



Trabalho de Conclusão de Curso

**Compilação de Recomendações para Auxiliar no  
Desenvolvimento de Novas Soluções para  
Problemas de Visão Computacional**

Nelson Gomes Neto  
ngn@ic.ufal.br

**Orientadores:**  
Willy Carvalho Tiengo  
Rodrigo de Barros Paes

Maceió, Março de 2021

Nelson Gomes Neto

**Compilação de Recomendações para Auxiliar no  
Desenvolvimento de Novas Soluções para  
Problemas de Visão Computacional**

Monografia apresentada como requisito parcial para  
obtenção do grau de Bacharel em Ciência da Com-  
putação do Instituto de Computação da Universidade  
Federal de Alagoas.

Orientadores:

Willy Carvalho Tiengo  
Rodrigo de Barros Paes

Maceió, Março de 2021



# Agradecimentos

Gostaria de deixar aqui meus agradecimentos para todos que tornaram essa jornada incrível.

Agradeço Daniella Pereira de Souza e Nelson Gomes Júnior (meus pais) por sempre me encorajarem a estudar e sempre dar o meu melhor.

Agradeço minha namorada Núbia Nayane da Silva Monteiro por ser a melhor parceria ao longo dessa jornada.

Agradeço Igor Castro, Paula Harumi, Júlia Carmo, Thábita Arely e todos os meus amigos por serem meus amigos ao longo dessa jornada.

Agradeço Mardone Alves, Thiago Cortez e todos meus amigos da escola (Objetivo) por acreditarem em mim e terem tornado meus dias melhores.

Agradeço Alan Luiz, Mel Santos, Rosa Cecília, Diego Feitoza, Franchesco Porciuncula, André Pereira e todos os meus professores da escola (Objetivo) por me ensinar uma sólida base de conhecimentos.

Agradeço Lucas Amorim, Daniel Vassalo, Gabriel Barbosa, Lucas Raggi, Wagner Fontes, França MacDowell e todos os meus amigos da UFAL por me desafiar e apoiar ao longo dessa jornada.

Agradeço Tiago Marinho, Alfredo Lima, Romário Pantaleão e todos os que competiram comigo (competições de programação) por me motivar a competir e aprender tanto.

Agradeço Thiago José, João Ayalla, Jackson Barbosa, Ascânio Sávio, Lucas Agra e todos os meus alunos da UFAL por me ensinarem tanto.

Agradeço Lucas Peixoto, Aydano Machado, Bruno Georgevich, todos os professores e todos os alunos que fizeram pesquisa comigo por me ensinar a pesquisar.

Agradeço Márcio Ribeiro, Alcino Dall’Igna, Baldoíno Fonseca e todos os meus professores da UFAL por não existir menos que o melhor.

Agradeço Rodrigo Paes, Willy Tiengo e todos os integrantes do Edge por ter me dado uma oportunidade de colocar em prática tudo que aprendi.

# Resumo

O objetivo deste trabalho é ser um guia para quem busca alavancar seus conhecimentos em Visão Computacional através da Aprendizagem de Máquina. Com este objetivo, foi realizada uma pesquisa em problemas de Visão Computacional da plataforma *Kaggle*; onde foram coletadas diversas estatísticas dos problemas e suas melhores soluções. Neste trabalho de monografia, foram exemplificados e explicados bons direcionamentos para criação de novas soluções para problemas de Visão Computacional utilizando Aprendizagem de Máquina. Por fim, através de testes práticos, foi possível concluir a relevância desta monografia.

**Palavras-chave:** Inteligência Artificial, Visão Computacional, Kaggle

# Abstract

The objective of this work is to be a guide to who seeks to elevate your knowledge in Computer Vision through Machine Learning. With this objective, a survey was done on Computer Vision problems from Kaggle's website. Through it, various features from the problems and their best solutions were extracted. In this monography: examples and good directions were given to the creation of new Machine Learning solutions to Computer Vision problems. Lastly, through practical tests, it was possible to conclude the relevance of this monography.

**Key-words:** Artificial Intelligence, Computer Vision, Kaggle

# Lista de Figuras

|      |   |    |
|------|---|----|
| 2.1  | Bloco residual . . . . .  | 8  |
| 2.2  | A linha vermelha é quando foram alteradas as 3 dimensões da <i>EfficientNet</i> . . .                 | 10 |
| 2.3  | Módulo <i>Inception</i> . . . . .   | 11 |
| A.1  | Página inicial do Kaggle . . . . .  | 25 |
| A.2  | Descrição de uma competição do Kaggle . . . . .   | 25 |
| A.3  | Descrição dos dados de uma competição do Kaggle . . . . .   | 26 |
| A.4  | <i>Notebooks</i> de uma competição do Kaggle . . . . .  | 26 |
| A.5  | Trechos de código dentro de um <i>notebook</i> . . . . .  | 27 |
| A.6  | Texto rico dentro de um <i>notebook</i> . . . . .   | 27 |
| A.7  | Placar público . . . . .  | 28 |
| A.8  | Placar privado . . . . .  | 28 |
| B.1  | Exemplos da base de dados MNIST . . . . .   | 41 |
| B.2  | Exemplos com Cacto . . . . .  | 43 |
| B.3  | Exemplos sem Cacto . . . . .  | 43 |
| B.4  | Distribuição da base de dados . . . . .   | 43 |
| B.5  | Exemplos de imagens com gatos . . . . .   | 44 |
| B.6  | Exemplos de imagens com cachorros . . . . .   | 44 |
| B.7  | Exemplos da base de dados, com os pontos-chave representados em vermelho .                            | 45 |
| B.8  | Frequência da cada ponto-chave . . . . .  | 45 |
| B.9  | Exemplos da base de dados . . . . .   | 46 |
| B.10 | Frequência de cada baleia . . . . .   | 46 |
| B.11 | Exemplos da base de dados . . . . .   | 52 |
| B.12 | Frequências de todas classes . . . . .  | 52 |
| B.13 | Exemplos da base de dados . . . . .   | 53 |
| B.14 | Frequências de todas classes . . . . .  | 53 |
| B.15 | Exemplos da base de dados com a máscara . . . . .   | 55 |
| B.16 | Frequências de todas classes . . . . .  | 55 |
| B.17 | <i>Autoencoder</i> . . . . .  | 55 |
| B.18 | Exemplo da base de dados (HUVEC-04 Plate4 G08, que apresenta classe: sirna <sub>1</sub> 21) . . . . . | 57 |
| B.19 | Exemplo da base de dados ((RPE-04 Plate3 F21, que apresenta classe: sirna <sub>4</sub> 93) . . . . .  | 57 |
| B.20 | Distribuição da base de dados . . . . .   | 57 |
| B.21 | 3 exemplos de desenho para alguns exemplos de classes . . . . .                                       | 59 |
| B.22 | Distribuição da base de dados . . . . .   | 59 |
| B.23 | Exemplos da base de dados . . . . .   | 60 |
| B.24 | Distribuição da base de dados . . . . .   | 60 |
| B.25 | Exemplos da base de dados . . . . .   | 61 |

|  |    |
|--|----|
| B.26 Exemplos de algumas classes da base de dados . . . . .  | 63 |
| B.27 Convoluções da solução . . . . .  | 63 |
| B.28 Exemplos da base de dados . . . . .   | 64 |
| B.29 Distribuição da base de dados . . . . .   | 64 |
| B.30 Exemplos da base de dados, o título da imagem apresenta o tipo e um número<br>(1 se possui hemorragia, 0 se não possui) . . . . .               | 66 |
| B.31 Distribuição da base de dados . . . . .   | 66 |
| B.32 Exemplos da base de dados com a máscara em vermelho, o título da imagem<br>indica a profundidade da imagem . . . . .                            | 67 |
| B.33 Distribuição da base de dados . . . . .   | 67 |
| B.34 Exemplos da base de dados com os defeitos circulados, seguindo o mapeamento<br>de cor: 1 - Verde, 2 - Vermelho, 3 - Azul, 4 - Amarelo . . . . . | 69 |
| B.35 Distribuição da base de dados . . . . .   | 69 |
| B.36 Exemplos da base de dados . . . . .   | 70 |
| B.37 Distribuição da base de dados . . . . .   | 70 |
| B.38 Exemplos da base de dados . . . . .   | 71 |
| B.39 Distribuição da base de dados . . . . .   | 71 |
| B.40 Exemplos da base de dados . . . . .   | 73 |
| B.41 Distribuição da base de dados . . . . .   | 73 |
| B.42 Exemplos da base de dados . . . . .   | 74 |
| B.43 Formação de grafemas em Bengali . . . . .   | 75 |
| B.44 Exemplos da base de dados . . . . .   | 76 |
| B.45 Distribuição da base de dados . . . . .   | 76 |
| B.46 Distribuição da base de dados . . . . .   | 78 |
| B.47 Exemplos da base de dados . . . . .   | 79 |
| B.48 Distribuição da base de dados . . . . .   | 79 |
| B.49 Exemplos da base de dados . . . . .   | 81 |
| B.50 Distribuição da base de dados . . . . .   | 81 |

# **Lista de Algoritmos**

# Conteúdo

|   |            |
|---|------------|
| <b>Lista de Figuras</b>                         | <b>iii</b> |
| <b>1 Introdução</b>                             | <b>1</b>   |
| <b>2 O Guia</b>                                 | <b>3</b>   |
| 2.1 A construção do guia                        | 3          |
| 2.2 Guia  | 3          |
| 2.2.1 Entendendo o problema                     | 4          |
| 2.2.2 Pré-processamento                         | 4          |
| 2.2.3 Definição do modelo                       | 5          |
| 2.2.4 Arquiteturas e modelos da literatura      | 6          |
| 2.2.5 Abordagens da literatura                  | 7          |
| 2.2.6 Treinamento                               | 11         |
| 2.2.7 Validação                                 | 13         |
| <b>3 Validando O Guia</b>                       | <b>17</b>  |
| 3.1 <i>Digit Recognizer</i>                     | 17         |
| 3.2 <i>Dogs vs. Cats Redux: Kernels Edition</i> | 17         |
| 3.3 <i>1ª Competição de IA EDGE-Huawei</i>      | 18         |
| <b>4 Conclusão</b>                              | <b>19</b>  |
| 4.1 Trabalhos futuros                           | 19         |
| <b>Referências bibliográficas</b>               | <b>21</b>  |
| <b>A Coleta e Anotação de Dados</b>             | <b>24</b>  |
| A.1 A plataforma Kaggle                         | 24         |
| A.2 A coleta de dados                           | 24         |
| A.3 Atributos                                   | 31         |
| A.3.1 Atributos do problema                     | 32         |
| A.3.2 Atributos da base de dados                | 34         |
| A.3.3 Atributos da solução                      | 35         |
| <b>B Catálogo de Competições</b>                | <b>40</b>  |
| B.1 <i>Digit Recognizer</i>                     | 40         |
| B.1.1 Base de dados                             | 40         |
| B.1.2 Solução                                   | 41         |
| B.1.3 Resultado                                 | 41         |
| B.2 <i>Aerial Cactus Identification</i>         | 41         |

|        |  |    |
|--------|--|----|
| B.2.1  | Base de dados                              | 42 |
| B.2.2  | Solução                                    | 42 |
| B.2.3  | Resultado                                  | 42 |
| B.3    | Dogs vs Cats                               | 42 |
| B.3.1  | Base de dados                              | 43 |
| B.3.2  | Solução                                    | 44 |
| B.3.3  | Resultado                                  | 44 |
| B.4    | Facial Keypoints Detection                 | 45 |
| B.4.1  | Base de dados                              | 45 |
| B.4.2  | Solução                                    | 45 |
| B.4.3  | Resultado                                  | 46 |
| B.5    | Humpback Whale Identification Challenge    | 46 |
| B.5.1  | Base de dados                              | 47 |
| B.5.2  | Solução                                    | 47 |
| B.5.3  | Resultado                                  | 50 |
| B.6    | Humpback Whale Identification              | 50 |
| B.6.1  | Base de dados                              | 50 |
| B.6.2  | Solução                                    | 50 |
| B.6.3  | Resultado                                  | 51 |
| B.7    | Dog Breed Identification                   | 51 |
| B.7.1  | Base de dados                              | 51 |
| B.7.2  | Solução                                    | 52 |
| B.7.3  | Resultado                                  | 52 |
| B.8    | Plant Seedlings Classification             | 53 |
| B.8.1  | Base de dados                              | 53 |
| B.8.2  | Solução                                    | 54 |
| B.8.3  | Resultado                                  | 54 |
| B.9    | Understanding Clouds from Satellite Images | 54 |
| B.9.1  | Base de dados                              | 54 |
| B.9.2  | Solução                                    | 54 |
| B.9.3  | Resultado                                  | 56 |
| B.10   | Recursion Cellular Image Classification    | 56 |
| B.10.1 | Base de dados                              | 56 |
| B.10.2 | Solução                                    | 56 |
| B.10.3 | Resultado                                  | 57 |
| B.11   | Quick, Draw! Doodle Recognition Challenge  | 57 |
| B.11.1 | Base de dados                              | 58 |
| B.11.2 | Solução                                    | 58 |
| B.11.3 | Resultado                                  | 59 |
| B.12   | State Farm Distracted Driver Detection     | 59 |
| B.12.1 | Base de dados                              | 60 |
| B.12.2 | Solução                                    | 60 |
| B.12.3 | Resultado                                  | 60 |
| B.13   | Kannada MNIST                              | 61 |
| B.13.1 | Base de dados                              | 61 |
| B.13.2 | Solução                                    | 61 |
| B.13.3 | Resultado                                  | 62 |
| B.14   | Leaf Classification                        | 62 |

|   |    |
|---|----|
| B.14.1 Base de dados . . . . .  | 62 |
| B.14.2 Solução . . . . .  | 62 |
| B.14.3 Resultado . . . . .  | 63 |
| B.15 Invasive Species Monitoring . . . . .                            | 64 |
| B.15.1 Base de dados . . . . .  | 64 |
| B.15.2 Solução . . . . .  | 64 |
| B.15.3 Resultado . . . . .  | 65 |
| B.16 RSNA Intracranial Hemorrhage Detection . . . . .                 | 65 |
| B.16.1 Base de dados . . . . .  | 65 |
| B.16.2 Solução . . . . .  | 65 |
| B.16.3 Resultado . . . . .  | 66 |
| B.17 TGS Salt Identification Challenge . . . . .                      | 66 |
| B.17.1 Base de dados . . . . .  | 67 |
| B.17.2 Solução . . . . .  | 67 |
| B.17.3 Resultado . . . . .  | 68 |
| B.18 Severstal: Steel Defect Detection . . . . .                      | 68 |
| B.18.1 Base de dados . . . . .  | 68 |
| B.18.2 Solução . . . . .  | 68 |
| B.18.3 Resultado . . . . .  | 69 |
| B.19 APTOS 2019 Blindness Detection . . . . .                         | 69 |
| B.19.1 Base de dados . . . . .  | 70 |
| B.19.2 Solução . . . . .  | 70 |
| B.19.3 Resultado . . . . .  | 71 |
| B.20 Airbus Ship Detection Challenge . . . . .                        | 71 |
| B.20.1 Base de dados . . . . .  | 71 |
| B.20.2 Solução . . . . .  | 72 |
| B.20.3 Resultado . . . . .  | 72 |
| B.21 Ultrasound Nerve Segmentation . . . . .                          | 72 |
| B.21.1 Base de dados . . . . .  | 73 |
| B.21.2 Solução . . . . .  | 73 |
| B.21.3 Resultado . . . . .  | 73 |
| B.22 Planet: Understanding the Amazon from Space . . . . .            | 74 |
| B.22.1 Base de dados . . . . .  | 74 |
| B.22.2 Solução . . . . .  | 74 |
| B.22.3 Resultado . . . . .  | 75 |
| B.23 Bengali.AI Handwritten Grapheme Classification . . . . .         | 75 |
| B.23.1 Base de dados . . . . .  | 76 |
| B.23.2 Solução . . . . .  | 76 |
| B.23.3 Resultado . . . . .  | 77 |
| B.24 SIIM-ACR Pneumothorax Segmentation . . . . .                     | 77 |
| B.24.1 Base de dados . . . . .  | 77 |
| B.24.2 Solução . . . . .  | 77 |
| B.24.3 Resultado . . . . .  | 78 |
| B.25 iMet Collection 2019 - FGVC6 . . . . .                           | 78 |
| B.25.1 Base de dados . . . . .  | 79 |
| B.25.2 Solução . . . . .  | 79 |
| B.25.3 Resultado . . . . .  | 80 |
| B.26 Northeastern SMILE Lab - Recognizing Faces in the Wild . . . . . | 80 |

---

|                                |    |
|--------------------------------|----|
| B.26.1 Base de dados . . . . . | 80 |
| B.26.2 Solução . . . . .       | 80 |
| B.26.3 Resultado . . . . .     | 81 |

# 1

## Introdução

Para muitos, I.A. (Inteligência Artificial) não é só mais um passo na evolução tecnológica, mas um dos passos mais importantes dela [1].

Segundo [2], a demanda por profissionais de I.A. dobrou durante aos anos de 2018 e 2019. E apesar da quantidade de profissionais estar crescendo ainda existe uma grande demanda do mercado, em 2019 existiam duas vagas para cada profissional da área. Cerca de 60% desses profissionais estavam empregados em empresas de tecnologia e finanças, em 2019.

Segundo o relatório anual de 2018 sobre Inteligência Artificial, feito pela Universidade de Stanford, as matrículas em cursos de Inteligência Artificial e Aprendizagem de Máquina vem crescendo ao redor de todo o mundo [3].

Assim, nota-se um fluxo crescente de profissionais e empresas que visam explorar o campo da Inteligência Artificial e Aprendizagem de Máquina. Por isso, todos os programadores(as) que desejam estar atualizados devem saber pelo menos o básico de modelos inteligentes e como utilizá-los.

Contudo, o mundo da Aprendizagem de Máquina, apesar de relativamente novo, já é bastante denso e pode causar espanto num momento inicial. Pois existem muitos modelos inteligentes e técnicas para lidar com dados. E por isso, programadores iniciantes ficam perdidos ao tentar resolver até mesmo problemas simples de Aprendizagem de Máquina, pois não existe um guia, ou uma sequência gradual, de técnicas a serem seguidas e modelos a serem experimentados.

O objetivo deste trabalho é ser um guia para quem busca alavancar seus conhecimentos em Visão Computacional através da Aprendizagem de Máquina. Para isto, através de um projeto de pesquisa (Apêndices A e B) do Centro de Inovações EDGE, foram catalogados problemas de visão computacional do *Kaggle*<sup>1</sup> e suas melhores soluções.

---

<sup>1</sup><https://www.kaggle.com/>

*Kaggle* é uma plataforma online que disponibiliza diversos problemas de Ciência de Dados. Muitos dos problemas são cadastrados por empresas que buscam as melhores soluções<sup>2</sup> para seus problemas. O ambiente da plataforma é gamificado e os problemas aparecem no formato de competições, onde vence aquele que alcançar a melhor pontuação em uma determinada métrica.

A pesquisa mencionada tinha como objetivo documentar as principais características das melhores soluções de cada competição e analisar quais as abordagens mais recorrentes. Assim, de posse desses dados, este trabalho irá fornecer uma sequência de passos básicos a serem seguidos, apontando quais técnicas utilizar, quando utilizá-las, e apresentar alguns modelos inteligentes já consolidados na literatura e como utilizá-los.

Por fim, serão resolvidas algumas competições do *Kaggle* seguindo os direcionamentos aqui presentes. A fim de demonstrar a praticidade do guia construído.

---

<sup>2</sup>É importante ressaltar que as métricas para classificar as soluções levam em consideração apenas a assertividade das predições, ignorando completamente eficiência energética, de memória, de tempo, etc.

# 2

## O Guia

A seguir será apresentado um conjunto de passos a serem seguidos por iniciantes que desejam construir suas primeiras soluções para problemas de Visão Computacional utilizando Aprendizagem de Máquina.

### 2.1 A construção do guia

As recomendações propostas neste trabalho foram obtidas a partir de análises feitas em cima da pesquisa presente nos Apêndices [A](#) e [B](#).

A pesquisa consistiu na coleta e anotação de dados referentes às características das competições de Visão Computacional presentes na plataforma *Kaggle*. Alguns dos dados anotados dizem respeito às características das bases de dados (como o tamanho da base, quantidade de classes, formato do dados, etc), características do problema (como o tipo do problema, métrica de avaliação, contexto), e características da melhor solução pública (como camadas utilizadas, técnicas de pré-processamento, técnicas de treinamento, etc). Vale ressaltar que todas as soluções analisadas utilizaram redes neurais, e por isso, muitas das técnicas aqui recomendadas combinarão melhor com modelos de redes neurais.

De posse dos dados anotados, foi possível observar que certos tipos de problemas costumam ser resolvidos com abordagens parecidas. E estas formam o guia proposto, onde para certos tipos de problemas é possível apontar caminhos a serem seguidos, pois já se mostraram serem abordagens recorrentes e de qualidade.

### 2.2 Guia

A seguir serão descritas e explicadas recomendações de abordagens recorrentes de cada etapa do desenvolvimento de uma solução inteligente.

### 2.2.1 Entendendo o problema

O primeiro passo é entender o problema enfrentado e suas características. Deve-se entender o tipo de dado a ser trabalhado e como ele está presente na base (se está estruturado de forma padronizada, se possui uma quantidade significativa de dados, etc). Este guia apresentará recomendações para bases em diversas configurações, e para problemas de quatro tipos diferentes:

- *Classifier* (Classificador) - este tipo de problema consiste em atribuir um rótulo à uma imagem ou fotografia inteira. Ver Apêndice A.
- *Object detection* (Detecção de objetos) - este tipo de problema consiste em atribuir um ou mais rótulos à uma imagem e mostrar a localização dos objetos na imagem, desenhando uma caixa delimitadora em volta de cada objeto. Ver Apêndice A.
- *Object segmentation* (Segmentação de objetos) - este tipo de problema tem uma finalidade parecida com a “detecção de objetos”, mas ao invés de detectar uma caixa delimitadora, deve-se detectar quais são os pixels específicos (conjunto chamado de “máscara”) que compõem cada objeto. Ver Apêndice A.
- *Feature pinning*: este tipo de problema consiste em apontar onde estão localizadas certas características da imagem. Ver Apêndice A.

### 2.2.2 Pré-processamento

Uma base de dados geralmente possui diversos tipos de problemas: ruídos nas imagens, imagens em tamanhos diferentes, imagens em preto e branco misturadas com coloridas, poucos dados, ocorrências desproporcionais de uma classe em relação a outras, etc.

Visto que a qualidade final do modelo está diretamente relacionada a qualidade da base de dados: é recomendável tentar mitigar bem estes problemas antes de seguir para os próximos passos.

Tendo em vista que as competições do *Kaggle* tem um foco maior na construção do modelo: a etapa de pré-processamento costuma ser bastante curta, pois os criadores das competições muitas vezes proveem bases de boa qualidade.

#### Limpeza da base

Apesar de não ser muito comum nos problemas observados, a base de dados pode conter imagens duplicadas ou com muito ruído<sup>1</sup>, o que dificulta o treinamento do modelo e por isso é recomendável lidar com esses tipos de situações.

---

<sup>1</sup>Entende-se por “ruído” a presença de informações que acabam por atrapalhar o aprendizado do modelo.

Uma forma de resolver ambos os problemas citados é executar uma análise manual em cima da base, a fim de encontrar e eliminar as imagens que se enquadrem nas características descritas. Porém este é um processo tedioso, desgastante e que pode ser muito demorado.

Felizmente, existem ferramentas capazes de identificar de forma automatizada imagens duplicadas na base, como o *ThumbsPlus* (disponível em [www.cerious.com](http://www.cerious.com)). Já o problema de identificar imagens desinteressantes<sup>2</sup> para a base, de forma automatizada, não é tão trivial.

### **Padronização dos dados**

Todas as soluções anotadas utilizam redes neurais, que por sua vez exigem que as imagens tenham as mesmas dimensões da entrada da rede neural. Por isso, recomenda-se que ao lidar com imagens de diferentes tamanhos seja feita uma padronização de suas dimensões, para que possam alimentar o modelo.

Também foi observado que 19 das 26 soluções utilizaram a técnica de normalização dos dados, onde cada pixel da imagem é convertido para um intervalo entre 0 e 1, e por isso também recomenda-se a utilização desta técnica [4].

### **Balanceamento da base**

Diz-se que uma base está desbalanceada quando esta apresenta uma diferença muito grande (ver Apêndice A) na quantidade de exemplos entre suas classes. Neste cenário, os modelos poderão ter problemas durante o treinamento pois seu poder de generalização ficará tendencioso para as classes de maior frequência.

Para resolver este problema existem duas abordagens: remover os excessos ou adicionar novos dados para as classes menos recorrentes. No caso da remoção de dados, esta pode ser feita de forma manual ou automatizada - para isso recomenda-se utilizar seleção aleatória das imagens a serem removidas. Já no caso de adição de dados, é desejável que sejam adicionadas imagens que não existem na base atual, mas caso este processo não seja viável o que pode ser feito é a geração aleatória de novas imagens aplicando transformações à imagens existentes, por exemplo: escolher X imagens aleatórias da base e aplicar uma rotação aleatória para cada uma, gerando, tecnicamente, imagens novas (processo de *Data Augmentation*).

### **2.2.3 Definição do modelo**

Atualmente, existem diversos tipos de redes neurais. Cada tipo possui vantagens, desvantagens e áreas de aplicação. Em problemas de Visão Computacional, os tipos de redes neurais que se destacam são as Redes Neurais Convolucionais. Isso ocorre porquê utilizando máscaras e filtros elas conseguem identificar padrões complexos e relevantes ao longo da imagem inteira. Pode-se

---

<sup>2</sup>Entende-se por “desinteressante” as imagens que apresentam muito ruído, classificação errada, ausência de informações para o treinamento, etc.

dizer que as redes convolucionais quebram a imagem de entrada em várias imagens menores e trabalha em cima destas aplicando as máscaras e filtros aprendidos (este comportamento é configurado através do parâmetro *stride*). Já as redes tradicionais, por assim dizer, tratam a imagem de entrada como uma peça única e buscam reconhecer os padrões considerando todos os pixels ao mesmo tempo.

## Camadas

Redes neurais possuem três tipos de camada: entrada, oculta e saída. A camada de entrada é responsável por receber os dados originais da imagem a ser processada e encaminhá-los à primeira camada oculta da rede. Uma rede neural pode ter diversas camadas ocultas, e cada uma delas possuem nós que recebem dados da camada anterior e encaminham dados para a camada seguinte após aplicar uma transformação (linear ou não). E este processo de encaminhar dados transformados de uma camada para a outra se propaga até a última camada da rede (saída), que é a resposta final do modelo para a respectiva entrada.

No contexto de redes convolucionais, algumas das camadas mais comuns observadas nas soluções do *Kaggle* foram (ver Tabela 2.1): *Conv2D*, *Dropout*, *MaxPooling2D*, *Batch Normalization*, *Dense*, *Flatten*, *Activation*, etc. Saber escolher e combinar cada tipo de camada é essencial para a construção e melhora de um modelo próprio satisfatório. Porém, como um dos objetivos deste trabalho é facilitar a vida de iniciantes, na seção seguinte serão recomendados modelos completos já consolidados na literatura.

### 2.2.4 Arquiteturas e modelos da literatura

As arquiteturas são formas de organizar as camadas e definir como elas interagem entre si, por exemplo *ResNet*. Já os modelos são construções específicas de uma determinada arquitetura, por exemplo *ResNet34*, *ResNet50*, *ResNet101* – geralmente varia-se apenas a quantidade camadas de um modelo para o outro.

#### Contextos e objetivos de arquiteturas

Muitas arquiteturas são construídas para contextos e objetivos específicos. O contexto de uma arquitetura diz respeito aos tipos de problemas e bases de dados que a arquitetura será utilizada, como: carros, pessoas, plantas, etc. Enquanto o objetivo da arquitetura é o propósito de criação, como: eficiência energética, eficiência computacional (tempo, memória, poder de processamento necessário), etc.

#### Arquiteturas

A Tabela 2.2 abaixo contém as arquiteturas recomendadas a serem utilizadas, junto de seus respectivos objetivos segundo seus artigos originais [5, 6, 7, 8, 9, 10, 11, 12].

| Camadas                | Quantidade |
|------------------------|------------|
| Conv2D                 | 15         |
| Dropout                | 12         |
| MaxPooling2D           | 12         |
| BatchNormalization     | 12         |
| Dense                  | 11         |
| Flatten                | 10         |
| Activation             | 5          |
| Lambda                 | 3          |
| AvgPool2d              | 2          |
| Concatenate            | 2          |
| DepthwiseConv2D        | 2          |
| GlobalMaxPooling2D     | 2          |
| Linear                 | 2          |
| ReLU                   | 2          |
| Reshape                | 2          |
| ZeroPadding2D          | 2          |
| Conv2DTranspose        | 2          |
| Add                    | 1          |
| GlobalAveragePooling2D | 1          |
| LeakyReLU              | 1          |
| MBCConv                | 1          |
| Multiply               | 1          |
| Sigmoid                | 1          |
| Subtract               | 1          |
| Swish                  | 1          |

Tabela 2.1: Quantidade de uso de cada camada nas soluções do *Kaggle*.

## Modelos

Identificada a arquitetura mais adequada para o problema, recomenda-se a utilização dos modelos presentes na Tabela 2.3 abaixo. Nela estão listados os modelos mais utilizados nas soluções observadas do *Kaggle*, assim como suas respectivas arquiteturas e frequências de uso.

### 2.2.5 Abordagens da literatura

Como mencionado na Subseção 2.2.3, para construir um modelo próprio é necessário entender como as camadas operam e interagem entre si para conseguir juntá-las de forma inteligente.

O que diferencia as arquiteturas citadas na seção anterior é justamente a forma inteligente com que elas combinam suas camadas. E para construir um modelo próprio é interessante conhecer pelo menos a ideia principal por trás de algumas arquiteturas consolidadas.

| Modelos         | Quantidade | Objetivo  |
|-----------------|------------|---|
| ResNet          | 7          | Utilizar blocos e atalhos para lidar com a dissipação do gradiente        |
| EfficientNet    | 5          | Escalabilidade eficiente  |
| Inception       | 3          | Remover ativações inúteis   |
| DenseNet        | 2          | Otimizar eficiência e acurácia através de uma rede profunda e com atalhos |
| ResNeXt         | 1          | Melhoras na ResNet  |
| InceptionResNet | 1          | Combinação de ideias da ResNet com Inception                              |
| VGG             | 1          | Rede profunda e com filtros bem pequenos                                  |
| Xception        | 1          | Melhoras na Inception   |

Tabela 2.2: Quantidade de uso de cada arquitetura nas soluções do *Kaggle*.

## ResNet

Quanto mais profunda é uma rede neural mais complexo é o seu processo de treinamento, pois a quantidade de parâmetros a serem ajustados aumenta com a profundidade da mesma. Além disso, redes profundas também apresentam um problema inerente à sua construção: o problema da dissipação do gradiente – ou *The Vanishing Gradient Problem*, em inglês. A presença deste problema significa que as camadas iniciais são praticamente negligenciadas durante o processo de treinamento, pois o ajuste de peso que chega a elas é muito pequeno – visto que o ajuste de peso sofre uma diminuição a cada “viagem” que faz ao longo da rede, partindo da camada de saída e indo até a camada de entrada.

O grande diferencial da *ResNet* é a forma como ela se propõe a resolver este problema: utilizando “atalhos” entre camadas. Um “atalho” funciona como uma camada com função identidade, pegando a saída de uma camada e passando-a diretamente para a entrada de outra, pulando as camadas que estiverem no caminho. Mais especificamente, o artigo original da *ResNet* define o chamado “bloco residual”, que consiste de duas camadas conectadas entre si e um atalho que liga a entrada do bloco à saída do mesmo [5], veja a Figura 2.1.

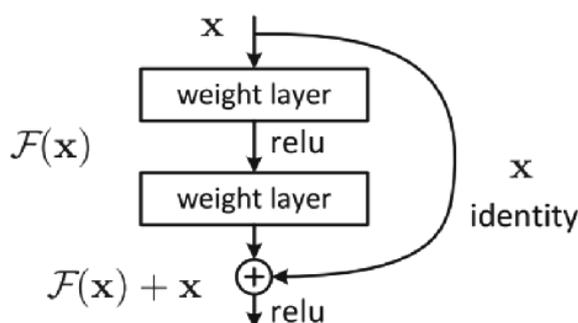


Figura 2.1: Bloco residual

Dessa forma, quando o ajuste de peso estiver navegando pela rede ele seguirá caminhos alternativos que encurtarão a distância entre as extremidades da rede, reduzindo a degradação

| Arquitetura     | Modelos           | Quantidade |
|-----------------|-------------------|------------|
| ResNet          | ResNet50          | 3          |
| ResNet          | ResNet34          | 3          |
| EfficientNet    | EfficientNetB3    | 2          |
| DenseNet        | DenseNet121       | 1          |
| InceptionResNet | InceptionResNetV2 | 1          |
| VGG             | VGG16             | 1          |
| ResNet          | ResNet101         | 1          |
| ResNeXt         | ResNeXt50         | 1          |
| Inception       | InceptionV4       | 1          |
| DenseNet        | DenseNet201       | 1          |
| Inception       | InceptionV3       | 1          |
| Inception       | InceptionResNetV2 | 1          |
| EfficientNet    | EfficientNetB5    | 1          |
| EfficientNet    | EfficientNetB2    | 1          |
| EfficientNet    | EfficientNet      | 1          |
| Xception        | Xception          | 1          |

Tabela 2.3: Quantidade de uso de cada modelo de cada arquitetura nas soluções do *Kaggle*.

do mesmo.

### EfficientNet

A principal ideia da *EfficientNet* foi encontrar uma forma de melhor escalar uma rede. No artigo são citadas três formas de escalar uma rede: aumentar a profundidade, aumentar a largura, aumentar a resolução (da imagem de entrada). É esperado que escalar um modelo em qualquer uma dessas três “dimensões” faça com que a acurácia aumente.

E, conforme o artigo original demonstra [6], ao escalar a rede em qualquer dimensão a acurácia da mesma realmente tende a aumentar. Além disso, foi demonstrado que o maior ganho de acurácia acontece ao escalar mais de uma dimensão ao mesmo tempo, cada uma sob uma constante específica, ver Imagem 2.2.

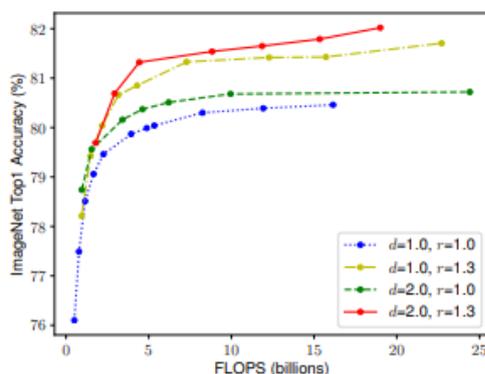


Figure 4. Scaling Network Width for Different Baseline Networks. Each dot in a line denotes a model with different width coefficient ( $w$ ). All baseline networks are from Table 1. The first baseline network ( $d=1.0, r=1.0$ ) has 18 convolutional layers with resolution  $224 \times 224$ , while the last baseline ( $d=2.0, r=1.3$ ) has 36 layers with resolution  $299 \times 299$ .

Figura 2.2: A linha vermelha é quando foram alteradas as 3 dimensões da *EfficientNet*

Com isso em mente, os autores desenvolveram um método de maximização para escalar o modelo de uma forma mais eficiente, buscando maximizar a acurácia e minimizar os requisitos computacionais através de uma *Grid Search*. Assim, foi possível escalar um modelo base de forma eficiente e comprovar a tese.

O modelo base da *EfficientNet* (chamado *EfficientNet-B0*) foi gerado a partir de uma busca descrita em [13], e trata-se basicamente de uma *MnasNet* modificada.

## Inception

A ideia por trás da *Inception* vem da observação de que maioria das ativações de uma rede profunda são descartáveis (muito próximas de zero ou redundantes), e então remover estas ativações levaria a um modelo mais eficiente [7].

Porém, simplesmente remover ativações degradaria a performance das multiplicações de matriz que a GPU executa, pois utilizaria matrizes esparsas, e isto não é muito otimizado na maioria dos *frameworks*. Então, foi criado um módulo *Inception*, que simula essa falta de ativações descartáveis (esparsamente densa), com o objetivo de encontrar objetos de diferentes escalas dentro da imagem. Isto é feito utilizando camadas densas, convoluções e *max-pooling*, como na Figura 2.3. Note como os tamanhos das camadas são pequenos ( $1 \times 1$ ,  $3 \times 3$ ,  $5 \times 5$ ) [7].

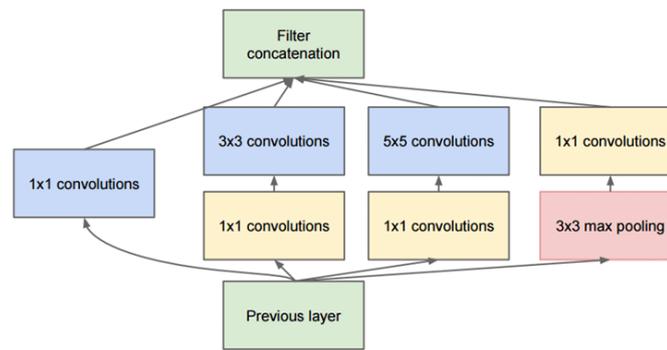


Figura 2.3: Módulo *Inception*

Uma outra ideia utilizada na *Inception* é utilizar um *Simple Global Average Pooling* no lugar de camadas totalmente conectadas (*Fully-Connected Layers*), reduzindo assim a quantidade de parâmetros da rede.

## 2.2.6 Treinamento

Após decidida a arquitetura do modelo é hora de pensar na etapa de “aprendizagem” do mesmo, a chamada **etapa de treinamento**. É aqui que as conexões do modelo serão ajustadas através de um processo que consiste em alimentar o modelo com dados de entrada e informar qual a saída esperada, para que o modelo possa ajustar suas conexões de forma a diminuir o erro da saída gerada para a saída esperada.

### *Data Augmentation*

Durante a pesquisa deste trabalho foi constatada uma grande utilização da estratégia de *Data Augmentation* (Aumento de Dados) nas soluções do Kaggle.

Esta técnica consiste em gerar novos dados através da aplicação de transformações (com um certo grau de aleatoriedade) nos dados originais, antes de iniciar o processo de treinamento.

Abaixo estão as Tabelas 2.4 e 2.5 que mostram a quantidade de vezes que esta técnica foi utilizada.

|                              |                  | Data Augmentation |       |
|------------------------------|------------------|-------------------|-------|
|                              |                  | True              | False |
| Enough data per class/target | Non-uniform data | 4                 | 3     |
| Enough data per class/target | Uniform Data     | 5                 | 2     |
| Few data per class/target    | Non-uniform data | 6                 | 0     |
| Few data per class/target    | Uniform Data     | 6                 | 0     |

Tabela 2.4: Quantidade de uso de *Data Augmentation* em bases uniformizadas, não uniformizadas, com dados suficientes por classe ou não.

|                     | Data Augmentation |       |
|---------------------|-------------------|-------|
|                     | True              | False |
| Classifier          | 17                | 2     |
| Object Segmentation | 3                 | 0     |
| Object Detection    | 1                 | 2     |
| Feature Pinning     | 0                 | 1     |

Tabela 2.5: Quantidade de uso de *Data Augmentation* por tipo de problema.

Assim, as seguintes recomendações quanto a utilização da técnica *Data Augmentation* podem ser feitas:

- **Utilizar** *Data Augmentation* quando a base de dados possui alguma classe com poucos dados (poucas ocorrências), segundo a Tabela 2.4;
- **Utilizar** *Data Augmentation* quando a base possui dados uniformes e uma quantidade boa de dados por classe;
- **Não utilizar** *Data Augmentation* quando os dados não são uniformes e a base possui uma boa quantidade de dados por classe;
- **Utilizar** *Data Augmentation* quando o problema for do tipo Classificador, segundo a Tabela 2.5.

Basicamente esta técnica é utilizada para melhorar a generalização do modelo. Pois ela pode ser utilizada para gerar “novos” dados a partir dos existentes, ou utilizada para simplesmente trazer uma certa “variação” aos dados bem uniformes da base.

### ***Transfer Learning***

*Transfer Learning* é outra técnica que foi observada em algumas soluções do *Kaggle*. O objetivo desta técnica é reaproveitar o “conhecimento” de um modelo previamente treinado. Através da reutilização dos pesos em um novo modelo e com isso diminuir o tempo necessário para treiná-lo.

Uma suposição muito comum ao utilizar *Transfer Learning* é que os dados de treinamento devem ser do mesmo universo/contexto dos dados futuros. Porém, em muitas aplicações reais essa suposição não se mantém [14]. Sendo possível alcançar bons resultados aplicando a técnica entre modelos de contextos diferentes.

Segundo os dados observados na pesquisa deste trabalho (ver Tabelas 2.6 e 2.7), não existe um cenário específico onde a utilização da técnica é essencial, indicando que sua utilização é arbitrária.

Caso o programador opte por utilizar a técnica em sua solução, recomenda-se utilizar a arquitetura *ResNet*, pois foi a arquitetura utilizada em maior frequência junto à técnica, ver Tabela 2.8.

|                     | Uses Transfer Learning |       |
|---------------------|------------------------|-------|
|                     | True                   | False |
| Classifier          | 11                     | 8     |
| Feature Pinning     | 0                      | 1     |
| Object Segmentation | 3                      | 0     |
| Object Detection    | 2                      | 1     |

Tabela 2.6: Quantidade de uso de *Transfer Learning* por tipo de problema.

|                              |                  |                    | Uses Transfer Learning |       |
|------------------------------|------------------|--------------------|------------------------|-------|
|                              |                  |                    | True                   | False |
| Enough data per class/target | Non-uniform data | Balanced Dataset   | 0                      | 0     |
|                              |                  | Unbalanced Dataset | 6                      | 1     |
|                              | Uniform Data     | Balanced Dataset   | 1                      | 2     |
|                              |                  | Unbalanced Dataset | 1                      | 3     |
| Few data per class/target    | Non-uniform data | Balanced Dataset   | 4                      | 1     |
|                              |                  | Unbalanced Dataset | 1                      | 0     |
|                              | Uniform Data     | Balanced Dataset   | 3                      | 1     |
|                              |                  | Unbalanced Dataset | 0                      | 2     |

Tabela 2.7: Quantidade de uso de *Transfer Learning* em bases com dados suficientes, uniformes e balanceados.

### Épocas de treinamento

A quantidade de “épocas” indica quantas vezes a base de treinamento será inteiramente passada ao modelo. Para decidir a quantidade de épocas ideal, deve-se levar em consideração diversos fatores, como quantidade de camadas ocultas, largura do modelo, tamanho da base, etc.

Como o objetivo deste trabalho é simplificar o processo de criação e treinamento de um modelo, a melhor recomendação que pode ser feita é executar o treinamento com várias épocas diferentes de forma crescente até que a função de perda da validação (ver Seção 2.2.7) pare de diminuir consideravelmente, ou seja, alcance o platô [15].

Vale ressaltar que treinar “infinitamente” não gerará um modelo perfeito, mas muito provavelmente um modelo muito ajustado à base de treinamento, ou seja, com baixo poder de generalização. É por isso que quando a função de perda alcançar o platô não há necessidade de prosseguir com o treinamento. Além disso, diminuir o *Learning Rate* (taxa de aprendizagem) também pode gerar melhores resultados.

### 2.2.7 Validação

A etapa de validação está intrinsecamente interligada à etapa de treinamento, pois ambas as etapas utilizam a base de dados anotada.

A diferença é que no treinamento o modelo usará as anotações para ajustar suas previsões

| Architecture    | Models            | Count |
|-----------------|-------------------|-------|
| ResNet          | ResNet50          | 3     |
| ResNet          | ResNet34          | 3     |
| EfficientNet    | EfficientNetB3    | 2     |
| DenseNet        | DenseNet121       | 1     |
| DenseNet        | DenseNet201       | 1     |
| EfficientNet    | EfficientNetB5    | 1     |
| EfficientNet    | EfficientNetB2    | 1     |
| Inception       | InceptionV3       | 1     |
| Inception       | InceptionV4       | 1     |
| InceptionResNet | InceptionResNetV2 | 1     |
| ResNet          | ResNet101         | 1     |
| ResNeXt         | ResNeXt50         | 1     |
| VGG             | VGG16             | 1     |
| Xception        | Xception          | 1     |

Tabela 2.8: Quantidade de uso de *Transfer Learning* em cada arquitetura e modelo.

futuras. Já na etapa de validação as anotações são usadas para analisar o poder de generalização do modelo.

Para fazer isso, é necessário dividir a base de treinamento em dois pedaços complementares, um destinado ao treinamento e outro à validação. Dessa forma, o modelo não poderá se especializar nos dados de validação e por isso, estes podem ser utilizados para medir a confiabilidade real do modelo.

A Tabela 2.9 mostra os métodos de validação observados durante a pesquisa deste trabalho. Cada método dita uma forma diferente de lidar com os dados de treinamento e validação, e o mais utilizado foi o método “padrão”, que consiste em fazer uma divisão simples dos dados em 2 grupos complementares fixos: um para treinamento e outro para validação. Neste método, cada grupo é populado com elementos aleatórios da base de dados e seus tamanhos são fixos, sendo definidos pelo programador antes de iniciar a etapa de treinamento.

Assim, para a validação do modelo recomenda-se a utilização do método “padrão”, com o tamanho da base de validação contendo até 20% da base de dados anotados, ver Tabela 2.10.

### Função de perda

A função de perda serve para calcular o erro das predições do modelo em comparação às respostas esperadas. A recomendação que pode ser feita mediante as soluções observadas durante a pesquisa é utilizar funções de perda do tipo *Cross-Entropy* em modelos classificadores, ver Tabela 2.11.

| Validation Process  | Count |
|---|-------|
| Default   | 8     |
| ImageDataGenerator  | 2     |
| Combination of Models   | 1     |
| MultilabelStratifiedShuffleSplit; Oversampling                                | 1     |
| Trained 3 times with increasing batch size                                    | 1     |
| Trained 3 models and answer is an weighted average                            | 1     |
| StratifiedKFold (10)  | 1     |
| Stratified Shuffle Split  | 1     |
| Pseudo-Labeling   | 1     |
| None  | 1     |
| Multi Output CNN;   | 1     |
| Data Generator; K-Fold (3)  | 1     |
| Image Pairs (50% matching whales; 50% different whales); L2 regularization    | 1     |
| Gradient Descent for all samples  | 1     |
| Fit One Cycle   | 1     |
| Fit Cycles; Fit with and without augmentation and resize image between cycles | 1     |
| Dropped all images without ship   | 1     |
| Twin Dataset; Data Augmentation; Variable Learning Rate                       | 1     |

Tabela 2.9: Frequência de uso de cada tamanho de conjunto de avaliação

| Tamanho do conjunto de validação      | Frequência |
|---------------------------------------|------------|
| Até 5% da base de treinamento         | 2          |
| Exatamente 10% da base de treinamento | 8          |
| Até 10% da base de treinamento        | 13         |
| Até 20% da base de treinamento        | 16         |
| Até 25% da base de treinamento        | 17         |
| Até 33% da base de treinamento        | 18         |
| 10.000 imagens de cada base de dados  | 1          |
| Lotes de tamanhos variados            | 1          |
| Nenhum                                | 4          |
| Desconhecido                          | 2          |

Tabela 2.10: Frequência de uso de cada tamanho de conjunto de avaliação

| Função de perda                             | Tipo de Problema |                    |                        |                     |
|---|------------------|--------------------|------------------------|---------------------|
|   | Classifier       | Feature<br>Pinning | Object<br>Segmentation | Object<br>Detection |
| Cross-entropy                               | 2                | 0                  | 0                      | 0                   |
| Binary Cross-entropy                        | 7                | 0                  | 0                      | 0                   |
| Mean Squared Error                          | 0                | 1                  | 0                      | 0                   |
| Unknown                                     | 5                | 0                  | 0                      | 1                   |
| Jaccard Loss                                | 0                | 0                  | 1                      | 0                   |
| Custom                                      | 1                | 0                  | 0                      | 0                   |
| Categorical Cross-entropy                   | 4                | 0                  | 0                      | 0                   |
| Binary Cross-entropy + Dice Loss            | 0                | 0                  | 1                      | 1                   |
| None  | 0                | 0                  | 1                      | 0                   |
| Weighted Binary Cross-entropy and Dice Loss | 0                | 0                  | 0                      | 1                   |

Tabela 2.11: Frequência de uso de cada tamanho de conjunto de avaliação

# 3

## Validando O Guia

Neste capítulo serão resolvidas três competições do Kaggle utilizando as recomendações feitas anteriormente a fim de demonstrar a utilidade prática deste trabalho. Cada competição tem características distintas entre si e por consequência apresentam desafios diferentes. As descrições completas de cada competição estão presentes no Apêndice B.

O objetivo final é demonstrar que as recomendações deste trabalho atendem bem as mais diversas situações.

### 3.1 *Digit Recognizer*

Esta competição apresenta uma base de dados balanceada, com imagens muito uniformizadas e em grande quantidade.

As características da solução construída para esta competição segundo as recomendações deste trabalho foram as seguintes: não foi preciso realizar nenhum tipo de pré-processamento, pois a base de dados foi muito bem construída; o modelo escolhido foi o *ResNet50*, arquitetura *ResNet*; a etapa de treinamento consistiu em uma simples divisão da base em treinamento (90% dos dados) e validação (10% dos dados), e com apenas 10 épocas já observou-se um bom resultado segundo a função de perda *Categorical Cross-Entropy*.

Como resultado final, a solução construída alcançou mais de 98% de acurácia. O código da solução encontra-se disponível em [Digit Recognizer - TCC UFAL](#).

### 3.2 *Dogs vs. Cats Redux: Kernels Edition*

Diferente da competição anterior, a base desta, apesar de grande, não possui dados nem um pouco uniformes (tanto em dimensões quanto em conteúdo).

Para a solução construída, na etapa de pré-processamento todas as imagens foram redimensionadas sem a preocupação de manter a razão de aspecto e também foram utilizadas as técnicas de normalização e *Data Augmentation* aplicando rotação (45° em ambos sentidos) e espelhamento horizontal às imagens.

O modelo escolhido foi novamente o *ResNet50*. Na etapa de treinamento foi realizada uma simples divisão da base em 90% para treinamento e 10% para validação. Empiricamente foi observado que com 10 épocas o modelo já convergia para um valor de perda estável, segundo a função de perda *Binary Cross-Entropy*.

Com isto, a solução construída obteve 5.11731 pontos nos placares público e privado, que equivale ao 1.235° lugar geral.

### 3.3 1ª Competição de IA EDGE-Huawei

Esta competição não está descrita nos anexos, já que foi realizada após a pesquisa deste trabalho. O objetivo desta competição é encontrar uma caixa delimitadora (*bounding box*) para 4 tipos de dispositivos eletrônicos, ou seja, trata-se de uma competição de *Object Detection* (detecção de objeto). O maior desafio dessa competição é treinar o modelo utilizando uma base extremamente pequena, contendo 135 imagens. Contudo, as imagens que compõem a base são bem uniformes.

Para a solução construída, na etapa de pré-processamento foram removidas as imagens que continham mais de 1 dispositivo, pois poderia atrapalhar o treinamento do modelo. Além disso, também foi utilizada a técnica de *Data Augmentation* para gerar imagens espelhadas, vertical e horizontalmente, e ampliadas em 20%. O que gerou uma base com tamanho final de 912 imagens.

O modelo escolhido foi, mais uma vez, o *ResNet50*. Na etapa de treinamento a base foi dividida, novamente, em 90% para treinamento e 10% para validação. Através de observações empíricas, com 300 épocas de treinamento o modelo começou a convergir para um valor de perda estável, segundo a função Erro Médio Quadrado (*Mean Squared Error - MSE*).

Finalmente, a solução construída obteve 710.21039 pontos no placar privado, que equivale ao segundo lugar geral<sup>1</sup>.

---

<sup>1</sup>Esta solução foi submetida depois do fim da competição. Por isso não consta na listagem pública



## Conclusão

O presente trabalho abordou diferentes etapas da construção de soluções para diversos problemas de aprendizagem de máquina voltados à visão computacional. O objetivo era estruturar uma espécie de guia para iniciantes na área, de forma a compilar recomendações do que fazer em cada etapa, segundo análises das melhores soluções de problemas presentes na plataforma *Kaggle*.

Para proporcionar o desenvolvimento desta monografia, foi realizado um trabalho de pesquisa para documentar as melhores soluções das competições de visão computacional presentes na plataforma *Kaggle*. Esta pesquisa consistiu em coletar informações a respeito das bases de dados, dos problemas apresentados e seus contextos, e das soluções construídas. Após a anotação dos dados foram aplicados alguns filtros para manter apenas soluções que atendessem certos critérios de qualidade.

De posse das soluções filtradas e de suas características anotadas de forma tabular, foram feitas análises estatísticas para saber quais as abordagens mais recorrentes e sob quais condições estas poderiam ser recomendadas para a construção de uma nova solução.

Após definidas as recomendações e seus cenários de uso, estas foram postas em prática a fim de validar o valor prático do "guia" construído. Para isso, foram selecionadas três competições do *Kaggle* com características bem distintas entre si. E em todas elas as soluções construídas seguindo as recomendações obtiveram resultados excelentes sem muito esforço, atestando assim a qualidade deste trabalho.

### 4.1 Trabalhos futuros

Para um trabalho futuro, deseja-se aplicar o método de pesquisa utilizado ao longo desta monografia para gerar recomendações de soluções em outros campos da aprendizagem de máquina.

Além disso, seria interessante encontrar competições onde as recomendações desse trabalho não alcançam bons resultados e entender a razão disso, afim de melhorar as recomendações aqui presentes.

## Referências bibliográficas

- [1] Lauren Goode. Google ceo sundar pichai compares impact of ai to electricity and fire, 2018.
- [2] David Kelnar. The state of ai: Divergence 2019. Technical report, 2019.
- [3] Erik Brynjolfsson Jack Clark James Manyika Juan Carlos Niebles Terah Lyons John Etchemendy Barbara Grosz Yoav Shoham, Raymond Perrault and Zoe Bauer. The ai index 2018 annual report. Technical report, Stanford University, Stanford, California, 2018.
- [4] Warren S. Sarle. comp.ai.neural-nets faq, part 2 of 7: Learning, 2002.
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 06 2016.
- [6] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.
- [7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- [8] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [9] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.
- [10] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

- [11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.
- [12] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.
- [13] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *CoRR*, abs/1807.11626, 2018.
- [14] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 10, pages 1345–1345, 2010.
- [15] Yuki Yoshida and Masato Okada. Data-dependence of plateau phenomenon in learning with neural network—statistical mechanical analysis. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124013, dec 2020.
- [16] Antônio Houaiss. *Dicionário Houaiss da Língua Portuguesa*, volume ? Editora Objetiva, 2001.
- [17] Jason Brownlee. 9 applications of deep learning for computer vision, 2019.
- [18] Pete Warden. How many images do you need to train a neural network?, 2017.
- [19] Dorian Pyle. *Data Preparation for Data Mining*, volume ? Morgan Kaufmann Publishers, 1999.
- [20] Hervé Abdi. Normalizing data, 2010.
- [21] Urvashi Jaitley. Why data normalization is necessary for machine learning models, 2018.
- [22] Richard Liaw Daniel Ho, Eric Liang. 1000x faster data augmentation, 2019.
- [23] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, pages 2–2, 2018.
- [24] Peter König Alex Hernández-García. Data augmentation instead of explicit regularization. *arXivpreprint arXiv:1806.03852*, pages 1–2, 2018.
- [25] Miguel Cárdenas-Monte. Sobreajuste - overfitting. Course notes, October 2015. Acesso em 05 jan. 2020.
- [26] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutionalneural networks. *Neural Information Processing Systems (NIPS)*, pages 5–5, 2012.
- [27] Algorithmia. Introduction to loss functions, 2018.

- 
- [28] François Chollet et al. Keras. <https://keras.io>, 2015.
- [29] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.
- [30] Efren Lopez Jimenez, Juan Vasquez-Gomez, Miguel Sanchez Acevedo, Juan Herrera Lozada, and Valeria Uriarte-Arcia. Columnar cactus recognition in aerial images using a deep learning approach. 05 2019.
- [31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [32] Paul-Louis Pröve. Squeeze-and-excitation networks, 2017.
- [33] Vinay Uday Prabhu. Kannada-mnist: A new handwritten digits dataset for the kannada language. *arXiv preprint arXiv:1908.01242*, 2019.
- [34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.
- [35] Joseph P Robinson, Ming Shao, Yue Wu, Hongfu Liu, Timothy Gillis, and Yun Fu. Visual kinship recognition of families in the wild. 2018.

# Apêndice A

## Coleta e Anotação de Dados<sup>1</sup>

### A.1 A plataforma Kaggle

Kaggle é uma plataforma online criada em 2010 por Anthony Goldbloom e Ben Hamner, e vendida em 2017 para a Google. Trata-se de uma plataforma que hospeda competições de *Data Science* (Ciência de Dados). Estas podem ser patrocinadas (onde uma empresa oferece um prêmio em dinheiro para as melhores soluções), ou simplesmente para conhecimento e aprendizagem. Geralmente as competições de conhecimento não têm prazo para envio de novas submissões, ao contrário das patrocinadas, que aceitam submissões somente até a data limite.

Cada competição contém uma descrição do problema e sua motivação, um conjunto de dados (com textos, imagens, áudios, etc), um ranking, e uma seção para discutir e compartilhar soluções, também denominadas como *notebooks*.

Sobre o ranking, também chamado de *leaderboard*, vale ressaltar que existem dois tipos de pontuação: pública e privada. Geralmente para calculá-las utilizam-se partes complementares da base de teste, por exemplo, pode-se usar 20% para calcular a pontuação pública e os outros 80% para a pontuação privada. Sendo esta última revelada apenas ao final da competição. No entanto, existem competições que não possuem o *leaderboard* privado, mas todas possuem o público.

### A.2 A coleta de dados

O primeiro passo para a coleta foi fazer uma filtragem, dentro da plataforma Kaggle, em busca de competições de visão computacional. Para filtrar as competições desse tipo foi utilizada a ferramenta de busca da própria plataforma, utilizando o termo “*image*”. Com isso, essa filtragem retornou 76 competições.

Em seguida, foi construída uma tabela para documentar o nome de cada competição, a métrica de avaliação utilizada por cada competição, a pontuação alcançada pela equipe em

---

<sup>1</sup>Autores: Daniel Humberto Cavalcante Vassalo, Nelson Gomes Neto

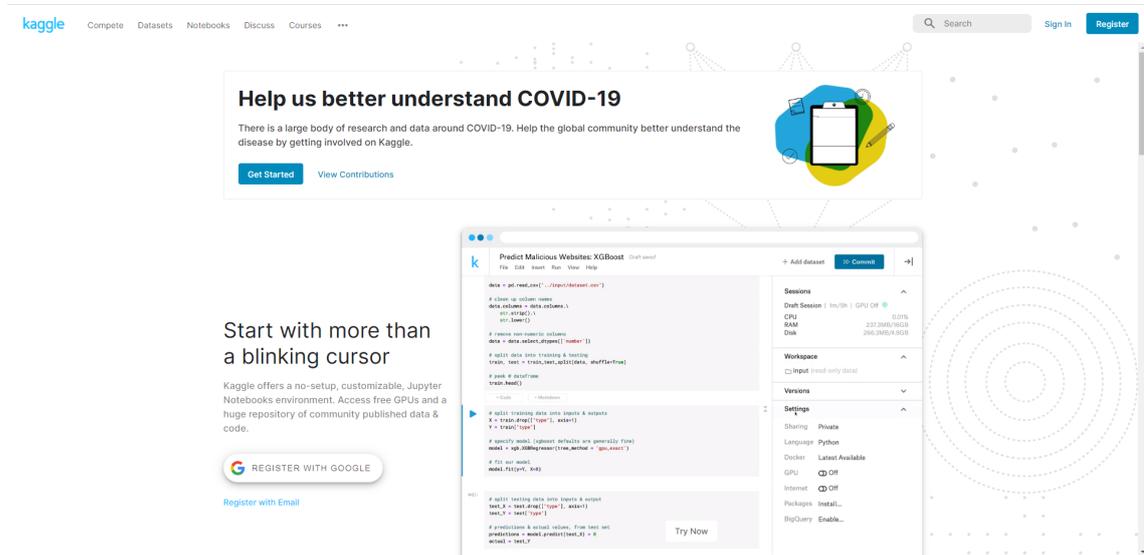


Figura A.1: Página inicial do Kaggle

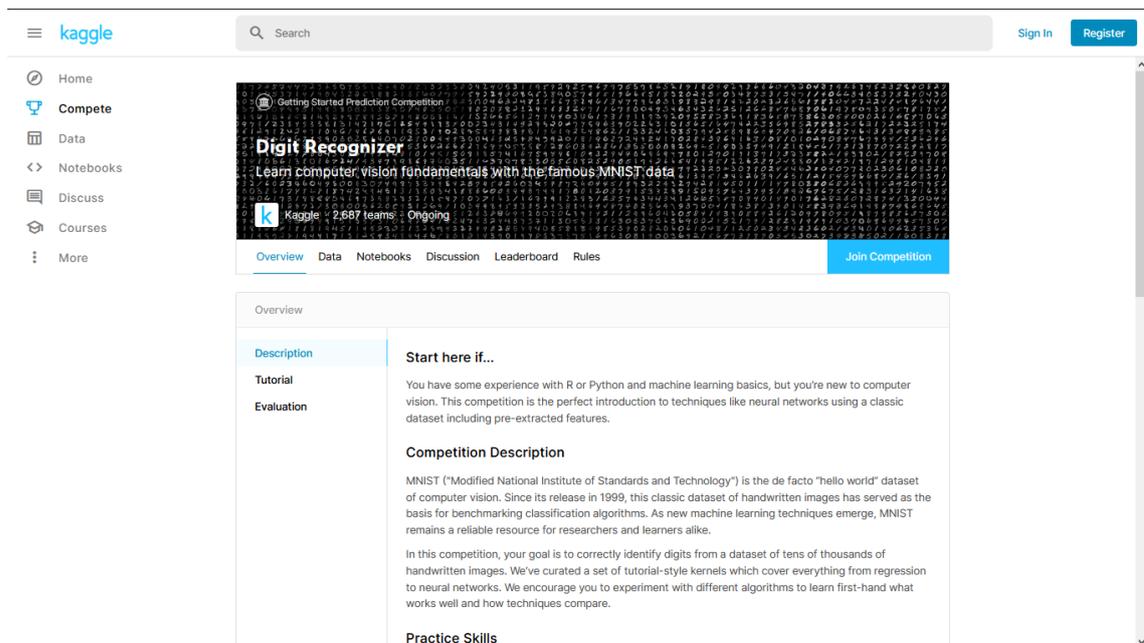


Figura A.2: Descrição de uma competição do Kaggle

The screenshot shows the Kaggle interface for the 'Digit Recognizer' competition. The left sidebar contains navigation options: Home, Compete, Data, Notebooks, Discuss, Courses, and More. The main content area features a search bar, 'Sign In', and 'Register' buttons. Below the competition banner, there are tabs for Overview, Data, Notebooks, Discussion, Leaderboard, and Rules, with a 'Join Competition' button. The 'Data Description' section provides the following information:

The data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

Each pixel column in the training set has a name like pixel $x$ , where  $x$  is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed  $x$  as  $x = i * 28 + j$ , where  $i$  and  $j$  are integers between 0 and 27, inclusive. Then pixel $x$  is located on row  $i$  and column  $j$  of a  $28 \times 28$  matrix, (indexing by zero).

Below the description, there is a 'Data (15 MB)' section with an API link and a 'Download All' button. A table lists the data sources:

| Data Sources                   | About this file    | Columns |
|--------------------------------|--------------------|---------|
| sample_submission... 2 columns | No description yet | ImageId |
| test.csv 784 columns           |                    | # Label |
| train.csv 785 columns          |                    |         |

Figura A.3: Descrição dos dados de uma competição do Kaggle

The screenshot shows the Kaggle interface for the 'Digit Recognizer' competition, specifically the 'Notebooks' tab. The left sidebar is the same as in Figure A.3. The main content area shows a list of notebooks sorted by 'Hotness'. The list includes the following entries:

- Applied Machine Learning**: 507 votes, 12h ago, GPU, tutorial, beginner, starter code, tpu, gpu. (151 comments)
- Digits Recognizer using keras - 0.974 Accuracy**: 3 votes, 7h ago, 0.9741 accuracy, tutorial, beginner, deep learning, neural networks, starter code. (1 comment)
- MNIST Digit Recognition**: 1 vote, 2h ago, GPU. (0 comments)
- Check Poor Predictions Manually [99.9%]**: 3 votes, 8h ago, beginner, deep learning. (3 comments)
- Starters Guide: Convolutional XGBOOST LB→99.5**: 4 votes, 8h ago, GPU, 0.99571 accuracy, GPU. (2 comments)
- Introduction to CNN Keras - 0.997 (top 6%)**: 4672 votes, 3y ago, beginner, cnn. (742 comments)

Figura A.4: Notebooks de uma competição do Kaggle

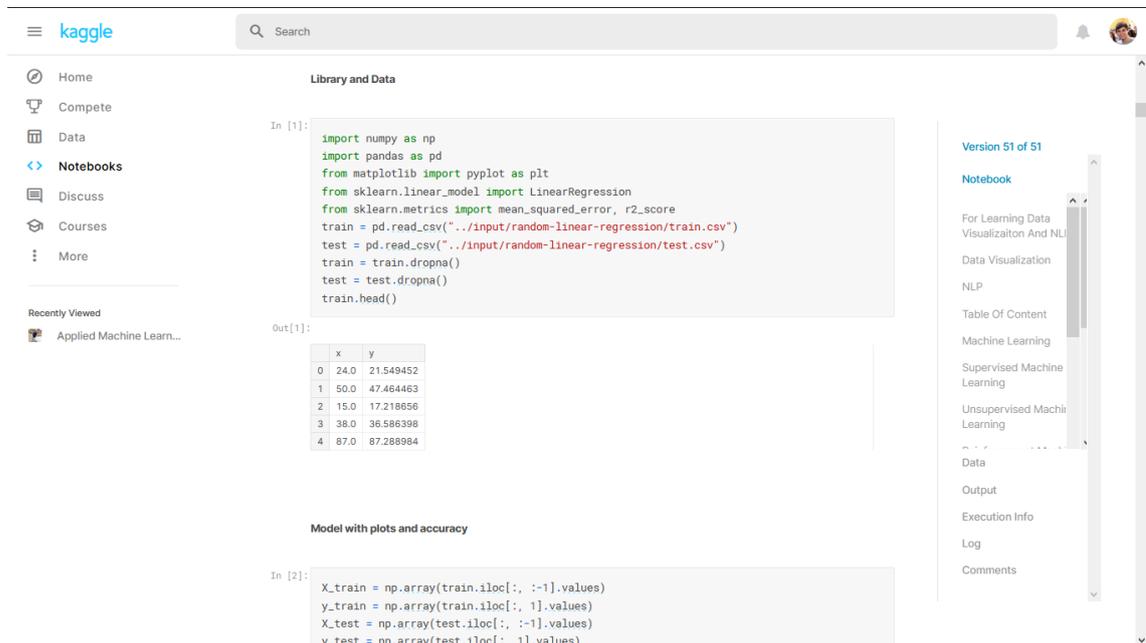


Figura A.5: Trechos de código dentro de um *notebook*

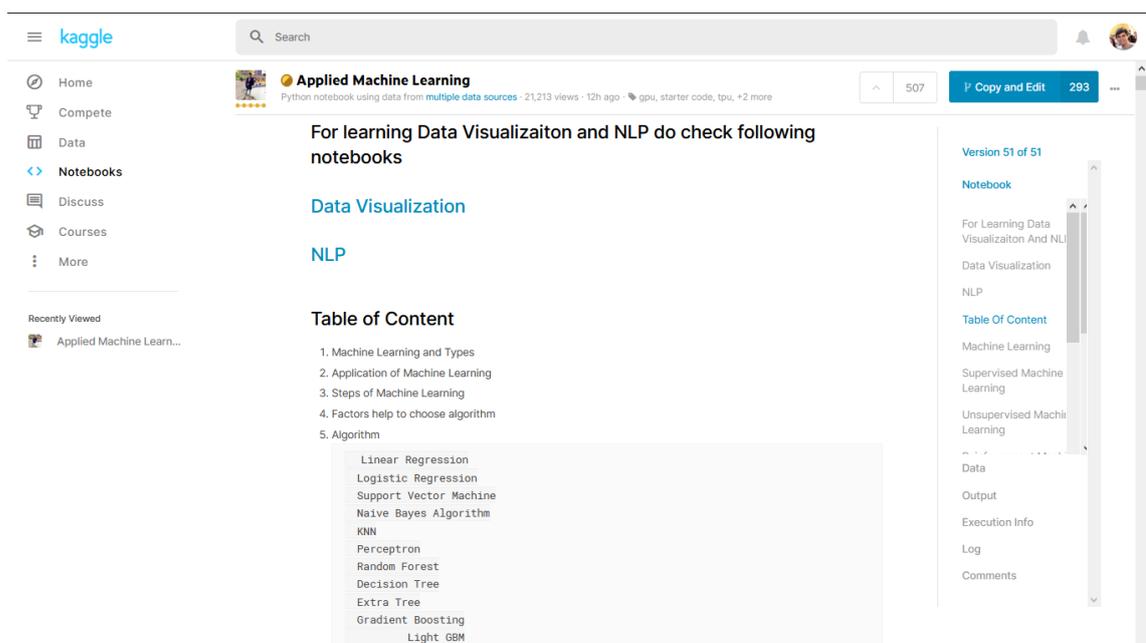


Figura A.6: Texto rico dentro de um *notebook*

Tabela A.1: Placares público e privado

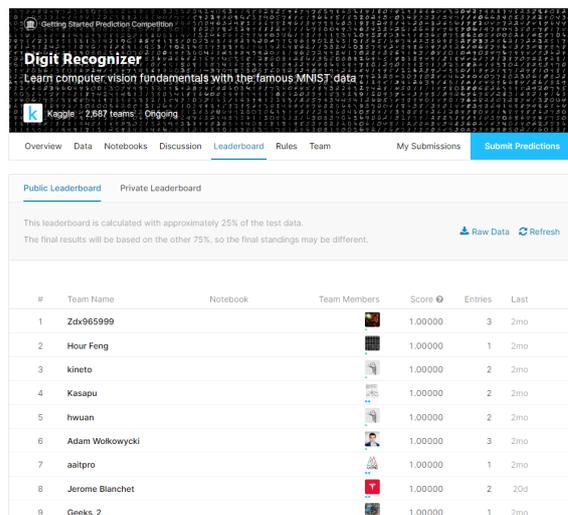


Figura A.7: Placar público

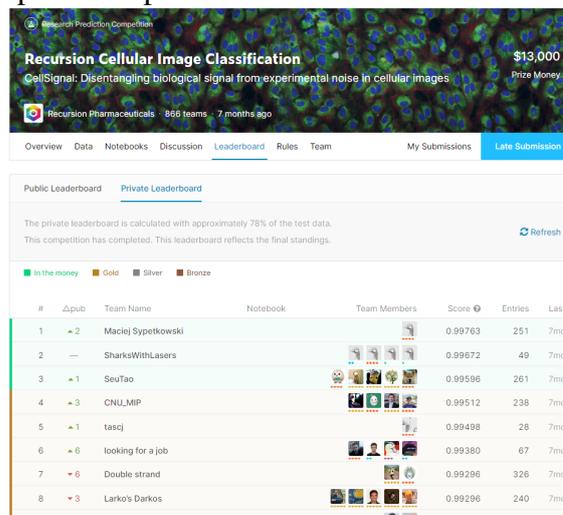


Figura A.8: Placar privado

primeiro lugar do ranking privado, a pontuação alcançada por essa equipe no ranking público (ou apenas a pontuação do primeiro lugar no ranking público, caso a competição não tenha ranking privado), e por fim as pontuações (pública e privada) alcançadas pelo melhor notebook publicado. Essa documentação foi feita manualmente para todas as 76 competições, ver tabela A.2.

| Competição   | Melhor Classificado | Melhor Notebook | Elegível   |
|--|---------------------|-----------------|------------|
| Quick, Draw! Doodle Recognition Challenge              | 0,955               | 0,923           | Verdadeiro |
| Avito Demand Prediction Challenge                      | 0,215               | 0,224           | Verdadeiro |
| Ultrasound Nerve Segmentation                          | 0,732               | 0,666           | Verdadeiro |
| Denoising Dirty Documents                              | 0,004               | 0,088           | Verdadeiro |
| RSNA Intracranial Hemorrhage Detection                 | 0,044               | 0,070           | Verdadeiro |
| Northeastern SMILE Lab - Recognizing Faces in the Wild | 0,923               | 0,902           | Verdadeiro |
| TGS Salt Identification Challenge                      | 0,896               | 0,854           | Verdadeiro |
| iMet Collection 2019 - FGVC6                           | 0,672               | 0,641           | Verdadeiro |
| Severstal: Steel Defect Detection                      | 0,909               | 0,910           | Verdadeiro |
| Humpback Whale Identification                          | 0,973               | 0,935           | Verdadeiro |
| Leaf Classification                                    | 0,000               | 0,006           | Verdadeiro |
| iMaterialist Challenge (Furniture) at FGVC5            | 0,120               | 0,974           | Verdadeiro |
| Humpback Whale Identification Challenge                | 0,786               | 0,786           | Verdadeiro |
| State Farm Distracted Driver Detection                 | 0,087               | 0,278           | Verdadeiro |
| Statoil/C-CORE Iceberg Classifier Challenge            | 0,082               | 0,137           | Verdadeiro |

|   |        |        |            |
|---|--------|--------|------------|
| APTOS 2019 Blindness Detection                          | 0,936  | 0,831  | Verdadeiro |
| SIIM-ACR Pneumothorax Segmentation                      | 0,868  | 0,825  | Verdadeiro |
| Airbus Ship Detection Challenge                         | 0,854  | 0,889  | Verdadeiro |
| Dogs vs. Cats Redux: Kernels Edition                    | 0,033  | 0,047  | Verdadeiro |
| Invasive Species Monitoring                             | 0,998  | 0,993  | Verdadeiro |
| Plant Seedlings Classification                          | 1,000  | 0,981  | Verdadeiro |
| Planet: Understanding the Amazon from Space             | 0,933  | 0,929  | Verdadeiro |
| Dog Breed Identification                                | 0,000  | 0,348  | Verdadeiro |
| Bengali.AI Handwritten Grapheme Classification          | 0,988  | 0,966  | Verdadeiro |
| Facial Keypoints Detection                              | 0,128  | 1,985  | Verdadeiro |
| Recursion Cellular Image Classification                 | 0,998  | 0,976  | Verdadeiro |
| Kannada MNIST   | 0,996  | 0,991  | Verdadeiro |
| Digit Recognizer  | 1,000  | 0,971  | Verdadeiro |
| Understanding Clouds from Satellite Images              | 0,672  | 0,646  | Verdadeiro |
| Aerial Cactus Identification                            | 1,000  | 1,000  | Verdadeiro |
| iMaterialist Challenge at FGVC 2017                     | 0,301  | -      | Falso      |
| Painter by Numbers                                      | 0,929  | 0,500  | Falso      |
| Google Cloud & YouTube-8M Video Understanding Challenge | 0,850  | 0,042  | Falso      |
| iNaturalist Challenge at FGVC 2017                      | 0,049  | -      | Falso      |
| First Steps With Julia                                  | 1,000  | -      | Falso      |
| NIPS 2017: Non-targeted Adversarial Attack              | 0,782  | -      | Falso      |
| Passenger Screening Algorithm Challenge                 | 0,024  | -      | Falso      |
| Intel & MobileODT Cervical Cancer Screening             | 0,770  | -      | Falso      |
| Yelp Restaurant Photo Classification                    | 0,832  | 0,640  | Falso      |
| Kuzushiji Recognition                                   | 0,950  | 0,564  | Falso      |
| Second Annual Data Science Bowl                         | 0,009  | -      | Falso      |
| The Nature Conservancy Fisheries Monitoring             | 1,062  | 2,208  | Falso      |
| Data Science Bowl 2017                                  | 0,400  | -      | Falso      |
| RSNA Pneumonia Detection Challenge                      | 0,255  | 0,199  | Falso      |
| NOAA Fisheries Steller Sea Lion Population Count        | 10,856 | 25,595 | Falso      |
| Lyft 3D Object Detection for Autonomous Vehicles        | 0,216  | 0,000  | Falso      |
| Right Whale Recognition                                 | 0,596  | -      | Falso      |
| Diabetic Retinopathy Detection                          | 0,850  | -      | Falso      |
| ImageNet Object Localization                            | 0,000  | -      | Falso      |
| iWildCam 2019 - FGVC6                                   | 0,399  | 0,163  | Falso      |
| Inclusive Images Challenge                              | 0,392  | 0,262  | Falso      |

|  |        |       |       |
|--|--------|-------|-------|
| Carvana Image Masking Challenge  | 0,997  | 0,823 | Falso |
| Cdiscount's Image Classification Challenge                                     | 0,796  | 0,167 | Falso |
| Draper Satellite Image Chronology  | 1,000  | 0,214 | Falso |
| Peking University/Baidu - Autonomous Driving                                   | 0,140  | 0,050 | Falso |
| Generative Dog Images  | 55,421 | 5,175 | Falso |
| Open Images 2019 - Object Detection  | 0,659  | 0,425 | Falso |
| Dstl Satellite Imagery Feature Detection                                       | 0,493  | 0,072 | Falso |
| Google AI Open Images - Object Detection Track                                 | 0,587  | 0,016 | Falso |
| Open Images 2019 - Visual Relationship   | 0,408  | 0,003 | Falso |
| Open Images 2019 - Instance Segmentation                                       | 0,526  | 0,031 | Falso |
| CIFAR-10 - Object Recognition in Images  | 0,955  | -     | Falso |
| Google AI Open Images - Visual Relationship Track                              | 0,285  | 0,003 | Falso |
| Google Landmark Recognition Challenge  | 0,304  | -     | Falso |
| iMaterialist Challenge (Fashion) at FGVC5                                      | 0,725  | -     | Falso |
| IEEE's Signal Processing Society - Camera Model Identification                 | 0,990  | 0,174 | Falso |
| Challenges in Representation Learning: Facial Expression Recognition Challenge | 0,712  | -     | Falso |
| CVPR 2018 WAD Video Segmentation Challenge                                     | 0,340  | 0,001 | Falso |
| NIPS 2017: Defense Against Adversarial Attack                                  | 0,953  | -     | Falso |
| Google Landmark Retrieval Challenge  | 0,627  | -     | Falso |
| Google Landmark Retrieval 2019   | 0,372  | -     | Falso |
| NIPS 2017: Targeted Adversarial Attack   | 0,402  | -     | Falso |
| PetFinder.my Adoption Prediction   | 0,453  | 0,446 | Falso |
| Human Protein Atlas Image Classification                                       | 0,594  | 0,456 | Falso |
| National Data Science Bowl   | 0,566  | -     | Falso |
| Data Science for Good: City of Los Angeles                                     | -      | -     | Falso |

Tabela A.2: Todas as competições de visão computacional do Kaggle, tabela simplificada

Após isso, uma filtragem manual foi feita para selecionar somente as competições que possuísem algum *notebook* com submissão avaliada, pois sem uma avaliação não seria possível classificar objetivamente a qualidade dos *notebooks*. Assim, essa filtragem resultou em 55 competições, que foram documentadas numa tabela, anotando-se o tipo de métrica de cada competição e as pontuações públicas e privadas de seus respectivos melhores notebooks.

Na etapa seguinte foi feita uma filtragem automatizada na tabela mencionada, onde as competições eram selecionadas somente se atendessem certos critérios de seleção. Esses critérios

serão descritos mais à frente, e eram diferentes para cada tipo de métrica. Pois, todas as métricas utilizadas nas competições podem ser divididas em dois tipos: “minimização”, quanto menor a pontuação melhor é a submissão; e “maximização”, quanto maior a pontuação melhor é a submissão.

Diante disso, foram elaborados critérios de seleção diferentes para cada tipo de métrica. Para as métricas de minimização, a melhor submissão do ranking (privado ou público) deve ter uma pontuação inferior ou igual a 1, e a pontuação do melhor notebook deve estar a uma distância de no máximo 2 pontos a mais da melhor submissão do ranking. Já para as métricas de maximização, a melhor submissão do ranking deve ter uma pontuação superior ou igual a 0,65, e a pontuação do melhor notebook deve estar a uma distância de no máximo 0,15 ponto a menos. Esses valores foram escolhidos visando eliminar as competições com resultados ruins. No caso das métricas de maximização o objetivo foi selecionar apenas as soluções que tivessem desempenho melhor do que “jogar uma moeda” para a tomada de decisões. Já para as métricas de minimização foi um pouco mais difícil definir um objetivo homogêneo para todas as métricas, pois suas funções decrescem em ritmos diferentes entre si, por isso os valores de corte deste tipo de métrica foram escolhidos de modo arbitrário. Vale ressaltar que as submissões do ranking e do notebook foram comparadas sob o mesmo tipo de pontuação – ou ambas sob a pontuação privada, ou ambas sob a pontuação pública (na ausência de uma pontuação privada). Ao final dessa etapa, a filtragem resultou em 30 competições, ver Tabela A.2.

De posse dessas 30 competições, foi feita manualmente uma anotação detalhada de cada competição e seu melhor notebook (solução pública), a fim de sintetizar as características mais importantes do problema apresentado na competição, da base de dados do problema e da solução proposta. No entanto, durante esse processo de análise alguns notebooks com a melhor pontuação tiveram que ser descartados, devido a falta de informações e utilização de modelos que não manipulavam imagens, resultando, por fim, num total de 26 competições válidas. As características/atributos anotados totalizam 28 colunas na tabela final<sup>2</sup>. Deixando a mesma num formato de 26 linhas por 28 colunas.

### A.3 Atributos

Atributo é “um aspecto qualitativo ou quantitativo que distingue um integrante de um conjunto observado” [16]. Ou seja, os atributos serão utilizados para descrever, através de uma tabela, as características de cada competição. Levando isso em consideração e trazendo para o contexto deste trabalho, os 28 atributos escolhidos podem ser divididos em 3 classes diferentes: atributos do problema; atributos da base de dados; e atributos da solução. A seguir serão descritos quais foram os 28 atributos escolhidos para fazer as anotações.

<sup>2</sup>Disponível em [https://docs.google.com/spreadsheets/d/1BKJ9\\_mIQdU504tIoJ7zHmxJeqKQo7gs3buQL9D7iIMs](https://docs.google.com/spreadsheets/d/1BKJ9_mIQdU504tIoJ7zHmxJeqKQo7gs3buQL9D7iIMs)

### A.3.1 Atributos do problema

#### Competição

Foram identificadas e selecionadas 26 competições de visão computacional da plataforma Kaggle. Assim, este atributo serve para identificar qual a competição associada à anotação.

#### Contexto

Toda competição apresenta um problema aos usuários da plataforma. E cada problema está inserido dentro de um contexto, seja no campo da medicina, vida selvagem, etc. Desse modo, este atributo informa qual o contexto do problema apresentado.

#### Métrica

Cada competição possui uma forma de avaliar as submissões dos competidores, chamada “métrica de avaliação”. Algumas das métricas mais comuns em problemas de aprendizagem de máquina são: acurácia, que avalia quantas predições foram feitas corretamente pelo modelo computacional; precisão, que avalia quantas predições positivas foram feitas corretamente; *recall*, que avalia quantos casos positivos foram classificados corretamente; etc.

Diante disso, este atributo informa qual a métrica utilizada para avaliar as submissões da competição.

#### Tipo de métrica

Como dito anteriormente, todas as métricas de avaliação utilizadas pelas competições podem ser divididas em dois tipos, minimização e maximização. E este atributo informa qual o tipo de métrica utilizada pela competição associada.

#### Tipo de problema

Dentre os problemas de visão computacional existentes no Kaggle, os que foram selecionados podem ser divididos em quatro tipos, sendo que os três primeiros seguem as definições dadas por Jason Brownlee em seu artigo online [17]:

- Classificação: este tipo de problema consiste em atribuir um rótulo à uma imagem inteira, ou fotografia.

Para isso, o problema define quais são os rótulos (classes) existentes, fornece um conjunto de imagens rotuladas e um conjunto (denominado “teste”) de imagens não rotuladas. E com isso, o participante deve construir um modelo capaz de identificar qual é a classe presente em cada imagem do conjunto de imagens não rotuladas (chamado de “conjunto de teste”).

- *Object detection* (detecção de objetos): este tipo de problema consiste em atribuir um ou mais rótulos à uma imagem e mostrar a localização dos objetos na imagem, desenhando uma caixa delimitadora em volta de cada objeto.

Para isso, problema define um conjunto de objetos a serem identificados, fornece um conjunto de imagens que apresentam esses objetos, juntamente com a caixa delimitadora e o rótulo de cada objeto presente na imagem, e também fornece um conjunto de imagens que apresentam os objetos, mas sem informar os rótulos nem as caixas delimitadoras. E com isso, o modelo construído pelo participante deve ser capaz de identificar os rótulos e as caixas delimitadoras dos objetos presentes numa imagem.

- *Object segmentation* (segmentação de objetos): este tipo de problema é parecido com “detecção de objetos”, mas ao invés de detectar uma caixa delimitadora, neste tipo de problema deve-se detectar quais são os pixels específicos (conjunto chamado de “máscara”) que compõem cada objeto.

Para isso, o problema define os tipos de objetos a serem identificados, fornece um conjunto de imagens rotuladas, juntamente com as máscaras dos objetos presentes em cada imagem, e fornece um conjunto de imagens sem rótulos nem máscaras. E com isso, o participante deve construir um modelo capaz de reconhecer as máscaras dos objetos presentes numa imagem.

- *Feature pinning*: este tipo de problema consiste em apontar onde estão localizadas certas características da imagem.

O problema define um conjunto de características que sempre estarão presentes nas imagens (como por exemplo, olhos, nariz e boca de um rosto humano), fornece um conjunto de imagens com a localização pontual de cada característica e um conjunto de imagens sem nenhuma localização anotada. E com isso, o participante deve construir um modelo capaz de indicar as localizações pontuais das características em cada imagem do conjunto de teste.

Diante disso, este atributo textual informa qual o tipo de problema da competição, dentre os quatro tipos descritos acima.

### **Quantidade de classes/alvos**

Cada tipo de problema mencionado acima visa classificar imagens ou localizar objetos/características presentes nas imagens. No contexto de aprendizagem de máquina, cada objeto/característica que deseja-se reconhecer recebe a denominação de *classe*. E este atributo numérico consiste em informar a quantidade de classes/alvos que o problema visa classificar/localizar.

### **A.3.2 Atributos da base de dados**

#### **Tamanho da base de dados para treinamento**

Toda competição provê uma base de imagens a serem utilizadas no processo de treinamento do modelo construído. Este atributo numérico informa qual a quantidade de imagens presentes no base de treinamento.

#### **Dimensões padronizadas**

O processo de construção de uma base de dados é um processo complexo, delicado e que requer muita atenção, nesse contexto, por parte dos organizadores das competições no Kaggle. Dentre as etapas deste processo uma delas é coletar dados (imagens), e dependendo da forma de coleta as imagens obtidas podem vir em dimensões diferentes ou iguais entre si. Por exemplo, suponha que todas as imagens tenham sido coletadas utilizando uma mesma câmera, certamente todas as imagens obtidas terão as mesmas dimensões, gerando uma base de dados com “dimensões padronizadas”. Suponha agora que toda a base de dados será composta de imagens obtidas através do buscador Google, dessa forma, a menos que seja feito algum tipo de tratamento sobre as imagens, elas certamente terão dimensões diferentes entre si.

Assim, o papel deste atributo booleano é indicar (através dos valores “verdadeiro” e “falso”) se a base de treinamento é composta de imagens com dimensões padronizadas ou não.

#### **Dados uniformes**

“Dados uniformes” é um termo adotado neste trabalho para indicar dados com um comportamento uniforme/controlado, ou seja, imagens que apresentam os objetos de interesse em localidades esperadas (como a região central de uma foto, por exemplo).

Como mencionado anteriormente, o processo de construção de uma base de dados é um processo que requer atenção e planejamento. No entanto, dependendo do problema, obter dados com um comportamento controlado pode ser praticamente impossível. Por exemplo, na construção de uma base de imagens de células sanguíneas o ambiente é totalmente controlado, pois tais amostras serão obtidas através de um microscópio, dentro de um laboratório controlado, etc. Em contrapartida, no caso de uma base de fotos de animais selvagens em seus habitats naturais controlar o ambiente e os sujeitos de estudo (os animais) é uma tarefa muito complicada, pois por se tratar de animais selvagens, que provavelmente vivem em reservas ambientais, não tem como garantir que todas as imagens obtidas conseguirão capturar os animais em localidades específicas e nem tem como interferir no meio ambiente para garantir que os animais se posicionem num local esperado.

Diante disso, a função deste atributo booleano é informar se os dados disponibilizados pela competição têm um comportamento uniforme, ou seja, se as imagens apresentam os objetos de interesse em localidades esperadas.

### **Base de dados balanceada**

Ao construir uma base de treinamento, é desejável que todas as classes estejam representadas em quantidades iguais de ocorrências entre si. Uma base nessas condições é denominada “balanceada”. Mas, isso nem sempre acontece, o que gera as chamadas “bases desbalanceadas”.

Neste trabalho adotamos o seguinte critério: para uma base ser classificada como “desbalanceada” é necessário que ela possua alguma classe com a quantidade de ocorrências equivalente a  $2/3$  ou menos da classe mais comum (que tem mais ocorrências).

Assim, este atributo booleano indica se a respectiva base está balanceada ou não, segundo o critério acima.

### **Tem poucos dados por classe**

Uma “regra de ouro” muito útil é: a base de treinamento precisa ter ao menos 1.000 imagens representativas de cada classe [18]. Diante disso, este atributo booleano informa se existe alguma classe do problema que viola esta regra, ou seja, se existe alguma classe com menos de 1000 ocorrências na base de treinamento.

## **A.3.3 Atributos da solução**

### **Link**

Como já mencionado, dentro de cada competição existe uma seção para divulgar soluções publicamente. Toda solução publicada se encontra no formato de documento *Jupyter Notebook*, que contém tanto trechos executáveis de código quanto ricos elementos textuais (como parágrafos, equações, figuras, links, etc).

Diante disso, este atributo simplesmente guarda o link para acessar o notebook da solução publicada.

### **Pontuação**

Ao passar pelo processo de avaliação, a métrica irá quantificar o quão boa é a resposta produzida pelo modelo construído. Esta quantificação se chama “pontuação”. Cada submissão pontua um certo valor segundo a métrica a que foi submetida.

Portanto, este é um atributo numérico que diz a pontuação alcançada pela respectiva solução na métrica da competição.

### **Pré-processamento**

Seja numa competição ou num caso concreto enfrentado por alguma empresa, muitas vezes as imagens que compõe a base de dados possuem certos problemas, como algum tipo de ruído, dimensões variadas, imagens coloridas misturadas com imagens em tons de cinza, etc.

Esse tipo problemático de dados por vezes requer alguma espécie de tratamento antes de ser aplicado ao modelo computacional, pois uma boa preparação dos dados é essencial para a construção de um modelo [19]. Este tipo de tratamento é chamado “pré-processamento”, cujo objetivo é fazer uma espécie de “limpeza”, ou melhoria, nas imagens que compõem a base de dados do problema.

Diante disso, este atributo trata de descrever em palavras a etapa de pré-processamento da solução apresentada.

### **Redimensiona todas as imagens**

Uma técnica comum na etapa de pré-processamento é o redimensionamento de todas as imagens da base de dados para dimensões padronizadas.

Tendo em vista que redes neurais têm entradas de dimensões fixas, e que a base de dados pode ter imagens com dimensões variadas (ou dimensões muito grandes), a prática do redimensionamento de todas as imagens é uma forma de contornar essas situações, seja para padronizar o tamanho das imagens (no caso de imagens com dimensões variadas), ou reduzir o custo computacional do modelo, pois no caso de imagens com dimensões muito grandes é natural pensar em reduzir a escala a fim de simplificar a estrutura do modelo.

Diante disso, este atributo booleano informa se na etapa de pré-processamento o competidor redimensionou todas as imagens da base de dados para um tamanho uniforme ou não.

### **Normaliza todas as imagens**

“Normalização” é um conceito que vem da estatística, e consiste em eliminar a unidade de medida e transformar os dados em valores dentro de um intervalo conhecido [20]. Trazendo para o contexto de aprendizagem de máquina, “normalização é uma técnica frequentemente aplicada na etapa de preparação dos dados. E seu objetivo é transformar os valores de colunas numéricas para uma escala comum, sem distorcer as diferenças nos intervalos de valores” [21]. Ou seja, são transformações que levam os valores numéricos originais para dentro de um intervalo conhecido, mantendo proporcionalmente as diferenças entre os valores.

Assim, este atributo booleano se encarrega de informar se a solução apresentada utiliza a técnica de normalização nas imagens, pois as imagens são representadas como vetores de números.

### **Faz outro pré-processamento**

Além do redimensionamento existem outras técnicas de pré-processamento, por exemplo remover a saturação das imagens, deixando-as em preto e branco; remover imagens com muito ruído ou duplicadas da base de dados; etc.

Por isso, este é um atributo booleano que indica se foi utilizada alguma outra técnica na etapa de pré-processamento, que não seja redimensionar todas as imagens, nem normalizá-las.

### **Dados externos**

No Kaggle, o organizador da competição disponibiliza a base de dados que deve ser utilizada na mesma. Mas pode ocorrer de existir uma base de dados maior na Internet, e mesmo que não tenha sido feita para essa competição, pode vir a melhorar os resultados do modelo proposto pelo participante.

Assim, este atributo booleano indica se na solução foram utilizados dados além dos disponibilizados pela competição.

### ***Data Augmentation* (Aumento de dados)**

Já na fase de treinamento do modelo uma técnica que pode ser aplicada é a chamada *data augmentation*, ou aumento de dados. “*Data augmentation* é uma estratégia que permite aos profissionais aumentarem significativamente a diversidade de dados disponíveis para o treinamento de modelos, sem ter que coletar novos dados” [22].

Esta é uma técnica muito comum de ser utilizada para evitar o problema de *overfitting* (sobreajuste) [23, 24]. *Overfitting* é um fenômeno estatístico, onde o modelo se ajusta muito bem (consegue fazer excelentes previsões) sobre os dados de treinamento, mas tem um péssimo desempenho em prever novos resultados (fora dos dados de treinamento) [25].

Este problema é mais propenso a ocorrer em modelos treinados sobre bases de dados pequenas, e aplicar métodos automatizados para aumentar a variedade dos dados é uma estratégia efetiva, e muito utilizada, para generalizar modelos [26, 23].

Alguns dos “aumentos” possíveis incluem transladar as imagens por uma quantidade aleatória de pixels, rotacioná-las em sentidos aleatórios, inverter as imagens horizontalmente, etc.

Contudo, o objetivo deste atributo booleano é apenas relatar se na respectiva solução foi utilizada ou não a técnica de aumento de dados.

### ***Utiliza transfer learning***

*Transfer learning* é uma técnica de aprendizagem de máquina que serve para reduzir o processo de treinamento [14]. A ideia dessa técnica é utilizar modelos previamente treinados, possivelmente em outros contextos, de forma a aproveitar o “conhecimento” adquirido por esses modelos.

No Kaggle, e muitas vezes em casos concretos, quando o problema é muito complexo, ou apresenta uma base de dados muito grande, treinar um novo modelo do zero pode ser muito custoso em termos de tempo e poder computacional. Nesses casos, intuitivamente o *transfer*

*learning* surge como uma opção, mas não há nenhuma garantia de que essa abordagem será sempre uma boa opção.

Com isso, este atributo booleano serve apenas para indicar se a técnica de *transfer learning* foi utilizada em algum momento da solução analisada.

### **Pesos utilizados no *transfer learning***

Nos modelos computacionais de aprendizagem de máquina, o “conhecimento” do mesmo fica armazenado ou, melhor dizendo, é representado pelos pesos das ligações entre os neurônios da rede neural.

Caso a respectiva solução tenha utilizado a técnica de *transfer learning*, esse atributo indica qual foi o conjunto de pesos utilizado/importado no processo.

### **Modelos/camadas**

Os modelos utilizados nas soluções observadas são redes neurais. Estas são formadas por uma combinação de camadas diferentes, cada uma com sua função específica. Além disso, numa única solução é possível combinar diversos modelos. Como cada modelo tem vários tipos de camada, cada uma com um propósito diferente, é interessante analisar quais modelos e quais camadas foram utilizados em cada solução.

Por isso, o propósito desse atributo é informar em palavras quais foram os modelos e/ou as camadas utilizadas em cada solução.

### **Ativações**

Durante a construção de um modelo, existem diversos tipos de ativações que podem ser utilizadas pelas camadas que o compõem. O tipo da ativação de cada camada define como os valores serão passados para a camada seguinte. Dependendo da função matemática de ativação, é possível obter comportamentos diferentes. Dessa forma, esse atributo descreve em palavras quais foram as funções de ativação utilizadas no modelo da solução observada.

### **Processo de treinamento**

O processo de treinamento de um modelo pode apresentar diversas particularidades, de forma a alcançar um melhor resultado. Como, por exemplo, o modelo pode ser treinado somente com imagens de uma determinada classe, ser treinado apenas com uma parcela da base de treinamento, etc.

Assim, este atributo descreve em palavras algumas particularidades da etapa de treinamento da solução analisada, como a técnica utilizada para executar o treinamento.

### **Tamanho do conjunto de avaliação**

De forma a evitar sobreajuste, no processo de treinamento, a solução deve realizar e testar o comportamento do modelo em um conjunto de avaliação. Esse conjunto de avaliação é um subconjunto da base de treinamento original que não deve ser utilizado para treinar o modelo e sim, somente, para avaliá-lo.

Com isso, este atributo numérico indica o tamanho do conjunto de avaliação utilizado na solução analisada.

### ***Loss function* (função de perda)**

“Uma função de perda é um método de avaliar o quão bem o modelo representa o conjunto de dados” [27]. Em outras palavras, a função de perda quantifica, matematicamente, o quão distante do esperado estão as previsões do modelo construído. No entanto, existem diferentes funções para calcular tal desempenho. Por isso, este atributo informa quais funções de perda foram utilizadas em cada competição.

### ***Callbacks***

*Callback* é um conjunto de funções que serão chamadas em certas etapas do processo de treinamento [28]. Algumas soluções utilizam *callbacks* para refinar estas etapas. Por isso, este atributo indica, em palavras, quais funções de *callback* foram utilizadas na solução do problema.

### **Biblioteca**

Implementar do zero os modelos é uma tarefa extremamente trabalhosa e sem muitas vantagens para quem busca resolver um problema sem ter que “reinventar a roda”. Por isso, é muito comum utilizar bibliotecas que auxiliam no desenvolvimento do modelo. E a função deste atributo é informar se e qual biblioteca de aprendizagem de máquina foi utilizada.

# Apêndice B

## Catálogo de Competições<sup>1</sup>

Foram selecionadas 26 competições de visão computacional da plataforma Kaggle. Para cada competição foi selecionada a melhor solução publicada, também chamada de notebook. Assim, a seguir serão descritas<sup>2</sup> todas as competições selecionadas, bem como as particularidades de cada base de dados e cada solução.

### B.1 Digit Recognizer

Esta competição apresenta um problema clássico para quem deseja ingressar no mundo da visão computacional. Nesta competição os participantes devem resolver o problema de reconhecimento de dígitos escritos à mão, ou seja, OCR – Optical Character Recognition. As entradas são dígitos arábicos, de 0 até 9. E o participante deve construir um modelo capaz de identificar qual o dígito apresentado.

Assim, esta competição apresenta um problema do tipo classificador, com 10 classes. A métrica de avaliação desta competição é a acurácia.

#### B.1.1 Base de dados

A base da competição é a MNIST<sup>3</sup> [29]. Para esta competição, foram selecionadas 42.000 imagens de dimensões 28x28 pixels, em escala de cinza. Cada imagem possui um dígito numérico (arábico) escrito à mão no centro da imagem. A frequência de cada dígito na base é praticamente a mesma, ou seja, tem dados suficientes para executar o treinamento.

---

<sup>1</sup> Autores: Daniel Humberto Cavalcante Vassalo, Nelson Gomes Neto

<sup>2</sup> As descrições das competições são basicamente resumo e uma tradução livre das descrições presentes no Kaggle, bem como nos casos de notebooks com explicações detalhadas. Além disso, todas as soluções selecionadas foram feitas na linguagem de programação Python.

<sup>3</sup> Disponível em: <http://yann.lecun.com/exdb/mnist/>

Tabela B.1: Imagens do Digit Recognizer

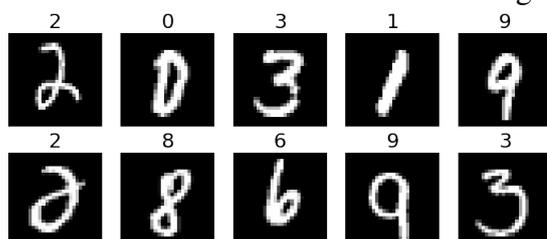


Figura B.1: Exemplos da base de dados MNIST

### B.1.2 Solução

O autor da solução selecionada<sup>4</sup>, durante a etapa de pré-processamento, normalizou todas as imagens (dividindo o valor de cada pixel por 255).

Para esta solução, o autor implementou sua própria rede neural *multilayer perceptron* (MLP), do zero. Nessa implementação, ele utilizou apenas uma camada oculta com 256 neurônios, utilizou a função de ativação *ReLU* ao longo da rede e *softmax* na camada de saída.

Já na etapa de treinamento, foi utilizada toda a base de treinamento para treinar o modelo. Além disso, os pesos foram atualizados através do processo de *gradient descent*, com o tamanho de cada passo mudando adaptativamente através de *root mean square prop* (RMSprop).

### B.1.3 Resultado

A solução descrita obteve uma acurácia de 97,057%. Apesar de não ser o melhor resultado, é um bom resultado para um modelo simples e implementado do zero.

Mesmo com esse modelo simples, ainda existe espaço para aplicação de algumas técnicas que possivelmente melhorariam o resultado, como *data augmentation* e treinamento em lotes. E para realmente alavancar o resultado, um modelo mais robusto seria indicado, como uma rede neural convolucional (CNN).

## B.2 Aerial Cactus Identification

Para avaliar o impacto da mudança climática na fauna e na flora, é essencial avaliar como as atividades humanas impactam áreas de proteção natural. Para isso, é necessário identificar o tipo de vegetação presente em áreas de proteção natural. Assim, o objetivo desta competição é identificar a presença (ou ausência) de um tipo específico de cactus (*Neobuxbaumia tetetzo*) em imagens aéreas. Ou seja, dada uma foto aérea, o modelo deve dizer a probabilidade da foto

<sup>4</sup><https://www.kaggle.com/scaomath/simple-mnist-numpy-from-scratch>

conter um cactus.

Desse modo, a competição apresenta um problema do tipo classificador, com 2 classes (“tem cacto” e “não tem cacto”), e para avaliar a submissão dos participantes a métrica escolhida foi acurácia.

### B.2.1 Base de dados

Esta base de dados [30] é composta por 17.500 fotos aéreas, onde 4.364 fotos não contém cacto e 13.163 contém, ou seja, esta base não é balanceada, pois existem muito mais instâncias com cacto do que sem. No entanto, ambas as classes possuem mais de 1.000 ocorrências, ou seja, possuem dados suficientes para treinar o modelo propriamente. Além disso, todas as fotos possuem as mesmas dimensões, 32x32 pixels, e comportamento uniforme, pois sempre que um cacto está presente ele aparece no centro da imagem.

### B.2.2 Solução

A solução selecionada<sup>5</sup> mostra que na etapa de pré-processamento o autor utilizou apenas a técnica de normalização para todas as imagens.

Nesta solução o autor construiu um modelo próprio, utilizando seguintes camadas da biblioteca Keras: *Activation*, *BatchNormalization*, *Conv2D*, *Dropout*, *GlobalAveragePooling2D* e *MaxPooling2D*. Durante toda a extensão do modelo foi utilizada a função de ativação *ReLU* nas camadas *Activation*, com exceção da camada de saída, que utilizou a função *sigmoid*.

Na fase de treinamento ele utilizou a técnica de *data augmentation* e utilizou o *Stratified K Fold* com 10 dobras, ou seja, a cada ciclo 10% da base de treinamento era reservada para fazer a validação do modelo. Ainda na fase de treinamento, o modelo utilizou os seguintes *callbacks*: *EarlyStopping*, *ModelCheckpoint* e *ReduceLRonPlateau*.

### B.2.3 Resultado

A solução descrita acima alcançou a pontuação máxima, 100% de acurácia. Mas ficou em 118º lugar no ranking público.

## B.3 Dogs vs Cats

Esta competição apresenta outro problema clássico para quem está ingressando no mundo da visão computacional. O objetivo desta competição é identificar se uma imagem possui um cachorro ou um gato. Ou seja, a competição apresenta um problema do tipo classificado, com 2 classes. As submissões serão avaliados pela métrica *logarithmic loss* – ou *log loss*.

<sup>5</sup><https://www.kaggle.com/horohoro/aerial-cactus-identification-simplecnn>

Tabela B.2: Imagens do Aerial Cactus Identification



Figura B.2: Exemplos com Cacto

Figura B.3: Exemplos sem Cacto

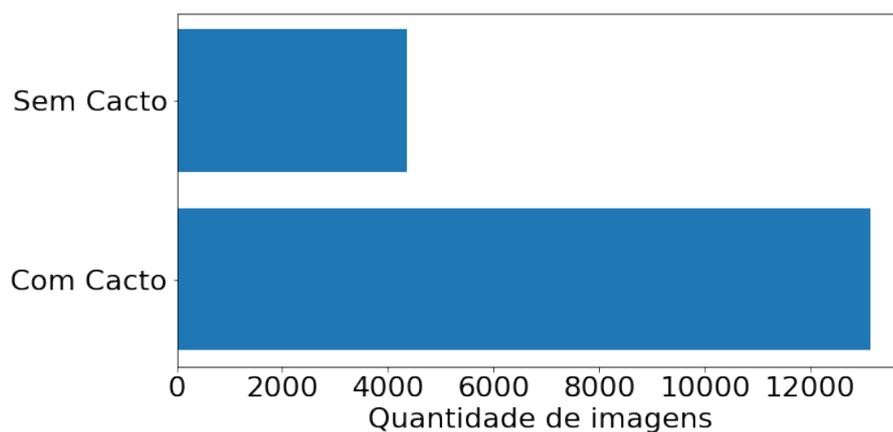


Figura B.4: Distribuição da base de dados

### B.3.1 Base de dados

A base de dados é composta por 25.000 imagens, uma metade das imagens possuem cachorros e a outra gatos. As dimensões das imagens não possuem um padrão, e o posicionamento dos animais também não seguem nenhum tipo de padrão (em algumas imagens aparecem seres humanos, em outros os animais aparecem muito pequenos em algum canto da foto, etc). Ou seja, a base não é “comportada”, mas possui dados suficientes para executar o treinamento do modelo.

Tabela B.3: Imagens do Dogs vs Cats

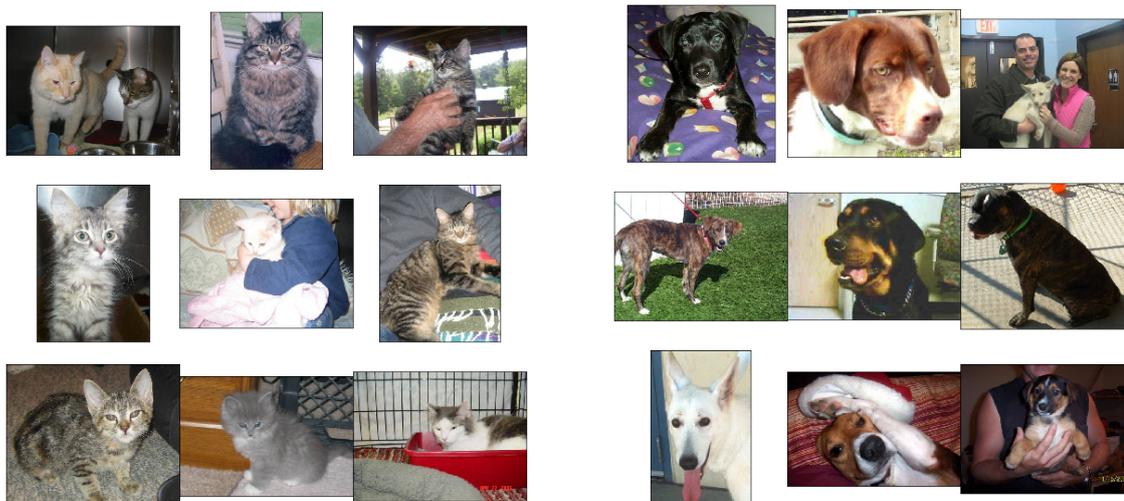


Figura B.5: Exemplos de imagens com gatos

Figura B.6: Exemplos de imagens com cachorros

### B.3.2 Solução

Na etapa de pré-processamento da solução<sup>6</sup> selecionada, as imagens foram redimensionadas de forma que pudessem servir de entrada para cada um dos modelos utilizados.

Para o modelo final desta solução foi utilizada a técnica de *ensemble*, que consiste de uma combinação de vários modelos previamente treinados, nesta solução foram utilizados três: VGG16, ResNet50 e InceptionV3. A combinação foi feita da seguinte forma: esses modelos recebem a imagem de entrada e a saída deles é concatenada (Exemplo: [1, 2] + [3, 4] = [1, 2, 3, 4]), em seguida o resultado da concatenação é conectado a uma camada *Dense* com função de ativação *sigmoid*, e essa tem apenas um número como saída: a probabilidade da foto ser de um cachorro. A função de perda, também chamada de *loss function*, do modelo é a *binary cross-entropy*.

Durante a fase de treinamento foi separado 10% da base de treino para validação do modelo. Também foi utilizada a técnica de *data augmentation*.

### B.3.3 Resultado

A pontuação dessa solução foi 0,04701 na métrica *log loss*.

<sup>6</sup><https://www.kaggle.com/renyuanfang/pretained-models>

## B.4 Facial Keypoints Detection

Nesta competição, foi definido um conjunto de pontos-chave (15) do rosto humano, como o centro do olho esquerdo, a ponta do nariz, o canto direito da boca, etc. E o objetivo é criar um modelo que seja capaz de identificar a posição de cada ponto-chave em imagens com um rosto humano. A métrica de avaliação desta competição é a *root mean squared erro* (RMSE), ou raiz quadrada do erro médio.

### B.4.1 Base de dados

Essa base de dados conta com 7.049 imagens. Todas as imagens têm as mesmas dimensões (96x96 pixels) e apresentam um rosto humano no centro (comportamento uniforme). No entanto, a base está desbalanceada, pois alguns pontos-chave aparecem uma quantidade de vezes significativamente menor, na base de treinamento. Mas, como cada ponto-chave está presente em mais de 1.000 imagens, pode-se dizer que a base não possui poucos dados, ou seja, os dados são suficientes para executar um bom treinamento.



Figura B.7: Exemplos da base de dados, com os pontos-chave representados em vermelho

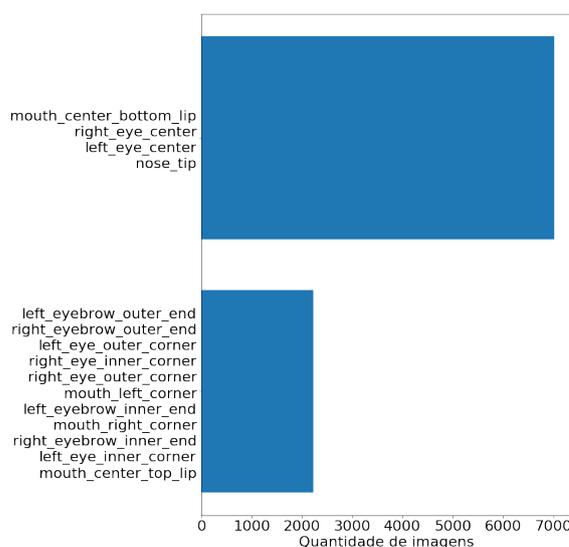


Figura B.8: Frequência da cada ponto-chave

### B.4.2 Solução

A solução<sup>7</sup> selecionada apenas normalizou as imagens, na etapa de pré-processamento.

O modelo proposto foi construído utilizando as seguintes camadas da biblioteca Keras: *BatchNormalization*, *Conv2D*, *Dense*, *Dropout*, *Flatten*, *MaxPooling2D*. Ao longo do modelo, foi

<sup>7</sup><https://www.kaggle.com/utkarsh4430/facial-keypoints-detection-basic-keras-model>

utilizada a função de ativação *leakyReLU*, com exceção da camada de saída, onde foi utilizada a função *ReLU*. A função de perda utilizada no modelo foi a *mean squared error*.

Já na etapa de treinamento, foi separado 10% da base de treino para validação do modelo. E então, o modelo foi treinado em 3 etapas: a primeira etapa consistiu de lotes com 64 imagens durante 100 épocas (de treinamento), a segunda utilizou lotes de 128 imagens durante 50 épocas, e por fim, a terceira etapa utilizou lotes de 256 imagens durante 50 épocas.

### B.4.3 Resultado

A solução descrita obteve 1,98535 pontos na métrica RMSE.

## B.5 Humpback Whale Identification Challenge

A preservação da vida marinha é uma preocupação que toma a atenção de muitos pesquisadores e cientistas. Para acompanhar a conservação da vida das baleias, os cientistas encarregados capturam imagens e vídeos para posteriormente fazerem anotações e análises devidas. O problema é que as anotações referentes à identificação das espécies das baleias fotografadas são feitas de modo manual.

Com isso, o objetivo desta competição é obter um classificador que consiga prever a espécie (dentre 4.251 possíveis) de uma baleia olhando apenas para uma foto da cauda dela. A métrica de avaliação desta competição é a *mean average precision at 5*.

Tabela B.4: Imagens do Humpback Whale Identification

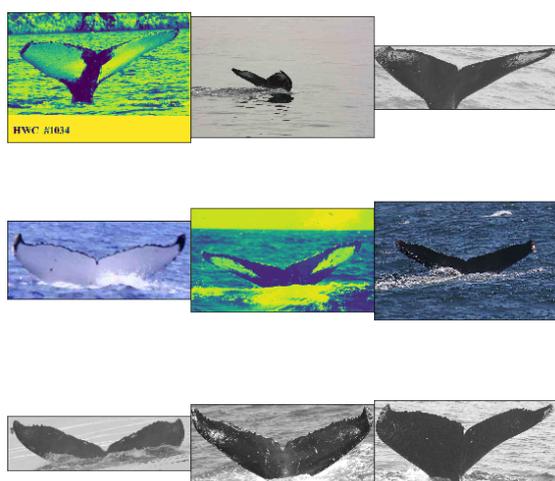


Figura B.9: Exemplos da base de dados

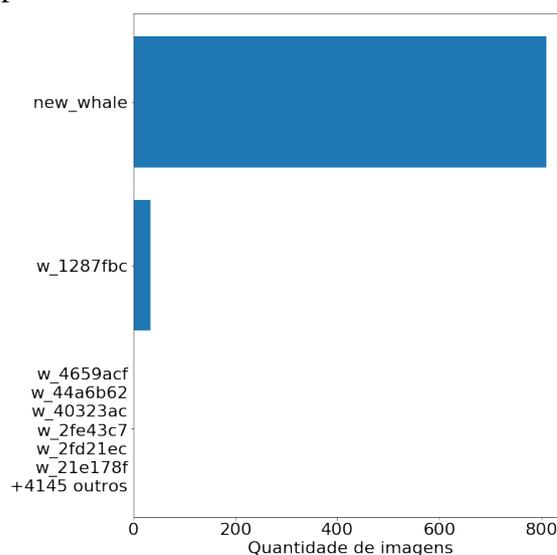


Figura B.10: Frequência de cada baleia

### B.5.1 Base de dados

A base apresentada foi construída e disponibilizada pelo grupo Happy Whale. No entanto, a base possui algumas características um tanto problemáticas, como a quantidade de imagens. A base é composta de 9.850 imagens e 4.251 classes, ou seja, em média existem 2 imagens para cada classe. Diante disso, os números indicam que a base está longe de ter dados suficientes para executar um treinamento decente. Além disso, as imagens não possuem nenhum tipo de padronização (existem imagens coloridas, em preto e branco, etc), elas não tem dimensões padronizadas, e para completar as caudas aparecem em regiões diferentes ao longo das fotos. Por fim, é possível observar que a base não está balanceada, pois a classe “new whale” aparece 8% das vezes na base de treinamento (muito mais que as demais classes).

### B.5.2 Solução

Diante de tantos problemas presentes na base de dados, o notebook<sup>8</sup> selecionado apresenta uma solução um tanto complexa. Resumidamente, o autor utilizou aprendizagem de máquina para refinar a base de treinamento, *data augmentation* para evitar sobreajuste, construiu uma rede neural siamesa e utilizou a técnica de *ensemble* para gerar a submissão final. A seguir toda esta solução será descrita em mais detalhes.

#### Construção de uma base refinada

Primeiramente, durante a etapa de pré-processamento, o objetivo do autor é criar uma “nova base” com imagens uniformes, contendo somente a caudas da baleia – este processo está descrito em maiores detalhes em outro notebook do mesmo autor (<https://www.kaggle.com/martinpiotte/bounding-box-model>). Para isso o autor criou um modelo que reconhece a caixa delimitadora da cauda, para então salvá-la e mais na frente poder utilizar somente a área da imagem que contém a cauda. Este modelo foi treinado em cima de 1.000 imagens retiradas da base de treinamento original, onde cada imagem possui uma tupla de coordenadas para a caixa delimitadora correspondente, e outras 200 imagens da mesma base foram utilizadas para fazer a validação do modelo. Mas antes de serem passadas para o modelo, as 1.200 imagens receberam alguns tratamentos: elas foram convertidas para preto e branco, depois foram comprimidas horizontalmente, foram todas redimensionadas para 128x128 pixels, e por fim foram normalizadas.

Além disso, o autor construiu manualmente uma lista negra de imagens que ele julgou não serem úteis para o treinamento, como imagens onde aparecem duas baleias, ou a cauda não está visível, etc.

Em seguida, na etapa de treinamento deste primeiro modelo, o autor utilizou a técnica de *data augmentation* para aplicar transformações aleatórias às imagens, como rotação, translação

<sup>8</sup><https://www.kaggle.com/martinpiotte/whale-recognition-model-with-score-0-78563>

e zoom. Assim, com o modelo treinado ele foi executado em toda a base original (de treinamento e teste), e para cada imagem foram salvas as coordenadas da caixa delimitadora da cauda (nas dimensões originais de cada foto, sem aplicar nenhum tipo de transformação).

### Mais pré-processamento

Prosseguindo na etapa de pré-processamento, após gerar a base de caixas delimitadoras, a próxima técnica utilizada foi detectar imagens duplicadas em toda a base original. Para identificar essas imagens foram utilizadas algumas métricas, como *Perceptual Hash*, tamanho e erro quadrático médio (em pixels) entre duas imagens.

Em seguida as imagens foram submetidas às seguintes transformações: rotação vertical (caso a imagem esteja de cabeça pra baixo), converter para preto e branco, e transformação afim (*affine transformation*).

### Modelo proposto

Após o pré-processamento, o autor descreve o modelo construído. A arquitetura do modelo proposto é uma rede neural siamesa, cujo funcionamento consiste em comparar duas imagens e decidir se elas são da mesma baleia ou de baleias diferentes. Dessa forma, ao comparar cada imagem da base de teste com cada imagem da base de treinamento, as baleias mais prováveis podem ser ordenadas por nível de semelhança.

Uma rede neural siamesa é composta de duas partes: extração de características das duas fotos (feita por um modelo denominado pelo autor do notebook como “*branch model*”) e comparação das características extraídas para decidir se as baleias são semelhantes ou não (feita por um modelo denominado pelo autor como “*head model*”).

Para o *branch model* o autor utilizou uma rede neural convolucional customizada, inspirada na estrutura do modelo ResNet [5]. Devido ao pequeno tamanho da base de dados, o autor aponta que tentou manter o número de parâmetros aprendidos pela rede relativamente baixo, mas mantendo o modelo suficientemente expressivo. Este modelo é composto de 6 blocos encadeados, onde a cada bloco a resolução processada vai diminuindo, devido às camadas de *Pooling* entre os blocos. Neste modelo foram utilizadas as seguintes camadas da biblioteca Keras: *Activation*, *BatchNormalization*, *Conv2D*, *GlobalMaxPooling2D* e *MaxPooling2D*. Ao longo dessas camadas foi utilizada somente a função de ativação *ReLU*.

Já no “*head model*” o autor afirma ter adotado uma abordagem não usual, ao invés de utilizar uma medida de distância entre as características, ele computou quatro coisas, para cada par de características: a soma, o produto, a diferença absoluta e o quadrado da diferença. Em seguida esses quatro valores são passados para uma rede neural pequena, que irá aprender a como relacionar certos valores iguais, ou próximos, a zero com as baleias semelhantes. E para isso essa pequena rede neural ele utilizou as seguintes camadas da biblioteca Keras: *Concatenate*,

*Conv2D, Dense, Flatten, Lambda e Reshape*. A saída desta rede passa pela função de ativação *sigmoid*. E a função de perda do modelo é a *binary cross-entropy*.

### **Processo de treinamento**

Antes de iniciar o processo de treinamento o autor faz umas observações importantes: a rede precisa dar uma pontuação alta para a baleia correta, ao mesmo tempo que dá uma pontuação baixa para todas as outras; além disso o modelo deve reconhecer baleias, e não paisagens, pois visto que a base de imagens é pequena é natural pensar que o modelo vá utilizar coisas como aves, formatos de ondas, etc, para fazer o reconhecimento. E para evitar este último cenário os dados apresentados ao modelo não podem ter viés. Pois, segundo o autor, se uma imagem aparece com frequência, é provável que o modelo aprenda a predizer sempre, por exemplo, que aquela imagem não é semelhante, sem aprender a como comparar a baleia propriamente. Então, durante o treinamento cada imagem deve aparecer uma mesma quantidade de vezes, sendo 50% das vezes como exemplos positivos e 50% como negativos.

Além disso, algumas imagens não foram utilizadas no processo de treinamento: todas as imagens da lista negra, todas as imagens duplicadas, todas as imagens da classe “new whale” e todas as baleias que só apresentavam uma imagem.

O processo de treinamento foi feito em 400 épocas. E à medida em que as épocas iam passando, os seguintes valores iam mudando: taxa de aprendizagem; regularização L2; e a constante K usada no cálculo para pontuar imagens semelhantes, a fim de dificultar o treinamento.

### **Construção da submissão**

Na etapa de predição, para decidir qual é a classe da baleia apresentada, primeiramente o autor verifica se a imagem apresentada é duplicata de alguma imagem do conjunto de treinamento, e, se for, a baleia “duplicada” é selecionada como uma candidata muito provável. Em seguida, para cada imagem que não seja da classe “new whale”, presente no conjunto de treinamento, é calculada a pontuação do par de imagens. Em seguida, a classe “new whale” é adicionada como uma candidata a partir de um certo limiar – obtido através de tentativa e erro pelo autor.

### **Segundo modelo e treinamento, para melhorar o resultado**

Em seguida o autor usou o modelo treinado para gerar mais dados de treinamento, para treinar um segundo modelo. Na verdade trata-se do mesmo modelo descrito acima, mas ambos os modelos serão mantidos para executar a técnica de *ensemble*. Para gerar mais dados de treinamento, o autor utilizou imagens do conjunto de teste que obtiveram uma pontuação maior que 0,999999 no modelo anterior, ou seja, selecionou apenas imagens que foram classificadas com uma “certeza muito alta”, segundo o modelo.

De posse dos dois modelos treinados, o autor utilizou a técnica de *ensemble* para mesclar os dois modelos. A abordagem utilizada consiste em fazer uma combinação linear das saídas dos dois modelos, sendo 0,45 o peso aplicado à saída do primeiro modelo e 0,55 o peso aplicado ao último. Esses pesos foram encontrados por tentativa e erro, segundo o autor. Em seguida o autor utilizou o mesmo processo de predição descrito acima, só que utilizando as saídas produzidas pelo *ensemble* descrito.

### B.5.3 Resultado

Após todo esse processo complexo, a solução descrita obteve 0,78563 pontos na métrica *mean average precision at 5*. Esse é um valor incrivelmente alto, levando em consideração o quão ruim era a base de dados. E diante de todos os tratamentos executados pelo autor, verifica-se o impacto que uma base ruim pode ter sobre o desenvolvimento de uma solução.

## B.6 Humpback Whale Identification

A motivação desta competição é a mesma da competição anterior – ajudar na preservação e monitoramento da vida marinha. No entanto, esta competição apresenta algumas diferenças, pois conta com 5.005 classes e mais que o dobro de imagens na base de dados. Mas o objetivo ainda é o mesmo: construir um modelo capaz de identificar a espécie da baleia, olhando apenas para uma foto de sua cauda. A métrica de avaliação também é a mesma – *mean average precision at 5*.

### B.6.1 Base de dados

A base desta competição contém 25.361 fotos de caudas de baleias. Apesar da quantidade de imagens ser maior do que na versão anterior, os problemas ainda permanecem – a quantidade de imagens é muito baixa para a quantidade de classes, as imagens não possuem dimensões padronizadas, nem comportamento uniforme, e além disso a base está desbalanceada (com a grande maioria das imagens sendo da classe “new whale”).

### B.6.2 Solução

Na fase de pré-processamento da solução<sup>9</sup> selecionada, os autores pegaram uma base de dados contendo as caixas delimitadoras das caudas das baleias e recortaram as imagens de acordo (com uma margem adicional de segurança, para eventuais imprecisões das caixas delimitadoras). Além disso, redimensionaram todas as imagens para as dimensões 384x384x1 pixels e as normalizaram.

<sup>9</sup><https://www.kaggle.com/code/gass/siamese-net-with-ensemble>

O modelo utilizado aqui foi inspirado em outro notebook (feito pelo mesmo autor da solução da competição anterior) e se trata de uma rede neural siamesa com *ensemble*. E a única diferença, segundo os autores, está na camada de entrada, que aceita os canais RGB. Além de terem pego a inspiração, os autores também reutilizaram os pesos obtidos pelo outro notebook, ou seja, utilizaram *transfer learning*.

A rede neural siamesa utilizada é constituída das seguintes camadas da biblioteca Keras: *BatchNormalization*, *Conv2D*, *Dense*, *Flatten*, *Lambda*, *GlobalMaxPooling2D*, *MaxPooling2D*. Ela utiliza as funções de ativação *ReLU*, ao longo das camadas, e *sigmoid*, na camada de saída. Além disso, a função de perda utilizada é a *binary cross-entropy*.

Apesar de ter sido utilizada a técnica de *transfer learning*, um treinamento ainda se faz necessário devido à modificação da camada de entrada. Diante disso, para o treinamento foram desconsideradas as imagens da classe “new whale”. E foi utilizada a técnica de *data augmentation* para executar transformações como recorte e translação nas imagens.

Da mesma forma que a solução anterior, o “valor da predição” de cada classe tem que estar acima de um certo limiar, para que ela seja apontada como a provável classe da imagem. Caso contrário a imagem é classificada como “new whale”.

### B.6.3 Resultado

A solução apresentada obteve 0,94067 pontos na métrica *mean average precision at 5*. E como pode-se ver, o resultado alcançado nesta competição é bem melhor do que o alcançado na competição anterior (0,78563). Tendo em vista que a abordagem foi praticamente a mesma, o problema foi o mesmo, e a principal diferença foi o tamanho da base de dados – que apesar de ainda continuar muito pequena, teve seu tamanho dobrado – podemos, provavelmente, atribuir este ganho de performance à base de dados um pouco maior.

No entanto, vale ressaltar que esse também é um resultado muito impressionante, pois tem-se em média 5 imagens por classe, nesta competição.

## B.7 Dog Breed Identification

Esta é outra competição com um problema clássico para quem está começando a aprender sobre visão computacional. Esta competição busca um classificador que seja capaz de indicar qual a raça, dentre 120, mais provável do cachorro presente na imagem. Nesta competição foi utilizada a métrica *multi-class log loss*.

### B.7.1 Base de dados

Esta base de dados conta com 10.222 imagens de cachorros. As dimensões das imagens não são padronizadas e nem o posicionamento dos cachorros nelas. Mas, a base é balanceada, pois não

existe nenhuma raça de cachorro com uma quantidade de imagens muito diferente das outras, e por conta disso pode-se dizer que a quantidade de dados é relativamente baixa para executar um treinamento decente, pois existem, em média, apenas 102 imagens por classe.

Tabela B.5: Imagens do Dog Breed Identification

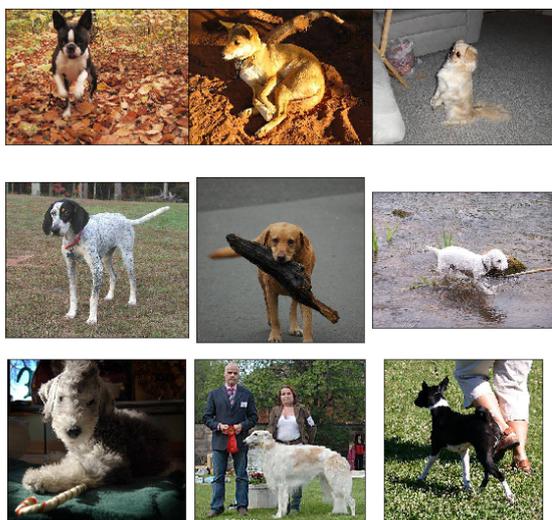


Figura B.11: Exemplos da base de dados

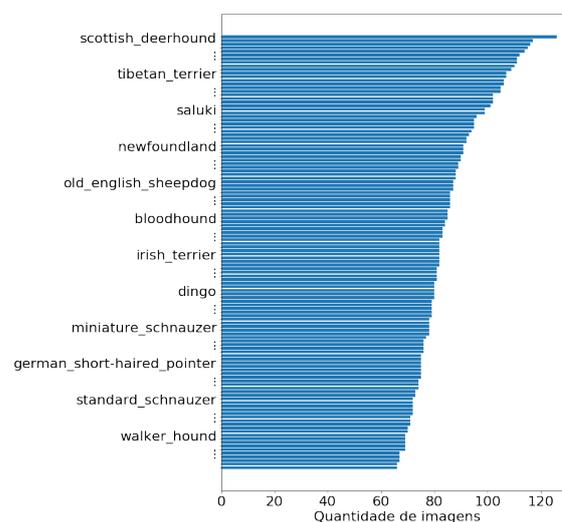


Figura B.12: Frequências de todas classes

## B.7.2 Solução

A solução<sup>10</sup> selecionada apenas redimensionou as imagens na etapa de pré-processamento.

Na etapa seguinte foi utilizada a técnica de *transfer learning*, com o modelo *ResNet101*. Assim, o modelo da solução já foi previamente pré-treinado, e foi importado utilizando a biblioteca FastAI.

Durante a etapa de treinamento, foi utilizada da técnica de data augmentation. O modelo foi treinado em etapas: primeiro sem utilizar a técnica de *data augmentation*; depois utilizando a referida técnica, e além disso redimensionando as imagens mais uma vez.

## B.7.3 Resultado

A solução descrita obteve 0,34782 pontos na métrica *multi-class log loss*. Apesar de ter sido um bom resultado, a utilização de algumas outras técnicas poderiam gerar um resultado ainda melhor, como a utilização da técnica *ensemble*, pois como a base de teste é um subconjunto da base *ImageNet* [31], a utilização desta técnica com outros modelos pré-treinados em cima da mesma base poderia gerar um resultado melhor.

<sup>10</sup><https://www.kaggle.com/slambert1/fastai-dog-breeds>

## B.8 Plant Seedlings Classification

Conseguir classificar a espécie de uma planta representa uma potencial melhora nos rendimentos dos cultivos, bem como uma melhor administração do meio ambiente. Dito isso, o objetivo desta competição é justamente criar um modelo capaz de identificar a espécie de uma planta a partir de uma foto. A métrica escolhida pelos organizadores desta competição foi a micro-averaged F1 score.

### B.8.1 Base de dados

A base disponibilizada é composta por 4.750 fotos de 12 tipos diferentes de plantas, sendo que a distribuição de imagens por classe não é balanceada, e as classes possuem poucos dados para treinamento (em média, 395 imagens por classe). Além disso, as imagens não apresentam dimensões padronizadas, mas possuem um comportamento uniforme.

Tabela B.6: Imagens do Plant Seedlings Classification

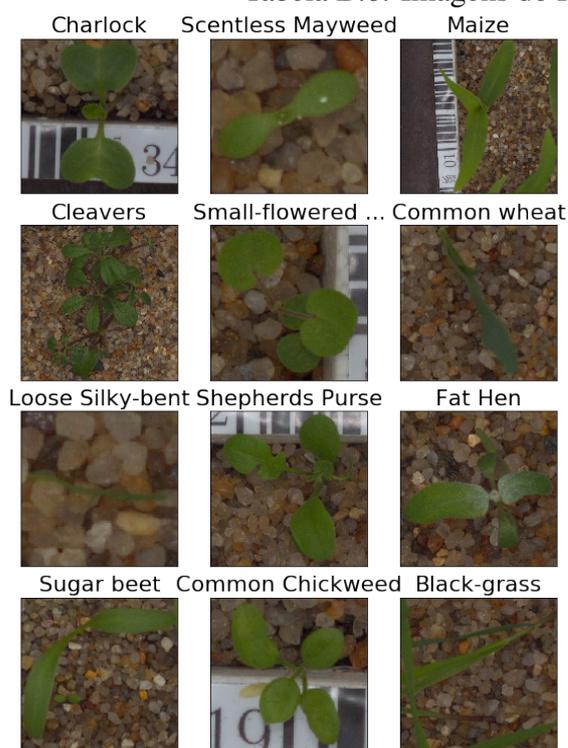


Figura B.13: Exemplos da base de dados

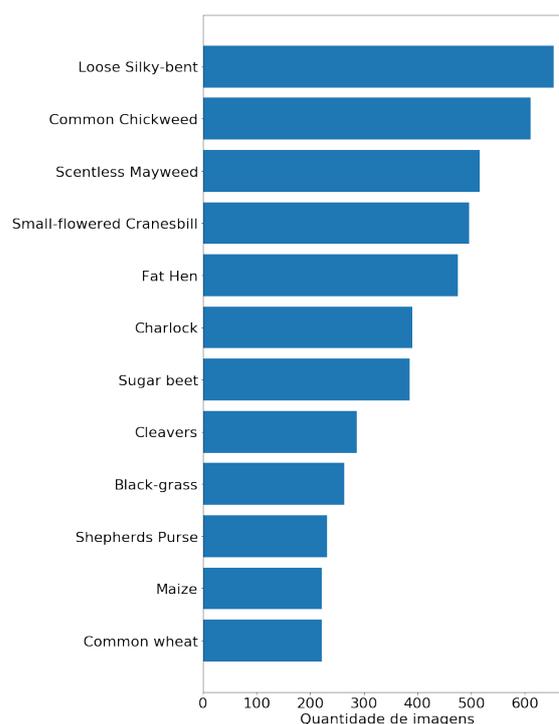


Figura B.14: Frequências de todas classes

## B.8.2 Solução

A solução<sup>11</sup> selecionada apresenta uma etapa de pré-processamento bem simples, onde o autor apenas normalizou os dados e redimensionou todas as imagens para as mesmas dimensões (299x299x3 pixels).

E na etapa de treinamento o autor optou por utilizar *transfer learning* com o modelo ResNet50, que utiliza ativações *ReLU* (ao longo do modelo) e *softmax* (na camada de saída), pré-treinado na base *ImageNet*. No entanto, o autor ainda executa um “ajuste fino” no modelo importado. Este treinamento é feito em ciclos de épocas variadas, onde é feito também um ajuste automático na taxa de aprendizagem. E para fazer a validação do treinamento foi utilizado 25% da base de treinamento.

## B.8.3 Resultado

A abordagem apresentada alcançou 0,9811 pontos na métrica *micro-averaged F1 score*.

## B.9 Understanding Clouds from Satellite Images

Entender o comportamento e organização das nuvens é um fator importante para compreender as mudanças climáticas. E visando facilitar esse processo, esta competição apresenta um problema do tipo segmentação de objetos, com 4 “alvos” (4 tipos de nuvens diferentes). A métrica de avaliação desta competição é a *dice coefficient*.

### B.9.1 Base de dados

A base de dados deste problema conta com 22.184 imagens de nuvens tiradas via satélite. Cada imagem apresenta de um à quatro dos tipos de nuvens. Todas as imagens tem dimensões 2.100x1.400 pixels e, devido ao tipo de problema, não possuem um comportamento uniforme.

### B.9.2 Solução

A solução<sup>12</sup> selecionada, na etapa de pré-processamento, apenas redimensionou e normalizou as imagens.

Este é um problema do tipo segmentação de objetos, mais precisamente, um problema onde deve ser encontrada a máscara (os pixels) que o objeto ocupa. Para isso, o modelo proposto foi dividido em duas partes.

---

<sup>11</sup><https://www.kaggle.com/droid021/fast-ai-v1>

<sup>12</sup><https://www.kaggle.com/cdeotte/without-ensemble-lb-0-665>

Tabela B.7: Imagens do Understanding Clouds from Satellite Images

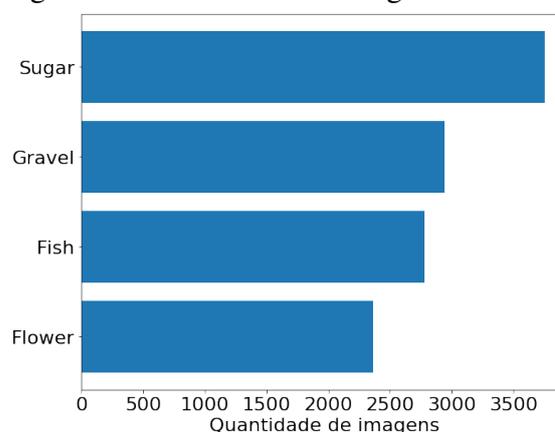
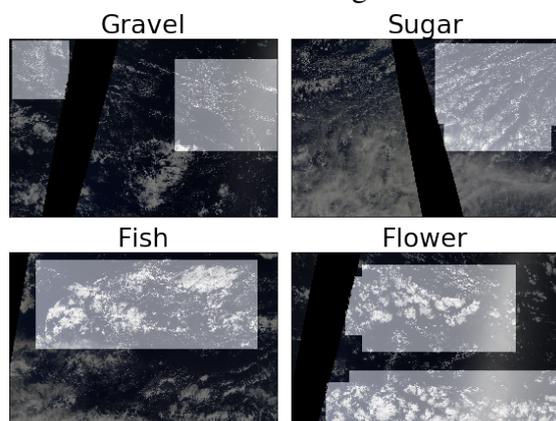


Figura B.15: Exemplos da base de dados com a máscara

Figura B.16: Frequências de todas classes



Figura B.17: Autoencoder

A primeira parte é responsável por classificar a imagem, indicando assim quais são os tipos de nuvens presentes na imagem com uma maior precisão. Para isto, o autor selecionou e combinou as classificações feitas por outros modelos encontrados no Kaggle.

A segunda parte é responsável por encontrar a máscara das nuvens na imagem, caso a primeira parte tenha classificado aquela imagem com algum tipo de nuvem. Para isso, foram treinados três modelos idênticos, esses modelos têm a arquitetura em U da rede *EfficientNetB2*, utilizaram *transfer learning* com pesos pré-treinados na base *ImageNet*, função de ativação *sigmoid* (na camada de saída) e função de perda *jaccard loss*.

Esse tipo de arquitetura é muito comum em problemas desse tipo, e utiliza a técnica conhecida como *autoencoder* (ver figura B.17). Nessa técnica, a rede neural apresenta menos neurônios nas camadas do meio, a fim de codificar os dados numa quantidade menor de variáveis. Similarmente, a rede apresenta na saída a mesma quantidade de neurônios da entrada. A rede neural é treinada de forma a conseguir gerar uma saída o mais próximo possível da entrada.

Durante a fase de treinamento, o autor utilizou as técnicas *data augmentation* e *K-Fold* com 3 dobras. Além disso utilizou os seguintes callbacks da biblioteca Keras: *EarlyStopping*, *ReduceLROnPlateau* e *ModelCheckpoint*. Este processo de treinamento foi aplicado aos três modelos.

Finalizando, o autor utilizou a técnica *test time augmentation* em cada um dos modelos e

combinou as predições resultantes, através de uma média.

### B.9.3 Resultado

A solução descrita obteve 0,66008 pontos na métrica *dice coefficient*, o que garantiu ao autor a 48ª posição no ranking privado.

## B.10 Recursion Cellular Image Classification

Entender a reação do organismo humano à certos medicamentos é uma parte fundamental do processo de criação de novos medicamentos. E visando agilizar esse processo de análise, os competidores desta competição devem criar um modelo capaz de classificar perturbações genéticas, dentre 1108 possíveis, a partir de imagens obtidas através de um microscópio. A métrica de avaliação desta competição é a *multi-class accuracy*.

### B.10.1 Base de dados

A base da competição possui 125.510 imagens, para 1.108 classes. Apesar das imagens terem dimensões padronizadas, comportamento uniforme e a base estar balanceada, esta apresenta poucos dados por classe.

### B.10.2 Solução

Na etapa de pré-processamento da solução<sup>13</sup> selecionada o autor constrói o que ele denominou de “*twin dataset*”, que se trata de um conjunto de imagens onde para cada imagem existe outra “gêmea” – imagem da mesma classe mas obtida de um experimento diferente. No processo de criação deste conjunto, todas as imagens são normalizadas e recortadas nas dimensões 256x256 pixels.

A rede construída é do tipo *squeeze excitation network*, que consiste basicamente de uma rede neural convolucional com uma leve modificação: ela adiciona pesos aos canais da rede, antes de passar a saída para a camada seguinte [32]. O modelo construído foi baseado na arquitetura *EfficientNet* e utilizou as seguintes camadas da biblioteca PyTorch: *BatchNorm2d*, *Conv2D*, *Flatten*, *MBCConv*, *Sigmoid* e *Swish*. E utiliza as funções de ativação *ReLU* e *swish*.

Na etapa de treinamento foi utilizada a técnica de *data augmentation* em cima do *twin dataset*. Além disso foram utilizadas taxas de aprendizagem variadas. E para fazer a validação do modelo foi utilizado 7% da base de treinamento.

Para finalizar, o autor utilizou a técnica *test time augmentation*, para dar uma leve melhoria nas predições finais do modelo.

<sup>13</sup><https://www.kaggle.com/hmendonca/fold1h4r3-arcenetb4-2-256px-rcic-lb-0-9759>

Tabela B.8: Imagens do Recursion Cellular Image Classification

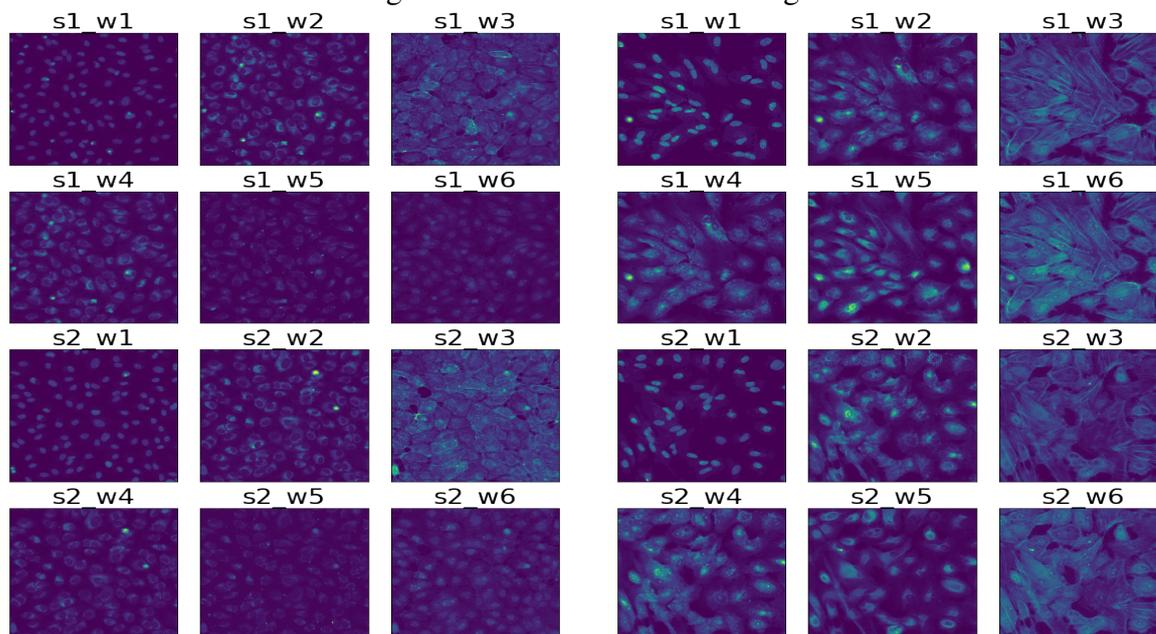


Figura B.18: Exemplo da base de dados (HUVEC-04 Plate4 G08, que apresenta classe:  $\text{sirna}_121$ )

Figura B.19: Exemplo da base de dados ((RPE-04 Plate3 F21, que apresenta classe:  $\text{sirna}_493$ )

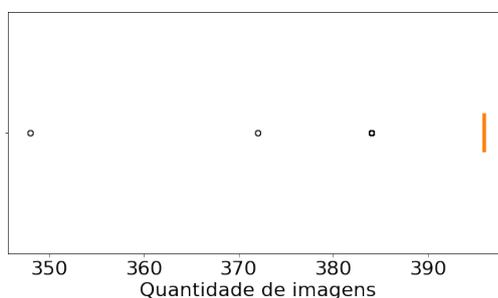


Figura B.20: Distribuição da base de dados

### B.10.3 Resultado

A solução apresentada alcançou 0,9759 pontos na métrica *multi-class accuracy*.

## B.11 Quick, Draw! Doodle Recognition Challenge

“Quick, Draw!” é um jogo educacional que tem como objetivo instruir as pessoas sobre o funcionamento da inteligência artificial. O objetivo desta competição é construir um modelo classificador melhor (mais preciso) do que o atualmente utilizado pelo jogo, que inclusive foi utilizado para criar a base de dados do problema (o que já é problemático, pois causou a existência de muito “ruído” na base: alguns dados foram classificados com uma categoria errada). O

classificador a ser construído deve ser capaz de prever 345 classes diferentes. As submissões da competição serão avaliadas pela métrica *mean average precision at 3*.

### B.11.1 Base de dados

A base é composta de 50 milhões de desenhos e foi anotada de modo automatizado utilizando o classificador do próprio jogo. A base contém 345 classes diferentes. Os desenhos que compõem a base consistem de vetores ordenados por data e hora. Os desenhos foram feitos na plataforma do próprio jogo, e por conta disso, elas possuem dimensões padronizadas e comportamento uniforme. Além disso a base é balanceada e possui uma quantidade muito significativa de exemplos por classe. Vale mencionar que o problema disponibiliza, na verdade, duas bases de dados, sendo a segunda uma versão simplificada da primeira, onde pontos desnecessários para a formação dos vetores são removidos. Então são duas bases diferentes mas contendo os mesmos desenhos.

### B.11.2 Solução

Na etapa de pré-processamento da solução<sup>14</sup> selecionada, para cada desenho o autor cria 3 imagens de tamanho 128x128x1 pixels com codificações diferentes e depois concatena as 3 imagens formando uma imagem única com dimensões 128x128x3 pixels – o primeiro canal codificado representa a presença de linhas, se um pixel intersecta um vetor então recebe o valor 255, caso contrário recebe 0; o segundo canal codifica os traços ao longo do tempo, os pixels do primeiro traço recebem valor 255, e esse valor vai decaindo em 13 pontos a cada próximo traço, tendo o valor mínimo em 125; e por fim, o terceiro canal codifica os pontos dos traços ao longo do tempo, de modo similar ao segundo canal, começando no valor 255 e sendo 20 o menor valor possível. Na figura ?? estão alguns exemplos da codificação descrita (retirados do notebook da solução), onde cada coluna representa a codificação de um canal e a última é a imagem RGB gerada pela concatenação das primeiras três.

O modelo utilizado foi o *MobileNet*, disponibilizado pela biblioteca Keras. Por baixo dos panos esse modelo é composto das seguintes camadas: *Activation*, *BatchNormalization*, *Conv2D*, *Dense*, *DepthwiseConv2D*, *Dropout*, *Flatten*, *MaxPooling2D*, *ReLU*, *Reshape*, *ZeroPadding2D*. E utiliza as funções de ativação *ReLU*, ao longo da rede, e *softmax*, na camada de saída. Além disso, a função de perda utilizada foi a *categorical cross-entropy*.

Na etapa de treinamento o autor utilizou a técnica *data augmentation* para inverter as imagens horizontalmente com probabilidade de 50%. O autor também utilizou o *callback ModelCheckpoint*, para salvar o progresso do treinamento de tempos em tempos. E também utilizou o *callback ReduceLROnPlateau*, que serve para modificar a taxa de aprendizagem durante o treinamento.

<sup>14</sup><https://www.kaggle.com/echomil/mobilenet-126x126x3-100k-per-class>

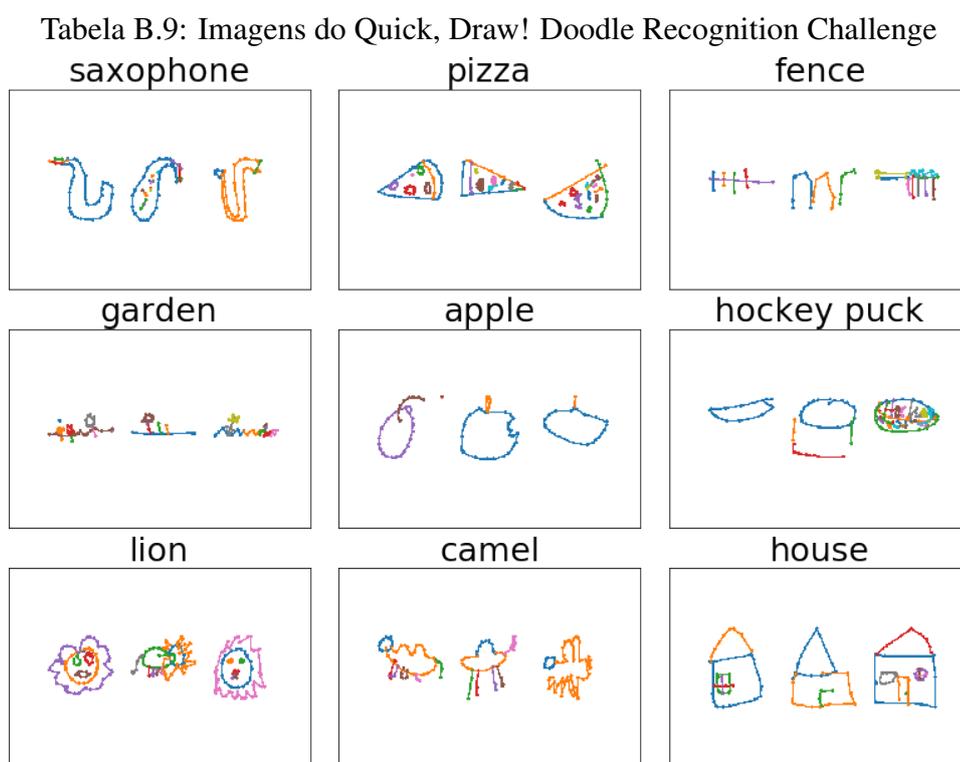


Figura B.21: 3 exemplos de desenho para alguns exemplos de classes

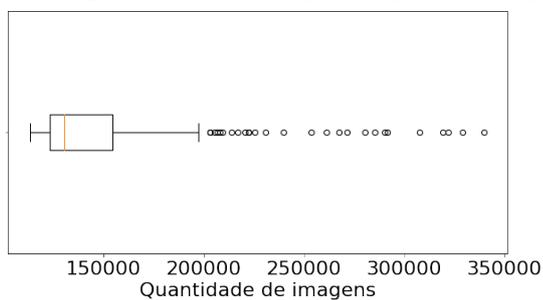


Figura B.22: Distribuição da base de dados

### B.11.3 Resultado

A solução apresentada obteve 0,923 pontos na métrica *mean average precision at 3*.

## B.12 State Farm Distracted Driver Detection

Motoristas desatentos estão mais propensos a causar acidentes de trânsito. E pensando nisso, esta competição busca um modelo classificador capaz de dizer se o(a) motorista presente numa foto está dirigindo atentamente ou se ele(a) está com sua atenção voltada a alguma outra coisa (no total existem 10 classes, ou “estados de atenção”). Os modelos serão avaliados pela métrica *log loss*.

### B.12.1 Base de dados

Esta base de dados possui cerca mais de 22.424 imagens, e são bem padronizadas: possuem as mesmas dimensões e seguem um padrão de posicionamento. Além disso, a base é balanceada e possui uma quantidade suficiente de dados para treinamento.

Tabela B.10: Imagens do State Farm Distracted Driver Detection



Figura B.23: Exemplos da base de dados

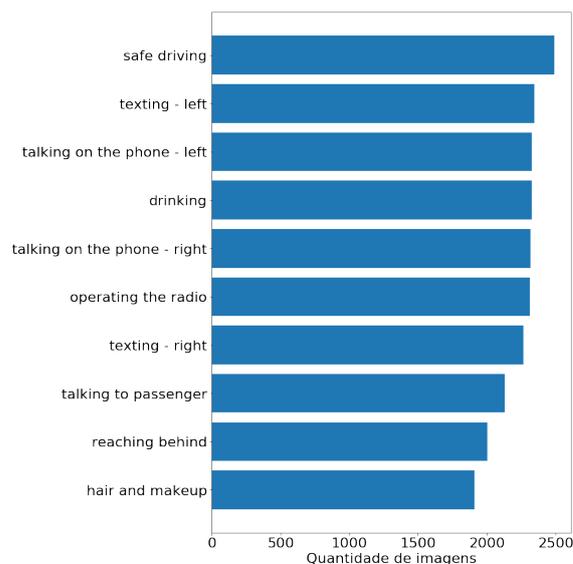


Figura B.24: Distribuição da base de dados

### B.12.2 Solução

A solução<sup>15</sup> selecionada, não fez nenhum tratamento na etapa de pré-processamento.

Para o modelo, o autor escolheu utilizar a técnica de *transfer learning* com a rede *ResNet34* pré-treinada em cima da base *ImageNet*.

Já durante a fase de treinamento, foi utilizada a técnica de *data augmentation* mas sem a transformação de inversão horizontal, visto que em algumas das classes é importante saber o lado em que o motorista está posicionado. Além disso foi separado 10% da base de treinamento para validação do modelo. E com isso, foi executado um breve treinamento para ajustar a *head* do modelo à base atual. E em seguida, toda a rede foi “descongelada” para um novo treinamento breve ser executado para ajustar a rede à base atual.

### B.12.3 Resultado

Com esta solução relativamente simples, foi possível alcançar uma pontuação de 0,27833 na métrica *log loss*.

<sup>15</sup><https://www.kaggle.com/byronkd/distracted-driver-fastai-easy-top-15-of-1b>

## B.13 Kannada MNIST

O objetivo desta competição é o mesmo da *Digit Recognizer*, mas no lugar de dígitos arábicos, serão fornecidos dígitos kannada. Kannada é uma linguagem falada por aproximadamente 45 milhões de pessoas do sudoeste indiano. Apesar do objetivo semelhante (até em quantidade de símbolos) e relativamente simples, é preciso atentar-se, pois existe uma quantidade considerável de símbolos parecidos representando dígitos diferentes. Assim, esta competição apresenta um problema do tipo classificador com 10 classes. E as submissões serão avaliadas pela acurácia.

### B.13.1 Base de dados

A base de dados [33] é balanceada, apresenta 60.000 imagens de dimensões 28x28 pixels em escala de cinzas e com comportamento uniforme (os símbolos aparecem no centro das imagens). Além disso, a quantidade de imagens é suficiente para executar um treinamento decente.

Tabela B.11: Imagens do Kannada MNIST

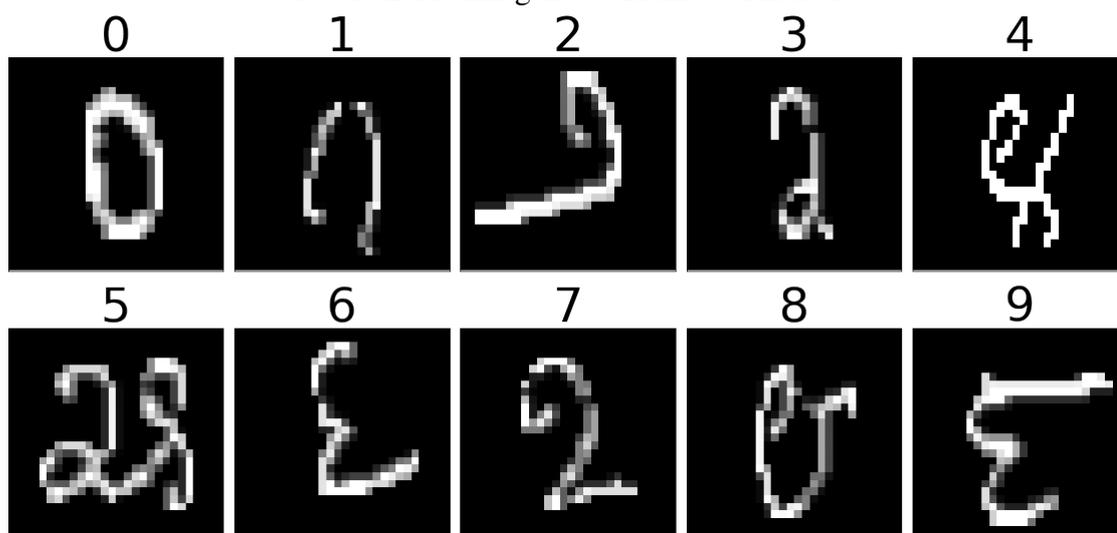


Figura B.25: Exemplos da base de dados

### B.13.2 Solução

A solução<sup>16</sup> selecionada apresenta não só um desempenho que fica entre os 2% melhores do placar, como também uma explicação detalhada de tudo que foi utilizado.

Durante a fase de pré-processamento, como a base de dados é bastante padronizada e de alta qualidade: o autor apenas normalizou todas as imagens.

Em seguida, o autor criou um modelo customizado utilizando as seguintes camadas da biblioteca Keras: *Conv2D*, *BatchNormalization*, *MaxPooling2D*, *Dropout*, *Flatten* e *Dense*. Na

<sup>16</sup><https://www.kaggle.com/benanakca/kannada-mnist-cnn-tutorial-with-app-top-2>

maioria das camadas, foi utilizada a função de ativação *LeakyReLU*, no resto delas não foi utilizada nenhuma função de ativação, com exceção da camada de saída, que utiliza a função *softmax*. A função de perda escolhida para o modelo foi a *categorical cross-entropy*.

Durante a fase de treinamento, o autor separou 20% da base para validação e preparou um gerador de dados, utilizando a técnica de *data augmentation*, com as seguintes transformações: rotação, translação, cisalhamento e zoom. A fim de diminuir o sobreajuste, foi utilizado um *callback* para reduzir a taxa de aprendizagem e outro para parar o treinamento no momento em que o modelo parou de melhorar (utilizando a função de perda da validação como métrica de melhora).

### B.13.3 Resultado

Seguindo estes passos, esta solução conseguiu 0,991 pontos de acurácia e ficou na 33ª posição.

## B.14 Leaf Classification

Existem aproximadamente meio milhão de espécies de planta no mundo. Historicamente, classificar as espécies corretamente é um problema complexo. E nesta competição o objetivo é criar um modelo classificador capaz de indicar a espécie (dentre 99 possíveis) de uma planta através de uma foto da mesma. No entanto, esta competição fornece um conjunto de características relacionadas a cada imagem, e espera-se que os competidores estendam o conjunto de características de forma a melhorar a classificação. A métrica de avaliação desta competição é a *log loss*.

### B.14.1 Base de dados

A base consiste de 1.584 imagens, e cada uma foi convertida para imagens “binárias”, onde os pixels da folha foram coloridos de branco e os pixels do fundo de preto. Somado a isso, cada imagem vem acompanhada de um conjunto de 3 características extraídas: um descritor contíguo de forma, um histograma de textura interior e um histograma de margem em escala precisa. Apesar das imagens não terem dimensões padronizadas, o comportamento é uniforme, as folhas estão localizadas no centro das imagens. Além disso, a base é balanceada, mas só tem 10 imagens por espécime, ou seja, tem pouquíssimos dados por classe.

### B.14.2 Solução

Na etapa de pré-processamento da solução<sup>17</sup> selecionada o autor normaliza e aplica uma escala em todas as imagens, para que o lado maior de cada imagem fique no tamanho de 96 pixels.

<sup>17</sup><https://www.kaggle.com/abhmul/keras-convnet-lb-0-0052-w-visualization>

Tabela B.12: Imagens do Leaf Classification

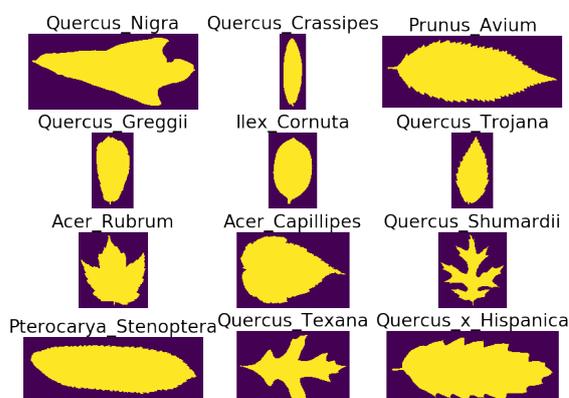


Figura B.26: Exemplos de algumas classes da base de dados

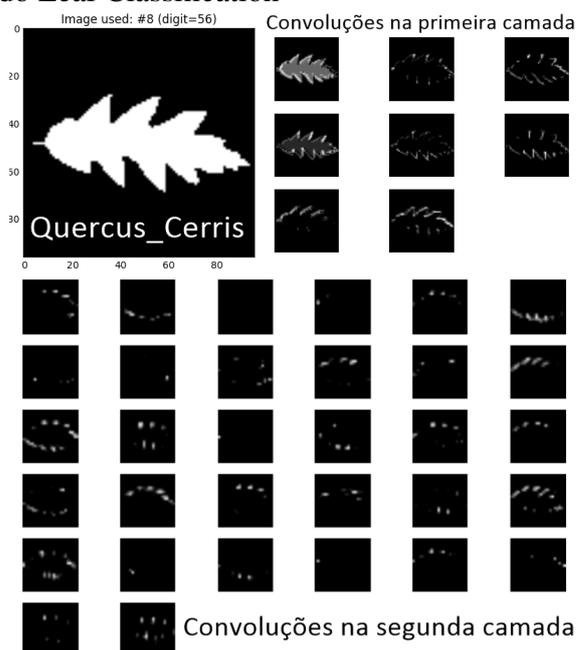


Figura B.27: Convoluções da solução

E antes de passar para o modelo, as imagens são redimensionadas para as dimensões 96x96x1 pixels.

O modelo proposto nesta solução consiste na junção de dois modelos diferentes: uma rede neural convolucional e um MLP (*multilayer perceptron*). O primeiro modelo é composto das seguintes camadas da biblioteca Keras: *Activation*, *Conv2D*, *Flatten*, *MaxPooling2D*. E utiliza a função de ativação *ReLU*. Em seguida a saída da rede convolucional é concatenada com as características pré-extraídas, disponibilizadas na base de dados. E esses dados concatenados são passados para um MLP composto de uma camada *Dense* e outra *Dropout*. Por fim, a saída deste MLP passa para uma camada *Dense* de 99 nós e função de ativação *softmax* para fazer a predição final. A função de perda utilizada no modelo foi a *categorical cross-entropy*.

No processo de treinamento foi utilizada a técnica de *data augmentation* para aplicar as seguintes transformações: rotação, zoom, inversão horizontal e vertical. E a técnica usada para validação foi a *stratified shuffle split*, com 10% da base de treinamento sendo reservada para validação.

### B.14.3 Resultado

A solução apresentada obteve 0,009 pontos na métrica *log loss*. Apesar desta ter sido a pontuação do notebook, o autor afirma que seu melhor desempenho, que o levou para a 37ª posição do ranking público, foi obtido usando essa mesma abordagem. E que devido a aleatoriedade de certas partes da solução, este resultado não se repetiu ao submeter o notebook.

## B.15 Invasive Species Monitoring

Na natureza existem várias espécies invasivas de plantas, animais, insetos, etc. Essas espécies podem ter efeitos prejudiciais para o meio ambiente, economia e até à saúde humana. Contudo, o custo para localizar e acompanhar essas espécies é alto. Então, buscando baratear esse custo, o objetivo desta competição é criar um modelo classificador que seja capaz de indicar se existe, ou não, uma espécie invasiva (hortênsia) numa foto tirada dentro de uma floresta. Os modelos serão avaliados através da métrica “área abaixo da curva ROC”.

### B.15.1 Base de dados

Esta base possui 2.295 imagens tiradas numa floresta brasileira. Todas as imagens possuem as mesmas dimensões, mas não apresentam nenhum padrão de posicionamento. A base está desbalanceada, pois 1.448 das imagens apresentam hortênsias e as outras 847 não. Além disso, a base apresenta poucos dados para a classe negativa.

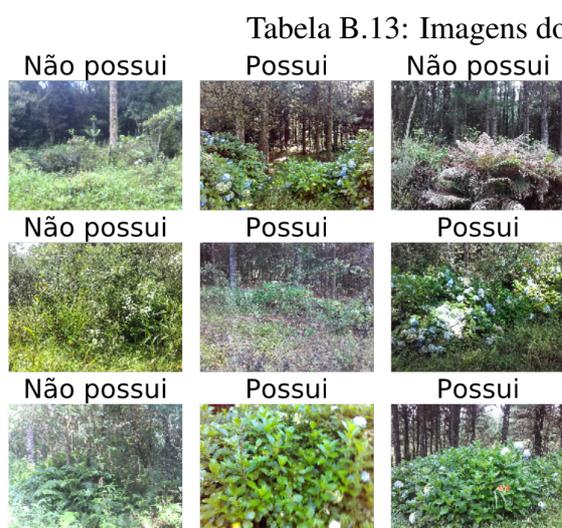


Figura B.28: Exemplos da base de dados

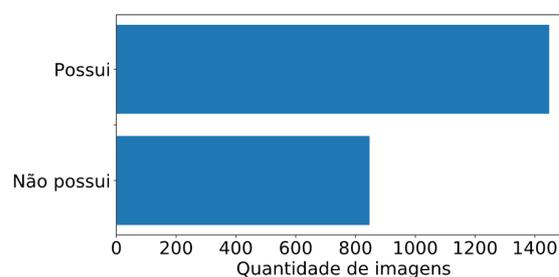


Figura B.29: Distribuição da base de dados

### B.15.2 Solução

Na solução<sup>18</sup> selecionada, durante a fase de pré-processamento o autor apenas normalizou e redimensionou todas as imagens para as dimensões 800x800x3 pixels.

Para o modelo proposto, foi utilizada a técnica de *transfer learning*. O modelo pré-treinado escolhido foi o *InceptionResNetV2*, com os pesos do pré-treinamento executado sobre a base *ImageNet*. Somado a isso, o autor adicionou uma camada *Dropout* seguida de uma camada

<sup>18</sup><https://www.kaggle.com/leesangju92/ism-inceptionresnetv2>

*Dense* com função de ativação *sigmoid*. Como função de perda, foi utilizada a *binary cross-entropy*.

Durante o treinamento, o autor utilizou um gerador de dados para aplicar a técnica de *data augmentation*, com rotação até 270° e inversão horizontal. Também foi separado 2% da base para validação do modelo. E então o modelo foi treinado de forma tradicional.

### B.15.3 Resultado

Com esta solução relativamente simples, o autor conseguiu uma pontuação de 0,99348 na métrica “área abaixo da curva ROC”.

## B.16 RSNA Intracranial Hemorrhage Detection

Hemorragia intra-cranial é o sangramento que ocorre dentro do crânio, e é um sério problema de saúde que leva à morte milhares de pessoas todos os anos. Conseguir identificar a posição e o tipo da hemorragia é um passo fundamental no tratamento de um paciente. Porém, esse processo demanda tempo e só pode ser feito por médicos altamente treinados. Assim, esta competição apresenta um problema do tipo classificador, e seu desafio é criar um modelo que seja capaz de detectar se existe uma hemorragia intra-cranial aguda (e seus 6 subtipos) numa radiografia. A métrica de avaliação das submissões desta competição é a *weighted multilabel logarithm loss*.

### B.16.1 Base de dados

A base contém 752.803 radiografias, com dimensões e comportamento padronizados. No entanto, apesar da base apresentar dados suficientes para executar o treinamento, ela não está balanceada.

### B.16.2 Solução

A solução<sup>19</sup> selecionada apresenta uma etapa de pré-processamento bem direta, consistindo basicamente de um redimensionamento de todas as imagens para as dimensões 256x256x3 pixels e um tratamento para aproximadamente 120 imagens que foram disponibilizadas numa configuração destoante do resto da base. E por fim, as imagens duplicadas são removidas.

O modelo escolhido se trata de um *EfficientNetB3* somado a uma camada *Dense* com 6 nós e função de ativação *sigmoid*. Nesta solução, a técnica de *transfer learning* foi utilizada no modelo citado, os pesos utilizados foram obtidos através de um pré-treinamento em cima da base *ImageNet*. A função de perda utilizada no modelo foi a *binary cross-entropy*.

<sup>19</sup><https://www.kaggle.com/krishnakatyal/keras-efficientnet-b3>

Tabela B.14: Imagens do RSNA Intracranial Hemorrhage Detection

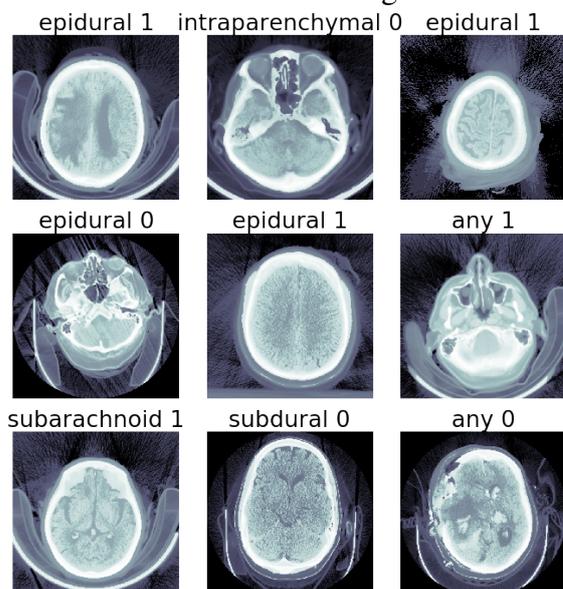


Figura B.30: Exemplos da base de dados, o título da imagem apresenta o tipo e um número (1 se possui hemorragia, 0 se não possui)

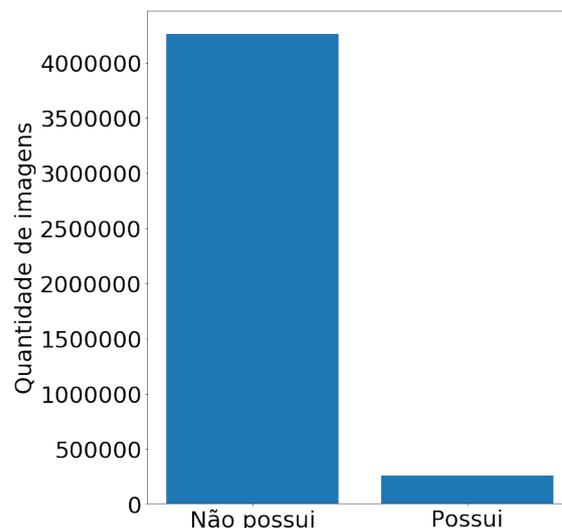


Figura B.31: Distribuição da base de dados

Na fase de treinamento, apesar de ter sido utilizada a técnica de *transfer learning*, o autor executa um treinamento na rede inteira. Para isso, foram utilizadas as técnicas *oversampling*, para compensar a base desbalanceada, e *data augmentation*, para executar as seguintes transformações nas imagens: recortar, inverter horizontal e verticalmente. E para fazer a validação do modelo, foi utilizada a abordagem *multilabel stratified shuffle split*, onde 10% do conjunto de treinamento foi reservado para validação.

### B.16.3 Resultado

Com isso, a solução apresentada obteve 0,076 pontos na métrica *weighted multilabel logarithm loss*.

## B.17 TGS Salt Identification Challenge

Em áreas da Terra com grandes acúmulos de óleo e gás, existe uma grande chance de também existir grandes depósitos de sal debaixo da superfície. Porém, descobrir a posição exata requer humanos especialistas no assunto. Além dos benefícios econômicos, essas reservas também apresentam perigos para as escavações de óleo e gás. Com isto em mente, o objetivo dessa competição é a criação de um modelo que seja capaz de segmentar a imagem em partes que

contém sal. Os modelos serão avaliados pela métrica *mean average precision at different intersection over union*, mas não apenas ela, pois será calculada também a precisão média alcançada por essa métrica sobre diferentes limiares.

### B.17.1 Base de dados

Esta base de dados possui 4.000 imagens, cada imagem tem uma profundidade associada. Todas as imagens têm as mesmas dimensões, mas não seguem um padrão de posicionamento. Além disso, a base está balanceada e possui dados suficientes para executar o treinamento.

Tabela B.15: Imagens do TGS Salt Identification Challenge

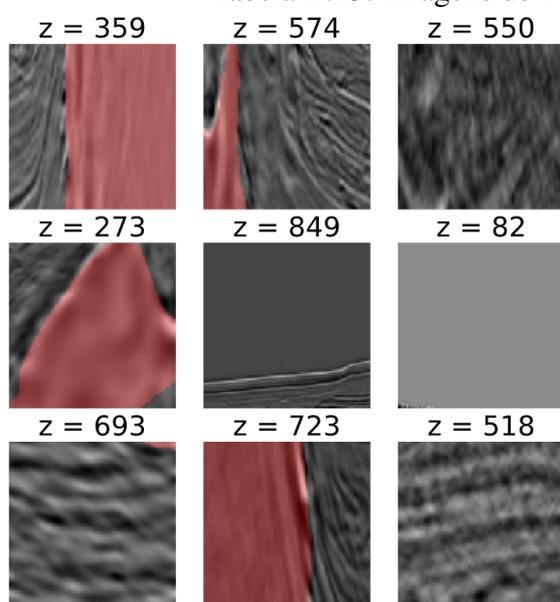


Figura B.32: Exemplos da base de dados com a máscara em vermelho, o título da imagem indica a profundidade da imagem

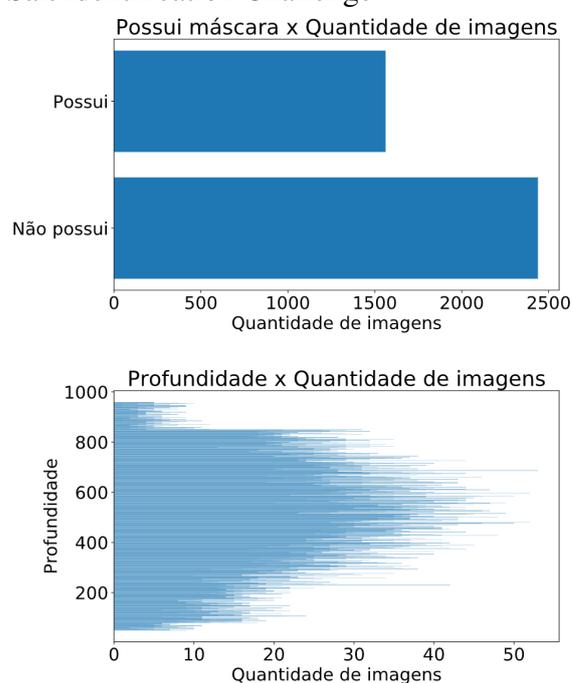


Figura B.33: Distribuição da base de dados

### B.17.2 Solução

A solução<sup>20</sup> selecionada, durante a fase de pré-processamento, apenas normalizou e redimensionou as imagens para as dimensões 128x128 pixels.

Para o modelo, o autor escolheu utilizar *transfer learning*. No começo do modelo, utilizou-se a rede *Xception* pré-treinado em cima da base *ImageNet*, em seguida, para realizar o *over-sampling* (segunda parte do *decoder*, foram utilizadas as seguintes camadas: *MinPooling2D*, *Conv2D*, *BatchNormalization*, *Conv2DTranspose*, *Dropout* e *ZeroPadding2D*). Também foram feitas algumas concatenações ao longo do *decoder*, como descrito no artigo de construção da

<sup>20</sup><https://www.kaggle.com/meaninglesslives/getting-0-87-on-private-lb-using-kaggle-kernel>

U-Net [34]. Ao longo do *decoder* foram utilizadas duas funções de ativação, *LeakyReLU* e identidade, com exceção da última camada que utilizou a função *sigmoid*. Como função de perda, foi utilizada a *binary cross-entropy* combinada com a *dice loss*.

Na etapa de treinamento, foi separado 20% da base para validação do modelo. Além disso, o autor utilizou dados externos, mais precisamente, ele utilizou a técnica de *pseudo labeling*. Para esta técnica, o autor selecionou três modelos de outros competidores, e quando a segmentação predita pelos três era aproximadamente a mesma: ele adicionava essa imagem e essa segmentação na base de treinamento. O autor também utilizou a técnica de *data augmentation* apenas com a transformação de inversão horizontal. O modelo foi treinado utilizando o um *callback* para diminuir a taxa de aprendizado ao longo do treinamento.

Durante a segmentação das imagens, o autor testou e escolheu o melhor dentre diversos limiares para decidir o valor necessário para os pixels preditos pelo modelo.

### B.17.3 Resultado

Com isso, esta solução conseguiu uma pontuação de 0,854 na métrica *mean average precision at different intersection over union*.

## B.18 Severstal: Steel Defect Detection

O aço é um material muito resistente e um dos mais importantes materiais de construção nos tempos atuais. E para ajudar a tornar a produção de aço mais eficiente, o desafio desta competição é fazer um modelo capaz de apontar a localização e a forma de 4 tipos diferentes de falhas/defeitos no aço a partir de uma foto da superfície do mesmo. Para avaliar as submissões, a métrica desta competição é a *mean dice coefficient*.

### B.18.1 Base de dados

A base contém 6.666 imagens, todas com dimensões padronizadas. Mas por se tratar de um problema de segmentação de objetos, o comportamento dos dados não é uniforme. Além disso a base é bem desbalanceada e somente uma classe possui dados suficientes para o treinamento.

### B.18.2 Solução

Pelo que está presente no notebook<sup>21</sup> selecionado, a etapa de pré-processamento parece consistir de um redimensionamento de todas as imagens para as dimensões 224x224x3 pixels, normalização e inversão horizontal (com 50% de chance).

<sup>21</sup><https://www.kaggle.com/khornlund/fork-of-sever-ensemble-3>

Tabela B.16: Imagens do Severstal: Steel Defect Detection

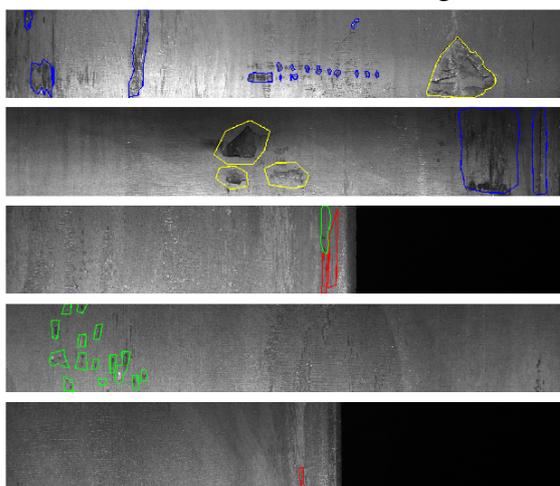


Figura B.34: Exemplos da base de dados com os defeitos circulosados, seguindo o mapeamento de cor: 1 - Verde, 2 - Vermelho, 3 - Azul, 4 - Amarelo

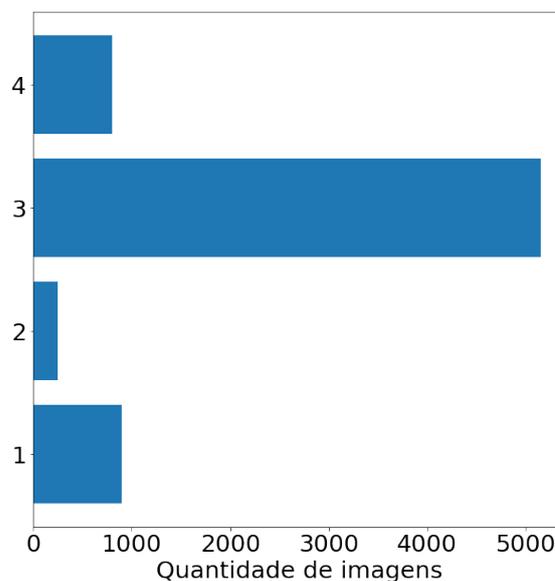


Figura B.35: Distribuição da base de dados

Para a construção do modelo foi utilizada a técnica de *ensemble* com oito modelos diferentes e pré-treinados pelo autor, fora deste notebook. Os oito modelos são variações das arquiteturas *EfficientNet*, *Inception* e *ResNeXt*. O autor não faz nenhum comentário sobre o treinamento desses modelos, mas alguns trechos do código parecem indicar que foi utilizada a técnica de *transfer learning* com pesos de modelos pré-treinados na base *ImageNet*.

Ou seja, neste notebook nenhum treinamento é executado, o autor apenas importa os modelos pré-treinados, executa o *ensemble*, faz as predições e finaliza executando uma etapa de pós-processamento nas máscaras geradas.

O pós-processamento consiste em zerar os pixels, em cada máscara, que estiverem abaixo de um certo limiar (0,5).

### B.18.3 Resultado

Com esta solução o autor obteve 0,9 pontos na métrica *mean dice coefficient* e ficou em 55º lugar no ranking privado.

## B.19 APTOS 2019 Blindness Detection

Retinopatia diabética é uma complicação do diabetes e pode vir a causar cegueira no enfermo. O objetivo dessa competição é construir um modelo que consiga detectar a condição e sua intensidade (dentre 5 possíveis) através de uma foto do olho do paciente, com a finalidade de

detectar cedo e de forma fácil a presença dessa condição no paciente afetado. Os modelos serão avaliados pelo “coeficiente de Kappa ponderado quadrático”.

### B.19.1 Base de dados

Esta base apresenta 3.662 imagens. Elas não possuem as mesmas dimensões entre si, mas o olho está bem centralizado em todas elas (comportamento uniforme). Além disso, a base é desbalanceada e algumas das classes possuem poucas imagens.

Tabela B.17: Imagens do APTOS 2019 Blindness Detection

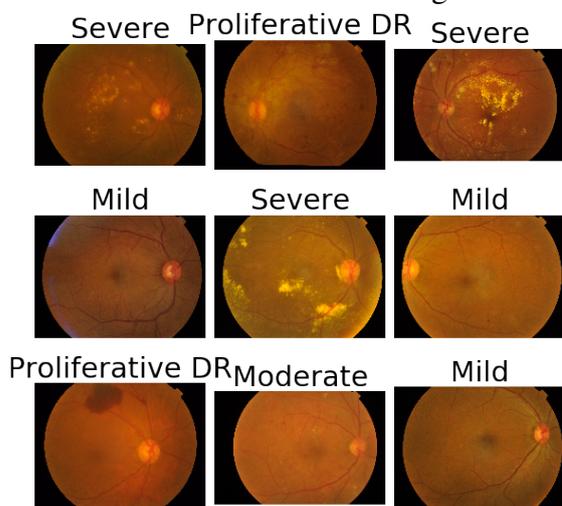


Figura B.36: Exemplos da base de dados

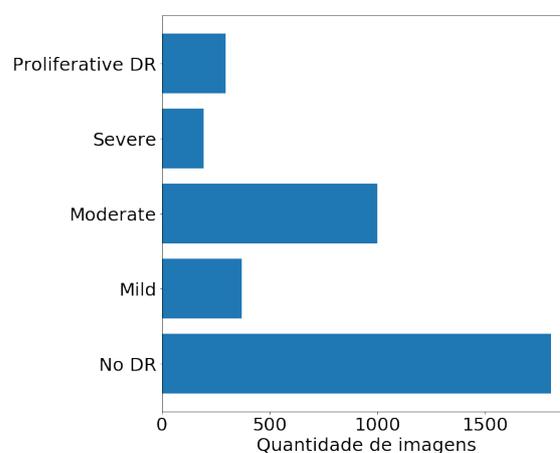


Figura B.37: Distribuição da base de dados

### B.19.2 Solução

A solução<sup>22</sup> selecionada é uma combinação de três soluções. Durante a fase de pré-processamento, todas as imagens foram redimensionadas (de forma que encaixassem corretamente nos modelos) e normalizadas nas três soluções. Em uma dessas soluções, aplicou-se desfoque gaussiano nas imagens.

As três soluções utilizaram redes já consolidadas na literatura, sendo duas delas a rede *EfficientNetB3* (pré-treinadas) e a outra a *DenseNet* (sem pré-treinamento).

Durante a fase de treinamento, foram utilizadas imagens de uma base de dados do mesmo problema mas de outro ano. Além disso, foi utilizada a técnica de *data augmentation* com transformações de: inversão horizontal, mudança de brilho, contraste e rotação de até 360°.

Na etapa de predição, as soluções que utilizaram a rede *EfficientNetB3* utilizaram a técnica de *test time augmentation*. As predições das três soluções foram combinadas através de uma

<sup>22</sup><https://www.kaggle.com/lextoumbourou/2-x-b3-densenet201-blend-0-926-on-private-lb/output?scriptVersionId=20305572>

média ponderada, gerando com isso a resposta final.

### B.19.3 Resultado

Seguindo estes passos, esta solução conseguiu uma pontuação de 0,924 no “coeficiente de Kappa ponderado quadrático”, alcançando 62<sup>a</sup> posição no placar.

## B.20 Airbus Ship Detection Challenge

Conforme cresce o tráfego marítimo, também aumentam-se as chances de infrações sobre o mar, como por exemplo pesca ilegal, tráfico de drogas, acidentes de navios com potencial de grandes danos ambientais, etc. Por conta disso, esta competição tem como objetivo a criação de um modelo que consiga encontrar e marcar (segmentar a imagem) todos os navios presentes numa foto. A métrica de avaliação desta competição é a *F2 score at different intersection over union*.

### B.20.1 Base de dados

Esta base de dados possui 192.556 imagens, todas com as mesmas dimensões (768x768). No entanto, as imagens não possuem nenhum padrão de posicionamento, várias até apresentam partes de cidades junto do mar. Além disso, a base não está balanceada mas apresenta dados suficientes para treinamento.

Tabela B.18: Imagens do Airbus Ship Detection Challenge

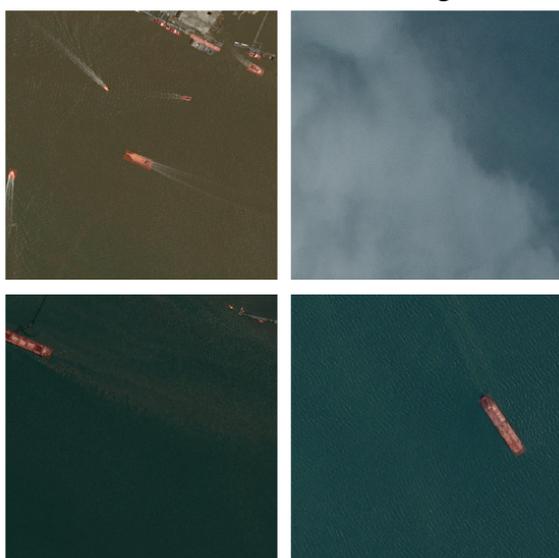


Figura B.38: Exemplos da base de dados

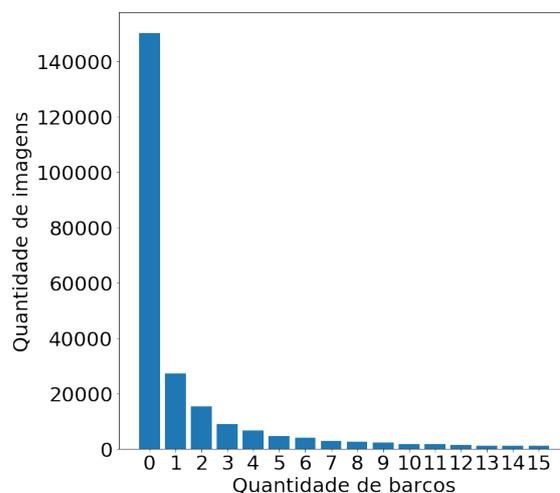


Figura B.39: Distribuição da base de dados

## B.20.2 Solução

A solução<sup>23</sup> selecionada é uma combinação de outras duas soluções<sup>24 25</sup>. Aqui não foi feito nenhum pré-processamento.

A solução utiliza dois modelos, um para segmentação e outro para classificação (detectar se existe pelo menos um navio na imagem). O modelo de segmentação é uma *ResNet34* pré-treinada e em forma de U (*U-net*). O modelo de classificação também é uma *ResNet34* pré-treinada, mas não em forma de U.

Para o treinamento do modelo de segmentação, foi separado 5% da base para validação. Também foi utilizada a técnica de *data augmentation* com transformações de: rotação (de até 20°), rotações diedrais (múltiplas de 90°) e iluminação randômica. A função de perda do modelo foi uma combinação ponderada da *dice loss* com a *focal loss*. Para aumentar a velocidade do treinamento a seguinte estratégia foi adotada. O modelo foi treinado algumas vezes, a cada treinamento a resolução das imagens era reduzida, diminuída a quantidade de épocas (iterações) e diminuída a taxa de aprendizagem (por limitação de tempo do Kaggle, o autor só treinou até a resolução 384x384). O primeiro desses treinamentos foi feito apenas nas camadas do *decoder*.

Para o treinamento do modelo de classificação, utilizou-se a mesma estratégia descrita acima, porém ao invés de treinar apenas as camadas do *decoder*, o primeiro dos treinamentos foi feito apenas nas camadas de predição.

Assim, foi utilizado o modelo de segmentação apenas em imagens classificadas como “possui navio”, segundo o modelo de classificação.

## B.20.3 Resultado

Com isso, foi possível alcançar uma pontuação de 0,889 na métrica *F2 score at different intersection over union*, ultrapassando até mesmo a pontuação do primeiro colocado (vale ressaltar que é possível conseguir um resultado ainda melhor que esse, mas o tempo limite de execução do notebook no Kaggle não permite).

## B.21 Ultrasound Nerve Segmentation

Inevitavelmente, todo processo cirúrgico envolve algum tipo de desconforto, em muitos casos, inclusive, dores fortes no pós-cirúrgico. Em todo caso, as dores geralmente são tratadas com o uso de narcóticos que trazem alguns efeitos colaterais indesejados. Pensando nisso, o objetivo desta competição é identificar e apontar com precisão estruturas de nervos em ultrassonografias, para auxiliar num processo de inserção de cateteres internos que bloqueiam ou suavizam a dor diretamente na origem. A métrica de avaliação desta competição é a *mean dice coefficient*.

<sup>23</sup><https://www.kaggle.com/ashishpatel26/unet34-submission>

<sup>24</sup><https://www.kaggle.com/iafoss/fine-tuning-resnet34-on-ship-detection/notebook>

<sup>25</sup><https://www.kaggle.com/iafoss/unet34-dice-0-87/notebook>

### B.21.1 Base de dados

A base de dados contém 5.635 imagens, todas com dimensões padronizadas, mas comportamento não uniforme (por se tratar de um problema de detecção de objetos). E por só ter um “alvo”, a base pode ser considerada balanceada e possui dados suficientes para o treinamento.

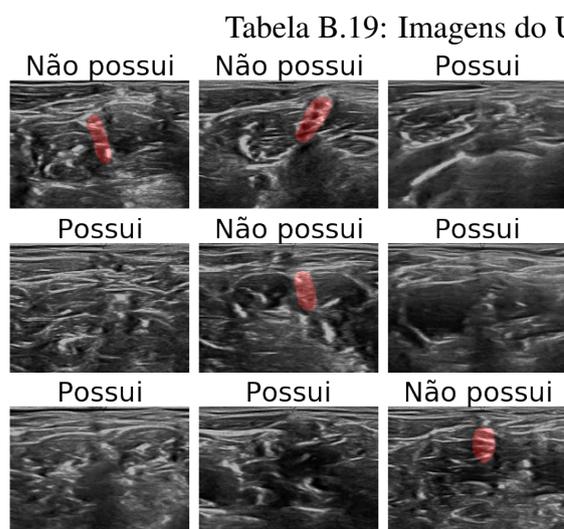


Figura B.40: Exemplos da base de dados

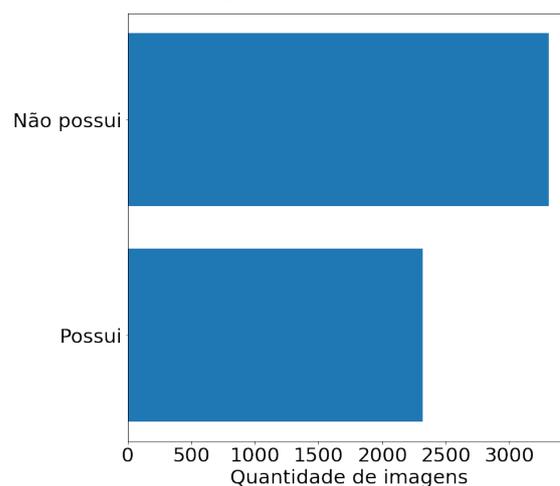


Figura B.41: Distribuição da base de dados

### B.21.2 Solução

Na solução<sup>26</sup> selecionada, a etapa de pré-processamento consiste basicamente em redimensionar todas as imagens para as dimensões 112x80x1 pixels e normalizá-las.

O modelo proposto é uma customização da arquitetura *U-Net*, utilizando as seguintes camadas da biblioteca Keras: *Activation*, *BatchNormalization*, *Conv2D*, *Dropout*, *Flatten*, *Max-Pooling2D*. Além disso, o modelo utilizou as funções de ativação *ReLU*, ao longo da rede, e *sigmoid*, na camada de saída. As funções de perda utilizadas foram a *weighted binary cross-entropy* e a *dice loss*.

No processo de treinamento foi utilizada a divisão padrão da base, uma parte para treinamento e outra para validação, sendo 20% da base de treinamento reservada para a validação do modelo.

### B.21.3 Resultado

A solução descrita obteve 0,66586 pontos na métrica *mean dice coefficient*.

<sup>26</sup><https://www.kaggle.com/gbatchkala/urss-final-code-script>

## B.22 Planet: Understanding the Amazon from Space

O desmatamento da floresta Amazônica causa muita perda da biodiversidade, contribui para a mudança climática, perdas de habitats, e outros efeitos devastadores no meio ambiente. Atualmente, existem diversos satélites capturando fotos da floresta para identificar onde está ocorrendo desmatamento. Porém as imagens têm baixa resolução, e nessas imagens, os métodos de classificação não conseguem diferenciar entre devastação humana e devastação por causas naturais. Dito isso, nesta competição, deverá ser criado um modelo classificador que seja capaz de identificar todas as classes (dentre 449 possíveis) que estão presentes numa imagem. A métrica de avaliação desta competição é a *mean F2 score*.

### B.22.1 Base de dados

A base contém 40.479 fotos aéreas, todas com as mesmas dimensões e comportamento uniforme. Apesar da base conter dados suficientes para cada classe, ela está desbalanceada.

Tabela B.20: Imagens do Planet: Understanding the Amazon from Space



Figura B.42: Exemplos da base de dados

### B.22.2 Solução

A etapa de pré-processamento da solução<sup>27</sup> selecionada consiste em redimensionar todas as imagens para as dimensões 128x128 pixels, preservando a quantidade de canais (3) e normalizá-las.

Foi utilizado o modelo *DenseNet121*, pré-treinado na base *ImageNet*, disponibilizado pela biblioteca Pytorch. Apesar do autor ter utilizado a técnica de *transfer learning*, ele executou um treinamento em toda a rede, para ajustá-la aos dados do problema.

Este treinamento foi feito em ciclos e foi utilizada a técnica de *data augmentation* para executar as seguintes transformações: inverter verticalmente, alterar o contraste, aplicar zoom e rotação. O treinamento também consiste em alterar o valor da taxa de aprendizagem ao longo

<sup>27</sup><https://www.kaggle.com/liuyd2018/planet-multi-label-image-classification>

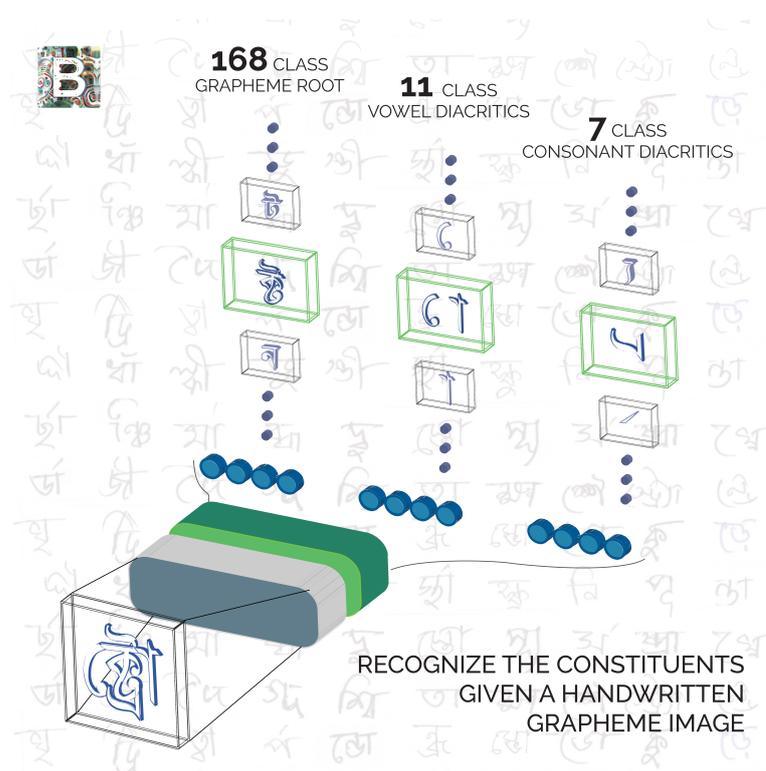


Figura B.43: Formação de grafemas em Bengali

dos ciclos. Além disso, para fazer a validação do modelo, foi utilizada uma separação simples, com 20% da base de treinamento sendo reservada para validação e o resto para o treinamento do modelo.

### B.22.3 Resultado

A solução descrita obteve 0,92887 pontos na métrica *mean F2 score*.

## B.23 Bengali.AI Handwritten Grapheme Classification

Bengali é o 5<sup>a</sup> idioma mais falado no mundo. A escrita desse idioma pode ser dividida em três grupos de símbolos: raiz de grafema (168 no total), vogais (11 no total) e consoantes (7 no total). Com isso, é possível combinar esses símbolos de modo a formar aproximadamente 13.000 grafemas diferentes. Um exemplo dessa formação pode ser visto na figura B.43.

Assim o objetivo desta competição é a criar modelos capazes de classificar os caracteres desse idioma. Para avaliar as submissões, a métrica desta competição é a *hierarchical macro-averaged recall*, ou seja, será calculada a cobertura média alcançada em cada um dos três grupos de símbolos, e a pontuação final será uma média ponderada dessas três coberturas.

### B.23.1 Base de dados

Esta base de dados possui 200.840 imagens de grafemas escritos à mão. Todas as imagens têm as mesmas dimensões, e apresentam os grafemas no centro das imagens. Apesar de não possuir imagens para todos os grafemas possíveis, todos os três grupos de símbolos estão representados por pelo menos 150 imagens. Note que a predição esperada para cada imagem não é o grafema inteiro (combinação completa), mas para cada um dos três grupos deve ser predito qual o símbolo deles que está presente.

Tabela B.21: Imagens do Bengali.AI Handwritten Grapheme Classification

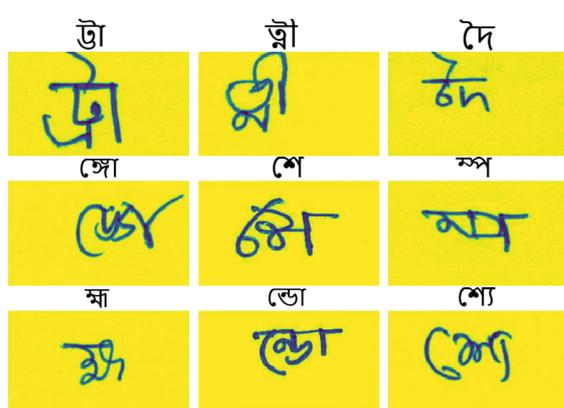


Figura B.44: Exemplos da base de dados

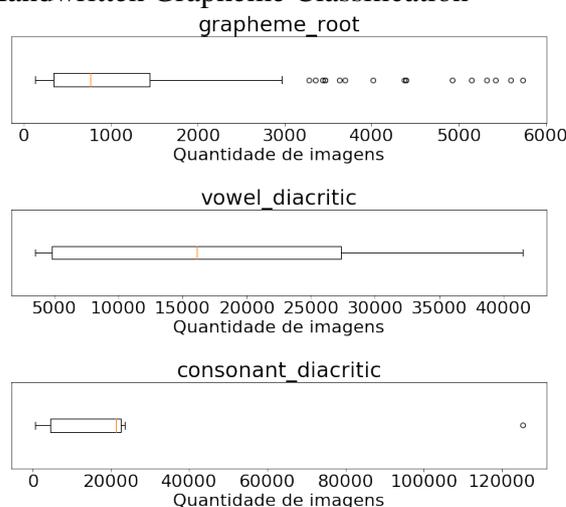


Figura B.45: Distribuição da base de dados

### B.23.2 Solução

A solução<sup>28</sup> selecionada, durante o pré-processamento, redimensionou, normalizou, removeu ruído e centralizou (encontrando a caixa delimitadora do contorno dos grafemas) todas as imagens.

O modelo foi criado utilizando as seguintes camadas da biblioteca Keras: *BatchNormalization*, *Conv2D*, *Dense*, *Dropout*, *Flatten* e *MaxPooling2D*. O modelo é do tipo *multi output CNN*, que é uma rede neural convolucional com múltiplas camadas de saída (3, neste caso, uma para cada grupo de símbolos). A função de ativação utilizada ao longo do modelo foi a *ReLU*, e nas camadas de saída foi utilizada a softmax. A função de perda escolhida foi a *categorical cross-entropy*.

Durante a etapa de treinamento, o autor separou 8% da base para fazer a validação do modelo. Foi utilizada a técnica de *data augmentation* com transformações de: rotação (até 8º graus), zoom (até 15%) e translação (até 15% horizontal e vertical). Para cada camada de

<sup>28</sup><https://www.kaggle.com/kaushal2896/bengali-graphemes-starter-eda-multi-output-cnn>

saída, foram utilizados *callbacks* para diminuir a taxa de aprendizado caso a acurácia deixasse de aumentar.

### B.23.3 Resultado

A solução descrita conseguiu alcançar 0,9506 pontos na métrica *hierachical macro-averaged recall*.

## B.24 SIIM-ACR Pneumothorax Segmentation

Pneumotórax é uma condição que causa falta de ar e pode levar a morte. Ela pode ocorrer através de lesões, dano por doenças pulmonares, ou até por razões não óbvias. O diagnóstico geralmente é feito por um radiólogo através de um raio x no peito, e pode ser difícil de ser confirmado. Por isso, essa competição busca um modelo capaz de encontrar, se existir, a região de manifestação dessa condição. As submissões serão avaliadas pela métrica *dice coefficient*.

### B.24.1 Base de dados

Esta base de dados conta com 12.047 radiografias de peito. Todas as imagens possuem as mesmas dimensões, apesar do peito estar centralizado na imagem, o posicionamento da condição, quando presente, não segue um padrão. Além disso, a base está balanceada e apresenta dados suficientes para executar um treinamento decente.

### B.24.2 Solução

A solução<sup>30</sup> selecionada, durante o pré-processamento, apenas normalizou e redimensionou todas as imagens para as dimensões 256x256x3 pixels.

O modelo escolhido pelo autor foi uma rede em forma de U (*U-Net*, o encoder escolhido para a rede foi uma *EfficientNetB4*, pré-treinada em cima da base *ImageNet*, e o *Decoder* foi definido pelo autor com as seguintes camadas da biblioteca Keras: *BatchNormalization*, *Conv2D*, *Conv2DTranspose*, *DepthwiseConv2D*, *Dropout* e *MaxPooling2D* (algumas dessas camadas foram concatenadas segundo a especificação da arquitetura *U-net*). Ao longo do *decoder*, foi utilizada a função de ativação *LeakyReLU*, e *sigmoid* na camada de saída. A função de perda utilizada foi uma combinação das funções *binary cross-entropy* e a *dice loss*.

Durante a etapa de treinamento, o autor separou 10% da base para executar a validação do modelo. Também foi utilizada a técnica de *data augmentation* com transformações de: inversão horizontal, contraste, gama, brilho, transformação elástica, distorção de grade, distorção ótica e

<sup>29</sup><https://www.kaggle.com/msafi04/pneumothoraxseg-visualization>

<sup>30</sup><https://www.kaggle.com/meaninglesslives/nested-unet-with-efficientnet-encoder>

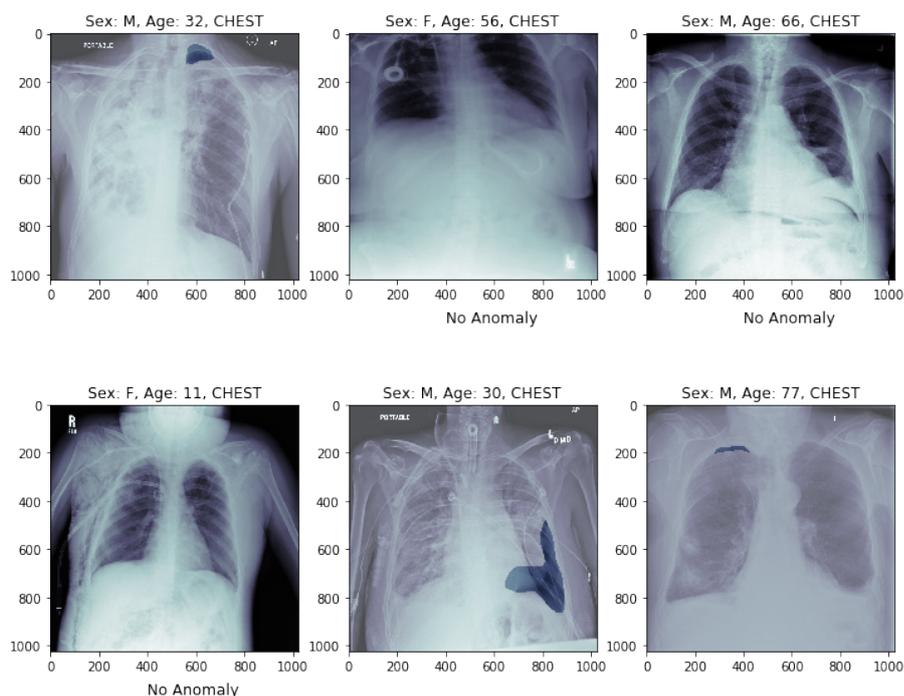
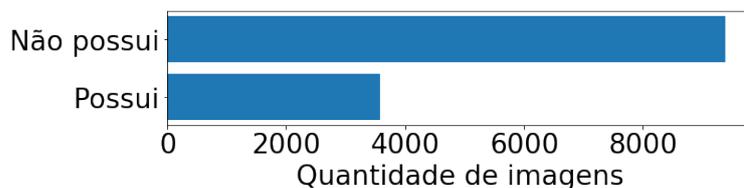
Tabela B.22: Exemplos da base de dados<sup>29</sup>.

Figura B.46: Distribuição da base de dados

recorte randomizado. O autor também utilizou um *callback* para variar a taxa de aprendizagem com *cosine annealing*.

Para as predições, o autor testou diversos limiares para decidir o valor mínimo para um pixel fazer parte da segmentação. Também foi utilizada a técnica de *test time augmentation*, mas apenas aplicando a transformação de inversão horizontal.

### B.24.3 Resultado

Seguindo isto, foi possível alcançar 0,8997 pontos na métrica *dice coefficient*.

## B.25 iMet Collection 2019 - FGVC6

O Museu Metropolitano de Arte em Nova Iorque possui uma coleção de mais de 1,5 milhão de objetos, em que 200 mil foram digitalizados com imagens. O catálogo destes objetos possui dados feitos por especialistas, e com uma grande variedade, como o nome do artista da obra, título, data, origem geográfica, etc.

Como as descrições feitas pelos especialistas têm um cunho histórico, pode ser um pouco complicado de relacionar certas obras que têm um aspecto visual parecido mas aspectos históricos completamente diferentes. Por isso, o objetivo desta competição é construir um modelo capaz de classificar certos atributos (dentre 1.103 possíveis) presentes nas imagens. A métrica de avaliação desta competição é a *mean F2 score*.

### B.25.1 Base de dados

A base contém 109.237 imagens, e elas não apresentam dimensões padronizadas, nem comportamento uniforme. Além disso, a base está desbalanceada e possui, em média, poucos dados por classe.

Tabela B.23: Imagens do iMet Collection 2019 - FGVC6



Figura B.47: Exemplos da base de dados



Figura B.48: Distribuição da base de dados

### B.25.2 Solução

A etapa de pré-processamento da solução<sup>31</sup> selecionada consiste em normalizar as imagens e recortá-las em 5 janelas de dimensões 336x336 pixels.

O modelo proposto utiliza a técnica de *ensemble* com nove modelos pré-treinados. As arquiteturas dos modelos não fica clara no notebook, mas devido ao código presente é provável

<sup>31</sup><https://www.kaggle.com/wx284260582/dataking-kernal>

que sejam todos da mesma arquitetura: *ResNeXt101* com *squeeze excitation*. O *ensemble* aqui consiste em fazer uma média das predições feitas por todos os modelos. E para a predição, cada classe deve ter uma probabilidade maior do que um limiar definido pelo autor, de 12%.

Quanto ao treinamento, nada é executado neste notebook, pois o autor simplesmente carregou os modelos pré-treinados e fez a média das suas predições. Também não fica clara a origem dos modelos pré-treinados, se foram importados pesos de alguma biblioteca, ou se foram modelos pré-treinados neste mesmo contexto pelo próprio autor.

### B.25.3 Resultado

A solução apresentada obteve 0,64100 pontos na métrica *mean F2 score*, garantindo a 38ª posição no ranking privado.

## B.26 Northeastern SMILE Lab - Recognizing Faces in the Wild

É comum parentes de sangue compartilharem traços faciais. O objetivo desta competição é a criação de um modelo que seja capaz de informar se duas pessoas são parentes de sangue através de apenas duas imagens, uma de cada rosto. As submissões serão avaliadas segundo a métrica “área abaixo da curva ROC”.

### B.26.1 Base de dados

Esta base apresenta 12.379 imagens de rostos humanos [35]. Todas as imagens possuem as mesmas dimensões e apesar de centralizadas no rosto, o mesmo pode estar de lado. As imagens estão separadas em pastas: cada família tem uma pasta, e cada indivíduo da família outra pasta (pois um indivíduo pode ter mais de uma imagem). Também existe uma tabela indicando quais pares de indivíduos fazem parte da mesma família. E vale ressaltar que a base pode ser considerada balanceada e possuidora de uma quantidade decente de dados para treinamento.

### B.26.2 Solução

A solução<sup>32</sup> selecionada é uma combinação de várias soluções. Durante o pré-processamento, as imagens apenas foram redimensionadas (para servir de entrada para os modelos escolhidos) e normalizadas.

O modelo é composto por 2 modelos pré-treinados, cada um deles recebe duas entradas (pois o modelo receberá duas imagens). Os modelos pré-treinados foram do tipo *FaceNet* e *ResNet50* (sem a camada de predição). Em seguida esses modelos foram conectados com as seguintes

<sup>32</sup><https://www.kaggle.com/mattemilio/smile-best-who-smile-last>

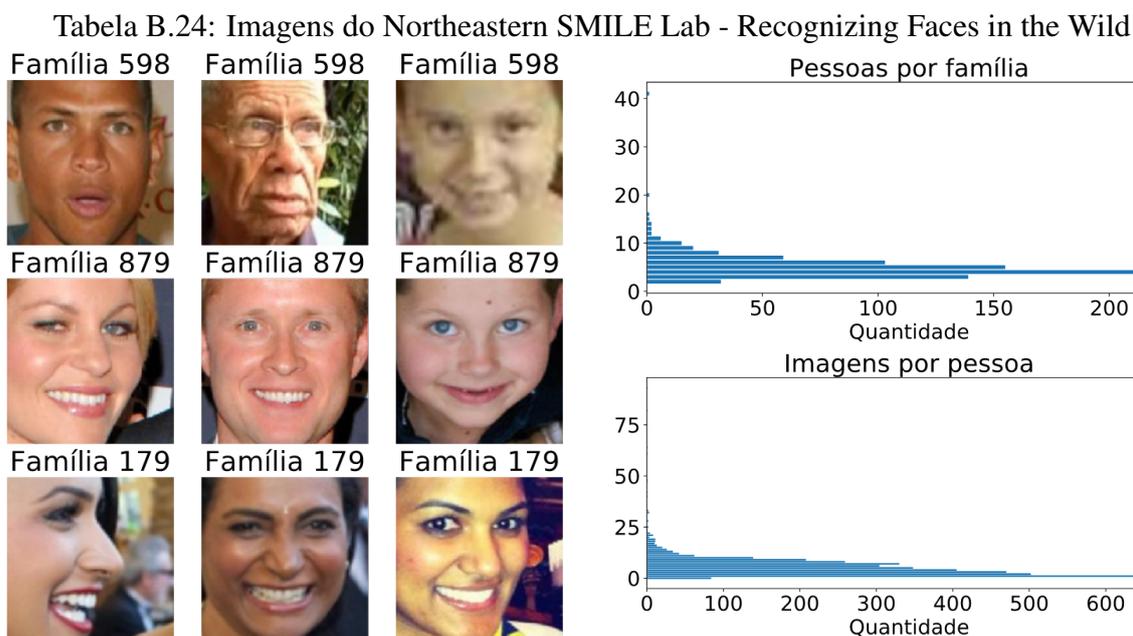


Figura B.49: Exemplos da base de dados

Figura B.50: Distribuição da base de dados

camadas da biblioteca Keras: *Conv2D*, *Dense*, *Dropout*, *Flatten* e *GlobalMaxPool2D*. Algumas dessas camadas foram combinadas através de: adição, subtração, multiplicação e concatenação. Ao longo dessas camadas, foi utilizada a função de ativação *ReLU*, e *sigmoid* na camada de saída. A função de perda do modelo foi a *binary cross-entropy*. As três últimas camadas dos modelos pré-treinadas e as seguintes foram liberadas para treinamento.

Para o treinamento, o autor separou uma família para validação, esta representa 10% da base. Também foi utilizado um *callback* para reduzir a taxa de aprendizagem quando a acurácia deixar de melhorar, durante a validação.

### B.26.3 Resultado

Com isso, foi possível alcançar uma pontuação de 0,9 na métrica “área abaixo da curva ROC”. O autor deste notebook encontra-se em primeiro no ranking privado com uma pontuação semelhante (0,923), mas não é informado no notebook se sua solução do primeiro lugar foi a apresentada acima.