Deep Duck : Duck Species Identification using Convolutional Neural Networks

Learning to differentiate duck species using machine learning

ANNEN Rayane, MARTINS Alexis

18.06.2023

Contents

1 Introduction

This project involves building an app to identify duck species using machine learning. We've gathered a diverse set of images from the web using a Python-based web scraper. To broaden our dataset, we applied data augmentation techniques like rotation and scaling. For processing, we're using Convolutional Neural Networks (CNNs) and transfer learning, which allow efficient and effective species recognition. This tool has potential uses in areas like ornithology and wildlife conservation.

2 The problem

We are trying to learn to differentiate eight unique classes of ducks, comprised of seven species of male ducks and one specie of female ducks. The focus on male ducks arises from their vibrant and distinct coloration, in contrast to the more uniform brownish and greyish hues of females.

The dataset collected for this project is very balanced as all the classes have the same amount of images. For each of the eight classes, we have a total of 50 images (40 used for training and 10 for testing). This amounts of 400 images in total, a manageable amount for detailed processing and analysis.

Duck Species	Training Images	Testing Images
Allier White Duck	40	10
Gadwall Duck	40	10
Mallard Duck	40	10
Mandarin Duck	40	10
Northern Shoveler Duck	40	10
Tufted Duck	40	10
Whistling Duck	40	10
Female Goosander	40	10

The intra-class diversity is relatively low, as all ducks within a specific class exhibit similar characteristics. However, the inter-class similarity is also quite low. Each species of duck we have chosen exhibits unique color patterns, reducing the chances of misclassification between classes. These two factors should allow the CNN to effectively learn the distinguishing features of each class and accurately identify the species of ducks in new unseen images.

3 Data Preparation

The initial step in our data preparation involved resizing and rescaling the images. To ensure compatibility with our model, we resized all images to a standard dimension of 224x224 pixels.

To enhance our dataset, we implemented data augmentation. These techniques included transformations such as rotation, zooming, etc… This process generated additional training samples.

Our dataset was split into two subsets: training and testing. We allocated 40 images from each class to the training set and the remaining 10 images to the test set, achieving an 80-20 split.

4 Model Creation

We decided to compare four different models to determine which one would be the most effective for our task. The difference between the models are the variation of epochs and neurons per layer. All the other parameters were constant for all the models because they were the most effective for our task.

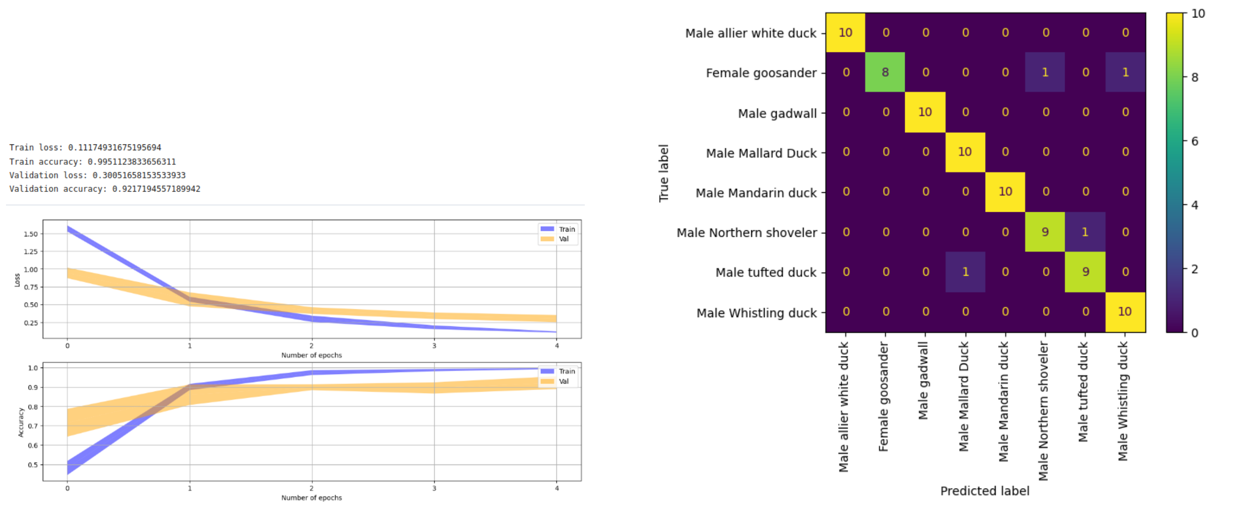
4.1 Hyperparameters exploration

We tried a lot of different hyperparameters and different models, we selected five models that we thought were the most different. We mainly modified the number of epochs and the number of neurons per layer. All the other hyperparameters were constant for all the models and were the default values of the model given. We also tried to add a dropout layer because we were curious about the results. The five models are the following:

- Model 1: 5 epochs, 50 neurons per layer
- Model 2: 5 epochs, 250 neurons per layer
- Model 3: 5 epochs, 500 neurons per layer
- Model 4: 10 epochs, 50 neurons per layer
- Model 5: 5 epochs, 250 neurons per layer with a dropout layer of 10%

4.1.1 Model 1

The problem with this first model is that on multiple runs, the consistency of the results varies a lot. Sometimes the model has very good performances and sometimes it has very bad performances. The images below are the results of an average run of the model.

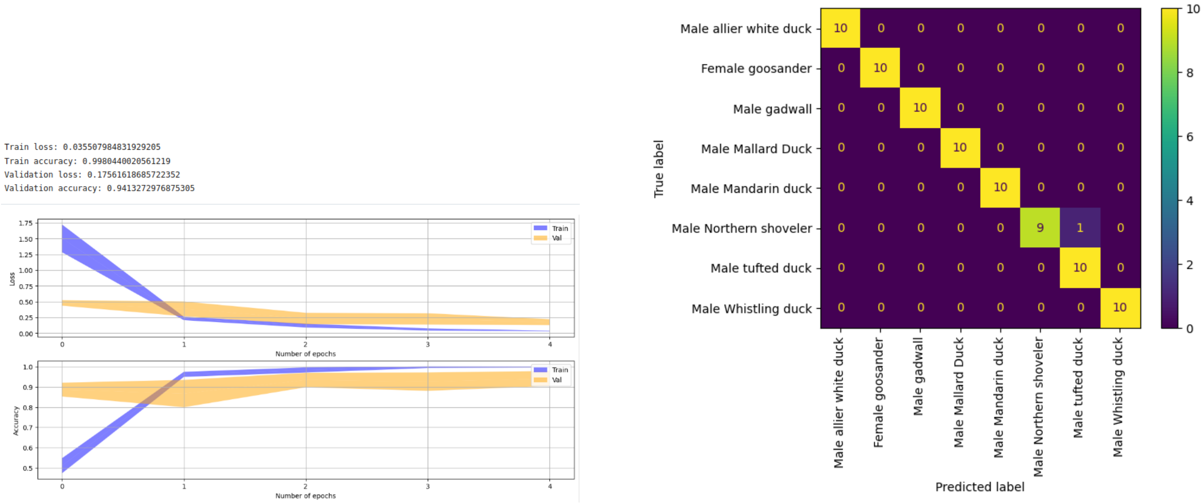


(a) Error graph **(b)** Confusion matrix

Figure 1: Model with 5 epochs and 50 neurons in the dense layer

4.1.2 Model 2

This model is one of the best models we tried. It has very good performances and it doesn't overfit as the complex models.

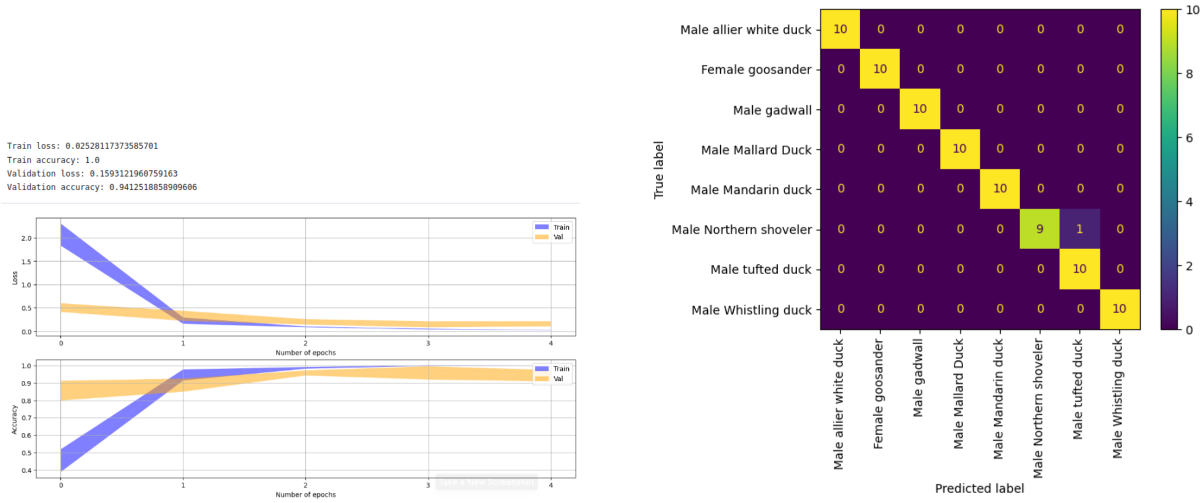


(a) Error graph **(b)** Confusion matrix

Figure 2: Model with 5 epochs and 250 neurons in the dense layer

4.1.3 Model 3

The loss and the accuracy of the model are very good. But this model has a problem of overfitting, we notice near the fifth epochs the model starts to overfit by increasing his loss.

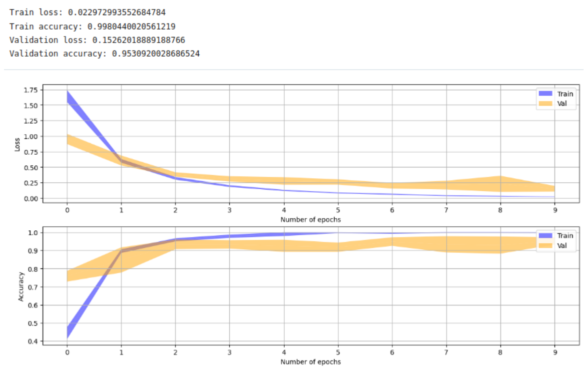


(a) Error graph **(b)** Confusion matrix

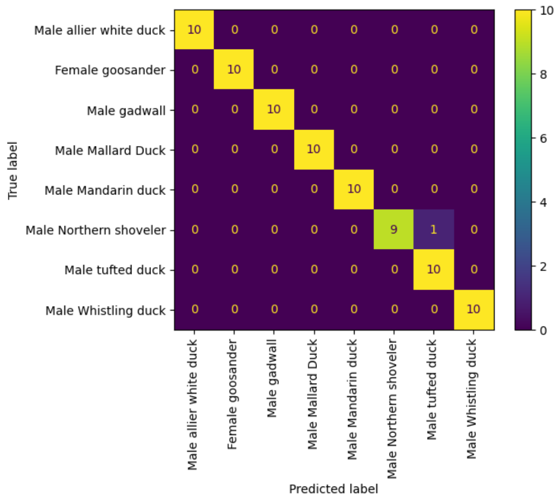
Figure 3: Model with 5 epochs and 500 neurons in the dense layer

4.1.4 Model 4

This model has also good performances but it has overfit few times on multiple runs. Plus it is not really worth the effort to train this model comparing to a more simple model with less epochs.



(a) Error graph

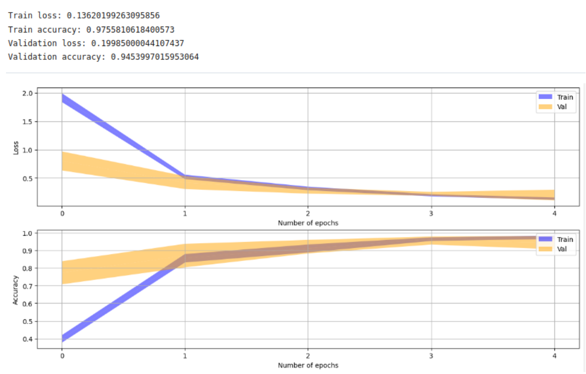


(b) Confusion matrix

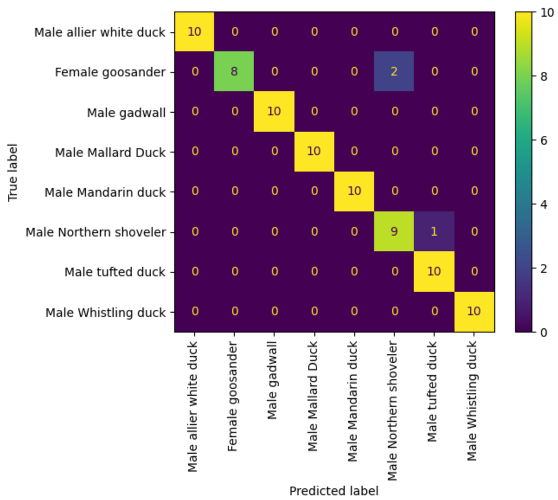
Figure 4: Model with 10 epochs and 50 neurons in the dense layer

4.1.5 Model 5

We tried to add dropout to our best model to see if it could improve the performances. But it didn't really change anything. Indeed, the model was even a little bit worse than the model without dropout.



(a) Error graph



(b) Confusion matrix

Figure 5: Model with 5 epochs and 250 neurons in the dense layer with a dropout layer of 50%

The model with the best accuracy was model 2.

4.2 Final model parameters

The final model we chose was model X had these parameters :

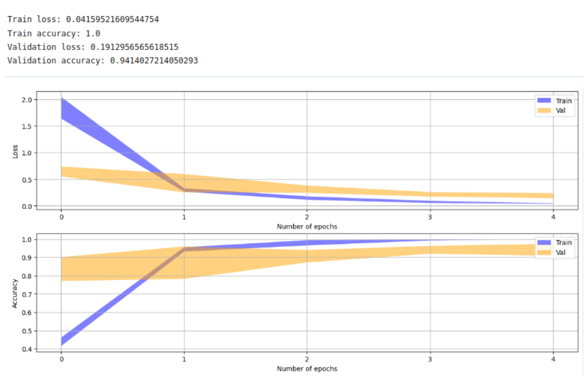
- 5 epochs
- 250 neurons per layer
- 1 layer
- 32 batch size
- 0.001 learning rate
- Optimizer : RMSprop
- Activation function : ReLU
- Loss function : Sparse Categorical Cross-Entropy

4.3 Transfer learning

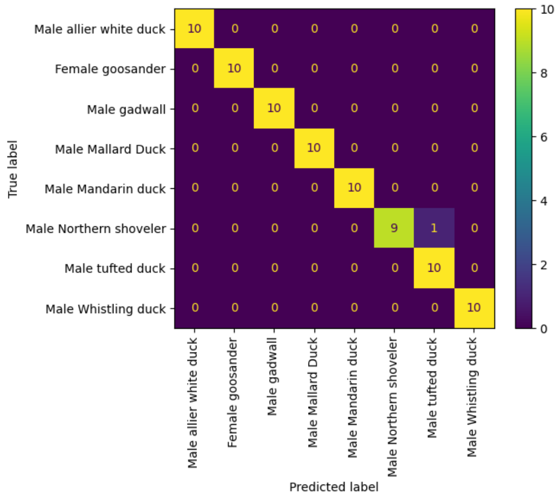
We performed the transfer learning by freezing all the layers except the last one. We used transfer learning because it is a very effective technique to train a model with a small dataset. The model we used for transfer learning is the MobileNetV2 model. With this method, we had a very good model without having to train it for a long time and this is the main advantage of transfer learning.

5 Results

5.1 Confusion matrix and plots



(a) Error graph



(b) Confusion matrix

Figure 6: Final model with 5 epochs and 250 neurons in the dense layer

5.2 F-score for each class

```
F1-score Male allier white duck : 1.0
F1-score Female goosander : 1.0
```

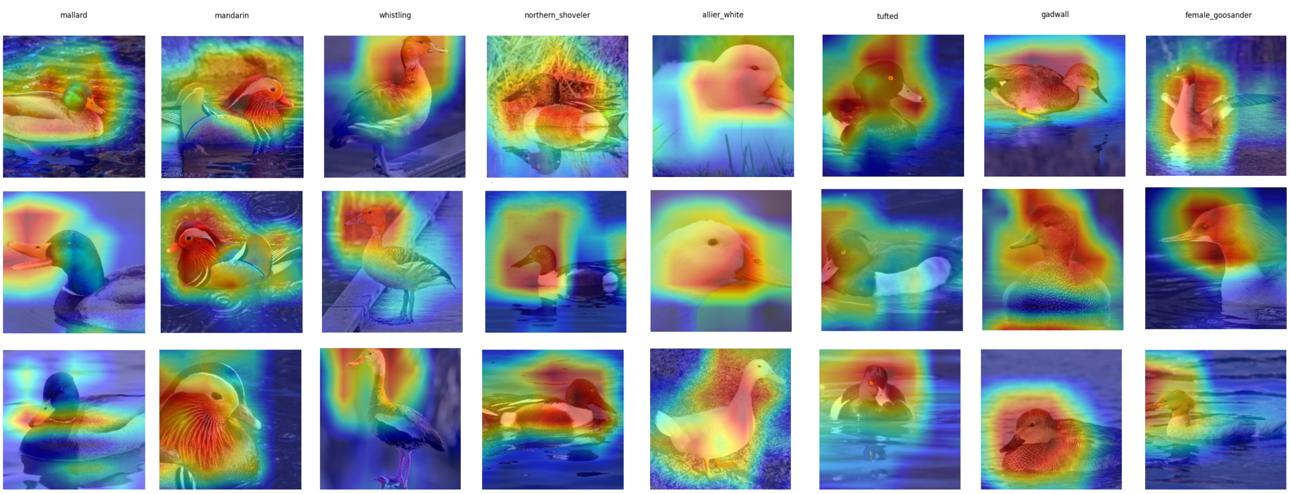
```
F1-score Male gadwall : 1.0
F1-score Male Mallard Duck : 1.0
F1-score Male Mandarin duck : 1.0
F1-score Male Northern shoveler : 0.9473684210526316
F1-score Male tufted duck : 0.9523809523809523
F1-score Male Whistling duck : 1.0

Model F1-score : 0.987468671679198
```

5.3 Results after model evaluation

5.4 Grad-cam analysis

First, let's analyze the features that the model focuses on to make its predictions.

- Mallard Duck : It seems to focus mainly on the head and the beak of the duck.
- Mandarin Duck : The model focuses on the head and the neck. It's understandable because the shape of the head is very unique.
- Whistling Duck : The focus is on the neck of the duck. Maybe it comes of the shape of it, it's very long and thin.
- Northern Shoveler Duck : The model focuses on the head and potentially on its eye. It is very particular because it's not a regular black eye like the others.
- Allier White Duck : The model focuses on the head and the beak of the duck. This duck don't have any particular feature more than just being all white.
- Tufted Duck : It's similar to the Northern, because the eye is also very special. The head is the main focus of the model for its black color and special eye.
- Gadwall Duck : It seems to recognize it on an overall shape of the duck, in most part of the images the whole duck is in the focus of the model.
- Goosander Duck : It focuses on the head and the beak of the duck. The beak is particular in this duck and it should help the model to recognize it.

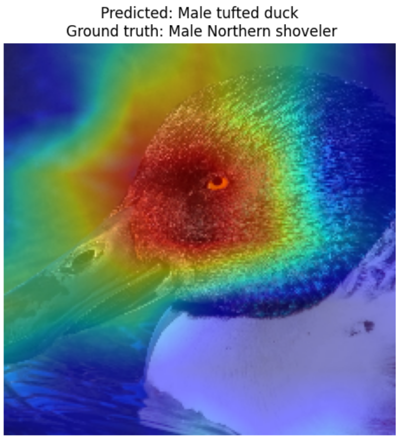


We also noticed few images of our model where the focus is not directly on the duck. The model seems to look at the background of the image. It could possible come from the fact of having a small dataset and the most part of the ducks are on a lake or a river. The model could have learned that the background is also important to recognize the duck.

5.5 Misclassified images

When we tried different models there was mainly two cases that could happen. The first one is having a part of the mallard ducks being misclassified as a northern shoveler duck, it also works the other way. It could possibly come from there head, it's very similar in terms of colors. This case was very rare compared to the second case which is having one particular image of a northern shoveler duck being misclassified as a tufted duck. This image is the following :

(a) Error graph



(b) Confusion matrix

Figure 7: Example of misclassified image

The only explanation we have is that the model is confused by the eye of the duck. Indeed, the eye of the northern shoveler duck is very similar to the eye of the tufted. Plus on the grad-cam view, we can see that the model focuses on the eye of the duck.

5.6 Dataset improvement

We could improve the dataset by adding more images of the following :

- More images : Having more ducks in the dataset with different angles and different backgrounds. Also we can think about different environments with for example less light, etc...

- Cleaning watermarks and texts : It wasn't a problem for this particular case, but it could have been a problem if the model started to focus on the watermarks. The solution is to remove them from the images or directly from the dataset.
- Take images directly from the device instead of the web : The images from the web are not always the best quality and also they could be very different from the images taken by the device. It would improve the experience on the device to have specific images for it.

5.7 Classes confusion

As we said before, the model has a tendency to sometimes confuse the mallard duck with the northern shoveler duck. Also, it confuses the northern shoveler duck with the tufted duck for one specific image.

///////// TODO : COMPLETE SI SMARTPHONE FONCTIONNE ///////////

6 Conclusion

In terms of performances the model is very good, it recognize our type of ducks in almost all the cases. We decided to keep the model simple as possible after the advices of the teacher. It also reduces a lot the training time without necessarily losing in performances. We could probably improve the model by adding the improvements we talked about before and create a more complex model with more specific layers.

The model works very well on our use case because we intentionally chose ducks that are very different from each other. We focused on male ducks because they are more colorful and it's easier to recognize them. Maybe if we also used the female ducks, our model would have been much less accurate because they are less colorful and all the female ducks look very similar.

For a personal point of view, we were very happy to work on this project. It was very interesting to see how we can create a model that can recognize images. The fact of applying the theory we learned in class on a real project was very interesting and we learned a lot from it. We were very enthusiastic to try tricks such as transfer learning, data augmentation, dropout or even the grad-cam analysis.