeiro valor(coordenada x), e o segundo valor(coordenada y).

Utilizando a função "findDiagonal", que irá encontrar o primeiro ponto diretamente diagonal ao ponto dado, iremos ter os dois pontos necessários para retirar cada casa do tabuleiro separadamente, e utilizando a variável "count", para dar-lhe um nome desde casa1 a casa81, escreve-las para uma pasta indicada na variável "path".

```
def findDiagonal(point, list):
```

Será mostrado os pontos na imagem original, como mostrado na figura 3.6.

Será mostrado um exemplo de um número extraído, como exposto na figura 3.7.

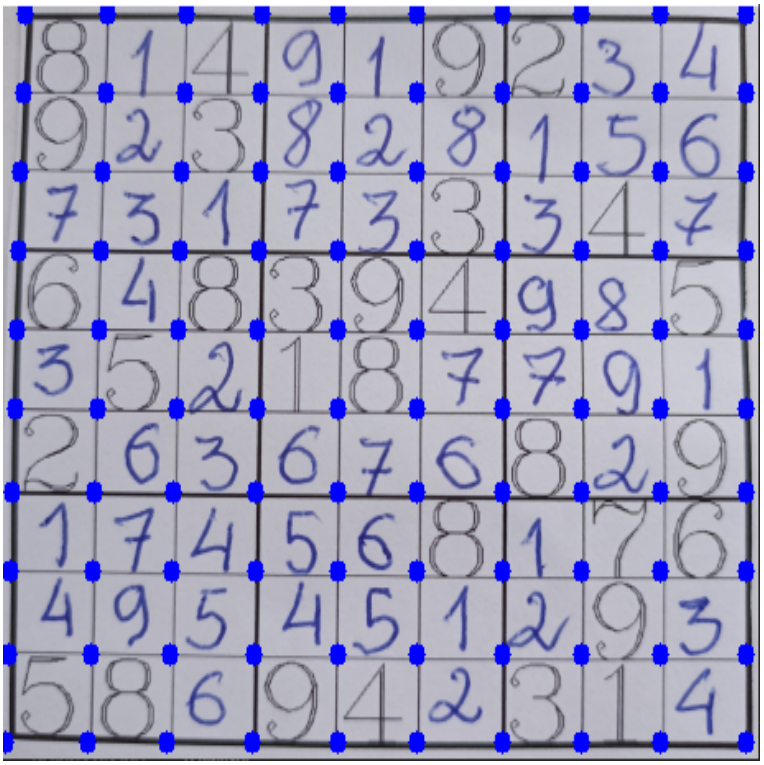

Figura 3.6: Exemplo de imagem com as pontos desenhados

Figura 3.7: Exemplo de número extraído

3.5.2 Perspetiva

Foram também tomadas em consideração, as situações em que as imagens não foram tiradas com uma perspetiva diferente da imagem exemplo (3.3), como demonstrado nas figuras 2.25, 2.24 e 2.23, logo foi necessário transformar a imagem, numa que seria mais fácil extrair os dígitos.
Ao transformar os tuplos dos cantos do tabuleiro de sudoku, coloca-los num array e em junção com os cantos extremos da imagem, são utilizados na função getPerspectiveTransform e warpPerspective do OpenCv e criam a imagem 3.8

Figura 3.8: Exemplo de imagem com perspetive transformada

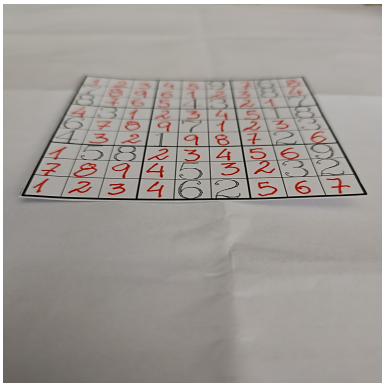

Figura 3.9: Imagem Original

3.6 Treinamento do Modelo

Nesta secção, irá ser explicado como foi feito o treinamento do modelo de reconhecimento, utilizando o dataset criado(explicado na secção3.5). Para a dita criação, foi utilizado a ferramenta "Google Colab", por facilidade de acesso a uma GPU, que seria necessária para o treinamento do modelo.

O modelo de reconhecimento de imagem utilizado foi o InceptionV3 3.10, uma rede neural convolucional(tipo de rede neural artificial que utiliza uma operação matemática chamada convolução em vez de uma matriz de multiplicação em pelo menos uma das suas camadas.) feita para facilitar a classificação de objetos.

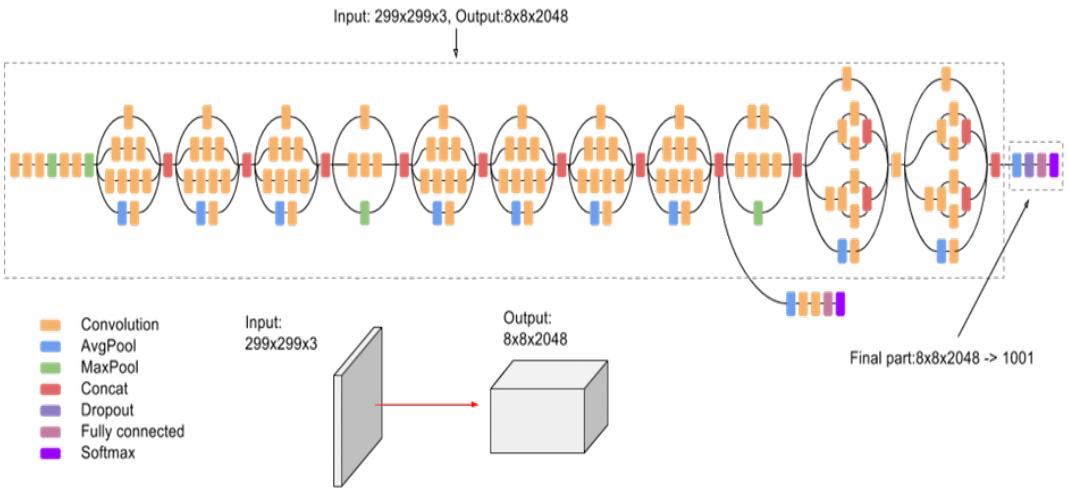

Figura 3.10: InceptionV3

A seguir tem que ser implementado as classes para o treinamento, neste caso as classes seriam os números de 1 a 9, ou seja 9 classes, para isto foi utilizado a função Dense do keras, e ativação softmax, pois o número de classes é superior a 2.

Neste caso as variável "folders"seria onde estariam pastas de 1 a 9, onde cada pasta continha as várias correspondentes imagens.

Agora sim, seria criado a variável "model", onde iria ser utilizado o modelo inception, e as varias classes de imagem.

O próximo passo seria dizer ao modelo qual o custo e o método de otimização usar, para esta finalidade foi utilizada a função compile.

A variável "loss"é utilizada para encontrar erro ou desvio do processo de aprendizagem, neste caso a "categorical crossentropy", têm de ser utilizada pois é necessária para classificação de modelo com varias classes.

A variável "optimizer"é utilizada para otimizar os "weights"do input ao comparar a previsão e a "loss". A variável "metrics"é utilizada para avaliar a atuação do modelo.

De seguida foram importadas as imagens de train e test, e foram aplicadas técnicas de melhoramento de dados e rescaling.

As funções de training set e test set, teriam 4789 imagens de 9 classes diferentes, e 531 imagens de 9 classes diferentes, respetivamente.

E finalmente é feito o treinamento do modelo ao longo de 10 epochs.

O output deste treinamento mostra a perda e precisão do treinamento do modelo e a perda e precisão da validação(ou seja, como o modelo se comporta ao encontrar dados que ainda não viu, ou foi treinado para). Este output será mostrado na tabela abaixo:

	Train-Loss	Train-Accuracy	Val-Loss	Val-Accuracy
Epoch-1	1.2607	0.5730	0.8578	0.7439
Epoch-2	0.7824	0.7467	0.6772	0.8041
Epoch-3	0.6687	0.7805	0.6365	0.8060
Epoch-4	0.5860	0.8046	0.5716	0.8211
Epoch-5	0.5630	0.8098	0.5850	0.8041
Epoch-6	0.5498	0.8154	0.5454	0.8004
Epoch-7	0.5327	0.8148	0.5329	0.8305
Epoch-8	0.4858	0.8398	0.4982	0.8380
Epoch-9	0.4753	0.8325	0.5119	0.8324
Epoch-10	0.4596	0.8453	0.4916	0.8267

Tabela 3.1: Output do "fitting"do modelo

Para entender melhor a precisão e a perda(loss) do modelo foram ambas traçadas nas imagens 3.11 e 3.12 respetivamente.

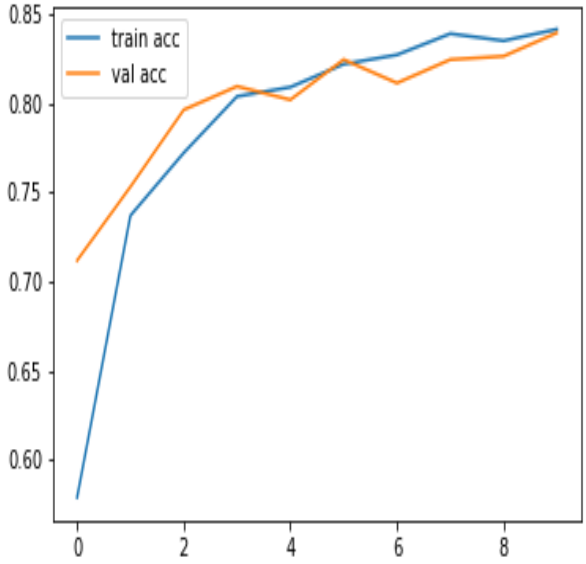

Figura 3.11: Precisão

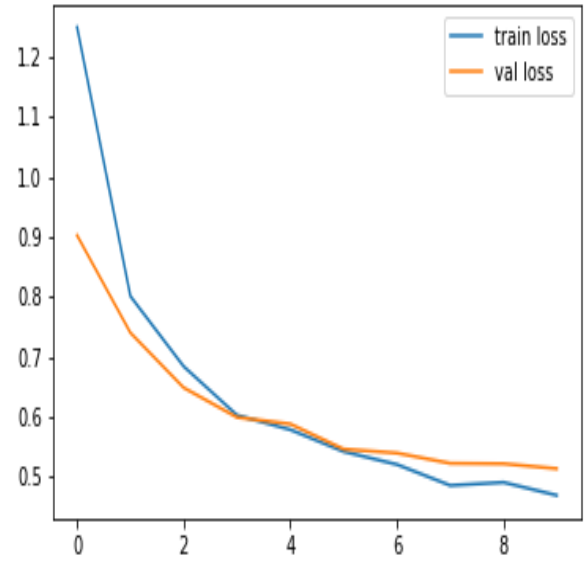

Figura 3.12: Perda

Finalmente para o uso do modelo no nosso programa foi guardado para uma variável.

3.7 Verificação

Com o conjunto de números escritos há mão, dados pelo utilizador e reconhecidos e transformados em dígitos, podemos agora proceder á verificação do puzzle, ou seja, verificar se o utilizador completou o sudoku corretamente, em atenção ás regras enunciadas na secção 2.2.

Para essa finalidade, foi criada uma pequena função que ao receber a matriz com os dígitos reconhecidos, irá verificar se o puzzle foi feito corretamente, e se não corrigi-lo e indicar ao utilizador onde ele errou.

O programa irá receber a matriz dada pela variável "m", e utilizando as funções de verificação mencionadas na secção 3.4, irá verificar cada posição da matriz, se houver algum erro, o programa irá corrigi-lo e irá explicar ao utilizador em que posição errou e que número devia ter sido colocado(como exemplificado no "print").

Como por exemplo, foi feito um puzzle com duas posições incorretas, e o programa teve o seguinte output demonstrado nas imagens 3.13,3.14 e 3.15:

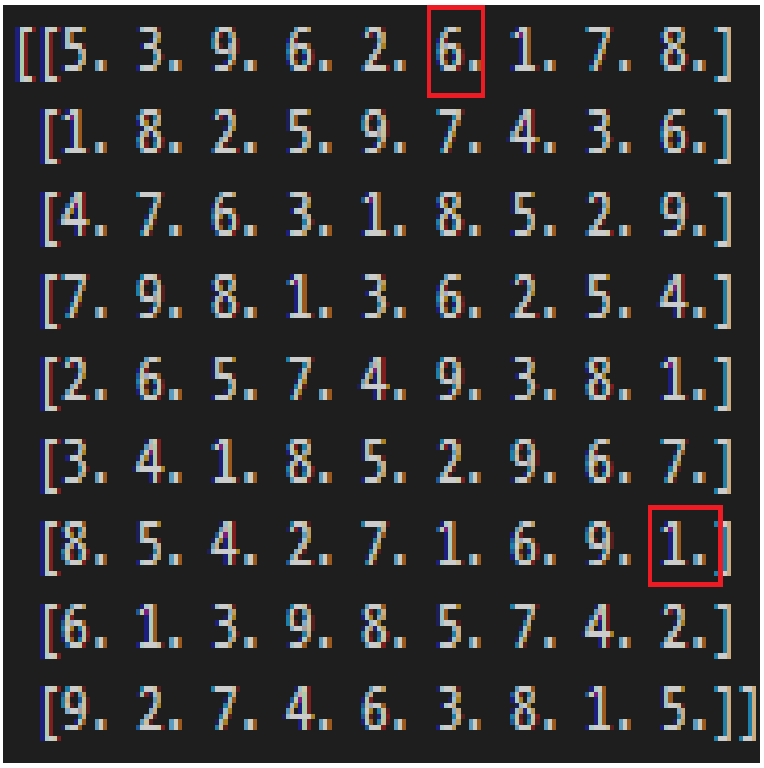

Figura 3.13: Matriz Incorreta, com números incorretos assinalados

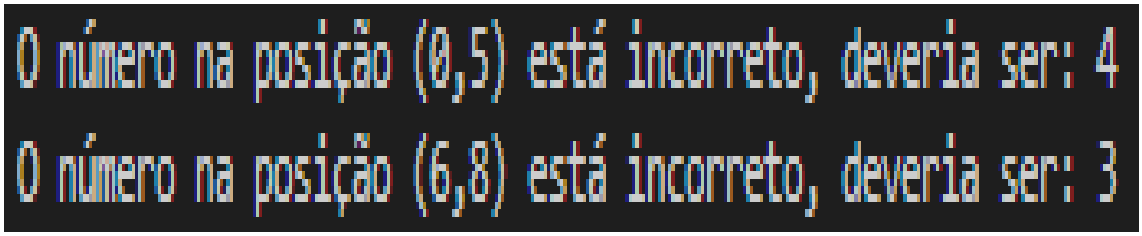

Figura 3.14: Correção

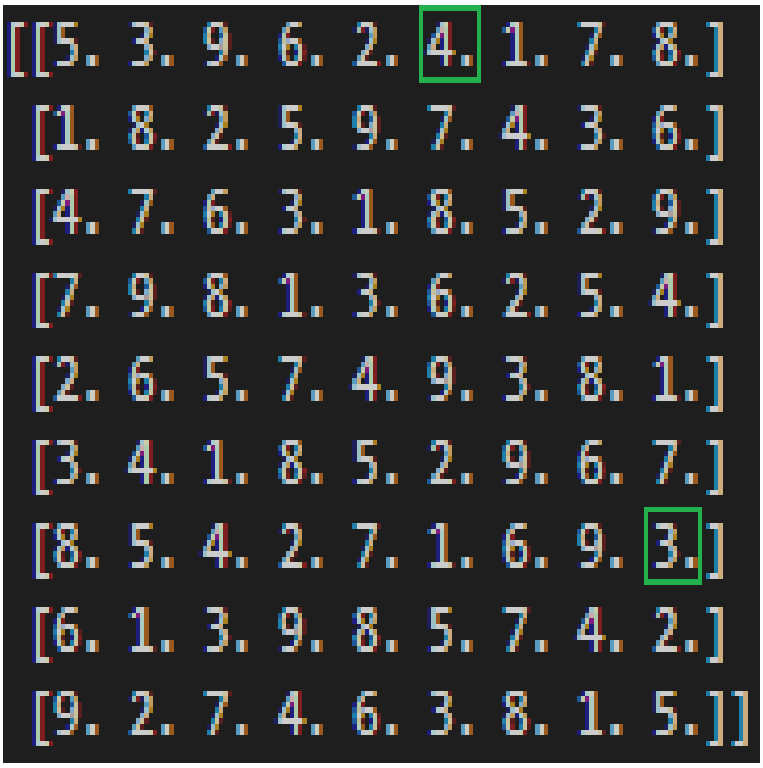

Figura 3.15: Matriz corrigida, com números corrigidos assinalados

3.8 Sudoku Solver

Também foi feito um programa que utilizando duas funções, irá resolver um programa dado pelo utilizador, esse o utilizador cria um tabuleiro, coloca alguns números no mesmo, tudo escrito á mão, e depois retira uma foto ou digitaliza, depois irá ocorrer uma operação como explicada na secção 3.6, onde neste caso irá ser recebida uma matriz, com os números escritos pelo utilizador e transformados em dígitos, com os espaços vazios representados com o 0.

```
    def checkLocked(arr_1,m):
```

Estão função irá verificar e irá guardar todos os números que foram escritos pelo utilizador na sua respetiva posição, pois não podem ser mudados.

```
    def resolve(i,j,m,a_1,arr_1,x,dir,locked):
```

Esta função irá receber duas variáveis "i","j", que representam as posições no tabuleiro que normalmente seria a posição 0,0, a variável "m"representa a matriz a ser preenchida, a variável "a-1"seria os números restantes possíveis para as posições no tabuleiro, "arr-1"que é um array com números de 1 a 9 utilizado para atualizar os números possíveis para uma posição no tabuleiro quando necessário, o x será o numero da casa em questão neste caso de 1 a 81, "dir"seria a direção que o programa está a percorrer o tabuleiro, ou seja caso precise de voltar a trás para mudar um número, que esteja incorreto, e finalmente a variável "locked"seria, a lista criada pela função anterior.

Capítulo

4

Testes

4.1 Introdução

Neste Capítulo irão ser mostrados os testes que foram feitos para testar a capacidade de reconhecimento do modelo.

Utilizando o modelo criado(3.6), foi testado usando 20 fotos de puzzles de sudoku diferentes, para identificar o que o modelo tinha mais dificuldade em reconhecer, utilizando várias combinações de iluminação, cor e tipo de utensílio de escrita, e o tipo de letra da pessoa(2.4.

A forma que os números foram reconhecidos e consequentemente escritos num array para a análise foi feito pela função:

```
def teste_model(model,path):
```

4.2 Testes

Como dito na introdução deste capítulo, foram feitos 20 testes para a testar o modelo, em que as especificações de cada um deles está exposto na seguinte tabela:

	Iluminação	Cor de Caneta	Letra
Teste 1	Tipo 1	Azul	2
Teste 2	Tipo 1	Lápis	1
Teste 3	Tipo 1	Verde	1
Teste 4	Tipo 1	Preta	2
Teste 5	Tipo 2	Preta	2
Teste 6	Tipo 2	Verde	1
Teste 7	Tipo 2	Azul	3
Teste 8	Tipo 2	Vermelha	2
Teste 9	Tipo 3	Azul	2
Teste 10	Tipo 3	Lápis	1
Teste 11	Tipo 3	Vermelha	2
Teste 12	Tipo 3	Verde	1
Teste 13	Tipo 4	Vermelha	1
Teste 14	Tipo 4	Azul	3
Teste 15	Tipo 4	Verde	1
Teste 16	Tipo 4	Azul	2
Teste 17	Tipo 5	Preta	2
Teste 18	Tipo 5	Lápis	1
Teste 19	Tipo 5	Azul	2
Teste 20	Tipo 5	Vermelha	1

Tabela 4.1: Testes feitos

Em que os tipos de iluminação, cor de letra e tipo de letra, são os tipos respetivamente dados na secção 2.4. Nas consecutivas subsecções, irá ser mostrado que imagens foram usadas, e quais os resultados de cada teste.

4.2.1 Lista de imagens usadas para o teste

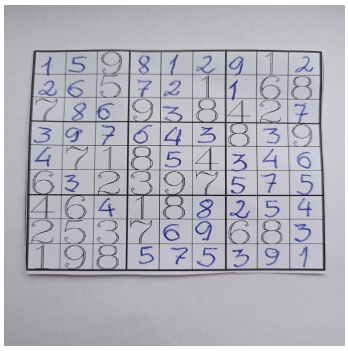

Figura 4.1: Imagem utilizada para o Teste Nº1

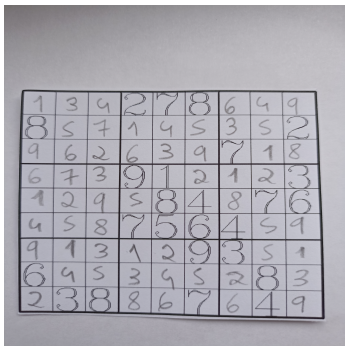

Figura 4.2: Imagem utilizada para o Teste Nº2

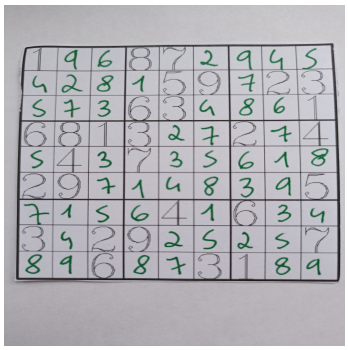

Figura 4.3: Imagem utilizada para o Teste Nº3

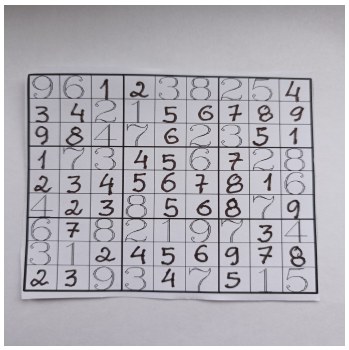

Figura 4.4: Imagem utilizada para o Teste Nº4

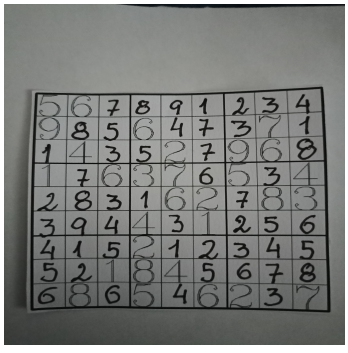

Figura 4.5: Imagem utilizada para o Teste Nº5

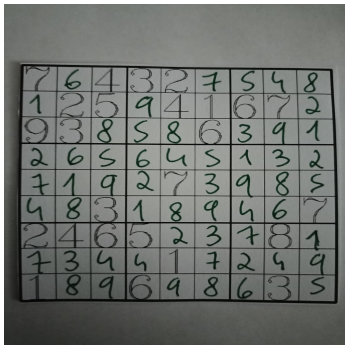

Figura 4.6: Imagem utilizada para o Teste Nº6

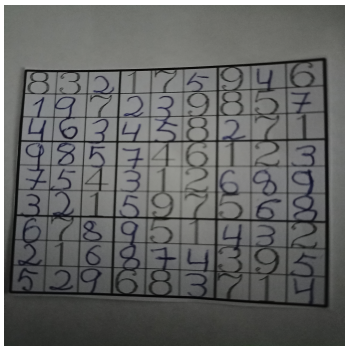

Figura 4.7: Imagem utilizada para o Teste Nº7

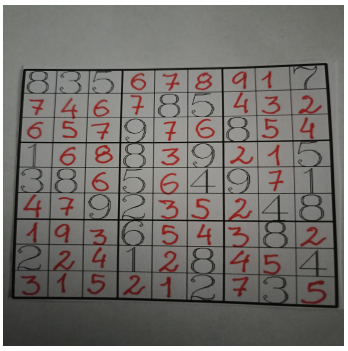

Figura 4.8: Imagem utilizada para o Teste Nº8

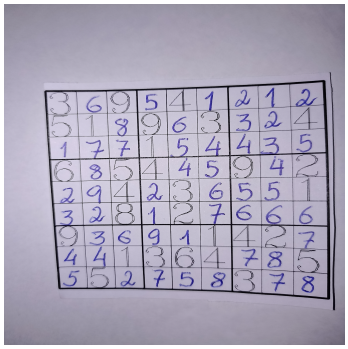

Figura 4.9: Imagem utilizada para o Teste Nº9

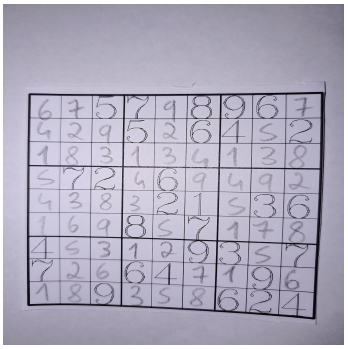

Figura 4.10: Imagem utilizada para o Teste Nº10

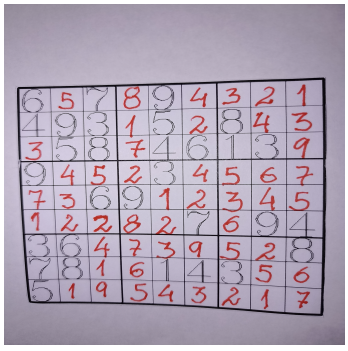

Figura 4.11: Imagem utilizada para o Teste Nº11

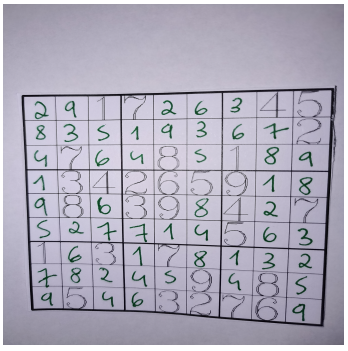

Figura 4.12: Imagem utilizada para o Teste Nº12

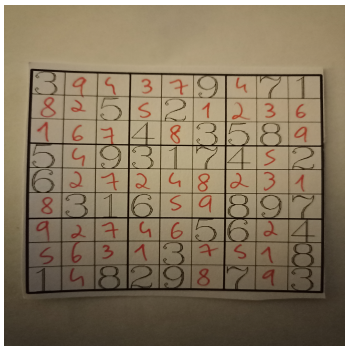

Figura 4.13: Imagem utilizada para o Teste Nº13

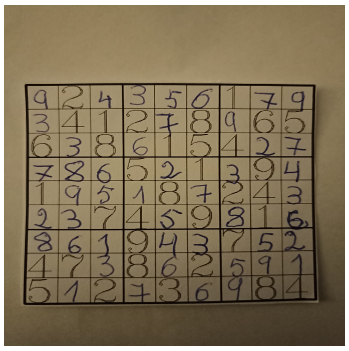

Figura 4.14: Imagem utilizada para o Teste Nº14

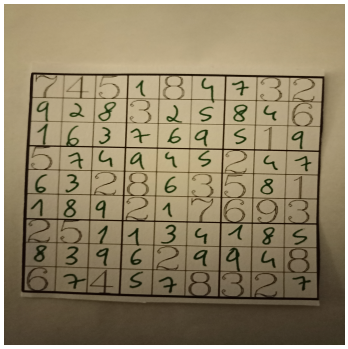

Figura 4.15: Imagem utilizada para o Teste Nº15

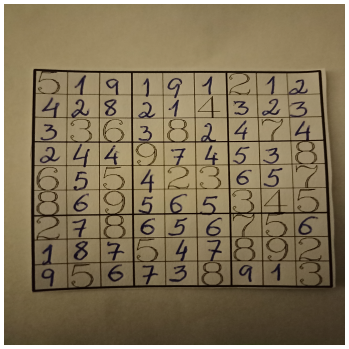

Figura 4.16: Imagem utilizada para o Teste Nº16

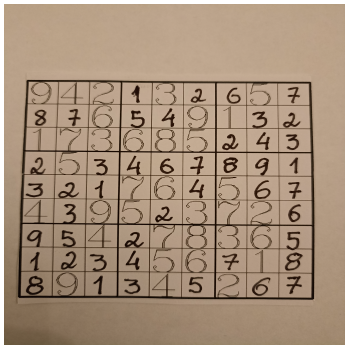

Figura 4.17: Imagem utilizada para o Teste Nº17

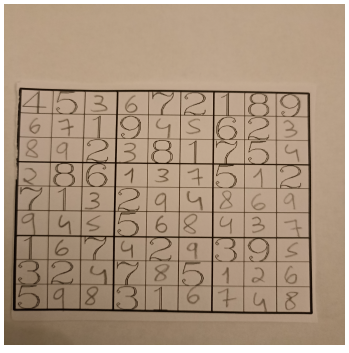

Figura 4.18: Imagem utilizada para o Teste Nº18

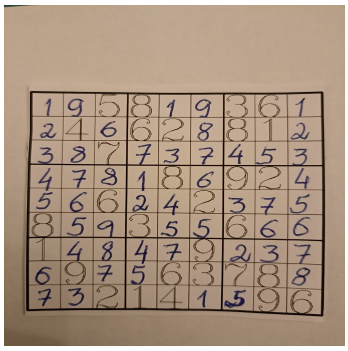

Figura 4.19: Imagem utilizada para o Teste Nº19

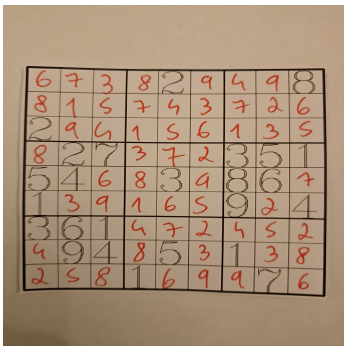

Figura 4.20: Imagem utilizada para o Teste Nº20

4.2.2 Teste Nº1

	Reconhecido	Correto
Casa 1	1	1
Casa 2	5	5
Casa 3	8	8
Casa 4	1	1
Casa 5	2	2
Casa 6	9	9
Casa 7	6	2
Casa 8	2	2
Casa 9	6	6
Casa 10	1	7
Casa 11	2	2
Casa 12	1	1
Casa 13	8	8
Casa 14	6	6
Casa 15	3	3
Casa 16	7	7
Casa 17	5	3
Casa 18	5	9
Casa 19	7	7
Casa 20	6	6
Casa 21	4	4
Casa 22	3	3
Casa 23	3	3

	Reconhecido	Correto
Casa 24	4	4
Casa 25	5	5
Casa 26	5	3
Casa 27	4	4
Casa 28	6	6
Casa 29	3	3
Casa 30	5	5
Casa 31	7	7
Casa 32	5	5
Casa 33	7	4
Casa 34	8	8
Casa 35	2	2
Casa 36	5	5
Casa 37	7	4
Casa 38	6	6
Casa 39	9	9
Casa 40	2	3
Casa 41	3	5
Casa 42	7	7
Casa 43	5	5
Casa 44	3	3
Casa 45	6	9
Casa 46	1	1

Tabela 4.2: Teste1

4.2.3 Teste Nº2

	Reconhecido	Correto
Casa 1	1	1
Casa 2	3	3
Casa 3	9	4
Casa 4	6	6
Casa 5	4	4
Casa 6	9	9
Casa 7	5	5
Casa 8	7	7
Casa 9	1	1
Casa 10	4	4
Casa 11	5	5
Casa 12	3	3
Casa 13	5	5
Casa 14	9	9
Casa 15	6	6
Casa 16	5	2
Casa 17	6	6
Casa 18	3	3
Casa 19	4	9
Casa 20	1	1
Casa 21	8	8
Casa 22	6	6
Casa 23	7	7
Casa 24	3	3
Casa 25	2	2
Casa 26	1	1
Casa 27	5	2
Casa 28	1	1

	Reconhecido	Correto
Casa 29	2	2
Casa 30	9	9
Casa 31	5	5
Casa 32	8	8
Casa 33	4	4
Casa 34	9	5
Casa 35	8	8
Casa 36	5	5
Casa 37	1	9
Casa 38	9	9
Casa 39	1	1
Casa 40	3	3
Casa 41	1	1
Casa 42	5	2
Casa 43	5	5
Casa 44	7	1
Casa 45	4	4
Casa 46	5	5
Casa 47	3	3
Casa 48	5	4
Casa 49	5	5
Casa 50	2	2
Casa 51	3	3
Casa 52	2	2
Casa 53	8	8
Casa 54	6	6
Casa 55	6	6
Casa 56	9	9

Tabela 4.3: Teste2

4.2.4 Teste Nº3

	Reconhecido	Correto
Casa 1	5	5
Casa 2	9	9
Casa 3	4	4
Casa 4	2	2
Casa 5	6	6
Casa 6	2	9
Casa 7	4	4
Casa 8	2	2
Casa 9	8	8
Casa 10	4	1
Casa 11	7	7
Casa 12	5	5
Casa 13	7	7
Casa 14	3	3
Casa 15	4	4
Casa 16	8	8
Casa 17	6	6
Casa 18	2	2
Casa 19	7	7
Casa 20	7	7
Casa 21	5	5
Casa 22	3	3
Casa 23	3	3
Casa 24	5	5
Casa 25	6	6

	Reconhecido	Correto
Casa 26	4	1
Casa 27	8	8
Casa 28	7	7
Casa 29	1	1
Casa 30	4	4
Casa 31	8	8
Casa 32	3	3
Casa 33	5	9
Casa 34	2	7
Casa 35	1	1
Casa 36	5	5
Casa 37	6	6
Casa 38	1	1
Casa 39	3	3
Casa 40	4	4
Casa 41	5	4
Casa 42	2	2
Casa 43	5	5
Casa 44	5	2
Casa 45	5	5
Casa 46	8	8
Casa 47	4	9
Casa 48	8	8
Casa 49	7	7
Casa 50	8	8
Casa 51	4	9

Tabela 4.4: Teste3

4.2.5 Teste Nº4

	Reconhecido	Correto
Casa 1	1	1
Casa 2	2	2
Casa 3	4	4
Casa 4	3	3
Casa 5	4	4
Casa 6	5	5
Casa 7	6	6
Casa 8	7	7
Casa 9	8	8
Casa 10	9	9
Casa 11	6	9
Casa 12	8	8
Casa 13	5	6
Casa 14	5	5
Casa 15	1	1
Casa 16	1	1
Casa 17	4	4
Casa 18	5	5
Casa 19	4	7
Casa 20	3	2
Casa 21	3	3
Casa 22	4	4
Casa 23	5	5

	Reconhecido	Correto
Casa 24	6	6
Casa 25	7	7
Casa 26	8	8
Casa 27	1	1
Casa 28	2	2
Casa 29	5	3
Casa 30	5	5
Casa 31	6	6
Casa 32	8	8
Casa 33	9	9
Casa 34	7	7
Casa 35	3	3
Casa 36	5	2
Casa 37	4	4
Casa 38	5	5
Casa 39	6	6
Casa 40	9	9
Casa 41	7	7
Casa 42	8	8
Casa 43	2	2
Casa 44	3	3
Casa 45	4	4
Casa 46	5	5

Tabela 4.5: Teste4

4.2.6 Teste Nº5

	Reconhecido	Correto
Casa 1	4	4
Casa 2	3	3
Casa 3	2	2
Casa 4	1	1
Casa 5	5	9
Casa 6	7	7
Casa 7	8	8
Casa 8	1	1
Casa 9	5	3
Casa 10	7	7
Casa 11	4	4
Casa 12	5	5
Casa 13	8	8
Casa 14	1	1
Casa 15	3	3
Casa 16	5	5
Casa 17	7	7
Casa 18	8	8
Casa 19	3	3
Casa 20	6	6
Casa 21	7	7
Casa 22	5	2
Casa 23	8	8
Casa 24	3	3
Casa 25	1	1
Casa 26	7	7
Casa 27	3	3
Casa 28	9	9
Casa 29	4	4
Casa 30	5	3
Casa 31	5	2
Casa 32	5	5
Casa 33	6	6
Casa 34	1	4
Casa 35	1	1
Casa 36	5	5
Casa 37	1	1
Casa 38	5	2
Casa 39	3	3
Casa 40	4	4
Casa 41	5	5
Casa 42	5	5
Casa 43	2	2
Casa 44	5	5
Casa 45	6	6
Casa 46	7	7
Casa 47	8	8
Casa 48	6	6
Casa 49	6	6
Casa 50	4	4
Casa 51	5	3

Tabela 4.6: Teste5

4.2.7 Teste Nº6

	Reconhecido	Correto
Casa 1	6	6
Casa 2	7	7
Casa 3	5	5
Casa 4	4	4
Casa 5	8	8
Casa 6	1	1
Casa 7	6	9
Casa 8	6	2
Casa 9	8	8
Casa 10	2	5
Casa 11	8	8
Casa 12	3	3
Casa 13	9	9
Casa 14	1	1
Casa 15	2	2
Casa 16	6	6
Casa 17	5	5
Casa 18	6	6
Casa 19	4	4
Casa 20	5	5
Casa 21	4	1
Casa 22	5	3
Casa 23	2	2
Casa 24	7	7
Casa 25	1	1
Casa 26	9	9
Casa 27	2	2
Casa 28	5	3

	Reconhecido	Correto
Casa 29	9	9
Casa 30	8	8
Casa 31	5	5
Casa 32	4	4
Casa 33	8	8
Casa 34	1	1
Casa 35	8	8
Casa 36	9	9
Casa 37	4	4
Casa 38	6	6
Casa 39	2	2
Casa 40	5	3
Casa 41	7	7
Casa 42	1	1
Casa 43	7	7
Casa 44	3	3
Casa 45	4	4
Casa 46	5	4
Casa 47	7	7
Casa 48	2	2
Casa 49	9	4
Casa 50	9	9
Casa 51	8	8
Casa 52	9	9
Casa 53	9	9
Casa 54	8	8
Casa 55	6	6
Casa 56	5	5

Tabela 4.7: Teste6

4.2.8 Teste Nº7

	Reconhecido	Correto
Casa 1	4	4
Casa 2	5	5
Casa 3	6	2
Casa 4	7	7
Casa 5	6	3
Casa 6	6	2
Casa 7	9	9
Casa 8	1	1
Casa 9	2	2
Casa 10	5	5
Casa 11	4	4
Casa 12	2	3
Casa 13	6	6
Casa 14	4	4
Casa 15	9	9
Casa 16	6	8
Casa 17	4	5
Casa 18	7	7
Casa 19	9	3
Casa 20	7	7
Casa 21	2	5
Casa 22	3	3
Casa 23	6	6

	Reconhecido	Correto
Casa 24	9	8
Casa 25	9	9
Casa 26	6	3
Casa 27	2	2
Casa 28	2	5
Casa 29	6	6
Casa 30	8	8
Casa 31	6	6
Casa 32	6	8
Casa 33	9	9
Casa 34	4	4
Casa 35	2	3
Casa 36	2	2
Casa 37	6	6
Casa 38	8	8
Casa 39	7	7
Casa 40	4	4
Casa 41	5	5
Casa 42	5	5
Casa 43	6	2
Casa 44	5	9
Casa 45	9	3
Casa 46	4	4

Tabela 4.8: Teste7

4.2.9 Teste Nº8

	Reconhecido	Correto
Casa 1	1	1
Casa 2	9	9
Casa 3	7	7
Casa 4	8	8
Casa 5	6	6
Casa 6	5	2
Casa 7	3	3
Casa 8	4	4
Casa 9	7	7
Casa 10	6	6
Casa 11	4	4
Casa 12	7	7
Casa 13	4	4
Casa 14	5	5
Casa 15	7	7
Casa 16	6	6
Casa 17	7	7
Casa 18	5	5
Casa 19	6	6
Casa 20	6	6
Casa 21	8	8
Casa 22	3	3
Casa 23	5	2
Casa 24	1	1
Casa 25	6	6

	Reconhecido	Correto
Casa 26	6	6
Casa 27	7	7
Casa 28	4	4
Casa 29	7	7
Casa 30	3	3
Casa 31	5	5
Casa 32	2	2
Casa 33	1	1
Casa 34	9	9
Casa 35	3	3
Casa 36	5	5
Casa 37	4	4
Casa 38	3	3
Casa 39	4	2
Casa 40	5	2
Casa 41	4	4
Casa 42	2	2
Casa 43	4	4
Casa 44	5	5
Casa 45	3	3
Casa 46	1	1
Casa 47	6	5
Casa 48	5	2
Casa 49	4	1
Casa 50	7	7
Casa 51	5	5

Tabela 4.9: Teste8

4.2.10 Teste Nº9

	Reconhecido	Correto
Casa 1	2	2
Casa 2	1	1
Casa 3	2	2
Casa 4	1	1
Casa 5	5	5
Casa 6	6	6
Casa 7	3	2
Casa 8	3	3
Casa 9	6	6
Casa 10	8	8
Casa 11	9	5
Casa 12	3	3
Casa 13	4	4
Casa 14	4	4
Casa 15	5	5
Casa 16	7	7
Casa 17	5	7
Casa 18	1	1
Casa 19	7	4
Casa 20	5	5
Casa 21	4	4
Casa 22	8	8
Casa 23	5	5
Casa 24	5	5
Casa 25	6	6
Casa 26	8	2
Casa 27	9	9
Casa 28	6	2
Casa 29	6	6
Casa 30	6	6
Casa 31	6	6
Casa 32	7	7
Casa 33	1	1
Casa 34	2	2
Casa 35	3	3
Casa 36	3	3
Casa 37	6	6
Casa 38	9	9
Casa 39	1	1
Casa 40	7	7
Casa 41	4	4
Casa 42	4	4
Casa 43	7	7
Casa 44	8	8
Casa 45	5	5
Casa 46	2	2
Casa 47	7	7
Casa 48	5	5
Casa 49	8	8
Casa 50	7	7
Casa 51	8	8

Tabela 4.10: Teste9

4.2.11 Teste Nº10

	Reconhecido	Correto
Casa 1	6	6
Casa 2	1	7
Casa 3	9	9
Casa 4	1	7
Casa 5	4	4
Casa 6	2	2
Casa 7	9	9
Casa 8	2	2
Casa 9	5	5
Casa 10	1	1
Casa 11	8	8
Casa 12	8	3
Casa 13	1	1
Casa 14	3	3
Casa 15	4	4
Casa 16	1	1
Casa 17	5	3
Casa 18	8	8
Casa 19	5	5
Casa 20	5	4
Casa 21	9	9
Casa 22	5	4
Casa 23	9	9
Casa 24	2	2
Casa 25	4	4
Casa 26	5	3

	Reconhecido	Correto
Casa 27	8	8
Casa 28	3	3
Casa 29	5	5
Casa 30	1	1
Casa 31	6	6
Casa 32	9	9
Casa 33	2	5
Casa 34	1	1
Casa 35	7	7
Casa 36	8	8
Casa 37	5	5
Casa 38	5	3
Casa 39	1	1
Casa 40	9	2
Casa 41	5	5
Casa 42	2	2
Casa 43	6	6
Casa 44	4	7
Casa 45	1	1
Casa 46	6	6
Casa 47	1	1
Casa 48	8	8
Casa 49	3	3
Casa 50	4	5
Casa 51	8	8

Tabela 4.11: Teste10

4.2.12 Teste Nº11

	Reconhecido	Correto
Casa 1	1	1
Casa 2	2	2
Casa 3	3	3
Casa 4	4	4
Casa 5	8	8
Casa 6	3	5
Casa 7	3	3
Casa 8	7	4
Casa 9	2	2
Casa 10	1	1
Casa 11	9	9
Casa 12	5	7
Casa 13	5	3
Casa 14	7	7
Casa 15	5	6
Casa 16	5	5
Casa 17	4	4
Casa 18	2	2
Casa 19	5	5
Casa 20	4	4
Casa 21	7	7
Casa 22	3	3
Casa 23	1	1
Casa 24	2	2
Casa 25	5	3
Casa 26	4	4
Casa 27	5	5
Casa 28	1	1
Casa 29	2	2
Casa 30	5	2
Casa 31	8	8
Casa 32	2	2
Casa 33	3	6
Casa 34	4	4
Casa 35	5	7
Casa 36	5	3
Casa 37	9	9
Casa 38	5	5
Casa 39	5	2
Casa 40	1	1
Casa 41	6	6
Casa 42	5	5
Casa 43	6	6
Casa 44	1	1
Casa 45	9	9
Casa 46	5	5
Casa 47	4	4
Casa 48	5	3
Casa 49	5	2
Casa 50	1	1
Casa 51	7	7

Tabela 4.12: Teste11

4.2.13 Teste Nº12

	Reconhecido	Correto
Casa 1	6	3
Casa 2	6	6
Casa 3	2	2
Casa 4	9	9
Casa 5	5	2
Casa 6	7	7
Casa 7	6	6
Casa 8	3	3
Casa 9	1	1
Casa 10	9	9
Casa 11	5	5
Casa 12	3	3
Casa 13	8	8
Casa 14	4	4
Casa 15	6	6
Casa 16	3	4
Casa 17	5	5
Casa 18	8	8
Casa 19	9	9
Casa 20	4	1
Casa 21	1	1
Casa 22	8	8
Casa 23	9	9
Casa 24	6	6
Casa 25	8	8

	Reconhecido	Correto
Casa 26	2	2
Casa 27	5	5
Casa 28	2	2
Casa 29	7	7
Casa 30	7	7
Casa 31	1	1
Casa 32	4	4
Casa 33	6	6
Casa 34	5	3
Casa 35	6	6
Casa 36	1	1
Casa 37	8	8
Casa 38	1	1
Casa 39	3	3
Casa 40	2	2
Casa 41	7	7
Casa 42	8	8
Casa 43	2	2
Casa 44	4	4
Casa 45	9	5
Casa 46	4	4
Casa 47	5	5
Casa 48	5	9
Casa 49	4	4
Casa 50	6	6
Casa 51	9	9

Tabela 4.13: Teste12

4.2.14 Teste Nº13

	Reconhecido	Correto
Casa 1	6	9
Casa 2	9	4
Casa 3	3	3
Casa 4	7	7
Casa 5	4	4
Casa 6	8	8
Casa 7	2	2
Casa 8	6	5
Casa 9	1	1
Casa 10	2	2
Casa 11	3	3
Casa 12	6	6
Casa 13	1	1
Casa 14	6	6
Casa 15	7	7
Casa 16	8	8
Casa 17	6	9
Casa 18	4	4
Casa 19	5	5
Casa 20	2	2
Casa 21	7	7
Casa 22	2	2
Casa 23	4	4

	Reconhecido	Correto
Casa 24	8	8
Casa 25	2	2
Casa 26	3	3
Casa 27	1	1
Casa 28	8	8
Casa 29	2	5
Casa 30	9	9
Casa 31	9	9
Casa 32	2	2
Casa 33	7	7
Casa 34	4	4
Casa 35	6	6
Casa 36	2	2
Casa 37	5	5
Casa 38	6	6
Casa 39	3	3
Casa 40	1	1
Casa 41	7	7
Casa 42	5	5
Casa 43	1	1
Casa 44	4	4
Casa 45	8	8
Casa 46	9	9

Tabela 4.14: Teste13

4.2.15 Teste Nº14

	Reconhecido	Correto
Casa 1	9	9
Casa 2	4	4
Casa 3	6	3
Casa 4	5	5
Casa 5	6	6
Casa 6	4	7
Casa 7	9	9
Casa 8	5	3
Casa 9	7	7
Casa 10	9	9
Casa 11	3	3
Casa 12	6	6
Casa 13	2	2
Casa 14	7	7
Casa 15	4	7
Casa 16	8	8
Casa 17	6	6
Casa 18	5	2
Casa 19	5	3
Casa 20	4	4
Casa 21	9	9
Casa 22	8	5
Casa 23	1	1

	Reconhecido	Correto
Casa 24	7	7
Casa 25	8	3
Casa 26	2	2
Casa 27	5	3
Casa 28	5	5
Casa 29	8	8
Casa 30	6	6
Casa 31	8	8
Casa 32	6	6
Casa 33	1	1
Casa 34	4	4
Casa 35	2	3
Casa 36	5	5
Casa 37	2	2
Casa 38	1	1
Casa 39	2	9
Casa 40	6	5
Casa 41	6	6
Casa 42	3	3
Casa 43	9	9
Casa 44	6	6
Casa 45	7	7
Casa 46	1	1

Tabela 4.15: Teste14

4.2.16 Teste Nº15

	Reconhecido	Correto
Casa 1	1	1
Casa 2	4	4
Casa 3	7	7
Casa 4	9	9
Casa 5	2	2
Casa 6	8	8
Casa 7	2	2
Casa 8	5	5
Casa 9	8	8
Casa 10	4	4
Casa 11	1	1
Casa 12	6	6
Casa 13	3	3
Casa 14	7	7
Casa 15	6	6
Casa 16	9	9
Casa 17	5	5
Casa 18	9	9
Casa 19	7	7
Casa 20	4	4
Casa 21	9	9
Casa 22	4	4
Casa 23	5	5
Casa 24	4	4
Casa 25	7	7
Casa 26	6	6
Casa 27	3	3
Casa 28	6	6
Casa 29	8	8
Casa 30	1	1
Casa 31	8	8
Casa 32	9	9
Casa 33	1	1
Casa 34	1	1
Casa 35	1	1
Casa 36	3	3
Casa 37	4	4
Casa 38	1	1
Casa 39	8	8
Casa 40	5	5
Casa 41	8	8
Casa 42	3	3
Casa 43	9	9
Casa 44	6	6
Casa 45	4	9
Casa 46	9	9
Casa 47	4	4
Casa 48	3	7
Casa 49	5	5
Casa 50	2	7
Casa 51	7	7

Tabela 4.16: Teste15

4.2.17 Teste Nº16

	Reconhecido	Correto
Casa 1	1	1
Casa 2	9	9
Casa 3	1	1
Casa 4	9	9
Casa 5	1	1
Casa 6	1	1
Casa 7	1	2
Casa 8	4	4
Casa 9	2	2
Casa 10	8	8
Casa 11	2	2
Casa 12	1	1
Casa 13	3	3
Casa 14	2	2
Casa 15	3	3
Casa 16	5	3
Casa 17	5	3
Casa 18	2	2
Casa 19	4	4
Casa 20	4	5
Casa 21	2	2
Casa 22	4	4
Casa 23	4	4
Casa 24	7	7
Casa 25	1	4

	Reconhecido	Correto
Casa 26	5	5
Casa 27	3	3
Casa 28	5	5
Casa 29	4	4
Casa 30	6	6
Casa 31	5	5
Casa 32	6	6
Casa 33	3	5
Casa 34	6	6
Casa 35	5	5
Casa 36	6	6
Casa 37	6	6
Casa 38	6	6
Casa 39	5	5
Casa 40	7	7
Casa 41	7	7
Casa 42	4	4
Casa 43	7	7
Casa 44	8	8
Casa 45	1	1
Casa 46	1	1
Casa 47	9	9
Casa 48	7	7
Casa 49	8	3
Casa 50	6	6
Casa 51	9	9

Tabela 4.17: Teste16

4.2.18 Teste Nº17

	Reconhecido	Correto
Casa 1	1	1
Casa 2	2	2
Casa 3	6	6
Casa 4	7	7
Casa 5	8	8
Casa 6	7	7
Casa 7	5	5
Casa 8	4	4
Casa 9	3	3
Casa 10	2	2
Casa 11	5	2
Casa 12	4	4
Casa 13	3	3
Casa 14	2	2
Casa 15	3	3
Casa 16	4	4
Casa 17	6	6
Casa 18	7	7
Casa 19	8	8
Casa 20	5	9
Casa 21	1	1
Casa 22	3	3
Casa 23	2	2

	Reconhecido	Correto
Casa 24	1	1
Casa 25	4	4
Casa 26	6	6
Casa 27	7	7
Casa 28	3	3
Casa 29	2	2
Casa 30	6	6
Casa 31	9	9
Casa 32	5	5
Casa 33	2	2
Casa 34	4	5
Casa 35	1	1
Casa 36	2	2
Casa 37	3	3
Casa 38	1	4
Casa 39	7	7
Casa 40	8	8
Casa 41	8	8
Casa 42	1	1
Casa 43	3	3
Casa 44	5	5
Casa 45	6	6
Casa 46	7	7

Tabela 4.18: Teste17

4.2.19 Teste Nº18

	Reconhecido	Correto
Casa 1	5	3
Casa 2	6	6
Casa 3	6	6
Casa 4	7	7
Casa 5	4	4
Casa 6	5	5
Casa 7	5	3
Casa 8	8	8
Casa 9	9	9
Casa 10	3	3
Casa 11	4	4
Casa 12	2	2
Casa 13	1	1
Casa 14	3	3
Casa 15	7	7
Casa 16	1	1
Casa 17	3	3
Casa 18	9	9
Casa 19	4	4
Casa 20	8	8
Casa 21	5	6
Casa 22	9	9
Casa 23	9	9

	Reconhecido	Correto
Casa 24	4	4
Casa 25	5	5
Casa 26	6	6
Casa 27	8	8
Casa 28	4	4
Casa 29	5	3
Casa 30	7	7
Casa 31	6	6
Casa 32	9	4
Casa 33	9	9
Casa 34	5	5
Casa 35	4	4
Casa 36	8	8
Casa 37	1	1
Casa 38	2	2
Casa 39	6	6
Casa 40	9	9
Casa 41	8	8
Casa 42	6	6
Casa 43	7	7
Casa 44	4	4
Casa 45	8	8

Tabela 4.19: Teste18

4.2.20 Teste Nº19

	Reconhecido	Correto
Casa 1	1	1
Casa 2	6	9
Casa 3	1	1
Casa 4	9	9
Casa 5	1	1
Casa 6	6	2
Casa 7	6	6
Casa 8	8	8
Casa 9	5	2
Casa 10	3	3
Casa 11	3	8
Casa 12	7	7
Casa 13	3	3
Casa 14	7	7
Casa 15	4	4
Casa 16	5	5
Casa 17	3	3
Casa 18	4	4
Casa 19	7	7
Casa 20	8	8
Casa 21	1	1
Casa 22	6	6
Casa 23	4	4
Casa 24	3	5
Casa 25	6	6
Casa 26	2	2
Casa 27	4	4
Casa 28	5	3
Casa 29	7	7
Casa 30	4	5
Casa 31	5	5
Casa 32	9	9
Casa 33	5	5
Casa 34	5	5
Casa 35	6	6
Casa 36	6	6
Casa 37	4	4
Casa 38	8	8
Casa 39	4	4
Casa 40	7	7
Casa 41	2	2
Casa 42	3	3
Casa 43	7	7
Casa 44	8	6
Casa 45	7	7
Casa 46	2	5
Casa 47	8	8
Casa 48	7	7
Casa 49	5	3
Casa 50	1	1
Casa 51	3	5

Tabela 4.20: Teste19

4.2.21 Teste Nº20

	Reconhecido	Correto
Casa 1	6	6
Casa 2	7	7
Casa 3	3	3
Casa 4	8	8
Casa 5	2	9
Casa 6	5	4
Casa 7	6	9
Casa 8	8	8
Casa 9	1	1
Casa 10	5	5
Casa 11	7	7
Casa 12	2	4
Casa 13	3	3
Casa 14	7	7
Casa 15	2	2
Casa 16	6	6
Casa 17	6	9
Casa 18	5	4
Casa 19	1	1
Casa 20	5	5
Casa 21	2	6
Casa 22	1	1
Casa 23	2	3
Casa 24	5	5
Casa 25	8	8
Casa 26	3	3
Casa 27	7	7
Casa 28	3	2

	Reconhecido	Correto
Casa 29	6	6
Casa 30	8	8
Casa 31	4	9
Casa 32	1	7
Casa 33	8	3
Casa 34	9	9
Casa 35	1	1
Casa 36	6	6
Casa 37	5	5
Casa 38	2	2
Casa 39	2	4
Casa 40	2	7
Casa 41	2	2
Casa 42	4	4
Casa 43	5	5
Casa 44	2	2
Casa 45	2	4
Casa 46	8	8
Casa 47	3	3
Casa 48	3	3
Casa 49	8	8
Casa 50	2	2
Casa 51	5	5
Casa 52	8	8
Casa 53	6	6
Casa 54	9	9
Casa 55	9	9
Casa 56	6	6

Tabela 4.21: Teste20

4.3 Resultados

Nesta secção irão ser estudados os resultados dos testes, o que o modelo reconheceu bem, e o que reconheceu mal, quais as situações em que estes erros ocorreram, o porquê e as possíveis soluções.

Na tabela seguinte irá ser mostrado a percentagem de precisão de todos os testes feitos.

	Nº de dígitos	Nº de dígitos errados	Precisão
Teste 1	46	10	78%
Teste 2	56	9	84%
Teste 3	51	9	82%
Teste 4	46	6	87%
Teste 5	51	7	86%
Teste 6	56	8	86%
Teste 7	46	16	65%
Teste 8	51	7	86%
Teste 9	51	6	88%
Teste 10	51	12	76%
Teste 11	51	12	76%
Teste 12	51	7	86%
Teste 13	46	5	89%
Teste 14	46	13	72%
Teste 15	51	3	94%
Teste 16	51	7	86%
Teste 17	46	4	91%
Teste 18	45	5	89%
Teste 19	51	11	78%
Teste 20	56	14	75%

Tabela 4.22: Testes feitos

4.3.1 Iluminação

Nesta subsecção irá ser explicada a influência que a iluminação teve no reconhecimento dos vários puzzles.

1. **Tipo 1** - Percentagem média de precisão - 82,75%
2. **Tipo 2** - Percentagem média de precisão - 80,75%
3. **Tipo 3** - Percentagem média de precisão - 81,5%
4. **Tipo 4** - Percentagem média de precisão - 85,25%
5. **Tipo 5** - Percentagem média de precisão - 83,25%

Pelos testes que foram feitos, a iluminação da imagem, têm média influência no resultado, o tipo 4 parece ter tido o melhor resultado, embora contenha o segundo pior resultado têm a maior média dos 5, enquanto os tipos com um maior balanço em termos de média, como o tipo 2, e em menor escala o tipo 1 e 5, o tipo 4 continua a ter uma maior média.

Logo embora a iluminação tenha alguma relevância no resultado, não é tão significativa como outros fatores.

4.3.2 Tipo de utensílio de escrita

Nesta subsecção irá ser explicada a influência que Tipo de utensílio de escrita teve no reconhecimento dos vários puzzles.

1. **Azul** - Foi utilizado em 6 testes - Percentagem média de precisão -> 77,8%
2. **Lápis** - Foi utilizado em 3 testes - Percentagem média de precisão -> 83%
3. **Preta** - Foi utilizado em 3 testes - Percentagem média de precisão -> 88%
4. **Vermelha** - Foi utilizado em 4 testes - Percentagem média de precisão -> 81,5%
5. **Verde** - Foi utilizado em 4 testes - Percentagem média de precisão -> 87%

Com estes resultados, parece que os utensílios de escrita para obter uma maior precisão no reconhecimento seria a caneta de cor preta a caneta de cor verde, e as piores seriam o lápis e a caneta de cor vermelha, mas um ponto que será necessário apontar seria que, embora a caneta azul tenha uma menor médio essa media seria influenciada pelo maior número de testes(logo há mais margem para erro), e a caneta azul foi utilizada pela pessoa com o tipo de letra menos reconhecido que seria o tipo 3, embora que mesmo que esses dois não fossem tido em conta, á média ainda seria pior que os dois maiores(82,5%).

Logo com estes resultados, parece que o tipo de utensílio de escrita têm um pouco de mais influência do que a iluminação, embora a combinação de tipo de utensílio de escrita e iluminação, também têm alguma relevância, pois embora se combinado a caneta de cor verde com iluminação de tipo 4, têm se a maior percentagem de sucesso(94%), e combinação de caneta de cor preta, com iluminação de tipo 5, têm-se a segunda maior(91%), e por outro lado combinado a caneta de cor verde com iluminação de tipo 1, têm-se uma

menor percentagem de sucesso(82%), e o mesmo ocorre na combinação a caneta de cor preta com iluminação de tipo 2, têm-se uma menor percentagem de sucesso(86%) embora não tão radical a diferença como a de cor verde.

4.3.3 Tipo de Letra

Nesta subsecção irá ser explicada a influência que Tipo de utensílio de escrita teve no reconhecimento dos vários puzzles.

1. Tipo de Letra 1 - Utilizado 9 vezes - Percentagem média de precisão -> 84,5%

2. Tipo de Letra 2 - Utilizado 9 vezes - Percentagem média de precisão -> 84%

3. Tipo de Letra 3 - Utilizado 2 vezes - Percentagem média de precisão -> 68,5%

Embora seja observado uma grande diferença entre os tipos 1 e 2, e o tipo 3, em que os tipos 1 e 2 foram utilizados 9 vezes, para o teste e o tipo 3 só 2, isto foi feito para refletir o número de puzzles de cada de tipo que foram enviados para treinamento e avaliação, em que havia significante menos puzzles com letra do tipo 3.

Isto leva a conclusão que é o tipo de letra ou a forma que as pessoas escrevem que influência mais os resultados, pois é a pessoa que têm uma forma de escrever mais semelhante á maior parte do conjunto de treinamento que obtém melhor resultados.

Logo, para chegar a uma maior percentagem de precisão, é necessário obter um grande número de puzzles divididos igualmente, em vários conjuntos de puzzles feitos por pessoas com tipos de letra diferentes, quanto maior o número e quanto mais diferente o tipo de letra melhor seria o resultado final.

4.3.4 Confusão entre Números

Nesta secção, vamos identificar quais os números que o modelo teve mais dificuldade em reconhecer e com quais outros números ele confundiu.

O resultado desta identificação está presente na tabela seguinte:

	1	2	3	4	5	6	7	8	9
1	-	0	0	5	0	0	1	0	0
2	1	-	3	1	19	7	0	2	1
3	0	5	-	0	26	5	0	4	2
4	3	3	1	-	7	0	4	0	4
5	0	6	4	5	-	3	0	1	3
6	0	1	1	0	3	-	0	1	0
7	4	2	1	4	3	0	-	0	0
8	0	1	0	0	0	2	0	-	1
9	1	3	0	5	6	8	0	0	-

Tabela 4.23: Confusão entre os números

Nestes testes, é óbvio que o programa têm mais problemas quando identificar o número 2 e 3, com confusão com o número 5, e também (embora em menos escala) o contrário também ocorre. Este e outros erros menos comuns, como por exemplo confusão da letra 1 com a letra 4, a letra 7 com a letra 1 e 4, e até a letra 9 com a letra 5 e 6, têm como causa o mesmo problema explicado na subsecção 4.3.3. A forma das pessoas escrever números causa o modelo de confundir uns números com outros.

Como por exemplo a pessoa com o tipo de letra(1), a forma que ele escreve a letra 5 é similar á maneira que escreve 2 e 3. Como é possível ver nas seguintes imagens:

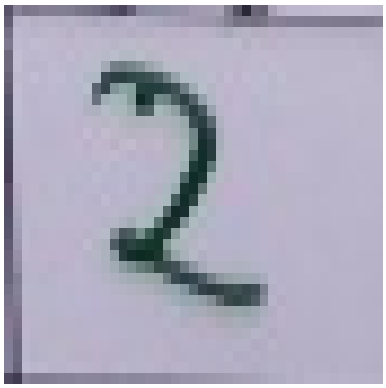

Figura 4.21: Digito 2

Figura 4.22: Digito 3

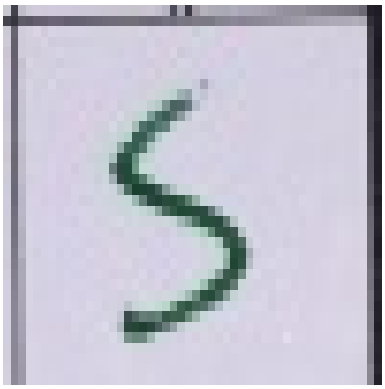

Figura 4.23: Digito 5

E o mesmo pode ser observado com a combinação de 1 e 7, e 6 e 9:

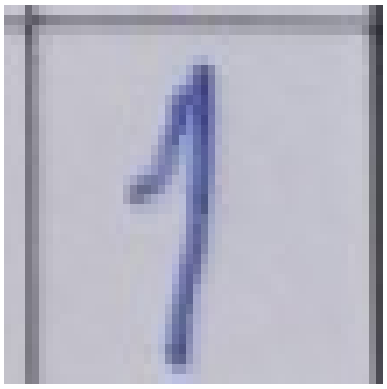

Figura 4.24: Digito 1

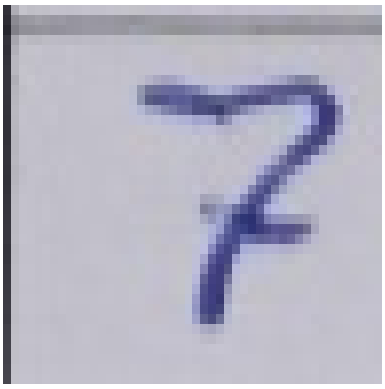

Figura 4.25: Digito 7

Figura 4.26: Digito 6

Figura 4.27: Digito 9

E também há a possibilidade destes erros terem causa na extração da imagem do puzzle, em que dependendo da perspetiva da foto e a forma que o

programa reconhece os pontos para a extração de cada digito individual, alguns dígitos podem ser parcialmente recortados, como exemplo na seguinte imagem:

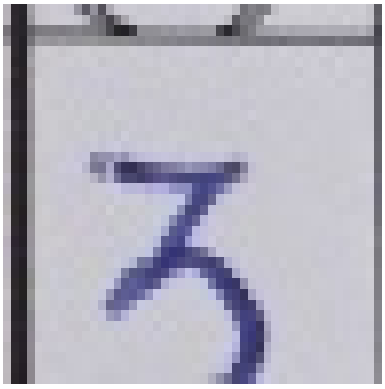

Figura 4.28: Digito 3 Recortado

Capítulo

5

Conclusões e Trabalho Futuro

5.1 Conclusões Principais

O principal objetivo deste projeto era a criação de um programa que conseguisse criar e disponibilizar puzzles de sudoku para um utilizador resolver, e consequentemente verificar e corrigir a resolução proposta pelo utilizador, como também resolver um puzzle dado pelo utilizador.

Para a resolução de tal objetivo foi feito um estudo pela história e funcionamento do puzzle conhecido como sudoku, um estudo do campo de inteligência artificial, e mais especificamente machine learning, e a criação manual da nossa parte de uma base de dados de puzzles de sudoku.

Foi proposto um método para gerar e preencher matrizes e transformá-las em imagens de puzzles sudoku, para "tratar"a imagem para que seja mais fácil e eficaz o treinamento dum modelo de reconhecimento utilizando a base de dados criada.

Foram feitos vários testes, para avaliar o desempenho desse model, na identificação e classificação dos números escritos á mão de um puzzle sudoku.

5.2 Trabalho Futuro

Para um trabalho futuro, gostaria de criar uma aplicação para melhor comunicação com o utilizador, pois o programa ,da maneira como está feito, é necessário utilizar o código para que ele funcione, por exemplo é necessário dar o caminho no código para extrair os vários números.Seria interessante criar uma interface boa para juntar o código todo para melhor interatividade.

E ainda mais no futuro usar este programa como base, para um aplicação que não só consegue reconhecer números como também letras, e também que consiga reconhecer palavras e frases completas, tudo claro escrito manualmente.

Bibliografia

[1] Samuel Reich. E. mathematician claims breakthrough in sudoku puzzle. *Nature*, 2012. [Online] https://www.nature.com/articles/nature.2012.9751.

[2] Open source computer vision. [Online] https://docs.opencv.org/3.4/.

[3] The history of sudoku. [Online] https://sudoku.com/how-to-play/the-history-of-sudoku/.

[4] Wesley Chai. A timeline of machine learning history. [Online] https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History.

[5] Bernd Klein. Intro to machine learning with python. [Online] https://python-course.eu/machine-learning/.

[6] IBM. What is machine learning? [Online] https://www.ibm.com/topics/machine-learning#toc-machine-le-K7VszOk6.

[7] Harshit Dwivedi. How to use google colab for deep learning – complete tutorial. [Online] https://neptune.ai/blog/how-to-use-google-colab-for-deep-learning-complete-tutorial.