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

EMPREGO DE INTELIGÊNCIA ARTIFICIAL NA ESCOLHA AUTOMÁTICA DE ALGORITMOS E PARÂMETROS DE TÉCNICAS DE PROCESSAMENTO DE IMAGENS OBTIDAS POR DRONES PARA APLICAÇÃO NO SENSORIAMENTO REMOTO

Hércules Carlos Dos Santos Pereira

INPE
São José dos Campos
2022.

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

EMPREGO DE INTELIGÊNCIA ARTIFICIAL NA ESCOLHA AUTOMÁTICA DE ALGORITMOS E PARÂMETROS DE TÉCNICAS DE PROCESSAMENTO DE IMAGENS OBTIDAS POR DRONES PARA APLICAÇÃO NO SENSORIAMENTO REMOTO

Hércules Carlos Dos Santos Pereira

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

INPE
São José dos Campos
2022.

RESUMO

Os drones ou veículos aéreos não tripulados (VANTs) 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 investigar a classificação de imagens obtidas por drones utilizando inteligência artificial (IA) e técnicas de processamento de imagens. 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: *InceptionV3*, *SqueezeNet*, *VGG*-16, *VGG*-19, *Painters* e *DeepLoc*. Como classificadores, foram usados: *Adaboost*, Floresta Aleatória, Regressão logística e Rede Neural Artificial. Os resultados demonstram que o extrator *VGG*-16 com o classificador Regressão Logística tiveram os melhores resultados, se comparados com as demais técnicas, atingindo acerto de até 100% nas métricas, Acurácia, F1-*Score*, *Precision* e *Recall*. Em uma segunda etapa da pesquisa, iniciou-se um processo onde foram comparadas as seguintes CNNs como extratores de características: *VGG*-16 e *EfficientNetB0*. E, nessa fase, os seguintes classificadores foram usados: Rede Neural Artificia e Máquina de Vetores de Suporte (SVM). Os resultados demonstram que o extrator *EfficientNetB0* com o classificador SVM tiveram os melhores resultados, obtendo um F1-*Score* de 99,5%.

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

LISTA DE FIGURAS

Figura 1 – Amostra de classe..... 12
Figura 2 - Modelo de Neurônio Artificial..... 13
Figura 3 - Técnica de aumento de dados..... 14
Figura 4 - Arquitetura básica de uma Rede Neural Convolucional..... 15
Figura 5 - Imagens Obtidas por Drone 19
Figura 6 - Tabela exemplo de extração de características..... 20
Figura 7 - Exemplo de código com extração e classificação das imagens 23
Figura 8 - Widget ambiente Orange - Rede Neural Artificial..... 24
Figura 9 - Média de Desempenho obtido pelo Algoritmos Rede Neural Artificial 25
Figura 10- Matriz de confusão gerada pelo algoritmo de Rede Neural Artificial..... 25
Figura 11 - Exemplo de classificação correta gerado pela Rede Neural Artificial 26
Figura 12 - Widget ambiente Orange - Regressão Logística..... 27
Figura 13 - Média de Desempenho obtido pelo Algoritmos Regressão Logística 28
Figura 14 - Matriz de confusão gerada pelo algoritmo de Regressão Logística..... 28
Figura 15 - Exemplo de classificação correta gerado pela Regressão Logística..... 29
Figura 16- Exemplo de erro de classificação Regressão Logística..... 30
Figura 17 - Widget ambiente Orange – Floresta Aleatória..... 30
Figura 18 - Média de Desempenho obtido pelo Algoritmos Floresta aleatória 31
Figura 19 - Matriz de confusão gerada pelo algoritmo de Floresta Aleatória..... 32
Figura 20 - Exemplo de classificação correta gerado pela Floresta Aleatória 32
Figura 21 - Exemplo de erro de classificação Floresta Aleatória 33
Figura 22 - Widget ambiente Orange - AdaBoost..... 34
Figura 23 - Média de Desempenho obtido pelo Algoritmos AdaBoost..... 35
Figura 24 - Matriz de confusão gerada pelo algoritmo de AdaBoost..... 35
Figura 25 - Exemplo de classificação correta gerado pela Adaboost 36
Figura 26 - Exemplo de erro de classificação Adaboost 36
Figura 27 - Resultados obtidos pelos extratores e classificadores 37
Figura 28 - Código Keras para modelo VGG-16..... 39
Figura 29 - Código Keras para modelo EfficientNetB0..... 39
Figura 30 - Função para o treinamento do modelo Rede Neural Artificial..... 40
Figura 31 - Melhor resultado obtido pela junção da CNN junto a Rede Neural artificial 40
Figura 32 - Desempenho Geral da Rede Neural Artificial 41
Figura 33 - Função para o treinamento do modelo SVM..... 43
Figura 34 - Os 5 Melhores Resultado Obtidos pela Máquina de Vetores de Suporte – SVM..... 43
Figura 35 - Desempenho Médio obtido pela Máquina de Vetores de Suporte - SVM..... 44

LISTA DE TABELAS

Tabela 1 – Atividades realizadas..... 10
Tabela 2 - Resultado da combinação CNN + Classificador Rede Neural Artificial..... 41
Tabela 3 - Media geral de desempenho obtidas Classificador Rede Neural Artificial 42
Tabela 4 - Resultado da combinação CNN + Classificador Máquina de Vetores de Suporte – SVM 44
Tabela 5 - Media de desempenho obtida pela SVM..... 45

SUMÁRIO

RESUMO **3**
LISTA DE FIGURAS **4**
LISTA DE TABELAS **5**
SUMÁRIO **6**
1 INTRODUÇÃO **7**
2 CRONOGRAMA DE ATIVIDADE E ETAPAS CONCLUIDAS **9**
3 MATERIAIS E MÉTODOS **11**
3.1 MÉTODOS **11**
3.1.1 CLASSIFICAÇÃO **11**
3.1.2 APRENDIZADO DE MÁQUINA **12**
3.1.3 REDE NEURAL ARTIFICIAL **12**
3.1.4 AUMENTO DE DADOS **13**
3.1.5 REDE NEURAL CONVOLUCIONAL COMO EXTRATOR DE CARACTERÍSTICAS **15**
3.1.6 *ADABOOST* **17**
3.1.7 FLORESTA ALEATÓRIA **18**
3.1.7 MÁQUINA DE VETORES DE SUPORTE - SVM **18**
3.2 MATERIAIS **18**
3.2.1 IMAGENS OBTIDAS POR DRONE **19**
3.2.2 TABELA DE CARACTERÍSTICAS **19**
3.2.2 MÉTRICAS **20**
4 EXPERIMENTOS **21**
5 DESENVOLVIMENTO **23**
5.1 ABORDAGEM – 1 ..... **23**
5.1.1 RESULTADOS **23**
5.1.2 REDE NEURAL **24**
5.1.3 REGRESSÃO LOGÍSTICA **27**
5.1.4 FLORESTA ALEATÓRIA **30**
5.1.5 *ADABOOST* **34**
5.2 ABORDAGEM - 2 ..... **38**
5.2.1 RESULTADOS **38**
5.2.2 EXTRATOR **38**
5.2.3 REDE NEURAL ARTIFICIAL **40**
5.2.4 MÁQUINA DE VETORES DE SUPORTE **43**
6 CONCLUSÃO **46**
REFERÊNCIAS **47**

1 INTRODUÇÃO

Os drones também conhecidos como aeronaves remotamente pilotadas (ARP) ou veículo aéreo não tripulado (VANT), são expostos a cenários de voo com diferentes terrenos, alturas e velocidade. Existem diferentes tipos de aeronaves variam, tais como as de asa rotativa, asa fixa e híbridos (ROOS, LORENA e SHIGUEMORI, 2016).

Nas aplicações de sensoriamento remoto, a variação de cenários de voo dificulta no processo de automatização da classificação das imagens, gerando uma variação de imagens pelas condições de voo. Sendo assim, não há um algoritmo único de processamento de imagens ou visão computacional para a classificação autônoma das imagens (Ross, Lorena e Shiguemori, 2016). Sendo assim, os parâmetros dos algoritmos variam conforme os cenários de voo (Fornari, Santiago Junior e Shiguemori, 2018). Desta forma, surge a necessidade de utilizar modelos de inteligência artificial auto adaptativo para as condições. Para simular os cenários em questão foram separados um conjunto de imagens de voo sobre áreas urbanas, com a mesma altura de voo.

O objetivo dessa pesquisa é investigar a classificação de imagens obtidas por drones utilizando inteligência artificial (IA) e técnicas de processamento de imagens. 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, usando o ambiente Orange (Orange, 2022). Para essa etapa, avaliou-se as seguintes Redes Neurais Convolucionais (CNNs) como extratores de características: *InceptionV3* (Szegedy et all, 2016), *SqueezNet* (Landola et all, 2016), *VGG*-16 e VGG-19 (Szegedy et all, 2016), *Painters* (Llenic, 2016) e *DeepLoc* (Almagro et all, 2017). Como classificadores, foram usados os algoritmos de Aprendizado de Máquina tradicionais: Redes Neurais Artificiais (Keras, 2022), *AdaBoost* (Freund e Schapire, 1995), Regressão Logística (M.D, 1944) e Floresta Aleatória (Ho, 1992).

Já em uma segunda 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, o experimento foi avaliado usando as CNNs *VGG*-16 (Szegedy et all, 2016) e *EfficientNetB0* (Tan e Le, 2020) como extratores de características, e os algoritmos tradicionais de Aprendizado de Máquina, Rede Neural Artificia e Máquina de Vetores de Suporte (SVM).

Os resultados do primeiro experimento demonstram que o extrator *VGG*-16 junto com o classificador Regressão Logística obtiveram o melhor resultado, com acerto de até 100% nas métricas: Acurácias, F1-*score*, *Precision* e *recall*. Os resultados do segundo experimento demonstram que a CNN *EfficientNetB0* (Tan e Le, 2020) combinada com o classificador SVM obteve um F1-*Score* de até 99,5%.

2 CRONOGRAMA DE ATIVIDADE E ETAPAS CONCLUIDAS

Assim como disposto no “formulário de proposta de bolsa PIBIC”, a metodologia para atingir os objetivos do projeto é descrita a seguir:

1. Obtenção das informações de diferentes sensores obtidos em voos de Drones: disponíveis em repositórios do IEAv;
2. Análise dos dados com uso de ferramentas de ciência dos dados, com uso do ambiente Orange e linguagem Python;
3. Tratamento dos dados, como remoção de ruídos, integração de sensores, normalização, separação dos dados. Serão utilizados o ambiente Orange e a linguagem Python;
4. Emprego de diferentes técnicas de aprendizagem de máquina com uso das bibliotecas *Keras*, *Tensor Flow*, *Scikit Learn* (SCIKIT-LEARN, 2020);
5. Ajustes dos parâmetros das técnicas;
6. Avaliação rigorosa do desempenho das técnicas para identificar quais são as mais adequadas no contexto dessa pesquisa;
7. Implementação das técnicas em computador embarcado, com a *Raspberry Pi*.

Em termos detalhados, as atividades foram definidas como sendo nove como mostra a Tabela 1 – Atividades realizadasTabela 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 esse relatório (1 de setembro de 2021 a 31 de agosto de 2022. Sendo assim, é possível afirmar que basicamente todas as atividades previstas foram cumpridas com 100% de realização. A atividade 4 não foi totalmente concluída pois decidiu-se ampliar o escopo da investigação com a segunda etapa (segundo experimento). A atividade 6 é bastante demandante em termos computacional e, por isso, não foi concluída. Mas, pode-se dizer que os objetivos da pesquisa foram alcançados. Um artigo foi submetido para uma revista, como previsto, e atualmente estão sendo trabalhadas as sugestões dos revisores para uma submissão para um outro veículo (conferência, revista).

Tabela 1 – Atividades realizadas

Atividade da Metodologia	**Previsão**	**Realização**
1. Obter imagens obtidas em voos de drones;	100%	100%
2. Estudar o ambiente Orange e a linguagem Python para análise dos dados. Analisar os dados;	100%	100%
3. Estudar o ambiente Orange e linguagem Python para tratamento dos dados. Tratar os dados;	100%	100%
4. Aplicar redes neurais artificiais (rasas ou profundas) para realizar a fusão de dados. Observação: essa atividade foi modificada para: Avaliar algoritmos de Aprendizado de Máquina para classificação automática de imagens;	100%	75%
5. Escrever relatório parcial;	100%	100%
6. Estudar bibliotecas Keras, Tensor Flow, Scikit Learn. Aplicar técnicas de aprendizagem de máquina;	100%	100%
7. Ajustar os parâmetros das técnicas;	100%	60%
8. Comparar desempenho das técnicas via avaliação rigorosa;	100%	100%
9. 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.	100%	100%

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).

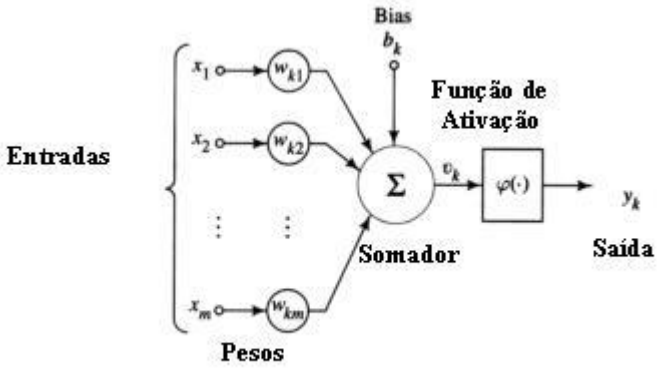
O método utilizado nos experimentos deste trabalho foi o aprendizado supervisionado, nessa abordagem se tem um conjunto de dados apresentados como entrada, também é apresentada o resultado esperado para a saída (Ludermir, 2021).

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



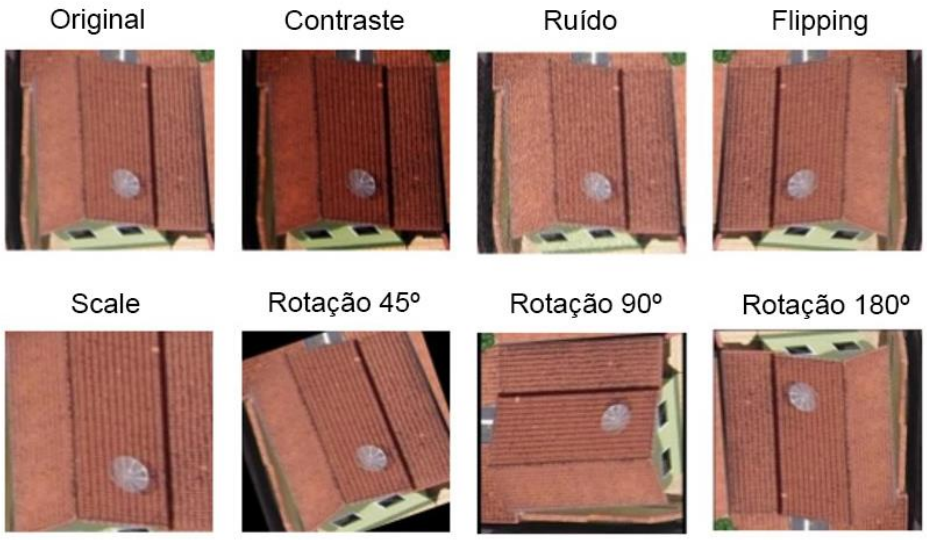
Fonte: Saulo Popov Zambiasi, 2008.

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 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 *3* - 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 3 - Técnica de aumento de dados

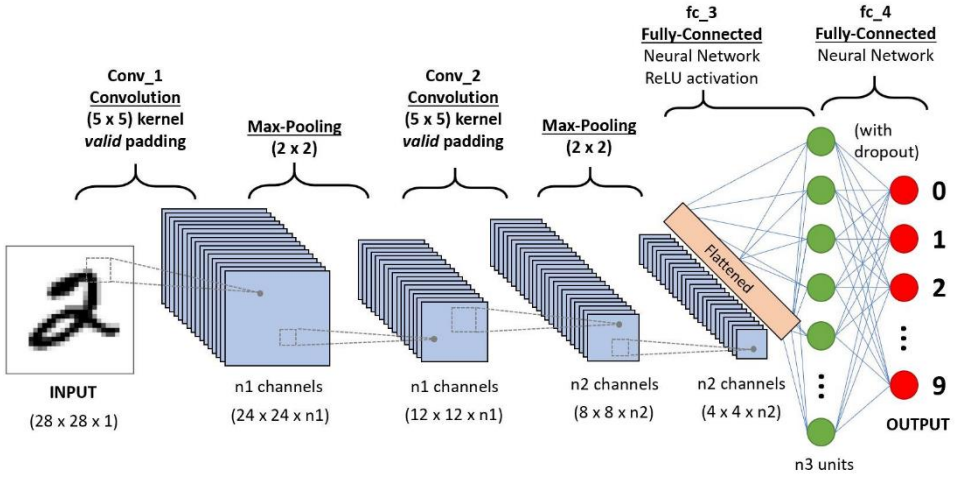


Fonte: Autoria própria (2021).

3.1.5 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 4 - Arquitetura básica de uma Rede Neural Convolucional



Fonte: Kenji, (2019).

A Figura *4* - 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 *SqueezNet* (Landola et all, 2016) é um modelo profundo de reconhecimento de imagens desenvolvido para ser menor e ter bom desempenho em seu treinamento. O modelo é pré-treinado com o conjunto de imagens *ImageNet* (ImageNet, 2021). A sua arquitetura é composta por uma camada de convolução denominada de (Conv1), oito módulos denominados *Fire* (Fire 2 a 9) e por fim uma última camada de convolução (Conv10). Nos filtros *Fire* são aplicados um aumento de forma gradual em todas as camadas da rede. Neste modelo é executado a cada dois passos um *Max-pooling* após as camadas *Conv1, Fire4, Fire8* e *Conv10*. O modelo *Fire* são compostos por filtros de 1 × 1 e 3 × 3.

As técnicas *VGG*-16 e *VGG*-19 (Szegedy et all, 2016) são duas redes neurais profundas 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. O modelo *VGG*-19 (Szegedy et all, 2016) segue a mesma estrutura com a mesma quantidade de camada (Conv1 - 5) a aplicação do *Max-pooling*. A camada Conv1 e Conv2 possuem 2 camadas convolucionais de 3 × 3, com a diferença que nas camadas Conv3 - 5 se tem 4 camadas com filtros de 3 × 3. O conjunto de dados *ImageNet* (ImageNet, 2021) citado nos modelos de CNN, é composto por 1000.000 imagens e mais de 80.000 classes rotuladas a mão (ImageNet, 2021).

O *Painters* (Llenic, 2016) é uma rede convolucional profunda treinada para reconhecimento de pinturas. O modelo é treinado com 79.433 imagens de 1.584 pintores diferentes. A sua arquitetura é composta por 24 camadas sendo que 14 são camadas convolucionais, 5 de *MaxPooling*, uma camada que cria um vetor único de recurso, 2 *dense* e 1 camada com a função *Softmax* de normalização de saída. A função de ativação usada é uma *PreRelu*, os filtros convolucionais são de dimensão 3 × 3 utilizados para reduzir informação da imagem.

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*.

Outra rede convolucional usada é o modelo *DeepLoc* (Almagro et all, 2017). No seu treinamento, foram usadas 21.882 imagens de células únicas de reparação de frações ósseas, proteína de levedura, oócitos de camundongos, e desenvolvimento de ameba.

3.1.6 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.

3.1.6 *AdaBoost*

O *Adaboost* (Freund e Schapire, 1995) é um algoritmo de aprendizado de máquina criado por Yoav e Robert em 1995 (Freund e Schapire, 1995). O conceito do *Ababoost* (Freund e Schapire, 1995) é a combinação de algoritmos fracos para criar um modelo de aprendizado mais robusto. Tem o objetivo é treinar preditores sequencialmente, onde cada um tenta corrigir o anterior. O algoritmo consegue melhorar os novos preditores conferindo onde o preditor anterior obteve um *underfitting*. Desta forma os novos preditores conseguem corrigir os erros anteriores.

3.1.7 Floresta aleatória

Floresta aleatória é o algoritmo aprendizado de máquina desenvolvido por Tim em 1992 (Ho, 1992), é composto por várias árvores de decisão, criadas de forma aleatória. O que se refere à floresta, é que o modelo cria uma combinação de árvores de decisão, treinada com método *bagging*. O método é usado para evitar *overfitting*, o *Bagging* consiste na criação de vários estimadores usando o mesmo modelo, tais como: árvore de decisão, KNN ou regressão linear. Nesta pesquisa foram usados os modelos de árvore de decisão para os testes com à floresta aleatória.

3.1.7 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 5 - Imagens Obtidas por Drone

Fonte: Marielcio, (2020).

A Figura *5* - Imagens Obtidas por DroneFigura *5* 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 6 - 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 *6* ilustra algumas características extraídas pela CNN, as linhas da tabela representam as imagens e as colunas são as características extraídas a partir destas imagens.

3.2.2 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:

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

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

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

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

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 ferramentas diferentes, na primeira abordagem se utilizou o ambiente Orange. O uso desta ferramenta tem o intuito de desenvolver os conhecimentos básicos sobre análise e 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. Para segunda abordagem utilizou as linguagens Python, as bibliotecas Pandas, *Tensorflow*, *Keras* e *scikit-learn*. Esta linguagem junto as bibliotecas são usadas para desenvolver conhecimento em análise, tratamento dos dados e a aplicação das técnicas de aprendizado de máquina usando as ferramentas citadas anteriormente.

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.

O primeiro experimento foi desenvolvido no ambiente Orange (Orange, 2022), onde, foram passadas duas estradas para o conjunto de treinamento e outra para o teste. Para extrair as características foram considerados diferentes modelos de rede neural convolucional, avaliou-se os seguintes extratores: *InceptionV3*, *SqueezNet*, *VGG*-16, *VGG*-19, *Painters* e *DeepLoc*, os algoritmos usados como classificadores foram a Rede Neural Artificial, *Adaboost*, Regressão Logística e Floresta aleatória. Após o treinamento foram anotados os resultados obtidos pela ferramenta de "*Teste ande score*" que possui as métricas *Acurracy*, F1-*score*, *Precision* e *Recall*. Por meio destes resultados foram gerados os gráficos que representa o desempenho obtidos por cada extrator e classificador.

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.

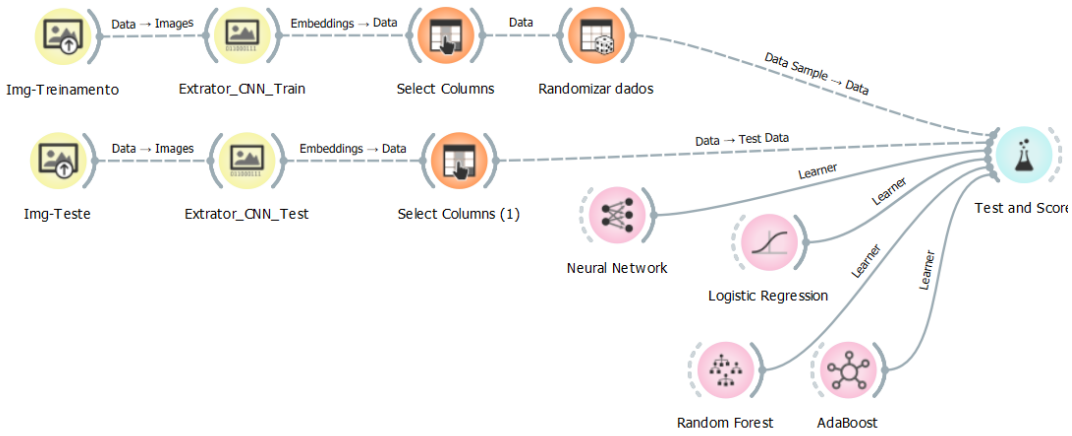
5 DESENVOLVIMENTO

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

5.1 Abordagem – 1

Na abordagem 1 foram feitos teste de combinação de extratores e classificadores. Sendo que nos classificadores foram variados diferentes parâmetros, por meio do ambiente Orange.

Figura 7 - Exemplo de código com extração e classificação das imagens



Fonte: Orange 3.32.0 (2022).

Um exemplo do código usado para a realização dos testes na abordagem 1 é apresentado na Figura 7, inicialmente são carregados o conjunto de imagens de treinamento e teste, depois são aplicados os diferentes extratores sobre as imagens, é feito a seleção das colunas, a randomizar dados de treinamento, e o treinamento dos algoritmos de Aprendizado de máquina e por último avaliar o resultado.

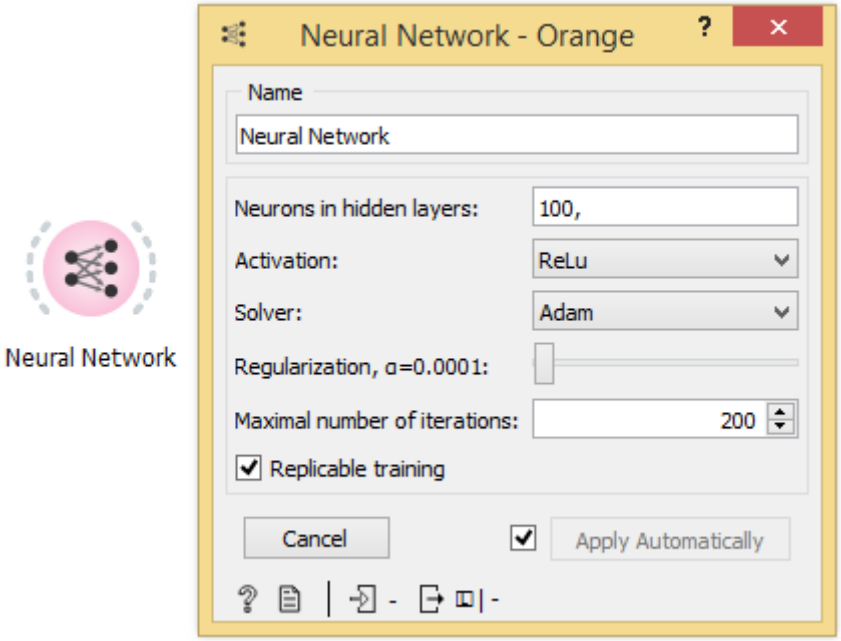
5.1.1 Resultados

Para demonstrar os resultados foram gerados gráficos e tabelas, com o desempenho geral obtido por cada extrator, desempenho geral gerado pelos classificadores considerando as métricas: CA, F1-*score*, *Precission*, *recall*.

5.1.2 Rede Neural

Para a RNA foi considerada a variação da quantidade de neurônio com duas camadas com 70 e 80, as funções de ativação Identidade, logística, tangente hiperbólica e Relu, os otimizadores de código L-BFGS-B (Liu e Nocedal, 1989), SGD (Inage e Hebishima, 2021) e Adam (Kingma e Ba, 2014) todos os testes feitos com 2.000 épocas de treinamento.

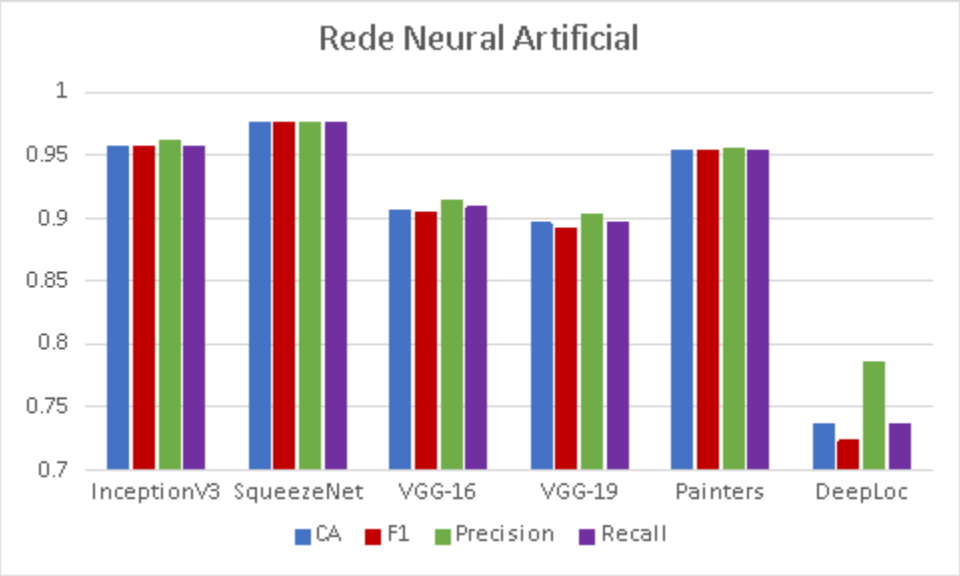
Figura 8 - Widget ambiente Orange - Rede Neural Artificial



Fonte: Orange 3.32.0 (2022).

O widget da "*Neural Network*" ou Rede neural é ilustrado na Figura *8*, neste teste foram variados o número de neurônio, quantidade de camadas "*Neurons in hidden layers*", a função de ativação "*Activation*", o otimizador de código "*Solver*" e o Número de interação "*Maximal number of interations*".

Figura 9 - Média de Desempenho obtido pelo Algoritmos Rede Neural Artificial



Fonte: Produção do autor (2022).

A Figura *9* - Média de Desempenho obtido pelo Algoritmos Rede Neural Artificial demonstra os resultados gerados pela rede neural considerando os 6 extratores usando no experimente e as métricas CA, F1, *Precision* e *Recall*. Neste experimento os extratores que obtiveram o desempenho mais alto foram o SqueezNet e InceptionV3 com valor entre 95% e 97% de acerto.

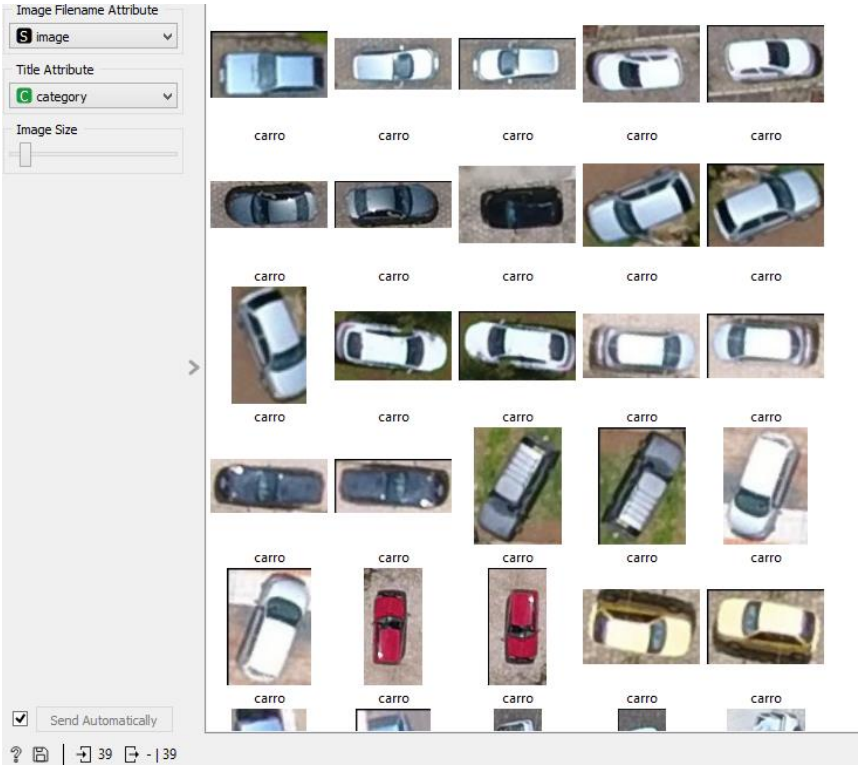
Figura 10- Matriz de confusão gerada pelo algoritmo de Rede Neural Artificial

		Predicted				
		arvore	**carro**	**casa**	**predio**	**Σ**
Actual	**arvore**	41	0	0	0	**41**
	carro	0	39	1	0	**40**
	casa	0	0	38	2	**40**
	predio	0	0	9	17	**26**
	Σ	**41**	**39**	**48**	**19**	**147**

Fonte: Orange 3.32.0 (2022).

A matriz de confusão da Figura *10*, ilustra o desempenho obtido pela rede neural. Esta matriz foi obtida combinando os seguintes parâmetros do algoritmo: uma camada com 70 neurônios, função de ativação tangente hiperbólica, otimizador de código Adam e 2000 épocas de treinamento. Os resultados da matriz demonstram que a rede classificou muito bem as classes arvore, carro e casa sendo que a quantidade classificada foi respetivamente 41, 39 e 38. Só classificando de forma errônea 9 prédios que os classificou como casa.

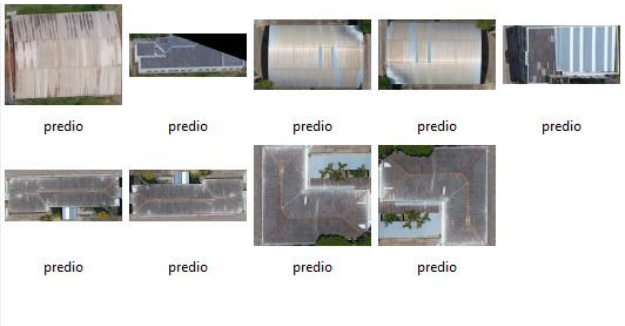
Figura 11 - Exemplo de classificação correta gerado pela Rede Neural Artificial



Fonte: Orange 3.32.0 (2022).

Um exemplo de classificação obtida pela rede é ilustrado na Figura *11*, onde o algoritmo classificou corretamente 39 imagens de carro.

Figura 12 - Exemplo de erro de classificação Rede Neural Artificial



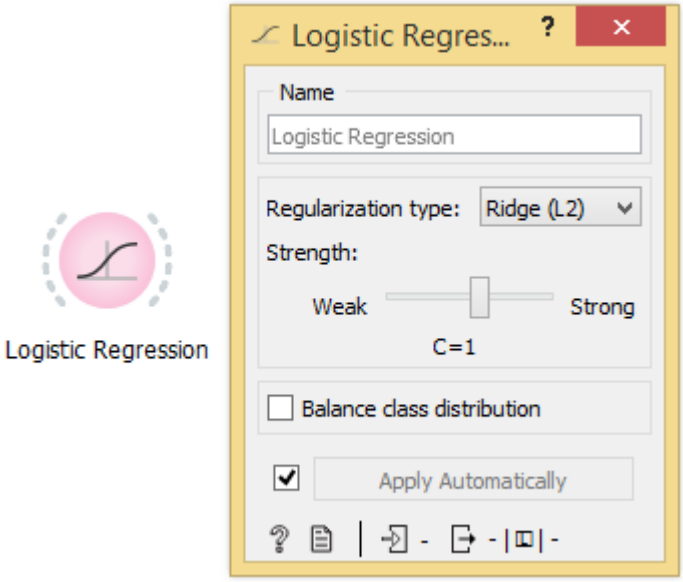
Fonte: Orange 3.32.0 (2022).

A maior quantidade de classificação errada foi obtida pela classe prédio, como é ilustrado na Figura 12, onde a rede classificou 9 prédios como casa.

5.1.3 Regressão Logística

A Regressão Logística é um algoritmo estatístico que busca a correlação entre variáveis categóricas por meio de probabilidade (M.D, 1944). O modelo de regressão multinomial é aplicado para classificar três ou mais categorias. Na técnica de Regressão logística se usou a variação do regularizador *Lasso*(L1) e *Ridge*(L2) (Kumar e Sahu, 2021), o nível de força para o regularizador 0,5 / 1 / 200 / 500 e 1000.

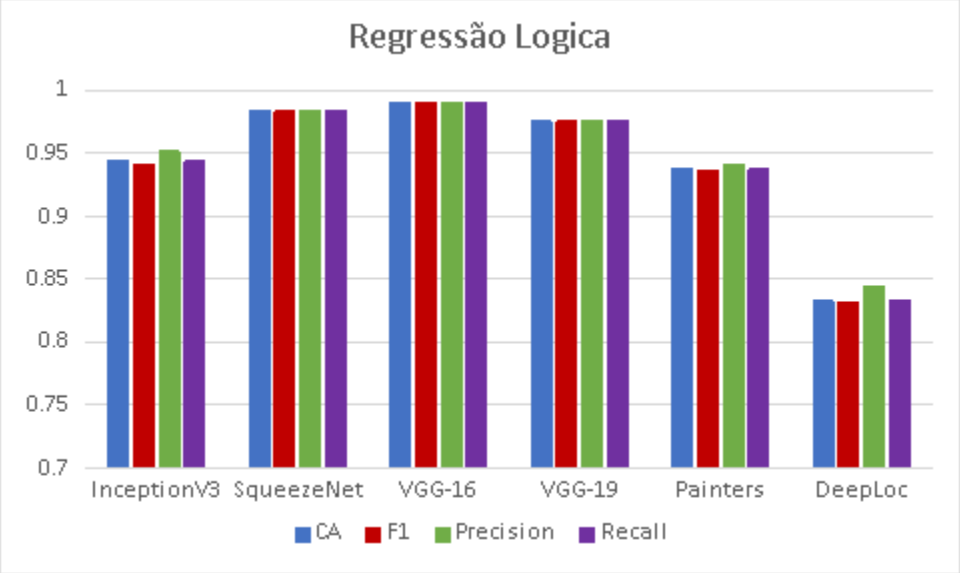
Figura 13 - Widget ambiente Orange - Regressão Logística



Fonte: Orange 3.32.0 (2022).

O widget "*Logistic Regression*" é usado para o teste da regressa logística, é ilustrado na Figura 13. Os parâmetros usados foram o tipo de regularizador "*Regulazatin Type*" e o nível de relevância para o regularizador "*Strength*".

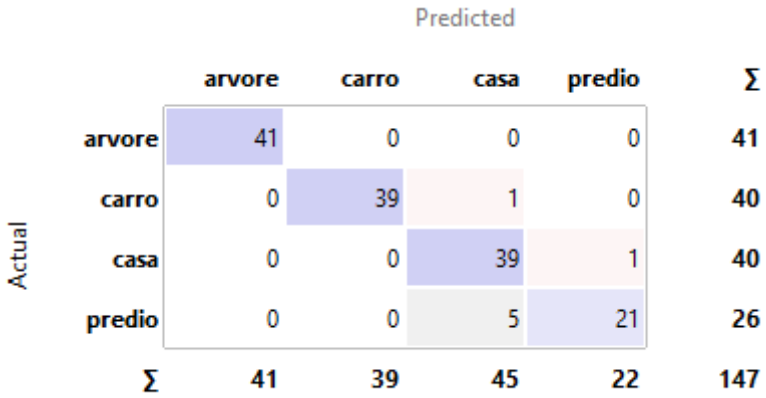
Figura 14 - Média de Desempenho obtido pelo Algoritmos Regressão Logística



Fonte: Produção do autor (2022).

Os resultados dos experimentos com a regressão logística são ilustrados na Figura 14, onde é possível demonstrar que o extrator *VGG*-16 possui o melhor resultados no contesto deste experimento, com acerto de até 99%.

Figura 15 - Matriz de confusão gerada pelo algoritmo de Regressão Logística

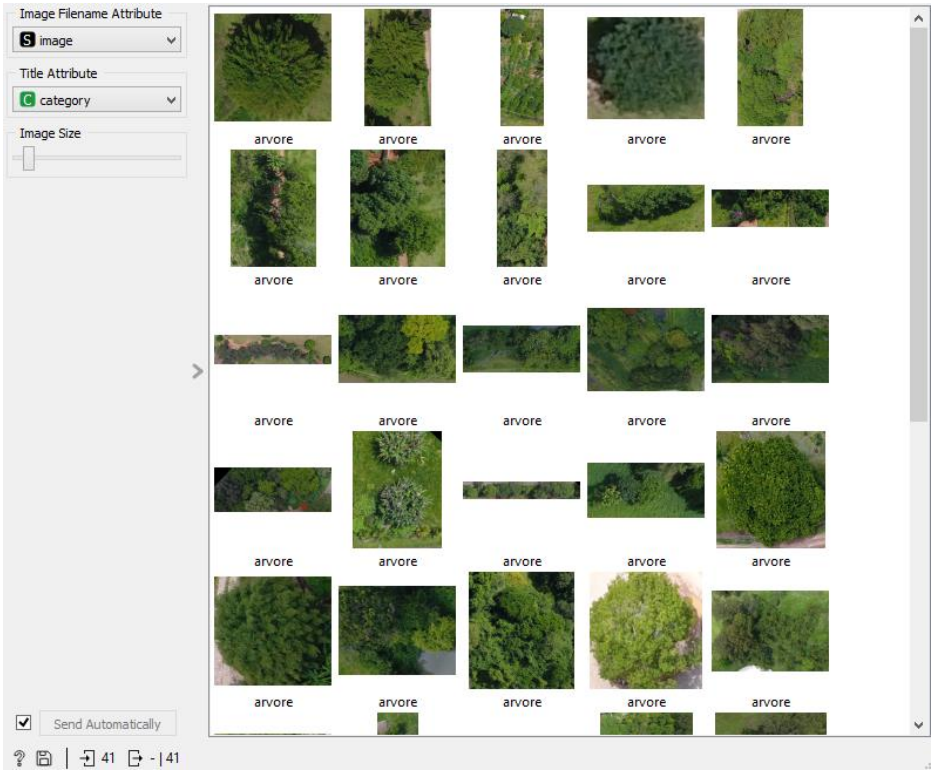


Fonte: Orange 3.32.0 (2022).

Os resultados presentes na matriz de confusão contidos na Figura 15 foram gerados pelo algoritmo de Regressão Logística, esta matriz foi gerada pela combinação dos parâmetros: regularizar *Ridge*(L2) e o nível de força do regularizado em 200. Os resultados mostram que a Regressão Logística conseguiu uma boa classificação onde acertou 41 casas, 39 carros, 39 casas e 21. Neste experimento o algoritmo só confundiu 1 carro com casa, 1 casa com prédio e 5

prédios com casa. Abaixo é demonstrado alguns dos acertos e erros gerados na classificação das imagens.

Figura 16 - Exemplo de classificação correta gerado pela Regressão Logística



Fonte: Orange 3.32.0 (2022).

A Figura 16 demonstra alguns dos resultados gerados pela classificação da classe arvore, onde o algoritmo conseguiu classificar corretamente 41 imagens de árvore.

Figura 17- Exemplo de erro de classificação Regressão Logística

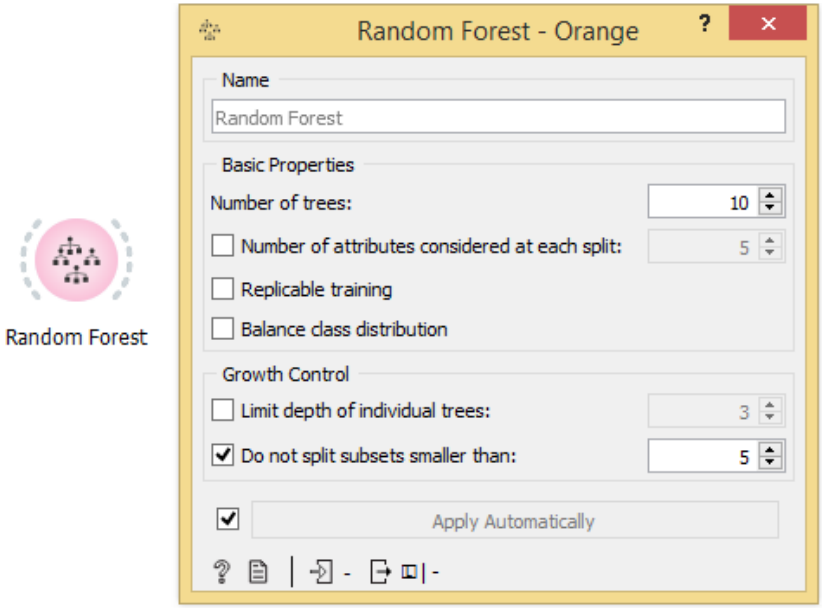
Fonte: Orange 3.32.0 (2022).

A classe que obteve a maior quantidade de erro foi a classe prédio onde o modelo classificou 5 prédios com casa, isso demonstrado na Figura 17.

5.1.4 Floresta Aleatória

Para os testes com a floresta aleatória foram considerados os seguintes parâmetros: o número de árvores variou em 50, 100 e 150 a limitação de divisão por cada nó da árvore em 20, 30, 40 e 50, limite de profundidade das árvores em 20, 30, 40 e 50, e a limitação de divisão de subconjunto que não seja menor que 5.

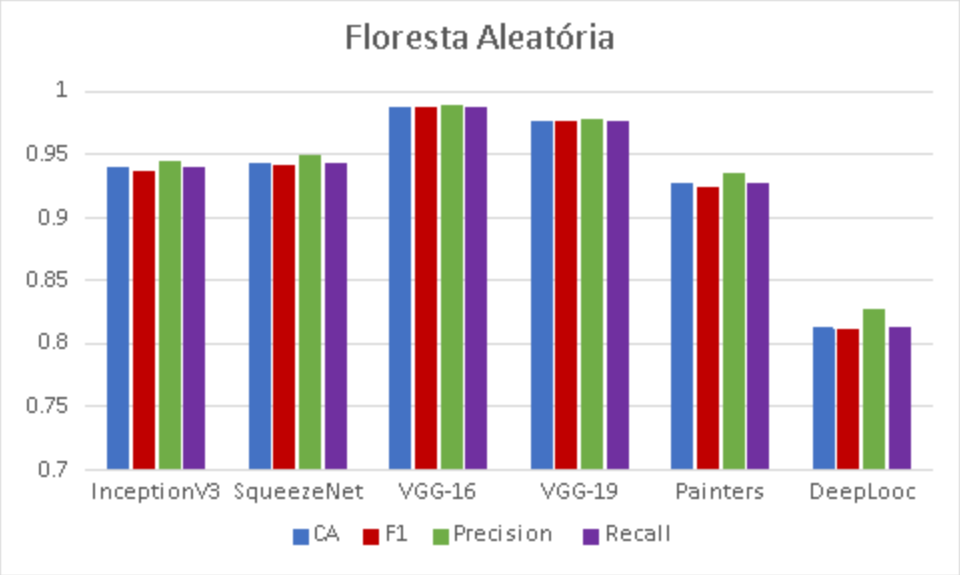
Figura 18 - Widget ambiente Orange – Floresta Aleatória



Fonte: Orange 3.32.0 (2022).

No experimento com a floresta aleatória foi usado o *widget* "*Random Forest*". Os parâmetros usados no experimento são demonstrados na Figura 18, o número de árvore "*Number of trees",* limitação de divisão por cada nó da árvore "*Number of atributes*.", limite de profundidade das árvores "*Limit depth of...*" e a limitação de divisão de subconjunto que não seja menor que "*Do not split subsets...*".

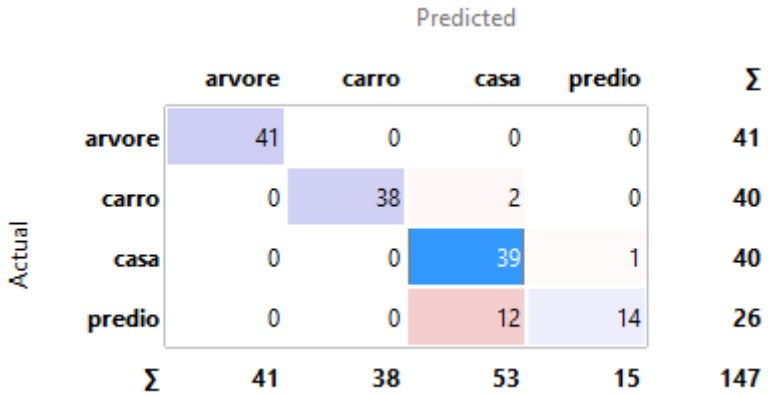
Figura 19 - Média de Desempenho obtido pelo Algoritmos Floresta aleatória



Fonte: Produção do autor (2022).

Os resultados dos experimentos com a floresta aleatória são ilustrados na Figura 19, ao analisar a figura chegamos à conclusão que o melhor extrator neste experimento foi a *VGG*-16 com o melhor desempenho se comparado com os demais extratores.

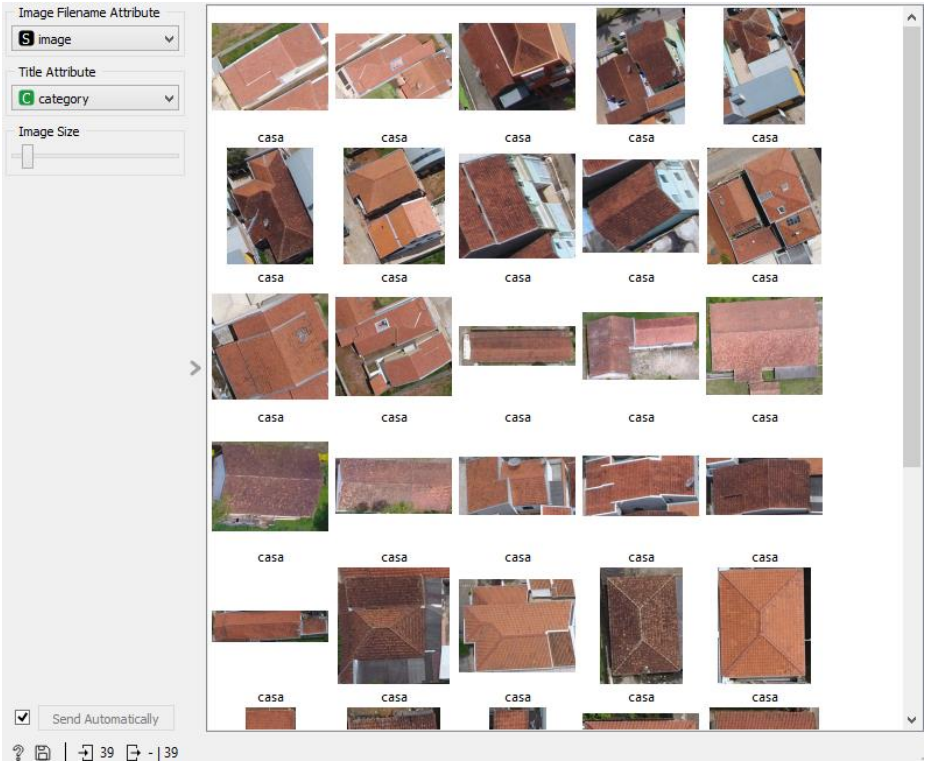
Figura 20 - Matriz de confusão gerada pelo algoritmo de Floresta Aleatória



Fonte: Orange 3.32.0 (2022).

A matriz de confusão presente na Figura 20 apresenta os resultados obtidos pela Floresta aleatória, considerando os seguintes parâmetros: número de árvore 150, *split* 20, limite profundidade 20 e o limite de subdivisão 5. Assim como os demais algoritmos, a floresta alatoaria classificou muito bem as 41 imagens de árvores, 38 carros e 39 casas e 14 de carro. O algoritmo confundiu 2 carros com casa, 1 casa com prédio e 12 prédio com casa.

Figura 21 - Exemplo de classificação correta gerado pela Floresta Aleatória



Fonte: Orange 3.32.0 (2022).

Alguns exemplos de classificação feita pela floresta aleatória são ilustrados na Figura 21, que contém as classificações de imagens de casas onde a rede classificou corretamente 39 imagens.

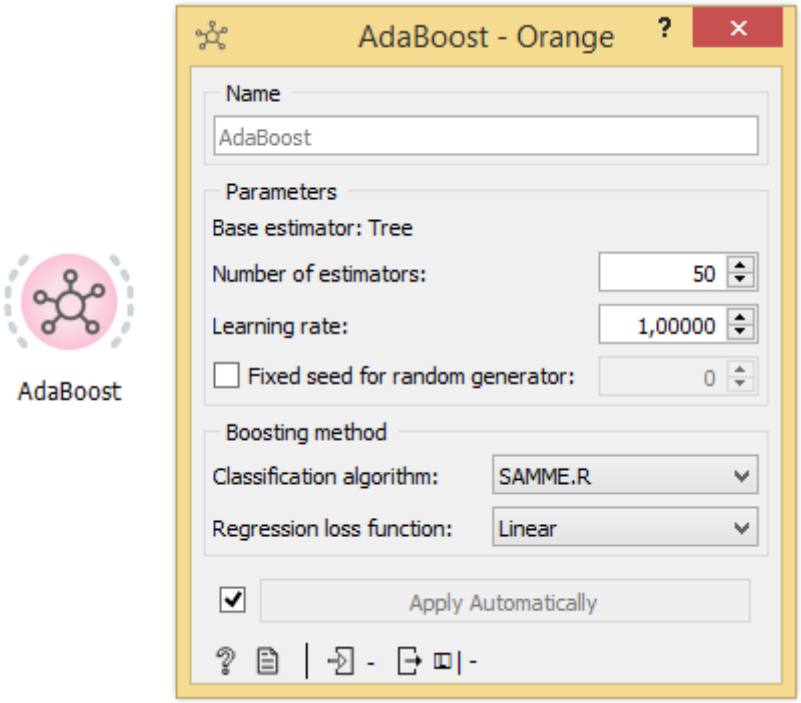
Figura 22 - Exemplo de erro de classificação Floresta Aleatória

Exemplo da classificação errada gerado pelo algoritmo floresta aleatória é ilustrando na Figura 22, onde o algoritmo classificou 12 imagens de prédio como casa.

5.1.5 *Adaboost*

O *AdaBoost* (Freund e Schapire, 1995) é um algoritmo de aprendizado de máquina desenvolvido por Yoav e Robert. Neste modelo, vários classificadores de baixo desempenho são combinados para gerar um classificador de alto desempenho. Os parâmetros usados no modelo *AdaBoost* (Freund e Schapire, 1995) foram o número de estimador, uma árvore com variação de 20, 30, 40, 50 e 100, o algoritmo de classificação *SAMME* e *SAMME.E* (Zhu et all, 2006) para estimar os valores dos pesos e as funções de perda de regressão linear, quadrada e exponencial.

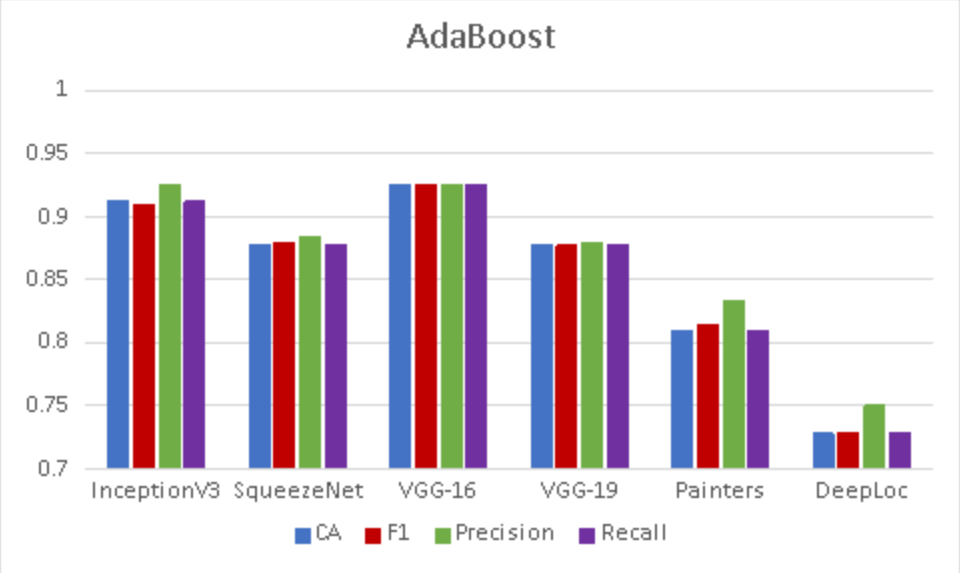
Figura 23 - Widget ambiente Orange - AdaBoost



Fonte: Orange 3.32.0 (2022).

O *widget* usado no experimento com o *AdaBoost* é ilustrado na Figura 23, os parâmetros variados foram o número de estimador "*Number of estimators*", algoritmo de classificação "*Classification algotithm*" e Função de perde "*Regression loss function*".

Figura 24 - Média de Desempenho obtido pelo Algoritmos AdaBoost



Fonte: Produção do autor (2022).

O desempenho obtido pela combinação dos extratores junto ao algoritmo *adaboost* é ilustrado na Figura 24, os resultados demonstraram que o *adaboost* obteve o melhor resultado com o extrator *VGG*-16. Obtendo um acerto acima de 90% se comparado aos demais extratores.

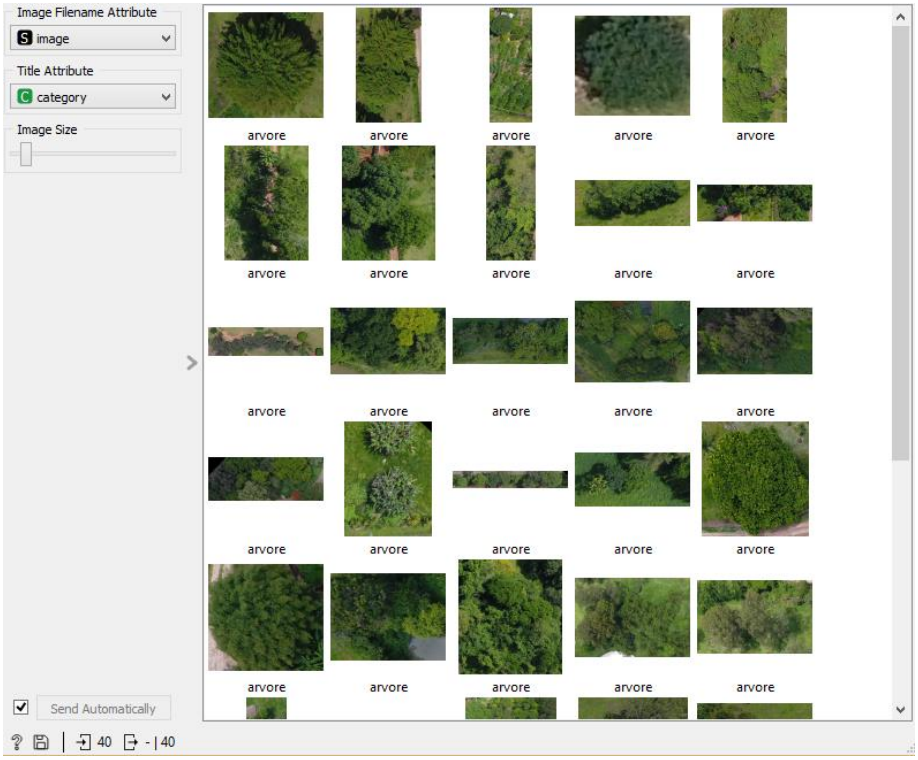
Figura 25 - Matriz de confusão gerada pelo algoritmo de AdaBoost

		Predicted				
		arvore	**carro**	**casa**	**predio**	**Σ**
Actual	**arvore**	40	1	0	0	**41**
	carro	0	34	6	0	**40**
	casa	0	1	34	5	**40**
	predio	0	0	14	12	**26**
	Σ	**40**	**36**	**54**	**17**	**147**

Fonte: Orange 3.32.0 (2022).

A Figura 25 ilustra a matriz de confusão gerada pela classificação do algoritmo *Adaboost*. Para este teste foram usando os seguintes parâmetros: número de estimador 20, algoritmo de classificação SAMME e função de perde linear. Neste teste o algoritmo classificou corretamente 40 arvores, 34 carros, 34 casas e 12 prédios, confundiu 1 arvore como carro, 6 carros como casa, 5 casas com prédio e 14 prédios com casa.

Figura 26 - Exemplo de classificação correta gerado pela Adaboost



Fonte: Orange 3.32.0 (2022).

Na classificação de imagens da classe arvore o algoritmo *Adaboost* conseguiu classificar muito bem, acertando um total de 40 imagem como é ilustrado na Figura 26.

Figura 27 - Exemplo de erro de classificação Adaboost

Fonte: Orange 3.32.0 (2022).

O algoritmo só teve maior dificuldade para detectar as imagens de prédio, confundindo 14 imagens de prédio com carro. Os erros de classificação são demonstrados na Figura 27.

Figura 28 - Resultados obtidos pelos extratores e classificadores

Extract	Classifier	CA	F1	Prec	Rec
InceptionV3	ANN	0.986	0.986	0.987	0.987
SqueezNet	ANN	0.966	0.996	0.967	0.966
VGG-16	ANN	1.000	1.000	1.000	1.000
VGG-19	ANN	0.993	0.988	0.976	1.000
Painters	ANN	0.973	0.973	0.975	0.973
DeepLoc	ANN	0.816	0.815	0.828	0.816
InceptionV3	LogisticR	0.946	0.946	0.955	0.946
SqueezNet	LogisticR	0.993	0.993	0.993	0.993
VGG-16	LogisticR	1.000	1.000	1.000	1.000
VGG-19	LogisticR	0.993	0.993	0.993	0.993
Painters	LogisticR	0.966	0.966	0.967	0.966
DeepLoc	LogisticR	0.884	0.883	0.888	0.884
InceptionV3	RandomF	0.973	0.972	0.975	0.973
SqueezNet	RandomF	0.952	0.951	0.959	0.952
VGG-16	RandomF	0.993	0.993	0.993	0.993
VGG-19	RandomF	0.993	0.993	0.993	0.993
Painters	RandomF	0.932	0.936	0.944	0.939
DeepLoc	RandomF	0.823	0.822	0.837	0.823
InceptionV3	AdaBoost	0.912	0.910	0.925	0.912
SqueezNet	AdaBoost	0.878	0.880	0.884	0.878
VGG-16	AdaBoost	0.925	0.925	0.926	0.925
VGG-19	AdaBoost	0.878	0.877	0.879	0.878
Painters	AdaBoost	0.810	0.815	0.833	0.810
DeepLoc	AdaBoost	0.728	0.729	0.750	0.728

Fonte: Produção do autor (2022).

A Figura 28 apresenta a média dos resultados obtidos por todas as combinações de extratores e classificadores, considerando as métricas descritas na Seção 3 de materiais e métodos. Analisando a Figura 28 percebemos que, de uma maneira geral, as combinações de extrator e classificador tiveram bom desempenho. Em todas as métricas (CA, F1, Prec e Rec), em 17 de 24 possibilidades, o valor da medida foi maior que 0,90. Isso dá um desempenho positivo global de 70,83%. Em particular, as combinações *VGG*-16 + *LogisticR* (regressão logística) e *VGG*-16 + RNA, obtiveram o valor máximo (1,00) em todas as métricas.

5.2 Abordagem - 2

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: *VGG*-16 e *EfficientNetB0* como extrator de características das imagens para criar o conjunto de treinamento e teste. Como classificador foram usadas a 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 anteriores usando o ambiente Orange e os resultados gerados na abordagem 2. 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.

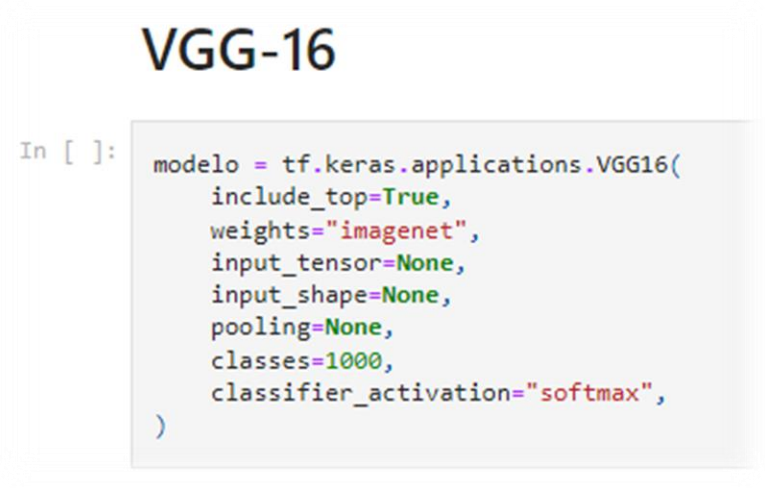
5.2.1 Resultados

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*.

5.2.2 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 *VGG*-16 e *EfficientNetB0*. A extração das características resultou em 2 arquivos do tipo CSV contendo os dados extraídos pela rede neural convolucional.

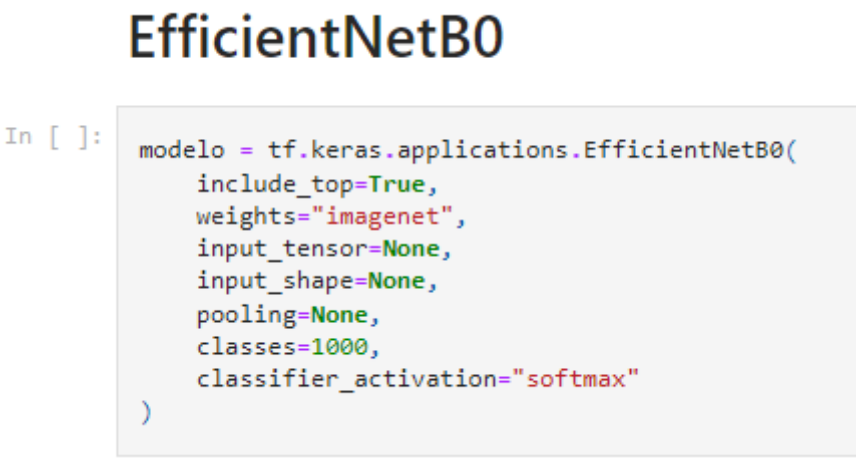
Figura 29 - Código Keras para modelo VGG-16



Fonte: Google Colab (2022).

A Figura 29 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*.

Figura 30 - Código Keras para modelo EfficientNetB0



Fonte: Google Colab (2022).

A Figura 30 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*".

5.2.3 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 31 - 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 31.

Os melhores resultados obtidos por cada extrator no experimento com a rede neural artificial são ilustrados na Figura 32.

Figura 32 - Melhor resultado obtido pela junção da CNN junto a Rede Neural artificial

Incorporador	Nº Neuronios	func_ativação	solver	iteração	F1	Precision	Recall
VGG-16	80, 70	ReLu	L-BFGS-B	2000	0,9685	0,9685	0,9693
EfficientNetB0	80, 70	Identity	Adam	2000	0,9922	0,9916	0,9929

Fonte: Produção do autor (2022).

A Figura 32 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*. Ao analisar a Figura 32 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%.

A avaliação do desempenho obtido da rede neural artificial sobre as características extraídas pelas CNNs, estão contidos na Tabela 2. Esta tabela é composta por 4 colunas, denominada CNN, F1, *Precision* e *recall.*

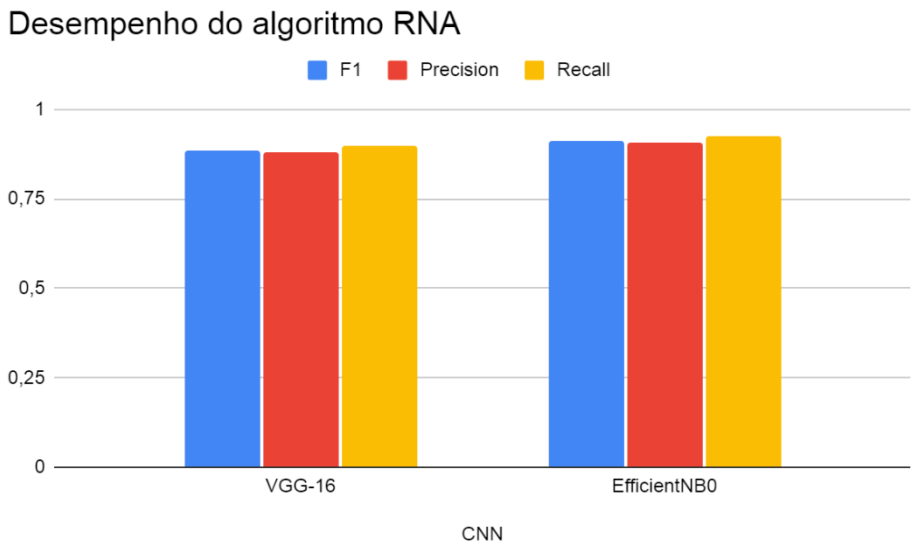
Tabela 2 - Resultado da combinação CNN + Classificador Rede Neural Artificial

MLP: Rede Neural Artificial			
CNN	F1	*Precision*	*Recall*
VGG-16	0,8852	0,8819	0,8985
EfficientNB0	0,9119	0,9087	0,9235

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%, isso é demonstrado na Tabela 2.

Figura 33 - 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 33 com o gráfico de desempenho da RNA, onde o extrator *EfficientNetB0* obteve um desempenho superior ao *VGG*-16.

Tabela 3 - Media geral de desempenho obtidas Classificador Rede Neural Artificial

ML: Media Geral - Rede Neural Artificial			
F1	*Precision*	*Recall*	Média
0,8689	0,8692	0,8833	0,8738

Fonte: Produção do autor (2022).

O resultado em um contexto geral obtido pela CNN está disposto na Tabela 3, esses resultados demonstras que a RNA obtém um desempenho de 87% de acerto considerando a média entre os extratores.

5.2.4 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 34 - 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 34 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 35 - Os 5 Melhores Resultado Obtidos pela Máquina de Vetores de Suporte – SVM

CNN	C	Kernel	Gama	Degree	f1-score	Precision	Recall
VGG-16	0,25	linear	auto	3	0,9408	0,9375	0,9460
EfficientNetB0	1	linear	auto	3	0,9949	0,9943	0,9956

Fonte: Produção do autor (2022).

Os melhores resultados de desempenho de cada CNN estão ilustrados na Figura 35, nesta figura contém a CNN os paramentos 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.

A avaliação de desempenho por cada CNN está contida na Tabela 4 - Resultado da combinação CNN + Classificador Máquina de Vetores de Suporte – SVM, nesta tabela contém as colunas e as métricas: f1-*score*, *Precision*, *Recall*.

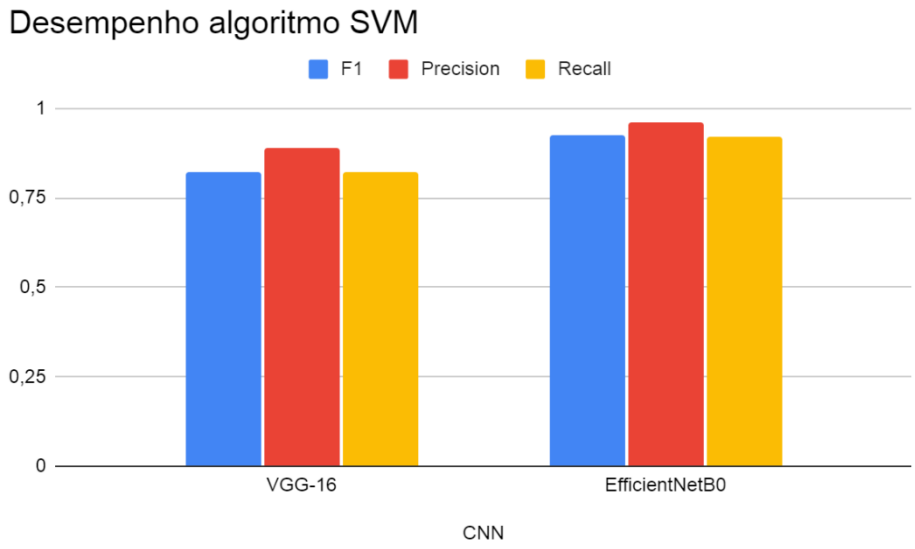
Tabela 4 - Resultado da combinação CNN + Classificador Máquina de Vetores de Suporte – SVM

ML-SVM				
CNN	F1-*score*	*Precision*	*Recall*	Média
VGG-16	0,8216	0,8905	0,8211	0,8444
EfficientNetB0	0,9260	0,9607	0,9222	0,9363

Fonte: Produção do autor (2022).

Os resultados contidos na Tabela 4 demonstram que a *EfficientNetB0* tem um desempenho médio de 93.6%.

Figura 36 - Desempenho Médio obtido pela Máquina de Vetores de Suporte - SVM



Fonte: Produção do autor (2022).

Os desempenhos gerados pelo algoritmo SVM são estão ilustrados na Figura 36, neste teste é possível notar que assim como no experimento com a RNA o *EfficcientNetB0* obteve um resultado superior ao *VGG*-16.

Tabela 5 - Media de desempenho obtida pela SVM

ML: Media Geral - Rede Neural Artificial			
f1-*score*	*Precision*	*Recall*	Média
0,8738	0,9256	0,8717	0,8904

Fonte: Produção do autor (2022).

Uma análise geral do desempenho obtido pela SVM é interessante, pois assim, é possível compara com a RNA e descobrir qual modelo melhor performou. Os resultados obtidos pela estão condidos na Tabela 5, a RNA obteve um desempenho médio de 87% já a SVM obteve um desempenho de 89%.

6 CONCLUSÃO

No relatório presente, foram apresentadas as atividades desenvolvidas no período de 1 agosto de 2021 a 31 agosto de 2022, referente ao projeto "Emprego De Inteligência Artificial Na Escolha Automática De Algoritmos E Parâmetros De Técnicas De Processamento De Imagens Obtidas Por Drones Para Aplicação No Sensoriamento Remoto". O objetivo da pesquisa era investigar a classificação de imagens obtidas por drones utilizando IA e técnicas de processamento de imagens. 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: *InceptionV3*, *SqueezeNet*, *VGG*-16, *VGG*-19, *Painters* e *DeepLoc*. Como classificadores, foram usados: *Adaboost*, Floresta Aleatória, Regressão logística e Rede Neural Artificial. Os resultados demonstram que o extrator *VGG*-16 com o classificador Regressão Logística tiveram os melhores resultados, se comparados com as demais técnicas, atingindo acerto de até 100% nas métricas Acurácia, F1-*score*, *Precision* e *Recall*. Em uma segunda etapa da pesquisa, iniciou-se um processo onde foram comparadas as seguintes CNNs como extratores de características: *VGG*-16 e *EfficientNetB0*. E, nessa fase, os seguintes classificadores foram usados: Rede Neural Artificial e Máquina de Vetores de Suporte. Os resultados demonstram que o extrator *EfficientNetB0* com o classificador SVM tiveram os melhores resultados, obtendo um F1-*Score* de 99,5%. Sendo assim, pode-se dizer que as atividades previstas foram realizadas satisfatoriamente sendo os objetivos alcançados. Como trabalho futuro, é proposto dar continuidade ao experimento 2 com mais redes CNNs e algoritmos tradicionais de Aprendizado de Máquina.

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.

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.

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.

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: **Error! Hyperlink reference not valid.**. Acesso em: 8 fev. 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.

Paulino, Ângelo; Guimarães, Lamartine ;Shiguemori, Elcio. (2020). Assessment of Noise Impact on Hybrid Adaptive Computational Intelligence Multisensor Data Fusion Applied to Real-Time UAV Autonomous Navigation. IEEE Latin America Transactions. 18. 295-302. 10.1109/TLA.2020.9085283.

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.

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.

FORNARI, Gabriel; Júnior, Valdivino; Shiguemori, Elcio. . Aracterização 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.

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).

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.

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.

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.

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.

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.

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.

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

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.

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.

ORANGE Data Mining. In: Orange é um kit de ferramentas de visualização de dados, aprendizado de máquina e mineração de dados de código aberto. 3.31.1, 2003. Disponível em: https://orangedatamining.com. Acessado em 18 ago 2022.

IEEE Latin american transactions. In: O IEEE Latin America. 3.31.1, 2021. Disponível em: https://latamt.ieeer9.org/index.php/transactions. Acessado em 18 ago 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. https://doi.org/10.1590/s0103-4014.2021.35101.007.

Y. Freund e R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” 1995.

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.

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.

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

D. C. Liu e J. Nocedal, “On the limited memory bfgs method for large scale optimization,” Mathematical Programming, vol. 45, pp. 503–528, Aug 1989.

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

D. P. Kingma e J. Ba, “Adam: A method for stochastic optimization,” 2014.

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.

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

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.

F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, e K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size,” 02 2016.

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.

Ló, Thiago Berticelli. Mapeamento de uso e cobertura da terra utilizando sensoriamento remoto e redes neurais convolucionais. 2022. 164 f. Tese (Doutorado em Engenharia Agrícola) - Universidade Estadual do Oeste do Paraná, Cascavel, 2022.

N. Ilenic, “Painters painter by numbers competition on kaggle,” 2016.

J. Almagro Armenteros, C. Sønderby, S. Sønderby, H. Nielsen, e O. Winther, “Deeploc: prediction of protein subcellular localization using deep learning,” Bioinformatics, vol. 33, no. 21, pp. 3387–3395, 2017.

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