The present work has two main goals. The first is to develop a script that identifies military aircrafts on images. Secondly, being able to classify the identified aircrafts into several aircraft types. The classification phase uses two approaches, training a CNN from scratch and using transfer learning. To carry out these objectives, we resorted to the python programing

Identification and Classification of Military Aircrafts

UC: Aprendizagem Profunda para Visão por Computador

Professor:
Tomás Brandão

Students:
Beatriz Santos, nº 108593
João Ferreira, nº 88639
Rodrigo Sarroeira, nº 92761

ISCTE – Instituto Universitário de Lisboa

iscte
BUSINESS SCHOOL

Data: 27 de janeiro de 2023

Index

1. Introduction

Military aircraft identification in images using deep learning and convolutional neuronal networks (CNN) is a crucial task in modern defense systems. With the increasing number of aircrafts in the sky, the need for efficient and accurate identification methods has become paramount. The use of deep learning and CNN has shown to be an effective approach for aircraft identification in images. In this paper, we propose a method for military aircraft identification and classification in images using deep learning and CNN. These kinds of networks are used to find features in images and recognize patterns between them.

The proposed method is evaluated on a dataset of real-world military aircraft images and the results demonstrate the effectiveness of our approach. This dataset was obtained in Kaggle website [X]. This dataset is composed of 14 400 images that contain at least one military aircraft. Originally, the images are grouped into 40 models of aircrafts. During the development of the project. these images will be grouped, once again, into 8 categories of military aircraft, which will represent a complexity reduction for the deep learning models. The categories are the following: Ataque, Bombardeiro, Busca e salvamento, Caça, Caça-Bombardeiro, reconhecimento, and transporte.

For each image in the dataset there is a comma separated values file (CSV) associated with it. This file contains fundamental information for the object identification phase. Each file contains information relative to the position of the aircrafts on the image, such as the pixel coordinates, widths, and heights of the object. This will allow evaluating the predictions of the object identifier algorithm with the actual position of the objects.

This study will be developed resorting to the python programming language, more specifically to the TensorFlow, Keras, and CV2 packages, that enable working with deep learning (DL) applied to images.

2. Methodology

In this section, the methods applied in the development of this study are presented. Firstly, the data description phase is presented in subsection 3.1, allowing understanding the data and its formats. Also, problems regarding the data can be detected in this phase. Subsection 3.2 contains the data transformations applied to the data, to prepare it for the models. Oversampling and data augmentation is applied to the data, given that number of observations of each class was very unbalanced. Once the data is prepared, the object detection phase is carried out. In this second phase, several CNN models are experimented with the goal of identifying military aircraft on images (subsection 4.1). To evaluate the performance of the different applied models, several metrics are calculated and considered, these are: precision, recall, and intersection over union. The comparison between the models is presented in subsection 4.2. After identifying the position of the object in the image, the goal is to classify it into 8 classes of military aircrafts. To solve this classification problem, we resort once again to CNN's. Section 5 presents the training and evaluation of 2 CNN's. Subsection 5.1 presents the training of two CNN's trained from scratch, while subsection 5.2 presents the same information for two CNN's using transfer learning. Finally, subsection 5.3 presents the evaluation of both networks and compares them.

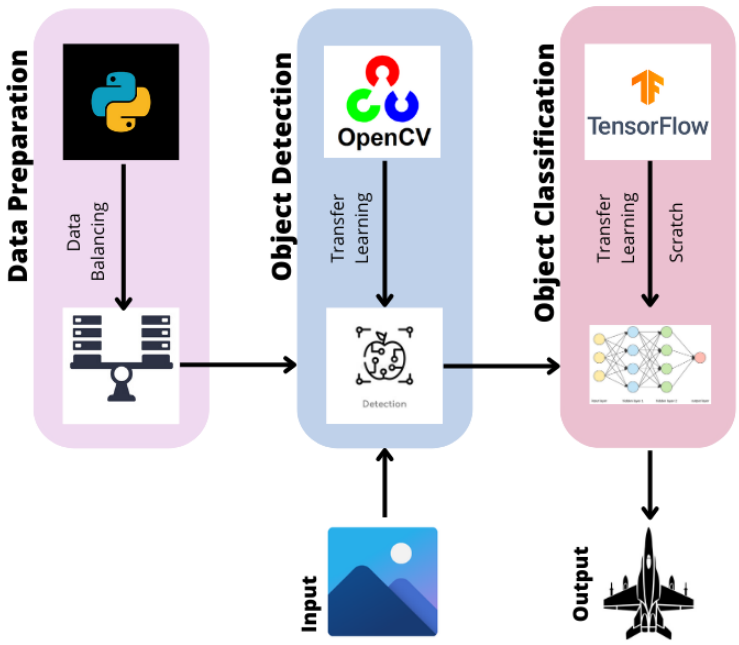

Figure 1. Methodology flowchart

3. Data Preparation

The original dataset contains images relative to 40 models of military aircraft. Given that the second goal of this study is to efficiently classify military aircrafts, a dimensionality reduction is going to take place. The 40 models will be grouped into 7 classes, these are: Ataque, Bombardeiro, Busca e salvamento, Caça, Caça-Bombardeiro, reconhecimento, and transporte. The two bar charts bellow (figure 2) present the distribution of images for the 40 models and the 7 classes.

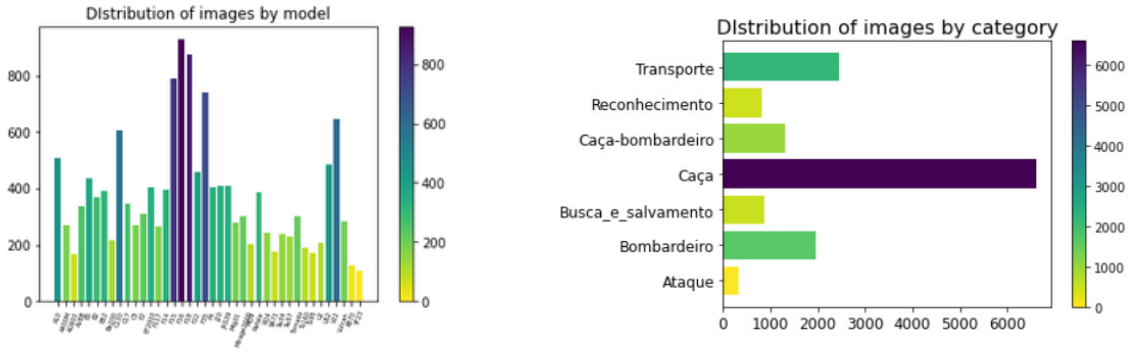

Figure 2. Bar charts presenting the proportion of images of each model and category.

It is possible to observe, in both cases, that the number of images is unbalanced. To solve this problem, we will resort to two techniques to balance the number of images per category. The first method is under sampling, in which observations from the dominant category are randomly removed. In this case, the dominant category is "Caça" and it contains 6600 images, we will start by reducing this number to 3000. Secondly, we will do the opposite, oversampling, that consists of increasing the number of observations of the non-dominant classes. To achieve this goal the following method will be applied. For each class, randomly select images and perform flip or rotation operations, until the number of images in that class reaches 3000, by that time the dataset will be balanced. The flip and rotation operations are randomly performed, following probability distributions. The vertical flip, the horizontal flip, and the rotation, all have the same probability, 33.3%. For each image, samples from a uniform distribution between 1 and 0 (figure 3 - left) are generated, if the value is greater than 2/3, then the correspondent operation takes place. For the rotation operation, the degree of the rotation is also given by a probability distribution, this time using a normal distribution of mean 0 and standard-deviation 10 (figure 3 - right).

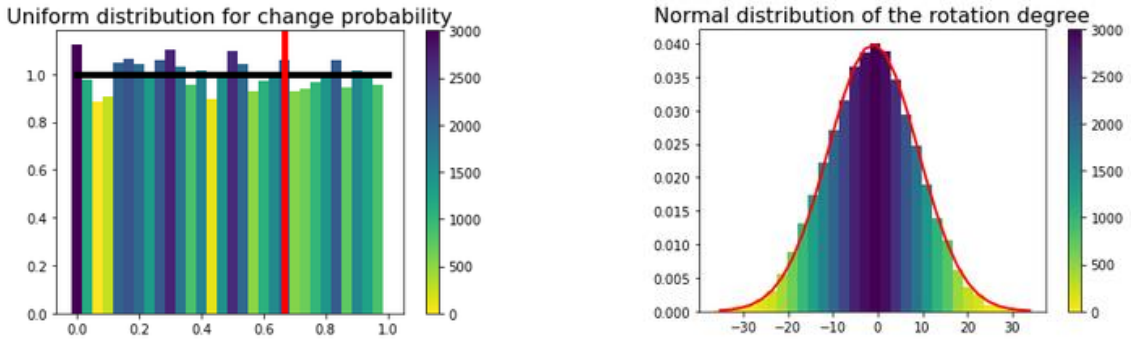

Figure 3. Random distributions of the transformation parameters

4. Military aircraft: Identification

4.1 Workflow

To identify military aircraft on images, we resort to a pre-trained object identification neuronal network that was trained to identify 90 classes of objects, one of them being airplanes. We resort to transfer learning given its simplicity and relation to our problem. The model in use is the SSD_MobileNet_V2, known in the area for generating accurate results. Although there is a strict relation between our problem and the data with which the network was trained, there still is a big gap in terms of results. One way of avoiding this gap would be to train our own object identifier given the dataset at hand.

4.2 Results

To evaluate the performance of the object identifier, several metrics are taken into consideration, such as the intersection over union (IoU), the precision, and the recall. The average IoU obtained was of 0.25, this value means that ¾'s of the area identified as being a plane wasn't actually one. To calculate the recall and precision metrics, the number of true positives (1159), false positives (6141), and false negatives (2055), were calculated. The precision value was of 0.159, while the recall was of 0.361. In a future continuation of this study, this section would be the most promising one to improve. We believe that training a object identifier to specifically identify the 7 classes of airplanes would drastically improve the 3 metrics presented above.

5. Military aircraft: Classification

5.1 Scratch CNN

To classify military planes, we firstly decided to train two CNN from scratch. In these CNN, we used Neural Networks that are widely known in the area (MobileNet and EfficientNetB0), but we set the "*model.trainable*" to True so we could train each NN's weights according to our problem.

5.1.1 EfficientNetB0

EfficientNetB0 is a convolutional neural network (CNN) architecture that is designed to be both efficient and effective for image classification tasks. It is a member of the EfficientNet family of models developed by Google, which are characterized by their use of compound scaling to improve both the depth and width of the network while keeping computational costs low. EfficientNetB0 is the smallest model in the EfficientNet family, with a lower number of parameters and lower computational cost compared to the other models. However, it still achieves high accuracy on image classification tasks, making it a good choice for resource-constrained settings such as mobile devices or edge devices.

As mentioned before, in this section we decided to train each neural network from scratch. Adding to the EfficientNetB0, we also added a few more layers to achieve a more suitable neural network for our problem. Figure 4 shows all the layers that the CNN has.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 224, 224, 3)       0

 random_flip (RandomFlip)    (None, 224, 224, 3)       0

 random_rotation (RandomRota  (None, 224, 224, 3)      0
 tion)

 efficientnetb0 (Functional)  (None, 7, 7, 1280)       4049571

 flatten (Flatten)           (None, 62720)             0

 dense (Dense)               (None, 256)               16056576

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 7)                 1799

=================================================================
Total params: 20,107,946
Trainable params: 20,065,923
Non-trainable params: 42,023
_________________________________________________________________
```

Figure 4. CNN summary with EfficientNetB0.

5.1.2 MobileNet

MobileNet is a convolutional neural network (CNN) architecture that is specifically designed to be efficient on mobile and embedded devices. It was developed by Google and published in a 2017 paper. The main feature of MobileNet is its use of depth wise separable convolutions, which are a type of convolution that separates the standard convolution operation into a depth wise convolution and a pointwise convolution. This allows the network to have a lower number of parameters and lower computational cost compared to other CNNs, while still achieving good accuracy on image classification tasks.

MobileNet also uses a technique called "width multiplier" to control the number of filters in the network, which allows for trade-offs between accuracy and computational cost. Additionally, it uses lightweight layers such as 1x1 convolutions and global average pooling to reduce the computational cost further. MobileNet has been used in many mobile and embedded applications, such as object detection, facial recognition, and image segmentation, and it has also been used as a feature extractor in transfer learning. There are many versions of MobileNet models, such as MobileNetV1, MobileNetV2, MobileNetV3 and so on, which are tailored to different types of devices and tasks. Adding to the MobileNet that we will include on one of ours CNN's trained from scratch, we also added a few more layers as we can see on the next picture.

Layer (type)	Output Shape	Param #
rescaling_2 (Rescaling)	(None, 224, 224, 3)	0
random_flip_1 (RandomFlip)	(None, 224, 224, 3)	0
random_rotation_1 (RandomRotation)	(None, 224, 224, 3)	0
mobilenet_1.00_224 (Functional)	(None, 7, 7, 1024)	3228864
flatten_1 (Flatten)	(None, 50176)	0
dense_2 (Dense)	(None, 256)	12845312
dropout_1 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 7)	1799

Total params: 16,075,975
Trainable params: 16,054,087
Non-trainable params: 21,888

Figure 5. CNN Summary with MobileNet.

5.2 Transfer Learning CNN

Transfer learning uses the knowledge of a pre-trained machine learning models that are trained with very high volumes of data applied to a different but related problem. Instead of starting the learning process from scratch, this method starts with patterns learned from solving a related task. A CNN can be seen as the joining of two parts (figure 6). Convolutional and pooling layers are responsible for identifying the features and the fully connected and output layers, which are responsible for the classification of images that present similar features, identical to a classical neural network.

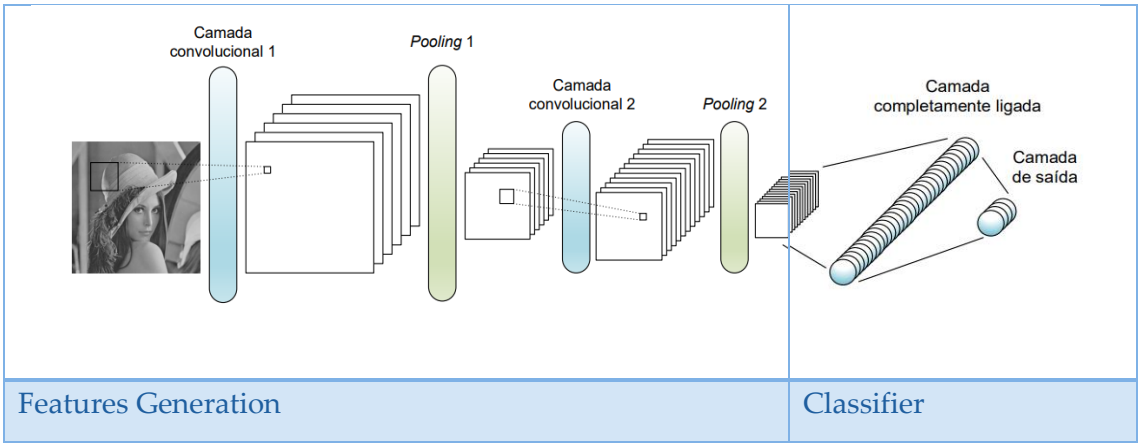

Figure 6. Transfer Learning

5.2.1 ResNet-50

ResNet-50 is a convolutional neural network that is 50 layers deep. ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision problems. The fundamental breakthrough with ResNet was it allowed to train extremely deep neural networks with more than 150 layers.

Convolutional Neural Networks have a major disadvantage, the "*Vanishing Gradient Problem*". During backpropagation, the value of gradient decreases significantly, thus hardly any change comes to weights. To overcome this, ResNet is used. This model makes use of "*Skip Connection*"**.** The skip connections add the outputs from previous layers to the outputs of stacked layers, making it possible to train much deeper networks than previously possible. As a result, ResNet improves the efficiency of deep neural networks with more neural layers while minimizing the percentage of errors. For this work, the following ResNet50 model (figure 7) was developed using transfer learning.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 128, 128, 3)]     0

 tf.__operators__.getitem (S  (None, 128, 128, 3)      0
 licingOpLambda)

 tf.nn.bias_add (TFOpLambda)  (None, 128, 128, 3)      0

 resnet50 (Functional)       (None, 4, 4, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 7)                 14343

=================================================================
Total params: 23,602,055
Trainable params: 14,343
Non-trainable params: 23,587,712
_________________________________________________________________
```

Figure 7. CNN Summary with ResNet50 using transfer learning.

The ResNet50 model was imported, with *include_top = False*, since it was not intended to include the original dense layers of the model and to train the weights of the imported model. The data augmentation technique was used to increase the number of images by adding slightly modified copies of existing images or newly created synthetic images from existing images. A method of pre-processing the images was included to change the dimensions and carry out normalizations. Finally, a pooling layer and a dense output layer were added to the model.

5.2.2 MobileNet

MobileNet uses depth wise separable convolutions which basically means it performs a single convolution on each color channel rather than combining all three and flattening it. This has the effect of filtering the input channels. And thus, reduces the number of arithmetic operations related to convolution at the expense of a small degradation in performance. The implementation of the MobileNet to our problem will allow two interesting comparisons. Firstly, it will be possible to compare the performance of a MobileNet trained from scratch with the transfer learning approach. Secondly, it allows the comparison between two different CNN's using transfer learning, MobileNet and ResNet50. Figure 8 presents the structure of the developed model, such as the data augmentation phase, the number of parameters, and the number of dense layers and neurons used to classify the features identified by the transfer learning CNN's.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 224, 224, 3)       0

 random_flip_1 (RandomFlip)  (None, 224, 224, 3)       0

 random_rotation_1 (RandomRo  (None, 224, 224, 3)      0
 tation)

 mobilenet_1.00_224 (Functio  (None, 7, 7, 1024)       3228864
 nal)

 flatten_1 (Flatten)         (None, 50176)             0

 dense_2 (Dense)             (None, 256)               12845312

 dropout_1 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 7)                 1799

=================================================================
Total params: 16,075,975
Trainable params: 12,847,111
Non-trainable params: 3,228,864
_________________________________________________________________
```

Figure 8. CNN Summary with MobileNet using transfer learning.

5.3 Comparison of results

This subsection is carried out with the goal of comparing the performance of the developed models. As stated above, four different CNNs were developed. Firstly, two convolutional neuronal networks were trained from scratch, the EfficientNetB0 and the MobileNet. Secondly, resorting to transfer learning, the ResNet50 and the MobileNet models were applied to our classification problem. Table 1 shows metrics that allow understanding which were the models that performed the best, given the problem at hand.

The EfficientNet model presents very good values for the training accuracy and training lost function, 97.1% and 0.1, respectively. On the other hand, the same two metrics calculated on the validation dataset present a drastic decrease. The validation accuracy drops to 15.5%, while the loss function reaches 492.5. These values show that the EfficientNet applied to this problem does not present good results. Also, given the big gap between the training and validation metrics, we can assume that the EfficientNet fell into an overfitting problem. These happens when the weights of the layers get to much used to the features of the training dataset, and, when in presence of new images, the model is not able to accurately classify the images.

The MobileNet, trained from scratch, presents good metrics for both the training and validation datasets. The training accuracy of this model was of 98,5%, while the loss value was of 0.05. The validation accuracy is higher than 90% and the validation loss is lower than 0.5, this means that this CNN model performed well for the problem at stake.

Both the models developed using the transfer learning approach present good metrics, for both the training and validation datasets. From these two, the ResNet50 presented the best validation accuracy (89,9%). It is also important to notice that this CNN model was the one presenting the best Loos function value for the validation dataset.

Usually, pre-trained networks perform better than networks trained from scratch, but this happens when the problem at hand is general or related to the training of the transfer learning model. In this case, given that the problem is very specific it is possible to observe that the MobileNet trained form scratch presented better results than the one developed with transfer learning approach. Given that the ResNet50 with transfer learning performed better than the MobileNet, it would be interesting to experiment training a ResNet50 from scratch and evaluate if the results would be even better.

Table 1 – CNN's metrics results.

	Convolutional Neural Networks			
	Scratch		Transfer Learning	
	EfficientNet	Mobile Net	ResNet50	MobileNet
Train Accuracy	0.9708	0.9854	0.9716	0.8838
Train Loss	0.1017	0.0496	0.1072	0.3245
Validation Accuracy	0.1550	0.9152	0.8989	0.8443
Validation Loss	492.5001	0.3895	0.2841	0.5539

6. Conclusion

In conclusion, we have successfully trained an object identifier to detect military planes and subsequently trained neural networks to classify those military airplanes. This process demonstrates the power of machine learning in identifying and classifying objects and has potential applications in various fields such as defense, security, and surveillance. Future work could focus on improving the accuracy of the object identifier and classifier, as well as exploring new ways to utilize the technology. Overall, this project highlights the importance of ongoing research and development in the field of machine learning, and the possibilities that it holds for the future.

References:

1. https://www.kaggle.com/datasets/a2015003713/militaryaircraftdetectiondataset
2. https://terravistabrasil.com.br/conheca-os-diferentes-tipos-de-avioes-militares-existentes/
3. https://arxiv.org/abs/1905.11946
4. https://arxiv.org/abs/1704.04861
5. https://arxiv.org/abs/1512.03385