BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

RECONHECIMENTO DE ESPÉCIES DE PEIXES UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

ALEXANDRE DA SILVA

Rio Verde, GO

2021

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO -
CAMPUS RIO VERDE
BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

RECONHECIMENTO DE ESPÉCIES DE PEIXES UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

ALEXANDRE DA SILVA

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia Goiano - Campus Rio Verde, como requisito parcial para a obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. André Da Cunha Ribeiro
Coorientador: Prof. Dr. Adriano Carvalho Costa
Instituto Federal Goiano

Rio Verde, GO

Julho, 2021

Sistema desenvolvido pelo ICMC/USP
Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas - Instituto Federal Goiano

da Silva, Alexandre
dD229r Reconhecimento de Espécies de Peixes Utilizando Redes Neurais Convolucionais / Alexandre da Silva; orientador André da Cunha Ribeiro; co-orientador Adriano Carvalho Costa. -- Rio Verde, 2021.
58 p.

TCC (Graduação em Ciência da Computação) -- Instituto Federal Goiano, Campus Rio Verde, 2021.

1. Redes Neurais Convolucionais. 2. Visão Computacional. 3. Aprendizado de Máquina. 4. Classificação de Espécies de Peixes. I. Ribeiro, André da Cunha, orient. II. Costa, Adriano Carvalho, co-orient. III. Título.

Responsável: Johnathan Pereira Alves Diniz - Bibliotecário-Documentalista CRB-1 n°2376

INSTITUTO FEDERAL
Goiano

Repositório Institucional do IF Goiano - RIIF Goiano
Sistema Integrado de Bibliotecas

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR PRODUÇÕES TÉCNICOCIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano, a disponibilizar gratuitamente o documento no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

Identificação da Produção Técnico-Científica

[ ] Tese	[ ] Artigo Científico
[ ] Dissertação	[ ] Capítulo de Livro
[ ] Monografia – Especialização	[ ] Livro
[X] TCC - Graduação	[ ] Trabalho Apresentado em Evento

[ ] Produto Técnico e Educacional - Tipo: _____

Nome Completo do Autor: Alexandre da Silva
Matrícula: 2017102201910323
Título do Trabalho: Reconhecimento de Espécies de Peixes Utilizando Redes Neurais Convolucionais

Restrições de Acesso ao Documento

Documento confidencial: [X] Não [ ] Sim, justifique: _____

_____

Informe a data que poderá ser disponibilizado no RIIF Goiano: 02/08/2021
O documento está sujeito a registro de patente? [ ] Sim [X] Não
O documento pode vir a ser publicado como livro? [ ] Sim [X] Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

1. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
2. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
3. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Rio Verde - GO, 02/08/2021.
Local Data

_____

Assinatura do Autor e/ou Detentor dos Direitos Autorais

Ciente e de acordo:

_____

Assinatura do(a) orientador(a)

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

Ata nº 19/2021 - GEPTNM-RV/DE-RV/CMPRV/IFGOIANO

ATA DE DEFESA DE TRABALHO DE CURSO

Ao(s) 29 dia(s) do mês de junho de 2021, às 14 horas e 30 minutos, reuniu-se a banca examinadora composta pelos docentes: André da Cunha Ribeiro (orientador), Adriano Carvalho Costa (membro), Heyde Francielle do Carmo França (membro), para examinar o Trabalho de Curso intitulado “RECONHECIMENTO DE ESPÉCIES DE PEIXES UTILIZANDO REDES NEURAIS CONVOLUCIONAIS” do(a) estudante ALEXANDRE DA SILVA, Matrícula nº 2017102201910323 do Curso de Bacharel em Ciência da Computação do IF Goiano – Campus Rio Verde. A palavra foi concedida ao(a) estudante para a apresentação oral do TC, houve arguição do(a) candidato pelos membros da banca examinadora. Após tal etapa, a banca examinadora decidiu pela APROVAÇÃO do(a) estudante. Ao final da sessão pública de defesa foi lavrada a presente ata que segue assinada pelos membros da Banca Examinadora.

(Assinado Eletronicamente)

André da Cunha Ribeiro

Orientador(a)

(Assinado Eletronicamente)

Adriano Carvalho Costa

Membro

(Assinado Eletronicamente)

Heyde Francielle do Carmo França

Membro

Observação:

( ) O(a) estudante não compareceu à defesa do TC.

Documento assinado eletronicamente por:

- **Adriano Carvalho Costa, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 29/06/2021 16:34:40.
- **Heyde Francielle do Carmo Franca, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 29/06/2021 16:08:58.
- **Andre da Cunha Ribeiro, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 29/06/2021 16:07:18.

Este documento foi emitido pelo SUAP em 29/06/2021. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 285490
Código de Autenticação: c3ed4fffc5

INSTITUTO FEDERAL GOIANO
Campus Rio Verde
Rodovia Sul Goiana, Km 01, Zona Rural, None, RIO VERDE / GO, CEP 75901-970
(64) 3620-5600

ALEXANDRE DA SILVA

RECONHECIMENTO DE ESPÉCIES DE PEIXES UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Trabalho de curso DEFENDIDO E APROVADO em 29 de junho de 2021, pela Banca Examinadora constituída pelos membros:

Dr. Adriano Carvalho Costa
Instituto Federal Goiano

Dr. Heyde Francielle do Carmo França
Instituto Federal Goiano

Prof. Dr. André da Cunha Ribeiro
Orientador

Rio Verde, GO

2021

Dedico esse trabalho a minha família, que me deu todo o suporte e apoio para concluir essa longa jornada.

AGRADECIMENTOS

Em primeiro lugar, aos meus pais, que dedicaram suas vidas para me cuidar e moldar quem sou. Por sempre me apoiar e me incentivar nas minhas escolhas.

Aos professores, pelos seus ensinamentos. Em especial, ao professor André Da Cunha Ribeiro, por ter sido meu orientador com dedicação e amizade. E ao professor Adriano Carvalho Costa, por ter sido meu coorientador e sempre se disponibilizado a ajudar.

À empresa Alevinos Rio Verde, pela disponibilização do espaço e dos peixes, para que pudéssemos tirar as fotos para a base de dados, que foi de grande utilidade para a elaboração deste trabalho.

RESUMO

SILVA, Alexandre. **Reconhecimento de Espécies de Peixes Utilizando Redes Neurais Convolucionais.** Junho, 2021. 46 f. Mografia – (Curso de Bacharel em Ciência da Computação), Instituto Federal de Educação, Ciência e Tecnologia Goiano - Campus Rio Verde. Rio Verde, GO, Junho, 2021.

Pelo crescente avanço, de técnicas de produção de peixes e estudo da vida aquática por meio de câmeras aquáticas, um classificador eficiente de espécies de peixes, tem sido muito demandado. Este trabalho apresenta, o desenvolvimento de um classificador baseado em Redes Neurais Convolucionais, para classificar as principais espécies de peixes de produção nacional. A técnica de transferência de aprendizado é utilizada para diminuir tempo de treinamento e dados necessários. Quatro modelos do estado da arte foram experimentados, no qual, o modelo Xception foi selecionado para ser utilizado como parte de nosso modelo. Um sistema é desenvolvido para disponibilizar o uso do classificador, para que possa ser implementado em dispositivos e sistemas inteligentes. Uma base de dados foi desenvolvida, para suprir a falta de imagens disponíveis e é proposto um algoritmo baseado em oversampling para resolver problemas com o desbalanceamento e aumentar a base de dados, de acordo com critérios estabelecidos. Foi obtido a acurácia de 99,54% ao se classificar 11 espécies de peixes alevinos com o modelo desenvolvido.

Palavras-chave: Redes Neurais Convolucionais. Visão Computacional. Aprendizado de Máquina. Classificação de espécies de peixes.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

A Organização das Nações Unidas (ONU) considera a pesca e aquicultura como atividades estratégicas para a segurança alimentar sustentável do planeta, pela sua capacidade de fornecer alimento proteico de qualidade e de gerar empregos (FAO, 2020).

A piscicultura, que é um ramo da aquicultura, é uma importante atividade no comércio brasileiro (ANDRADE; YASUI, 2003), que segundo a Associação Brasileira da Piscicultura, cresceu 38,7% de 2014 a 2020, produzindo cerca de 802.930 toneladas com receita de 8 bilhões em 2020. Diante desse crescimento acelerado, novas tecnologias para auxílio no manejo são necessárias, para otimizar e automatizar processos da cadeia de produção (FERREIRA et al., 2012).

Dentre as práticas de manejo da piscicultura, tem se a classificação ou separação de espécies de peixes no final da cadeia de produção, manipulação de peixes para biometria e entre outros. A realização desse manejo manual pode causar stress no peixe, que por sua vez libera cortisol provocando a depressão do sistema imunológico, deixando o peixe suscetível a doenças (DINIZ; HONORATO, 2012).

Câmeras aquáticas tem sido usada por biologistas para estudar a vida marinha, pois o estudo por monitoramento subaquático é importante para estimar a existência de espécie e quantidade de peixes, além de ajudar a entender o comportamento e iterações das espécies presentes na vida marinha (JOLY et al., 2015). Porém, a análise de vídeo manualmente é algo muito trabalhoso, além de ser necessário um profissional da área (QIU et al., 2018).

Sistemas capazes de classificar espécies de peixes tem papel importante na oceanografia, nas ciências aquáticas e piscicultura em sistemas de automação para auxiliar nas práticas de manejo.

Algumas características dos vídeos subaquáticos tornam a classificação uma tarefa complicada, como: Baixa qualidade da água, péssima iluminação e as vezes baixa resolução de vídeos para se obter grandes quantidades de material, tornando a classificação de peixes uma pesquisa desafiadora (QIN et al., 2016).

Técnicas para classificação, podem ser divididas em três áreas (SHAFAIT et al., 2016):

- Reconhecimento de peixe morto. ex: Uso na indústria.
- Reconhecimento de peixe vivo em ambiente artificiais. ex: Manejo na piscicultura.
- Reconhecimento de peixe vivo em ambientes naturais. ex: Estudo da vida aquática.

A classificação de espécies de peixes em ambientes artificiais, tem algumas outras aplicações práticas, como a classificação de espécies em aquários para entretenimento feito por Khalifa, Taha e Hassanien (2018).

O desenvolvimento de um sistema de reconhecimento eficiente para classificação de espécies de peixes, que pode ser aplicado em diversos cenários, tem uma grande demanda na atualidade (JOLY et al., 2015). No entanto, o seu desenvolvimento é complexo, pelo fato de ter que lidar com situações onde a qualidade da imagem é precária. Imagens de peixes de baixo da água geralmente são borradas e tem uma baixa iluminação e visualização dos peixes, além de ser limitado a quantidade de imagens catalogadas disponíveis.

A tarefa de classificação das espécies, se torna ainda mais complicada pela biodiversidade e também pelos fundos complexos nas imagens (ZHENG et al., 2018). Há também dificuldades de se diferenciar umas espécies das outras, pela similaridade de cor e forma (TAMOU et al., 2018).

Tamou et al. (2018), Deep e Dash (2019) demonstraram sistemas de reconhecimento capazes de identificar espécies de peixes em vídeos subaquático utilizando Redes Neurais Convolucionais (RNC) como parte principal do sistema. Foi reforçado por Tamou et al. (2018) que a análise manual dos vídeos gerados pelo monitoramento é um trabalho inviável pela quantidade de material, demandando tempo e um grande esforço, além de poder ocorrer erros humanos. Mostrando a necessidade de automatizar tal processo.

Neste trabalho, será proposto um sistema de reconhecimento de espécies de peixes, das principais espécies utilizadas na piscicultura, sendo elas Pacu, Tambaqui, Tambatinga, Carpa Capim, Carpas Coloridas, Lambari, Lambari Rosa, Pintado Judiará, Jundiá Rosa, Tilápia e Cachara Pura com a utilização de RNC, a fim de classificar com alto nível de precisão peixes fora ou dentro da água, em habitat artificial. O sistema deverá ser capaz de abstrair características gerais e reconhecê-lo em qualquer fundo que seja diferente de sua cor. Para isto, é necessária uma base de dados robusta que consiga trazer todas as abstrações do problema.

A quantidade de imagens disponíveis de peixes catalogadas é pequena e para realizar o treinamento e testes de modelos baseados em CNN é necessária uma grande quantidade de imagens. Dessa forma, o modelo consegue abstrair mais características importantes do peixe.

Para alcançar tal objetivo, buscamos levantar uma base de dados que esteja incluído todas as situações descritas, coletamos diversas bases de dados disponíveis na internet e montamos o nosso próprio conjunto de dados com parceria da empresa Alevinos Rio Verde e Núcleo de Ensino, Pesquisa e Extensão em Aquacultura (NEPEAQUA) do Instituto Federal Goiano - Campus Rio Verde.

2 REVISÃO DE LITERATURA

2.1 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é um modelo matemático não-linear usado para encontrar relacionamentos complexos entre a entrada e a saída. A RNA é formada por unidades que são denominadas neurônios artificiais (NA), unidades de processamento ou processadores que realizam uma soma ponderada de suas entradas relacionadas a um peso (SCHMIDHUBER, 2015). O modelo matemático dos neurônios artificiais teve seu surgimento inspirado nos neurônios biológicos do nosso sistema nervoso e proposto por McCulloch e Walter Pitts em 1943 (MCCULLOCH; PITTS, 1943).

Cada neurônio biológico tem como partes principais os dendritos, corpo celular e o axônio, veja a figura 1. Os impulsos nervosos enviados de outros neurônios chegam pelos dendritos, o corpo celular processa as informações e se a soma dos mesmos superar um limite, o neurônio é excitado e propaga um impulso nervoso pelo axônio que chegará nos dendritos de outros neurônios (MCCULLOCH; PITTS, 1943). O sistema nervoso é uma rede formada por bilhões desses neurônios trocando informações entre si.

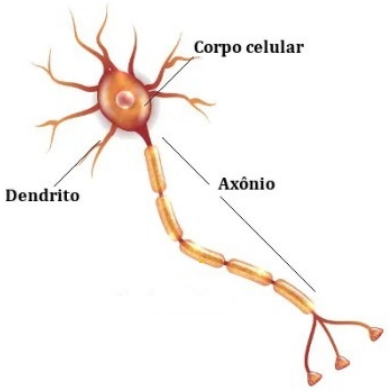


Figura 1 – Partes principais de um neurônio biológico. Fonte: Site Brasil Escola

A saída do neurônio de uma RNA é definida pelas fórmulas 1 e 2, onde y_j é uma entrada do neurônio, w_{ij} é o peso sináptico relacionado a entrada, $\boldsymbol{\beta}$ é o bias que tem como função aumentar ou diminuir a entrada líquida e $\varnothing$ é a uma função de ativação (veja sessão 2.1.1). A variável j é um neurônio da camada atual, i um neurônio da camada anterior e k um neurônio da camada posterior.

Podemos relacionar essas duas fórmulas com o neurônio biológico, com as entradas y_j sendo os dendritos, o somatório e $\varnothing$ exercendo a função do corpo celular e y o estimulo nervoso que é transmitido pelo axônio. O $\varnothing$ define se o sinal será propagado ou não.

$$z_i = \sum_{j=1}^{n} (y_j \times w_{ij}) + \boldsymbol{\beta} \tag{1}$$

$$y_i = \varnothing(z_i) \tag{2}$$

Rosenblatt (1958) propôs o perceptron, que são NAs organizados em camada, um perceptron simples é capaz apenas de resolver problemas linearmente separáveis, por ter apenas camada de entrada e saída. A rede não possui uma representação interna, pois os padrões de entradas fornecidos são diretamente mapeados em um conjunto de saídas.

Para resolver problemas que não são lineares, foi introduzido a topologia de RNA como conhecemos hoje, chamada de perceptrons de múltiplas camadas (PMC). Os PMC são camadas alinhadas entre a entrada e a saída, onde os neurônios da camada posterior estão ligados aos da camada atual, formando uma rede em que cada ligação entre dois NA é chamado de conexão sináptica. O processamento nessas camadas é importante para a rede entender os padrões complexos da camada de saída (SURYANARAYANA et al., 2008).

Tais camadas são separadas em 3 tipos, sendo elas: A camada de entrada, que irá receber os valores de entrada. A(s) camada(s) escondidas(s), que são as camadas intermediarias da rede. Por fim, a camada de saída, onde se obtém o resultado. As camadas escondidas são assim chamadas em razão de elas não serem observadas diretamente (BISHOP et al., 1995).

A representação de uma rede neural simples é demonstrada na figura 2, essa rede tem uma configuração de topologia sendo 2-3-3-1, com cada valor representando o número de neurônios em cada camada de acordo com a sequência.

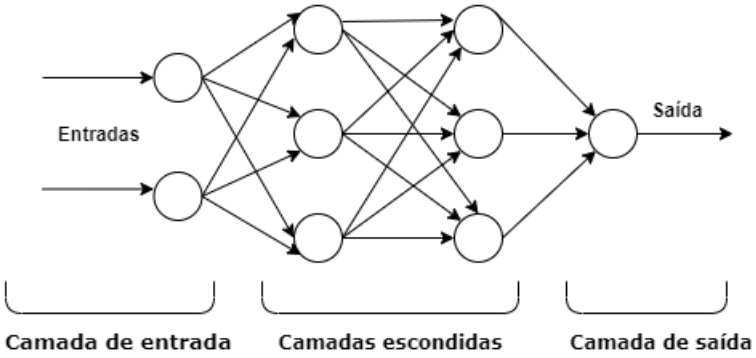


Figura 2 – Rede Neural 2-3-3-1. Fonte: Autor.

Para o problema de classificação, essa rede em especial tem capacidade de resolver apenas problemas binários, pelo o fato de ter um NA na camada de saída, podendo assumir apenas dois estados, verdadeiro ou falso, por exemplo. Para a classificação multiclasse, é preciso adicionar a quantidade de NA compatível com o número de classes do problema, no nosso caso, a quantidade de espécies de peixes que desejamos classificar e usar uma função de ativação adequada na camada de saída.

Um modelo é dito ser de aprendizado profundo, quando têm várias camadas intermediárias, pois se tem uma grande quantidade de parâmetros para serem treinados.

2.1.1 Função de Ativação

Funções de ativação (FA), são usadas para calcular o peso da soma e biases nas redes neurais, e para controlar suas saídas. Podendo ser lineares ou não, dependendo da função que ela representa (NWANKPA et al., 2018).

Os NAs produzem resultados lineares por natureza com a equação 1, sendo necessário uma função para transformar as saídas em não-lineares. Essa função é essencial para a RNA ter capacidade representativa e assim aprender padrões nos dados.

A posição de uma função de ativação na RNA depende do seu objetivo, sendo duas possíveis posições. A primeira é após as camadas ocultas, para converter as saídas em não lineares, e assim propagar os sinais através das conexões sinápticas. A segunda posição é na saída, para realizar as predições (NWANKPA et al., 2018).

2.1.1.1 ReLU

Nair e Hinton (2010) propôs o uso da Unidade Linear Retificada (ReLU, na sigla em inglês), veja a equação 3. A ReLU se mostrou ter uma melhor performance e generalização. A função realiza um limite, onde números menores que 0 são transformados em 0, caso contrário é o próprio valor.

$$\varnothing(z) = \max\{0,z\} = \begin{cases} z, \text{se } z > 0 \\ 0, \text{se } z \leqslant 0 \end{cases} \tag{3}$$

Assim, optamos por utilizar a função ReLU, pois também o tempo de treinamento utilizando ela é muito menor em comparação com outras nas redes profundas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; LECUN; BENGIO; HINTON, 2015).

Uma limitação importante do uso dessa função, é que ela acaba causando sobreajuste facilmente comparado com outras funções, mas esse problema é em parte resolvido introduzindo a camada de eliminação, veja seção 2.1.5.

A função de ativação ReLU é principalmente usada nas camadas ocultas, na última camada que é a de saída, é utilizado uma função capaz de mapear a saída em diferentes classes, sendo ela a softmax.

2.1.1.2 Softmax

A função softmax é usada em modelos que resolvem problemas de múltiplas classes. Ela calcula a distribuição de probabilidade das saídas do modelo, transformando os valores entre 0 e 1, onde a soma de todas as saídas resulta em 1 e o maior valor é a resposta esperada (NWANKPA et al., 2018).

A função é descrita a seguir na equação 4. Onde, i representa o índice do NA da saída sendo calculado, j o índice de todos os NA da saída e z o vetor dos neurônios da

saída.

$$\varnothing(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{4}$$

2.1.2 Treinamento

Há dois métodos de treinamento de uma rede neural, o treinamento supervisionado (TS) e o treinamento não-supervisionado (TNS). No TS há um supervisor que fornece pares de entrada e saída esperada, sendo possível calcular o erro da resposta atual em relação a desejada, com isso os pesos são modificados para minimizar o erro total (SCHMIDHUBER, 2015). Já no TNS, não há supervisor e a RNA deve ser capaz de achar correlação ou redundância nos padrões de entrada.

Neste trabalho utilizaremos o algoritmo chamado retropropagação ou pelo termo mais conhecido em inglês back-propagation, que é um algoritmo de TS.

2.1.2.1 Algoritmo de retropropagação

O problema de se treinar uma RNA, pode ser visto como um problema de otimização, que visa minimizar ou maximizar um escalar que é calculado por uma função objetiva com respeito aos seus parâmetros. A otimização é realizada alterando os parâmetros com o objetivo de minimizar ou maximizar a função objetiva. A função objetiva, também conhecida como função de perda ou função de custo, é uma função que mede o quão ruim está o modelo.

Rumelhart, Hinton e Williams (1986) propôs o algoritmo de retropropagação, que aplica o método da descida do gradiente estocástico e de maneira iterativa vai ajustando os pesos das conexões para otimizar a função objetiva. Com a utilização desse algoritmo, temos 2 passos importantes no treinamento de uma RNA.

O primeiro passo, é o passar adiante ou feed-forward em inglês, que consiste em propagar as entradas através da rede, aplicando as fórmulas 1 e 2 para todos os neurônios em sequência da camada de entrada até a camada de saída.

O segundo passo, é o de retropropagação, que consiste em ajustar os pesos da camada de saída até a camada de entrada. Os pesos são ajustados utilizando a equação 5, também conhecida como regra delta. Com η sendo a taxa de aprendizado, C a função de custo e t é o passo do tempo. $\frac{\partial C}{\partial w_t}$ é o calculo do gradiente da função de custo em relação aos parâmetros w_t.

$$\triangle w_{t+1} = w_t - \eta \frac{\partial C}{\partial w_t} \tag{5}$$

O algoritmo de retropropagação é apresentado no algoritmo 1.

Há alguns desafios ao se trabalhar com a descida do gradiente estocástico, como decidir qual valor escolher para a taxa de aprendizado. Um valor muito pequeno acarreta

Algoritmo 1: Algoritmo de retropropagação

Entrada: Taxa de aprendizado η

início
  Inciar parâmetros aleatórios;
  enquanto *não atingir critério de parada* **faça**
    Pegar amostras da base de treinamento e apresentar a rede;
    Calcular estimativa do gradiente: $g \leftarrow \frac{\partial C}{\partial w_t}$;
    Ajustar os parâmetros: $\triangle w_{t+1} = w_t - \eta g$;
  fim
fim

em uma convergência muito lenta e um valor grande pode fazer com que a função de perda fique flutuando em volta do mínimo local.

O mínimo local é um ponto, onde se obtém o menor valor calculado pela função de perda, em uma região próxima a ele (GORI; TESI, 1992), veja figura 3. Na figura também há a presença do mínimo global, que é o menor valor possível calculado pela função objetiva. Quando o modelo fica preso nessas regiões de mínimos locais, ele não converge de maneira satisfatória.

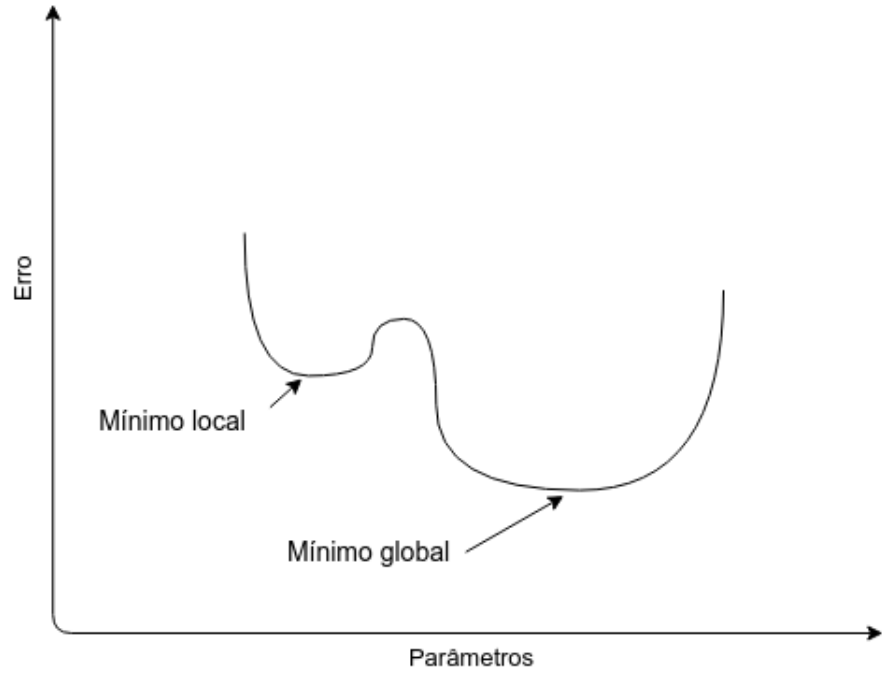


Figura 3 – Exemplo de mínimo local. Fonte: Autor.

Outro problema com a descida do gradiente estocástico, é que a mesma taxa de aprendizado é aplicado para todos os pesos, isso é um problema quando o conjunto de dados é esparso, pois se uma característica é rara, será desejado utilizar um valor maior, caso contrário, um valor menor se ela for frequente (RUDER, 2016).

2.1.2.2 Método Adam

Ruder (2016) na sua revisão de métodos de otimização, recomenda a utilização do método Estimativa de Momento Adaptativo (Adam, na sigla em inglês) proposto por Kingma e Ba (2014). O Adam, que é uma variação do método descida do gradiente estocástico, calcula a taxa de aprendizado adaptativa para cada peso e tem diversas vantagens sobre outros métodos, algumas das vantagens destacadas pelos autores são: Requer pouca memória, tem um custo computacional pequeno e adequado para problemas que são grandes em termos de dados e/ou parâmetros. Dito isso, iremos utilizar o Adam como método de otimização em nossos modelos.

No Adam, para se calcular a atualização de um peso, é necessário calcular a estimativa do primeiro momento m_t e a estimativa do segundo momento v_t. O cálculo desses dois momentos são demonstrados na equação 6 e 7. As duas equações dependem do calculo da derivada parcial, descrita na equação 8 e das variáveis β_1 e β_2, que os autores recomendam que sejam 0,9 e 0,999 respectivamente.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{6}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{7}$$

$$g_t = \varnothing'(z_{t-1}) \tag{8}$$

Os valores m_t e v_t são vetores iniciados com zeros, por conta disso, os autores notaram que os valores são tendenciosamente levados a ser zero. Para resolver esse problema, introduziram as equações 9 e 10 para corrigir a tendência.

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{9}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{10}$$

Finalmente, a equação 11 é usada para atualizar os pesos, onde ϵ é recomendado ser 10^{-8}.

$$\triangle w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v_t} + \epsilon}} \hat{m_t} \tag{11}$$

2.1.2.3 Variáveis de treinamento

Há algumas variáveis importantes para escolher no treinamento, como o tamanho do lote, a quantidade de épocas e os passos por época.

O tamanho do lote é a quantidade de amostras que serão consideras antes de realizar os ajustes dos pesos, pois a estimativa do erro do gradiente irá ser calculado

considerando essa quantidade de amostras. Esse valor geralmente é estimado levando em conta a quantidade de recursos disponíveis, pois a quantidade de imagens do lote será carregada na memória.

As épocas são as quantidades de iterações sobre a base de dados de treinamento para se finalizar o treinamento, ou seja, uma época é uma iteração na base de dados. Não tem nenhum critério a ser seguido para se estimar a quantidade de épocas previamente, no entanto, deve ter em mente que se um valor muito pequeno for considerado o modelo provavelmente não será convergido a tempo para próximo do mínimo local. Se o valor for muito alto o modelo pode sofrer sobreajuste por ter treinado excessivamente na base de treinamento, um método para contornar esse problema é apresentado na sessão 2.1.4.1.

Os passos por época é a quantidade de iterações do lote para se ter finalizado uma época. Geralmente é calculado sendo o número de amostras sobre o tamanho do lote, para que cada época passe por toda a base de treinamento.

2.1.3 Subajuste e Sobreajuste

Há dois principais desafios quando estamos trabalhando com aprendizado de máquina, que é o sobreajuste (Overfitting, em inglês) e subajuste (Underfitting, em inglês), que segundo Goodfellow, Bengio e Courville (2016), correspondem a dois fatores que determinam o quão bom é um algoritmo de aprendizado de máquina:

- Tornar o erro do treinamento o mais próximo de 0.
- Tornar a diferença entre o erro de treinamento e de teste pequeno.

O sobreajuste ocorre quando a rede atinge um erro pequeno na base de treinamento, mas na base de teste os resultados são bem inferiores. Na prática, isso significa que a rede não conseguiu generalizar o conhecimento, e os relacionamentos complexos que encontrou existe apenas na base de treinamento.

O subajuste é quando o modelo não conseguiu um erro pequeno no treinamento, mostrando que o modelo não é apropriado para a tarefa, por não ter a capacidade necessária. Logo, quando o modelo tem uma alta capacidade, pode ocorrer o sobreajuste, pois acabou memorizando propriedades apenas da base de treinamento. Um exemplo demonstrando tais situações são expostos na figura 4.

Determinar a capacidade de um modelo é difícil, pois a capacidade efetiva é limitada pelo algoritmo de otimização, pois o algoritmo não encontra os melhores parâmetros, mas sim os que reduzem significativamente o erro. Sendo assim, a capacidade efetiva definida pelo algoritmo, pode ser menor que a capacidade real do modelo.

Como mencionado acima, as chances de um modelo sofrer sobreajuste ou subajuste depende da capacidade do modelo, a figura 5 mostra bem como é essa relação. Observamos na figura 5, que a curva de erro do treinamento e teste tem um comportamento diferente, na primeira parte ambos tem altos valores de erro, que é a área do subajuste, de acordo que a capacidade aumenta, o erro do treinamento vai diminuindo e a lacuna entre a curva

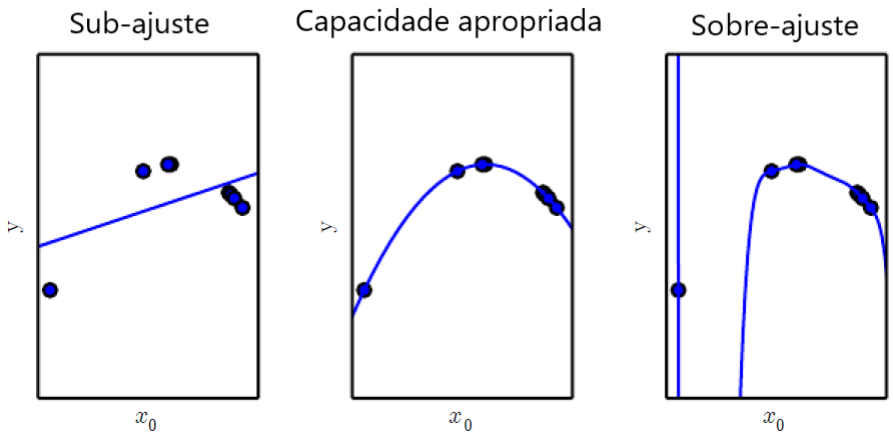


Figura 4 – Exemplo de 3 modelos que são treinados para resolver um problema, sendo que o primeiro (da esquerda), não tem a capacidade necessária, o segundo modelo tem a capacidade ideal para o problema e o terceiro tem uma capacidade muito grande, causando o sobreajuste. Fonte: Goodfellow, Bengio e Courville (2016).

da base de teste aumenta, sendo a área do sobreajuste.

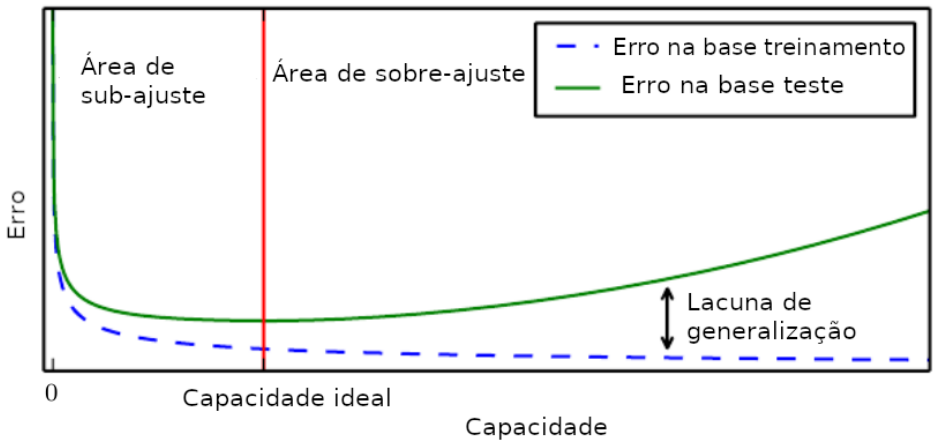


Figura 5 – Relação entre o erro e a capacidade. Fonte: Goodfellow, Bengio e Courville (2016)

2.1.4 Regularização

Regularização são métodos que tem o propósito de limitar a capacidade do modelo, restringindo o espaço de soluções para reduzir o erro de generalização, mas não o de treinamento (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.1.4.1 Parada Antecipada

Algo complicado em se definir é por quanto tempo um modelo deve ser treinado, considerando que nosso objetivo no treinamento é otimizar a função objetiva. No entanto,

só essa informação não é suficiente e podemos ter sérios problemas durante o treinamento, como o sobreajuste.

Quando o modelo tem capacidade representativa para a tarefa que será empregada e é treinado por um número muito grande de épocas, sem nenhum critério de parada, pode ocorrer o sobreajuste. Ficando limitado a base de treinamento, sem ter um poder de generalização para novas amostras do domínio da tarefa (GOODFELLOW; BENGIO; COURVILLE, 2016). Veja a sessão 2.1.3 para mais detalhes sobre o sobreajuste.

A forma mais comum de resolver esse problema é utilizar o método da parada antecipada. Esse método simples consiste em acompanhar durante o treinamento uma métrica de desempenho do modelo na base de validação de cada época treinada. A parada antecipada verifica se ela melhorou por um número n de épocas do último melhor resultado, caso não, o treinamento é interrompido e os melhores parâmetros das últimas n épocas são restaurados para o modelo.

Neste trabalho será utilizado o valor 3 para n, por conta da limitação de tempo de treinamento disponível.

2.1.5 Camada de eliminação

Redes neurais profundas tem um grande poder de aprendizado e são capazes de encontrar relacionamentos complexos entre a entrada e a saída, no entanto, diversos problemas surgem diante da grande quantidade de parâmetros que essas redes possuem. Além de serem lentas para treinar e usar, o problema com sobreajuste se torna algo complicado de se lidar, ainda mais quando se detém um número pequeno de amostras (SRIVASTAVA et al., 2014).

A camada de eliminação lida muito bem com esse problema, descartando aleatoriamente e temporariamente alguns neurônios e suas conexões durante o treinamento. Cada neurônio tem uma probabilidade p que é independente de qualquer outro, p pode ser calculado usando um conjunto de validação ou apenas ser 50%.

Tendo uma rede neural com N camadas escondidas, sendo l em {0,..., N - 1 } o índice das camadas ocultas, $\boldsymbol{z}^{(l)}$ é o vetor de entrada da camada l, $\boldsymbol{y}^{(l)}$ é o vetor de saída da camada l. $W^{(l)}$ e $\boldsymbol{b}^{(l)}$ são os pesos e bias da camada l. A operação feed-forward de uma rede neural padrão pode ser descrita na equação 12, sendo i qualquer neurônio da camada escondida e $\varnothing$ qualquer função de ativação.

$$\begin{aligned} z_i^{(l+1)} &= W_i^{(l+1)} y^{(l)} + b_i^{(l+1)}, \\ y_i^{(l+1)} &= \varnothing(z_i^{(l+1)}). \end{aligned} \tag{12}$$

Com a camada eliminação, a operação se torna.

$$\begin{aligned} r_j^{(l)} &\sim Bernoulii(p), \\ \tilde{y}^{(l)} &= r^{(l)} * y^{(l)}, \\ z^{(l+1)} &= W^{(l+1)}\tilde{y}^{(l)} + b_i^{(l+1)}, \\ y_i^{(l+1)} &= \varnothing(z_i^{(l+1)}). \end{aligned} \tag{13}$$

O $*$ representa o produto de Hadamard na equação 13, para cada camada l, $r_j^{(l)}$ é um vetor de variáveis aleatórias independentes de Bernoulli, em que cada uma tem probabilidade p de ser 1. Esse vetor é multiplicado pelo produto Hadamard com as saídas da camada, para criar as saídas refinada $\tilde{y}^{(l)}$. As saídas que restaram então são passadas para alimentar a próxima camada e o processo se repete para cada camada. Importante ressaltar que é apenas no treinamento que é feito essa eliminação com parte dos neurônios, durante o teste da rede todos os pesos são escalados como $W_{test}^{(l)} = pW^{(l)}$.

2.1.6 Regularização dos pesos

A regularização dos pesos tem o objetivo de reduzir o sobreajuste mantendo os pesos em valores muito pequenos (GOODFELLOW; BENGIO; COURVILLE, 2016). É importante ressaltar, que geralmente os pesos dos bias não são regularizados, pois as entradas são constantes.

Para controlar os valores dos pesos, é realizado uma penalização Ω no valor durante o treinamento, essa penalização é controlada pelo hiper-parâmetro α, que tem um valor entre 0 e 1. O Ω é adicionado à função objetiva J, na fórmula 14, onde θ é vetor de pesos, X vetor de entrada, C o vetor de rótulos corretos e $\tilde{J}$ a função objetiva normalizada.

$$\tilde{J}(\theta; X; C) = J(\theta; X; C) + \alpha\Omega(\theta) \tag{14}$$

Há duas abordagens principais para realizar a penalização nos pesos, que são as normas $L1$ e $L2$, a norma $L1$ realiza a regularização sobe a soma absoluta dos pesos e é mostrado na fórmula 15.

$$\Omega(\theta) = ||\omega||_1 = \sum_j |w_j| \tag{15}$$

A norma $L2$ é calculada como a raiz quadrada da soma dos valores do vetor ao quadrado, veja fórmula 16. As duas normas podem ser usadas juntas ou separadamente.

$$\Omega(\theta) = ||\omega||_2 = \sqrt{\sum_j |w_j|^2} \tag{16}$$

2.2 Redes Neurais Convolucionais

Na área de reconhecimento de objetos em imagens, as RNCs são as que produzem melhores resultados (KRIZHEVSKY et al, 2012), sendo o estado da arte nesse campo. As RNCs são capazes de extrair características importantes das imagens (ZHENG et al, 2018). Uma RNC pode ser dividida em duas partes. A primeira é o extrator de características, onde se encontram as camadas convolucionais e agrupamento. A segunda é um classificador, que pode ser uma rede neural tradicional como mostra a figura 6, ou qualquer outro classificador como máquina de vetores de suporte. Neste trabalho iremos usar uma rede neural de múltiplas camadas.

As RNCs possuem características que as tornam invariantes com respeito às transformações nas imagens que alimentam o modelo, permitindo uma capacidade de reconhecer padrões que são deslocados, inclinados ou distorcidos nas imagens (LECUN; BENGIO et al., 1997; LECUN; BENGIO; HINTON, 2015).

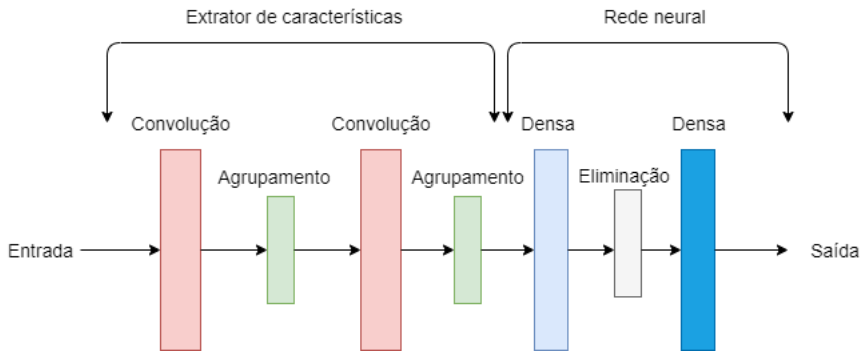


Figura 6 – Rede neural convolucional típica. Fonte: Autor.

2.2.1 Camada de Convolução

As RNCs são compostas por camadas convolucionais, a convolução é uma operação matemática que é usada para filtrar sinais das imagens. Para isso é utilizado um filtro, também conhecido como kernel, que tem uma dimensão definida por altura, largura e profundidade. Com a profundidade sendo 3, pois estamos trabalhando com imagens coloridas e elas tem 3 canais de cores.

O filtro vai deslizando por toda a imagem pulando um certo número de pixels, chamado de stride. Em cada parte do plano de entrada que o filtro sobrepõe, é realizado o produto entre o respectivo pixel sobreposto, pelo o valor do kernel. Após esse processo, tudo é somado obtendo o resultado final da localização atual (DUMOULIN; VISIN, 2016), como mostra a figura 7. Após percorrer toda a imagem realizando esse mesmo processo, o resultado final obtido é chamado de mapa de características.

A área sobreposta na entrada é comumente chamada de campo receptivo local, que extrai informações locais importantes como cantos e arestas. De modo resumido, cada

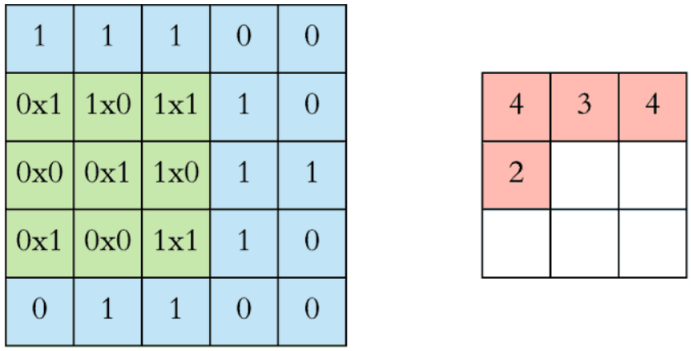


Figura 7 – Exemplo de filtro 3x3x1 com stride igual a 1. Fonte: Autor.

unidade no mapa de característica, pega sua entrada do campo receptivo local da camada anterior, como mostra a figura 8.

Todos os pesos do filtro são os mesmos para todas as unidades do mapa de características atual, utilizando o conceito de pesos compartilhados, que consiste em ter diversas conexões controladas por um mesmo parâmetro.

Uma informação que é importante na imagem pode variar de posição, utilizando essa técnica, essa informação será extraída independentemente da localização na imagem (LECUN et al., 1989). Essa técnica reduz bastante o número de parâmetros livres a serem treinados, reduzindo também a quantidade necessária de dados para treinamento (LECUN; BENGIO et al., 1997).

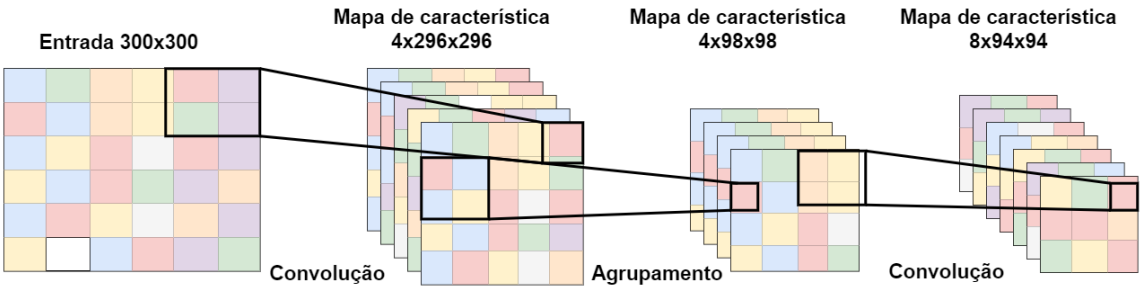


Figura 8 – Esquema de rede neural convolucional. Fonte: Autor.

Na prática irá haver um número alto de filtros por cada camada convolucional, por exemplo 32, e cada filtro irá produzir um mapa de características distinto, pois não há compartilhamento de pesos entre os filtros, apenas entre as unidades de um mapa de características. Assim, seguindo o exemplo, teríamos 32 mapas de características que serão passados adiante na rede, com cada um contento uma informação importante sobre a imagem.

No processo de convolução, o resultado obtido é um conjunto de mapa de características, cujo a dimensão espacial (altura e largura) vai diminuindo de acordo que o fluxo vai indo mais profundo na rede. Essa redução depende do tamanho do filtro, da quantidade de passos que ele se move pelo plano e do preenchimento de zeros na borda da imagem.

Com a operação de convolução o resultado sempre terá sua dimensão espacial menor que a entrada, as vezes é desejado manter o tamanho atual, até para evitar que a dimensão caia rapidamente durante o fluxo da rede. Visando esses objetivos, é utilizado a técnica de se concatenar zeros no início e no fim de algum eixo, veja figura 9.

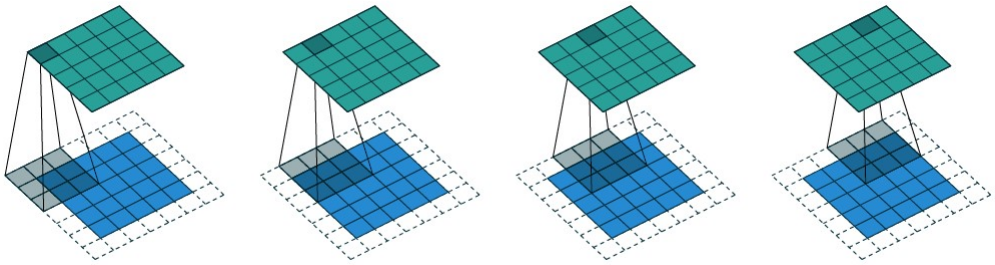


Figura 9 – Exemplo de preenchimentos com zeros na borda. Fonte: Dumoulin e Visin (2016)

A análise das propriedades entre as camadas convolucionais se torna mais simples pelo fato de elas não se interagirem através dos eixos de altura e largura, porém a relação entre essas propriedades não é simples de se inferir (DUMOULIN; VISIN, 2016).

O tamanho da saída é definido pela fórmula 17, para qualquer i k, s e $p = 0$, onde:

- i é o tamanho da imagem, sendo $i_1 = i_2 = i$.
- k tamanho do filtro, sendo $k_1 = k_2 = k$.
- s é quantidade passos, sendo os mesmos para os dois eixos $s_1 = s_2 = s$.
- p a quantidade de zeros na borda da imagem, sendo o mesmo para ambos os eixos $p_2 = p_2 = p$.

$$o = \lfloor \frac{i + 2p - k}{s} \rfloor + 1 \tag{17}$$

Pelo fato de as vezes o último passo possível não coincida para o filtro chegar no final da entrada, é necessário usar a função de piso e algumas unidades da entrada são descartadas.

2.2.2 Camada de Agrupamento

A camada de agrupamento reduz o tamanho espacial da imagem (altura e largura), compactando-a e extraindo as características mais relevantes, no intuito de diminuir os ruídos da imagem e evitar o sobreajuste (AZIZPOUR et al., 2015). O agrupamento reduz a sensibilidade da saída com distorções na imagem e deslocamento (LECUN et al., 1998).

A camada de agrupamento percorre toda a saída da camada anterior agrupando os pixel por uma janela de tamanho $L * A$, e resumindo cada um desses grupos pegando o maior valor (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), veja a figura 10. A janela desliza na imagem, da esquerda para direita, pulando um número de pixel chamado de

stride. Ao alcançar a borda direita, retorna na extremidade esquerda e desce o número de stride definido, repetindo esse processo até percorrer toda a imagem.

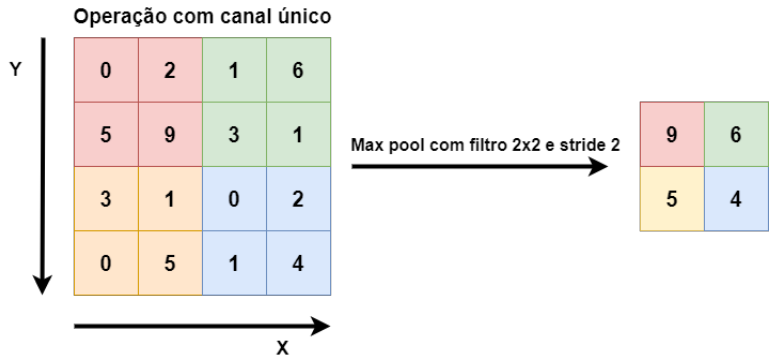


Figura 10 – Exemplo de agrupamento 2x2 com stride igual a 2. Fonte: Autor.

2.3 Transferência de Aprendizado

Treinar uma RNC do início exige uma quantidade de dados muito grande e um custo computacional enorme para se alcançar o desempenho desejado, devido a grande quantidade de parâmetros que a RNC possui. A nossa base de dados levantada (veja sessão 3.2) é insuficiente para treinar uma RNC do início. Para superar esses problemas, usaremos os pesos de modelos já treinados na base de dados ImageNet (Deng et al., 2009), como um ponto de início.

Esse método de usar parâmetros de modelos já treinados em base de dados de larga escala, é chamado de transferência de aprendizado. Usando esse método, não precisaríamos treinar do zero nossos modelos, pois os parâmetros transferidos conseguem extrair representações de nível médio da imagem (OQUAB et al., 2014).

É possível utilizar a transferência de aprendizado de duas formas (TAMOU et al., 2018):

- Extrator de características - Remover algumas das últimas camadas e utilizar o restante como extrator de características.
- Aperfeiçoamento - Treinar novamente as últimas camadas na base de dados de interesse e ajustar os parâmetros do modelo via back-propagation.

A transferência de aprendizado foi utilizado com sucesso para classificação de peixes por Siddiqui et al. (2018), Tamou et al. (2018), Santos e Gonçalves (2019), Sun et al. (2016), Qiu et al. (2018), Olsvik et al. (2019), Mathur et al. (2020), Christensen et al. (2018), Szymak e Gasiorowski (2020), demostrando a eficiência do método no domínio do nosso problema e por isso iremos utilizá-la em nosso trabalho apenas no extrator de características.

2.4 Métricas de Desempenho

Algo muito importante em se decidir são as métricas de desempenho, elas são usadas para avaliar o desempenho do modelo e guiar na sua modelagem entre várias alternativas (SUN; WONG; KAMEL, 2009; BRANCO; TORGO; RIBEIRO, 2015).

A escolha de uma métrica que não representa o desempenho do modelo de forma correta, pode nos trazer resultados finais indesejáveis. Por isso, deve-se escolher métodos apropriados para o problema que está trabalhando e o objetivo, para demonstrar o desempenho real do modelo, como por exemplo, dar mais ênfase as amostras de classes minoritárias classificadas de maneira errada.

Primeiramente, vamos nos familiarizar com a matriz de confusão binária, para termos entendimento sobre termos fundamentais, pois as métricas são feitas a partir da matriz (SOKOLOVA; JAPKOWICZ; SZPAKOWICZ, 2006). A matriz é representada na tabela 1, ela nós traz a quantidade de exemplos reconhecidos corretamente ou incorretamente para cada classe. Com vp, representando Verdadeiro Positivo e o exemplo foi classificado corretamente para classe positiva. Fp, Falso Positivo e o exemplo foi classificado incorretamente pertencente a classe positiva. Vn, Verdadeiro Negativo e o exemplo foi classificado corretamente pertencente a classe negativa. E fn, Falso Negativo, o exemplo classificado erroneamente pertencente a classe negativa. A classe positiva é a classe de interesse que se está analisando.

Tabela 1 – Matriz de confusão binária

	Predição positiva	Predição negativa
Classe positiva	vp	fn
Classe negativa	fp	vn

A métrica de desempenho mais comum usada nos problemas de classificação é a acurácia (BRANCO; TORGO; RIBEIRO, 2015), mostrada na equação 18 para classificação binária. A acurácia diz a taxa de acerto geral do modelo, dividindo as corretas predições pelo número total realizado.

$$\text{Acurácia} = \frac{vp + vn}{vp + fp + fn + vn} \tag{18}$$

Quando trabalhamos com uma base de dados muito desbalanceada, ao usar a acurácia, surge o problema em que a classe minoritária tem menos efeito na acurácia, de acordo em que cresce a diferença com a classe majoritária (SUN; WONG; KAMEL, 2009). Por exemplo, vamos considerar que temos uma base de dados com 100 exemplos, sendo 90 para a classe negativa e 10 para a classe positiva, se todos os exemplos da classe positiva fossem classificados errados e da classe positiva corretamente, teríamos uma acurácia de 90%, dando uma falsa impressão de qualidade do modelo.

No objetivo de escolher métricas para lidar com um conjunto de dados desbalanceado, deve-se considerar preferências de acordo com a distribuição dos dados (BRANCO; TORGO; RIBEIRO, 2015), considerando a classe minoritária do exemplo anterior como a classe positiva, podemos calcular a taxa de verdadeiro positivo (vp) ou revocação com a equação 19 e teríamos um resultado de 0%, indicando dificuldades do modelo aprender a classe positiva. A revocação traz a indicação dos exemplos da classe positiva que foram classificados incorretamente.

$$\text{Revocação} \rightarrow R = \frac{vp}{vp + fn} \tag{19}$$

Outra métrica de bastante interesse é a precisão, definida na equação 20, com ela podemos ter a informação da relevância dos exemplos classificados corretamente.

$$\text{Precisão} \rightarrow P = \frac{vp}{vp + fp} \tag{20}$$

Fazendo uma análise sobre precisão e revocação, vemos que para a maximização da revocação, iremos diminuir o número de falso negativo. Na maximização da precisão iremos diminuir o número de falso positivo.

Usar cada uma delas separadamente tem suas desvantagens, a não ser que o foco é minimizar fp ou fn apenas. Analisar diversas métricas simultaneamente não é algo prático, com isso surgir a F-score, uma métrica que nos traz um desempenho geral do modelo, ela é formada pela média harmônica entre precisão e revocação, demonstrada na equação 21.

$$\text{F-score} = 2\frac{RP}{R + P} \tag{21}$$

A média harmônica entre dois números tende a ser mais próximo do menor valor (SUN; WONG; KAMEL, 2009), ou seja, para F-score ter um valor alto, tanto precisão quanto a revocação devem ter valores altos.

Mais uma métrica é usada frequentemente para análise com base de dados desbalanceada, a Características Operacionais do Receptor (COR). Essa métrica gera um gráfico onde temos a taxa de vp (revocação) e a taxa de fp, veja equação 22. As taxa são calculadas para diferente valores de limite. A taxa de confiança de uma classificação tem que atingir no mínimo esse valor para ser considerada vp.

$$\text{Taxa de fp} = \frac{fp}{fp + vn} \tag{22}$$

A COR é construída com a revocação no eixo y e a taxa de fp no eixo x, um exemplo é demonstrado na figura 11. O ponto ideal é o canto superior esquerdo, cujo o valor de fp tende a 0 e vp tende a 1. Ao plotar o gráfico temos uma curva descrevendo o comportamento da capacidade de predição do modelo independente da distribuição das classes na base de dados (PROVOST; FAWCETT; KOHAVI, 1998).

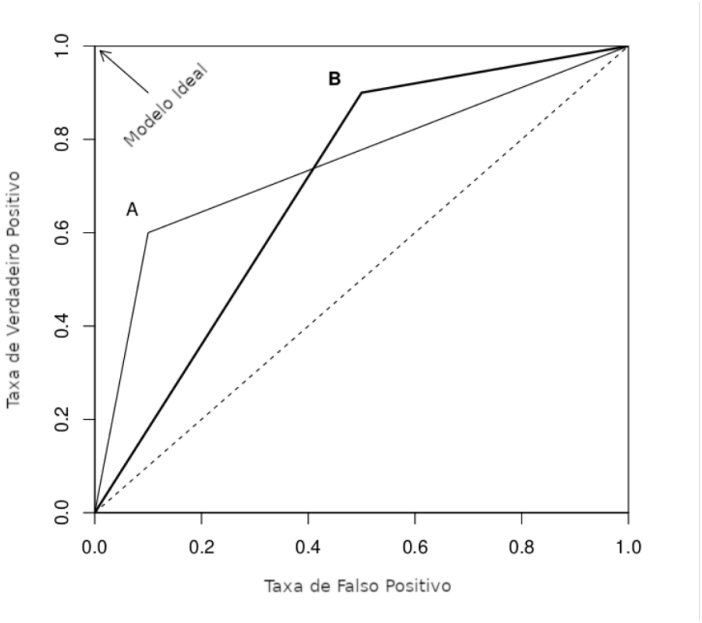


Figura 11 – Curva COR. Fonte Branco, Torgo e Ribeiro (2015)

Analisar a curva COR graficamente não é algo prático, diante dessa limitação Bradley (1997) sugeriu o uso da área sobre a curva como métrica de avaliação de performance, possuindo agora um único valor para análise, que varia entre 0,5 e 1. Quanto maior melhor o desempenho do modelo.

Outra métrica usada é a log loss, a definição dela para problemas multiclasse está na equação 23, onde n é quantidade de imagens da base de teste, m é quantidade de classes, y_{oc} é 1 se o pertence à classe c e 0 caso contrário, e p_{oc} é a probabilidade da observação pertencer a classe c. Quanto menor o valor de log loss melhor é o resultado e quanto maior pior é o resultado.

$$log\ loss = -\frac{1}{n}\sum_{o=1}^{n}\sum_{c=1}^{m} y_{oc} log(p_{oc}) \tag{23}$$

Há diversas métricas que podem ser usadas para avaliar o quão bem está um modelo, visando diferentes perspectivas. Para nós auxiliar na escolha das métricas adequadas para o domínio do nosso problema, recorremos a literatura que revisamos para este trabalho, a distribuição das métricas mais usadas são apresentadas na tabela 2.

A tabela 2 mostra que a maioria dos trabalhos usam mais de uma métrica para avaliar o modelo, isso se deve a dois fatores: Para efeitos de comparação, pois muitos trabalhos apresentam seus resultados com apenas uma métrica, e para ter uma visão de várias perspectivas do modelo.

Levando em consideração o que foi discutido acima, iremos adotar as métricas f-score e acurácia para avaliar os modelos. A f-score por conter informações sobre precisão e revocação ao mesmo tempo. A acurácia, para podermos comparar facilmente com outros trabalhos. Depois será utilizado a matriz de confusão e a curva COR, para avaliar a

Métricas	Artigos
Acurácia	Deep e Dash (2019), Qin et al. (2016), Tamou et al. (2018) Qiu et al. (2018), Khalifa, Taha e Hassanien (2018) Santos e Gonçalves (2019), Meng, Hirayama e Oyanagi (2018) Zheng et al. (2018), Montalbo e Hernandez (2019) Siddiqui et al. (2018), Shafait et al. (2016), Olsvik et al. (2019)
Precisão	Deep e Dash (2019), Christensen et al. (2018) Rathi, Jain e Indu (2017), Salman et al. (2016) Santos e Gonçalves (2019), Tamou et al. (2018) Montalbo e Hernandez (2019), Siddiqui et al. (2018)
Revocação	Deep e Dash (2019), Salman et al. (2016) Montalbo e Hernandez (2019), Siddiqui et al. (2018)
F-score	Santos e Gonçalves (2019), Montalbo e Hernandez (2019)
Log loss	Yang et al. (2018)
Matriz de confusão	Montalbo e Hernandez (2019), Siddiqui et al. (2018) Shafait et al. (2016), Olsvik et al. (2019)

Tabela 2 – Distribuição de métricas usadas nos artigos. Fonte: Autor.

performance do nosso modelo final.

2.5 Balanceamento da base de dados

Os métodos mais utilizados para balanceamento da base de dados, são baseados em amostragem, que consiste em modificar a estrutura do conjunto para deixar o número de amostras equivalentes para as classes, seja removendo (undersampling) ou adicionando (oversampling) novas amostras (MAIONE et al., 2020). Eles são amplamente utilizados, pelo melhor desempenho proporcionado, ao treinar um classificador com a base de dados balanceada.

O oversampling cria novas observações da classe minoritária a partir das informações contidas nos dados originais. Essa geração de novas entradas pode ser feita sinteticamente. O undersampling reduz o desbalanceamento da base de dados focando na classe majoritária, ou seja, elimina aleatoriamente amostras da classe com maior número de ocorrências.

3 MATERIAIS E MÉTODOS

3.1 Método Proposto

Tamou et al. (2018) argumenta que para o reconhecimento de espécies de peixes, treinar uma CNN do início não é recomendado por conta de não haver uma base de dados suficiente para um bom treinamento. Diante das dificuldades de se treinar uma CNN do início, neste trabalho utilizamos o conceito de transferência de aprendizado, que é utilizar os parâmetros de um modelo pré-treinado como um ponto de início. Quatro modelos pré-treinados serão testados e o que apresentar melhores resultados vai ser selecionado.

Os pesos são transferidos da seguinte forma, primeiramente é realizado o carregamento dos pesos do modelo pré-treinado, removemos os pesos das últimas camadas que são as densas. Após isso, os pesos são inseridos em nosso extrator de características, que é o modelo selecionado sem as mesmas camadas.

Dois treinamentos são realizados, o segundo ajustando todos os pesos do modelo e o primeiro treinamento sem ajustar os pesos recém transferidos, para não perde-los devido o custo alto que a função objetiva teria por conta das camadas não treinadas do classificador.

Os modelos que serão testados são Xception (CHOLLET, 2017), InceptionResNet-V2 (SZEGEDY et al., 2017), EfficientNet-B4 (TAN; LE, 2019) e ResNet-V2 (HE et al., 2016) todos os modelos treinados na base de dados ImageNet. Para a classificação, utilizaremos uma ou mais camadas densas com a ativação Relu seguido de uma camada de eliminação e mais uma camada densa usando a ativação softmax para realizar a predição. Veja a figura 12.

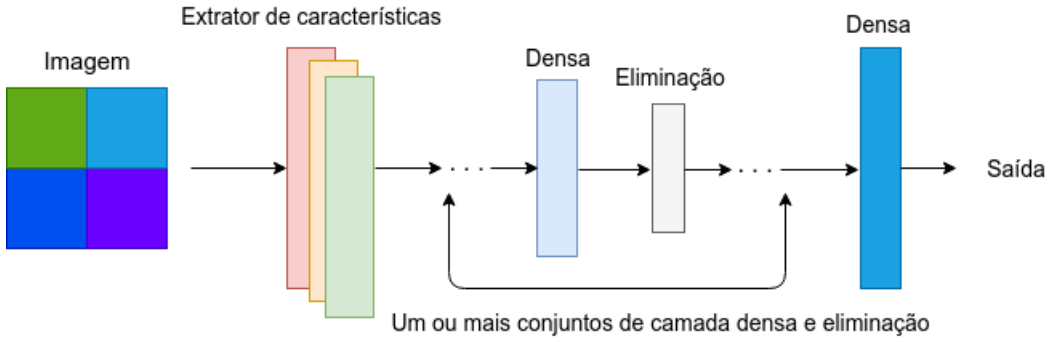


Figura 12 – Modelo

O motivo de todos serem treinados na base de dados ImageNet é por conta de uma observação feita por Azizpour et al. (2015). Ele destaca que as características aprendidas sobre a base dados, atingem altos níveis de generalização para outras tarefas, tornando-se uma representação mais transferível pelo fato de as classes serem muito diversificadas.

O critério de escolha dos modelos selecionados foi a comparação de desempenho na base de dados ImageNet. Os modelos tiveram acurácia top-1 sendo Xception 79%, InceptionResNet-V2 80,30%, EfficientNet-B4 82,9% e ResNet-V2 79,0%.

3.1.1 Avaliando diferentes modelos

Definir a quantidade de camadas e neurônios na segunda parte do modelo não é algo trivial, em vista do enorme conjunto de possibilidades. Uma solução usada é tratar o problema da otimização de um modelo, como um problema de busca, assim, são definidas diversas quantidades diferentes de camadas e neurônios para serem testados com diferentes combinações de otimizadores e outras variáveis de treinamento.

O problema desse método é a quantidade computacional excessiva que ele requer, elevando o tempo de execução muito além do que podemos conceber.

Para delimitar o campo de busca, se definiu 8 diferentes configurações de modelos, essa avaliação tem como objetivo identificar se na parte do classificador é preferível ter diversas ou poucas camadas densas e se elas devem ter um número alto ou baixo de neurônios, os modelos avaliados são mostrados na figura 13, onde N é a quantidade de espécies.

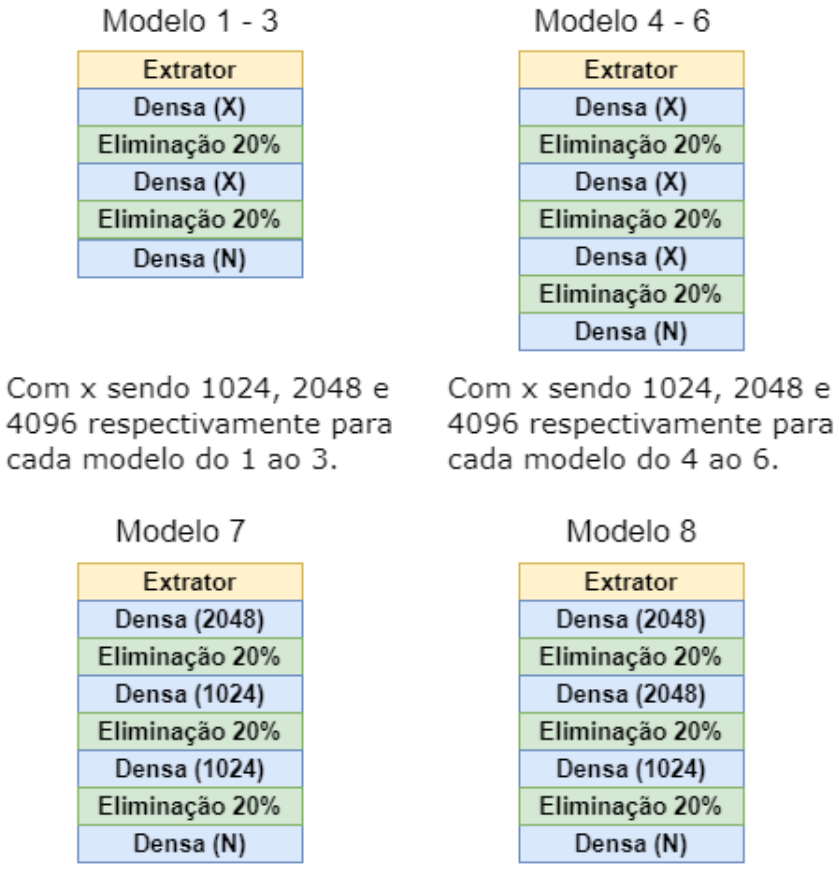


Figura 13 – Modelagens para a segunda parte do modelo.

Os modelos observados na figura 13, tem como valor padrão para camada de

eliminação o valor 20%. Foi mantido assim para diminuir a quantidade de modelos, após a definição do modelo serão avaliados outros valores.

Os modelos serão avaliados nas bases de dados AlevinosRV e base geral. O foco deste trabalho é desenvolver um sistema de classificação para a base de dados AlevinosRV, porém, também será proposto um modelo especifico para a base geral, no objetivo de se ter melhores parâmetros para transferência de aprendizado.

Além de utilizar a transferência de aprendizado de um modelo pré-treinado na base de dados ImageNet, será realizado a transferência de parâmetros do modelo treinado na base geral para o modelo que será treinado na base AlevinosRV. O objetivo é conseguir características de peixes globais e ter uma melhor performance após o treinamento.

3.1.2 Regularização

Tendo definido a modelagem principal, o objetivo é otimizar algumas variáveis que foram mantidas fixas durante os últimos experimentos e experimentar diferentes técnicas de regularização para reduzir o sobreajuste.

Como explicado na sessão 2.1.5, na camada de eliminação é definido a porcentagem de neurônios que serão "retirados" aleatoriamente e temporariamente durante o treinamento. Serão avaliados os valores 30%, 40% e 50% para a camada de eliminação, visando encontrar os que otimizam os resultados da modelagem definida. O mesmo valor é atribuído para todas as camadas de eliminação.

As técnicas de regularização dos pesos discutidos na sessão 2.1.6, serão avaliados para descobrir se um em específico nos traz melhores resultados. Fixamos o parâmetro α sendo 0,01 para as normas.

Será avaliado os valores 0,001, 0,0001 e 0,00001 para α da norma escolhida, para se verificar se é obtido melhores resultados do que a última avaliação com o valor 0,01.

3.2 Base de Dados

A base de dados é uma parte fundamental do trabalho, pois sem uma base de dados adequada ao modelo, ele pode falhar na generalização do problema. Foi levantado duas bases de dados, uma desenvolvida e outra com bases de dados disponíveis na internet junto com a que desenvolvemos.

3.2.1 Aquisição das imagens

Desenvolvemos nossa própria base de dados tirando fotos de 11 espécies de peixes alevinos, vamos referir a essa base como AlevinosRV. As fotos foram tiradas por um celular XIAOMI Mi 9 Lite e Moto G5, em 4 datas diferentes entre 2018 e 2020, sendo que dois padrões foram utilizados para tirar as imagens. O primeiro consiste em retirar o peixe da água e colocar em cima de um fundo branco. Já o segundo, os peixes são colocados dentro

de um aquário transparente onde a água não é totalmente limpa e as imagens são tiradas, veja na figura 14. O intuito é obter imagens mais próximas de seu ambiente natural, com um padrão de fundo diversificado na base de dados o modelo tem menores chances de ficar "viciado"em um fundo especifico. Ao todo foram tiradas 1455 imagens e a distribuição por espécies é descrito na tabela 3.

(a) Ambiente artificial

(b) Peixe fora da água em um fundo branco

Figura 14 – Exemplos de como foram tiradas as imagens.

Espécie	Quantidade
Cachara pura	86
Carpa capim	69
Carpas coloridas KOI	181
Tilápia	76
Jundiá rosa	36
Lambari	69
Lambari rosa	48
Pacu	198
Pintado judiará	66
Tambaqui	324
Tambatinga	277
Total	1455

Tabela 3 – Distribuição das imagens por espécies

3.2.2 Desafios impostos pela base de dados

Em um problema de classificação entre dois objetos como um sofá e uma bola de futebol, não é tão complicado para um modelo aprender a diferenciar ambos. Na nossa base de dados ocorre dois casos que dificulta de maneira significativa a classificação correta das classes.

O primeiro é a diferença de cor entre peixes da mesma espécie, um caso é retratado na figura 15. Já o segundo caso, é a semelhança entre peixes de diferentes espécies, veja na figura 16.

Tamou et al. (2018) teve dificuldades de classificação correta nesse contexto. As semelhanças são ainda maiores quando os peixes são alevinos, o que deixa a classificação

uma tarefa desafiadora, e mesmo que RNC profundas aprendam características de alto nível, não é certeza que essas características aprendidas colaboram com a classificação correta (ZHENG et al., 2018). No primeiro caso, percebemos que se o modelo dar uma atenção maior as cores do peixe não seria algo que ajudasse realmente na classificação.

(a) Carpa colorida

(b) Carpa colorida

Figura 15 – Diferenças entre peixes da mesma espécie.

(a) Tambaqui

(b) Tambatinga

Figura 16 – Semelhanças entre peixes de diferentes espécies.

3.2.3 Base de dados geral

Colhemos diversas bases de dados de imagens de espécies de peixes disponíveis na internet e juntamos para formar um conjunto só, e vamos nos referir a ele como base geral.

A base geral é composta pelos conjuntos Fish4Knowledge (FISHER et al., 2016), base de dados da competição realizada no Kaggle chamada de "The Nature Conservancy Fisheries Monitoring", a base composta por imagens que foram tiradas em um aquário e a base AlevinosRV.

Cada base tem suas características distintas, como o fundo da imagem, posição da câmera, ambiente e condições do ambiente. Essas diferentes características tornam o conjunto rico em diversidade, assim tendo mais possibilidade de uma generalização do modelo e menos chance de que ele fique "viciado"em alguma característica forte de uma base de dados, como por exemplo, o fundo da imagem.

No total, se somam 65 espécies, totalizando 33009 imagens, sendo tanto de água doce e salgada. A tabela 4 mostra a distribuição da quantidade de imagens por espécies.

Espécie	Quantidade	Espécie	Quantidade
Abudefduf vaigiensis	98	Molinesia preta	40
Acanthurus nigrofuscus	218	Myripristis kuntee	450
Acara bandeira	40	Neoglyphidodon nigroris	16
Acara bandeira marmorizado	40	Neoniphon sammara	299
Acara disco	49	Oscar	40
Albacore tuna	1719	Oscar albino	40
Amphiprion clarkii	4049	Pacu	238
Balistapus undulatus	41	Palhaco	81
Barbus ouro	40	Papagaio	40
Barbus sumatra	40	Paulistinha	40
Beta	40	Pempheris vanicolensis	29
Bigeye tuna	200	Piau tres pintas	40
Cachara pura	86	Pintado jundiara	66
Canthigaster valentini	147	Platy laranja	41
Carpa capim	69	Platy rubi	40
Carpas coloridas KOI	261	Platy sangue	40
Chaetodon lunulatus	2534	Plectroglyphidodon dickii	2683
Chaetodon trifascialis	190	Pomacentrus moluccensis	181
Chromis chrysura	3593	Scaridae	56
Dascyllus reticulatus	12112	Scolopsis bilineata	49
Dolphinfish	117	Shark	176
Dourado	40	Siganus fuscescens	25
Hemigymnus fasciatus	241	Tambaqui	302
Hemigymnus melapterus	42	Tambatinga	324
Jundia rosa	36	Telescopio	40
Kinguio	40	Tetra negro	40
Kinguio cometa calico	40	Tilapia	76
Kinguio korraco	40	Tricogaster	40
Lambari	69	Tucunare	40
Lambari rosa	48	Yellowfin tuna	734
Lampris guttatus	67	Zanclus cornutus	21
Lutjanus fulvus	206	Zebrasoma scopas	90
Mato grosso	40		
		Total	33009

Tabela 4 – Distribuição das imagens por espécies da base de dados geral

3.2.4 Aumento da base de dados

A quantidade de imagens que obtemos ainda não é considerada suficiente mesmo utilizando transferência de aprendizado. Olsvik et al. (2019) comenta que as técnicas de aumento de dados, tem como objetivo principal reduzir o sobreajuste. O aumento

dos dados é realizado pegando amostras do conjunto de dados da base de treinamento e gerando novas amostras com métodos de transformações que preservam a classe.

No intuito de aumentar a base de dados, geramos novas imagens rotacionando, transpondo e mudando a escala. Para auxiliar na escolha dos valores que são necessários para aplicar a cada método, levantamos pela bibliografia feita os métodos e valores utilizados quando disponíveis, a tabela 5 expõem os métodos encontrados.

Métodos	Valores e artigos
Rotação	-180° á 170° - Meng, Hirayama e Oyanagi (2018), -10° á 10° - Deep e Dash (2019), 90° , 180° , 270° para cada imagem - Qiu et al. (2018), 0° á 30° - Santos e Gonçalves (2019), Olsvik et al. (2019), Zheng et al. (2018), Yang et al. (2018)
Escala	100% á 120% - Santos e Gonçalves (2019), 100% á 200% - Tamou et al. (2018), Olsvik et al. (2019), Yang et al. (2018)
Transposição	Horizontal e vertical - Qiu et al. (2018) Horizontal - Christensen et al. (2018), Olsvik et al. (2019), Horizontal - Tamou et al. (2018), Yang et al. (2018)

Tabela 5 – Visão geral dos métodos de transformações de imagens utilizados pelos trabalhos na área. Fonte: Autor.

Optamos por rotacionar as imagens entre -170° a 170° aplicar escala entre 80% á 130% e transposição horizontal e vertical.

Definir como será feito o aumento da base de dados não é uma tarefa fácil, pois não basta apenas gerar um número grande de imagens de forma arbitrária, nosso objetivo é fazer o aumento de dados de forma que seja realizado a diminuição do desbalanceamento, ao mesmo tempo que não tornamos o conjunto de dados pouco representativo gerando muitas imagens em excesso com a mesma imagem, o que pode acabar causando o sobreajuste do modelo.

Para lidar com o aumento e balanceamento dos dados, desenvolvemos um algoritmo baseado em oversampling que gera novas imagens para cada classe de acordo com a classe dominante. No algoritmo 2, temos que é necessário definir dois valores, o máximo que cada classe pode crescer e o mínimo que ela deve crescer. Esses valores devem ser definidos levando em conta o desbalanceamento e o tamanho da base de dados atual. Da linha 4 a linha 6, queremos saber se o mínimo aumentado em todo o seu potencial, será maior que a classe dominante incrementado o mínimo, caso sim, essa quantidade será considerado para efeitos de cálculos, sendo a classe dominante. Isso é feito para limitar o crescimento das outras classes em relação a ela e manter a base de dados mais balanceada possível. Após determinar o número de imagens que resultará na classe dominante, é feito o cálculo de novas imagens para cada classe em relação a dominante, o cálculo na linha 9 é simplesmente

a divisão entre os dois valores, o resultado será quantidade de vezes que a classe analisada é inferior a classe dominante, essa quantidade será o fator gerador, ou seja, ele nos dirá quantas novas imagens serão geradas, limitando-se aos valores máximos e mínimos definidos.

Algoritmo 2: Algoritmo de aumento de dados

Entrada: Vetor de classes C, minGeradoPorClasse e maxGeradoPorClasse
Saída: Base de dados aumentada

```
1  início
2      classeMax ← max(C);
3      classeMin ← min(C);
4      qtdClasseMinGerada ← classeMin * classeMaxGeradoPorClasse ;
5      qtdMinClasseMaxGerado ← classeMax * minGeradoPorClasse ;
6      classeMax ← max( qtdClasseMinGerada, qtdMinClasseMaxGerado );
7      para classe em C faça
8          qtdClasseAtual ← tamanho(classe);
9          fatorGerador ← classeMax / qtdClasseAtual;
10         fatorGerador ← max(fatorGerador, minGeradoPorClasse);
11         fatorGerador ← min(fatorGerador, maxGeradoPorClasse);
12         qtdNovasImagens = (fatorGerador * qtdClasseAtual) - qtdClasseAtual;
13         gerarImagens (classe, qtdNovasImagens);
14     fim
15 fim
```

Como uma forma de medir o desbalanceamento, calculamos um valor escalar para representar seu efeito negativo, descrito na equação 24, o valor vária entre 1 e 0, quanto mais perto de 1, maior o desbalanceamento e mais perto de 0, menor o desbalanceamento.

$$e = \frac{1}{n}\sum_{i=1}^{n} 1 - \frac{c_i}{max} \tag{24}$$

3.3 Desenvolvimento

Os experimentos realizados irão seguir a seguinte ordem, primeiro iremos avaliar os modelos do estado da arte para nosso problema e escolher o que melhor dar resultados pelas métricas definida na sessão 2.4, para atuar como extrator de características, então iremos realizar diversos experimentos com diferentes configurações de uma rede neural para a tarefa de classificação, após isso será feito um refinamento dos parâmetros utilizados para treinamento a fim de escolher as melhores configurações, por fim será realizado teste de eficiência com o algoritmo de aumento de dados.

3.3.1 Ferramentas

Utilizamos a linguagem python com a biblioteca de alto nível Keras com o backend TensorFlow para desenvolver e treinar nosso modelo.

3.3.1.1 Python

Python é uma linguagem de programação de alto nível e orientada a objetos, além de ser modular, interpretada e multiplataforma. Python se tornou uma linguagem de programação famosa pela a sua sintaxe fácil e acessível. Umas das vantagens do python, é que ele conta com uma quantidade enorme de bibliotecas, tanto nativa quanto de terceiros, oferecendo ferramentas simples para se trabalhar com aprendizado de máquina.

Outra grande vantagem é a possibilidade de se interagir com o código por meio de um terminal ou Jupyter Notebook. O python é também usado para criar interfaces gráficas e web services (RASCHKA, 2015).

3.3.1.2 TensorFlow

TensorFlow pode ser definido como uma interface para escrever algoritmos de aprendizado de máquina, tendo diversas características interessantes que o deixa uma opção muito atraente para se desenvolver algoritmos e realizar treinamentos com grandes quantidades de dados (ABADI et al., 2016).

Umas dessas características é que ele oferece diversos níveis de abstração, sendo ideal para quem está começando e para quem já é experiente e precisa de mais flexibilidade. TensorFlow é a segunda geração de sistemas da google para implementar modelos de aprendizado de máquina em larga escala.

3.3.1.3 Visão geral do sistema

Os experimentos foram realizados na plataforma online Colab, que pertence ao Google e nela podemos utilizar alguns recursos computacionais de alto desempenho para acelerar nossos treinamentos, como GPUs e TPUs, as TPUs são Unidades de Processamento de Tensores, que são dispositivos criado para processar com alto desempenho esse tipo de dado que é o tensor, o tensor é o dado que o TensorFlow usa. Também utilizamos para treinamento um workstation equipado com um Xeon, 16GB de memória RAM e uma placa de vídeo Nvidea Quadro P-4000.

3.3.2 Parâmetros de treinamento inicial

O otimizador será o Adam, por ser eficiente e requerer pouca memória (KINGMA; BA, 2014). A função de custo será a entropia cruzada categórica. Como já mencionado anteriormente, a parada antecipada levará em conta 3 épocas. A taxa de aprendizado terá valor mínimo de 0,00001 e valor máximo de 0,0004 com decaimento de peso. O tamanho do lote será de 128 quando estivermos treinando o modelo no colab e 64 quando estivermos usando a workstation para treinamento.

3.4 Sistema desenvolvido

Com o modelo definido e treinado, ele será embarcado em um sistema para classificar imagens. O sistema é uma Interface de programação de aplicativo (API, na sigla em inglês). Uma API pode ser definida como um software intermediário entre duas aplicações.

Nosso objetivo com a API é que ela possa ser implementada em diversos sistemas e aplicativos de forma simples, oferecendo uma comunicação com o modelo que está em execução em um servidor.

Na figura 17, encontra-se um esquema exemplificando sua utilização, onde um computador captura imagens dos peixes através de uma câmera e envia uma requisição com a imagem pela API, para que seja realizado a classificação pelo modelo que está em execução no servidor, e após o processamento é retornado o resultado.

Figura 17 – Esquema de utilização da API.

A API foi desenvolvida com a linguagem python junto com a biblioteca FastApi. O python foi escolhido por ser a mesma linguagem utilizada para desenvolver o modelo e a biblioteca FastApi, por ser uma opção simples para desenvolver APIs. 2 rotas principais são implementadas na API, uma rota para realizar a classificação de uma imagem e outra rota para obter informações das espécies que o modelo é capaz de classificar.

4 Resultados e discussão

4.1 Modelos do estado da arte

Os modelos do estado da arte citados na sessão 3.1, foram treinados na base de dados AlevinosRV e na base de dados geral. A tabela 6 mostra os resultados de cada modelo na base AlevinosRV e a tabela 7 para a base geral.

	Acurácia (%)	F-score (%)
Xception	**95,57**	**95,80**
InceptionResnetV2	92,20	92,28
ResNetV2	80,12	81,05
EfficentB5	22,47	0

Tabela 6 – Resultados de modelos do estado da arte na base AlevinosRV.

	Acurácia (%)	F-score (%)
Xception	**99,16**	**99,17**
InceptionResnetV2	98,42	98,46
EfficentB5	98,17	98,20
ResNet152V2	96,93	96,98

Tabela 7 – Resultados de modelos do estado da arte na base de dados geral.

Analisando a tabela 6, o modelo Xception foi o que melhor apresentou resultados em todas as métricas, com uma diferença considerável do segundo melhor resultado, mostrando que ele é bastante eficiência para extrair características que sejam úteis para a classificação correta das espécies. O modelo EfficientB5 mostrou total ineficiência para o domínio do problema, não conseguindo encontrar relacionamentos entre a entrada e a saída com essa base de dados.

Já a tabela 7, mostra que o Xception ainda é o que melhor apresenta resultados nas 2 métricas, mas dessa vez, sem uma diferença muito discrepante dos demais. Isso se deve pela base de dados geral ser bem maior e os modelos conseguem aprender e abstrair mais características relevantes para a classificação, mostrando que a base de dados influência diretamente no resultado final.

Diante dos resultados, o modelo Xception foi o escolhido para atuar como extrator de características no nosso modelo, pois além de ter tido os melhores resultados, ele se mostrou eficiente ao lidar com uma base de dados com pouca representatividade.

4.2 Avaliando diferentes modelos

Os resultados dos experimentos com os modelos descritos na sessão 3.1.1 na base AlevinosRV está na tabela 8 e os resultados na base geral estão na tabela 9.

	Acurácia (%)	F-score (%)
Modelo 1	93,98	93,94
Modelo 2	**94,91**	**95,12**
Modelo 3	94,44	94,20
Modelo 4	**94,91**	94,34
Modelo 5	93,98	93,65
Modelo 6	93,52	93,72
Modelo 7	89,35	89,44
Modelo 8	92,59	92,59

Tabela 8 – Resultados de diversas configurações de modelos utilizando o Xception na base AlevinosRV.

	Acurácia (%)	F-score (%)
Modelo 1	98,75	98,71
Modelo 2	98,87	98,87
Modelo 3	98,43	98,49
Modelo 4	98,90	98,94
Modelo 5	98,23	98,22
Modelo 6	98,29	98,33
Modelo 7	98,39	98,46
Modelo 8	**99,30**	**99,27**

Tabela 9 – Resultados de diversas configurações de modelos utilizando o Xception na base geral.

A tabela 8 mostra que o modelo 2 teve um desempenho melhor na métrica f-score e empatou com o modelo 4 na acurácia. O resultado no f-score teve uma diferença relevante do segundo melhor resultado, mostrando que o modelo conseguiu manter tanto a precisão quanto a revocação em níveis mais altos dos demais, indicando uma classificação mais confiável para as espécies minoritárias.

Pelo fato da base de dados ter um número pequeno de classes e amostras, era esperado um modelo com um número menor de conexões, fosse lidar melhor com o problema, pois com um número muito grande de conexões, precisa de uma base de dados maior e com um domínio mais complexo para se valer delas, se não ocorre o sobreajuste.

Já na base de dados geral, os resultados na tabela 9 demonstram que o modelo 8 teve o melhor desempenho em todas as métricas. Em comparação com o modelo 2, há a adição de uma camada na penúltima posição tendo 1024 NA e as anteriores são iguais. Podendo concluir que 2048 é um número suficientes de NAs por camada, para lidar com

classificação de peixes utilizando o Xception como extrator de características e com as características das bases de dados aqui testadas.

Tivemos um modelo diferente que melhor se saiu nos resultados em cada base de dados, o que é esperado pela diferença muito grande entre os conjuntos. Como o conjunto de dados AlevinosRV é bem menor em números de imagens e classes em relação a base de dados geral, o modelo resultante há de ter menos conexões para não causar o sobreajuste.

É proposto o uso dos dois modelos para se empregar a tarefa de classificação de peixes em dois contextos: o modelo 2 deve ser usado quando se tem o objetivo de fazer apenas a classificação de espécies específicas, nos moldes do conjunto levantado. E o modelo 8 para uma classificação com um número muito grande e variedade de espécies, sendo mais "robusto"para a tarefa.

4.3 Regularização

4.3.1 Camada de eliminação

Os resultados do modelo 2 na base AlevinosRV são expostos na tabela 10 e a tabela 11 mostra os resultados do modelo 8 na base geral, com diferentes valores para a camada de eliminação.

	Acurácia (%)	F-score (%)
Modelo 2 - 20%	94,91	95,12
Modelo 2 - 30%	95,37	95,36
Modelo 2 - 40%	91,20	91,82
Modelo 2 - 50%	**95,83**	**95,83**

Tabela 10 – Resultados do modelo 2 com diferentes valores na camada de eliminação.

	Acurácia (%)	F-score (%)
Modelo 8 - 20%	**99,30**	99,27
Modelo 8 - 30%	99,08	99,13
Modelo 8 - 40%	99,00	99,00
Modelo 8 - 50%	**99,30**	**99,32**

Tabela 11 – Resultados do modelo 8 com diferentes valores na camada de eliminação.

A partir dos resultados expostos, temos que na camada de eliminação, o valor ideal seria de 50% para os dois modelos. No modelo 2, os resultados foram superiores para todas as métricas, com 95,83% de acurácia e f-score. O modelo 8 teve resultados superiores, porém apertado na métrica f-score tendo 99,32% e empatando na acurácia com o modelo que tem 20% na camada de eliminação.

4.3.2 Regularização dos pesos

As tabelas 12 e 13 mostram os resultados obtidos nos experimentos com regularização de pesos.

Norma	Acurácia (%)	F-score (%)
L1	39,35	0,0
L2	**98,15**	**98,15**
L1 e L2	42,13	0,0

Tabela 12 – Resultados do modelo 2 com diferentes normalizadores de pesos.

Norma	Acurácia (%)	F-score (%)
L1	36,14	35,02
L2	**97,35**	**98,06**
L1 e L2	36,14	34,77

Tabela 13 – Resultados do modelo 8 com diferentes normalizadores de pesos.

Entre as três normas, a L2 foi a única que trouxe bons resultados em nossos testes, melhorando em até 2,32% os resultados do modelo 2, atingindo 98,15% nas duas métricas, como mostra a tabela 12. No modelo 8 o resultado com a norma L2 foi inferior do último obtido, mas ainda em alto patamar, esse resultado pode ser talvez melhorado alterando o parâmetro α da norma L2.

As tabelas 14 e 15 contém os resultados obtidos com os diferentes valores para α da norma L2. Analisando os resultados da tabela 14, temos que para o modelo 2, não teve ganho de performance com nenhum dos valores proposto para α.

É observado, que nos resultados com o modelo 8 na tabela 15, obtivemos ganho de performance em relação ao último obtido, no entanto, os resultados ainda são inferiores em relação aos testes com a camada de eliminação na tabela 11, mas os valores ficaram bem próximos entre si, tendo uma pequena diferença.

Tomamos como decisão não usar regularização de pesos no modelo 8, por não ter tido ganho de performance que justificasse seu uso.

α	Acurácia (%)	F-score (%)
0,001	95,83	96,5
0,0001	**97,22**	**97,22**
0,00001	**97,22**	**97,22**

Tabela 14 – Resultados de diferentes valores de α da norma L2 no modelo 2.

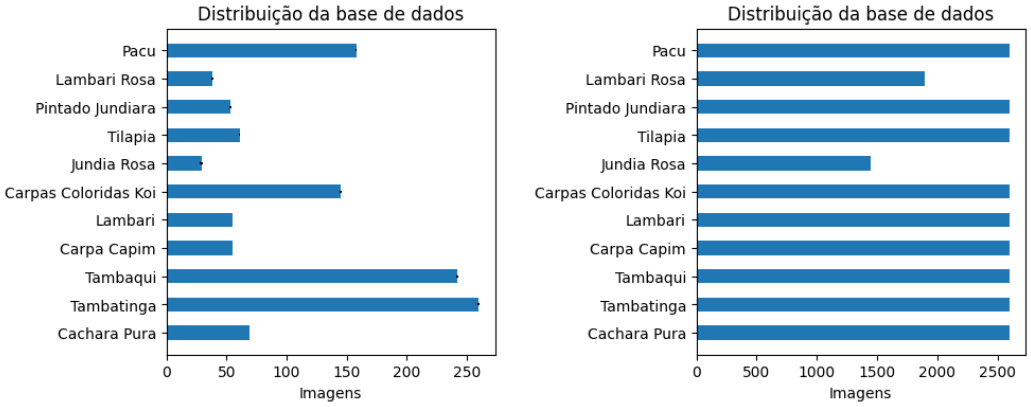
α	Acurácia (%)	F-score (%)
0,001	99,22	99,28
0,0001	**99,28**	**99,31**
0,00001	99,16	99,22

Tabela 15 – Resultados de diferentes valores de α da norma L2 no modelo 8.

4.4 Aumento da base de dados

Como explicado anteriormente na sessão 3.2.4, quando temos uma base de dados muito pequena, acabamos se deparando com problemas de sub-ajuste, não só isso, se a base de dados for muito desbalanceada, algumas classes acaba sendo suprimida pelas classes dominantes e o modelo não consegue aprender as características das classes minoritárias.

Aplicando a métrica de avaliação definida na equação 24 na base de dados AlevinosRV, temos que, para a base de dados sem tratamento, o valor de 0,59 e com o tratamento utilizando mínimo gerado por classe igual a 1000% e máximo 5000%, o valor de 0,06. Mostrando que o algoritmo reduziu de maneira significativa o desbalanceamento de acordo com a métrica de avaliação definida. O gráfico na figura 18 mostra a comparação do antes e depois.



(a) Sem aplicação do algoritmo (b) Após aplicação do algoritmo 2

Figura 18 – Base de dados AlevinosRV com e sem tratamento de aumento de dados. Fonte Autor.

A base geral é extremamente desbalanceada, aplicando a métrica de avaliação, temos o valor de 0,95. Indicando uma enorme diferença entre as classes dominantes e minoritárias, e de fato, a diferença de quantidade de imagens é extremamente alta, a classe dominante tem 12112 imagens e a minoritária apenas 16.

Após aplicar o algoritmo de aumento de dados, para aumentar no máximo 5000% cada classe e sem mínimo, obtivemos o valor de 0,56. Assim, diminuindo drasticamente o desbalanceamento, esse número poderia até ser melhorado aumento o limite de crescimento, no entanto, não é nosso objetivo tornar a base de treinamento extremamente grande e balanceada, no entanto, com baixa representatividade e diversidade do problema.

Realizando o treinamento com os novos conjuntos de dados, temos que, os resultados na tabela 16, mostram um ganho de performance nos dois modelos. No modelo 2 temos um ganho de 0,92% com as 2 métricas e o modelo 8 teve um leve ganho de performance.

	Acurácia (%)	F-score (%)
Modelo 2	99,07	99,07
Modelo 8	99,35	99,39

Tabela 16 – Performance dos modelos com a base de dados aumentada.

4.5 Transferência de aprendizado do modelo 8 para o modelo 2

Com o modelo 8 treinado, agora temos parâmetros no extrator de características com a capacidade de extrair características globais de peixes, por causa da imensa base de dados que ele foi treinado. Sendo parâmetros mais transferíveis para o nosso modelo 2, que será empregado na base AlevinosRV.

O resultado após transferir os parâmetros para o modelo 2 é mostrado na tabela 17, observamos que obtivemos uma melhora de 0,47% na performance do modelo, atingido o melhor resultado obtido de 99,54% nas 2 métricas.

	Acurácia (%)	F-score (%)
Modelo 2	99,54	99,54

Tabela 17 – Performance do modelo 2 como transferência de aprendizado do modelo 8.

4.6 Comparando com outros trabalhos e evolução do modelo

É feita a comparação da performance do nosso modelo com outros trabalhos relacionados que, utilizaram a base de dados Fish4Knowladge. Para realizar a comparação, dividimos a base de dados de acordo com o trabalho que tem o melhor resultado descrito. A tabela 18, mostra os resultados dos nossos modelos 2 e 8, em comparação com outros trabalhos da área.

Os dois modelos desenvolvidos, se mostraram ter um desempenho superior dos outros trabalhos, demonstrando a eficiência dos nossos modelos.

A figura 19, mostra a evolução do modelo desenvolvido com a base de dados AlevinosRV, de acordo em que foram realizadas as principais etapas do desenvolvimento. É possível verificar a importância de cada etapa, mostrando que apenas a definição da modelagem não é suficiente para termos um bom classificador.

Na figura 20, temos a matriz de confusão para podermos analisar em quais espécies o modelo tem mais dificuldades. O modelo teve dificuldades apenas em diferenciar as

Trabalho	Acurácia (%)
Modelo 2	99,73
Modelo 8	99,50
Tamou et al. (2018)	99,47
Rathi, Jain e Indu (2017)	99,29
Olsvik et al. (2019)	99,27
Deep e Dash (2019)	98,79
Qin et al. (2016)	98,64
Mathur et al. (2020)	98,03

Tabela 18 – Resultados de trabalhos utilizando a base de dados Fish4Knowladge

espécies Carpa Capim e Carpas KOI, uma das cores das Carpas Koi é bem parecida com a Carpa Capim, principalmente quando são alevinos.

Aplicando a curva COR com 10 limites para uma melhor análise de desempenho e calculando a área sobre a curva, é obtido o valor de 0,99. Demonstrando que o modelo é de fato eficiente. A figura 21 mostra a curva graficamente.

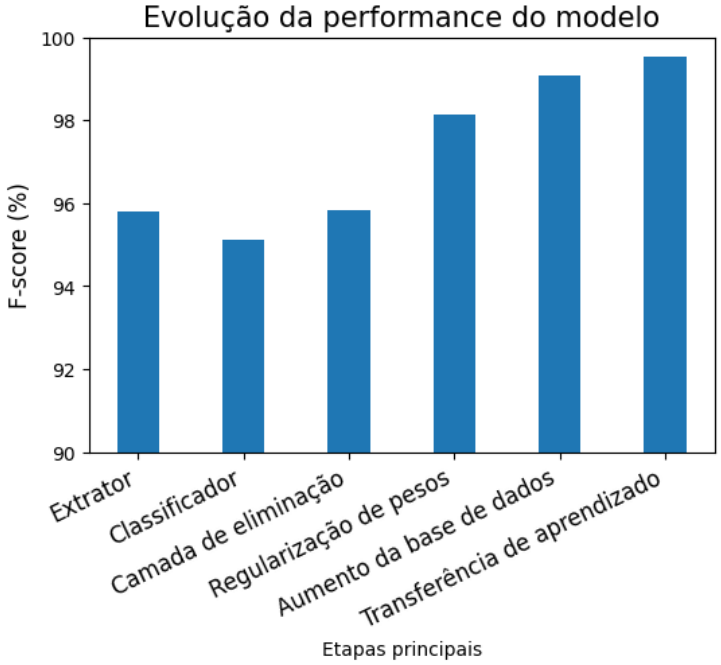


Figura 19 – Evolução do desempenho do modelo.

4.7 Sistema para classificação

O sistema foi desenvolvido com sucesso e implementado em um sistema WEB, desenvolvido em parceria com NEPEAQUA. O sistema WEB, tem a finalidade de gerar relatórios das imagens enviadas com a classificação da espécie, a figura 22 mostra um exemplo de uso.

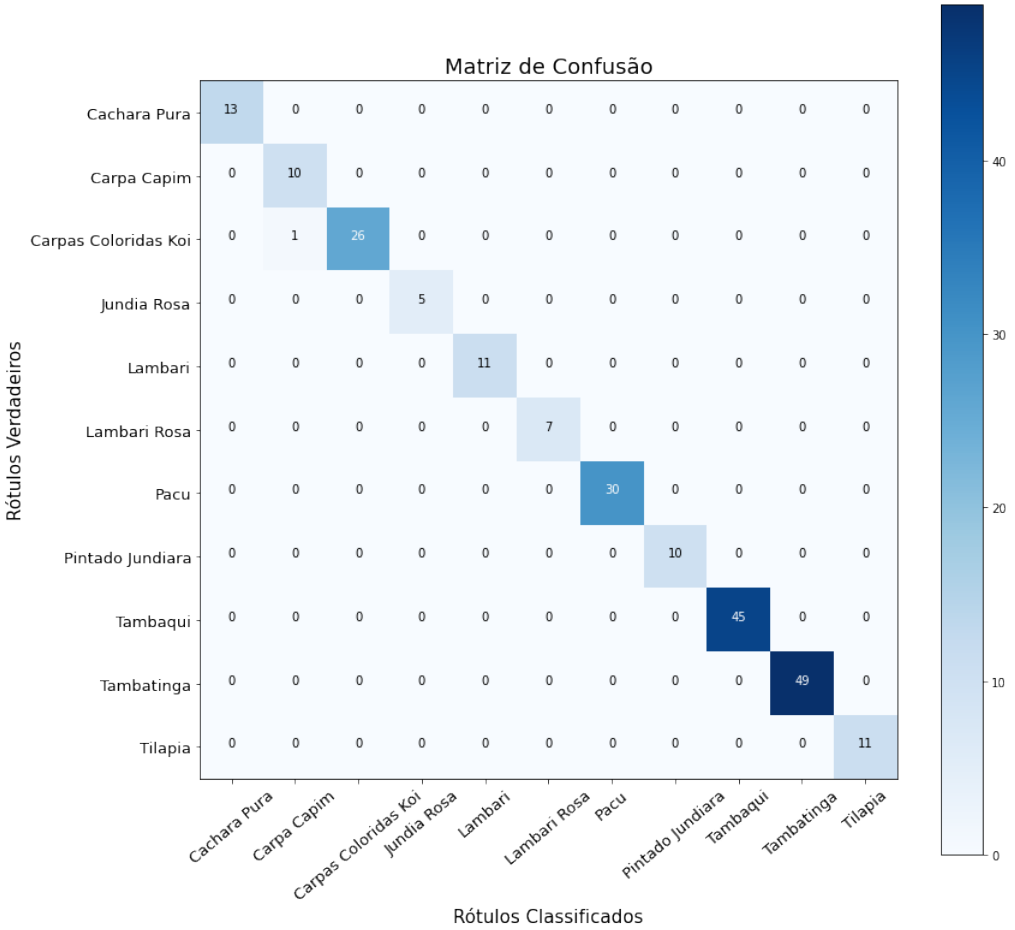


Figura 20 – Matriz de confusão do modelo 2 na base de dados AlevinosRV.

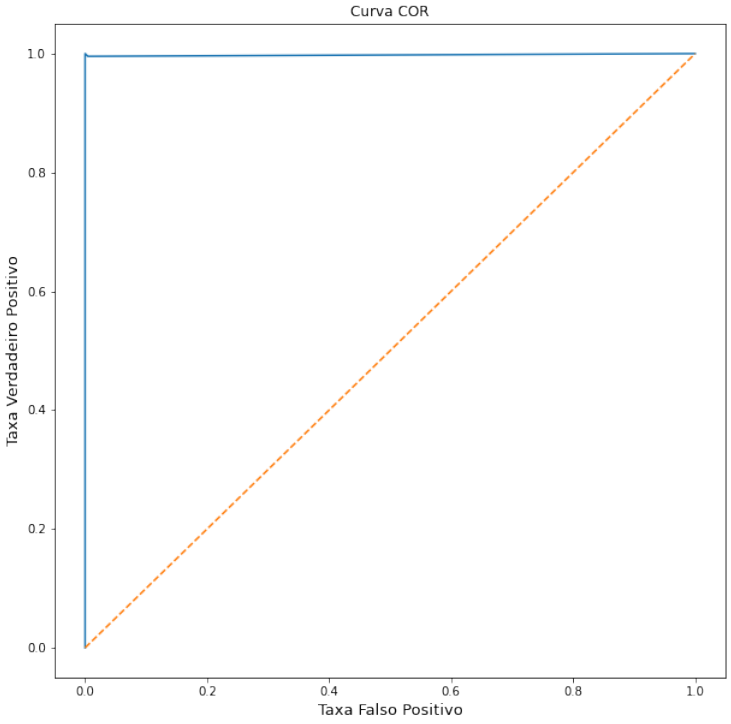


Figura 21 – Curva COR do modelo 2 na base de dados AlevinosRV.

No exemplo, duas imagens são enviadas para o nosso sistema através da rota "/predict", que retorna o nome da espécie e a confidencialidade da predição, como mostra o lado direito da figura, que se encontra aberto o DevTools, que são ferramentas para desenvolvedores web, e nela podemos visualizar as requisições feitas.

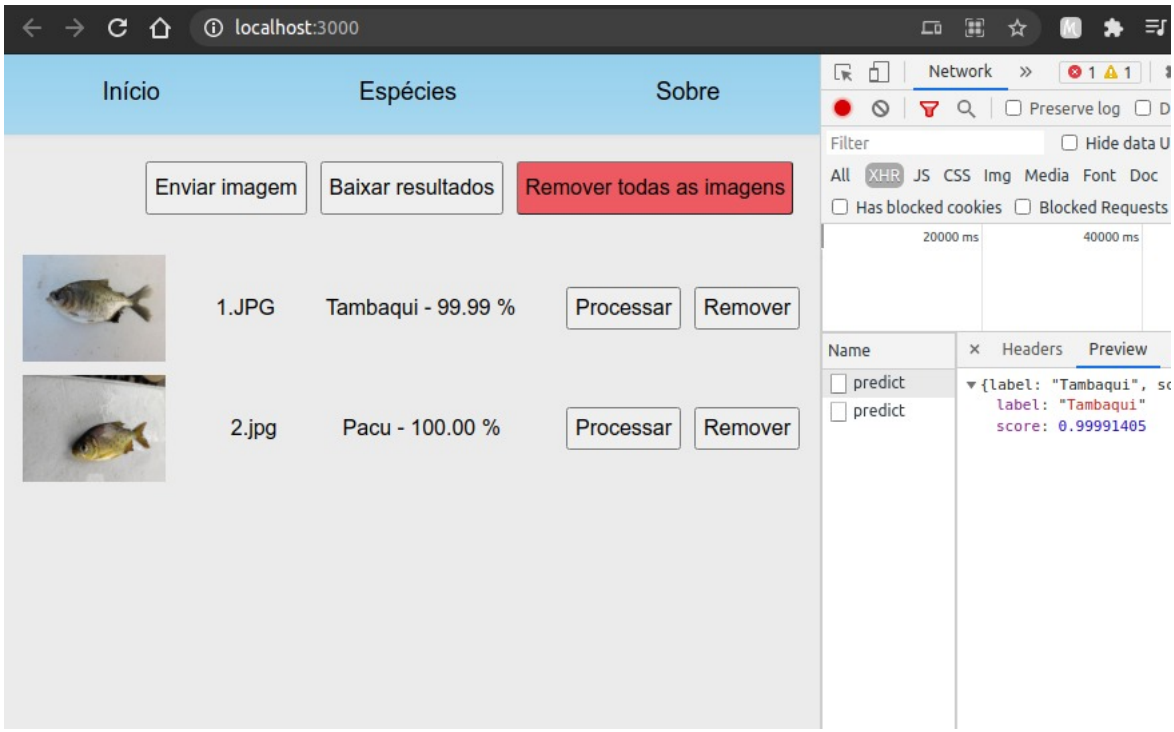


Figura 22 – Sistema WEB utilizando o sistema desenvolvido para classificar imagens.

O sistema pode estar na internet ou não, dependendo de como for disponibilizado seu uso. É possível manter o sistema funcionando em uma rede local, para que seja utilizado apenas pelos dispositivos conectados à rede, ou pode ser disponibilizado através da internet.

O uso através da internet tem a limitação da própria qualidade e disponibilidade da conexão, no entanto, com o advento da rede 5G, isso não será um obstáculo e cada vez mais dispositivos inteligentes funcionaram através da internet.

Uma grande vantagem do sistema desenvolvido, é que o modelo pode ser modificado a qualquer momento, sem necessitar a alteração ou intervenção no que o implementa.

5 CONCLUSÃO

Neste trabalho, buscamos desenvolver um modelo eficiente para classificação de espécies de peixes, para ser embarcado em sistemas e dispositivos que possam ser usados para auxiliar na automatização de processos na aquicultura, auxiliar cientistas nas análises de vídeos subaquáticos e até para entretenimento de pessoas em aquários.

Com esse desafio, fizemos uso de diversas técnicas e métodos para que o modelo proposto, possa classificar espécies com alto nível de confiança.

O modelo resultante para ser treinado com nossa base de dados AlevinosRV se mostrou muito eficiente utilizando o modelo Xception como extrator de características, poucas camadas densas no classificador e com um número considerado médio de neurônios por camada. Isso se deve ao fato do tamanho da nossa base de dados e o número de espécies ser pequeno, necessitando de um modelo com menos camadas para se ajustar de maneira satisfatória ao conjunto de dados. Conseguimos o resultado de 99,54% de acurácia e f-score.

Um ponto importante, foi como a regularização dos pesos nas camadas densas com a norma L2, ajudaram de maneira significativa aumentar o desempenho do modelo, mostrando a importância de se controlar o tamanho dos pesos quando trabalhamos com uma base de dados pequena.

O modelo utilizado com a base geral é um pouco mais profundo que o modelo 2, tendo uma camada densa a mais com 1028 neurônios. Essa camada densa extra é justificada pelo fato da base de dados ser bem maior que a AlevinosRV em quantidade de imagens e espécies. Um fato importante para se destacar, é que nesse modelo não foi necessário utilizar regularização dos pesos, por justamente a base de dados ser maior. Com esse modelo, conseguimos uma acurácia de 99,35%, 99,39% de F1-score.

A transferência de aprendizado se mostrou ser bem eficiente utilizando modelos do estado da arte, em especial o modelo Xception que escolhemos para ser nosso extrator de características. Utilizamos também a transferência de aprendizado, ao se transferir parâmetros do modelo treinado na base geral para o modelo que é treinado na base de dados AlevinosRV, aumentando a eficiência do mesmo.

A base de dados levantada foi utilizada com sucesso para classificação das espécies que coletamos as imagens. O algoritmo que desenvolvemos para aumentar a base de dados e reduzir o desbalanceamento, se mostrou eficiente em desempenhar esse papel, ajudando a melhorar a performance do modelo.

O sistema desenvolvido para disponibilizar o uso do modelo de forma simples e flexível, se mostrou ser uma boa opção para integrar em outros sistemas e dispositivos que necessitam de um classificador. O sistema foi registrado no Instituto Nacional da Propriedade Industrial, intitulado de ”API de Reconhecimento Automático de Espécies de

Peixe", cujo o número do processo é BR512021000956-8.

5.1 Trabalhos Futuros

O sistema de classificação pode ser estendido para realizar a predição da localização do peixe, aumentando o número de aplicações do mesmo, sendo possível por exemplo, a contagem da quantidade de peixes em uma imagem.

A aquisição das amostras é parte fundamental do processo de se construir um classificador. Fazer a captura das imagens de forma manual retirando o peixe da água ou colocando-o em um aquário transparente, é algo muito trabalhoso. O desenvolvimento de um esquema de captura de imagens dentro da água com uma câmera aquática, aumentaria a velocidade de aquisição das imagens e resultaria em imagens que retratasse melhor as condições do ambiente dos peixes, resultando em um classificador mais assertivo para imagens do mundo real.

O algoritmo desenvolvido para gerar novas imagens, pode ser melhorado para considerar as particularidades das imagens de cada classe. Por exemplo, agrupar as imagens que são parecidas entre si, as que tem o mesmo fundo por exemplo e gerar mais imagens de um grupo que tem um número menor de amostras do que outro que predomina na classe, assim, equilibrando as características presentes na base de dados.

Referências

ABADI, M. et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. Citado na página 29.

ANDRADE, D. R.; YASUI, G. S. Manejo da reprodução natural e artificial e sua importância na produção de peixes no brasil. *Revista Brasileira de Reprodução Animal*, MINISTERIO DA AGRICULTURA, v. 27, n. 2, p. 166–172, 2003. Citado na página 1.

AZIZPOUR, H. et al. From generic to specific deep representations for visual recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. [S.l.: s.n.], 2015. p. 36–45. Citado 2 vezes nas páginas 15 e 21.

BISHOP, C. M. et al. *Neural networks for pattern recognition*. [S.l.]: Oxford university press, 1995. Citado na página 4.

BRADLEY, A. P. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, Elsevier, v. 30, n. 7, p. 1145–1159, 1997. Citado na página 19.

BRANCO, P.; TORGO, L.; RIBEIRO, R. A survey of predictive modelling under imbalanced distributions. *arXiv preprint arXiv:1505.01658*, 2015. Citado 4 vezes nas páginas , 17, 18 e 19.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1251–1258. Citado na página 21.

CHRISTENSEN, J. H. et al. Detection, localization and classification of fish and fish species in poor conditions using convolutional neural networks. In: IEEE. *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*. [S.l.], 2018. p. 1–6. Citado 3 vezes nas páginas 16, 20 e 27.

DEEP, B. V.; DASH, R. Underwater fish species recognition using deep learning techniques. In: IEEE. *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*. [S.l.], 2019. p. 665–669. Citado 4 vezes nas páginas 2, 20, 27 e 37.

Deng, J. et al. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2009. p. 248–255. Citado na página 16.

DINIZ, N. M.; HONORATO, C. A. Algumas alternativas para diminuir os efeitos do estresse em peixes de cultivo-revisão. *Arquivos de ciencias veterinarias e zoologia da UNIPAR*, v. 15, n. 2, 2012. Citado na página 1.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016. Citado 3 vezes nas páginas , 13 e 15.

FAO. The state of world fisheries and aquaculture 2020. Sustainability in action, 2020. Citado na página 1.

FERREIRA, J. G. et al. Progressing aquaculture through virtual technology and decision-support tools for novel management. In: *Global Conference on Aquaculture 2010*. [S.l.: s.n.], 2012. Citado na página 1.

FISHER, R. B. et al. *Fish4Knowledge: collecting and analyzing massive coral reef fish video data*. [S.l.]: Springer, 2016. v. 104. Citado na página 25.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 5 vezes nas páginas , 9, 10, 11 e 12.

GORI, M.; TESI, A. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 14, n. 1, p. 76–86, 1992. Citado na página 7.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado na página 21.

JOLY, A. et al. Lifeclef 2015: multimedia life species identification challenges. In: SPRINGER. *International Conference of the Cross-Language Evaluation Forum for European Languages*. [S.l.], 2015. p. 462–483. Citado 2 vezes nas páginas 1 e 2.

KHALIFA, N. E. M.; TAHA, M. H. N.; HASSANIEN, A. E. Aquarium family fish species identification system using deep neural networks. In: SPRINGER. *International Conference on Advanced Intelligent Systems and Informatics*. [S.l.], 2018. p. 347–356. Citado 2 vezes nas páginas 1 e 20.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado 2 vezes nas páginas 8 e 29.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *NIPS*. [S.l.: s.n.], 2012. Citado 2 vezes nas páginas 5 e 15.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Citado 2 vezes nas páginas 5 e 13.

LECUN, Y.; BENGIO, Y. et al. Convolutional networks for images, speech, and time series. 1997. Citado 2 vezes nas páginas 13 e 14.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural computation*, MIT Press, v. 1, n. 4, p. 541–551, 1989. Citado na página 14.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, November 1998. Citado na página 15.

MAIONE, C. et al. Balanceamento de dados com base em oversampling em dados transformados. Universidade Federal de Goiás, 2020. Citado na página 20.

MATHUR, M. et al. Crosspooled fishnet: transfer learning based fish species classification model. *Multimedia Tools and Applications*, Springer, v. 79, n. 41, p. 31625–31643, 2020. Citado 2 vezes nas páginas 16 e 37.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 3.

MENG, L.; HIRAYAMA, T.; OYANAGI, S. Underwater-drone with panoramic camera for automatic fish recognition based on deep learning. *Ieee Access*, IEEE, v. 6, p. 17880–17886, 2018. Citado 2 vezes nas páginas 20 e 27.

MONTALBO, F. J. P.; HERNANDEZ, A. A. Classification of fish species with augmented data using deep convolutional neural network. In: IEEE. *2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)*. [S.l.], 2019. p. 396–401. Citado na página 20.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Icml*. [S.l.: s.n.], 2010. Citado na página 5.

NWANKPA, C. et al. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018. Citado na página 5.

OLSVIK, E. et al. Biometric fish classification of temperate species using convolutional neural network with squeeze-and-excitation. In: WOTAWA, F. et al. (Ed.). *Advances and Trends in Artificial Intelligence. From Theory to Practice*. Cham: Springer International Publishing, 2019. p. 89–101. Citado 5 vezes nas páginas 16, 20, 26, 27 e 37.

OQUAB, M. et al. Learning and transferring mid-level image representations using convolutional neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2014. Citado na página 16.

PROVOST, F.; FAWCETT, T.; KOHAVI, R. The case against accuracy estimation for comparing induction algorithms 1998. In: *Proceedings of the 15th international conference on machine learning ICML-98 Morgan Kaufmann. San Mateo, CA*. [S.l.: s.n.], 1998. Citado na página 18.

QIN, H. et al. Deepfish: Accurate underwater live fish recognition with a deep architecture. *Neurocomputing*, Elsevier, v. 187, p. 49–58, 2016. Citado 3 vezes nas páginas 1, 20 e 37.

QIU, C. et al. Improving transfer learning and squeeze-and-excitation networks for small-scale fine-grained fish image classification. *IEEE Access*, IEEE, v. 6, p. 78503–78512, 2018. Citado 4 vezes nas páginas 1, 16, 20 e 27.

RASCHKA, S. *Python machine learning*. [S.l.]: Packt publishing ltd, 2015. Citado na página 29.

RATHI, D.; JAIN, S.; INDU, S. Underwater fish species classification using convolutional neural network and deep learning. In: IEEE. *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)*. [S.l.], 2017. p. 1–6. Citado 2 vezes nas páginas 20 e 37.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 4.

RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. Citado 2 vezes nas páginas 7 e 8.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986. Citado na página 6.

SALMAN, A. et al. Fish species classification in unconstrained underwater environments based on deep learning. *Limnology and Oceanography: Methods*, Wiley Online Library, v. 14, n. 9, p. 570–585, 2016. Citado na página 20.

SANTOS, A. A. dos; GONÇALVES, W. N. Improving pantanal fish species recognition through taxonomic ranks in convolutional neural networks. *Ecological Informatics*, Elsevier, v. 53, p. 100977, 2019. Citado 3 vezes nas páginas 16, 20 e 27.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks*, Elsevier, v. 61, p. 85–117, 2015. Citado 2 vezes nas páginas 3 e 6.

SHAFAIT, F. et al. Fish identification from videos captured in uncontrolled underwater environments. *ICES Journal of Marine Science*, Oxford University Press, v. 73, n. 10, p. 2737–2746, 2016. Citado 2 vezes nas páginas 1 e 20.

SIDDIQUI, S. A. et al. Automatic fish species classification in underwater videos: exploiting pre-trained deep neural network models to compensate for limited labelled data. *ICES Journal of Marine Science*, Oxford University Press, v. 75, n. 1, p. 374–389, 2018. Citado 2 vezes nas páginas 16 e 20.

SOKOLOVA, M.; JAPKOWICZ, N.; SZPAKOWICZ, S. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In: SPRINGER. *Australasian joint conference on artificial intelligence*. [S.l.], 2006. p. 1015–1021. Citado na página 17.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. Citado na página 11.

SUN, X. et al. Fish recognition from low-resolution underwater images. In: IEEE. *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. [S.l.], 2016. p. 471–476. Citado na página 16.

SUN, Y.; WONG, A. K.; KAMEL, M. S. Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence*, World Scientific, v. 23, n. 04, p. 687–719, 2009. Citado 2 vezes nas páginas 17 e 18.

SURYANARAYANA, I. et al. Neural networks in fisheries research. *Fisheries Research*, Elsevier, v. 92, n. 2-3, p. 115–139, 2008. Citado na página 4.

SZEGEDY, C. et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2017. v. 31, n. 1. Citado na página 21.

SZYMAK, P.; GASIOROWSKI, M. Using pretrained alexnet deep learning neural network for recognition of underwater objects. *NAŠE MORE: znanstveno-stručni časopis za more i pomorstvo*, Sveučilište u Dubrovniku, v. 67, n. 1, p. 9–13, 2020. Citado na página 16.

TAMOU, A. B. et al. Transfer learning with deep convolutional neural network for underwater live fish recognition. In: IEEE. *2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS)*. [S.l.], 2018. p. 204–209. Citado 7 vezes nas páginas 2, 16, 20, 21, 24, 27 e 37.

TAN, M.; LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2019. p. 6105–6114. Citado na página 21.

YANG, X. et al. Deep learning for practical image recognition: Case study on kaggle competitions. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2018. p. 923–931. Citado 2 vezes nas páginas 20 e 27.

ZHENG, Z. et al. Fish recognition from a vessel camera using deep convolutional neural network and data augmentation. In: IEEE. *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*. [S.l.], 2018. p. 1–5. Citado 4 vezes nas páginas 2, 20, 25 e 27.