Weather Classification

HW2 - Machine Learning
"Sapienza" University of Rome

Valerio Coretti 1635747

April 5, 2020

1 Introduction

Neural networks are the basis of deep learning, and while they may look like black boxes, they are trying to get the same thing as any other model to make good predictions. In this homework we want to solve the problem of the weather calssification, in particular, we have to recognize the weather from an image.

To do this we used the Neural Networks approach. We had a dataset with 4000 images. Each image belongs to a class. Therefore the problem is a Multiclass Classification problem and we have four classes of images: *haze*, *sunny*, *snowy*, *rainy*. The image distribution is very *balanced*, indeed we have 1000 images for class.

What we will see in the following sections is a description of what we did to solve this problem. We worked with *Keras* that is a simple and efficient Python OpenSource tool to implement neural networks. Furthermore, we did not work in the local environment but we used *Colaboratory* that is a tool provided by Google and this is very powerful because it allows us to use a free GPU.

2 Preprocessing

In this part, we modify the dataset. First of all, we split it in *training set*, with 3200 images, and *test set*, whit 800 images.

The structure of the dataset is the following:

```
MWI-Dataset
├── HAZE
│   ├── HAZE-1-1-1_001_ORI.jpg
│   ├── ...
│   └── HAZE-1-2-3_100_ORI.jpg
├── RAINY
│   ├── RAINY-1-1-1_001_ORI.jpg
│   ├── ...
│   └── RAINY-1-2-3_100_ORI.jpg
├── SNOWY
│   └── ...
└── SUNNY
    └── ...
```

This directory structure allows us to use the Keras class called *ImageDataGenerator*. This class is for automated image loading and preprocessing. All the images are loaded with RGB color with batch size 32 and we stretched the images to the target size (200x200). Furthermore, we set the following parameters:

- *rescale* : this is the rescaling factor. Multiply the data by the value provided (after applying all other transformations). Setted to 1/255
- *zoom_range* : this is the range for zoom the image. Setted to 0.1
- *rotation_range* : degree range for random rotations. Setted to 45

In the following paragraphs, we analyze the results of the used networks trough a graphic representation of the accuracy and loss in the training and test set.

3 Well known Networks

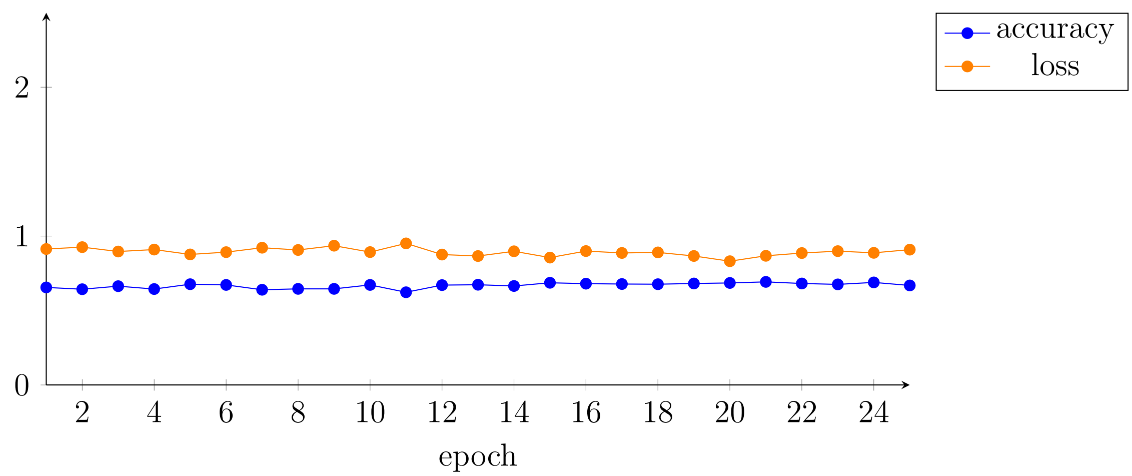
As a first attempt we tried to use two of the well-known networks: *LeNet* and *AlexNet*.

3.1 LeNet

The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier. We test it with 100 epoch but we stop the experiment to the 25^{th} epoch because by increasing the epoch the results did not improve but suffered very small differences and were stable on a certain level.

Input Shape	200 x 200
Total Params	19.566.296
Trainable params	19.566.296
Non trainable params	0

Figure 1: **LeNet – Accuracy and Loss results in test set**



Test set performance:

- Loss: 0.912283;
- Accuracy: 0.679087;

Already with LeNet, we do not have bad performance both in terms of accuracy and loss. Looking at the confusion matrix this network seems to be quite accurate on the haze and sunny classes compared to the other two. We try to do better switching to AlexNet.

3.2 AlexNet

AlexNet is much larger than LeNet, indeed it consists of 5 Convolutional Layers and 3 Dense Layers. More convolutional kernels extract features in the images. The first two convolutional levels are followed by the Max Pooling levels. The third, fourth and fifth convolutional layers are connected directly. The fifth conv is followed by a Max Pooling layer, whose output is divided into a series of two dense levels. The last dense level fits into a softmax classifier. The activation function used for all layers is ReLu, only the last dense layer uses a softmax classifier.

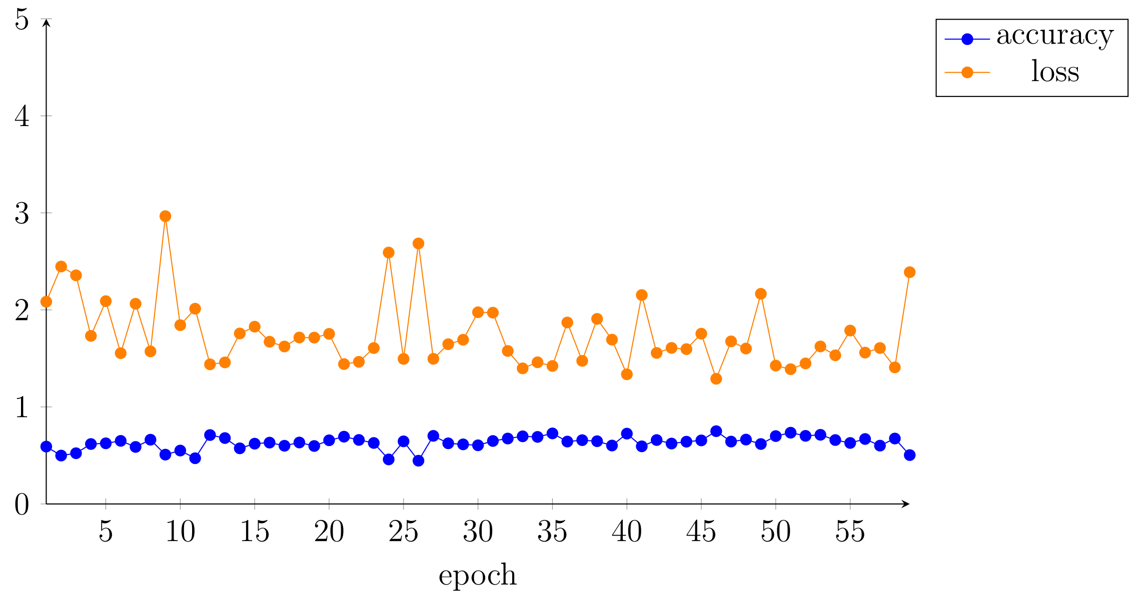
Input Shape	118 x 224
Total Params	28.083.756
Trainable params	28.062.620
Non trainable params	21.136

Test set performance:

- Loss: 1.742811;
- Accuracy: 0.634615;

AlexNet seems worse than LeNet. This probably means that the *AveragePooling* is a better choice than the *MaxPooling* for this type of dataset. The most common mistake of this network is to predict a rainy image as a snowy image. Increasing the epochs we would have good results in the train set, close to 90%, but this does not happen for the performance in the test set and for this reason we stop the train at the 60^{th} epoch.

Figure 2: **AlexNet – Accuracy and Loss results in test set**



We stop here our experiments with well-known networks because the most recent are too complex and in the next section we will try to use neural networks created from scratch.

4 My Neural Networks

In this section, we will see two neural networks created by me. What we will see is the improvement of the performance with very simple networks.

4.1 ValerioNet 1

ValerioNet 1 is a very simple convolutional neural network. It has two Convolution layers each of which is followed by a (2x2) AveragePooling level. The output is then passed to a Flatten layer and then to two Dense layers among which we have a level of Dropout to reduce overfitting. The activation function used is ReLu. In the two Conv layers, we use two different kernels, the first has a size of (5x5) and the second of (3x3). We use also a padding value set to *same*.

Input Shape	200 x 200
Total Params	16.021.432
Trainable params	16.021.43
Non trainable params	0

Figure 3: **ValerioNet1 – Accuracy results**

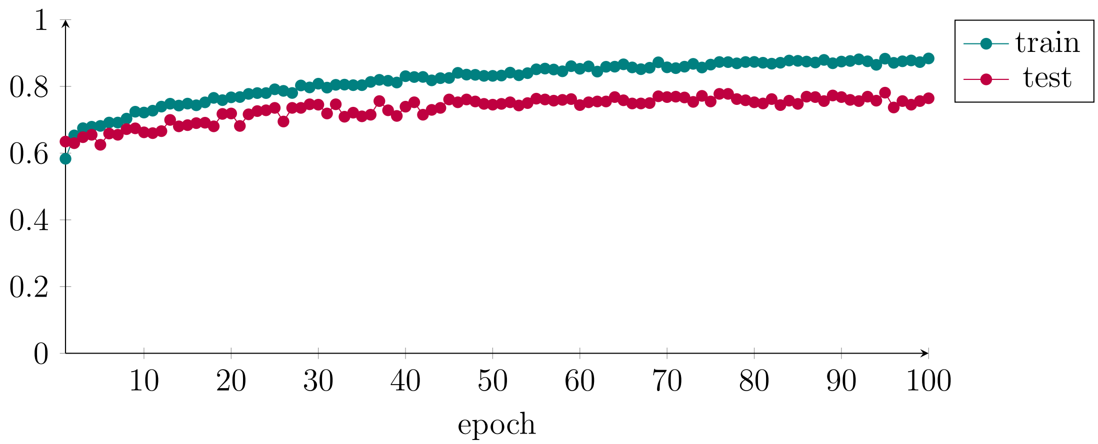
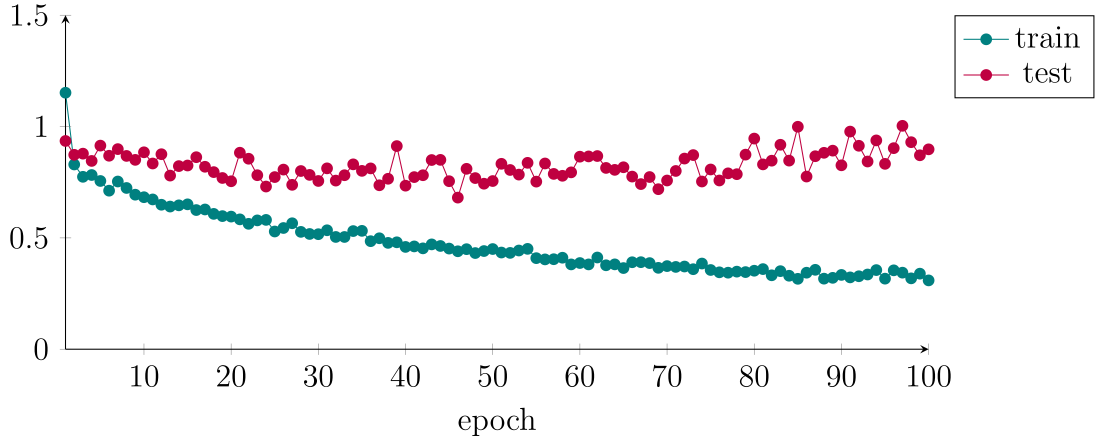


Figure 4: **ValerioNet1 − Loss results**



Test set performance:

- Loss: 0.870954;
- Accuracy: 0.769231;

Oh well this results are very nice than those seen before. My net is better than AlexNet and LeNet, this means that i won (:D). But analyze the results. In the following figure we see the confusion matrix and the value of precision and recall:

```
Found 800 images belonging to 4 classes.
25/25 [==============================] - 8s 327ms/step
[[155  15  13  17]
 [  8 147  35  10]
 [  7  34 152   7]
 [ 12  16  14 158]]
True                 Predicted           errors  err %
------------------------------------------------------------------
RAINY           ->   SNOWY               35      4.38 %
SNOWY           ->   RAINY               34      4.25 %
HAZE            ->   SUNNY               17      2.12 %
SUNNY           ->   RAINY               16      2.00 %
HAZE            ->   RAINY               15      1.88 %
SUNNY           ->   SNOWY               14      1.75 %
HAZE            ->   SNOWY               13      1.62 %
SUNNY           ->   HAZE                12      1.50 %
RAINY           ->   SUNNY               10      1.25 %
RAINY           ->   HAZE                8       1.00 %
SNOWY           ->   HAZE                7       0.88 %
SNOWY           ->   SUNNY               7       0.88 %
```

```
Found 800 images belonging to 4 classes.
25/25 [==============================] - 8s 321ms/step
              precision    recall  f1-score   support

        HAZE      0.852     0.775     0.812       200
       RAINY      0.693     0.735     0.714       200
       SNOWY      0.710     0.760     0.734       200
       SUNNY      0.823     0.790     0.806       200

    accuracy                          0.765       800
   macro avg      0.770     0.765     0.766       800
weighted avg      0.770     0.765     0.766       800
```

As we can see from the images, this network has nice results but also, in this case, the most common error is the misclassification of the rainy with snowy. This network is very simple and not very deeper, generally, two hidden layers are enough but in theory, a deeper network is better because it needs fewer neurons to reach the results like two hidden layers. For this reason, in the following section, we try to make ValerioNet1 deeper and we compare the results.

4.2 ValerioNet 2

ValerioNet 2 is deeper than the previous version. It is a mix of the three networks seen before. Indeed it has 5 Convolutional layers, the first two use *tanh* activation function, and a (5x5) kernel, like LeNet layers, and they are followed by a (2x2) AveragePooling level. The next three Conv layers are like AlexNet, indeed the 3^{rd}, 4^{th} and the 5^{th} are followed by a Batch Normalization layer and use ReLu activation function, with (3x3) kernels. The output is then passed to a Flatten layer and then to two Dense layers among which we have a level of Dropout and other two levels of Batch Normalization, to reduce overfitting. The last Dense layer uses a softmax classifier. The optimizer used is *adam*. The complete structure is the following:

ValerioNet 2			
layer	**kernels**	**strides**	**activation**
Convolutional 6	5x5	1x1	tanh
AveragePooling 2x2		2x2	
Convolutional 16	5x5	1x1	tanh
AveragePooling 2x2		2x2	
Convolutional 16	3x3	1x1	ReLu
BatchNormalization			
Convolutional 128	3x3	1x1	ReLu
BatchNormalization			
Convolutional 256	3x3	1x1	ReLu
AveragePooling 2x2		2x2	
BatchNormalization			
Flatten			
Dense 1000			ReLu
Dropout 0.5			
BatchNormalization			
Dense 500			ReLu
BatchNormalization			
Dense 4			softmax

Input Shape	200 x 200
Total Params	113.726.024
Trainable params	113.722.224
Non trainable params	3.800

Figure 5: **ValerioNet2 – Accuracy results**

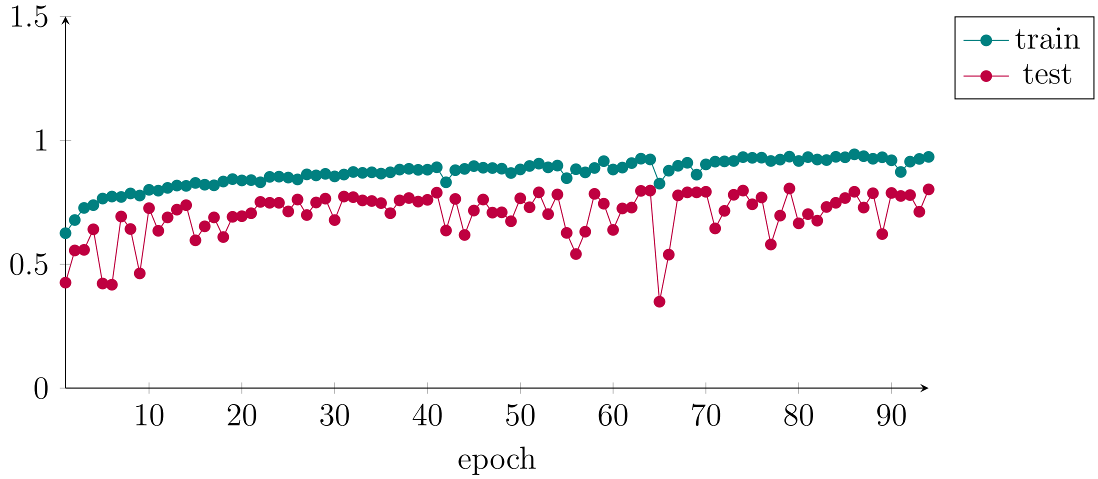
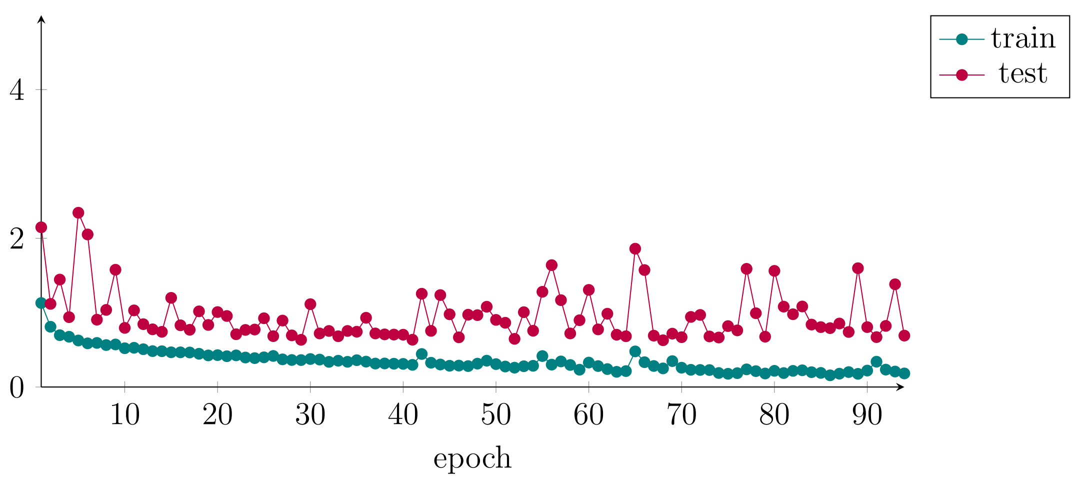


Figure 6: **ValerioNet2 – Loss results**



Final epoch performance:

- Loss: 0.1825;
- Accuracy: 0.9328;

Test set performance:

- Loss: 0.691415;
- Accuracy: 0.801683;

Confusion matrix and score:

```
Found 800 images belonging to 4 classes.
25/25 [==============================] - 9s 366ms/step
              precision    recall  f1-score   support

        HAZE      0.886     0.815     0.849       200
       RAINY      0.698     0.810     0.750       200
       SNOWY      0.790     0.735     0.762       200
       SUNNY      0.843     0.835     0.839       200

    accuracy                          0.799       800
   macro avg      0.804     0.799     0.800       800
weighted avg      0.804     0.799     0.800       800
```

```
Found 800 images belonging to 4 classes.
25/25 [==============================] - 9s 343ms/step
[[163  12   9  16]
 [  4 162  23  11]
 [  5  44 147   4]
 [ 12  14   7 167]]
True                 Predicted          errors  err %
-------------------------------------------------------
SNOWY            ->  RAINY              44      5.50 %
RAINY            ->  SNOWY              23      2.88 %
HAZE             ->  SUNNY              16      2.00 %
SUNNY            ->  RAINY              14      1.75 %
HAZE             ->  RAINY              12      1.50 %
SUNNY            ->  HAZE               12      1.50 %
RAINY            ->  SUNNY              11      1.38 %
HAZE             ->  SNOWY              9       1.12 %
SUNNY            ->  SNOWY              7       0.88 %
SNOWY            ->  HAZE               5       0.62 %
RAINY            ->  HAZE               4       0.50 %
SNOWY            ->  SUNNY              4       0.50 %
```

Once again we have improved performance to a very good result. Looking at the confusion matrix, even with this network the problem is in the classification of the rainy images, but we improved the performance in the other classes especially for the haze class. The problem of this network is that, as we can see from the graphs, the results in the test set is a bit unstable. So we won't immediately discard ValerioNet1 because, even with slightly lower results, it is more stable and could be the best choice.

However, even trying to modify this network to increase performance, this is the best result we have been able to achieve. But this was predictable given the very small size of the dataset. For this reason, we stop our analysis with neural networks here and continue reporting some experiments with *Transfer Learning*.

5 Transfer Learning

Transfer Learning is a Machine Learning technique based on which a model is trained and developed for activity and is then reused in a second related activity. It refers to the situation in which what was learned in a setting is used to improve optimization in another setting. Transfer Learning allows us to start with the learned features on the dataset and adjust the structure of the model, instead of starting the learning process on the data from scratch with random weight initialization. It is usually applied when there is a small dataset.

In this homework, we use Transfer Learning to improve the performances of ValerioNet1 and ValerioNet2.

5.1 ValerioNet 1 - transfer model

Figure 7: **ValerioNet1 Transfer – Accuracy results**

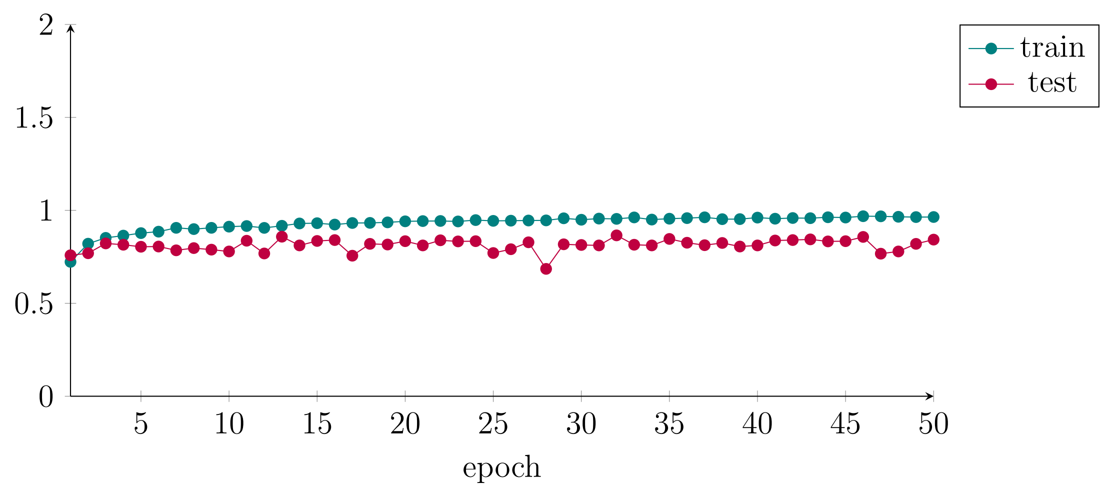
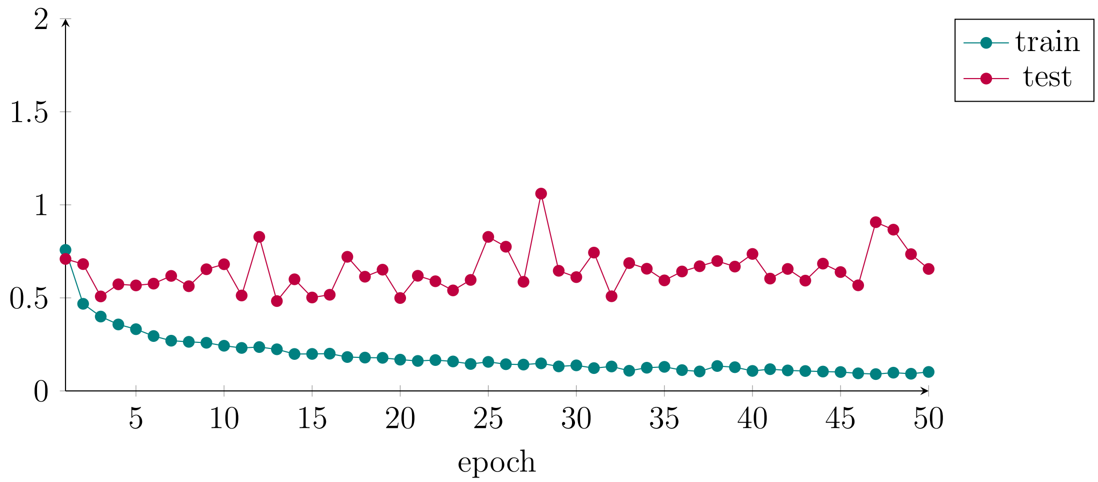


Figure 8: **ValerioNet1 Transfer – Loss results**



Final epoch performance:

- Loss: 0.1020;
- Accuracy: 0.9641;

Test set performance:

- Loss: 0.655136;
- Accuracy: 0.842548;

As we expected the results with the transfer model have improved and seem to be very good! As we can see in the charts the models are more stable and more reliable. We also report the results of the confusion matrix and the various scores to confirm what has just been said.

```
Found 800 images belonging to 4 classes.
25/25 [==============================] - 12s 466ms/step
              precision    recall  f1-score   support

        HAZE      0.927     0.825     0.873       200
       RAINY      0.737     0.910     0.814       200
       SNOWY      0.863     0.755     0.805       200
       SUNNY      0.880     0.880     0.880       200

    accuracy                          0.843       800
   macro avg      0.852     0.842     0.843       800
weighted avg      0.852     0.843     0.843       800
```

```
Found 800 images belonging to 4 classes.
25/25 [==============================] - 12s 487ms/step
[[165  15   5  15]
 [  2 182   9   7]
 [  6  41 151   2]
 [  5   9  10 176]]
True                 Predicted          errors  err %
-------------------------------------------------------
SNOWY           ->   RAINY              41      5.12 %
HAZE            ->   RAINY              15      1.88 %
HAZE            ->   SUNNY              15      1.88 %
SUNNY           ->   SNOWY              10      1.25 %
RAINY           ->   SNOWY              9       1.12 %
SUNNY           ->   RAINY              9       1.12 %
RAINY           ->   SUNNY              7       0.88 %
SNOWY           ->   HAZE               6       0.75 %
HAZE            ->   SNOWY              5       0.62 %
SUNNY           ->   HAZE               5       0.62 %
RAINY           ->   HAZE               2       0.25 %
SNOWY           ->   SUNNY              2       0.25 %
```

5.2 ValerioNet 2 - transfer model

Figure 9: **ValerioNet2 Transfer – Accuracy results**

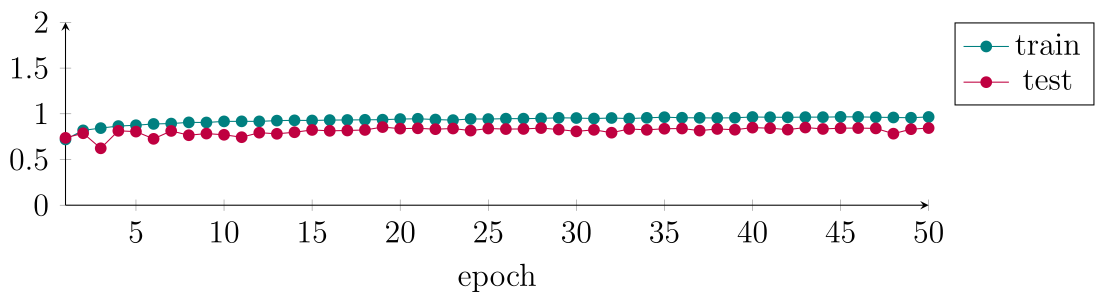
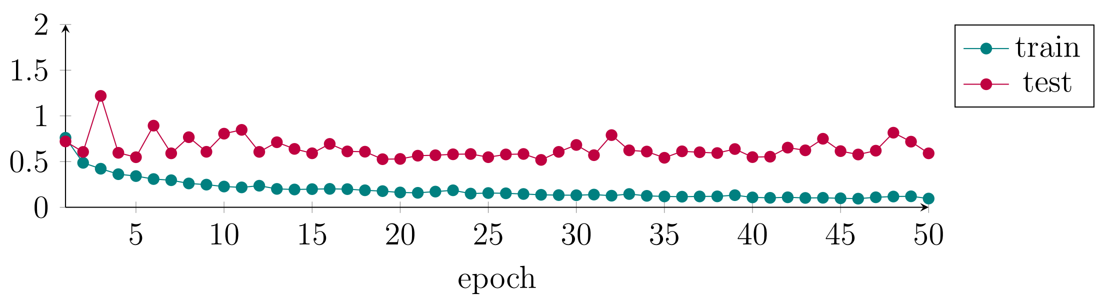


Figure 10: **ValerioNet2 Transfer – Loss results**



Final epoch performance:

- Loss: 0.0955;
- Accuracy: 0.9656;

Test set performance:

- Loss: 0.658900;
- Accuracy: 0.835337;

Confusion matrix and score:

```
Found 800 images belonging to 4 classes.
25/25 [==============================] - 9s 355ms/step
              precision    recall  f1-score   support

        HAZE      0.861     0.870     0.866       200
       RAINY      0.840     0.760     0.798       200
       SNOWY      0.760     0.870     0.811       200
       SUNNY      0.883     0.830     0.856       200

    accuracy                          0.833       800
   macro avg      0.836     0.833     0.833       800
weighted avg      0.836     0.833     0.833       800
```

```
Found 800 images belonging to 4 classes.
25/25 [==============================] - 9s 372ms/step
[[174   3   9  14]
 [  7 152  33   8]
 [  9  17 174   0]
 [ 12   9  13 166]]
True                 Predicted          errors  err %
-------------------------------------------------------
RAINY           ->  SNOWY              33      4.12 %
SNOWY           ->  RAINY              17      2.12 %
HAZE            ->  SUNNY              14      1.75 %
SUNNY           ->  SNOWY              13      1.62 %
SUNNY           ->  HAZE               12      1.50 %
HAZE            ->  SNOWY              9       1.12 %
SNOWY           ->  HAZE               9       1.12 %
SUNNY           ->  RAINY              9       1.12 %
RAINY           ->  SUNNY              8       1.00 %
RAINY           ->  HAZE               7       0.88 %
HAZE            ->  RAINY              3       0.38 %
```

Also, in this case, we have improved all the scores compared to the original model, obtaining really good results. However it turns out to be slightly worse than *ValerioNet1− Transfer Model* and this confirms the fact that ValerioNet2 was probably less stable than ValerioNet1, so we prefer the latter. For this reason, in the next sections, we will only use ValerioNet1 and its transfer model to make predictions.

6 SMART-I Dataset

In addition to the dataset with which we did the training we also had that of the company SMART−I that contains 3038 images. To test our models we therefore decided to make predictions on this dataset. Accuracy in this case is not a good measure for the evaluation since the dataset is really unbalanced: 0 *haze*, 521 *rainy*, 1421 *snowy*, 1096 *sunny*. We do the experimentation only with ValerioNet1 and relative transfer model because ValerioNet2, being unstable, does not perform very well. The following are the results:

ValerioNet1:

```
Found 3038 images belonging to 4 classes.
95/95 [==============================] - 25s 261ms/step
              precision    recall  f1-score   support

        HAZE      0.000     0.000     0.000         0
       RAINY      0.283     0.852     0.425       521
       SNOWY      0.764     0.569     0.652      1421
       SUNNY      0.627     0.097     0.168      1096

    accuracy                          0.447      3038
   macro avg      0.419     0.379     0.311      3038
weighted avg      0.632     0.447     0.438      3038
```

```
Found 3038 images belonging to 4 classes.
95/95 [==============================] - 24s 255ms/step
[[  0   0   0   0]
 [ 31 444  31  15]
 [148 417 808  48]
 [ 63 709 218 106]]
True                 Predicted          errors  err %
-------------------------------------------------------
SUNNY           ->   RAINY              709     23.34 %
SNOWY           ->   RAINY              417     13.73 %
SUNNY           ->   SNOWY              218     7.18 %
SNOWY           ->   HAZE               148     4.87 %
SUNNY           ->   HAZE               63      2.07 %
SNOWY           ->   SUNNY              48      1.58 %
RAINY           ->   HAZE               31      1.02 %
RAINY           ->   SNOWY              31      1.02 %
RAINY           ->   SUNNY              15      0.49 %
```

Test set performance:

- Loss: 4.196499;
- Accuracy: 0.447005;

ValerioNet1−Transfer Model

```
              precision    recall  f1-score   support

        HAZE      0.000     0.000     0.000         0
       RAINY      0.278     0.816     0.415       521
       SNOWY      0.811     0.575     0.673      1421
       SUNNY      0.703     0.169     0.272      1096

    accuracy                          0.470      3038
   macro avg      0.448     0.390     0.340      3038
weighted avg      0.681     0.470     0.484      3038
```

```
[[  0   0   0   0]
 [ 67 425  26   3]
 [144 385 817  75]
 [ 28 718 165 185]]
True                 Predicted          errors  err %
-------------------------------------------------------
SUNNY           ->   RAINY              718     23.63 %
SNOWY           ->   RAINY              385     12.67 %
SUNNY           ->   SNOWY              165     5.43 %
SNOWY           ->   HAZE               144     4.74 %
SNOWY           ->   SUNNY              75      2.47 %
RAINY           ->   HAZE               67      2.21 %
SUNNY           ->   HAZE               28      0.92 %
RAINY           ->   SNOWY              26      0.86 %
RAINY           ->   SUNNY              3       0.10 %
```

Test set performance:

- Loss: 3.305563;

- Accuracy: 0.469717;

In both cases, we do not have good performances but we directly analyze those of *ValerioNet1−Transfer Model* which are slightly better. Being the dataset unbalanced, we take the weighted average of the various scores, and we see that the images classified as correct only 68% were correct, instead of all the corrected images only 47% were classified as correct. The misclassification of sunny as rainy and snowy as rainy is the most common error. So with this dataset, our models seem to classify the images as more rainy than they should be.

However, even if the results are not very good, this test helped us to confirm ValerioNet1-Transfer Model as the final model.

7 Conclusion

We report in a table the results:

Model	Input Shape	Accuracy	Loss
LeNet	200x200	0.679087	0.912283
AlexNet	118x224	0.634615	1.742811
ValerioNet1	200x200	0.769231	0.870954
ValerioNet2	200x200	0.801683	0.691415
ValerioNet1 TL	200x200	0.842548	0.655136
ValerioNet2 TL	200x200	0.835337	0.658900

As we can see ValerioNet1−TL is the best and for this reason, we have chosen it to make the predictions in the blind set.

References

[1] *Keras Documentation.*
`https://keras.io`

[2] *LeNet-5 – A Classic CNN Architecture.*
`https://engmrk.com/lenet-5-a-classic-cnn-architecture/`

[3] *Understanding AlexNet.*
`https://www.learnopencv.com/understanding-alexnet/`

[4] *Transfer Learning - Wikipedia.*
`https://en.wikipedia.org/wiki/Transfer_learning`