Object Recognition using Convolutional Neural Networks

Authors : Junod Arthur & Häffner Edwin

1. Introduction

Identifying different types of trees can be tricky, especially for those of us who aren't tree experts. The subtle differences in leave can be hard to spot if you don't know what you're looking for. That's why we decided to develop a model that can recognize and classify various tree species using images of their leaves.

This tree identification model could be really handy for learning more about the different types of trees around us.

We collected images by taking some pictures ourselves and then we took more from the internet using "Bing image downloader" since we couldn't have enough data just by ourselves.

We will be using transfer learning with MobileNetV2 to classify the pictures.

2. The Problem

Here are the trees we're trying to classify :

- Ash: Fraxinus excelsior
- Beech: Fagus sylvatica
- European Spindle Tree: Euonymus europaeus
- Forsythia: Forsythia x intermedia
- Gean (Wild Cherry): Prunus avium
- Hazel: Corylus avellana
- Hornbeam: Carpinus betulus
- Horse Chestnut: Aesculus hippocastanum
- Red Robin (Photinia): Photinia x fraseri
- Silver Birch: Betula pendula
- Wayfaring Tree: Viburnum lantana

We went outside and took between 15 to 40 photos of leaves per tree listed above. Then as said earlier, we populated each class with more images found on the internet.

Example of pictures :

Beech

Hornbeam

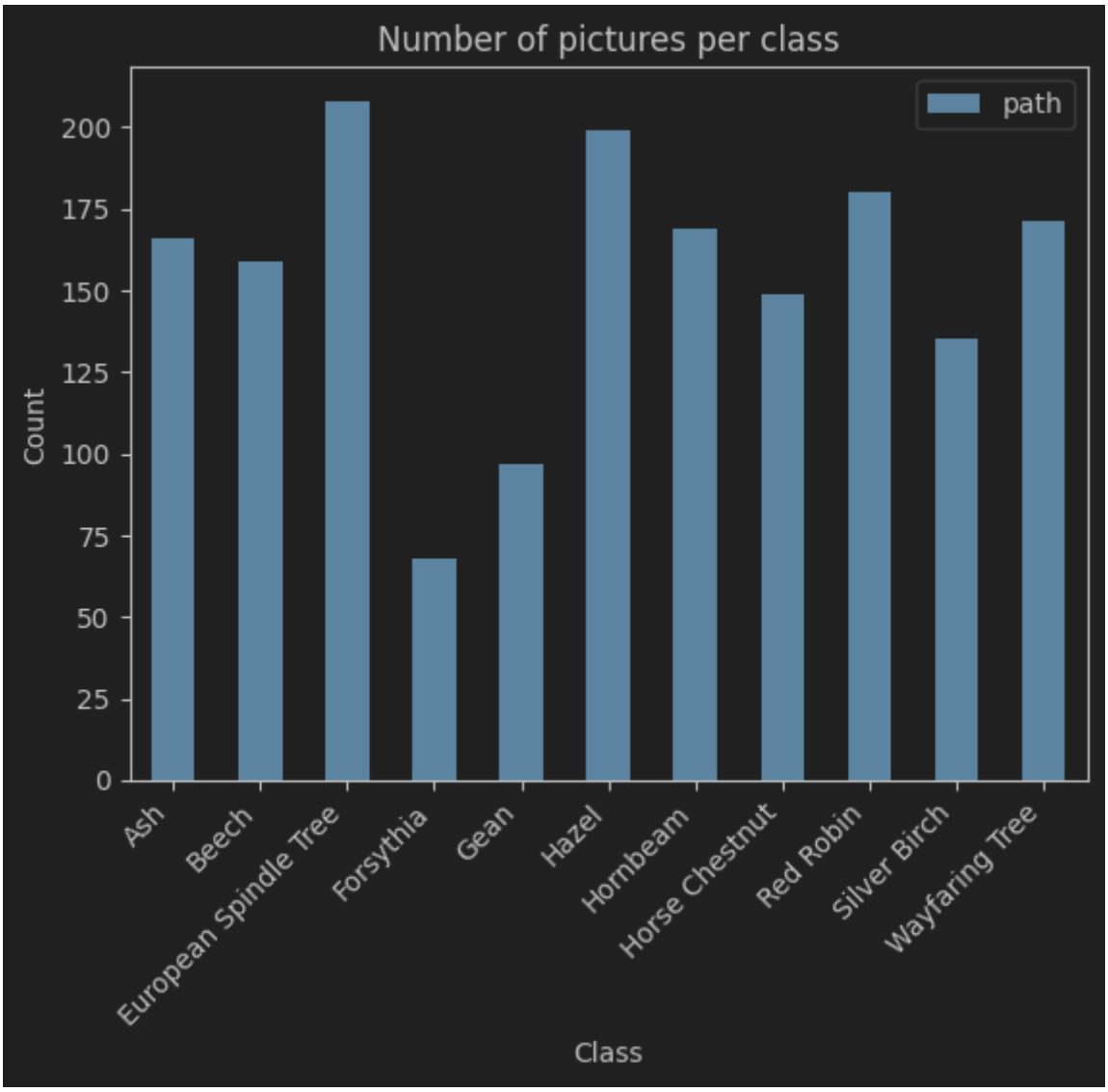
We can notice we only have deciduous trees so the only difference between different classes of tree (considering we use images of leaf to classify our images) is the shape of their leaves and sometimes the color of their leaves. And as we'll see later, this will be a big problem for classifying which leaves correspond to which tree.

3. Data Preparation

We have done some preprocessing to the images to avoid having a really low quality dataset :

- Filtering
  - After downloading all the images from bing, 200 for each tree, we manually sorted and deleted the pictures that weren't leaves as well as we could.
- Normalization
  - Since we were performing transfer learning using the MobileNet V2 model, which was pre-trained on images with a resolution of 224x224 pixels, we resized all our input images to match this resolution.

Our first dataset :



As you can see, there's not a lot of *Forsythia* and *Gean*, the issue was that most of the picture we gathered on Bing were flowers of the trees but we wanted to focus our model on recognition through the leaves.

Overall it's not a huge dataset but it should be big enough to provide us good enough results.

Our final dataset :

We used data augmentation on this dataset to try to correct the dataset imbalance, as we will see later, it will have an effect on our results :

We tried to remove some more images, we had lots of images that had water marks as they weren't free to use, image services such as "Alamy" were polluting our downloaded data. So we tried to remove them in hope to get better results. But since what we're trying to classify is quite difficult, removing this bad data didn't really change our result by much. So as for the final training we're going to show you, this "garbage" data is still present as it's the data that gave us the best F1-score.

So it's the same initial data as the one presented above, but this time we augmented the images during the training.

4. Model Creation

First try :

We tried this model as it was the one we used for the first part of this lab, after a lot of trial and error :

```
added_layers = [
        GlobalAveragePooling2D(),
        Dense(128, activation='relu',
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
        Dropout(0.5),
        Dense(len(LABEL_NAMES), activation='softmax',
kernel_regularizer=tf.keras.regularizers.l2(0.001))
    ]
```

It is fairly basic but we can easily try other parameters from there. But we still added a L2 regularization to avoid having an over-fitting problem right from the start.

As for the epochs, optimizer and the learning rate, we used the same settings as in the first part of the lab as we thought they were quite alright.

By using transfer learning on MobileNetV2, we could easily obtain pretty good results with the image recognition without having to retrain a whole model to detect base features in an image.

Last try :

We tried unfreezing few layers of MobileNetV2, using CNN layers on top of the MLP style layering we did on the first try. We added hidden layers, changed the dropout value, change the optimizer, the learning rate, we did everything we could think of and the only thing that made our training better was to add batch normalization.

So our final hidden layers are these :

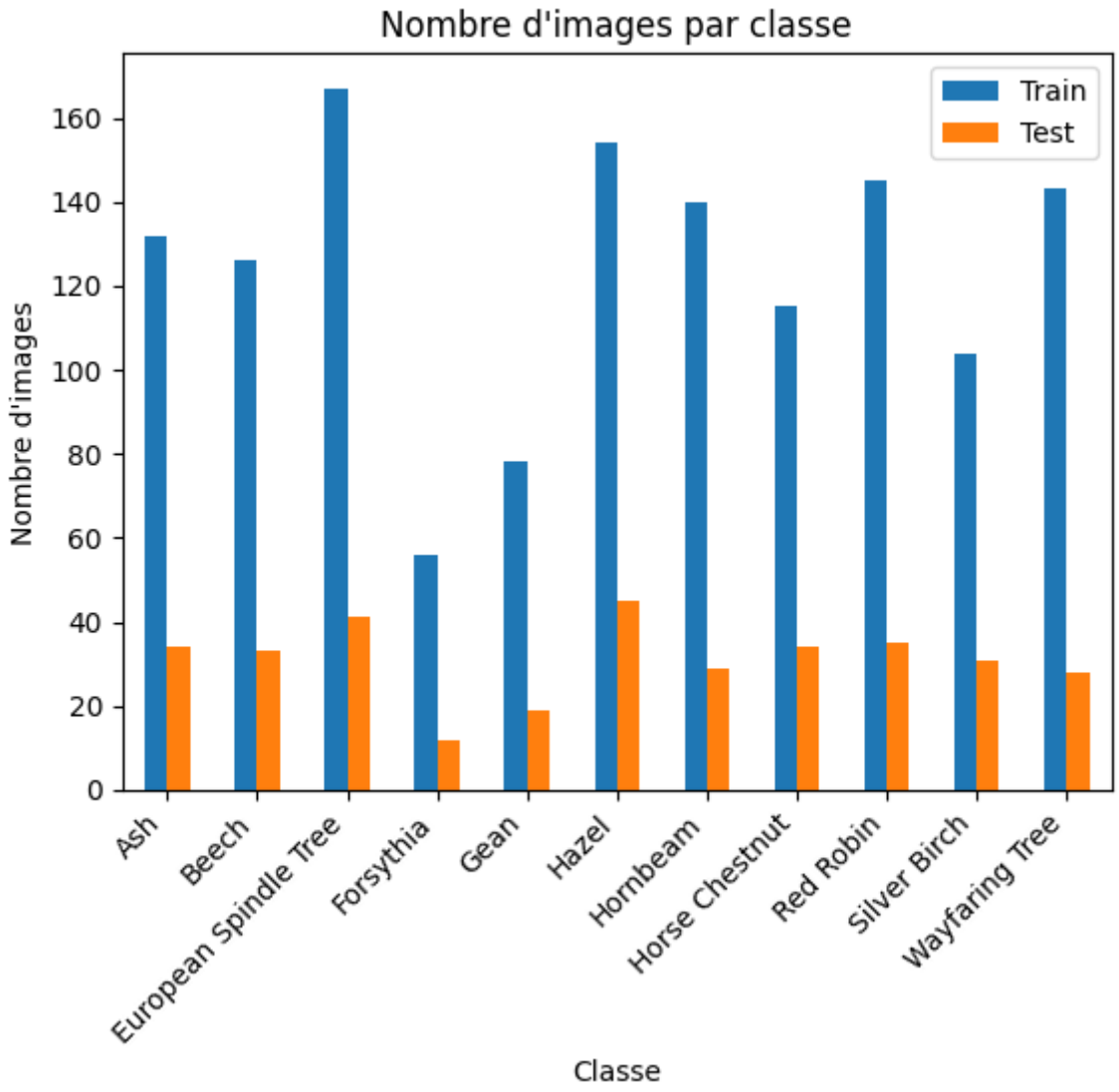
```
added_layers = [
        GlobalAveragePooling2D(),
        Dense(128, activation='relu',
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
        Dropout(0.5),
        BatchNormalization(),
        Dense(len(LABEL_NAMES), activation='softmax',
kernel_regularizer=tf.keras.regularizers.l2(0.001))
    ]
```

As for the hyper parameters we used RMSprop as the optimizer with the default learning rate values. We did 5 KFolds, 10 epochs and batches of 32.

For the learning rate values, we tried 0.0001 and 0.01 instead of the default 0.001 and we couldn't really see any improvements, if anything, our results were worse.

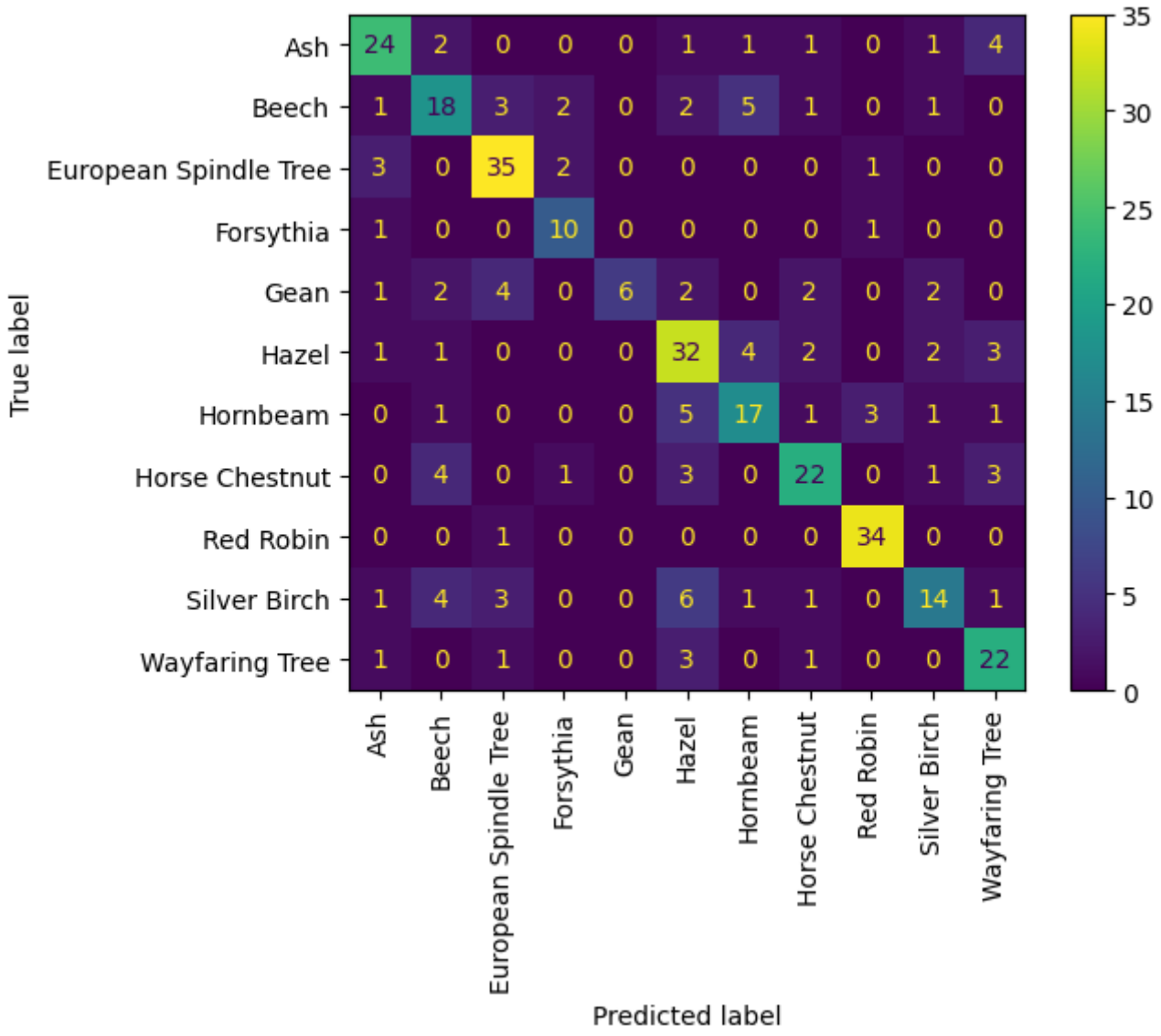
5. Results

In all of our tries, we separated our dataset into a training set and a test set. We're using 20% of our total dataset to test it and then calculate our F1 score based on these tests.

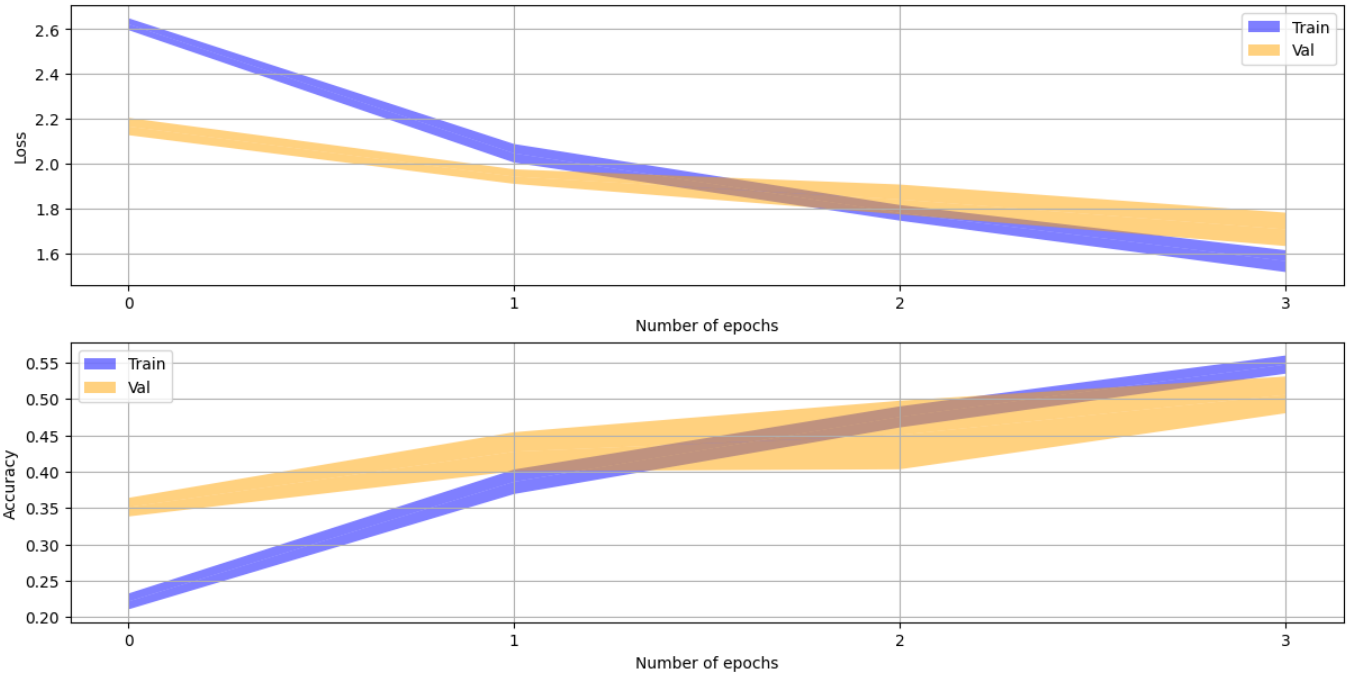


First dataset :

Our confusion matrix :



The loss and accuracy :



F1 score :

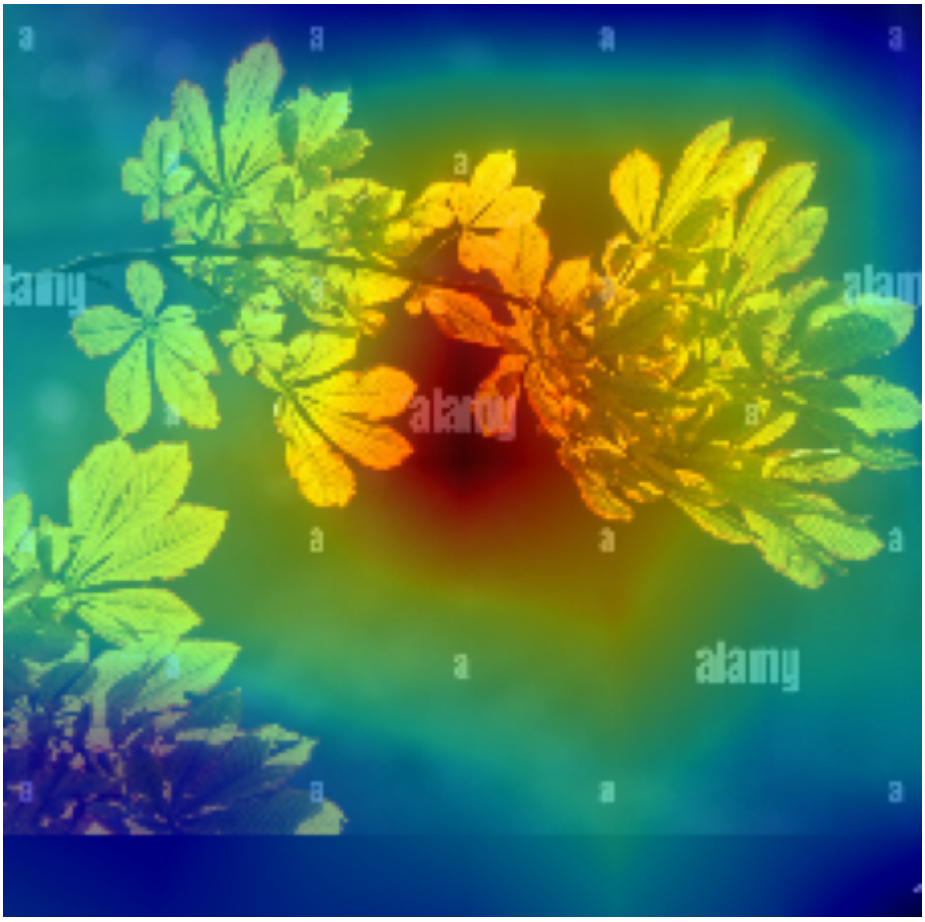
/

```
F-score for Ash: 0.7164179104477612
F-score for Beech: 0.5538461538461539
F-score for European Spindle Tree: 0.7954545454545454
F-score for Forsythia: 0.7407407407407407
F-score for Gean: 0.48
F-score for Hazel: 0.6464646464646465
F-score for Hornbeam: 0.5964912280701754
F-score for Horse Chestnut: 0.676923076923077
F-score for Red Robin: 0.918918918918919
F-score for Silver Birch: 0.5283018867924528
F-score for Wayfaring Tree: 0.7096774193548387
Average F-score: 0.669385138819392
```

We used a CAM to highlight the problems when we got mislabeled pictures :

We can see in some of the pictures that the model tried to focus it's attention on the watermark that was on it. Our model could be be trying to identify some class by finding the watermark on the picture, so that's a comportment we would have to watch out for.

Real: Horse Chestnut, Pred: Wayfaring Tree

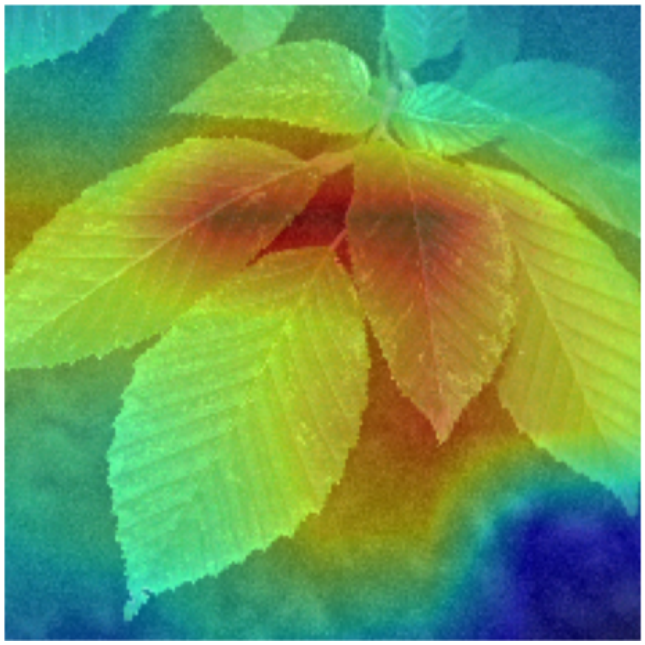


Other mispredicted pictures are the ones containing flowers (we tried to manually get rid of most of them but some still creeped their way into it), and we see that our model focus on them. It leads it to. for example, label the *Gean* tree as a *Wayfaring tree* which, similarly, has white flowers.

Real: Gean, Pred: Wayfaring Tree

But a large number of them were just the model not focusing on a leaves but unidentified entities.

Real: Hornbeam, Pred: Hazel

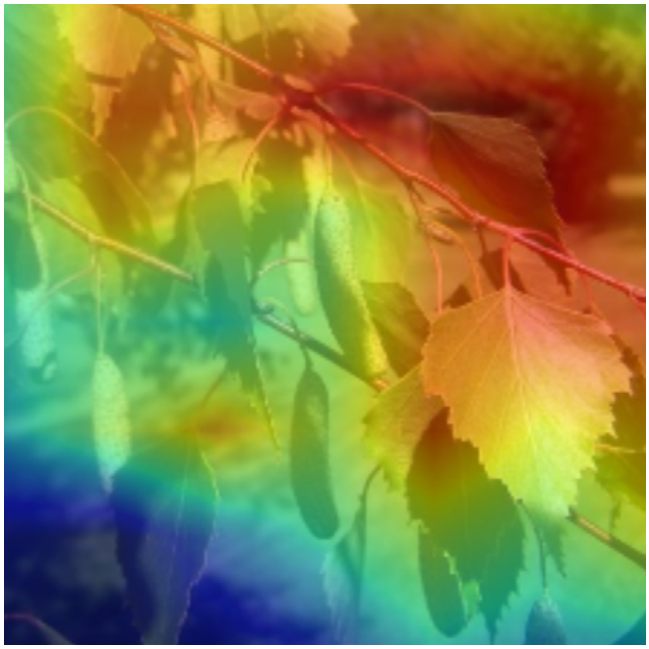


Real: Hazel, Pred: Silver Birch

It also mislabelled some pictures because it focused on other "objects" or the back of a folded leaf in the picture.

Real: Hazel, Pred: Silber Birch

Real: Silver Birch, Pred: Beech

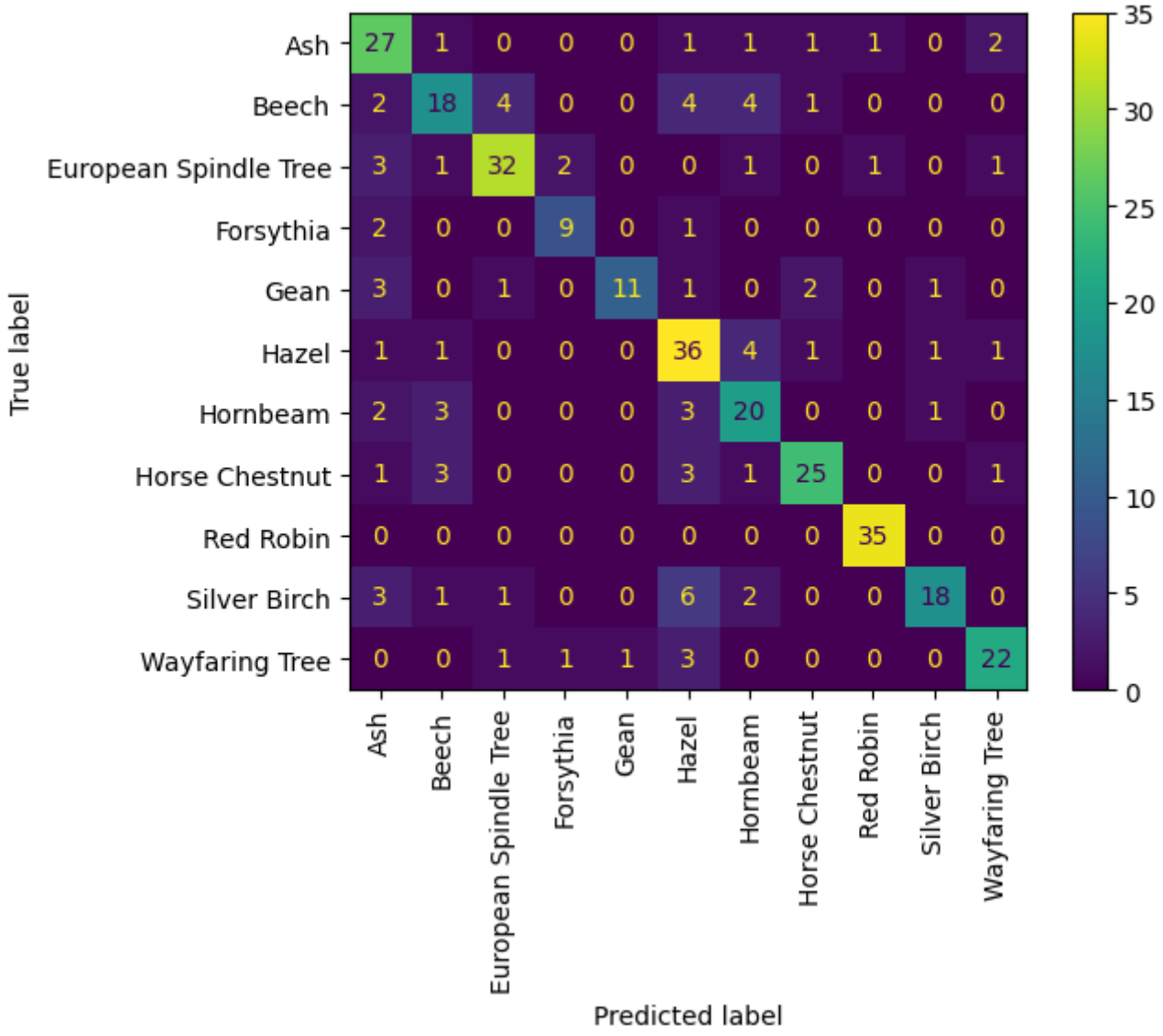
Real: Hazel, Pred: Silber Birch	Real: Silver Birch, Pred: Beech
	

As written before the *Forsythia* have less images in total compared to the other classes and is less often predicted with the correct label, so one of the possible improvement we could make to our dataset is to retrieve more pictures for this class.

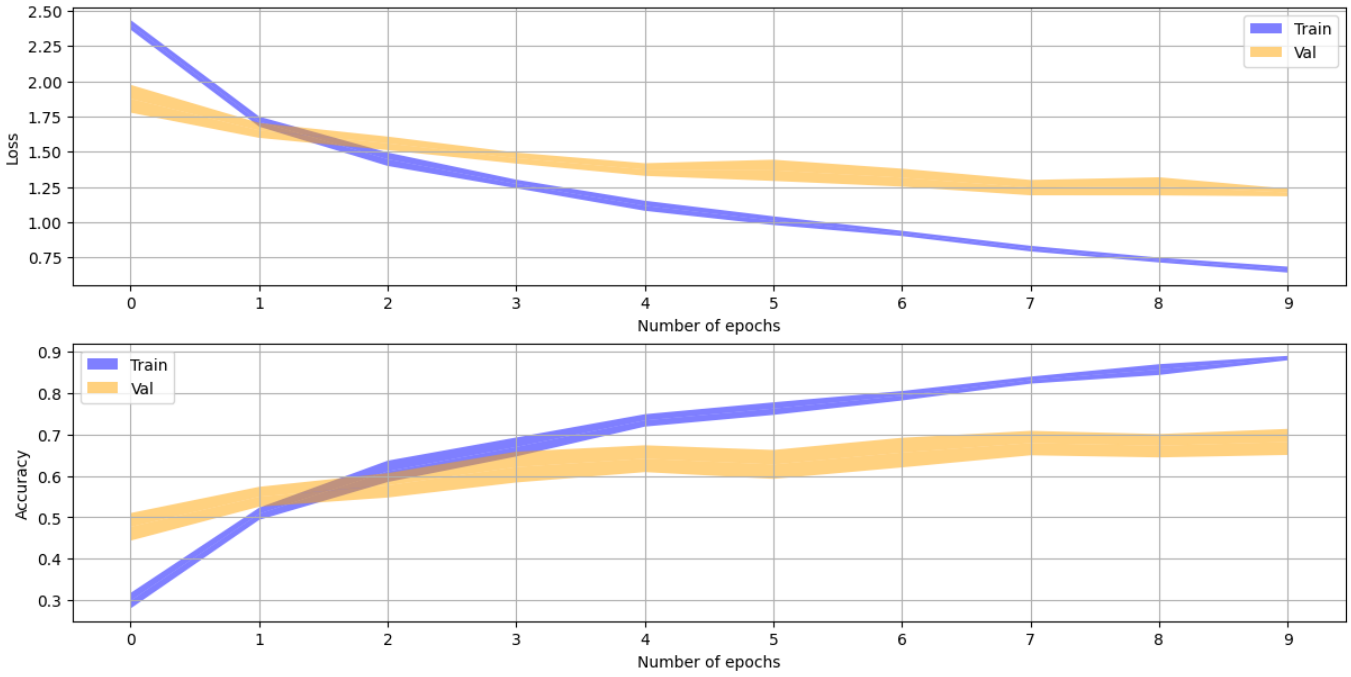
The *Silver Birch* was the most confused class with 17 false prediction label. It's possible it comes from the similar looks of the leaves compared to others or maybe because the fruit of the *Silver Birch* looks close to other fruits in our selected classes.

Final dataset :

Our confusion matrix :



The loss and accuracy :



F1 score :

```
F-score for Ash: 0.6923076923076923
F-score for Beech: 0.5901639344262295
F-score for European Spindle Tree: 0.8
F-score for Forsythia: 0.75
F-score for Gean: 0.7096774193548387
F-score for Hazel: 0.6990291262135923
F-score for Hornbeam: 0.6451612903225806
F-score for Horse Chestnut: 0.78125
F-score for Red Robin: 0.9722222222222222
F-score for Silver Birch: 0.6923076923076923
F-score for Wayfaring Tree: 0.8
Average F-score: 0.73928357974135
```

We see that by using batch normalization we improved our results. But it's not the only thing we've changed in here. We also used the function to augment the image with these settings :

```
image_augmentations = Sequential([
    layers.RandomFlip("horizontal_and_vertical"),
    layers.RandomRotation(0.3),
    layers.RandomZoom(0.3),
    layers.RandomContrast(0.3),
    layers.RandomTranslation(height_factor=0.2, width_factor=0.2),
])
```

In the end, on our heat maps, we were finding the same issues we discussed on our first try, there was just less of them. We could also see a lot of watermarks being detected instead of the actual leaves. We tried to remove this data in other tries but it didn't make the training results better as weird as it sounds. So as this was our best shot, we kept this one as the showcase of our training.

We can also see that the trees with the least data like the *Gean* went from having an F1 score of 0.48 to an F1 score of 0.71, we can suppose that data augmentation helped a lot here.

In general, the F1 score were better for every trees but the *Ash* that had a diminution of 0.02, it could be due to the randomness of the training more than our changes in the training.

6. Conclusions

We were impressed how the use of transfer learning with MobileNetV2 simplified the recognition for the pictures and provided us with good enough results.

However, our model still has performances issues regarding some classes. Gathering more pictures of these classes could maybe improve the successful prediction rate of our model. The issue in classifying tree leaves is that they are quite similar, in shapes and in colors. This is evident in our training, as the Red Robin, the one tree with some red leaves, had the highest F1 score of all, approximately 0.97.

We also observed that the model was strongly drawn to detecting the tree's fruits or flowers whenever they were present.

We also could add more type of tree to identify, it would allow our model (once embedded in a phone) to be more useful in real condition where most of the tree we would encounter wouldn't be part of our selected classes.