Trabalho de Conclusão de Curso

Análises de Competições Presentes na Plataforma Kaggle para Auxiliar no Desenvolvimento de Novas Soluções para Problemas de Visão Computacional

Daniel Humberto Cavalcante Vassalo
dhcv@ic.ufal.br

Orientadores:

Rodrigo de Barros Paes
Willy Carvalho Tiengo

Maceió, Março de 2021

Daniel Humberto Cavalcante Vassalo

Análises de Competições Presentes na Plataforma Kaggle 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 Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientadores:

Rodrigo de Barros Paes

Willy Carvalho Tiengo

Maceió, Março de 2021

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecário Responsável: Jone Sidney A. de Oliveira – CRB-4 - 1485

V337a Vassalo, Daniel Humberto Cavalcante.
Análises de competições presentes na Plataforma Kaggle para auxiliar no desenvolvimento de novas soluções para problemas de visão computacional / Daniel Humberto Cavalcante Vassalo, - 2021.
83 f. : il. col.

Orientador: Prof. Rodrigo de Barros Paes.
Coorientador: Prof. Willy Carvalho Tiengo
Monografia (Trabalho de conclusão de curso em Ciência da Computação) – Universidade Federal de Alagoas, Instituto de Computação. Maceió, 2021.

Bibliografia: f. 23 – 25.
Apêndice: f. 26 - 83

1. Aprendizagem de Máquina. 2. Visão Computacional. 3. Ciência de Dados. 4. Plataforma Kaggle. I. Título.

CDU: 004.89

Agradecimentos

Gostaria de agradecer a meus pais, Kátia Lia e José Humberto, e irmãs, Vanessa e Lívia, pelo constante apoio e auxílio durante toda a minha trajetória.

Também agradeço aos meus amigos e companheiros de jornada, Nelson Gomes, Gabriel Barbosa, Lucas Amorim, Lucas Raggi, França Mac Dowell e Wagner Fontes, por estarem sempre ao meu lado compartilhando dos desafios que surgiram durante a graduação. Sem essas companhias a trajetória teria sido muito mais difícil.

Agradeço a meus amigos de longa data, Xexeu, Pedro, Fofis, Zé, Luiz, Vição, Stutz, Nicole, Carol, Leticia, e todos os demais que sempre estiveram dispostos a ouvir e fazer críticas acerca do desenvolvimento do trabalho.

Agradeço também a todos os meus professores que exigiram sempre o melhor de mim durante todo meu processo de formação, e principalmente a meus orientadores, Rodrigo Paes e Willy Tiengo, que foram essenciais para a minha formação e inicio da atuação profissional.

Também vale o agradecimento a meus colegas de trabalho por compartilharem conhecimentos e conselhos fundamentais para a vida profissional e até mesmo para o processo de formação.

Daniel Humberto Cavalcante Vassalo

"O homem não teria alcançado o possível se, repetidas vezes, não tivesse tentado o impossível."

– Weber, *Max*

Resumo

Este trabalho visa analisar e discutir os dados coletados sobre problemas de visão computacional e suas melhores soluções encontradas na plataforma Kaggle. O objetivo destas discussões é tentar entender quais são as abordagens mais utilizadas em certos tipos de problemas e porquê elas são mais recorrentes do que outras. Para isso, antes do desenvolvimento desta monografia foi feito um trabalho de pesquisa para coletar os dados referentes às características de algumas competições de visão computacional presentes na plataforma Kaggle. Para este trabalho foram levantados questionamentos acerca do desenvolvimento de novas soluções para problemas de visão computacional. Cada questionamento foi amplamente discutido e embasado com análises sobre os dados coletados na pesquisa e conceitos presentes na literatura. Por fim, concluiu-se a importância da análise de soluções já existentes para auxiliar na discussão de certas etapas do desenvolvimento de soluções de novos problemas.

Palavras-chave: aprendizagem de maquina; visão computacional; ciência de dados; Kaggle.

Abstract

This work aims to analyze and discuss previously collected data regarding computer vision problems and their best solutions on the Kaggle platform. The purpose of these discussions was to try to understand what are the most used approaches in some types of problems and why they are used more often than others. To accomplish that, before the development of this monography, a research project was carried out to collect data about the main characteristics of some computer vision competitions on Kaggle's platform. For this work, relevant questions were raised about the development of new solutions to computer vision problems. Each question was widely discussed and substantiated with an analysis of the data collected in the research project and concepts found in the literature. Finally, it concludes the importance of the analysis of existing solutions to assist in the discussion of certain steps of the process of development of solutions to new problems.

Key-words: machine learning; computer vision; data science; Kaggle.

Lista de Figuras

2.1 Impacto do balanceamento na pontuação. Métricas de minimização possuem 2 casos de base desbalanceada e 4 balanceadas, enquanto métricas de maximização possuem 10 casos de bases desbalanceadas e 10 balanceadas. 4
2.2 Regressão linear sobre o tamanho da base de dados e a quantidade de classes da competição 8
2.3 Gráficos mostrando a relação entre pontuação alcançada e quantidade média de imagens por classe, divididos por tipos de métrica e com destaque para o balanceamento das bases. 9
2.4 *ReLU* à esquerda e *Leaky ReLU* à direita. 12
2.5 *Building block* (bloco de construção) da arquitetura *ResNet* 15

A.1 Página inicial do Kaggle 27
A.2 Descrição de uma competição do Kaggle 27
A.3 Descrição dos dados de uma competição do Kaggle 28
A.4 *Notebooks* de uma competição do Kaggle 28
A.5 Trechos de código dentro de um *notebook* 29
A.6 Texto rico dentro de um *notebook* 29
A.7 Placar público 30
A.8 Placar privado 30

B.1 Exemplos da base de dados MNIST 43
B.2 Exemplos com Cacto 45
B.3 Exemplos sem Cacto 45
B.4 Distribuição da base de dados 45
B.5 Exemplos de imagens com gatos 46
B.6 Exemplos de imagens com cachorros 46
B.7 Exemplos da base de dados, com os pontos-chave representados em vermelho . 47
B.8 Frequência da cada ponto-chave 47
B.9 Exemplos da base de dados 48
B.10 Frequência de cada baleia 48
B.11 Exemplos da base de dados 54
B.12 Frequências de todas classes 54
B.13 Exemplos da base de dados 55
B.14 Frequências de todas classes 55
B.15 Exemplos da base de dados com a máscara 57
B.16 Frequências de todas classes 57
B.17 *Autoencoder* 57
B.18 Exemplo da base de dados (HUVEC-04 Plate4 G08, que apresenta classe: sirna$_1$21) 59

B.19 Exemplo da base de dados ((RPE-04 Plate3 F21, que apresenta classe: sirna$_4$93) 59
B.20 Distribuição da base de dados 59
B.21 3 exemplos de desenho para alguns exemplos de classes 61
B.22 Distribuição da base de dados 61
B.23 Exemplos da base de dados 62
B.24 Distribuição da base de dados 62
B.25 Exemplos da base de dados 63
B.26 Exemplos de algumas classes da base de dados 65
B.27 Convoluções da solução 65
B.28 Exemplos da base de dados 66
B.29 Distribuição da base de dados 66
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) 68
B.31 Distribuição da base de dados 68
B.32 Exemplos da base de dados com a máscara em vermelho, o título da imagem indica a profundidade da imagem 69
B.33 Distribuição da base de dados 69
B.34 Exemplos da base de dados com os defeitos circulados, seguindo o mapeamento de cor: 1 - Verde, 2 - Vermelho, 3 - Azul, 4 - Amarelo 71
B.35 Distribuição da base de dados 71
B.36 Exemplos da base de dados 72
B.37 Distribuição da base de dados 72
B.38 Exemplos da base de dados 73
B.39 Distribuição da base de dados 73
B.40 Exemplos da base de dados 75
B.41 Distribuição da base de dados 75
B.42 Exemplos da base de dados 76
B.43 Formação de grafemas em Bengali 77
B.44 Exemplos da base de dados 78
B.45 Distribuição da base de dados 78
B.46 Distribuição da base de dados 80
B.47 Exemplos da base de dados 81
B.48 Distribuição da base de dados 81
B.49 Exemplos da base de dados 83
B.50 Distribuição da base de dados 83

Conteúdo

Lista de Figuras **iv**

1 Introdução **1**

2 Análises e Discussões **3**
2.1 Qual o impacto do contexto do problema? . . . 3
2.2 Qual o impacto de uma base de dados desbalanceada na solução? . . . 3
2.2.1 Impacto no modelo . . . 4
2.2.2 Impacto na pontuação . . . 4
2.2.3 Impacto no pré-processamento . . . 5
2.2.4 Conclusão . . . 5
2.3 Qual linguagem utilizar para desenvolver a solução? . . . 6
2.4 Qual biblioteca utilizar para construir o modelo? . . . 6
2.5 Quantas imagens são necessárias para treinar um modelo? . . . 7
2.5.1 Poucas imagens por classe implicam em resultados piores? . . . 8
2.6 O que fazer na etapa de pré-processamento? . . . 8
2.7 Qual métrica utilizar para avaliar o modelo? . . . 10
2.8 Qual função de perda utilizar? . . . 11
2.9 Qual função de ativação utilizar? . . . 12
2.10 Quais camadas utilizar? . . . 12
2.11 Quantas camadas o modelo deve ter? . . . 13
2.12 Como combinar camadas? . . . 14
2.13 Quando utilizar *transfer learning*? . . . 16
2.14 Qual o modelo mais indicado para aplicar a técnica de *transfer learning*? . . . 16
2.15 O que fazer na etapa de treinamento? . . . 17
2.15.1 Quando utilizar a estratégia de *data augmentation*? . . . 18
2.16 Como validar o modelo? . . . 19
2.16.1 Qual deve ser o tamanho da base de validação? . . . 19

3 Conclusão **21**
3.1 Trabalhos futuros . . . 22

Referências bibliográficas **23**

A Coleta e Anotação de Dados **26**
A.1 A plataforma Kaggle . . . 26
A.2 A coleta de dados . . . 26
A.3 Atributos . . . 33
A.3.1 Atributos do problema . . . 34

A.3.2 Atributos da base de dados 36
A.3.3 Atributos da solução 37

B Catálogo de Competições **42**
B.1 Digit Recognizer 42
B.1.1 Base de dados 42
B.1.2 Solução 43
B.1.3 Resultado 43
B.2 Aerial Cactus Identification 43
B.2.1 Base de dados 44
B.2.2 Solução 44
B.2.3 Resultado 44
B.3 Dogs vs Cats 44
B.3.1 Base de dados 45
B.3.2 Solução 46
B.3.3 Resultado 46
B.4 Facial Keypoints Detection 47
B.4.1 Base de dados 47
B.4.2 Solução 47
B.4.3 Resultado 48
B.5 Humpback Whale Identification Challenge 48
B.5.1 Base de dados 49
B.5.2 Solução 49
B.5.3 Resultado 52
B.6 Humpback Whale Identification 52
B.6.1 Base de dados 52
B.6.2 Solução 52
B.6.3 Resultado 53
B.7 Dog Breed Identification 53
B.7.1 Base de dados 53
B.7.2 Solução 54
B.7.3 Resultado 54
B.8 Plant Seedlings Classification 55
B.8.1 Base de dados 55
B.8.2 Solução 56
B.8.3 Resultado 56
B.9 Understanding Clouds from Satellite Images 56
B.9.1 Base de dados 56
B.9.2 Solução 56
B.9.3 Resultado 58
B.10 Recursion Cellular Image Classification 58
B.10.1 Base de dados 58
B.10.2 Solução 58
B.10.3 Resultado 59
B.11 Quick, Draw! Doodle Recognition Challenge 59
B.11.1 Base de dados 60
B.11.2 Solução 60
B.11.3 Resultado 61

B.12 State Farm Distracted Driver Detection . . . 61
B.12.1 Base de dados . . . 62
B.12.2 Solução . . . 62
B.12.3 Resultado . . . 62
B.13 Kannada MNIST . . . 63
B.13.1 Base de dados . . . 63
B.13.2 Solução . . . 63
B.13.3 Resultado . . . 64
B.14 Leaf Classification . . . 64
B.14.1 Base de dados . . . 64
B.14.2 Solução . . . 64
B.14.3 Resultado . . . 65
B.15 Invasive Species Monitoring . . . 66
B.15.1 Base de dados . . . 66
B.15.2 Solução . . . 66
B.15.3 Resultado . . . 67
B.16 RSNA Intracranial Hemorrhage Detection . . . 67
B.16.1 Base de dados . . . 67
B.16.2 Solução . . . 67
B.16.3 Resultado . . . 68
B.17 TGS Salt Identification Challenge . . . 68
B.17.1 Base de dados . . . 69
B.17.2 Solução . . . 69
B.17.3 Resultado . . . 70
B.18 Severstal: Steel Defect Detection . . . 70
B.18.1 Base de dados . . . 70
B.18.2 Solução . . . 70
B.18.3 Resultado . . . 71
B.19 APTOS 2019 Blindness Detection . . . 71
B.19.1 Base de dados . . . 72
B.19.2 Solução . . . 72
B.19.3 Resultado . . . 73
B.20 Airbus Ship Detection Challenge . . . 73
B.20.1 Base de dados . . . 73
B.20.2 Solução . . . 74
B.20.3 Resultado . . . 74
B.21 Ultrasound Nerve Segmentation . . . 74
B.21.1 Base de dados . . . 75
B.21.2 Solução . . . 75
B.21.3 Resultado . . . 75
B.22 Planet: Understanding the Amazon from Space . . . 76
B.22.1 Base de dados . . . 76
B.22.2 Solução . . . 76
B.22.3 Resultado . . . 77
B.23 Bengali.AI Handwritten Grapheme Classification . . . 77
B.23.1 Base de dados . . . 78
B.23.2 Solução . . . 78
B.23.3 Resultado . . . 79

B.24 SIIM-ACR Pneumothorax Segmentation 79
B.24.1 Base de dados 79
B.24.2 Solução 79
B.24.3 Resultado 80
B.25 iMet Collection 2019 - FGVC6 80
B.25.1 Base de dados 81
B.25.2 Solução 81
B.25.3 Resultado 82
B.26 Northeastern SMILE Lab - Recognizing Faces in the Wild 82
B.26.1 Base de dados 82
B.26.2 Solução 82
B.26.3 Resultado 83

1

Introdução

A aprendizagem de máquina está presente em diversos contextos da sociedade moderna, desde a recomendação inteligente de produtos online até o reconhecimento de voz utilizado para acender uma lâmpada. Dentre as diversas áreas que utilizam aprendizagem de máquina, a de principal interesse para este trabalho é a "visão computacional", cujas tarefas envolvem a utilização do computador para executar atividades visuais, como reconhecer objetos em fotos, pessoas, doenças em radiografias, etc.

Esta área vem crescendo numa velocidade acentuada, o que traz consigo o surgimento de cada vez mais modelos computacionais. Isso faz com que, frente à um problema de visão computacional, muitas vezes seja difícil decidir qual modelo utilizar ou quais técnicas de validação e treinamento são mais adequadas àquele contexto.

Segundo a Forbes, 83% dos negócios dizem que a I.A. é uma prioridade estratégica para seus negócios hoje [1]. Por conta dessa expectativa do mercado, os modelos inteligentes vêm se desenvolvendo numa velocidade muito expressiva e os programadores se veem na necessidade de se manterem sempre atentos aos novos modelos e técnicas surgentes na literatura, pois existem cada vez mais modelos, cada um com seus prós e contras, e muitas técnicas para tratamento de dados, treinamento e validação de modelo.

Somado a isso, o desenvolvimento de soluções baseadas em aprendizagem de máquina requer do programador experiência empírica de desenvolvimento de soluções de outros problemas, preferencialmente semelhantes ao problema a ser resolvido. Em muitos casos, o desenvolvedor toma decisões e escolhe abordagens de soluções baseando-se exclusivamente em seu conhecimento empírico. Com desenvolvedores experientes, isso de fato não é um grande problema nem apresenta grandes dificuldades, mas para aqueles em fases iniciais de aprendizado pode ser extremamente desafiador e até mesmo ser fator de desmotivação, caso não consiga soluções com resultados satisfatórios logo nos primeiros problemas resolvidos. Nesse contexto, seria de grande valia um guia que pudesse dar dicas e recomendações para o desenvolvedor. Esse é o objetivo de um projeto de pesquisa desenvolvido no laboratório EDGE da UFAL, no

qual este trabalho está inserido (ver Apêndices A e B). Mais especificamente, o foco deste trabalho é realizar uma análise aprofundada dos dados levantados nessa pesquisa, buscando discutir acerca de alguns tópicos relevantes para o desenvolvimento de novas soluções e apontar quais as abordagens mais recorrentes, a fim de facilitar o ingresso de iniciantes na área.

Para isso, o presente trabalho está organizado em dois capítulos, que se preocupam, respectivamente, em analisar e discutir sobre os dados coletados, apontando quais as implicações deles em alguns pontos do desenvolvimento de soluções; e por fim, concluir quais as principais contribuições das discussões apresentadas.

Os apêndices deste trabalho contém a pesquisa mencionada anteriormente e também fazem parte de outro trabalho, feito em conjunto ao presente, que tem por objetivo construir e testar um guia prático de como desenvolver novas soluções, como uma espécie de ponto de partida para iniciantes na área.

Vale ressaltar que para completo entendimento do presente trabalho, é desejável que o leitor tenha familiaridade com alguns conceitos de ciência de dados, aprendizagem de máquina e redes neurais convolucionais.

2

Análises e Discussões

A seguir encontram-se algumas perguntas relevantes no contexto de desenvolvimento de um modelo/solução para problemas de visão computacional, e que guiarão as discussões propostas. O objetivo de tais discussões é esclarecer porquê certas decisões são tomadas mais frequentemente que outras, utilizando os dados coletados na pesquisa para saber quais são as decisões mais tomadas, e artigos científicos, livros e sites confiáveis para embasar as justificativas de tais decisões.

2.1 Qual o impacto do contexto do problema?

O contexto em si não faz diferença para uma rede neural, mas sim para o problema, a base de dados e a solução. Pois dependendo do contexto, a definição do problema deve exigir uma rigorosa avaliação para garantir que as soluções possam ser utilizadas num caso concreto, como por exemplo em problemas do contexto de medicina, onde deve-se garantir que os resultados gerados sejam realmente confiáveis.

Existem também contextos onde a coleta da base de dados é muito onerosa ou trabalhosa, limitando assim o seu tamanho, ou qualidade. Por consequência, tais aspectos podem causar impactos significativos nas fases de pré-processamento e treinamento da solução.

2.2 Qual o impacto de uma base de dados desbalanceada na solução?

Durante a pesquisa, foram levantados questionamentos acerca do impacto de uma base desbalanceada em alguns pontos do desenvolvimento da solução.

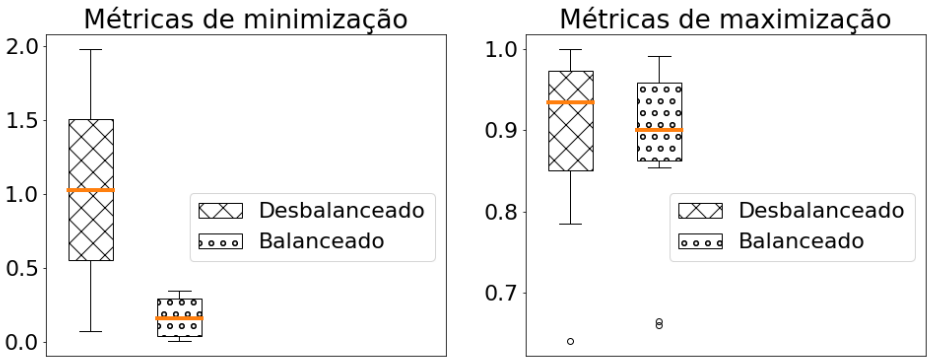


Figura 2.1: Impacto do balanceamento na pontuação. Métricas de minimização possuem 2 casos de base desbalanceada e 4 balanceadas, enquanto métricas de maximização possuem 10 casos de bases desbalanceadas e 10 balanceadas.

2.2.1 Impacto no modelo

Quando um modelo é treinado com um conjunto pequeno de dados, ele tende a se ajustar muito às amostras do conjunto de treinamento e isso resulta em uma generalização pobre[1] [2]. Ou seja, uma base enviesada, com poucos dados para uma certa classe, irá deixar o modelo também enviesado, a menos que seja feito algum tratamento nos dados ou seja adotada alguma estratégia específica de treinamento, a fim de evitar o *overfitting* (sobreajuste).

2.2.2 Impacto na pontuação

Para analisar se há ou não impacto na pontuação, as métricas foram divididas em 2 tipos: minimização, quanto menor a pontuação melhor é a solução; e maximização, quanto maior a pontuação melhor é a solução.

Observando a Figura 2.1, os dados coletados indicam que uma base desbalanceada não causa nenhum impacto significativo na pontuação alcançada pelas soluções.

No entanto, a ausência de uma diferença significativa nas pontuações tem uma razão. Ao verificar o impacto do uso de *data augmentation* nas soluções, a Tabela 2.1 mostra que as pontuações são melhores nas soluções que utilizam esta estratégia. Indicando que o uso de *data augmentation* em bases desbalanceadas ajuda a amenizar os impactos negativos na pontuação, pois, de modo geral, a estratégia melhora a pontuação média das soluções por conta do consequente aumento da base a fim de compensar o desbalanceamento da mesma.

[1] No entanto, existem campos de pesquisa recentes que utilizam poucos dados de treinamento para retreinar ou aprimorar modelos, mas tal abordagem foge do escopo deste trabalho.

Tipo da métrica	Usa *data augmentation*	Base desbalanceada		Base balanceada	
		Pontuação média	Frequência	Pontuação média	Frequência
0	Falso	1.98535	1	-	-
0	Verdadeiro	0.076	1	0.16959	4
1	Falso	-	-	0.856357	4
1	Verdadeiro	0.89621	10	0.88388	6

Tabela 2.1: Frequência e pontuação média das soluções por cada tipo de métrica e uso de *data augmentation*, divididos também pelo tipo de balanceamento da base de dados.

2.2.3 Impacto no pré-processamento

De todas as soluções anotadas durante a pesquisa, somente duas não fizeram nenhum tipo de pré-processamento, tendo uma a base balanceada e a outra não. Além disso, os dados indicam que tanto para bases balanceadas quanto para desbalanceadas, a complexidade de pré-processamento não parece variar de forma significativa. A tabela 2.2 apresenta uma análise da relação entre o balanceamento das bases e a utilização de "outras técnicas de pré-processamento"[2], que não sejam redimensionamento nem normalização (pois elas não resolvem o problema do desbalanceamento). Assim, fica evidenciado que quando a base está desbalanceada, a maioria das soluções não trata esse problema na etapa de pré-processamento.

Base de dados balanceada	Usa outras técnicas de pré-processamento	Frequência
Falso	Falso	7
Falso	Verdadeiro	5
Verdadeiro	Falso	11
Verdadeiro	Verdadeiro	3

Tabela 2.2: Frequência de uso de outras técnicas de pré-processamento em bases balanceadas e não balanceadas

2.2.4 Conclusão

Em resumo, uma base desbalanceada não é impedimento para alcançar uma boa pontuação. E apesar da Tabela 2.3 mostrar que *data augmentation* pode ser utilizada independentemente do balanceamento da base, esta estratégia tem especial importância em bases desbalanceadas, pois ao se deparar com uma é muito recomendado utilizá-la para evitar o *overfitting*.

[2] Técnicas como recortar imagens, criar uma nova base para treinamento, criar novas imagens (com *encondings* diferentes), remover duplicatas, criar uma *blacklist*, etc.

Base de dados balanceada	Usa *data augmentation*	Frequência
Falso	Falso	1
Falso	Verdadeiro	11
Verdadeiro	Falso	4
Verdadeiro	Verdadeiro	10

Tabela 2.3: Frequência de uso da técnica de *data augmentation* em bases balanceadas e não balanceadas

2.3 Qual linguagem utilizar para desenvolver a solução?

Absolutamente todas as soluções anotadas utilizaram a linguagem de programação Python para implementar suas abordagens, desde o pré-processamento até a construção do modelo.

Essa preferência pela linguagem também ocorre fora do Kaggle e não é por acaso. Segundo um levantamento feito pelo Github [3], em 2018 Python foi a linguagem de programação mais utilizada para pesquisa e desenvolvimento de aprendizagem de máquina pelos usuários da plataforma. Além de ser muito utilizada, a linguagem é simples e proporciona a criação de códigos enxutos e legíveis, contém diversas bibliotecas e *frameworks* de aprendizagem de máquina, possui uma grande comunidade e suporte empresarial, e por ser muito "portátil" consegue rodar em diversos sistemas operacionais diferentes sem complicações [4].

2.4 Qual biblioteca utilizar para construir o modelo?

Como todas as soluções utilizaram a linguagem de programação Python, é natural que as bibliotecas utilizadas sejam desta linguagem. A Tabela 2.4 mostra que a biblioteca mais utilizada foi a TensorFlow, aparecendo em mais de 65% (17 de 26) das soluções observadas. Tal recorrência provavelmente se dá por conta de sua grande comunidade ativa em fóruns online e por ser uma biblioteca completa.

Biblioteca	Frequência
TensorFlow	17
Fast.ai	5
PyTorch	2
Keras	1
Nenhuma	1

Tabela 2.4: Frequência de uso de cada biblioteca nas soluções observadas

2.5 Quantas imagens são necessárias para treinar um modelo?

É intuitivo pensar que o tamanho da base de dados será proporcional à quantidade de classes presentes na mesma. No entanto, os dados coletados na pesquisa não mostram uma relação direta muito forte entre o crescimento na quantidade de classes e o crescimento no tamanho da base de dados. Pois mesmo ao remover as bases consideradas *outliers* (ver a Tabela 2.5) – bases que possuem uma quantidade de dados acima de uma variação considerada "normal" (3º quartil) – e fazer uma regressão linear (Figura 2.2) foi obtido o coeficiente de determinação $R^2 = 0,456$, indicando que a regressão não conseguiu se ajustar muito bem aos dados, ou seja, não há um comportamento/regra clara sendo seguida para os tamanhos das bases observadas à medida que a quantidade de classes aumenta.

Quantidade de classes	Tamanho médio da base	Outlier
1	70.079	Falso
2	12.234	Falso
4	14.425	Falso
5	3.662	Falso
6	752.803	Verdadeiro
10	41.474	Falso
12	4.750	Falso
15	7.049	Falso
99	1.584	Falso
120	10.222	Falso
345	50.000.000	Verdadeiro
449	40.479	Falso
1.103	109.237	Falso
1.108	125.510	Falso
4.251	9.850	Falso
5.005	25.361	Falso
13.000	200.840	Falso

Tabela 2.5: Tamanho médio da base de acordo com a quantidade de classes da competição

Assim, o tamanho da base parece ser algo particular a cada situação, pois depende da dificuldade em coletar dados para a mesma ou da complexidade do problema (visto que dependendo do contexto e dos recursos de cada instituição organizadora, o poder de coleta de dados varia).

No entanto, existe uma "regra de ouro" que a comunidade da ciência de dados adota. Ela afirma que uma boa base de dados deve ter pelo menos mil imagens para cada classe a ser reconhecida[3] [5].

[3]No entanto, existem campos de pesquisa recentes que buscam criar modelos capazes de fazer classificações com pouquíssimos dados de treinamento, porém isso foge do escopo deste trabalho, visto que nenhuma solução anotada utilizou tal abordagem.

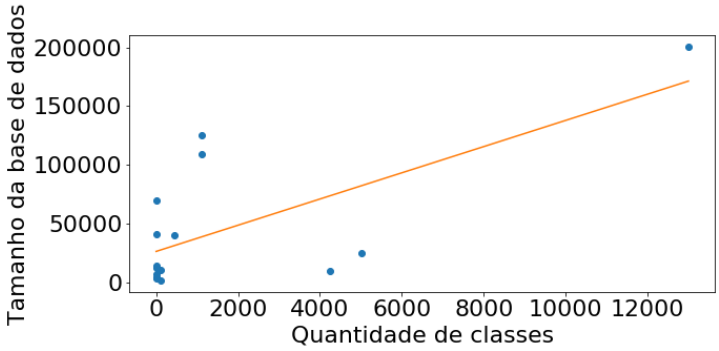


Figura 2.2: Regressão linear sobre o tamanho da base de dados e a quantidade de classes da competição

2.5.1 Poucas imagens por classe implicam em resultados piores?

Para responder a essa pergunta foram plotados dois gráficos (um para cada tipo de métrica), mostrando a relação entre a pontuação obtida e a quantidade média de imagens por classe da base de dados, ver Figura 2.3.

Os gráficos mostram não haver nenhuma relação entre a quantidade média de imagens por classe e a pontuação alcançada (nem mesmo nas bases desbalanceadas), pois mesmo nos casos com poucas imagens por classe foi possível alcançar pontuações excelentes, superando até algumas pontuações de casos com quantidades maiores de imagens por classe.

2.6 O que fazer na etapa de pré-processamento?

É na etapa de pré-processamento que serão feitos alguns tratamentos em cima dos dados antes de alimentar o modelo computacional. Nas soluções analisadas durante o desenvolvimento da pesquisa, observou-se que as seguintes técnicas foram utilizadas: normalização, redimensionamento, recorte, aplicação de transformações (rotação, translação, escala, etc), aplicação de filtros de cor, remoção de duplicatas, extração de *features* (características) usando MLP (*Multilayer Perceptron*), remoção de ruído com *gaussian blur subtraction* (ver Tabela 2.6).

Dentre as técnicas citadas, as mais utilizadas foram redimensionamento e normalização (ambas apareceram em dezenove soluções), sendo utilizadas em conjunto catorze vezes. Além disso, também foi constatado que apenas duas soluções não fizeram nenhum tipo de pré-processamento.

O fato da técnica de redimensionamento ter sido a mais utilizada se deu porque redes neurais convolucionais não aceitam como entrada imagens de tamanhos variados entre si, todas as entradas precisam ter as mesmas dimensões, por conta das operações matriciais [6]. E a grande maioria das soluções observadas utilizaram redes neurais convolucionais (ver Seção 2.10).

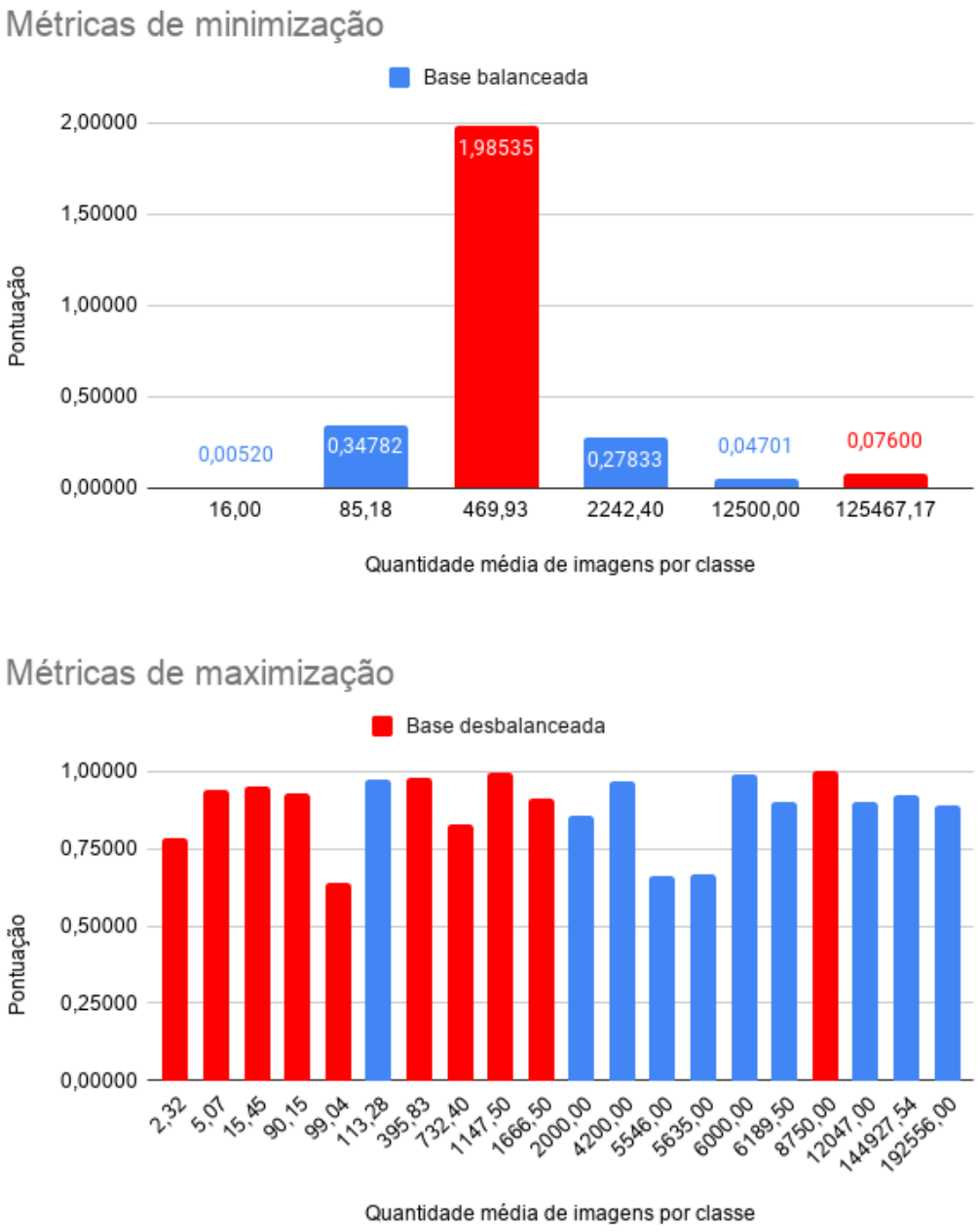


Figura 2.3: Gráficos mostrando a relação entre pontuação alcançada e quantidade média de imagens por classe, divididos por tipos de métrica e com destaque para o balanceamento das bases.

Técnicas de pré-processamento	Frequência
Redimensionar todas as imagens	19
Normalizar todas as imagens	19
Remover duplicatas	2
Nada	2
Extração de *features* usando MLP	1
Recorte das imagem em janelas	1
Construção de *twin dataset*	1
Pilha de imagens com diferentes *encodings*	1
Criação manual de uma lista negra	1
Recorte padrão das imagens	1
Rotacionar imagens de cabeça para baixo	1
Conversão de imagens para preto e branco	1
Gaussian blur subtraction	1
Recorte inteligente das imagens *bounding box*	1
Affine tranformation	1

Tabela 2.6: Técnicas de pré-processamento utilizadas nas soluções analisadas, ordenadas por frequência de uso.

2.7 Qual métrica utilizar para avaliar o modelo?

Não existe uma métrica única capaz de avaliar de modo geral um modelo. A escolha da métrica depende dos objetivos de quem está construindo o modelo. E o modelo deve ser construído visando os objetivos do problema enfrentado. Ou seja, a escolha da métrica vai depender do tipo de problema e o objetivo desejado.

No entanto, conforme a Tabela 2.7 sugere, geralmente em problemas do tipo "segmentação de objetos" usa-se *dice coefficient*. Em problemas do tipo "classificador" é possível observar que as métricas mais utilizadas são, respectivamente: *log loss*, acurácia, precisão e *F-score*. Já para os problemas de "detecção de objetos" e "*feature pinning*" não é possível observar nenhuma tendência, pois a quantidade de dados foi insuficiente para refletir qualquer tendência.

Também é interessante observar que dentre as quatro métricas citadas de "classificadores", duas delas possuem uma forte relação, a precisão e o *F-Score*. Pois, o cálculo do *F-Score* utiliza a precisão (ver Equação 2.1 [7]). Assim, pode-se dizer que a precisão é a métrica mais relevante em problemas do tipo "classificador", pois está envolvida em 6 competições no total (3 aparecendo sozinha e 3 aparacendo dentro do *F-Score*).

$$F - Score = 2 * \frac{precisão * recall}{precisão + recall} \tag{2.1}$$

Tipo de problema	Métrica	Frequência
Segmentação de objetos	*Dice Coefficient*	3
Detecção de objetos	*Dice Coefficient*	1
Detecção de objetos	*F-Score*	1
Detecção de objetos	Precisão	1
Feature Pinning	Raiz Quadrada do Erro Médio	1
Classificador	*Log Loss*	5
Classificador	Acurácia	4
Classificador	*F-Score*	3
Classificador	Precisão	3
Classificador	Área Abaixo da Curva ROC	2
Classificador	Coeficiente de Kappa Ponderado Quadrático	1
Classificador	*Recall*	1

Tabela 2.7: Frequência de uso de cada métrica em cada tipo de problema

2.8 Qual função de perda utilizar?

Para a análise a seguir, foi constatado durante a pesquisa que em 6 soluções não foi possível encontrar quais funções de perda foram utilizadas. Diante disso, a análise levou em conta somente as outras 20 soluções, que tiveram as funções de perda anotadas.

A função de perda implica diretamente em como a rede neural vai "aprender com os erros". O que os dados obtidos mostram (ver Tabela 2.8) é que em problemas do tipo "classificador", geralmente utilizá-se alguma das funções de *cross-entropy* (*binary, categorical* ou a padrão), pois foram utilizadas pela maioria (73,68%) das soluções analisadas. E em problemas do tipo "detecção de objetos" as soluções utilizaram a função *binary cross-entropy* junto à função *dice loss*. Para os problemas de "segmentação e objetos" e "*feature pinning*" nenhuma tendência foi observada, pois a quantidade de dados foi insuficiente para refletir qualquer tendência.

Tipo de problema	Função de Perda	Frequência
Segmentação de objetos	*Binary Cross-entropy + Dice Loss*	1
Segmentação de objetos	*Jaccard Loss*	1
Segmentação de objetos	Nenhuma	1
Detecção de objetos	*Binary Cross-entropy + Dice Loss*	1
Detecção de objetos	Não encontrado	1
Detecção de objetos	*Weighted (Binary Cross-entropy) and (Dice Loss)*	1
Feature Pinning	*Mean Squared Error*	1
Classificador	*Binary Cross-entropy*	7
Classificador	Não encontrado	5
Classificador	*Categorical Cross-entropy*	4
Classificador	*Cross-entropy*	2
Classificador	Customizada	1

Tabela 2.8: Frequência de uso de cada função de perda em cada tipo de problema

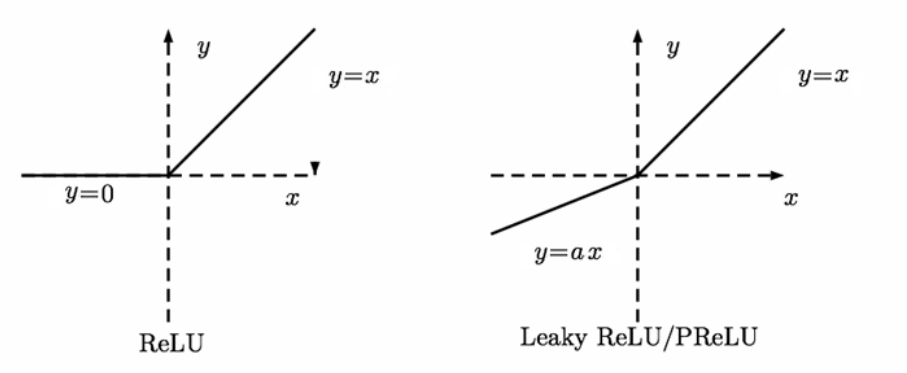


Figura 2.4: *ReLU* à esquerda e *Leaky ReLU* à direita.

2.9 Qual função de ativação utilizar?

Os dados mostram que todas as soluções utilizaram alguma das seguintes funções de ativação: *ReLU*, *softmax*, *sigmoid*. Estas foram utilizadas, respectivamente, por 73,08%, 46,15% e 42,3% das soluções (ver Tabela 2.9). Aqui consideramos que a função *Leaky ReLU* é um tipo de *ReLU*, pois esta pode ser vista como uma *ReLU* levemente modificada, como é possível ver na Figura 2.4 a *Leaky ReLU* modifica somente a função para valores negativos. Além disso, a função *ReLU* pode ainda ser vista como uma *Leaky ReLU* com coeficiente zero ($a = 0$), o que evidencia a relação de semelhança entre as funções.

	Ativações					
Tipo do problema	ReLU	Leaky ReLU	Swish	Linear	Softmax	Sigmoid
Classificador	13	1	1	2	11	7
Feature Pinning	1	1	0	1	0	0
Segmentação de objetos	2	1	0	0	1	2
Detecção de objetos	3	1	0	0	0	2
Total	19	4	1	3	12	11

Tabela 2.9: Frequência de uso das funções de ativação em cada tipo de problema

Também é importante apontar onde essas funções de ativação foram utilizadas. Dentre todas as soluções observadas, a função *ReLU* nunca foi utilizada na camada de saída (provavelmente porque seu domínio de saída é infinito). Ao passo que quando as funções *softmax* e *sigmoid* foram utilizadas, estas sempre foram utilizadas na camada de saída (provavelmente por conta de suas definições matemáticas serem atreladas à ideia de probabilidade).

2.10 Quais camadas utilizar?

Antes de prosseguir para as análises dos dados, é importante ressaltar que as camadas a serem utilizadas dependem de como o programador pretende lidar com os dados ao longo da rede e da

forma como o modelo precisa ser estruturado para o problema. Ou seja, é necessário entender as operações que cada camada faz para saber quando e quais camadas utilizar. Também é interessante estudar a arquitetura de modelos consolidados da literatura, como *ResNet* [8], para entender como ele combina e agrupa diferentes tipos de camadas.

As análises a seguir farão observações referentes à frequência de uso de algumas camadas e de alguns modelos da literatura. Assim, espera-se apontar quais as camadas mais utilizadas e se há alguma relação de agrupamento/combinação entre elas.

Como a Tabela 2.10 mostra, as camadas mais utilizadas são, respectivamente: *Conv2D, Dropout, MaxPooling2D, BatchNormalization, Dense* e *Flatten*. E vale ressaltar que as frequências mostradas na referida tabela não levam em conta as utilizações das camadas nas soluções que utilizam apenas modelos prontos. Ou seja, se uma solução utilizou apenas modelos da literatura, essa não contabilizou a utilização de nenhuma camada.

Nesses casos o que foi contabilizado foi apenas o uso do modelo, ver Tabela 2.11. Nesta, foram contabilizados os usos de modelos da literatura ao longo das soluções, independente da solução ter utilizado somente modelos prontos ou não. E como pode-se observar, os modelos mais utilizados são da arquitetura *ResNet*, provavelmente por conta de sua popularidade e consolidação na comunidade, visto que esta arquitetura surgiu em 2015.

Agora é interessante observar que a arquitetura *ResNet* é constituída basicamente das camadas *Conv2D, BatchNormalization, AvgPool2D, Dense*, residual, e utiliza ativações *ReLU* ao longo da rede e *softmax* na camada de saída. Ou seja, os modelos mais utilizados também utilizam algumas das camadas mais utilizadas observadas. Para entender um pouco sobre como a *ResNet* agrupa/combina as camadas *Conv2D, BatchNormalization* e a ativação *ReLU* veja a Seção 2.12.

2.11 Quantas camadas o modelo deve ter?

A quantidade de camadas de um modelo está relacionada à complexidade capturada pelos filtros. De forma simplificada, quanto mais camadas, mais detalhes serão capturados e considerados semanticamente pelo modelo, ou seja, o modelo vai aprender cada vez mais a como relacionar os detalhes capturados entre si. Redes mais profundas (com uma quantidade maior de camadas) tendem a obter maior acurácia [9].

Camadas “maiores” e em quantidades maiores melhoram os modelos, mas também aumentam o poder computacional necessário para treiná-los, ou seja, requerem mais horas para executar o treinamento. Além desse aumento no poder computacional, camadas demais (ou grandes mais) aumentam a chance de *overfitting* (sobreajustes), pois os filtros podem ficar específicos demais [10]. E por isso faz-se necessária a utilização de estratégias para mitigar esse tipo de problema decorrente de modelos muito grandes.

Existe ainda um outro problema no treinamento de redes que apresentam uma grande quan-

Camada	Frequência
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
MBConv	1
Multiply	1
Sigmoid	1
Subtract	1
Swish	1

Tabela 2.10: Frequência de uso de cada camada.

tidade de camadas, o *vanishing gradient* [11]. Nesse problema, o vetor gradiente, gerado para calibrar os pesos da rede, tem seus valores diminuídos progressivamente à medida que avança da última até a primeira camada, fazendo com que as primeiras camadas não sofram ajustes significativos ao longo do treinamento. Por isso, mesmo que o programador deseje colocar uma grande quantidade de camadas em seu modelo é preciso saber dosá-las, ou mitigar este problema com o uso de técnicas específicas, vide a estratégia adotada pela arquitetura *ResNet* (explicada brevemente na próxima seção).

2.12 Como combinar camadas?

Combinar as camadas corretamente para criação de um modelo é um passo fundamental e delicado. Não existe muita limitação quanto a como combiná-las, e em muitos casos é um processo que envolve muita tentativa e erro. No entanto, existem intuições que é bom conhecer, sobre como combinar certas camadas. Por exemplo, no artigo original da ResNet [8], os autores

Modelo	Frequência
ResNet34	3
ResNet50	3
EfficientNetB3	2
Siamese Neural Network	2
DenseNet121	1
DenseNet201	1
EfficientNetB2	1
EfficientNetB4	1
EfficientNetB5	1
InceptionResNetV2	1
InceptionV4	1
MobileNet	1
Multilayer Perceptron	1
ResNeXt50	1
Xception	1

Tabela 2.11: Frequência de uso de cada modelo.

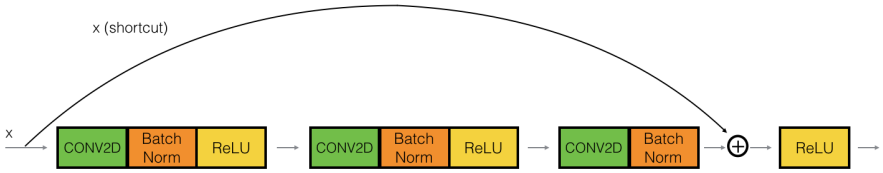


Figura 2.5: *Building block* (bloco de construção) da arquitetura *ResNet*[4].

explicam como as camadas foram organizadas em "blocos" (intuição), e explicam também as características específicas de cada bloco. A seguir está um breve resumo do funcionamento destes blocos.

A arquitetura *ResNet* utiliza as camadas *Conv2D, BatchNormalization* e a ativação *ReLU* para compor os chamados *building blocks* (blocos de construção). Estes blocos seguem a estrutura presente na Figura 2.5, começando com uma camada *Conv2D*, seguida de uma *BatchNormalization* e uma ativação *ReLU*, em seguida essa sequência se repete mais duas vezes e, por fim, acontece o chamado *shortcut* (atalho), cujo objetivo é fornecer um caminho alternativo para o gradiente durante a etapa de aprendizado, que serve para mitigar o problema de *vanishing gradient*. O que acontece no encontro do "caminho normal" com o atalho é simplesmente uma adição da entrada do bloco com a saída gerada pelo "caminho normal".

[4]https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

2.13 Quando utilizar *transfer learning*?

Em algumas situações executar o treinamento de um modelo pode ser um processo muito custoso (em termos de dinheiro e/ou tempo). E nesses casos pode-se optar por utilizar a estratégia de treinamento *Transfer Learning*, que consiste em reaproveitar conhecimento obtido previamente pelo mesmo modelo numa situação/problema diferente.

Intuitivamente pode-se pensar que para o bom funcionamento desta técnica seja necessário que o contexto alvo seja semelhante ao contexto em que o modelo foi previamente treinado. Mas este não é um requerimento para o uso desta técnica [12].

E isso é o que os dados coletados sugerem, pois não há uma tendência clara sobre quando utilizar a técnica baseando-se apenas no contexto, ver Tabela 2.12.

Contexto	# Usa *transfer learning*	# Não usa *transfer learning*
OCR	0	3
Flora	2	2
Fauna	3	1
Biometria	1	1
Análise de Imagens Aéreas	3	0
Medicina	3	2
Rabisco	0	1
Câmera de Painel	1	0
Terreno	1	0
Controle de Qualidade; Produção de Ferro	1	0
Descrição de Arte	1	0

Tabela 2.12: Frequência da técnica de *transfer learning* em cada tipo de contexto.

Vale ainda citar algumas das recomendações feitas por Chris von Csefalvay [13]. Segundo ele, a técnica é melhor aplicada quando se utiliza modelos pré-treinados em bases de dados muito maiores que a base final, pois na base maior podem ocorrências de casos muito específicos que talvez não estejam presentes na base final. O autor também afirma que mesmo em bases balanceadas, a utilização da técnica pode gerar ganhos, caso haja a possibilidade de *overfitting* (no caso de uma base pequena, por exemplo).

2.14 Qual o modelo mais indicado para aplicar a técnica de *transfer learning*?

Novamente, para escolher um modelo deve-se ter uma noção das operações que ele executa e o ambiente em que o modelo será executado. Pois, dependendo das limitações do ambiente de execução pode ser inviável utilizar alguns modelos pré-treinados disponíveis. Por exemplo, uma máquina com pouco poder de processamento não será capaz de lidar com modelos profundos.

Tendo isso em mente, as soluções anotadas utilizaram as seguintes arquiteturas (ordenadas por quantidade de usos): *ResNet, EfficientNet, Inception, DenseNet, ResNeXt, InceptionResNet, VGG, Xception*. Onde *ResNet* e *EfficientNet* aparecem em mais da metade das soluções com *transfer learning*, cerca de 68% (11 de 16).

Cada uma das arquiteturas citadas apresenta uma abordagem diferente, com vantagens e desvantagens que nem sempre se traduzem em melhores resultados nas competições. Além disso, cada arquitetura apresenta uma variedade de modelos específicos, como *ResNet-34, ResNet-50, InceptionV1, InceptionV3*, etc. Por isso, é muito difícil afirmar um motivo técnico para maior utilização de uma determinada arquitetura ou modelo. Assim, a utilização maior das arquiteturas *ResNet* e *EfficientNet* provavelmente ocorre devido a sua popularidade e facilidade de uso, pois *ResNet* surgiu em 2015 e os modelos *EfficientNet* [14] foram disponibilizados na biblioteca *TensorFlow* no momento de sua criação, em 2019.

Assim, os dados (ver Tabela 2.13) indicam que se o programador deseja utilizar *transfer learning*, é recomendado utilizar algum modelo *ResNet* ou *EfficientNet*, pois cada um desses tipos possui diversos modelos com configurações específicas (como *ResNet34, ResNet50, EfficientNetB2, EfficientNetB3*, etc) e que exigem diferentes poderes de processamento.

Modelos	Frequência
ResNet	7
EfficientNet	5
Inception	3
DenseNet	2
ResNeXt	1
InceptionResNet	1
VGG	1
Xception	1

Tabela 2.13: Frequência de uso de cada modelo com a técnica de *transfer learning*

2.15 O que fazer na etapa de treinamento?

Durante esta etapa é muito importante ter consciência de que alcançar o melhor resultado possível numa métrica não necessariamente significa ter feito o melhor modelo, ou o melhor treinamento. Pois, isso pode se dar devido ao fenômeno do *overfitting* (sobreajuste), ou seja, o modelo conseguiu predizer muito bem os dados de treinamento, mas isso não se mantém para dados novos.

Por isso, saber o que fazer no treinamento é fundamental para criação de modelos generalizáveis sem perda de eficiência. Dessa forma, é necessário fazer algum tipo de validação e/ou tratamento, para garantir que o modelo está conseguindo aprender sem perder generalização.

Dentre as técnicas utilizadas pelas soluções observadas (ver Tabela 2.14), vale apontar: divisão simples da base em "treinamento" e "validação" foi a técnica mais utilizada (chamada de "divisão simples" na Tabela 2.14); utilização de *twin dataset*, que consiste em apresentar para a rede imagens de mesma classe, para que ela aprenda a aproximar mais imagens de classes iguais, ao mesmo tempo que aprende a separar imagens de classes diferentes; divisão da base utilizando *K-Fold*; embaralhamento das imagens na base de treinamento; utilização da técnica de *pseudo-labelling* para aumentar o tamanho da base de dados e aumentar a generalização do modelo; e utilização da técnica de *data augmentation*, também para aumentar a generalização do modelo.

Técnicas de treinamento	Frequência
Divisão simples; *data augmentation*	9
Divisão simples	1
Data augmentation	1
Twin dataset; taxa de aprendizagem variável; *data augmentation*	1
Treinamento de 3 modelos e a resposta final é a média ponderada dos 3; *data augmentation*	1
Stratified K-Fold (K = 10); *data augmentation*	1
K-Fold (K = 3); *data augmentation*	1
Stratified Shuffle Split; *data augmentation*	1
Pseudo-Labelling; *data augmentation*	1
Multilabel Stratified Shuffle Split; *oversampling*; *data augmentation*	1
CNN com múltiplas saídas; *data augmentation*	1
Pares de imagens (50% de baleias equivalentes e 50% de baleias diferentes); regularização L2; *data augmentation*	1
Treinamento com política de 1 ciclo; *data augmentation*	1
Treinamento em ciclos; treinamento com e sem *data augmentation*; redimensionamento das imagens entre ciclos	1
3 treinamentos, aumentando o tamanho do *batch* progressimanete	1
Gradiente descendente em todos os casos	1
Remover todas as imagens sem a presença da classe alvo	1
Combinação de modelos	1

Tabela 2.14: Frequência de uso das técnicas de treinamento

2.15.1 Quando utilizar a estratégia de *data augmentation*?

A técnica de aumento de dados (*data augmentation*) é utilizada no treinamento com o objetivo de aumentar a generalização do modelo e garantir que uma imagem nunca seja apresentada da mesma forma mais de uma vez para a rede.

Com isso em mente, os dados apontam que se a base tiver alguma classe com poucos dados, menos de 1.000 casos (ver Tabela 2.15), é recomendado utilizar a técnica em questão. Visto que 1.000 é a quantidade mínima ideal de casos segundo a "regra de ouro" mencionada

anteriormente na Seção 2.5.

Também foi observada uma alta utilização nas soluções de problemas que apresentavam dados uniformes, cerca de 84,6% (11 de 13). Visto que essa padronização pode ocasionar uma pobre generalização do modelo, é recomendado utilizar *data augmentation* para inserir uma certa variabilidade aos dados de treinamento.

Além disso, foi observada uma taxa de utilização especialmente alta nas soluções de problemas do tipo "classificador", 17 de 19 soluções (cerca de 95%) utilizaram a técnica (ver Tabela 2.16), e do tipo "segmentação de objetos", em que 100% das soluções também utilizaram a técnica (mesmo tendo apenas 3 competições essa observação não deve ser desvalorizada).

		Usa *data augmentation*	
		Verdadeiro	Falso
Pelo menos 1000 imagens por classe	Dados não uniformes	4	3
Pelo menos 1000 imagens por classe	Dados uniformes	5	2
Menos de 1000 imagens numa classe	Dados não uniformes	6	0
Menos de 1000 imagens numa classe	Dados uniformes	6	0

Tabela 2.15: Frequência de uso da técnica *data augmentation* em bases com poucos dados ou dados não uniformes

Tipo de problema	Usa *data augmentation*	Frequência
Classificador	Falso	2
Classificador	Verdadeiro	17
Feature Pinning	Falso	1
Detecção de objetos	Falso	2
Detecção de objetos	Verdadeiro	1
Segmentação de objetos	Verdadeiro	3

Tabela 2.16: Frequência de uso da técnica *data augmentation* em cada tipo de problema

2.16 Como validar o modelo?

Como já citado anteriormente, existem algumas técnicas para separar a base em "conjunto de treinamento" e "conjunto de validação", como *K-Fold, Stratified Shuffle Split*, etc. No entanto, os dados mostram que a técnica mais utilizada é a divisão simples, onde uma porcentagem da base será destinada ao treinamento e a outra à validação.

2.16.1 Qual deve ser o tamanho da base de validação?

Os dados indicam que deve-se utilizar no máximo algo em torno de 30% da base de treinamento para validação, pois cerca de 70% das soluções (18 de 26) assim o fazem. E, vale ressaltar, que

a divisão mais utilizada foi reservar 10% da base para validação, cerca de 30% das soluções (8 de 26) fizeram isso (ver tabela 2.17).

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 2.17: Frequência de uso de cada tamanho de conjunto de avaliação

3

Conclusão

O presente trabalho abordou diferentes tipos de problemas de visão computacional e quais as abordagens mais comuns para cada um. O objetivo era discutir sobre algumas perguntas norteadoras baseando-se nesse conhecimento prévio, ou seja, as abordagens mais recorrentes para cada tipo de problema.

Primeiramente, para viabilizar o desenvolvimento desta monografia, foi feito um trabalho de pesquisa com o objetivo de coletar dados de diversas competições de visão computacional da plataforma Kaggle. Foram anotados dados referentes às características dos problemas, de suas soluções e de suas bases de dados (Apêndice A). A próxima etapa da pesquisa consistiu em fazer diversas filtragens para manter apenas os dados das competições com resultados que atendessem certas métricas de qualidade.

Em posse dos dados filtrados, foram feitos resumos de cada competição e suas soluções (Apêndice B), a fim de descrever textualmente, e não em forma codificada (como encontrada no Kaggle), cada abordagem apresentada, bem como seus resultados.

E com isso, para este trabalho foram levantadas algumas perguntas relevantes que aparecem durante o desenvolvimento de uma nova solução para um problema de visão computacional. Estas perguntas foram respondidas a partir de discussões feitas sobre os dados coletados. Nas discussões, as abordagens observadas mais recorrentes indicavam uma tendência a ser seguida para certos tipos de problemas, e com isso, foram levantadas explicações para tais observações. Devido ao fato dos dados embasadores das discussões terem sido obtidos de boas soluções já existentes, espera-se que as conclusões reflitam a qualidade das soluções analisadas.

Além disso, a base de conhecimento obtida ao final do trabalho de pesquisa possui muita relevância para a área. Pois constitui um sumário de abordagens para cada tipo de problema, além disso, foi obtida através de vários processos humanos que não podem ser automatizados, e contém somente dados de boas soluções já existentes. Por isso, ela pode ser reutilizada em trabalhos futuros da área, ou ser utilizada como forma de consulta rápida para problemas específicos

muito parecidos com os documentados. A base construída está disponível em http://bit.ly/tcc-kaggle-database.

Desse modo, conclui-se que a partir da análise dos dados de competições de visão computacional foi possível levantar e discutir questionamentos relevantes para novas soluções de problemas futuros nesta área. Vale ressaltar que, o domínio de discussão não cobre todo o espectro de problemas possíveis de visão computacional, e quanto mais competições forem analisadas maior será a relevância de tais discussões e observações.

3.1 Trabalhos futuros

Para um trabalho futuro deseja-se utilizar o conhecimento e as análises construídas ao longo desta monografia para criar um guia de como criar uma nova solução para problemas de visão computacional, voltado para iniciantes na área.

Além disso, seria interessante que o guia fosse posto em prática, sendo utilizado para criar soluções de problemas já existentes, a fim de validá-lo e consequentemente reafirmar a importância das discussões aqui presentes.

Referências bibliográficas

[1] Louis Columbus. How artificial intelligence is revolutionizing business in 2017, 2017.

[2] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, pages 2–2, 2018.

[3] Thomas Elliott. The state of the octoverse: machine learning, 2019.

[4] Harkiran Kaur. Why is python the best-suited programming language for machine learning?, 2019.

[5] Pete Warden. How many images do you need to train a neural network?, 2017.

[6] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, page 319, Berlin, Heidelberg, 1999. Springer-Verlag.

[7] Yutaka Sasaki. The truth of the f-measure. *Teach Tutor Mater*, 01 2007.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 06 2016.

[9] Koustubh Sinhal. Resnet, alexnet, vggnet, inception: Understanding various architectures of convolutional networks, 2018.

[10] Michael Grogan. Is there any relation between number of hidden layers in a neural network and performance?, 2018.

[11] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[12] 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.

[13] Chris von Csefalvay. Transfer learning: the dos and don'ts, 2019.

[14] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[15] Antônio Houaiss. *Dicionário Houaiss da Língua Portuguesa*, volume ? Editora Objetiva, 2001.

[16] Jason Brownlee. 9 applications of deep learning for computer vision, 2019.

[17] Dorian Pyle. *Data Preparation for Data Mining*, volume ? Morgan Kaufmann Publishers, 1999.

[18] Hervé Abdi. Normalizing data, 2010.

[19] Urvashi Jaitley. Why data normalization is necessary for machine learning models, 2018.

[20] Richard Liaw Daniel Ho, Eric Liang. 1000x faster data augmentation, 2019.

[21] Peter König Alex Hernández-García. Data augmentation instead of explicit regularization. *arXivpreprint arXiv:1806.03852*, pages 1–2, 2018.

[22] Miguel Cárdenas-Monte. Sobreajuste - overfitting. Course notes, October 2015. Acesso em 05 jan. 2020.

[23] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutionalneural networks. *Neural Information Processing Systems (NIPS)*, pages 5–5, 2012.

[24] Algorithmia. Introduction to loss functions, 2018.

[25] François Chollet et al. Keras. https://keras.io, 2015.

[26] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[27] 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.

[28] 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.

[29] Paul-Louis Pröve. Squeeze-and-excitation networks, 2017.

[30] Vinay Uday Prabhu. Kannada-mnist: A new handwritten digits dataset for the kannada language. *arXiv preprint arXiv:1908.01242*, 2019.

[31] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[32] 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[1]

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

[1] Autores: Daniel Humberto Cavalcante Vassalo, Nelson Gomes Neto

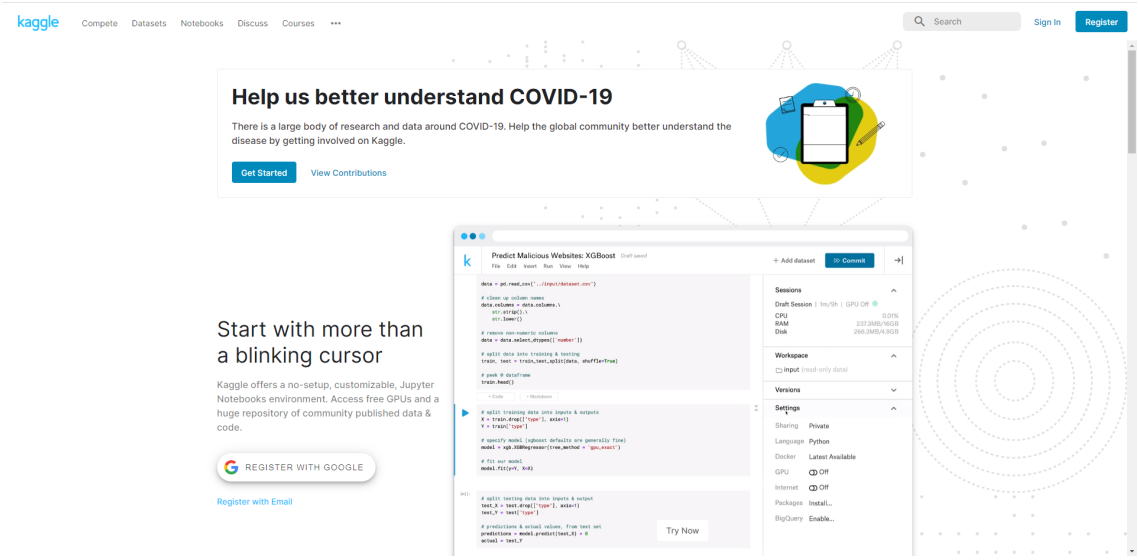


Figura A.1: Página inicial do Kaggle

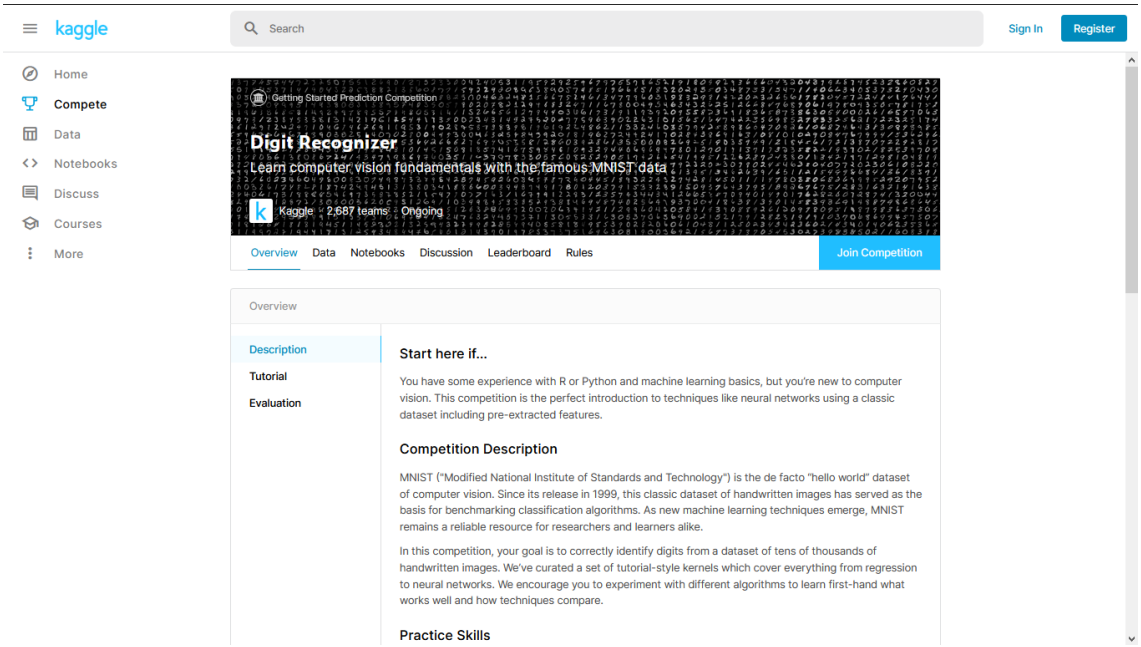


Figura A.2: Descrição de uma competição do Kaggle

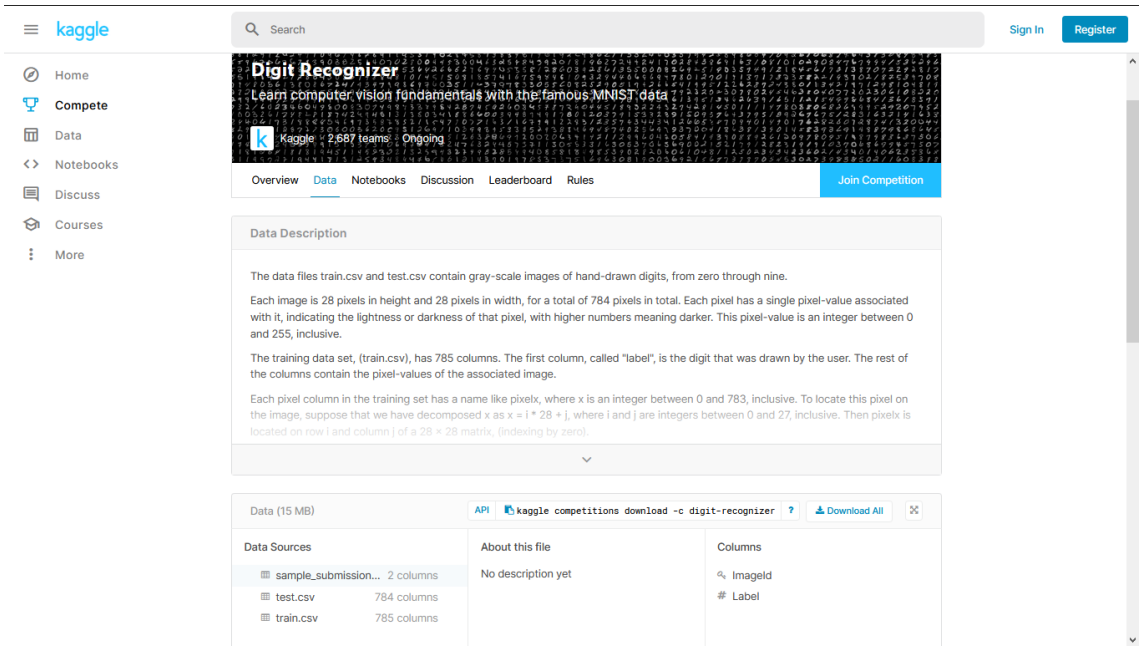


Figura A.3: Descrição dos dados de uma competição do Kaggle

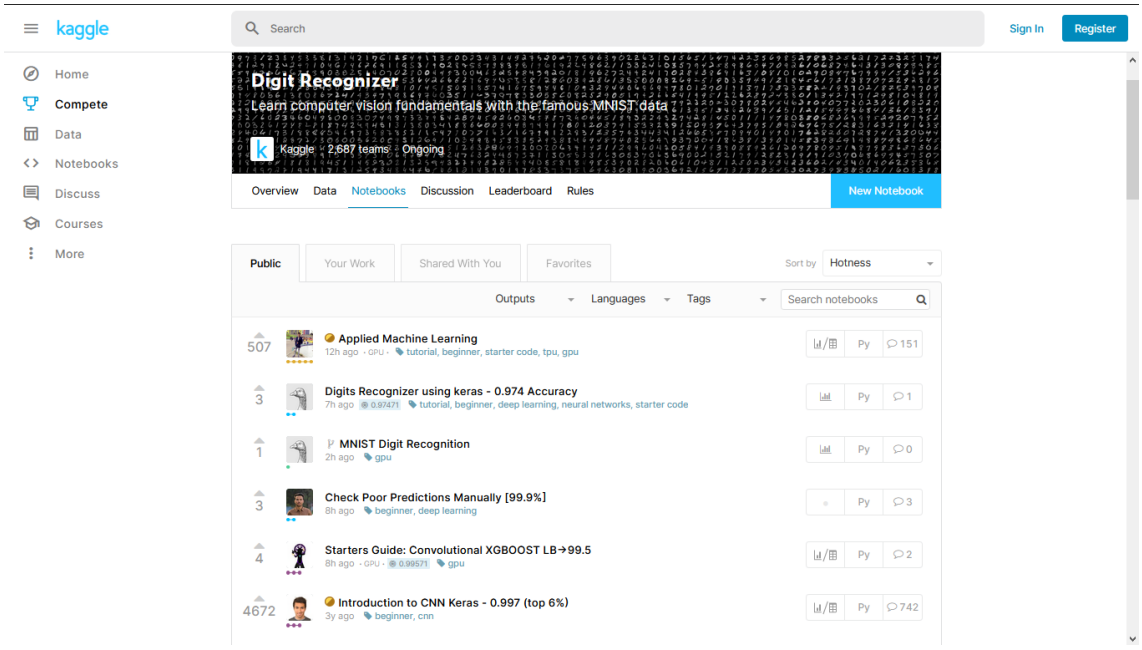


Figura A.4: *Notebooks* de uma competição do Kaggle

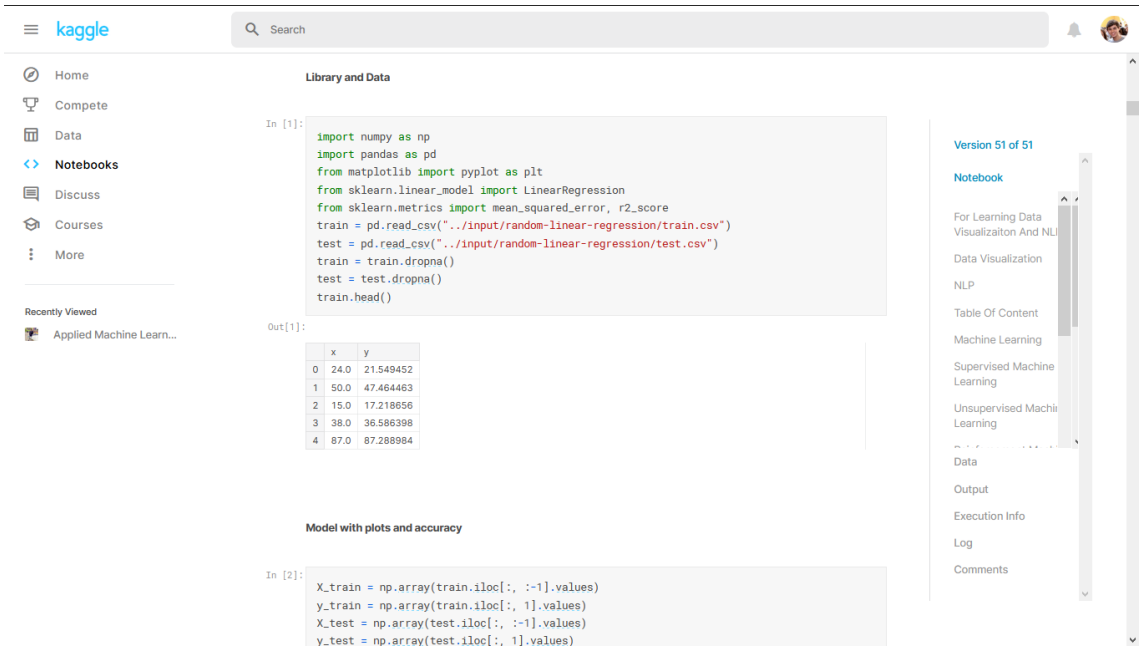


Figura A.5: Trechos de código dentro de um *notebook*

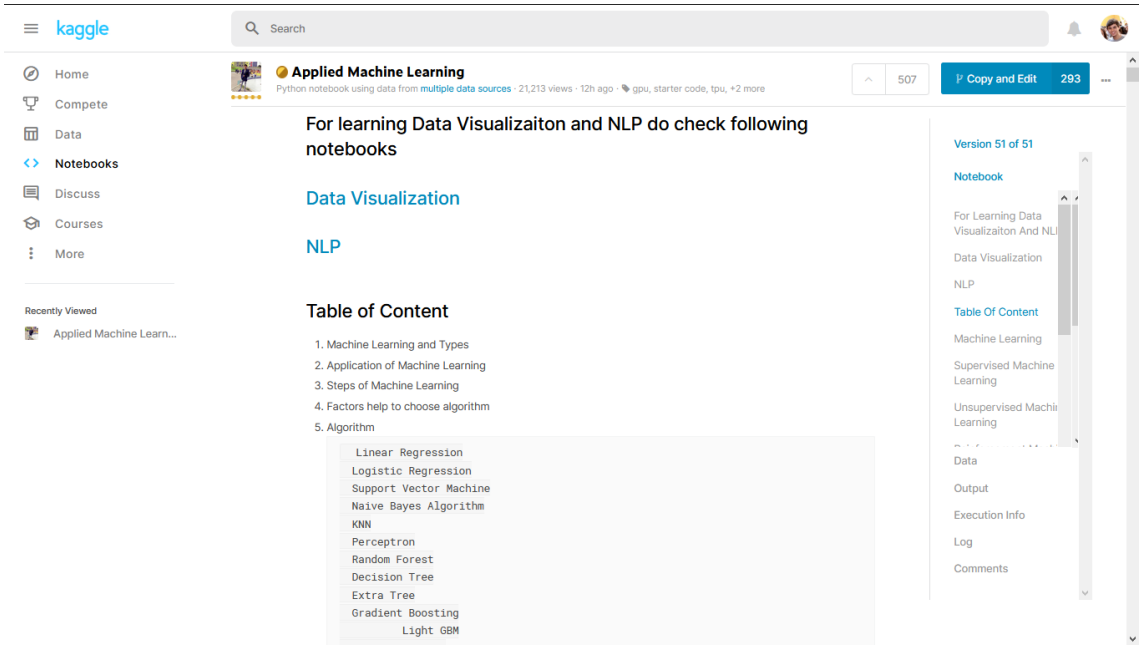


Figura A.6: Texto rico dentro de um *notebook*

Tabela A.1: Placares público e privado

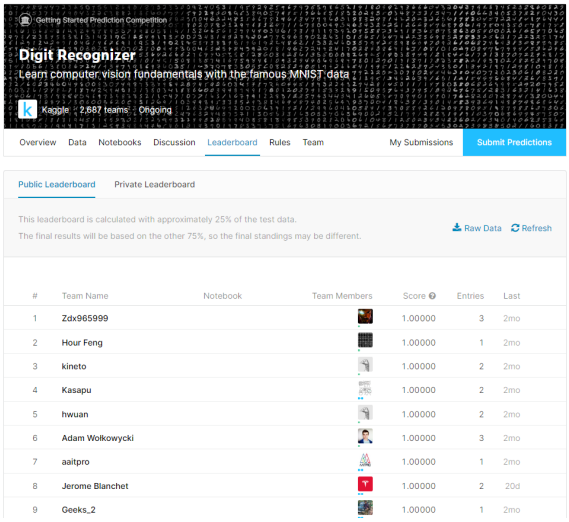


Figura A.7: Placar público

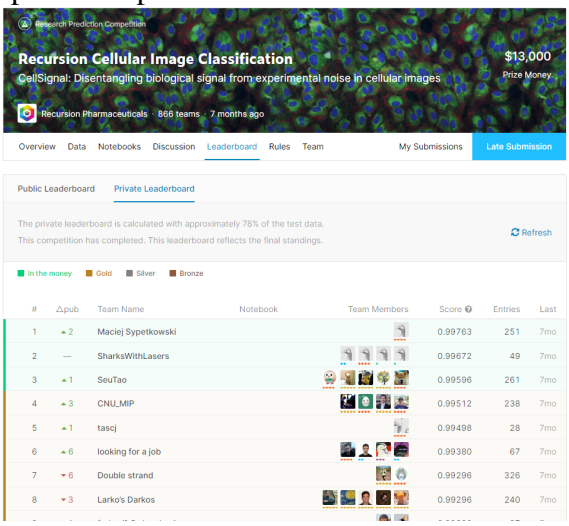


Figura A.8: Placar privado

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 (publica 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 possuíssem 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 à 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[2]. 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" [15]. 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.

[2]Disponível em https://docs.google.com/spreadsheets/d/1BKJ9_mIQdU504tIoJ7zHmxJeqKQo7gs3buQL9D7ilMs

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 dividas 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 [16]:

- 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 pixeis 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 mascaras. 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 [5]. 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 [17]. 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 [18]. 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" [19]. 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" [20].

Esta é uma técnica muito comum de ser utilizada para evitar o problema de *overfitting* (sobreajuste) [2, 21]. *Overfitting* é um fenômeno estatístico, onde o modelo se ajusta muito bem (consegue fazer excelentes predições) sobre os dados de treinamento, mas tem um péssimo desempenho em predizer novos resultados (fora dos dados de treinamento) [22].

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 [23, 2].

Alguns dos "aumentos" possíveis incluem transladar as imagens por uma quantidade aleatória de pixeis, 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 [12]. 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" [24]. Em outras palavras, a função de perda quantifica, matematicamente, o quão distante do esperado estão as predições 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 [25]. 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[1]

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[2] 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[3] [26]. Para esta competição, foram selecionadas 42.000 imagens de dimensões 28x28 pixeis, 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.

[1] Autores: Daniel Humberto Cavalcante Vassalo, Nelson Gomes Neto

[2] 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.

[3] Disponível em: http://yann.lecun.com/exdb/mnist/

Tabela B.1: Imagens do Digit Recognizer

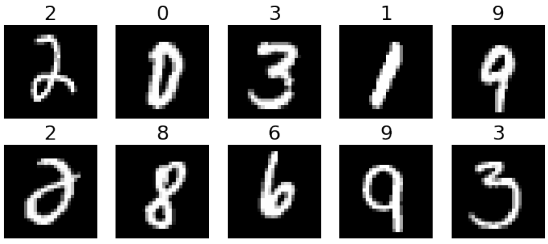


Figura B.1: Exemplos da base de dados MNIST

B.1.2 Solução

O autor da solução selecionada[4], 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

[4] 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 [27] 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 pixeis, e comportamento uniforme, pois sempre que um cacto está presente ele aparece no centro da imagem.

B.2.2 Solução

A solução selecionada[5] 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 118° 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*.

[5] 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

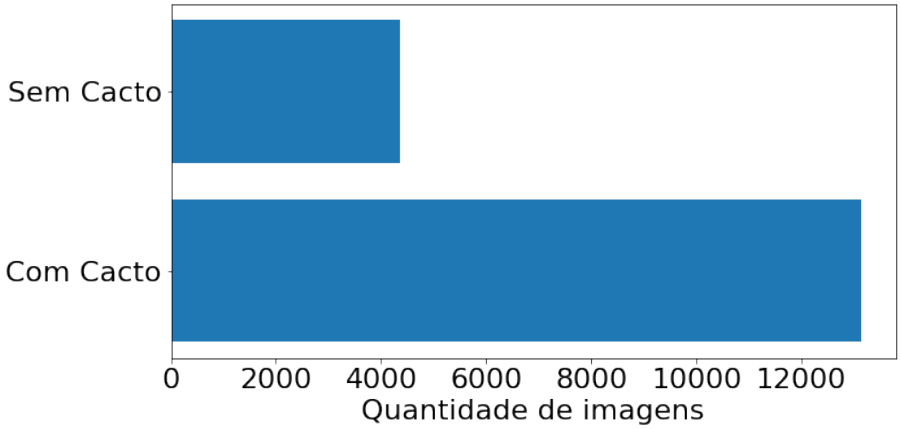


Figura B.4: Distribuição da base de dados

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

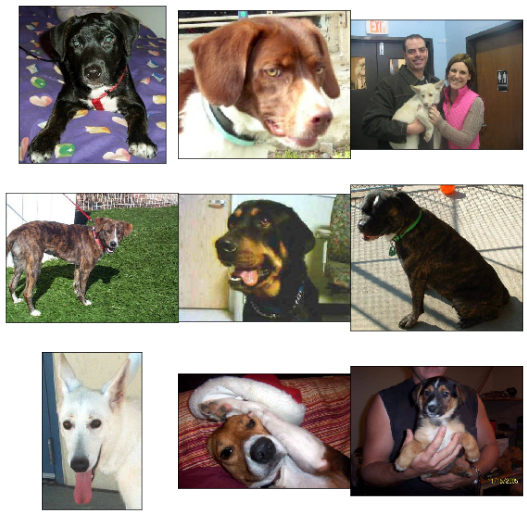


Figura B.6: Exemplos de imagens com cachorros

B.3.2 Solução

Na etapa de pré-processamento da solução[6] 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 funcion*, 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*.

[6] 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 pixeis) 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.

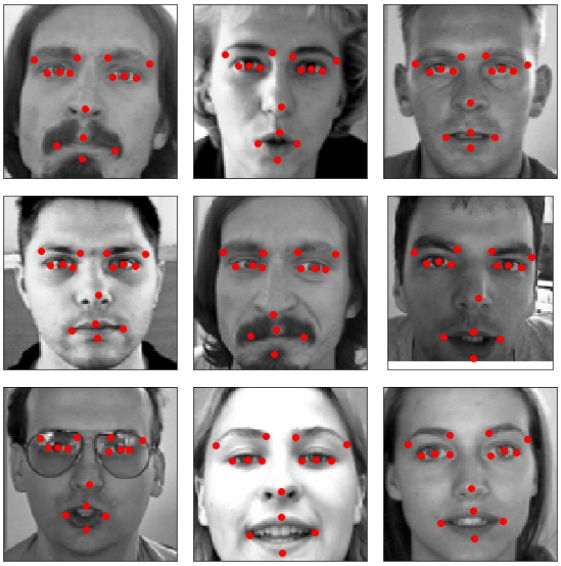


Figura B.7: Exemplos da base de dados, com os pontos-chave representados em vermelho

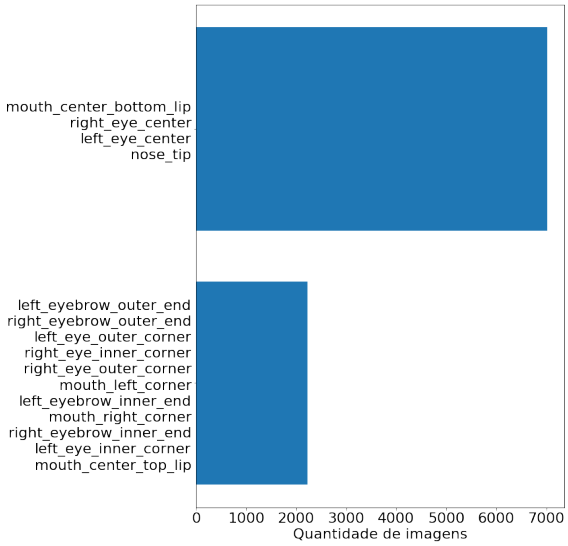


Figura B.8: Frequência da cada ponto-chave

B.4.2 Solução

A solução[7] 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

[7] 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 predizer 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

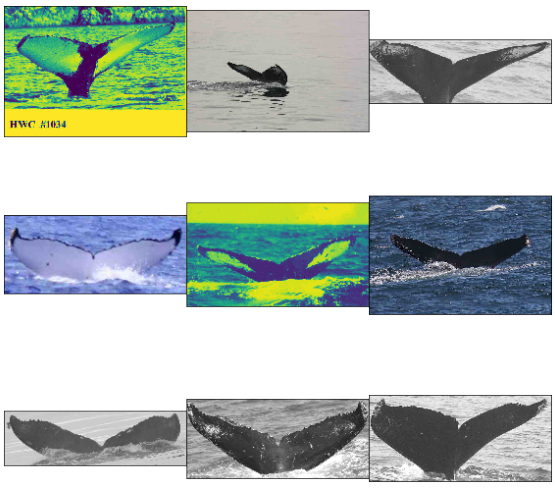


Figura B.9: Exemplos da base de dados

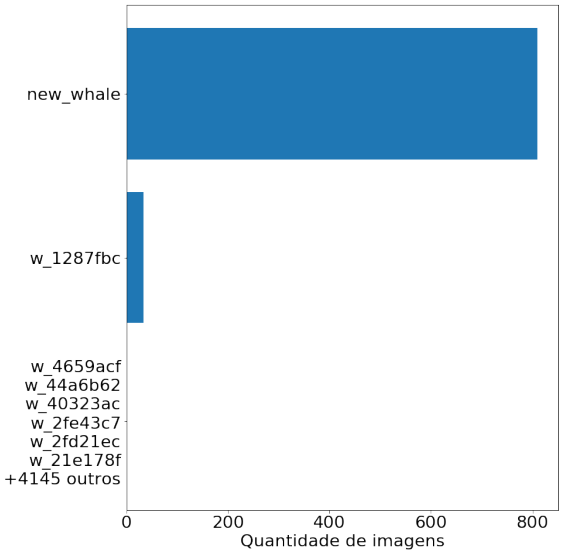


Figura B.10: Frequência de cada baleia

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[8] 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 pixeis, 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

[8] 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 pixeis) 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 [8]. 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, Lamba 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[9] 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 pixeis e as normalizaram.

[9] https://www.kaggle.com/codegass/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

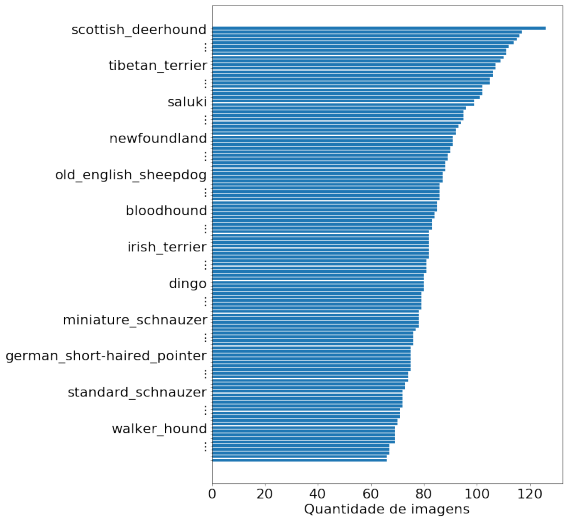


Figura B.12: Frequências de todas classes

B.7.2 Solução

A solução[10] 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* [28], a utilização desta técnica com outros modelos pré-treinados em cima da mesma base poderia gerar um resultado melhor.

[10] 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

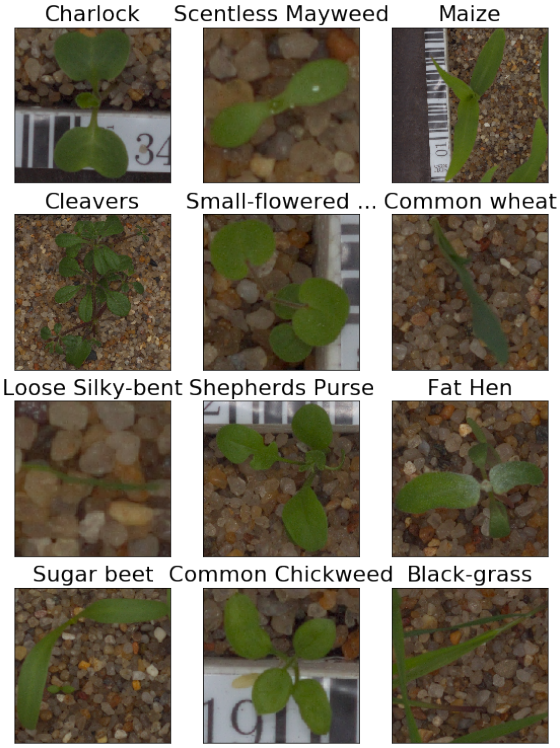


Figura B.13: Exemplos da base de dados

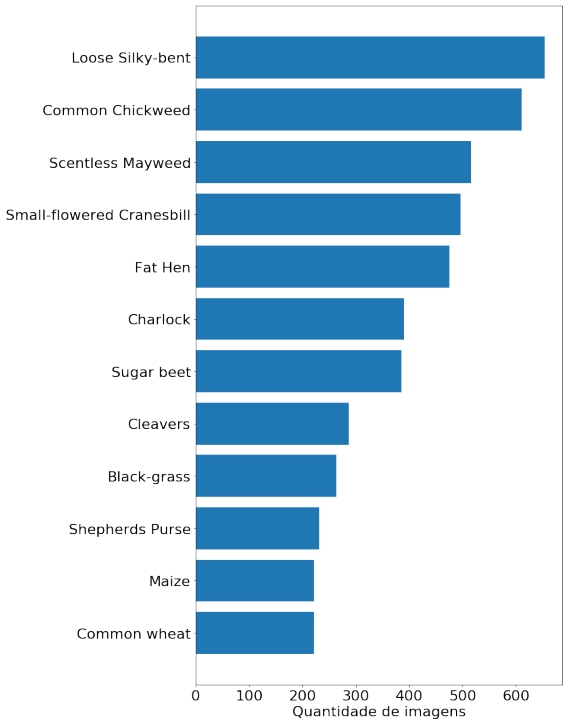


Figura B.14: Frequências de todas classes

B.8.2 Solução

A solução[11] 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 pixeis).

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 pixeis e, devido ao tipo de problema, não possuem um comportamento uniforme.

B.9.2 Solução

A solução[12] 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 pixeis) que o objeto ocupa. Para isso, o modelo proposto foi dividido em duas partes.

[11] https://www.kaggle.com/droid021/fast-ai-v1
[12] https://www.kaggle.com/cdeotte/without-ensemble-lb-0-665

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

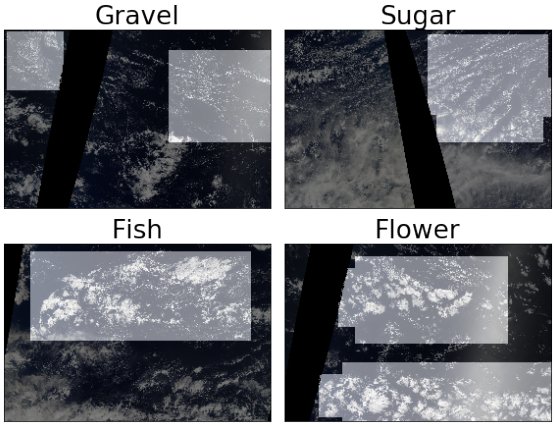


Figura B.15: Exemplos da base de dados com a máscara

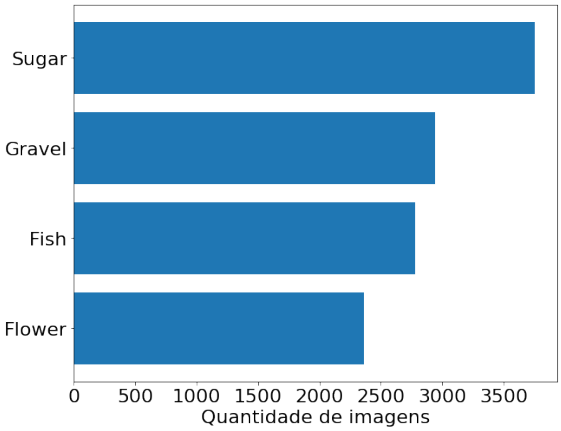


Figura B.16: Frequências de todas classes

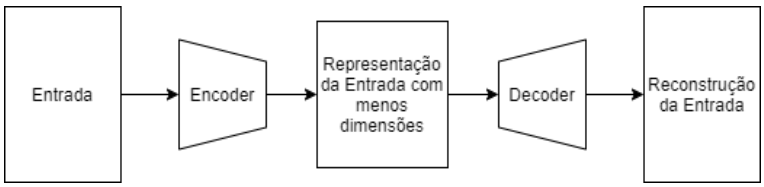


Figura B.17: *Autoencoder*

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 obeteve 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 pertubações 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[13] 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 pixeis.

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 [29]. O modelo construído foi baseado na arquitetura *EfficientNet* e utilizou as seguintes camadas da biblioteca PyTorch: *BatchNorm2d, Conv2D, Flatten, MBConv, 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 melhorada nas predições finais do modelo.

[13] https://www.kaggle.com/hmendonca/fold1h4r3-arcenetb4-2-256px-rcic-lb-0-9759

Tabela B.8: Imagens do Recursion Celluar Image Classification

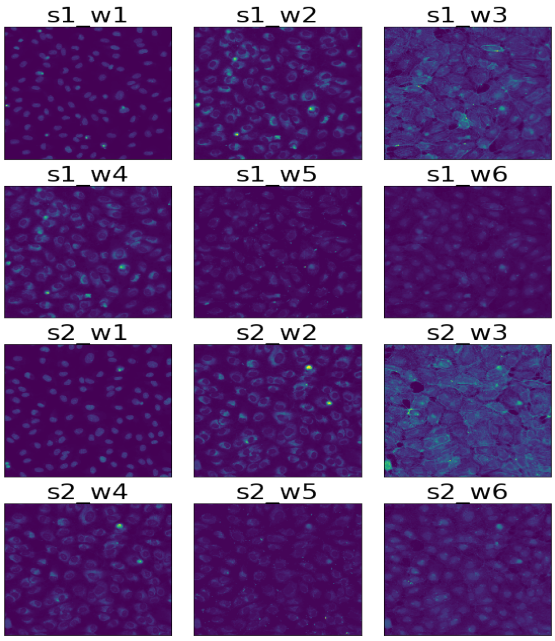


Figura B.18: Exemplo da base de dados (HUVEC-04 Plate4 G08, que apresenta classe: sirna$_1$21)

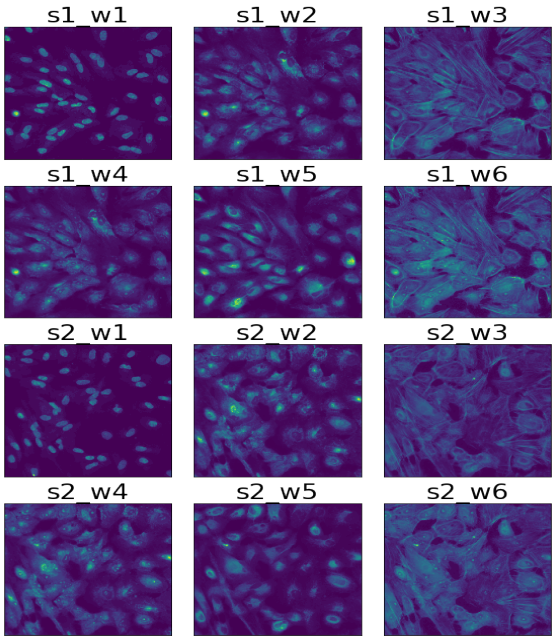


Figura B.19: Exemplo da base de dados ((RPE-04 Plate3 F21, que apresenta classe: sirna$_4$93)

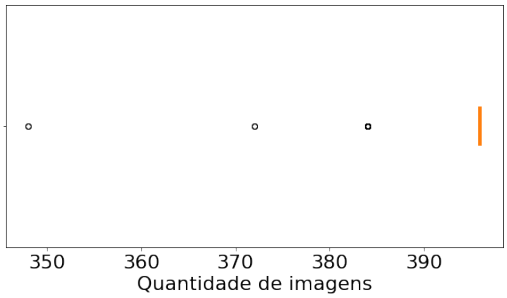


Figura B.20: Distribuição da base de dados

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 predizer 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[14] selecionada, para cada desenho o autor cria 3 imagens de tamanho 128x128x1 pixeis com codificações diferentes e depois concatena as 3 imagens formando uma imagem única com dimensões 128x128x3 pixeis – 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 pixeis 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 ultima 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.

[14] https://www.kaggle.com/echomil/mobilenet-126x126x3-100k-per-class

Tabela B.9: Imagens do Quick, Draw! Doodle Recognition Challenge

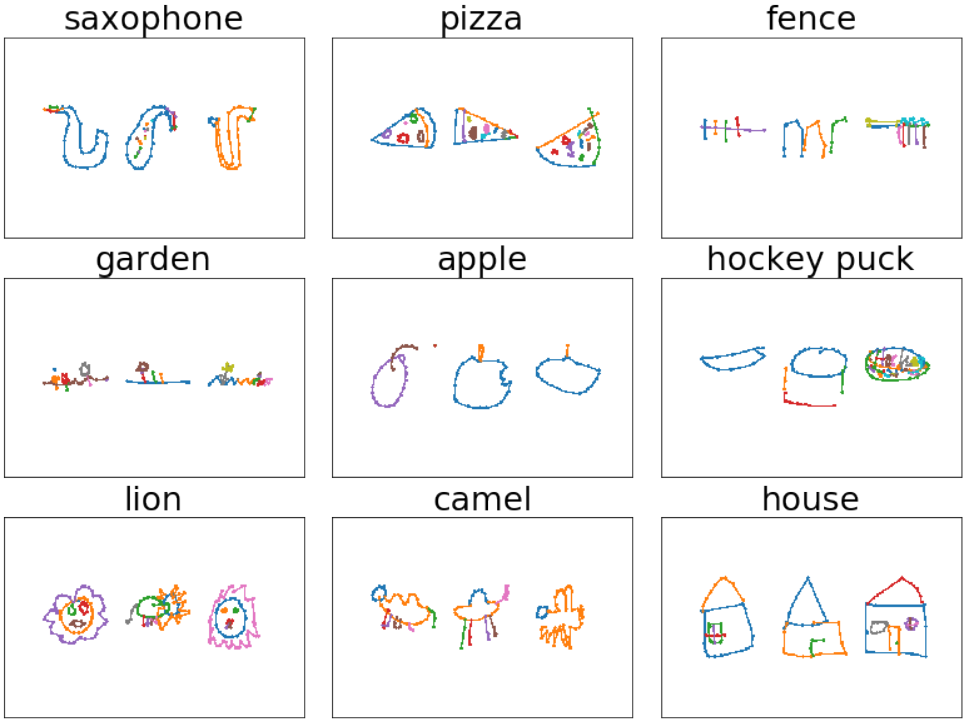


Figura B.21: 3 exemplos de desenho para alguns exemplos de classes

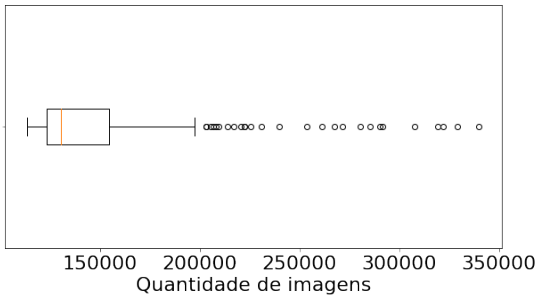


Figura B.22: Distribuição da base de dados

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

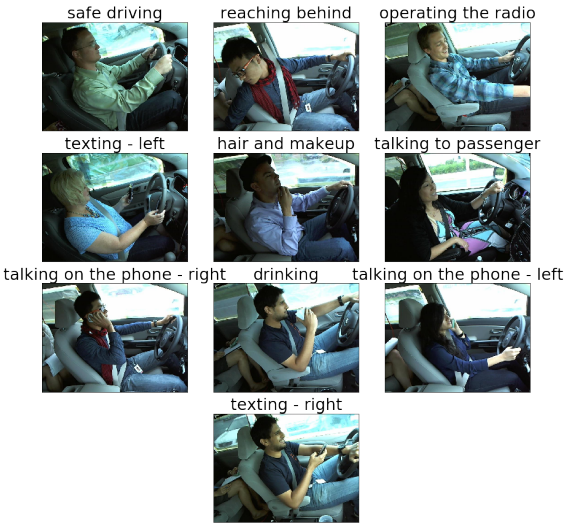


Figura B.23: Exemplos da base de dados

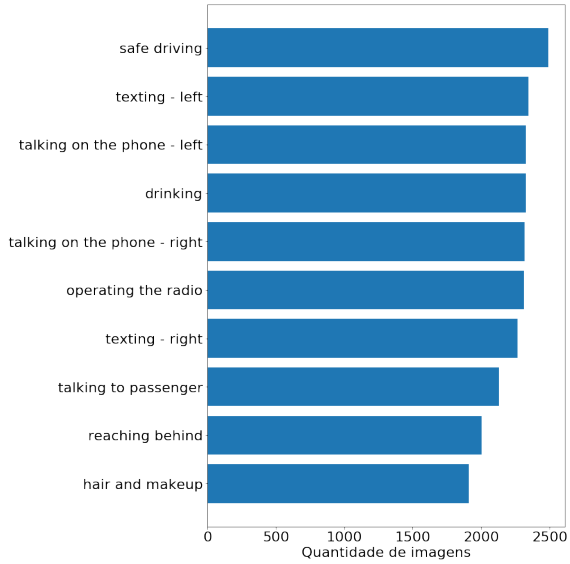


Figura B.24: Distribuição da base de dados

B.12.2 Solução

A solução[15] 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*.

[15] https://www.kaggle.com/byronkd/distracted-driver-fastai-easy-top-15-of-lb

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 [30] é balanceada, apresenta 60.000 imagens de dimensões 28x28 pixeis 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

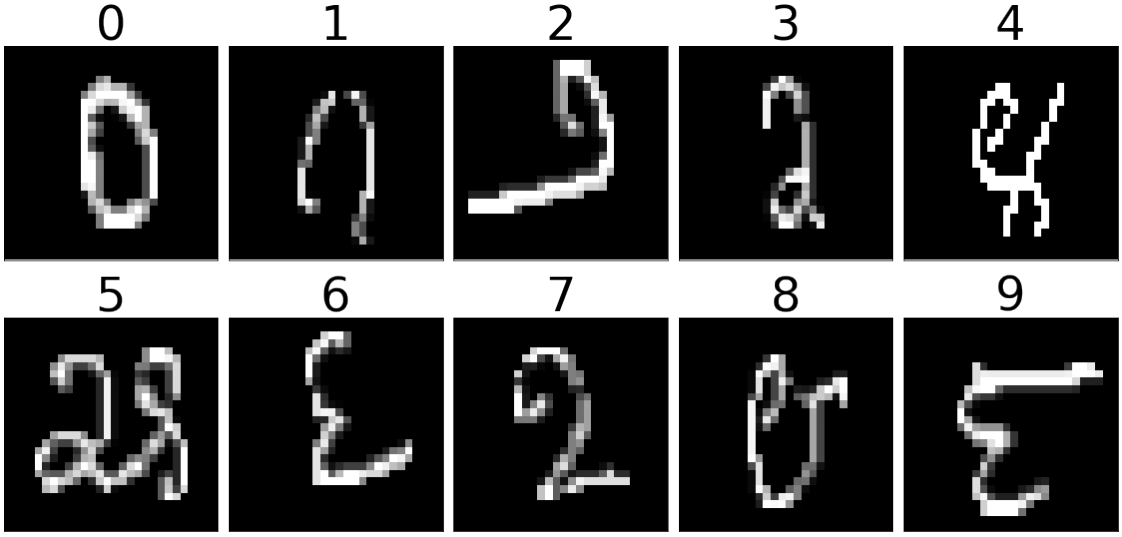


Figura B.25: Exemplos da base de dados

B.13.2 Solução

A solução[16] 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

[16] 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 parar 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 pixeis da folha foram coloridos de branco e os pixeis 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[17] 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 pixeis.

[17] https://www.kaggle.com/abhmul/keras-convnet-lb-0-0052-w-visualization

Tabela B.12: Imagens do Leaf Classification

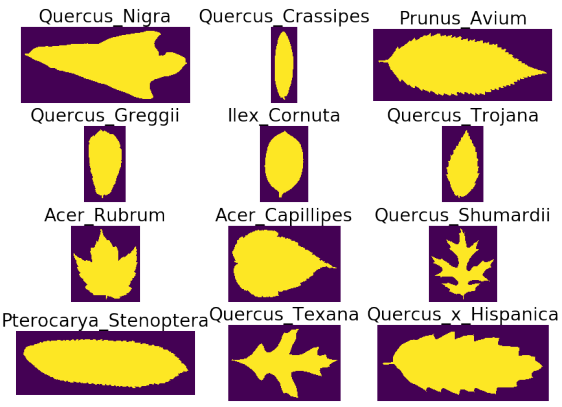


Figura B.26: Exemplos de algumas classes da base de dados

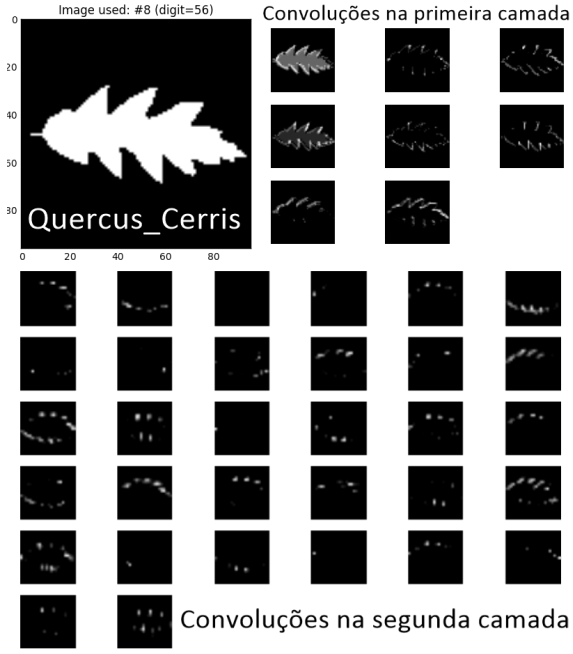


Figura B.27: Convoluções da solução

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

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.

Tabela B.13: Imagens do Invasive Species Monitoring

Figura B.28: Exemplos da base de dados

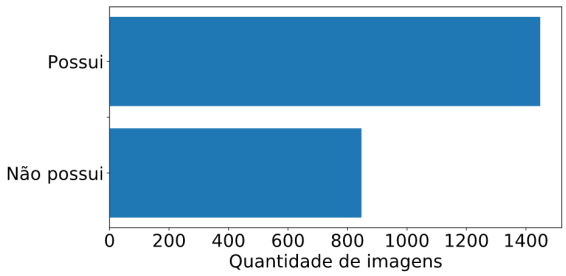


Figura B.29: Distribuição da base de dados

B.15.2 Solução

Na solução[18] selecionada, durante a fase de pré-processamento o autor apenas normalizou e redimensionou todas as imagens para as dimensões 800x800x3 pixeis.

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 cadama *Dropout* seguida de uma camada

[18] 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 270º 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[19] selecionada apresenta uma etapa de pré-processamento bem direta, consistindo basicamente de um redimensionamento de todas as imagens para as dimensões 256x256x3 pixeis 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*.

[19] https://www.kaggle.com/krishnakatyal/keras-efficientnet-b3

Tabela B.14: Imagens do RSNA Intracranial Hemorrhage Detection

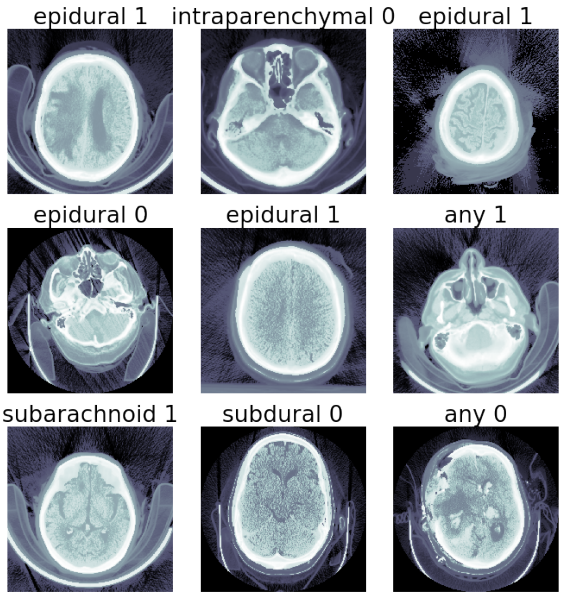


Figura B.30: Exemplos da base de dados, o título da imagem apresenta o tipo e um número (1 se possui hemorragia, 0 se não possui)

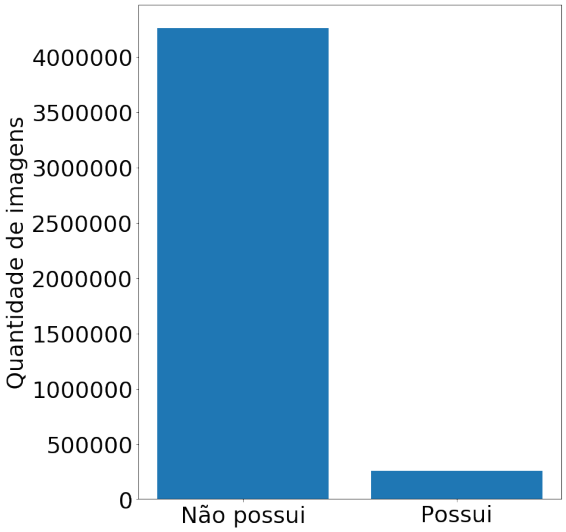


Figura B.31: Distribuição da base de dados

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

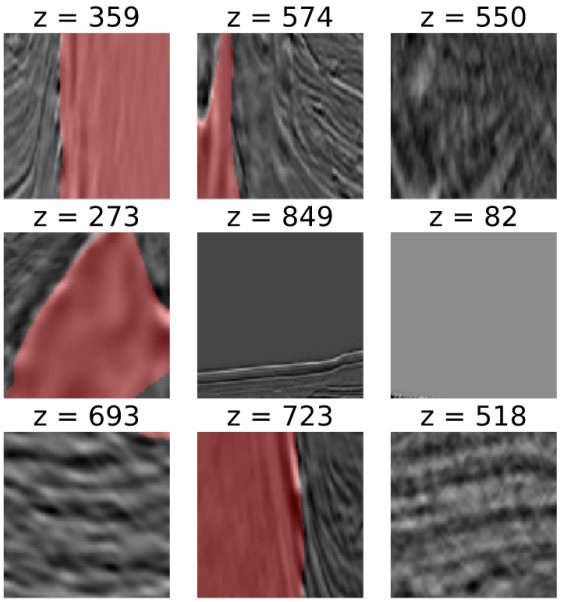


Figura B.32: Exemplos da base de dados com a máscara em vermelho, o título da imagem indica a profundidade da imagem

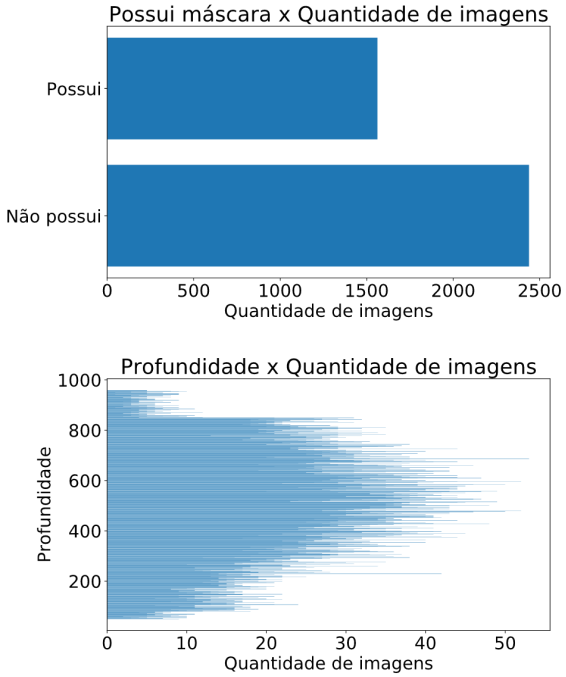


Figura B.33: Distribuição da base de dados

B.17.2 Solução

A solução[20] selecionada, durante a fase de pré-processamento, apenas normalizou e redimensionou as imagens para as dimensões 128x128 pixeis.

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 *oversampling* (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

[20] https://www.kaggle.com/meaninglesslives/getting-0-87-on-private-lb-using-kaggle-kernel

U-Net [31]. 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 pixeis 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[21] selecionado, a etapa de pré-processamento parece consistir de um redimensionamento de todas as imagens para as dimensões 224x224x3 pixeis, normalização e inversão horizontal (com 50% de chance).

[21] https://www.kaggle.com/khornlund/fork-of-sever-ensemble-3

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

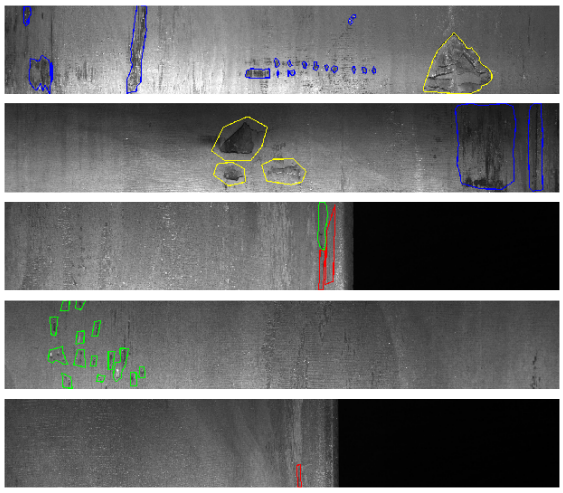


Figura B.34: Exemplos da base de dados com os defeitos circulados, seguindo o mapeamento de cor: 1 - Verde, 2 - Vermelho, 3 - Azul, 4 - Amarelo

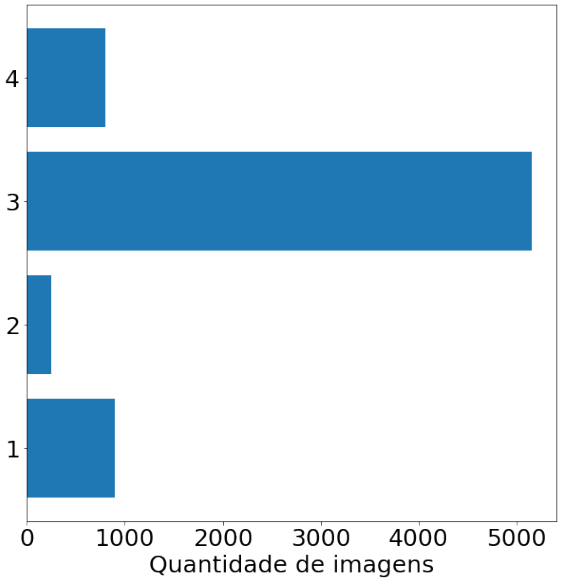


Figura B.35: Distribuição da base de dados

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-treinandos 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 pixeis, 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

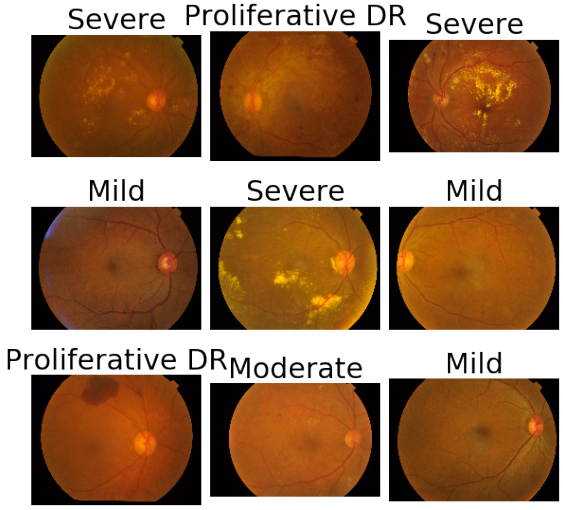


Figura B.36: Exemplos da base de dados

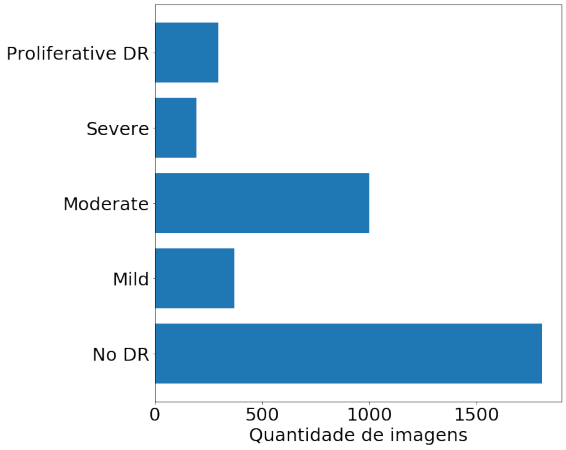


Figura B.37: Distribuição da base de dados

B.19.2 Solução

A solução[22] 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, constraste e rotação de até 360º.

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

[22] 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 62ª 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

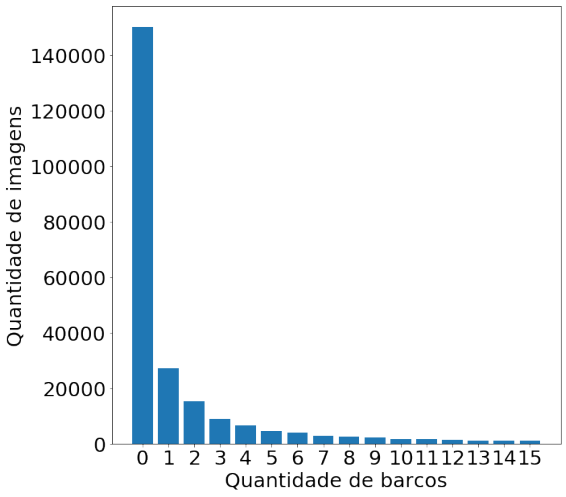


Figura B.39: Distribuição da base de dados

B.20.2 Solução

A solução[23] selecionada é uma combinação de outras duas soluções[24] [25]. 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 20º), rotações diedrais (múltiplas de 90º) 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*.

[23] https://www.kaggle.com/ashishpatel26/unet34-submission
[24] https://www.kaggle.com/iafoss/fine-tuning-resnet34-on-ship-detection/notebook
[25] 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.

Tabela B.19: Imagens do Ultrasound Nerve Segmentation

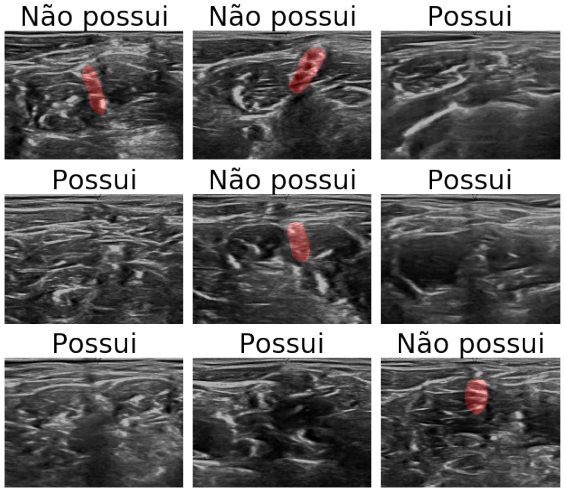


Figura B.40: Exemplos da base de dados

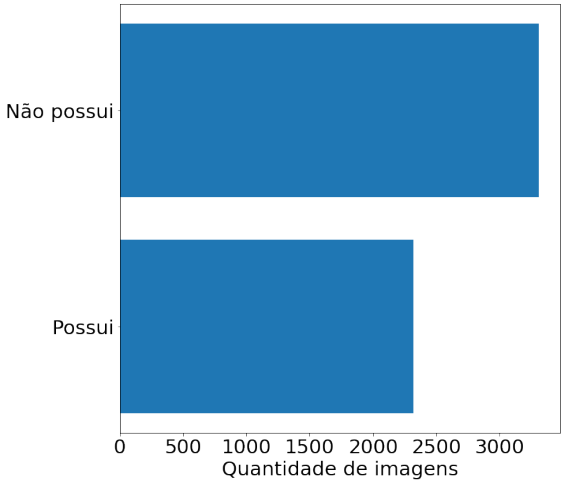


Figura B.41: Distribuição da base de dados

B.21.2 Solução

Na solução[26] selecionada, a etapa de pré-processamento consiste basicamente em redimensionar todas as imagens para as dimensões 112x80x1 pixeis 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, MaxPooling2D*. 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*.

[26] 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

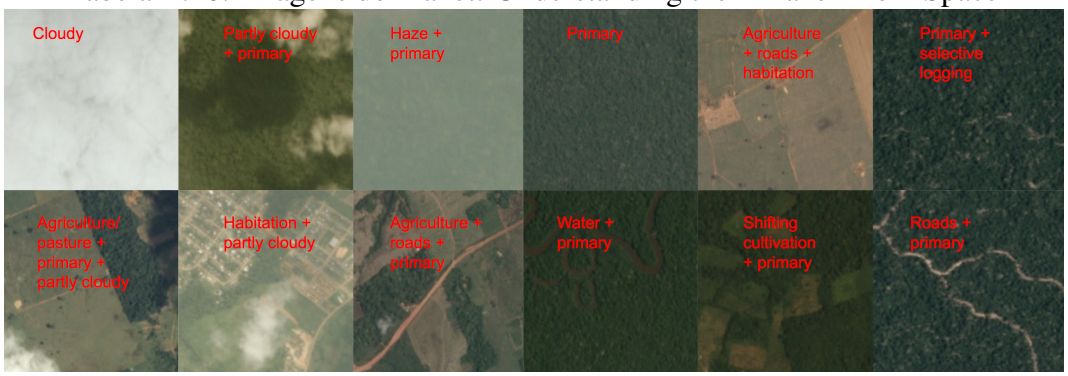


Figura B.42: Exemplos da base de dados

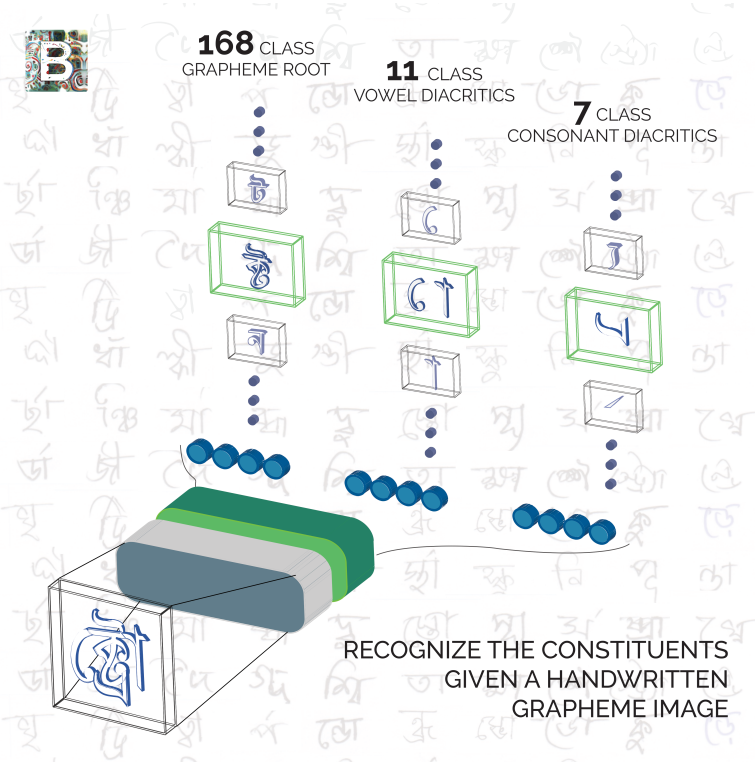
B.22.2 Solução

A etapa de pré-processamento da solução[27] selecionada consiste em redimensionar todas as imagens para as dimensões 128x128 pixeis, 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

[27] https://www.kaggle.com/liuyd2018/planet-multi-label-image-classification



scheme.png

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 d 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 5ª 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 *hierachical 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

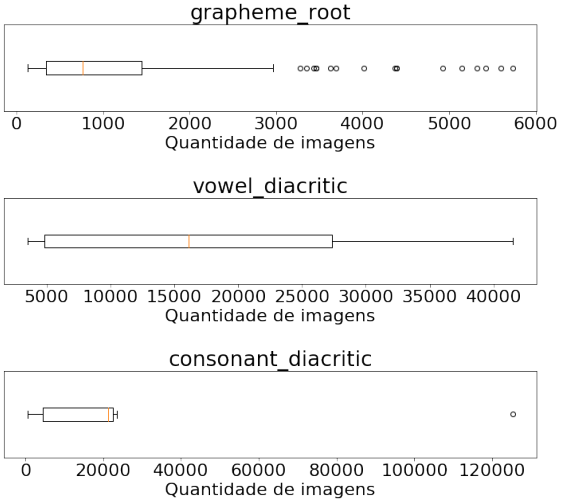


Figura B.45: Distribuição da base de dados

B.23.2 Solução

A solução[28] 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

[28] 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[30] selecionada, durante o pré-processamento, apenas normalizou e redimensionou todas as imagens para as dimensões 256x256x3 pixeis.

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

[29] https://www.kaggle.com/msafi04/pneumothoraxseg-visualization
[30] https://www.kaggle.com/meaninglesslives/nested-unet-with-efficientnet-encoder

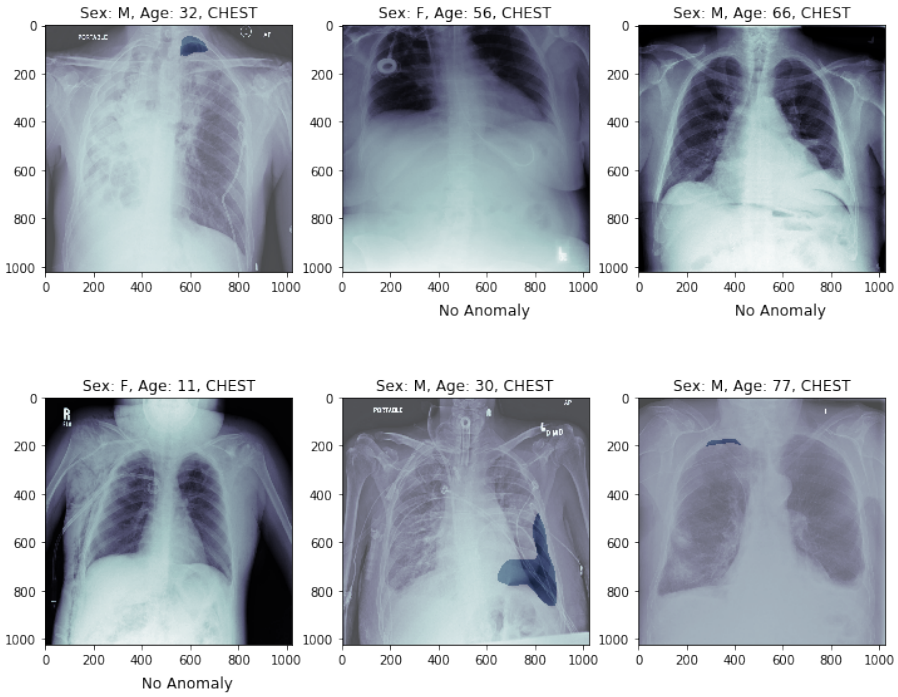


Tabela B.22: Exemplos da base de dados[29].

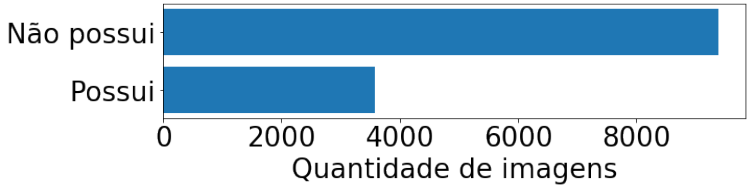


Figura B.46: Distribuição da base de dados

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

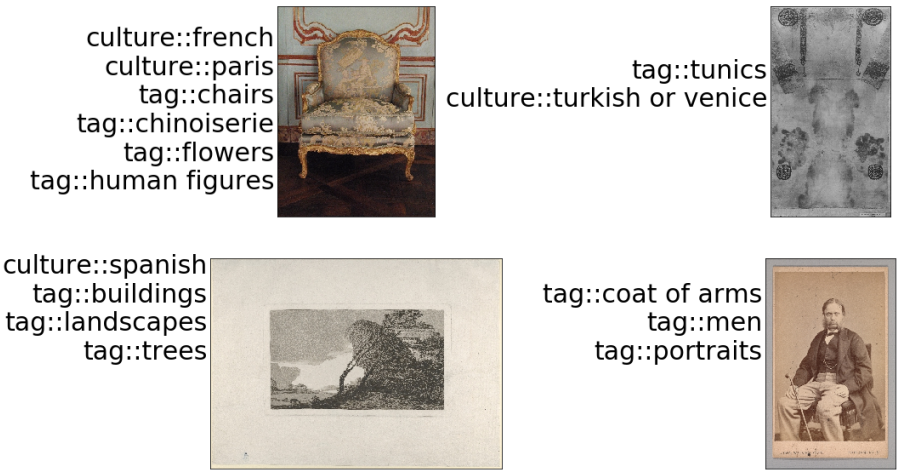


Figura B.47: Exemplos da base de dados

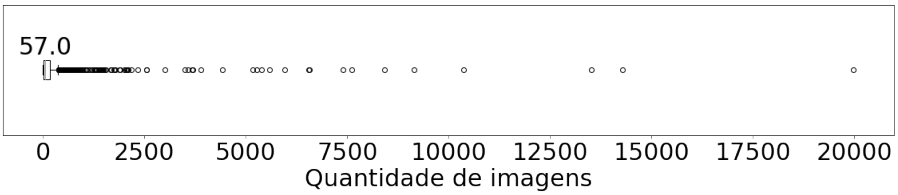


Figura B.48: Distribuição da base de dados

B.25.2 Solução

A etapa de pré-processamento da solução[31] selecionada consiste em normalizar as imagens e recortá-las em 5 janelas de dimensões 336x336 pixeis.

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

[31] 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 [32]. 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[32] 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

[32] https://www.kaggle.com/mattemilio/smile-best-who-smile-last

Tabela B.24: Imagens do Northeastern SMILE Lab - Recognizing Faces in the Wild

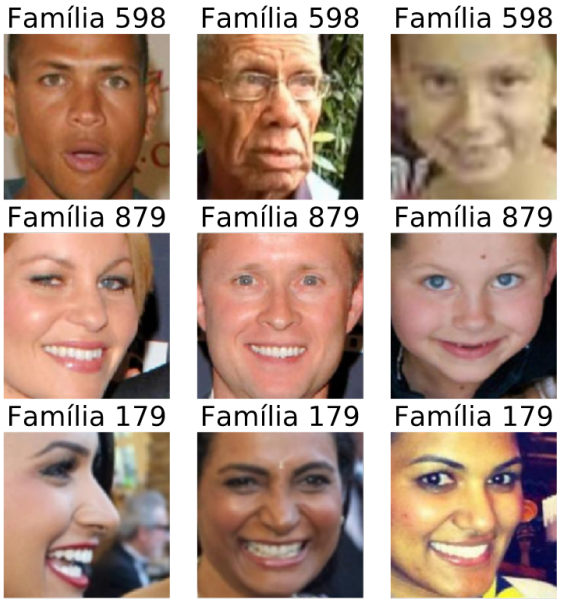


Figura B.49: Exemplos da base de dados

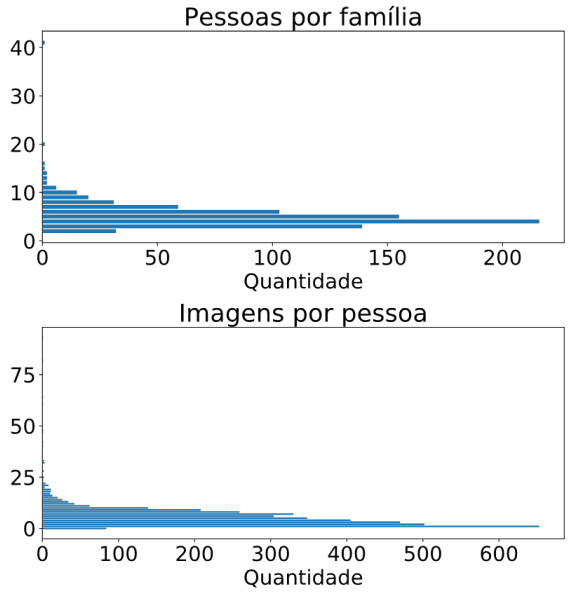


Figura B.50: Distribuição da base de dados

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.