MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

ANÁLISE DE TÉCNICAS DE APRENDIZADO PROFUNDO E DE APRENDIZADO DE MÁQUINA TRADICIONAL PARA CLASSIFICAÇÃO DE IMAGENS DE DRONES

Hércules Carlos Dos Santos Pereira

Relatório de iniciação científica do programa PIBIC, orientados pelo
Dr. Valdivino Alexandre de Santiago Júnior
e pelo Dr. Elcio Hideiti Shiguemori.

INPE
São José dos Campos
Dezembro de 2022

RESUMO

Os veículos aéreos não tripulados (VANTs), ou drones, são usados em diversas aplicações tais como segurança, sensoriamento remoto, em âmbito militar, entre outros. A classificação automática de imagens obtidas por drones é importante para, por exemplo, melhorar a autonomia desses sistemas no que tange à resposta a desastres e situações de emergências em áreas de difícil acesso. O objetivo desta pesquisa é fazer uma análise de técnicas de aprendizado profundo e de aprendizado de máquina tradicional para classificação de imagens de drones. Com isso, espera-se determinar se é mais viável usar uma rede neural profunda apenas como extrator de características das imagens, deixando a tarefa de classificação para algoritmos de aprendizado de máquina tradicional, ou se é mais relevante usar uma rede neural profunda tanto como extrator de características mas também como classificador. Em uma primeira etapa, foi considerada uma abordagem de classificação multiclasse (4 classes), onde foi realizado um processo de aumento de dados (*data augmentation*) sobre o conjunto de treinamento. Portanto, fez-se a avaliação das seguintes redes neurais convolucionais (CNNs) como extratores de características: *EfficientNetB0*, *EfficientNetB7*, *EfficientNetV2L*, *MobileNet, NASNetLarge* e *VGG*-16. Como classificadores, foram usados: regressão logística, rede neural artificial e *support vector machines* (SVMs). Os resultados demonstram que o extrator *EfficientNetB0* junto ao classificador SVM tiveram os melhores resultados, se comparados com as demais técnicas de extrator e classificador, atingindo acerto de até 99,6% nas métricas, Acurácia, F1-*Score*, *Precision* , *Recall* e Tempo de execução.

Palavras-chave: Drones, Inteligência artificial, Aprendizado de máquina, Aprendizado profundo, Redes neurais convolucionais.

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Os veículos aéreos não tripulados (VANTs), também conhecidos como drones, são expostos a cenários de voo com diferentes terrenos, alturas e luminosidade. Existem diferentes tipos de aeronaves, tais como as de asa rotativa, asa fixa e híbridos (Ross, Lorena e Shiguemori, 2016).

Nas aplicações de sensoriamento remoto, a variação de cenários de voo dificulta o processo de automatização da classificação das imagens, gerando uma variação de imagens com características distintas pelas condições de voo. Sendo assim, não há um algoritmo único de visão computacional ou inteligência artificial (IA) para a adequada classificação autônoma das imagens (Ross, Lorena e Shiguemori, 2016). Um algoritmo pode ser melhor em um cenário de voo mas pode não ser o melhor em outro cenário de voo (Fornari, Santiago Junior e Shiguemori, 2018).

Aprendizado profundo (deep learning - DL) e seus modelos/técnicas/algoritmos relacionados, tais como redes neurais convolucionais profundas (deep convolutional neural networks - CNNs) (Ghanbari et al., 2021), são a principal causa do grande interesse em IA nos últimos anos. A variedade de uso de DL é significativa, incluindo VANTs e aplicações de sensoriamento remoto (Mohammadi, Watson e Wood, 2019).

As CNNs são interessantes porque podem extrair, automaticamente, as características da imagem (extrator), e também classificar as imagens automaticamente. No entanto, alguns estudos usam a CNN apenas como extrator de características, deixando a tarefa de classificação em si para outros algoritmos tradicionais de aprendizado de máquina (machine learning - ML), tais como random forests (RF) (Ho, 1995) e support vector machines (SVMs) (Boser, Guyon e Vapnik, 1992). A motivação para fazer isso é porque classificadores tradicionais, tais como os citados acima, tiveram grande sucesso em áreas como sensoriamento remoto (Guo et al., 2016). Neste contexto, visto que existem vários modelos de CNNs e algoritmos tradicionais de ML disponíveis, é importante investigar e realizar experimentos que possam subsidiar a decisão sobre a combinação de extrator e classificador mais adequada para classificação automática de imagens no espectro visível obtidas por drones.

O objetivo desta pesquisa é investigar a classificação de imagens obtidas por drones utilizando técnicas de aprendizado profundo e de aprendizado de máquina tradicional. Em

uma primeira etapa, foram desenvolvidos códigos usando a linguagem Python (Python, 2022), as bibliotecas Pandas (Pandas, 2008), o framework Tensorflow (TensorFlow, 2015), Keras (Keras, 2021) e scikit-learn (scikit-learn, 2021). Para essa etapa, foram usadas as CNNs EfficientNetB0 (Tan e Le, 2020), EfficientNetB7 (Tan e Le, 2020), EfficientNetV2L (Le e Tan, 2021), MobileNet (Howard et al., 2017), NASNetLarge (Zoph et al, 2018) e VGG-16 (Szegedy et all, 2016), como extratores de características, e os algoritmos tradicionais de ML, rede neural artificial, regressão logística e SVM.

Já em uma segunda etapa, foram propostas duas abordagens, sendo a primeira com a combinação de extrator CNN e classificador CNN. Na segunda abordagem, se utilizou de extratores CNN e Classificadores ML tradicionais, a fim de comparar com resultados obtidos anteriormente. Neste experimento as mesmas CNNs utilizadas para a extração de características das imagens fizeram o papel de classificador.

2 CRONOGRAMA DE ATIVIDADES E ETAPAS CONCLUÍDAS

Assim como disposto no "formulário de proposta de bolsa PIBIC", a metodologia (conjunto de atividades) para atingir os objetivos do projeto é descrita a seguir:

1. Obtenção das imagens dos drones para compor os conjuntos de dados. Tais imagens já estão disponíveis em repositório do IEAV;
2. Seleção das CNNs e dos algoritmos tradicionais de ML para fazer parte da avaliação;
3. Implementação/adaptação das CNNs e dos algoritmos tradicionais de ML em *Keras*, *Tensorflow* e *Scikit-learn*;
4. Avaliação rigorosa do desempenho das técnicas para identificar quais são as mais adequadas. Configuração 1: Extrator (CNN) + Classificador (ML tradicional);
5. Avaliação rigorosa do desempenho das técnicas para identificar quais são as mais adequadas. Configuração 2: Extrator + Classificador (CNN), Extrator (CNN) + Classificador (ML tradicional);
6. Implementação em computador de drone e realização de voos;
7. Submissão de artigo para conferência ou revista especializada na área de pesquisa.

Em termos detalhados, as atividades foram definidas como sendo nove como mostra a Tabela 1 a seguir. Na Tabela 1, a coluna **Previsão** mostra a porcentagem prevista para a realização da atividade, e a coluna **Realização** mostra a porcentagem que foi realizada da atividade, considerando o período a que se refere este relatório (1 de setembro 2022 a 31 de dezembro de 2022). Considerando o período a que se refere este relatório parcial, e conforme mostrado na Tabela 1, apenas as atividades de A1 a A5 deveriam ter sido desenvolvidas, sendo a atividade A5 este relatório parcial. Portanto, todas as atividades (A1 a A5) previstas no período a que se refere este relatório foram 100% concluídas. Além disso, mesmo que as atividades A6 e A7 estivessem sido previstas para serem realizadas após o período a que se refere este relatório parcial, as realizações das mesmas foram antecipadas, sendo que a atividade A6 foi 50% concluída e a atividade A7 foi também 50% concluída. As atividades A8 e A9 não estavam previstas para serem realizadas no período a que se refere este relatório parcial.

A seguir, são mostrados os detalhamentos dos desenvolvimentos das atividades.

Tabela 1 – Atividades previstas e realizadas

Atividade da Metodologia	**Previsão**	**Realização**
A1. Obtenção das imagens dos drones para compor os conjuntos de dados;	100%	100%
A2. Selecionar e implementar as Redes Neurais Convolucionais e os algoritmos de Aprendizado de Máquina tradicionais para serem aplicados sobre as imagens. Implementar extrator e classificador;	100%	100%
A3. Aplicar/adaptar Rede Neural Convolucional como extrator de características apenas, e classificar os resultados com os algoritmos de Aprendizado de Máquina tradicionais;	100%	100%
A4. Ajustar os parâmetros das técnicas de Aprendizado de Máquina;	100%	100%
A5. Escrever relatório parcial;	100%	100%
A6. Aplicar/adaptar Rede Neural Convolucional como extrator de características e classificador, e algoritmos de Aprendizado de Máquina tradicionais como classificador;	0%	50%
A7. Comparar desempenho das abordagens feitas nas atividades A3 e A6 por meio de uma avaliação rigorosa;	0%	50%

A8. Embarcar o sistema em um computador de bordo de um drone e realizar voo;	0%	0%
A9. Submeter artigo para conferência e/ou workshop e/ou simpósio na área de Aprendizado de Máquina ou Inteligência Artificial, e elaborar relatório final de atividades.	0%	0%

3 MATERIAIS E MÉTODOS

O desenvolvimento do projeto consistiu no uso da combinação de Rede Neural Convolucional como extrator de característica e as técnicas de aprendizagem de máquina como classificador. A seguir é apresentada uma lista dos métodos e materiais empregados nesta pesquisa.

3.1 Métodos

Os métodos empregados na pesquisa foram a aprendizagem de máquina, redes neurais artificias, rede neural convolucional, regressão logística e máquina de vetores de suporte.

3.1.1 Classificação

A classificação é um método realizado por algoritmos de aprendizado de máquina e *deep learning*, tem o objetivo de descobrir a categoria a qual um determinado conjunto pertence. Um exemplo desta classificação é o método usado nesta pesquisa, onde são extraídas as características de uma imagem, é aplicado a classificação sobre estas características para descobrir a qual categoria essas informações pertencem. Nesta pesquisa as categorias selecionadas foram: árvore, casa, carro e prédio. Na Figura 1 é ilustrada algumas amostras de imagem de casa, árvore, carro e prédio.

Figura 1 – Amostra de classe

Fonte: Autoria própria, 2020.

3.1.2 Aprendizado de Máquina

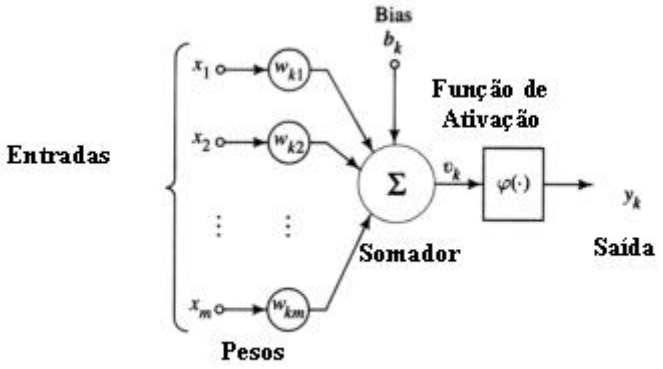
Machine learning ou aprendizagem de máquina é uma ramificação da área de inteligência artificial. O método também é usado na análise de dados e se baseia na ideia de que um sistema ou máquina tem capacidade de aprender por dados, ou seja, o modelo se adapta e aprende a analisar padrões. Alguns dos tipos de aprendizado de máquina são: o aprendizado supervisionado, não supervisionado, semi-supervisionado e por reforço segundo (Ludermir, 2021).

O método utilizado nos experimentos deste trabalho foi o aprendizado supervisionado, nessa abordagem se tem um conjunto de dados apresentados como entrada, também é apresentada o resultado esperado para a saída (Ludermir, 2021).

3.1.3 Rede Neural Artificial

As RNA se baseiam em neurônios biológicos. Uma rede neural artificial (Rosenblatt, 1958) é composta por vários neurônios artificiais, modelos matemáticos. Eles possuem entradas, pesos, processamento e função de ativação que será enviada à saída. As entradas são as características dos problemas. Ela é multiplicada pelo seu respectivo peso e o somador realiza o processamento, sendo a soma das multiplicações das entradas vezes o peso. A função de ativação é responsável por ativar ou inibir a intensidade do neurônio que serve como o resultado na saída. Na Figura 2 - Modelo de Neurônio Artificial é ilustrado um neurônio artificial.

Figura 2 - Modelo de Neurônio Artificial



Fonte: Saulo Popov Zambiasi, 2008.

As redes neurais artificiais são compostas por vários neurônios e camadas. As redes neurais profundas possuem diversas camadas e diversos neurônios. É importante definir a topologia mais adequada para um problema e os dados, com isso deve-se utilizar hiper-parâmetros adequados como o número de neurônios, camadas e funções de ativação (Zambiasi, 2008).

3.1.4 Transferência de aprendizado

Também conhecida como *Transfer learning*, é um modelo de aprendizagem de máquina de transferência de aprendizado. Este treinamento consiste na seguinte maneira, é implementado um modelo de rede neural convolucional pré-treinado com conjunto de imagens como o *ImageNet* (ImageNet, 2021). Então o modelo já possui conhecimento armazenado o que resulta em um treinamento mais rápido e com uma taxa de acerto mais alta.

A Figura 3 ilustras o código implementado para o carregamento do conjunto de dado.

Figura 3 - Carregamento do conjunto de dado

```
# Lista com o caminho de cada imagem
#nome_imagens = os.listdir(caminhos+classes[0])


# lista para salvar o caminho de cada imagem
# caminho_imagens = []

nome_img_predio = []
nome_img_carro = []
nome_img_casa = []
nome_img_arvore = []

nomes_img_completo = []


# Separa cada classe em um lista separada
for clas in range(len(classes)):
  #nome_imagens = os.listdir(classes[0])
  nome_imagens = os.listdir(path_train+classes[clas])
  nome_imagens_ordenado = sorted(nome_imagens)
  if clas == 0:
    for i in nome_imagens_ordenado:
      nome_img_predio.append(paths_t[0]+i)
  elif clas == 1:
    for i in nome_imagens_ordenado:
      nome_img_carro.append(paths_t[1]+i)
  elif clas == 2:
    for i in nome_imagens_ordenado:
      nome_img_casa.append(paths_t[2]+i)
  else:
    for i in nome_imagens_ordenado:
      nome_img_arvore.append(paths_t[3]+i)
```

Fonte: Autoria própria (2022).

Inicialmente é armazenado o caminho onde se encontro o conjunto de imagem, utiliza as variáveis nome_img_predio, nome_img_carro, nome_img_casa e nome_img_arvore para armazenar o caminho completo de cada imagem de acordo com sua respectiva classe. Isso é demonstrado na Figura 3.

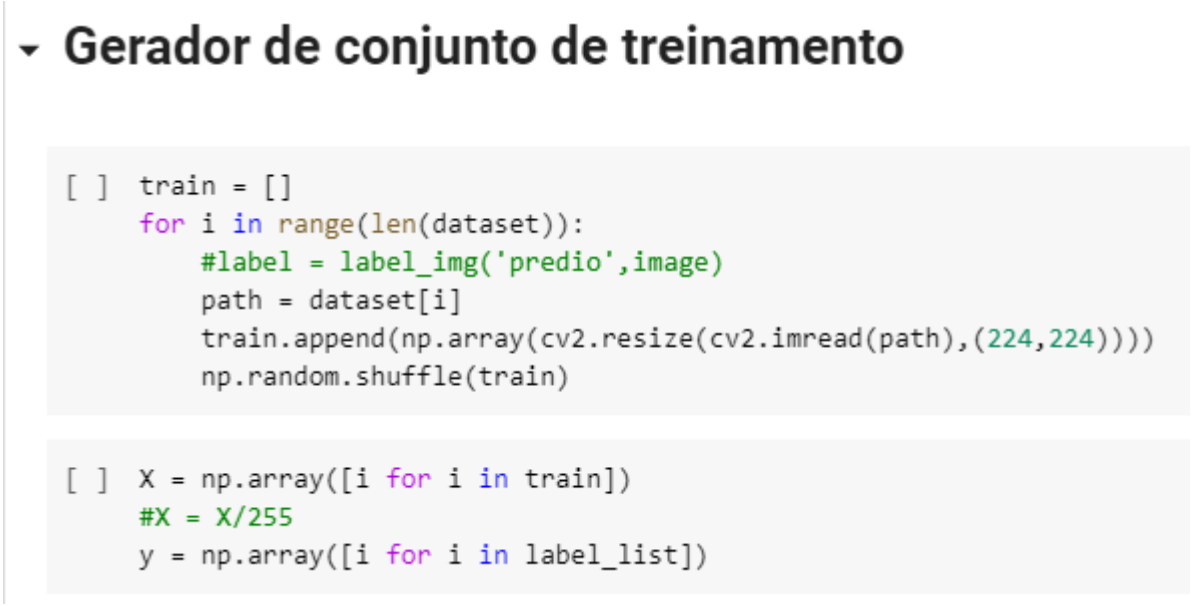
Figura 4 - Rótulos das imagens para o conjunto de treinamento

```
# Rotulos das imagens conjunto de Treinamento
len_conjunto = [524,850,800,840]

'''
Predio [0]
Carro  [1]
Casa   [2]
Arvore [3]
'''

label_list = []
for i in len_conjunto:
  for y in range(i):
    if i == 524:
      label_list.append(0)
    elif i == 850:
      label_list.append(1)
    elif i == 800:
      label_list.append(2)
    elif i == 840:
      label_list.append(3)
```

Fonte: Autoria própria (2022).

Os rótulos servem para que o modelo entenda a qual classe uma matriz de imagem presente no conjunto de treinamento. Conforme ilustrado na Figura 4 os rótulos são representados a partir de uma numeração de 0 a 3 onde 0 prédio, 1 carro, 2 casa e 3 arvore. A lógica do código presente na figura é o seguinte, o loop irá percorrer o tamanho do conjunto de imagem, por exemplo prédio e adicionar o valor 0 ao array Label_list até atingir a quantidade 524 que é a quantidade de imagens de prédio. Depois é aplicado um normalizador resultando no seguinte *array numpy* [0, 1, 0, 0] uma representação binaria da classe predio.

Figura 5 - Gerador de conjunto de treinamento



Fonte: Autoria própria (2022).

Neste documento será apresentado somente a geração do conjunto de treinamento já que tanto o conjunto de treinamento quanto de teste usa o mesmo gerador, a única diferença entre eles é o caminho em que se encontra os arquivos. Sobre o *dataset* composto caminho completo para cada imagem é pego o path = caminho da imagem, o *array train* armazena um *array numpy* contendo a imagem lida e redimensionada em 224 x 224 por fim o *array train* é randomizado isso é ilustrado na . A variável X armazena um *array numpy* com todo o conjunto de treinamento. A variável y armazena um *array* com os rótulos *"labels"* de cada imagem.

Figura 6 - Criação do conjunto de treinamento

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_val = train_test_split(X, y, test_size = 0.2, random_state=42)
```

Fonte: Autoria própria (2022).

O conjunto é gerado e normalizado por uma função já implementada pela biblioteca *scikit-learning* a *train_test_split*, como é ilustrado na Figura 6.

Figura 7 - Carregamento o modelo VGG-16

```
[ ]  ## Carregar o modelo VGG16
     base_model = VGG16(weights="imagenet", include_top=False, input_shape=X_train[0].shape)
```

Fonte: Autoria própria (2022).

O modelo é carregado pela função VGG16, com os pesos pré-treinado com o banco de imagens *ImageNet*, e o tamanho da entrada do modelo com as mesmas dimensões que a primeira amostra do conjunto de treinamento isso é ilustrado na Figura 7.

Figura 8 - Criação do novo modelo com *transfer learning* VGG-16

```
#add our layers on top of this model
from tensorflow.keras import layers, models
from tensorflow.keras.layers.experimental.preprocessing import Rescaling

flatten_layer = layers.Flatten()
dense_layer_1 = layers.Dense(50, activation='relu')
dense_layer_2 = layers.Dense(20, activation='relu')
prediction_layer = layers.Dense(4, activation='softmax')


model = models.Sequential([
    base_model,
    flatten_layer,
    dense_layer_1,
    dense_layer_2,
    prediction_layer
])
```

```
[ ]  from tensorflow.keras.callbacks import EarlyStopping

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

Fonte: Autoria própria (2022).

Depois o novo modelo é implementado para receber os dados pré-treinados pelo modelo anterior, o “model” recebe o modelo sequencial com a camada top sendo composta pelo modelo VGG-16 que foi pré-treinado junto as novas camadas que foram criadas para o novo modelo como demonstrado na Figura 8.

Figura 9 - Treinamento do modelo com transfer leaning

```
[ ] from tensorflow.keras.callbacks import EarlyStopping
    es = EarlyStopping(monitor='val_accuracy', mode='max', patience=4,  restore_best_weights=True)

    model.fit(train_ds, train_labels, epochs=50, validation_split=0.2, batch_size=32, callbacks=[es])

    Epoch 1/50
    61/61 [==============================] - 25s 251ms/step - loss: 3.3952 - accuracy: 0.2578 - val_loss: 2.1142 - val_accuracy: 0.2650
    Epoch 2/50
    61/61 [==============================] - 16s 256ms/step - loss: 1.6625 - accuracy: 0.3532 - val_loss: 1.7397 - val_accuracy: 0.3043
    Epoch 3/50
    61/61 [==============================] - 13s 211ms/step - loss: 1.2869 - accuracy: 0.4378 - val_loss: 1.7269 - val_accuracy: 0.2981
    Epoch 4/50
    61/61 [==============================] - 13s 208ms/step - loss: 1.1234 - accuracy: 0.5176 - val_loss: 1.7489 - val_accuracy: 0.2857
    Epoch 5/50
    61/61 [==============================] - 12s 206ms/step - loss: 1.0204 - accuracy: 0.5752 - val_loss: 2.0043 - val_accuracy: 0.2961
    Epoch 6/50
    61/61 [==============================] - 13s 206ms/step - loss: 0.9860 - accuracy: 0.5887 - val_loss: 1.8736 - val_accuracy: 0.3023
    <keras.callbacks.History at 0x7f513665d3d0>
```

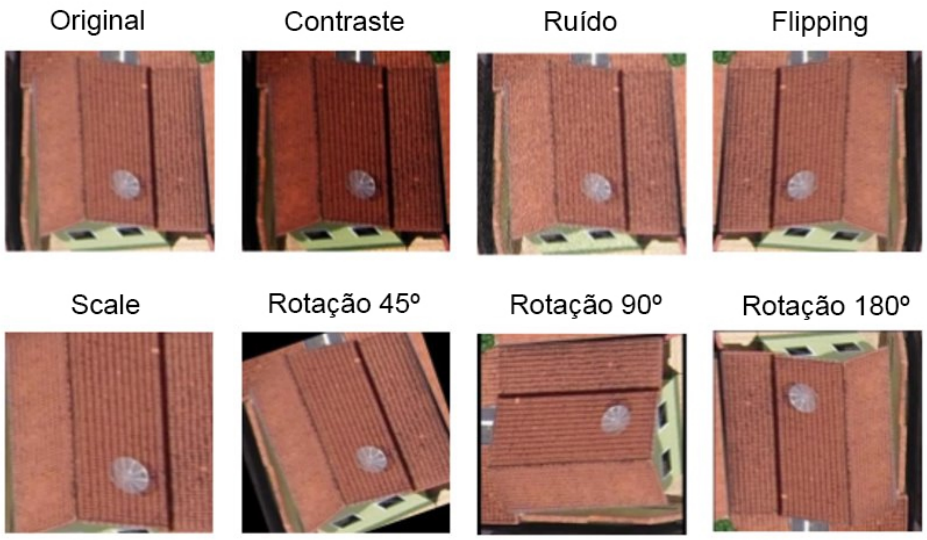
Fonte: Autoria própria (2022).

Por último é selecionado as métricas para avaliação do desempenho dos modelos como Loss, accuracy. O modelo é treinado com a função “fit”, que recebe os parâmetros de conjunto de treinamento, rótulos, épocas de treinamento, divisão de conjunto de treinamento, tamanho do lote de treinamento e as métricas de avaliação. A Figura 9 demostra o processo que foi citado anteriormente.

3.1.5 Aumento de Dados

De modo geral, o treinamento eficaz de redes neurais artificiais requer muitos dados. No caso de poucos dados, as técnicas de aumento de dados ou *Data Augmentation* são indicadas (Perez e Wang, 2017). A Figura 10 - Técnica de aumento de dados ilustra o resultado da aplicação das técnicas de aumento de dados em uma imagem. O método usado em imagem consiste em aplicar filtros e transformações sobre o conjunto de imagens, tais como: rotacionar, transformar, aplicar ruídos, ampliar imagem, escurecer e destorcer (Perez e Wang, 2017). No contexto desta pesquisa foi aplicado esta técnica sobre o conjunto de treinamento com 146 imagens que resultou em 3005 imagens.

Figura 10 - Técnica de aumento de dados

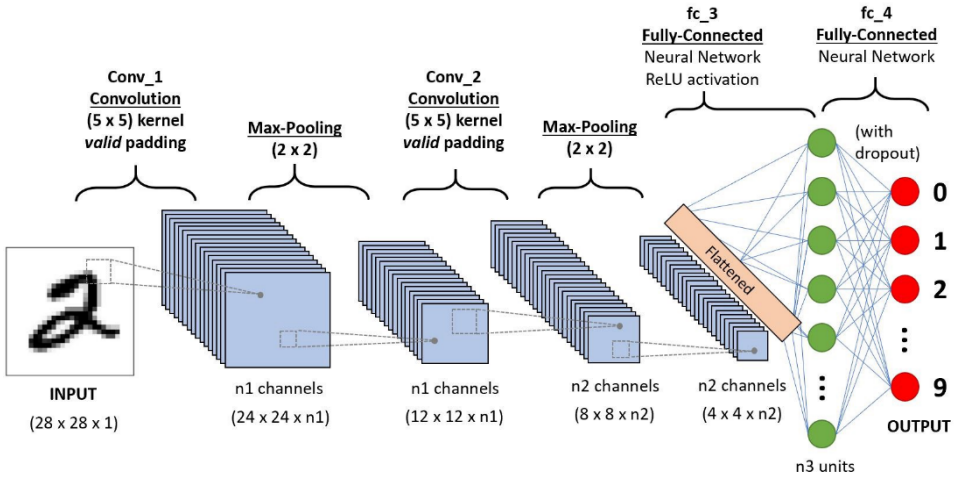


Fonte: Autoria própria (2021).

3.1.6 Rede Neural Convolucional Como Extrator de Características

Antes de abordarmos a extração de características é importante entender sobre, as funcionalidades básicas de uma rede neural convolucional ou CNN (Berticelli, 2022). A convolução é uma representação simplificada da imagem de entrada após a aplicação de uma técnica de *pooling*, essa técnica também garante saber a posição em que se encontra um objeto de interesse. Uma CNN (Berticelli, 2022) é um algoritmo de *deep learning*, que possui a capacidade de aplicar filtros para a extração de linhas, curvas e bordas a cada camada, os filtros são transformados em um mapa de características. Por último, esse mapa de característica é passado para uma rede neural artificial realizar uma classificação, segmentação ou detecção.

Figura 11 - Arquitetura básica de uma Rede Neural Convolucional



Fonte: Kenji, (2019).

A Figura 11 - Arquitetura básica de uma Rede Neural Convolucional ilustra uma CNN (Berticelli, 2022), onde recebe como entrada a figura de um número 2, a rede aplica filtros convolucionais que obtém um mapa de características e depois o *Max-pooling* é usado para localizar a posição do objeto. Esse modelo faz o processo convolução e depois aplica o *Max-pooling* 2 vezes, por fim é usado uma rede neural para classificar as informações extraídas.

O *EfficientNetB0* é uma técnica de aprendizado profundo treinada com 1000 classes do banco conjunto *ImageNet* (ImageNet, 2021). Sua arquiteta foi desenvolvida para ter o melhor desempenho usando o redimensionamento de profundidade, largura e resolução da rede. O modelo foi proposto para o redimensionamento dos modelos da família *EfficientNet*.

A *EfficientNet* é uma série de algoritmo de aprendizagem profunda da denominada família *EfficientNet*, neste trabalho foram utilizados os modelos *EfficientNetB0* (Tan e Le, 2020) e *EfficenteNetB7*(Tan e Le, 2020). Esses modelos foram pre-treinada com 1000 classes do banco conjunto *ImageNet* (ImageNet, 2021). O *EfficientNetB0* (Tan e Le, 2020) Sua arquiteta foi desenvolvida para ter o melhor desempenho usando o redimensionamento de profundidade, largura e resolução da rede. O modelo foi proposto para o redimensionamento dos modelos da família *EfficientNet*.

O *EfficientNetV2L* (Le e Tan, 2021) é o modelo da família *EfficienteNet* que possui a maior velocidade em seu treinamento e eficiência de parâmetros para o treinamento (Le e Tan, 2021). O modelo é 6,8 vezes menor do que os modelos da última geração, com uma precisão de 87,3% de acerto sendo pré-treinado com o conjunto de imagem ImageNet.

O *NASNetLarge* é o modelo de rede neural profunda para a classificação de imagens, neste modelo foi utilizado grande conjunto de imagem a partir do banco de imagem ImageNet. O diferencial deste modelo é a utilização de cópia das células do conjunto de imagem, em cada camada de convolução e a utilização de uma nova técnica de regularização denominada de "ScheduledDropPath" (Zoph et al, 2018) que serve para melhor a capacidade de generalização do modelo. Este modelo atinge uma taxa de erro de 2,4% e uma precisão de 74% de acerto.

O MobileNet é o modelo de rede neural convolucional para classificação de imagem desenvolvida para ser eficiente (Tan e Le, 2020). Este modelo foi desenvolvido para ser eficiente rodando em aplicações de dispositivos moveis e sistemas embarcados. Em sua arquitetura o modelo consegue escolher automaticamente o melhor dimensionamento de sua arquitetura para se adequar a cada aplicação.

As técnicas *VGG*-16 é uma rede neural profunda utilizada para reconhecimento de imagens. Desenvolvido pelo grupo *Visual Geometry Group* da Universidade de Oxfod, tal modelo também foi treinado com o banco de imagens *ImageNet* (ImageNet, 2021). A CNN *VGG*-16 (Szegedy et all, 2016) é composta por 5 camada denominada Conv, a cada camada se

tem um *Max-pooling* com exceção da Conv1 já que recebe como entra a imagem. As camadas Conv1 e Conv2 são compostas por 2 camadas convolucionais 3 × 3, as camadas Conv3 a 5 são compostas por 3 camadas também com filtro 3 × 3.

3.1.7 Regressão Logística

O algoritmo de regressão logística é uma técnica estatística com intuito de produzir, por meio de um conjunto de treinamento, um modelo para prever valores por meio de classes. Ou seja, verificar a influência de uma variável de interesse sobre uma variável *y* e assim, criar um modelo matemático capaz de prever valores de *y* com base em novos valores de variáveis *x*. Nesta pesquisa é usado a classificação de multiclasse, onde foram treinados o algoritmo para classificar 4 classes considerando os cenários de voo. O algoritmo traça retas para buscar a melhor separação entre as informações.

3.1.8 Máquina de Vetores de Suporte - SVM

A SVM é um algoritmo de aprendizado de máquina, usado para classificação e regressão. O objetivo do algoritmo é achar a separação entre as classes, esta separação das classes é chamada de hiperplano. Este algoritmo consegue resolver problemas que sejam linearmente separáveis.

3.2 Materiais

A seguir são ilustrados os materiais, usados na pesquisa, as imagens obtidas por drone, Tabela de características e métricas.

3.2.1 Imagens Obtidas por Drone

Um conjunto de imagens composto por vinte e cinco imagens de área urbana obtidos por uma aeronave remotamente pilotada (ARPs) com 5 cm de resolução, quatro mil de largura e três mil de comprimento.

Figura 12 - Imagens Obtidas por Drone

Fonte: Marielcio, (2020).

A Figura 12 mostra o exemplo de uma imagem obtida por voo de drone. O conjunto foi usado para extrair as classes, possuem vinte e cinco imagens com resolução de cinco centímetros, com dimensões de quatro mil de largura e três mil de altura. (Lacerda, Damião e Shiguemori, 2020). As classes foram extraídas através de recorte com técnicas de processamento de imagens, as classes recortadas foram de casa, carro, árvore e prédio.

3.2.2 Tabela de Características

O resultado da ativação da CNN sobre o conjunto de imagem é um vetor com 1000 colunas que representa a quantidade de classe usada no treinamento desta mesma CNN. Estes vetores são armazenados em um *Dataframe* usando a biblioteca Pandas (Pandas, 2008). Por fim é atribuído uma coluna que representa a classe com os seguintes códigos: prédio: 0, carro:1, casa: 2 e árvore: 3. Ao final, o *Dataframe* de treinamento é composto por 3005 linhas com 1001 colunas, e 147 linhas com 1001 colunas para o conjunto de teste.

Figura 13 - Tabela exemplo de extração de características

	N1	N2	N3	N4	N5
0	0.000110	0.000690	0.000115	0.000099	0.000293
1	0.000163	0.000479	0.000062	0.000062	0.000223
2	0.000078	0.000289	0.000393	0.000485	0.000705
3	0.000157	0.000083	0.000066	0.000054	0.000060
4	0.000077	0.000095	0.000044	0.000062	0.000049

5 rows × 1001 columns

Fonte: Google Colab (2022).

A Figura 13 ilustra algumas características extraídas pela CNN, as linhas da tabela representam as imagens e as colunas são as características extraídas a partir destas imagens.

3.2.3 Métricas

Para avaliar o desempenho da combinação de extrator de características e classificador, foram usadas as seguintes métricas: (CA), precisão (prec), *recall* (Rec), e F1-*score* (F1). O CA é a precisão da classificação dos objetos classificados corretamente, ou seja, é a acurácia. Precisão, *recall*, e F1-*score* são calculados como mostrados a seguir:

$$CA = \frac{VP + VN}{VP + VN + FP + FN}$$

$$\text{prec} = \frac{VP}{VP + FP}$$

$$Rec = \frac{VP}{VP + FN}$$

$$F1 = \frac{2 \times \text{prec} \times Rec}{\text{prec} + Rec}$$

Onde VP verdadeiro positivo, VN verdadeiro negativo, FP falso positivo, e FN falso negativo.

4 EXPERIMENTOS

Nesta pesquisa foram feitas duas abordagens com o mesmo objetivo, buscar a melhor combinação de extrator e classificador. Mas com métodos diferentes, na primeira abordagem se utilizou extrator CNN + Classificador ML e na segunda abordagem CNN como extrator e classificador. O uso destes métodos tem o intuito de aprimorar os conhecimentos na análise, tratamento de dados e o uso de diferentes técnicas de aprendizagem de máquina para classificar e prever valores. Isto garantindo um conhecimento iniciação sobre algoritmos e modelos de inteligência artificial. Em ambas as abordagens se utilizou as linguagens *Python*, as bibliotecas Pandas, *Tensorflow*, *Keras* e *scikit-learn*. O que distingue a segunda abordagem da primeira é a aplicação de diferentes métodos e algoritmos para a classificação das imagens.

O conjunto de dados selecionado para os experimentos é derivado de 20 imagens de voo de drone sobre áreas urbanas com dimensão de 3000 x 4000 pixels obtidas por uma câmera com resolução de 5 cm, a partir deste conjunto foram extraídas as classes: árvore, casa, carro e prédio por meio de recorte. Que resultou em um conjunto de 292 imagens, este conjunto foi dividido em treinamento e teste, assim, ambos ficaram com 146 imagens, é um conjunto considera pequeno para ser usados no treinamento. Para aumentar o conjunto foram utilizadas técnicas de aumento de dados que consistem em aplicar rotação em 90 e 180 graus, espelhamento da imagem, acrescentar ruídos e aumentado contraste, as combinações dessas técnicas resultou em 3005 imagens.

No ambiente do Google colaboratory (Google Colaboratory, 2021), foram desenvolvidos e executados os códigos de Extrato-CNN e Classificador-ML. Ambos os códigos foram escritos na linguagem Python e usando as bibliotecas *Tensorflow* (Tensorflow, 2015), Keras (Keras, 2021) e *scikit-learn* (scikit-learn, 2021) para usar as CNN já implementadas e os algoritmos de aprendizado de máquina. É passado o conjunto de imagem para CNN (Berticelli, 2022) extrair as características, o resultado é um vetor com 1000 colunas que representam a quantidade de classe em que o modelo foi pré-treinado. Usando a biblioteca Pandas (Pandas, 2008) é criado um *Dataframe* onde são armazenados o vetor com as características das imagens e uma coluna chamada classe que representa a classe à qual pertence essa imagem, depois deste processo o *Dataframe* salvo em formato CSV. Este processo é feito tanto para o conjunto de treinamento quanto para o conjunto de teste. Todas as CNN (Berticelli, 2022) foram treinadas com 1000 classes do conjunto de imagens

ImageNet (ImageNet, 2021). As características das imagens citadas representam a porcentagem em que a imagem de entrada pertencer às classes usadas no pré-treinamento.

Para que os dados possam ser usados no treinamento das técnicas de aprendizado de máquina é necessário separar e randomizar os dados usando uma função do *scikit-learn*. O treinamento das técnicas de ML é feito sobre o *Dataframe* de treinamento e a avaliação é feita sobre o conjunto de teste. Para calcular o desempenho, a coluna alvo do *Dataframe* teste é guardado em uma variável, depois sobre o conjunto de teste e realizar a previsão da coluna alvo (classe), considerando as métricas F1-*score*, *Precision* e *Recall*. Os resultados gerados por cada extrator + classificador são salvos em uma tabela para ser analisado o desempenho.

5 DESENVOLVIMENTO

No desenvolvimento será explicado os processos realizados na pesquisa de forma detalhada, os materiais, métodos empregados, análises e resultados obtidos.

5.1 Abordagem

A abordagem feita nesta pesquisa, foi o uso das redes neurais convolucionais para extrair as características das imagens, a partir destas informações foram aplicadas diferentes técnicas de aprendizado de máquinas para classificar a qual classe as informações pertencem. Neste experimento foram usadas as redes convolucionais: *EfficientNetB0, EfficientNetB7, EfficientNetV2L, MobileNet, NASNetLarge e VGG*-16 como extrator de características das imagens para criar o conjunto de treinamento e teste. Como classificador foram usadas a Regressão logística, Rede neural artificial e a Máquina de vetores de suporte. Para avaliar o desempenho das CNN é proposto a comparação dos resultados obtidos nos experimentos, considerando as métricas F1-*Score*, *Precission*, *Recall* e tempo de execução. O intuito é analisar a combinação de extrator junto ao um classificador, obtém um resultado superior se comparado a classificação usando os algoritmos CNN e ML somente como classificador.

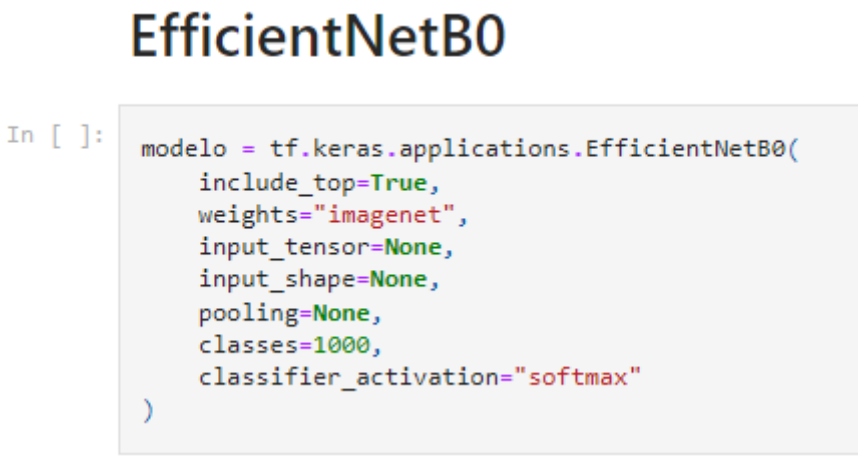
5.1.1 Resultados

Para demonstrar os resultados foram gerados gráficos e tabelas, com o desempenho geral obtido por cada extrator, para avaliar o desempenho gerado pelos classificadores foram consideradas as seguintes métricas: F1-*score*, *Precission*, *recall,* tempo de execução.

5.1.2 Extrator

Os primeiros resultados obtidos no experimento foram com o extrator de características, onde foi gerado os conjuntos de treinamento e teste. Sobre os conjuntos de treinamento e teste foram ativadas as redes neurais convolucionais profundas *EfficientNetB0, EfficientNetB7, EfficientNetV2L, MobileNet, NASNetLarge e VGG*-16. A extração das características resultou em 2 arquivos do tipo CSV contendo os dados extraídos pela rede neural convolucional.

Figura 14 - Código Keras para modelo EfficientNetB0



Fonte: Google Colab (2022).

A Figura 14 demonstra um exemplo de código usado para carregar o modelo *EfficientNetB0*, com os pesos "*weights=imagenet*" já pré-treinado com 1000 classe "classes =1000" do conjunto *ImageNet* e a função de classificação *softmax* "*Classifier_activation=softmax*".

Figura 15 - Código Keras para modelo EfficientNetB7

EfficientNetB7

```
modelo = tf.keras.applications.EfficientNetB7(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax"
)
```

Fonte: Google Colab (2022).

A Figura 15 demonstra um exemplo de código usado para carregar o modelo *EfficientNetB7*, com os pesos "*weights=imagenet*" já pré-treinado com 1000 classe "classes =1000" do conjunto *ImageNet* e a função de ativação *softmax* "*Classifier_activation=softmax*".

Figura 16 - Código Keras para modelo EfficientNetV2L

EfficientNetV2L

```
modelo = tf.keras.applications.EfficientNetV2L(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
    include_preprocessing=True,
)
```

Fonte: Google Colab (2022).

A Figura 16 demonstra um exemplo de código usado para carregar o modelo *EfficientNetV2L*, com os pesos "*weights=imagenet*" já pré-treinado com 1000 classe "classes =1000" do conjunto *ImageNet,* a função de ativação *softmax* "*Classifier_activation=softmax*" e entrado pré-processado "include_preproessing=True".

Figura 17 - Código Keras para modelo NASNetLarge

NASNetLarge

```
[ ]  modelo = tf.keras.applications.NASNetLarge(
         input_shape=None,
         include_top=True,
         weights="imagenet",
         input_tensor=None,
         pooling=None,
         classes=1000
     )
```

Fonte: Google Colab (2022).

A Figura 17 demonstra um exemplo de código usado para carregar o modelo *NASNetLarge*, com os pesos "*weights=imagenet*" já pré-treinado com 1000 classe "classes =1000" do conjunto *ImageNet* e a função de ativação *softmax* "*Classifier_activation=softmax*".

Figura 18 - Código Keras para modelo MobileNet

MobileNet

```
[ ]  modelo = tf.keras.applications.MobileNet(
        input_shape=None,
        alpha=1.0,
        depth_multiplier=1,
        dropout=0.001,
        include_top=True,
        weights="imagenet",
        input_tensor=None,
        pooling=None,
        classes=1000,
        classifier_activation="softmax"
    )
```

Fonte: Google Colab (2022).

A Figura 18 demonstra um exemplo de código usado para carregar o modelo *MobileNet*, como o controle de largura da rede "*alpha*=1.0", o multiplicador de profundidade para convolução ou multiplicador de resolução denominado de "*depth_multiplier*=1", taxa de perda "*dropout*=0.001", os pesos "*weights=imagenet*" já pré-treinado com 1000 classe "classes=1000" do conjunto *ImageNet* e a função de ativação *softmax* "*Classifier_activation=softmax*".

Figura 19 - Código Keras para modelo VGG-16

VGG-16

```
In [ ]:  modelo = tf.keras.applications.VGG16(
            include_top=True,
            weights="imagenet",
            input_tensor=None,
            input_shape=None,
            pooling=None,
            classes=1000,
            classifier_activation="softmax",
        )
```

Fonte: Google Colab (2022).

A Figura 19 ilustra um exemplo de código usado para carregar o modelo *VGG*-16, usando as bibliotecas *Tensorflow* e *Keras*. Para este modelo foi escolhido os seguintes parâmetros: o peso "*weights= imagenet*", número de classe pré-treinada "classes =1000" com conjunto de imagem *ImageNet* e a última camada para a ativação do classificador *"Classifier_activation=softmax"*. O *softmax* é a função que retorna a porcentagem a qual a imagem de entra pertence com base no número de classe. Os modelos foram carregados a partir da biblioteca *Keras*.

5.1.3 Regressão logística

No experimento como o algoritmo regressão logística foram selecionados os parâmetros: *Penalty*: L1 e L2, força de regularização "C": 0.5, 0.75 e 1, otimizador de código *solver*: saga e número máximo de interação *Max_inter*: 30, 60, 80 e 100

Figura 20 - Função para o treinamento do modelo Regressão Logística

```
[ ]  def Load_model(Penalty, C_, Solver, Max_iter):
       # inica a contagem do tempo
       inicio_treino = init_time()

       # Treinamento
       model = LogisticRegression(
                                  penalty=Penalty,
                                  random_state=0,
                                  C= C_,
                                  solver=Solver,
                                  max_iter= Max_iter,
                                  multi_class = 'auto').fit( x_treino, y_treino)
       final_treino = time_ex(inicio_treino)

       return {'Model':model, 'Tempo': final_treino}
```

Fonte: Google Colab (2022).

Foi desenvolvido uma função para carregar o modelo, que possui como entrada os parâmetros de norma de pena "Penalty", a força de regularização "C", otimizador de código "*Solver*" e número máximo de interação "Max_iter". Como saída é retornado um dicionário contendo o modelo treinado com os parâmetros passados na entrada, essa função é demonstrada na Figura 20.

Os melhores resultados obtidos pelo experimento com a Regressão Logística estão ilustrados na Figura 21 .

Figura 21 - Melhor resultado obtido pela junção da CNN junto a Regressão Logística

CNN	Penalty	C	Solver	Max_inter	f1-score	Precision	Recall	Tempo (min)
VGG-16	L1	1,00	saga	60	0,8239	0,8313	0,8273	24,4693
MobileNet	L1	0,75	saga	30	0,6836	0,7133	0,6835	3,1013
NASNetLarge	L1	1,00	saga	100	0,5301	0,6101	0,552	7,6264
EfficientNetB0	L2	1,00	saga	30	0,8419	0,8651	0,8330	3,3745
EfficientNetB7	L1	1,00	saga	80	0,8440	0,8527	0,8440	9,9670
EfficientNetV2L	L1	1,00	saga	100	0,7984	0,8184	0,8045	12,5818

Fonte: Produção do autor (2022).

A Figura 21 ilustra uma tabela com os resultados gerados pela combinação de extratores, a tabela é composta por 9 colunas sendo elas: CNN, *Penalty*, C, *Solver*, *Max_iter*, F1-*score*,

Precision e *Recall* e Tempo (s). Ao analisar a Figura 21 é possível concluir que a melhor combinação de CNN com a Regressão Logística foi obtida pelos parâmetros: *EfficientNetB0*, norma de pena *Penalty*: L2, a força de regularização C: 1, *Solver*=saga, e número máximo de interação: 30. A combinação desses parâmetros obteve um acerto de 84% no tempo de 3,3 minutos.

O desempenho obtido pela combinação CNN mais o classificador Regressão Logística estão dispostos na Tabela 2.

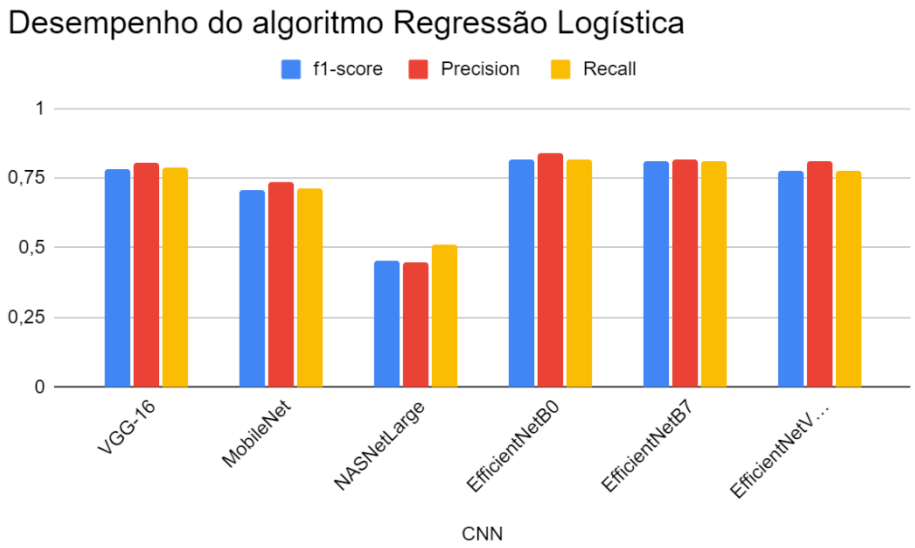
Tabela 2 - Resultado da combinação CNN + Classificador Regressão Logística

ML: Regressão Logística				
CNN	f1-score	Precision	Recall	Tempo (min)
VGG-16	0,7850	0,8048	0,7861	24,4545
MobileNet	0,7101	0,7374	0,7105	3,1249
NASNetLarge	0,4561	0,4504	0,5092	7,5419
EfficientNetB0	0,8161	0,8404	0,8153	3,4018
EfficientNetB7	0,8096	0,8192	0,8096	9,9074
EfficientNetV2L	0,7781	0,8111	0,7781	12,4959

Fonte: Produção do autor (2022).

A Tabela 2 possui 5 colunas sendo elas: CNN, F1-*score*, *Precision*, *Recall* e Tempo. Ao analisar a tabela é possível notar que o melhor resultado foi obtido pelo extrator *EfficienteNetB0* com a precisão de 84% no tempo de 3,4 minuto.

Figura 22- Desempenho Geral da Regressão Logística



Fonte: Produção do autor (2022).

O desempenho obtido pelo algoritmo de regressão logística é ilustrado na Figura 22, onde cotem o gráfico com os resultados obtidos com cada extrator. Nesta imagem é possível notar que o *EfficientNetB0* foi o melhor extrator para esse experimento.

Na Tabela 3 estão contidos o desempenho médio obtido junto aos extratores pelo modelo de regressão logística.

Tabela 3 - Media geral de desempenho obtidas CNN + Classificador Regressão Logística

ML: Regressão Logística - Media Geral				
F1	Precisão	Recall	Média	Tempo (min)
0,7258	0,7439	0,7348	0,7348	0,0642

Fonte: Produção do autor (2022).

De um ponto de vista geral, a Tabela 3 obteve um acerto médio de 73% no tempo de 0,06 minuto para este experimento, isso considerando o resultado médio obtido por todas as CNN utilizadas no experimento.

5.1.4 Rede neural artificial

Para os experimentos com a Rede neural artificial foram selecionados os parâmetros: número de neurônio: 80,70, função de ativação: *Identity*, *Logistic*, Relu e tangente hiperbólica, otimizador de código: L-BFGS-B, SGD e Adam e o número de interação em 2000.

Figura 23 - Função para o treinamento do modelo Rede Neural Artificial

```
def Load_model(func_ativacao, solve_val, iteracao_val):
  # inica a contagem do tempo
  model = MLPClassifier(hidden_layer_sizes= (80,70),
                        activation=func_ativacao,
                        solver= solve_val,
                        random_state=1,
                        max_iter = iteracao_val).fit(x_treino, y_treino)

  return {'Model':model}
```

Fonte: Google Colab (2022).

Foi desenvolvido uma função para carregar o modelo, que possui como entrada os parâmetros de função de ativação "func_ativacao", otimizador de código "*solver_val*" e número de interação "interacao_val". Como saída é retornado um dicionário contendo o modelo treinado com os parâmetros passados na entrada, essa função é demonstrada na Figura 23.

Os melhores resultados obtidos por cada extrator no experimento com a rede neural artificial são ilustrados na Figura 24.

Figura 24 - Melhor resultado obtido pela junção da CNN junto a Rede Neural artificial

Incorporador	Nº Neuronios	func_ativação	solver	iteração	F1	Precision	Recall	Tempo (min)
VGG-16	80, 70	ReLu	L-BFGS-B	2000	0,9685	0,9685	0,9693	24,9692
MobileNet	80, 70	ReLu	Adam	2000	0,9435	0,9451	0,9429	3,5889
NASNetLarge	80, 70	tanh	Adam	2000	0,9461	0,9453	0,9488	8,0610
EfficientNetB0	80, 70	Identity	Adam	2000	0,9922	0,9916	0,9929	3,4681
EfficientNetB7	80, 70	ReLu	Adam	2000	0,9946	0,9956	0,9938	10,0632
EfficientNetV2L	80, 70	Identity	Adam	2000	0,9928	0,9941	0,9916	12,6251

Fonte: Produção do autor (2022).

A Figura 24 ilustra uma tabela com os resultados dos extratores, esta tabela contém as colunas incorporador, número de neurônio, função de ativação, otimizador de código e número de intenção, os desempenhos medidos por F1-*score*, *Precision*, *Recall* e Tempo. Ao analisar a Figura 24 é possível concluir que o melhor resultado foi gerado pela combinação dos seguintes parâmetros: extrator: *EfficientNetB0*, número de neurônios: 80,70, função de ativação: *Relu*, otimizador Adam e 2000 interações. Esta combinação obteve um acerto de 99,2% no tempo de 3,4 minutos.

A avaliação do desempenho obtido pela rede neural artificial, considerando as características extraídas pelas CNNs, estão contidos na Tabela 4. Esta tabela é composta por 5 colunas, denominadas CNN, F1, *Precision,* R*ecall*, Tempo.

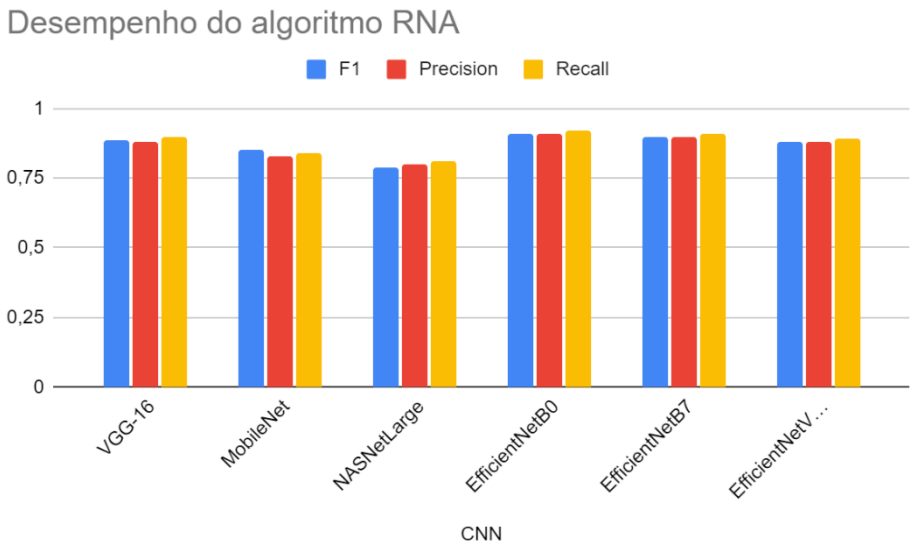
Tabela 4 - Resultado da combinação CNN + Classificador Rede Neural Artificial

MLP: Rede Neural Artificial				
CNN	F1	Precision	Recall	Tempo (min)
VGG-16	0,8852	0,8819	0,8985	24,3958
MobileNet	0,8492	0,8477	0,8611	3,0788
NASNetL	0,7886	0,7997	0,8109	7,4968
EfficientNB0	0,9119	0,9087	0,9235	3,3508
EfficientNB7	0,9002	0,8990	0,9112	9,8624
EfficientNV2L	0,8784	0,8780	0,8945	12,4533

Fonte: Produção do autor (2022).

Os resultados gerados pela rede neural artificial demonstraram que o melhor extrator foi o *EfficientNetB0* que obteve um acerto médio de 91% com tempo de execução de 3,3 minuto, isso é demonstrado na Tabela 4.

Figura 25 - Desempenho Geral da Rede Neural Artificial



Fonte: Produção do autor (2022).

Por meio do desempenho do algoritmo é possível analisar de forma visual os resultados obtidos pela rede neural artificial, isso é demonstrado na Figura 25 com o gráfico de desempenho da RNA, onde o extrator *EfficientNetB0* obteve um desempenho superior aos demais extratores.

Tabela 5 - Media geral de desempenho obtidas Classificador Rede Neural Artificial

ML: Media Geral - Rede Neural Artificial				
F1	Precision	Recall	Média	Tempo (min)
0,9755	0,9756	0,9761	0,9757	10,1063

Fonte: Produção do autor (2022).

O resultado em um contexto geral obtido pela CNN está disposto na Tabela 5, esses resultados demonstras que a RNA obtém um desempenho de 97% acerto no tempo de 10 minuto considerando a média entre as métricas obtidas pelos extratores.

5.1.5 Máquina de vetores de suporte

Com o algoritmo de aprendizado de máquina SVM foram testados os seguintes parâmetros: regularizador C: (0,25/ 0,5/ 0,75/ 1), tipos de *Kernel*: (*linear*, *poly*, *rbf*, *sigmoid*), coeficiente do *kernel* denominado Gama: auto e *Degree*: (3).

Figura 26 - Função para o treinamento do modelo SVM

```
def Load_model(C_, Kernel, Degree):
  model = make_pipeline(StandardScaler(),
                        SVC(C= C_,
                            kernel=Kernel,
                            degree=Degree,
                            gamma='auto',
                            decision_function_shape='ovr',
                            random_state=2))

  model.fit(x_treino, y_treino)

  return {'Model':model}
```

Fonte: Google Colab (2022).

A Figura 26 ilustra a função usada no ambiente colab, para o treinamento do algoritmo. A função tem como entrada o regularizador “c_”, *kernel* e coeficiente do *kernel* “Degree”, esta função retorna um dicionário contendo o treinamento do modelo.

Figura 27 - Os 5 Melhores Resultado Obtidos pela Máquina de Vetores de Suporte – SVM

CNN	Penalty	C	Solver	Max_inter	f1-score	Precision	Recall	Tempo (min)
VGG-16	L1	1,00	saga	60	0,8239	0,8313	0,8273	24,4693
MobileNet	L1	0,75	saga	30	0,6836	0,7133	0,6835	3,1013
NASNetLarge	L1	1,00	saga	100	0,5301	0,6101	0,552	7,6264
EfficientNetB0	L2	1,00	saga	30	0,8419	0,8651	0,8330	3,3745
EfficientNetB7	L1	1,00	saga	80	0,8440	0,8527	0,8440	9,9670
EfficientNetV2L	L1	1,00	saga	100	0,7984	0,8184	0,8045	12,5818

Fonte: Produção do autor (2022).

Os melhores resultados de desempenho de cada CNN estão ilustrados na Figura 27. Essa figura contém uma tabela com 10 colunas composta pelos paramentos, CNN, C, *Kernel*, Gama, *Degree*, F1-*score*, *Precision*, *Recall*, Média e Tempo e as métricas de desempenho. A melhor combinação foi *EfficientNetB0* com os parâmetros C: 1, *kernel*: linear, Gama: auto e *Degree*: 3 com o desempenho médio de 99,5% de acerto no tempo de 3,3 minutos.

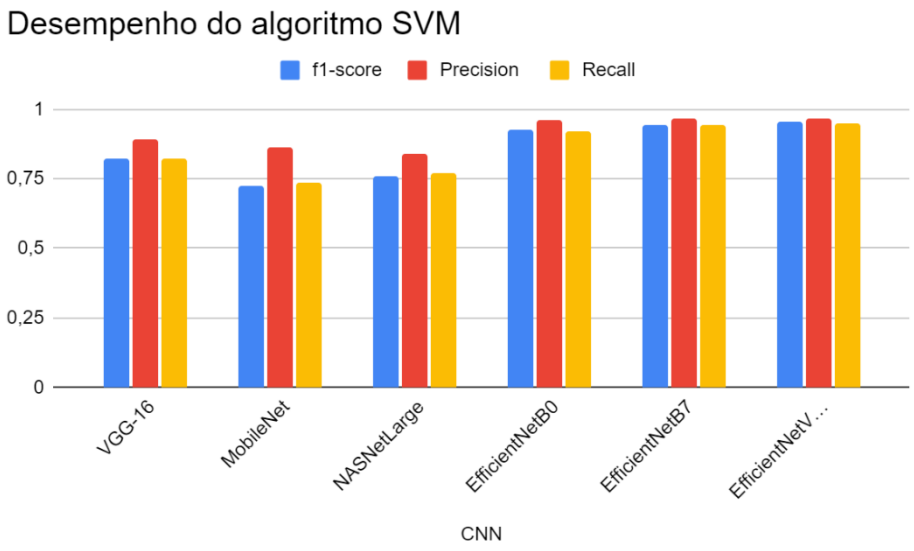
Tabela 6 - Resultado da combinação CNN + Classificador SVM

ML-SVM					
CNN	f1-score	Precision	Recall	Média	Tempo (min)
VGG-16	0,8216	0,8905	0,8211	0,8444	24,3809
MobileNet	0,7242	0,8650	0,7367	0,7753	3,0564
NASNetLarge	0,7620	0,8400	0,7685	0,7902	7,4805
EfficientNetB0	0,9260	0,9607	0,9222	0,9363	3,3407
EfficientNetB7	0,9444	0,9649	0,9422	0,9505	9,8472
EfficientNetV2L	0,9533	0,9656	0,9513	0,9567	12,4357

Fonte: Produção do autor (2022).

A média dos resultados obtidos pela combinação entre as CNNs como extrator junto ao classificador SVM estão contidos na Tabela 6, analisando-a é possível notar que o maior valor seria atribuído pelo *EfficienteNetV2L* isso se desconsideramos a métrico tempo, mas ao consideramos esta métrica notamos que o melhor resultado foi obtido pelo *EfficientNetB0* com acerto médio de 93% em um tempo de 3,3 minutos.

Figura 28 - Desempenho Geral da Rede Neural Artificial



Fonte: Produção do autor (2022).

A Figura 28 apresenta o gráfico com os desempenhos obtidos pelo algoritmo SVM, nesta imagem é possível notar que ambos os modelos da família *EfficienteNet* B0, B6 e V2L obtiveram um acerto equivalente, o que nos leva a considerar a métrico tempo como decisiva.

6 CONCLUSÃO

No relatório presente, foram apresentadas as atividades desenvolvidas no período de 1 setembro de 2022 a 31 dezembro de 2022, referente ao projeto "Análise de Técnicas de Aprendizado Profundo e de Aprendizado de Máquina Tradicional para Classificação de Imagens de Drones". Em uma primeira etapa da pesquisa, iniciou-se um processo em que foram comparadas as seguintes CNNs usadas como extratores de características apenas: *EfficientNetB0, EfficientNetB7, EfficientNetV2L, MobileNet, NASNetLarge, VGG*-16. E, nessa fase, os seguintes classificadores tradicionais foram usados: regressão logística, rede neural artificial e SVM. Os resultados demonstram que o extrator *EfficientNetB0* com o classificador SVM tiveram os melhores resultados, obtendo um F1-*Score* de 99,5%, e um tempo de execução de 3,45 minutos. A variável tempo é um ponto muito importante nos experimentos realizados, já que pode-se notar que nem sempre o melhor resultado é gerado no menor tempo. Todas as atividades (A1 a A5) previstas no período a que se refere este relatório foram 100% concluídas. Além disso, mesmo que as atividades A6 e A7 estivessem sido previstas para serem realizadas após o período a que se refere este relatório parcial, as realizações das mesmas foram antecipadas, sendo que a atividade A6 foi 50% concluída e a atividade A7 foi também 50% concluída. Portanto, pode-se afirmar que as atividades previstas, no período a que se refere este relatório parcial, foram plenamente concluídas, havendo mesmo uma antecipação de realização de atividades que estavam previstas para etapas posteriores do projeto. Como trabalhos futuros, é proposto dar continuidade para a conclusão das atividades A6, A7, A8 e A9.

REFERÊNCIAS

ARTIFICIAL Inteligência (AI): O que é IA?. In: Https://www.oracle.com/br/artificial-intelligence/what-is-ai/. [S. l.], 2021. Disponível em: https://www.oracle.com/br/artificial-intelligence/what-is-ai/. Acesso em: 28 jan. 2021.

B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, (New York, NY, USA), p. 144–152, Association for Computing Machinery, 1992.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, e Z. Wojna, "Rethinking the inception architecture for computer vision," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (Los Alamitos, CA, USA), pp. 2818–2826, IEEE Computer Society, jun 2016.

FORNARI, Gabriel; Júnior, Valdivino; Shiguemori, Elcio. Caracterização e desenvolvimento de um sistema autoadaptativo para estimação da posição de veículos aéreos não tripulados baseado em imagens. 2020. 183f. Tese de Doutorado – Computação Aplicada, São José dos Campos.

F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," Psychological review, vol. 65 6, pp. 386–408, 1958.

H. Ghanbari, M. Mahdianpari, S. Homayouni, and F. Mohammadimanesh,"A meta-analysis of convolutional neural networks for remote sensing applications," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 3602–3613, 2021.

GOOGLE COLABORATORY. In: O Google Colaboratory é um ambiente de notebooks Jupyter que não requer configuração e é executado na nuvem do Google. Escreva e execute códigos em Python e R, 2021. Disponível em: https://www.image-net.org/. Acesso em: 28 ago. 2022.

X. Guo, X. Huang, L. Zhang, L. Zhang, A. Plaza, and J. A. Benediktsson,"Support tensor machines for classification of hyperspectral remote sensing imagery," IEEE Transactions on Geoscience and RemoteSensing, vol. 54, pp. 3248–3264, June 2016.

T. K. Ho, "Random decision forests," in Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1, pp. 278–282vol.1, Aug 1995.

HOWARD, Andrew G. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. [s. l.], 17 abr. 2017. DOI 10.48550/arXiv.1704.04861. Disponível em: https://arxiv.org/abs/1704.04861.

IBM (Tecnologia da informação) et al. O que é Inteligência Artificial?. In: ORACLE (Tecnologia e informática) et al. O que é Inteligência Artificial?. [S. l.], 2021. Disponível em: https://www.ibm.com/br-pt/cloud/learn/what-is-artificial-intelligence. Acesso em: 8 fev. 2022.

IMAGENET. In: ImageNet é um grande banco de dados visual projetado para uso em pesquisa de software de reconhecimento de objetos visuais, 2021. Disponível em: https://www.image-net.org/. Acesso em: 28 ago. 2022.

J. B. M.D., "Application of the logistic function to bio-assay," Journal of the American Statistical Association, vol. 39, no. 227, pp. 357–365, 1944.

J. Zhu, S. Rosset, H. Zou, e T. Hastie, "Multi-class adaboost," Statistics e its interface, vol. 2, fev 2006.

KERAS. In: O Keras é uma biblioteca de rede neural de código aberto escrita em Python. 2.8.0. [S. l.], 2021. Disponível em: https://keras.io/. Acesso em: 8 mai. 2022.

KENJI, Bruno. Machine Learning para Leigos, 2019. Disponível em: https: https://www.venturus.org.br/machine-learning-para-leigos/. Acesso em: 29 ago. 2022.

K. Simonyan e A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, Ma 7-9, 2015, Conference Track Proceedings (Y. Bengio e Y. LeCun, eds.), 2015.

LACERDA, Marielcio Gonçalves. Abordagem GEOBIA para Imagens VHR Obtidas por Aeronaves Remotamente Pilotadas e Sensores Satelitais com o Uso de Classificadores Individuais e Ensemble. 2020. 191f. Dissertação de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

LE, Quoc V.; TAN, Mingxing. EfficientNetV2: Smaller Models and Faster Training. EfficientNetV2: Smaller Models and Faster Training, [s. l.], 23 jun. 2021. DOI 10.48550/arXiv.2104.00298. Disponível em: https://arxiv.org/abs/2104.00298. Acesso em: 16 dez. 2022.

Ludermir, Teresa Bernarda. Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências. Estudos Avançados [online]. 2021, v. 35, n. 101 [Acessado 28 ago 2022], pp. 85-94. Disponível em: <https://doi.org/10.1590/s0103-4014.2021.35101.007>. Epub 19 Abr 2021. ISSN 1806-9592. 10.1590/s0103-4014.2021.35101.007.

M. Tan e Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

M. E. Mohammadi, D. P. Watson, and R. L. Wood, "Deep learning based damage detection from aerial sfm point clouds," Drones, vol. 3,no. 3, 2019.

O QUE é Machine Learning. In: ORACLE (Tecnologia e informática) et al. O que é Machine Learning. [S. l.], 2021. Disponível em: https://www.oracle.com/br/data-science/machine-learning/what-is-machine-learning/. Acesso em: 8 fev. 2022.

PANDAS. In: Pandas é uma biblioteca de software criada para a linguagem Python para manipulação e análise de dados. 1.3.5, 2008. Disponível em: https://pandas.pydata.org. Acesso em: 28 ago. 2022.

PAULINO, Â. C.; Guimarães, L. N. F.; Shiguemori, E. H. Assessment of Noise Impact on Hybrid Adaptive Computational Intelligence Multisensor Data Fusion Applied to Real-Time UAV Autonomous Navigation. IEEE Latin America Transactions, v. 18, p. 295-302, 2020.

PYTHON 3.7.13: Python é uma linguagem de programação interpretada, interativa e orientada a objetos. O Python é uma das linguagens de programação mais usadas no mundo e possui uma baixa curva de aprendizagem (PYTHON SOFTWARE FUNDATION, 2022).

RESOURCE limits. In: Resource limits. [S. l.], Ex.: Disponível em: https://research.google.com/colaboratory/faq.html#resource-limits. Acesso em: 11 mar. 2021.

Roos, D. R.; Lorena, A. C.; Shiguemori, E.H. Comparing ORB and AKAZE for visual odometry of unmanned aerial vehicles. In: Conference of Computational Interdisciplinary Science, 2016, São José dos Campos. Proceedings of Conference of Computational Interdisciplinary Science. São José dos Campos: INPE, 2016. v. 4. p. 121.

SCITKIT-LERARN. In: A scikit-learn é uma biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python. 1.0.2, 2021. Disponível em: https://keras.io/. Acesso em: 28 ago. 2022.

S. Inage e H. Hebishima, “Application of monte carlo stochastic optimization (MOST) to deep learning,” CoRR, vol. abs/2109.02441, 2021.

TAN, Mingxing; LE, Quoc V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. [s. l.], 11 set. 2020. DOI 10.48550/arXiv.1905. Disponível em: https://arxiv.org/abs/1905.11946. Acesso em: 15 dez. 2022.

TENSORFLOW. In: TensorFlow é uma biblioteca de código aberto para aprendizado de máquina aplicável a uma ampla variedade de tarefas. 2.8.2 [S. l.], 2015. Disponível em: https://www.tensorflow.org/?hl=pt-br. Acesso em: 8 mai. 2022.

T. K. Ho, A Theory of Multiple Classifier Systems and Its Application to Visual Word Recognition. PhD thesis, USA, 1992. UMI Order No. GAX92-22062.

V. Kumar e M. Sahu, “Evaluation of nine machine learning regression algorithms for calibration of low-cost pm2.5 sensor,” Journal of Aerosol Science, vol. 157, p. 105809, 2021.

WANG, Jason; PEREZ, Luis. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. Computer Vision and Pattern Recognition, https://arxiv.org/abs/1712.04621, 13 dez. 2017.

ZAMBIASI, Saulo Popov. O Neurônio Artificial. [S. l.], 2008. Disponível em: https://www.gsigma.ufsc.br/~popov/aulas/rna/neuronio_artificial/index.html. Acesso em: 29 ago. 2022.

ZOPH, Barret; VASUDEVAN, Vijay; SHLENS, Jonathon; LE, Quoc V. Learning Transferable Architectures for Scalable Image Recognition. [s. l.], 11 abr. 2018. DOI 10.48550/arXiv.1707.07012. Disponível em: https://arxiv.org/abs/1707.07012. Acesso em: 16 dez. 2022.