Artificial Neural Networks and Deep Learning
Homework 1

Nicola Dean
10617541
nicola.dean
@mail.polimi.it

Marco Fasanella
10617541
marco.fasanella
@mail.polimi.it

Raffaello Fornasiere
10617541
raffaello.fornasiere
@mail.polimi.it

Christian Spano
10617541
christian.spano
@mail.polimi.it

1 Introduction

In order to touch all the important aspects of the procedure of finding the best solution to this classification problem, we started from our own self-made model to more sophisticated methodologies.
We can summarize our approaches in:

- Vanilla network
- Transfer Learning and Fine Tuning
- Ensemble method

Furthermore in all the attempts made, we used two classes, specifically **created to automate and support the model creation procedure**:

- Dataset Helper
- Model Helper

Through the continuous attempts, and support of methods in the two helper classes, we managed to find our best model and reach 0.8691 accuracy in the competiton.

2 Dataset Helper and Model Helper

In order to ease the development of our best solution, we thought it would have been useful to create two ad-hoc helper libraries. As a matter of fact, a lot of code is often repeated or many lines of code are needed just to plot or to get insightful results (such as the confusion matrix or the accuracy). In particular, what we did was to create two helpers:

- **Dataset helper**: for dealing with everything related to the dataset; for instance (just for mentioning the most relevant functions) we created functions to perform *augmentation* and *cut-out/cut-mix* of the dataset. Other functions were intended for processing data (e.g., normalization)
- **Model helper**: for dealing with everything related to the models; this helper was intended for easing the construction of models. For instance, we create some functions for plotting insightful graphs (e.g., confusion matrix, residuals, predictions, etc.) and some other functions for managing the models (e.g., save the model in a specified directory or create checkpoints during training)

At the end, these helpers indeed turned out to be really helpful. Just with a line we had everything we needed.

3 First try: vanilla network

–NICOLA– -IMG della rete (dal lab) (magari orizzontale) risultati considerazioni

3.1 Batch Normalization

A first attempt was also adding a Batch Normalization + Relu Activation Layer before our Pooling layers. This lead to poor result due to the fact that the network was too small.

3.2 Our homemade CNN

So deepening the CNN gave this solution much significant improvements.
METTERE IMMAGINE CNN FINALE — RAFFAELLO—

3.3 Considerations

Best result consideration and observations
Commento da Christian: qui specifichiamo l'importanza della augmentation e che abbiamo pensato potesse risolvere il problema della bassa accuracy (infatti alla fine è megliorato) + che poi abbiamo espanso con cut-out e cut-mix – magari dire brevemente il perché, secondo noi, con augmentation/cutout è migliorato.

4 Transfer Learning and Fine Tuning

We then noticed that we needed a big change on the approach to use, because the homemade CNN was performant, but not enough. So **we started using transfer learning and Fine tuning**

4.1 Approach: Freezing Layers

The *modus operandi* that will be used from now is: freezing all layers while training on our augmented dataset the keras.application CNN, and then unlock a small amount of layers to the second phase of training (fine tuning) as near as possible to the output.

4.2 VGG16

The first network we decided to utilize with this approach was VGG16. We started from this network mainly because, generally speaking, it has given great results in many different Deep Learning tasks; so, we thought it was a good network to start with. Indeed, at the end, it did not deceive us. More in details, at the very first we trained this network neither applying augmentation nor fine-tuning. However, as expected, we got poor results (an accuracy of about 60%). Applying data-augmentation, the accuracy risen up to roughly 70%,

that is an improvement in accuracy of 10%.
At this point, to boost up even more the accuracy, we decided to do fine-tuning treating the number of freezed layer as an hyperparameter to be learned. Indeed, we trained the network with k freezed layers and we pick up the one such that

$$k^* = \underset{k}{\text{argmax}}\{test\_accuracy_k\}$$

TO BE CONTINUED...

4.2.1 Results

4.3 VGG19

VGG-19 is a convolutional neural network that is 19 layers deep, so we kept the freezed layers in the range of the first 8-14.
Different Data Augmentations (with different seeds and augmentation parameters) were performed between the two phases, to even increase randomisation in the two training processes.

4.3.1 Results

Freezed Layers	Accuracy	Precision	Recall	F1
8	0.8169	0.7989	0.7651	0.763
9	0.8225	0.8181	0.7682	0.7776
10	0.8338	0.8161	0.7929	0.8001
11	0.7577	0.7109	0.715	0.7048
12	0.7944	0.766	0.7504	0.7489
13	0.8028	0.7806	0.754	0.7596

Table 1: Results with Transfer Learning and Number of freezed layers for VGG19.

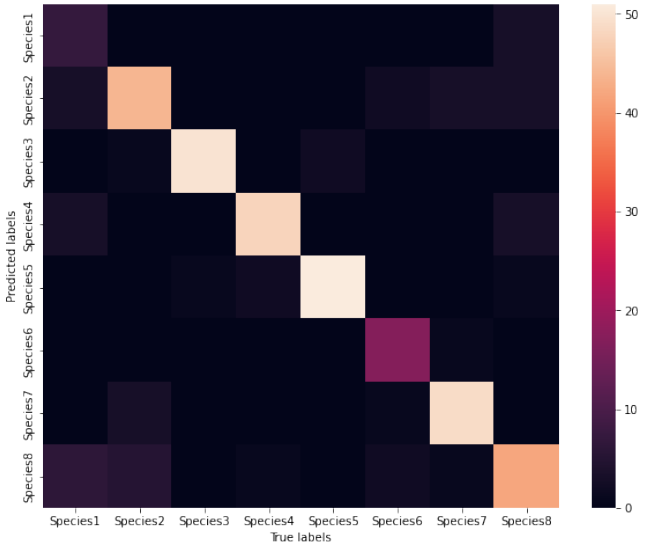

Figure 1: Confusion Matrix of best configuration with VGG19.

4.3.2 Considerations

One of the most particular observation that we can made after experimenting the first attempts of freezing, is that freezing the net until the pooling and not between convolutions leads to a better accuracy.

4.4 Xception

4.5 Other Models

4.5.1 Resnet

–NICOLA–

4.5.2 GoogleNet

4.6 EfficientNet

–NICOLA–

5 Ensemble

–NICOLA– Approccio provato a mischiare modelli c'era bias perchè avevano seed diversi

5.0.1 Results

6 If we had more time..

con più tempo cosa avremmo provato

7 Our Submissions

Model	Details	Result
VGG16	8 Freezed Layers	0.8325
VGG19	10 Freezed Layers	0.8230

Table 2: Results with Transfer Learning and Number of freezed layers for VGG19.

8 Conclusions

Considerazioni finali e best model fattoo

References