Report: HomeWork_02

Federico Kieffer
(n.m.1722271)

The present work focuses on the solution of a Machine Learning (ML) multi-class image classification problem. In particular, a 4-class weather classification problem has been tackled using Convolutional Neural Networks (CNN). More precisely, we refer to weather classification from images as the task of predicting if the weather is *Haze*, *Rainy*, *Snowy* or *Sunny*. This image classification problem has been addressed in two different ways: firstly, a CNN has been defined and trained from scratch and, secondly, transfer learning and fine tuning from a pre-trained model have been applied.

PROBLEM 1:

Each of the two above mentioned networks have been trained using two different sets of images: at first, only the original images of the MWI dataset have been used and, secondly, an augmented version of this dataset has been fed to the CNN. The performances of the resulting models have been then compared. In particular, in view of the data augmentation to be performed, only 400 images of the MWI original dataset have been considered so as to avoid computational inefficiency of the implemented algorithms.
In particular, the data augmentation has been performed following the idea of learning dissimilar features of the same input image. At this purpose, each original image has been filtered in three different ways (ref. Code "Problem1", Section: "Create augmented training set") :

1) Low-Pass Filter (LPF)

2) Spectral Filter

3) High-Pass Filter (HPF)

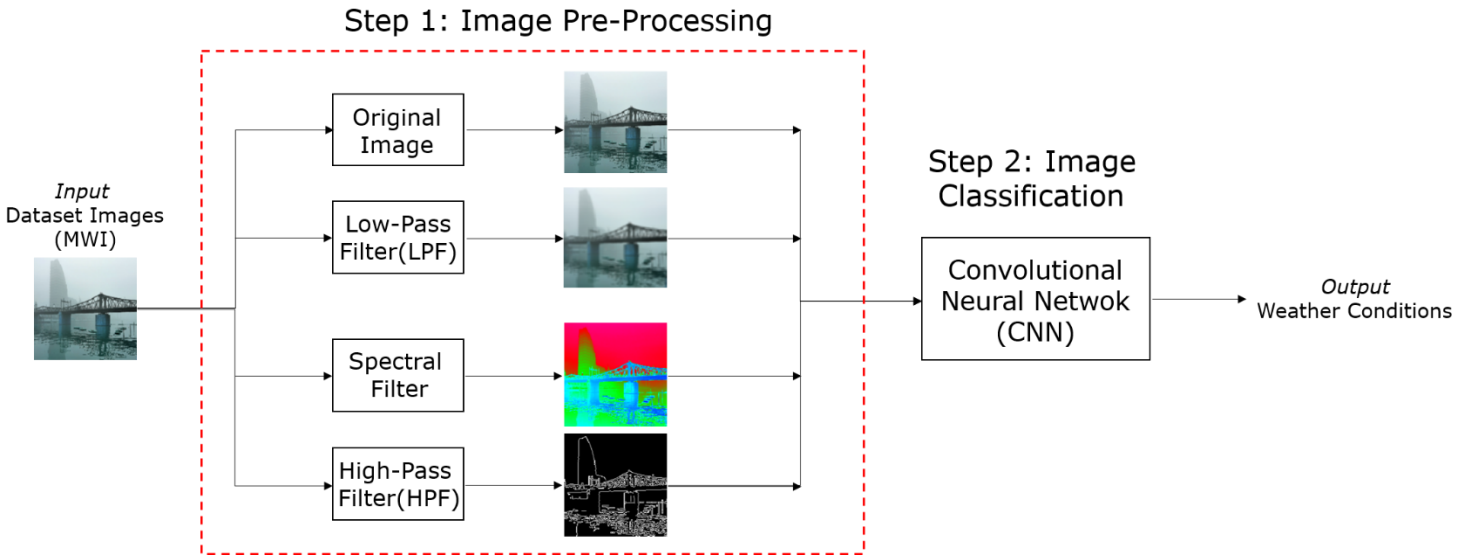

Figure 1: Image Pre-Processing

The LPF operation produces an image Blurring. In this context, "frequency" refers to the change of pixel values. Around edges, pixel values change rapidly and this is why filtering out high frequencies yields blurred images, in which edges are not clearly distinguishable.

For the same reason, HPF returns an edge-detected version of the image.

Finally, a spectral filter has been applied applied to the input image to highlight the regions of low contrast, which enables the classifier to learn features that are more global. In other words, this last image type essentially extracts the hazy or sunny sky itself, while in the high-pass-filtered images the objects that characterize the image are highlighted.

To these pre-processed images is added the original dataset of 400 images, yielding an overall number of 1600 training samples (figure 1).

Of course, each of the pre-processed images has been associated to the label that characterizes the corresponding original image. As further pre-processing step, all images have been reshaped so as to make their dimensions consistent with the input layer of the network. In particular, a scaling of the images has been performed, resizing them all into a 200x200x3 (color images) square format. In addition, the images have been normalized by dividing by 255 all their numerical pixel values. Finally, both the original and the augmented dataset

have been shuffled so ad to avoid that too many images of the same class or type were consecutively fed to the CNN during the training phase.
The implemented CNN has quite a simple structure, which is summarized in the following table (ref. Code "Problem1", Section: "Create the model")

Layer Type	Num of Filters/Units	Size of Kernel/Pool	Stride	Activation	Padding
1° Conv. layer	80	1x1	2x2	'relu'	'valid'
Max-pooling layer	1	2x2	1x1	/	'valid'
2° Conv. layer	60	1x1	2x2	'relu'	'valid'
Max-pooling layer	1	2x2	2x2	/	'valid'
3° Conv. layer	40	1x1	1x1	'relu'	'valid'
Max-pooling layer	1	2x2	1x1	/	'valid'
1° Full-conn. layer	100	/	/	'relu'	/
Dropout layer	/	/	/	/	/
Output Layer	4	/	/	'softmax'	/

Table 1: KiffaNet

The images' features are learnt via a sequence of three convolutional layers and two fully-connected layers, which include the classification (output) layer too. A dropout layer has been as well included so as to reduce the overfitting phenomena that may occur during the training phase. In particular, it has been chosen to ignore 30% of the units during the back-propagation algorithm, so as to prevent the CNN to perfectly shape on the data. Finally, the model has been configured for the training phase by specifying the optimizer (adam) and loss function (sparse-categorical-crossentropy) used and the metric (accuracy) monitored during it. In particular, the performances of the two resulting classifiers have been evaluated using as test set 400 images from the MWI dataset. This section of the MWI dataset is of course disjoint from the one used as training set. The training has been carried out over 20 epochs, using bathces of 16 samples. Moreover, the validation-accuracy parameter has been monitored over each epoch so as to trigger an early stopping of the training in case its value did not increase after 4 iterations (ref. Code "Problem1", Section: "Training setup"). Here follows the comparison of the train and test accuracy and loss

observed during the training phase of the two classifiers (ref. Code "Problem1", Section: "Loss and Accuracy plots"):

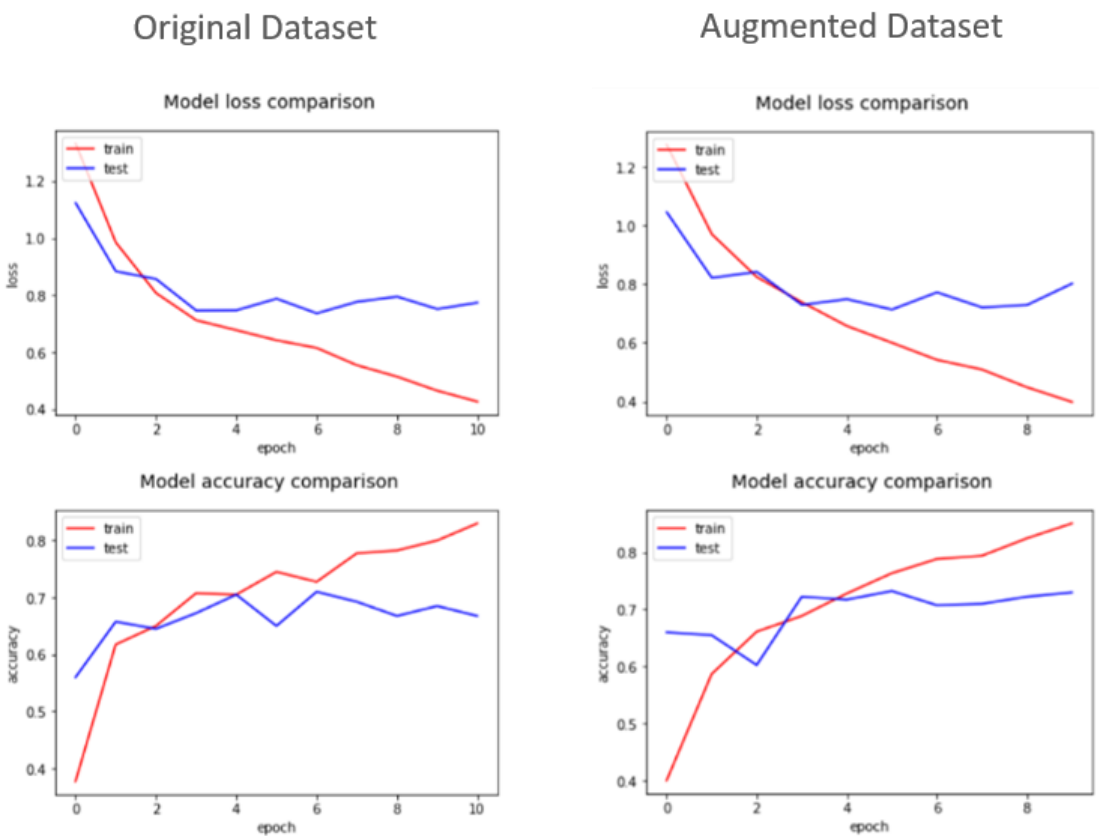

Figure 2: Loss and Accuracy comparison for the model trained on original (left) and augmented (right) data

Although the behaviours of the monitored quantities are quite similar, the best classifier is the one resulting from the training on the augmented dataset. In fact, apart from the classifying performances, this model is also less affected by overfitting as the offset between the train and test accuracy curves is slightly reduced. The performance metrics associated to the two classifiers are in the following reported (Ref. Code "Problem1", Sections: "Performance metrics plot" and "Confusion matrix plot") :

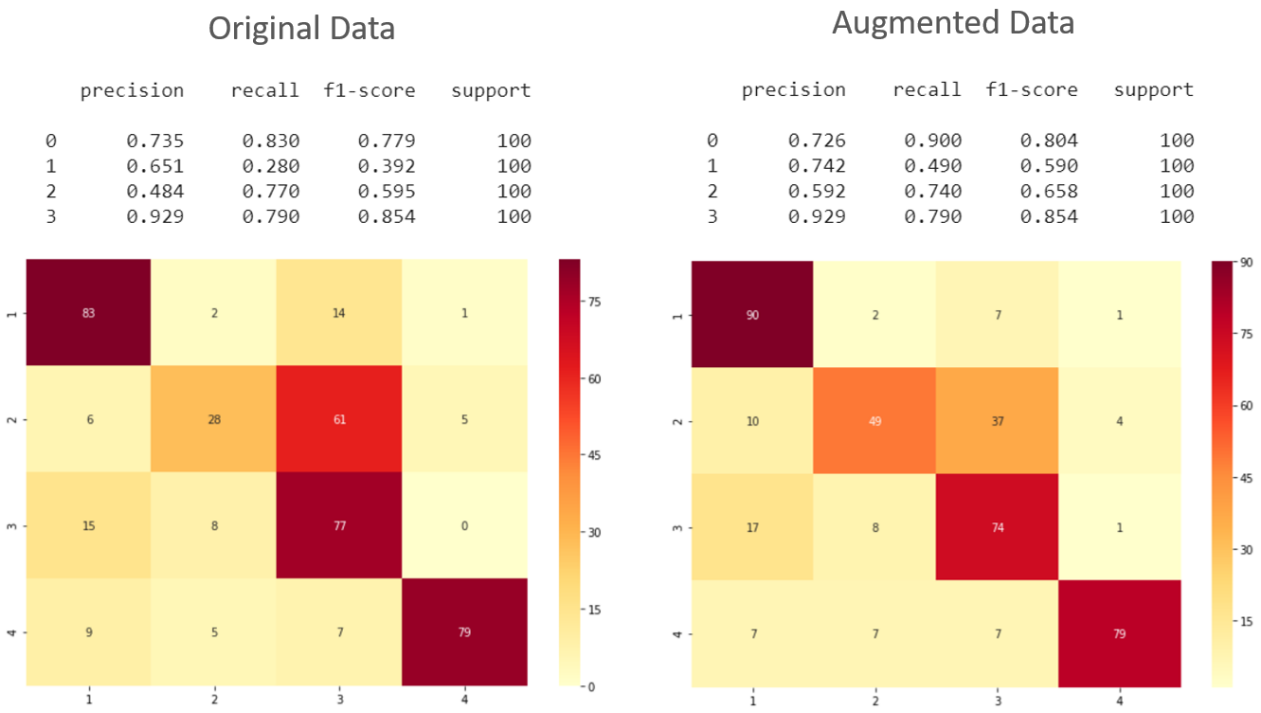

Figure 3: Performance metrics for the model trained on original (left) and augmented (right) data

It is quite evident that both the two classifiers struggle the most in distinguishing images from the two classes *Rainy* and *Snowy*, while they exhibit better performances in correctly classifying *Haze* and *Sunny* images. This result was quite predictable as the features that characterize *Rainy* and *Snowy* images tend to be quite similar and, for this reason, they are more often misinterpreted. More precisely, the most evident defect of the first model is represented by its attitude in wrongly classifying as *snowy* images labeled as *rainy.* The classifier trained on the augmented dataset introduces some improvements with respect to this issue, which are highlighted by both an higher precision of *snowy* and a higher recall of *rainy*. More generally, this rise of performances is also pointed out by an higher f1-score for both the *Rainy* and *Snowy* classes*.* However, the classification of those two more 'critical' labels still remains the most challenging issue for this specific ML problem.

PROBLEM 2:

As second approach, transfer learning techniques have been applied to the same ML problem. First of all, images have been pre-processed by applying the same transformation and filtering procedures already adopted in problem 1.
Secondly, fine-tuning strategies have been performed using both the original and augmented datasets above mentioned. More precisely, the VGG-16 convolutional layers, with weights trained on the ImageNet dataset, have been used as feature extractor. To these *bottom* convolutional layers, four *top* fully-connected layers (including the output layer) have been added, yielding the following overall structure:

Layer Type	Num Of Filters/Units	Size of Kernel/Pool	Stride	Activation	Padding	Kernel Regularizer
VGG-16's Convolutional Layers						
Flatten layer	/	/	/	/	/	/
Batch normalization	/	/	/	/	/	/
Dropout layer	/	/	/	/	/	/
1° full-conn. layer	4096	/	/	'relu'	/	0.01
Batch normalization	/	/	/	/	/	/
Dropout layer	/	/	/	/	/	/
2° full-conn. layer	2048	/	/	'relu'	/	0.01
Batch normalization	/	/	/	/	/	/
Dropout layer	/	/	/	/	/	/
3° full-conn. layer	1024	/	/	'relu'	/	0.01
Batch normalization	/	/	/	/	/	/
Batch normalization	/	/	/	/	/	/
Output layer	4	/	/	'softmax'	/	/

Table 2: Overall Model to be Fine-Tuned

The fine-tuning of this resulting structure has been performed by setting as non-trainable all the feature extractor layers (*frozen* layers) and training only the remaning *top* fully-connected layers (ref. Code "Problem2", Section: "Transfer learning").

Several methods have been applied in order to reduce overfitting. Firstly, Batch-Normalization has been added to each dense layer. This produces a slight regularization effect by normalizing the output of the previous activation layer. Secondly, kernel regularizers allow to apply penalties on layer parameters. These penalties are incorporated in the loss function that the network optimizes during the training phase. Finally, dropout layers have been as well included. In particular, it has been chosen to ignore 40% of the units during the back-propagation algorithm execution. The training has been carried out over 20 epochs, using bathces of 16 samples. Moreover, the validation-accuracy parameter has been monitored over each epoch so as to trigger an early stopping of the training in case its value did not increase after 4 iterations (ref. Code "Problem2", Section: "Training setup"). Here follows the comparison of the train and test accuracy and loss observed during the training phase of the two classifiers:

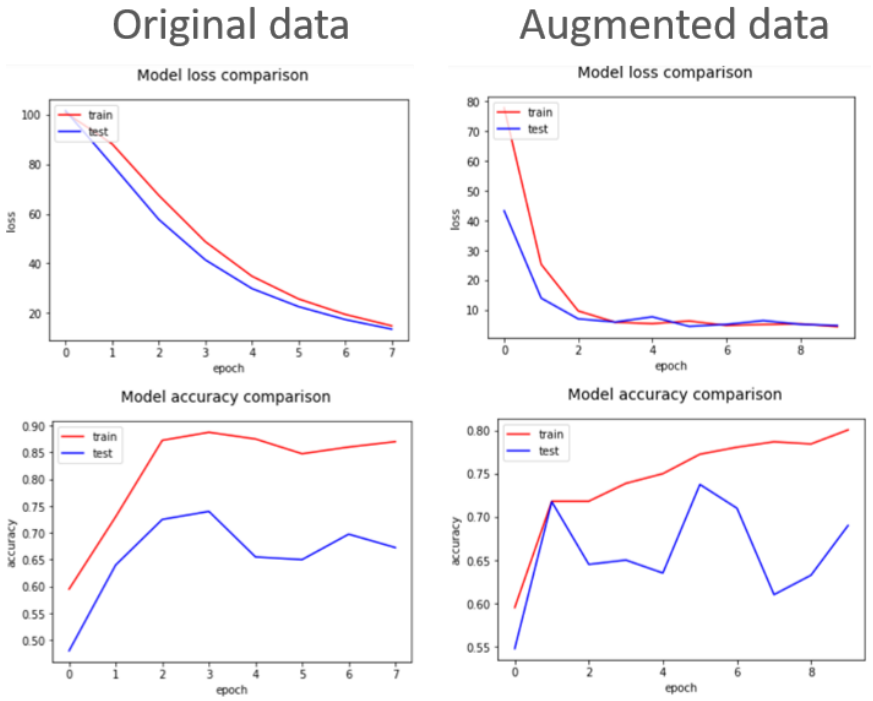

Figure 4: Loss and Accuracy comparison for the model trained on original (left) and augmented (right) datasets

Although several techniques to reduce overfitting have been applied, both the resulting models are still quite affected by this phenomenon. Nevertheless, as highlighted by the smaller offset between the train and test accuracy curves, overfitting is less accentuated for the model fine-tuned on the augmented dataset. This achievement is likely to be linked to the use of an higher number of samples in the training phase (data augmentation). The performance metrics associated to the two classifiers are in the following reported:

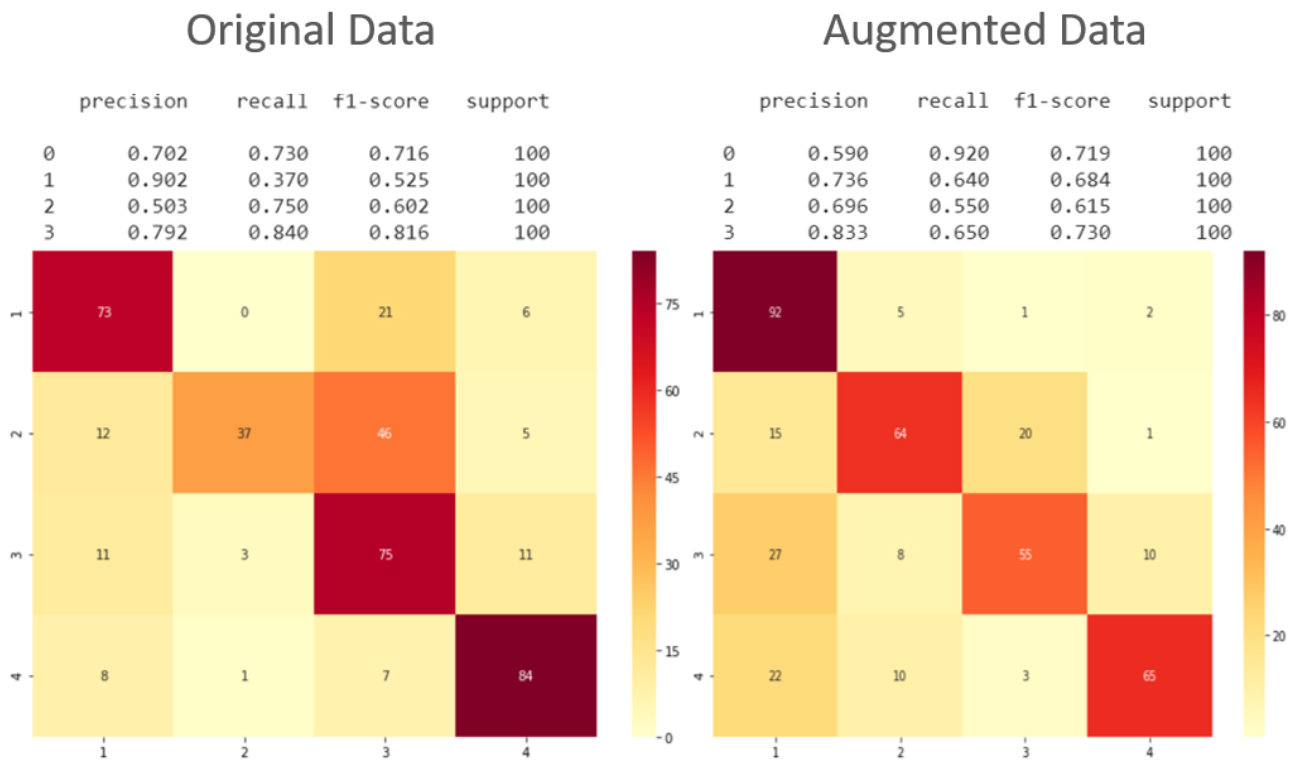

Figure 5: Performance metrics for the model trained on original (left) and augmented (right) data

Once again, the classes that the models struggle the most with are *Rainy* and *Snowy.* The CNN trained on the augmented dataset introduces some slight performance improvements with respect to this issue, which are highlighted by an higher f1-score for both the two classes. However, the price to pay for these enhacements is a sensible performance reduction with respect to the *Sunny* class. The most evident defect of this second classifier is its tendency to classify

as *Haze* the samples of other classes, as confirmed by the quite highly populated first column of the corresponding confusion matrix.
All things considered, the best model among the four above mentioned is the one that has been fine-tuned on the augmented dataset. In fact, this classifier exhibits relatively better performances on all four classes. In other words, there is not one specific class whose corresponding performances are remarkably poorer with respect to the others and therefore this classifier is *on average* the best one. However, if for some reason the *false negatives* associated to the *Rainy* class could be considered of "minor" importance, then the classifier resulting from the training of the *KiffaNet* on the augmented dataset would have been definitely the best one.

CONCLUSIONS:

Two different approaches for a 4-class weather classification problem from images have been proposed. In both cases, two different datasets versions have been used for training a CNN and the performances of the resulting models have been assessed. In particular, accuracy, precision, recall and f1-score have been considered as most significative performance metrics. The associated confusion matricies have been also derived. In the first case, a CNN has been defined and trained from scratch. The best-performing classifier has been in this case obtained using the augmented version of the dataset as input to the CNN. In the second problem transfer learning methodologies have been applied and, in particular, the training of the considered CNN has been performed making use of fine-tuning techniques. Once again, the most satisfacory results have been achieved thanks to the augmented dataset. Basing on the performances assessment, this last classifier has turned out to be the most suitable for this specific ML multi-class classification problem.