ML – Homework2 –

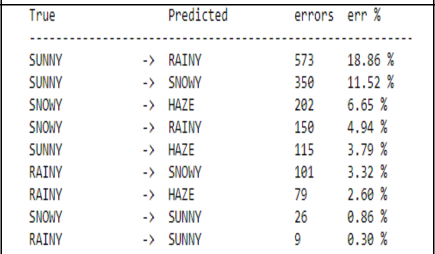
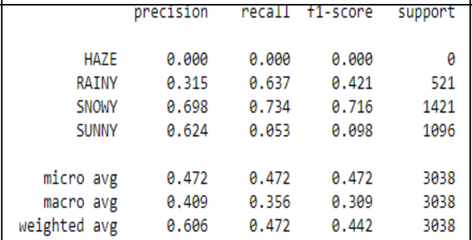
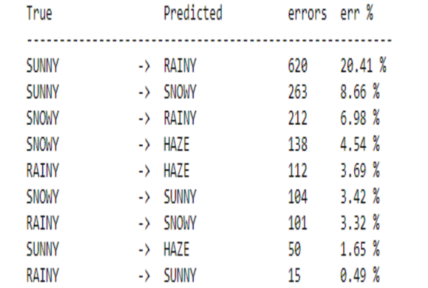
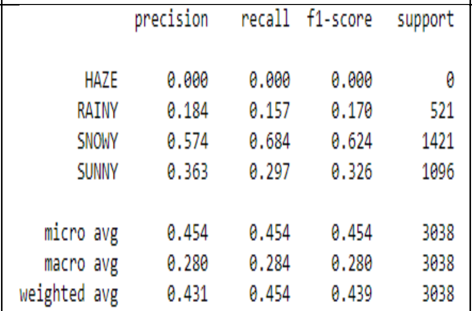
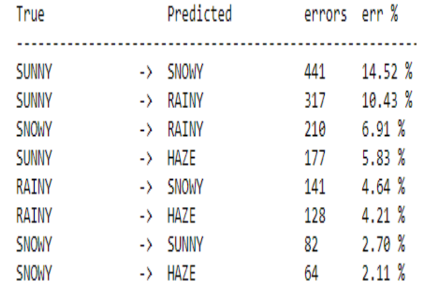
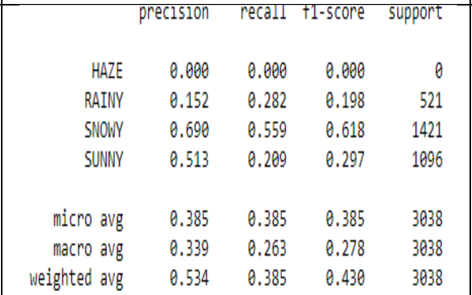
Multi-class weather classification

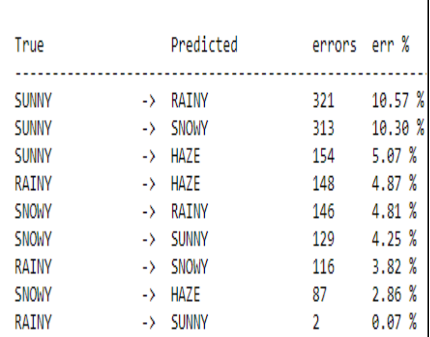
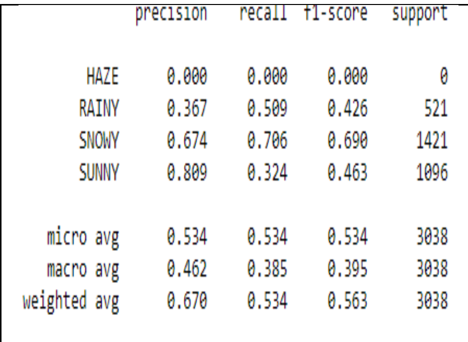
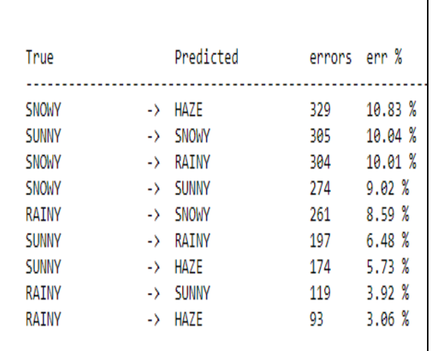
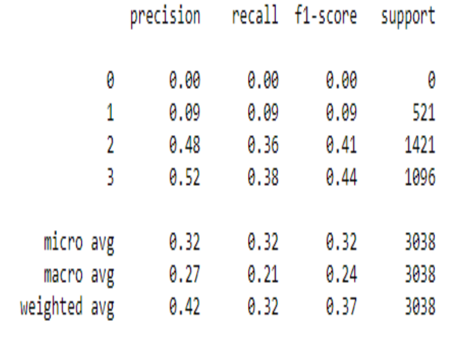
Theodoros Sofianos 1867968

Introduction: For the needs of this homework, 4 models were created. I have implemented 2 CNN models from scratch, one using the 400 images from MWI dataset 1.1 and the other using the 2000 images MWI dataset, as training set. Also, I created a model using transfer learning and fine tuning, trained for both of the datasets listed above. Lastly, I performed feature extraction from the transfer learning model and fed these features as inputs to a linear SVM, using the 2000 images dataset. All of these models were tested with the TestSet Weather dataset, compiled of 3038 images. The images of the test set are from a CCTV camera and because of the images diversity between this test set and the training set(angle of picture taken, images at night), the above models were also checked using cross validation. Although the test set contains only 3 classes, the models were trained with all the 4 classes of the training set. For that purpose, an empty folder called HAZE was added to the test set dataset. Also, since the testset is really imbalanced, the metric to evaluate which of my models classifies better its instances is the f1 score, weighted by the support of each class.

Data Augmentation and image rescaling: The Data Augmentation part is essential for designing a convolutional neural network, either from scratch or using transfer learning. Since the images inside the test folder are different from the ones in the training set (angle of the picture taken, pictures at night), I tried to increase the data augmentation of the training set, so that the model is prevented from overfitting on it and in order to have some predictability skills on the test set. More precisely, the image brightness, zoom range, width and height range will be changed in every image in the test set, by a random factor inside a range. Horizontal flipping of the image will also be introduced, along with zca whitening, to reduce the redundancy in the matrix of pixel images ,in order to highlight better the structures and features in the image to the learning algorithm. The classical RGB representation of the pixel matrix will be rescaled by 255, so all the values will be in the range of (0,1). Of course, since the images don't have the same dimensions, we will crop, scale and pad them using keras, to bring them to the desired dimensions, depending on the model. Obviously, higher number of pixels as input to our CNN will automatically make its training more computationally expensive, so I tried to keep the code efficient for reproducibility, using also the tensorflow gpu as backend. Lastly, a random seed was introduced during the random data augmentation and training-validation splits, for reproducibility as well.

Models evaluation:

Name of model	Trained with dataset	Confusion Matrix	Classification report	Accuracy (test set)	Accuracy (validation)
RandNet (from scratch)	MWI 1.1 400	 True Predicted errors err % -- SUNNY -> RAINY 573 18.86 % SUNNY -> SNOWY 350 11.52 % SNOWY -> HAZE 202 6.65 % SNOWY -> RAINY 150 4.94 % SUNNY -> HAZE 115 3.79 % RAINY -> SNOWY 101 3.32 % RAINY -> HAZE 79 2.60 % SNOWY -> SUNNY 26 0.86 % RAINY -> SUNNY 9 0.30 %	 precision recall f1-score support HAZE 0.000 0.000 0.000 0 RAINY 0.315 0.637 0.421 521 SNOWY 0.698 0.734 0.716 1421 SUNNY 0.624 0.053 0.098 1096 micro avg 0.472 0.472 0.472 3038 macro avg 0.409 0.356 0.309 3038 weighted avg 0.606 0.472 0.442 3038	0.471692	0.775000 (1-fold)
RandNet2 (from scratch)	MWI 1.1 2000	 True Predicted errors err % -- SUNNY -> RAINY 620 20.41 % SUNNY -> SNOWY 263 8.66 % SNOWY -> RAINY 212 6.98 % SNOWY -> HAZE 138 4.54 % RAINY -> HAZE 112 3.69 % SNOWY -> SUNNY 104 3.42 % RAINY -> SNOWY 101 3.32 % SUNNY -> HAZE 50 1.65 % RAINY -> SUNNY 15 0.49 %	 precision recall f1-score support HAZE 0.000 0.000 0.000 0 RAINY 0.184 0.157 0.170 521 SNOWY 0.574 0.684 0.624 1421 SUNNY 0.363 0.297 0.326 1096 micro avg 0.454 0.454 0.454 3038 macro avg 0.280 0.284 0.280 3038 weighted avg 0.431 0.454 0.439 3038	0.453917	0.7091 (+/-0.1122) (5-folds)
TransferNet	MWI 1.1 400	 True Predicted errors err % -- SUNNY -> SNOWY 441 14.52 % SUNNY -> RAINY 317 10.43 % SNOWY -> RAINY 210 6.91 % SUNNY -> HAZE 177 5.83 % RAINY -> SNOWY 141 4.64 % RAINY -> HAZE 128 4.21 % SNOWY -> SUNNY 82 2.70 % SNOWY -> HAZE 64 2.11 %	 precision recall f1-score support HAZE 0.000 0.000 0.000 0 RAINY 0.152 0.282 0.198 521 SNOWY 0.690 0.559 0.618 1421 SUNNY 0.513 0.209 0.297 1096 micro avg 0.385 0.385 0.385 3038 macro avg 0.339 0.263 0.278 3038 weighted avg 0.534 0.385 0.430 3038	0.4338	0.7855 (+/- 0.0145) (5-folds)

TransferNet	MWI 1.1 2000	 True Predicted errors err % -- SUNNY -> RAINY 321 10.57 % SUNNY -> SNOWY 313 10.30 % SUNNY -> HAZE 154 5.07 % RAINY -> HAZE 148 4.87 % SNOWY -> RAINY 146 4.81 % SNOWY -> SUNNY 129 4.25 % RAINY -> SNOWY 116 3.82 % SNOWY -> HAZE 87 2.86 % RAINY -> SUNNY 2 0.07 %	 precision recall f1-score support HAZE 0.000 0.000 0.000 0 RAINY 0.367 0.509 0.426 521 SNOWY 0.674 0.706 0.690 1421 SUNNY 0.809 0.324 0.463 1096 micro avg 0.534 0.534 0.534 3038 macro avg 0.462 0.385 0.395 3038 weighted avg 0.670 0.534 0.563 3038	0.534233	Not calculated, 0.833 on training set
Linear SVM after feature extraction	MWI 1.1 2000	 True Predicted errors err % -- SNOWY -> HAZE 329 10.83 % SUNNY -> SNOWY 305 10.04 % SNOWY -> RAINY 304 10.01 % SNOWY -> SUNNY 274 9.02 % RAINY -> SNOWY 261 8.59 % SUNNY -> RAINY 197 6.48 % SUNNY -> HAZE 174 5.73 % RAINY -> SUNNY 119 3.92 % RAINY -> HAZE 93 3.06 %	 precision recall f1-score support 0 0.00 0.00 0.00 0 1 0.09 0.09 0.09 521 2 0.48 0.36 0.41 1421 3 0.52 0.38 0.44 1096 micro avg 0.32 0.32 0.32 3038 macro avg 0.27 0.21 0.24 3038 weighted avg 0.42 0.32 0.37 3038	0.3232	Not calculated, 0.989 on training set

Explaining design and implementation choices of models:

First of all, apart from the data augmentation, in order to prevent models from overfitting, we will introduce stopping callback during the training of all the models, so if the validation(or test in some models) set accuracy doesn't improve after 4 epochs, the training of the model is stopped at that point. Also, all the models will be trained at most for 20 epochs, for running time purposes. Furthermore, dropouts and batch normalization will be used again to prevent overfitting.

For the first RandNet model, I have chosen some small data augmentation and target size(118,224). Inisde the CNN, I have 3 convolutional layers with 16,32 and 64 2x2 filters respectively, with valid padding, each one connected with a 2 by 2 max polling layer. After flattening the last convolutional layer, I have put 2 dense layers, with 100 and 4(number of classes). The optimizer is rmsprop, with the default learning rate, and all the activation functions are relu, except of course the last one, which is softmax, so we get the probabilities of each image belonging to each of the 4 classes we have.

The second RandNet model is similar to the first one. I have increased the data augmentation and the target size(225,225) but also the kernel size of the filters in the second and third convolutional layer(4x4 and 5x5 respectively). Lastly, the optimizer is SGD with learning rate 1e-04 and momentum=0.9, so that the model converges faster. The idea behind this model was to pass to the filters a bigger window of pixels, so to check if the weather can be better predicted by passing to the filters a bigger region of the original images.

For the TransferNet model, the data augmentation and target size(350,450) were increased quite a lot. I also introduced the brightness range for the data augmentation of the pictures, after noticing that some images in the test set were taken at night, in contrary with the training set. After taking the weights from the VGG16 model, I have trained just the final convolutional layer, flatten it, and passed it to 2 dense layers, with 64 and 4 neurons respectively. The optimizer for this model is the SGD, with learning rate 1e-03 and momentum 0.9. In total, around 7 million parameters were trained.

For the Linear Svm feature extractor model, I passed the last pooling layer as input to a linear SVM, without training the parameters from the loaded VGG16 model. The advantage of this method is that it’s really fast, since we don’t train the CNN. I also experimented only with the linear version of SVM, without kernel trick, for computational complexity motives.

Explaining results and selection of the best model:

For the first RandNet model, the accuracy on the validation test is 70%, although its really worse at the test set(47%). By looking at the classification report, we can observe that we have a good f1 score for the SNOWY class, an average one for the RAIN and a really bad one for the SUNNY class. More concretely, this model has low precision on the RAIN class and a really small recall(5%)for the SUNNY one. This can also be confirmed by looking at the confusion matrix, where it becomes obvious that the model constantly misclassifies SUNNY as either RAINY or

SNOWY. This model is clearly biased to this specific test set and it's only marginally better than the baseline estimator(1421/3038=46% accuracy)

With the second RandNet model, the f1 score of the SNOWY class is increased, but the score on the other classes is decreased. This model's accuracy on the test set is worse than the baseline estimator. By performing a cross validation, we see that the average accuracy is 70%, but the standard deviation is more than 10%, so this model doesn't look stable. However, the big difference in the cross validation accuracy and the test accuracy confirms that the 2 datasets are highly diversified.

The TransferNet model, trained in 400 images, has similar results with the RandNet2 in terms of accuracy and classification report on the test set. However, this model is much more concrete on the cross validation, since it has accuracy 78.5% with standard deviation almost 1% . So this model has good predicting ability when it comes to images in the training set, but performs poorly on the test set, because of the differences between the images taken. We can claim that this model is better than the 2 RandNet models, although it performs worse at this specific test set, since it has better results on the cross-validation, something that shows us that the model performs worse on the test set because of the diversity of the images between test and training set and not because of overfitting.

The same TransferNet model, trained in the 2000 images dataset, improves vastly the results on the test set. The accuracy increased almost by 10%, compared to the TransferNet model of the 400 images, and the classification report shows that the f1 score increased in all the classes. More precisely, both the precision and the recall for the RAINY and SNOWY classes increased and the f1 score of SNOWY is almost 70%, something that means that the model actually is good at predicting the SNOWY instances of the test set, although not so good at the RAINY ones. The precision on the SUNNY class increased dramatically to 80%, but the recall is still relatively low, although it also increased. By looking at the confusion matrix, we see that the model has the same 'pattern' of errors, it has high error percentage when the actual class is SUNNY and it predicts either RAINY or SNOWY, but first the error rates are smaller when training with the 2000 images and second these patterns of errors exist in all of my models, so it’s more because of the nature of the images in the training and test set and not because of the models’ structure and weaknesses. It is clear that the best model is the TransferNet, when trained for the 2000 images dataset, since it performs better in both the training set but also to the test set, compared with all the other models. For running time purposes, this model wasn’t checked on cross validation. However, since it is the same model as before, which was really stable on the cross validation, just trained using more data, and since it has better results on the test set, we can be sure that it is not overfitting on the training set.

The Linear SVM method after feature extraction performs greatly on the training set, with accuracy of almost 99%, but it is clearly overfitting, since it performs terribly on the test set, much worse than the baseline estimator. By checking the classification report, the model is very bad at predicting the RAINY instances and practically ignores this class, that’s why it has such a small value of recall, and, apart from the usual misclassification ‘patterns’ that we saw in the previous models, it also misclassifies many SUNNY instances as HAZE, something that both intuitively but also after comparing with the results of other models, makes us understand that this model is the worst in terms of predicting ability. That’s why it wasn’t checked using cross validation.

Comments:

It is not surprising that the best model was the TransferNet one, when trained with 2000 images. Generally, it is hard to train from scratch a CNN with limited computing resources. That is why it is common to use very deep pre-trained models, such as VGG16, which is trained with the imagenet dataset, compiling of over 15 million images and 22.000 different categories, and train just their last layer(s). Also, it is well known that CNNs require a big amount of training images, depending obviously on the application.

We saw that the TransferNet model improved when we trained it using 2000 images rather than 400. This means that perhaps if we increase further the training test size, it may perform even better, both in the training set and the test set. However, for running time purposes I didn’t train any of the models using more than 2000 images. Also, no matter how much we train the model, it will always have differences in the validation and the test set, because of the different angles and time that the pictures in the training set and test set were taken. What we could try to do is to combine the training and test set, train the model and then validate it. In that way, the model will have more predicting ability for images taken from a CCTV camera and under different circumstances.