Homework 2 - Machine Learning: Weather Classification

Edoardo Piroli - 1711234

December 15, 2019

Contents

1 Introduction

1.1 Assignment

The assignment of the homework was to build an image classifier capable of assigning pictures to one of the following classes: {*RAINY*, *HAZE*, *SUNNY*, *SNOWY*}; based on how the weather was like at the moment of the picture's capture. In particular, the request was to address the former problem in 2 different ways:

1. Defining and training a CNN from scratch for this particular task;
2. Applying transfer learning and fine-tuning on a pre-trained model.

1.2 Dataset

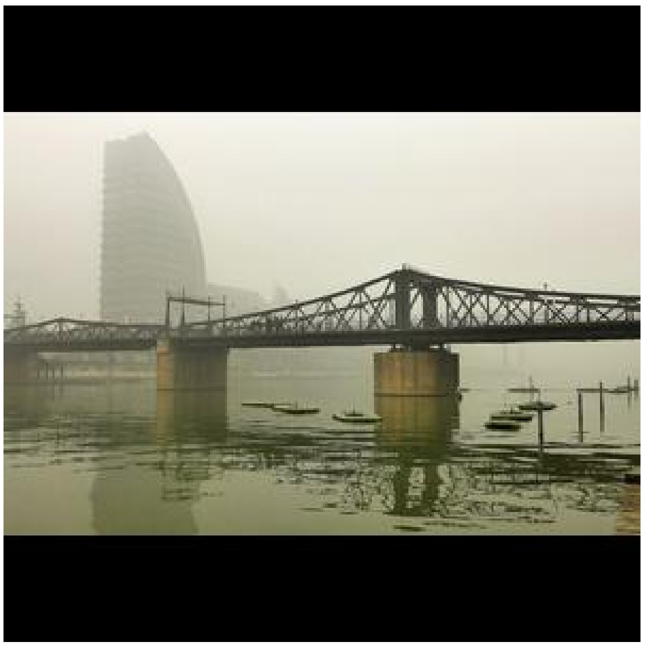
The provided datasets[1] are separated in several different archives and folders. I have chosen to use *MWI-Dataset-1.1_2000* for training, *MWI-Dataset-1.2.4* for testing and *MWI-Dataset-1.2.5* for validation. All these datasets are perfectly balanced, in particular the training dataset is composed by 2000 pictures, 500 per class, while the validation and testing datasets are composed by 400 pictures each, 100 per class.

Chosen resolution The pictures in the datasets don't share a common resolution, this is why I had to resize them in order to be able to train and test.

In particular, I have chosen to use 299x299 as the pictures' resolution; I have tried other resolutions but this seemed to outperform the others(as expected since the chosen pre-trained model was originally trained on that resolution). I have tried 2 different approaches to resize the pictures:

1. **Padded**: Preserving the original aspect ratio, I have resized the longest side to 299, keeping constant the aspect ratio; i.e. proportionately resizing also the other side; and pasted the so-obtained picture on a black 299x299 square; basically adding some padding to the edges. As an example of this resizing process here is one sample picture:

(a) Original picture with resolution 900x601


(b) Resized picture with resolution 299x299

2. **Non-Padded**: I've used the keras.preprocessing.image.ImageDataGenerator flow_from_directory method to resize the pictures.

Although, differently from what I expected, the second, more classical, method of resizing performed better.

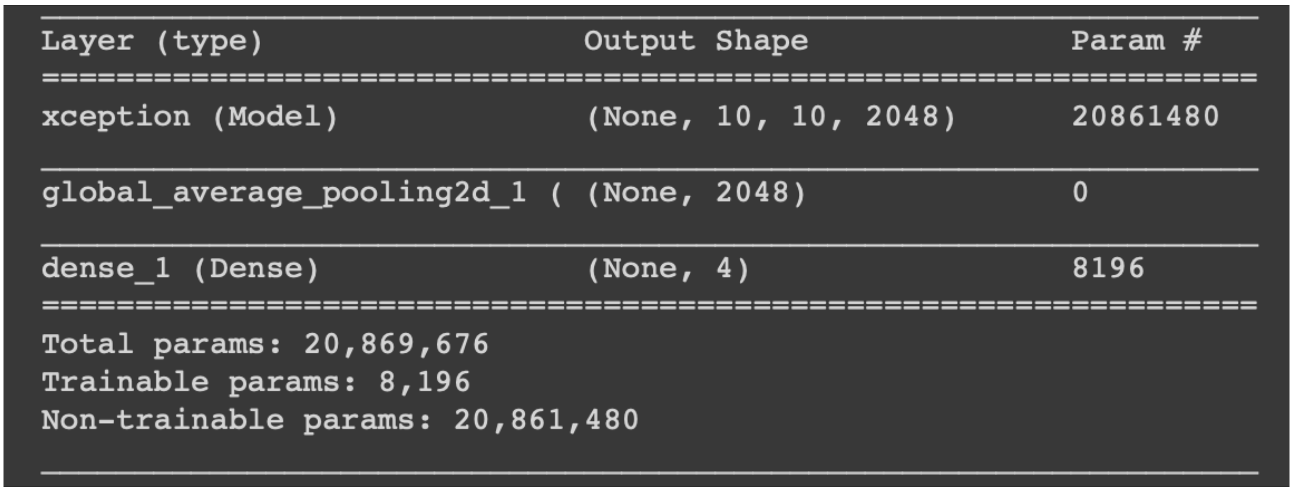
[1]available here: https://drive.google.com/drive/folders/1UzH28Q8xki8_DMYdDgHxi40-CJ800Kaq

2 Models

2.1 Pre-trained Model

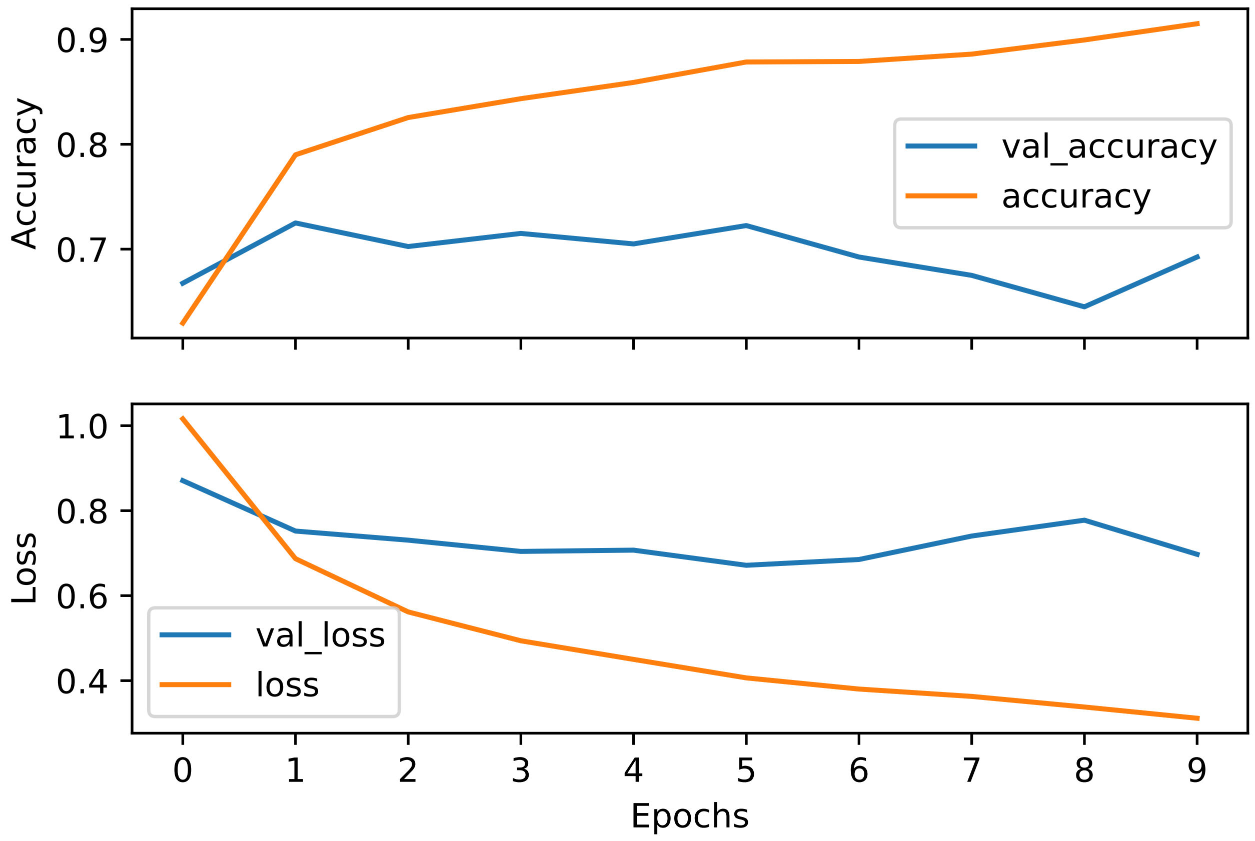
After looking at the pre-trained models available here, I have chosen to use the Xception model, characterised by a good ratio between size and accuracy.

The first pre-trained model that I tried is the following one, note that the xception model is completely frozen, i.e. its layers are not going to be modified during the backtracking phase:

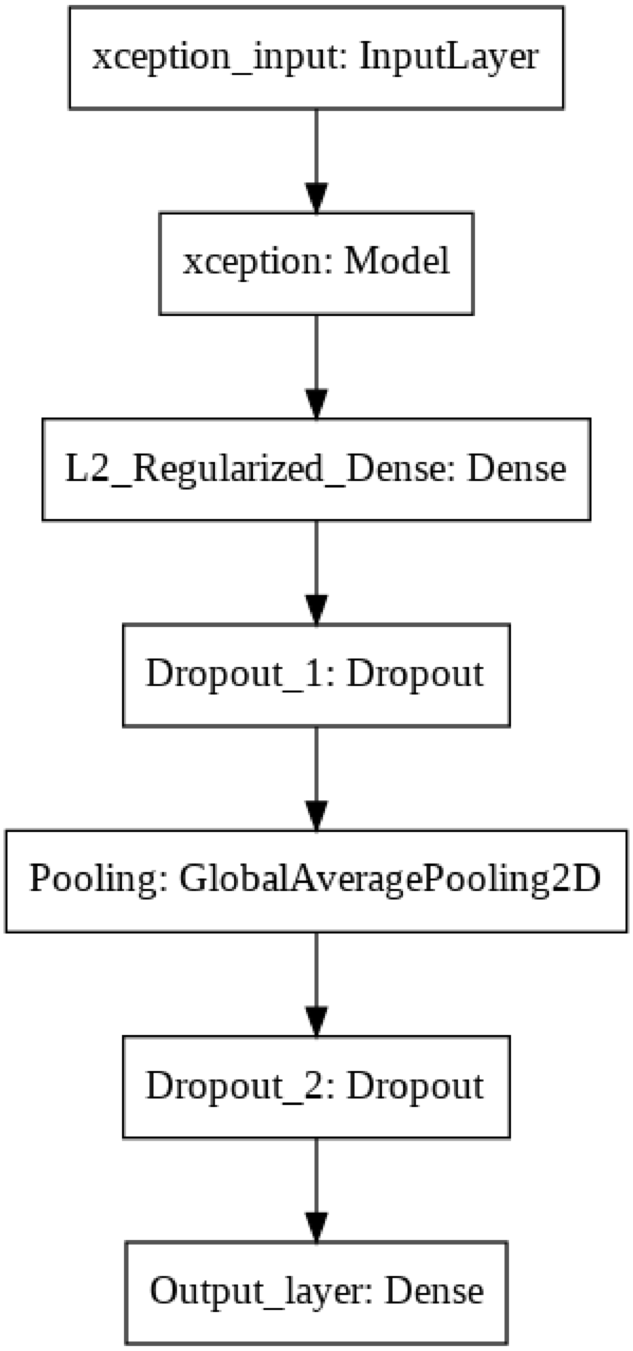


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Model)             (None, 10, 10, 2048)      20861480
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 8196
=================================================================
Total params: 20,869,676
Trainable params: 8,196
Non-trainable params: 20,861,480
_________________________________________________________________
```

Although, also, due to the small number of samples in the training dataset, this model performed poorly, clearly overfitting in less than 10 epochs:



In order to alleviate this problem I have decided to try to add some dropout and regularization terms in the model, the final model which I obtained is the following one:


xception_input: InputLayer
xception: Model
L2_Regularized_Dense: Dense
Dropout_1: Dropout
Pooling: GlobalAveragePooling2D
Dropout_2: Dropout
Output_layer: Dense

2.2 CNN Model

In order to be able to better estimate the influence of the pre-trained weights on a niche task, like this one, with few training samples I have decided to use the very same model with the xception model's weights randomly initialized.

3 Training

3.1 Hyperparameters

Both the models share the following hyperparameters:

- **Dense size**: The size of the regularized dense layer;
- **Regularization term**: The value for the kernel regularizer and the bias one;
- **Dropout rate**: Shared among the 2 dropout layers;
- **Activation Functions**: I have used 'softmax' for the output layer and 'relu' for the regularized dense layer;
- **Optimization Function**: I have just used RMSProp;
- **Learning rate**: The value used by RMSProp;
- **Rho**: Other value used by RMSProp which I have set to 0.9, as suggested in the keras documentation;
- **Epochs**: The number of epochs to train the models;
- **Batch size**: I have used 50.

3.2 GridSearch

In order to try to optimize as much as possible the 2 models I wanted to tune the hyperparameters performing a GridSearch, which in the end I carried out only partially due to time and computational power limitations.

In particular, I wanted to test the following values:

- **Dense size**: {64, 128};
- **Regularization term**: {0.001, 0.01};
- **Dropout rate**: {0.3, 0.5};
- **Learning rate**: {0.0001, 0.001};
- **Epochs**: I have used early stopping on the validation loss with patience=3