COMPUTER VISION (2023)

REPORT: **Task 2**

Group members:

Gerard Castro (1), gcastrca26@alumnes.ub.edu, gerardc98

Àlex Pujol (2), apujolvi43@alumnes.ub.edu, alex7

1. SUMMARY OF CONTRIBUTIONS

- In the previous task, in order to address the problem of bias mitigation we studied the distribution of the sample images for each of the categories. We took advantage of that data exploration to define a proper sample weights method.
- Then experimented with an original custom sample weights by modifying the weights reasonably, and by modifying the underlying loss function. Also, a new bias based loss function was introduced.
- Regarding the backbone of the model, the experiments were done over a ResNet50, a VGG and an EfficientNet, all available in keras.

1.2. Custom loss strategy (in the case of task 2 or 3)

There are mainly three different function loss strategies for which we did some experiments.

- The first approach, called custom baseline, uses the same definition as baseline sample weights but considering every compound category. Hence, we applied the formula

$$w_j = \frac{n_{samples}}{n_{classes} n_{samples,j}}$$

where w_j is the weight for each compound group j, $n_{samples}$ is the number of samples in the train set, $n_{comp-group}$ is the number of compound groups (78 in our case) and $n_{samples,j}$ is the number of samples of the group j.

Observe that this definition of sample weights is prone to outliers when $n_{comp-group}$ is high and $n_{samples,j}$ irregularly distributed. We proposed a redefinition of the sample weights to solve this issue. The introduce the **log-normalized sample weights 1 (lnsw1)**

$$\bar{w}_j = \log\left(\frac{n_{samples}}{n_{comp-groups} n_{samples,j}} + 1\right)$$

$$w_j = \bar{w}_j \,/\, N_{norm}$$

where $N_{norm} = \frac{1}{n_{samples}} \sum_{j\geq 1}^{N} \bar{w}_j$ is a normalization term. The number of outliers in the sample weights was satisfactorily reduced by log-normalization.

The weight for outliers was even more reduced by means of the **log-normalized sample weights 2 (lnsw2)**:

$$\bar{w}_j = \frac{1}{N_{norm}} \frac{\log(n_{samples} + 1)}{n_{comp-groups} \log(n_{samples,j} + 1)}$$

- Then we tried another approach by building a loss function bases on the **bias scores**. The loss function was defined by aggregating the squares of the bias metrics.

$$L = (1 + B_g)^2 + (1 + B_e)^2 + (1 + B_f)^2 + (1 + B_a)^2.$$

We added 1 to the bias scores to avoid decreasing the weight of the bias when are less than one.

- Finally, we tried several loss functions that are already implemented in `tf.keras.losses` and checked how they performed in relation to the baseline loss function that is

`MeanSquaredError()`. These loss functions also admit the use of weighted samples to modify their behavior. In detail, we chose `Huber()` and `MeanAbsolutePercentageError()`. MAPE an extension of MAE but based on a percentage and is easy to interpret. Huber lies in between MSE and MAE in the sense that MSE strongly punish outliers (is the square of the errors) and MAE do not consider outliers that much (is the average of the errors). Table 3 models used MSE and MAPE losses. Other runs are not illustrated due to the lack of space and since we considered that are no illustrative.

1.3. Training strategy

At the beginning the training strategy remained the same as in the baseline model, that is models are trained in two stages. First just the extra fully connected layers and the rest of the model. However, for the last experiments we decided to include a third stage where the whole model is trained using a different loss function than in previous stages.

The discussion for the model selection was based in a trade-off between the bias scores and the MAE metric. We tried to reduce MAE while mitigating the biases.

For the experiments we tested over a pretrained ResNet, a VGG model and an EfficientNet. We chose the first two models since they were the ones used on the previous task, to facilitate the comparison. The last model is in fact the pretrained EfficientNetB0 with the imagenet weights. We selected it because it is a computationally efficient model, widely used for image recognition problems and easy to access from keras. Besides of the backbone and the added fully connected layers, we removed the last layer of each model and embedded dense layers of sizes 512, 256, 128 and 32 nodes, and two dropout layer before and after the layer 512. Other hyperparameters in discussion for the above models were the number of epochs and the learning rate. The optimizer remained the same as in starting-kit.

2. EXPERIMENTS AND RESULTS

Experiments were made following the above-mentioned custom loss and training loss strategies. Table 1 encapsulates the first experiment. Some executions were made to study the impact of the different sample weights already defined to see a comparison between them. We intended to see if it is useful to give a different sample weight to each compound group and how much does the outliers on sample weights impact the final result. The loss function was `MeanSquaredError()` for every model. Notice that the last column is the aggregated bias score defined before.

Table 1: Comparing the results for different sample weight.

Model	***Weights***	***Comp. Groups***	***Epoch***	***LR***	***Bg***	***Bf***	***Be***	***Ba***	***MAE***	***L***
ResNet50	*No*	*No*	*12*	*1e-5*	0.168	0.150	0.926	7.602	10.472	80.390
ResNet50	*S-kit*	*No*	*12*	*1e-5*	0.540	0.279	0.738	3.804	11.076	30.106
ResNet50	*S-kit*	*Yes*	*12*	*1e-5*	0.089	0.774	0.828	4.284	12.281	35.595
ResNet50	*Inw1*	*Yes*	*60*	*5e-6*	0.295	0.448	0.727	5.504	7.330	49.058
ResNet50	*Inw2*	*Yes*	*60*	*5e-6*	0.615	1.225	0.230	3.975	7.322	33.822

We can observe that our custom sample weights did not perform as well as expected, the aggregated score was not able to surpass the baseline model. Previous runs suggested increasing the number of training epochs for the last models. However, even in this situation we were not able to fix all the biases as a whole, but only the gender bias and the ethnicity bias in different experiments.

The next experiment (Table 2) consisted of doing similar runs as before but changing the backbone to VGG19 and EfficientNet, the goal was to see if there was any improvement by changing the model and using the sample weights approach.

Table 2: Comparing the results for different backbones.

Model	***Weights***	***Comp. Groups***	***Epoch***	***LR***	***Bg***	***Bf***	***Be***	***Ba***	***MAE***	***L***
VGG19	*S-kit*	*No*	*60*	*5e-6*	*0.455*	*0.547*	*0.499*	*2.308*	*9.470*	17.700
VGG19	*Inw2*	*Yes*	*60*	*5e-6*	*0.071*	*0.501*	*0.866*	*7.982*	*8.252*	87.633
EfficientNet	*S-kit*	*No*	*60*	*5e-6*	1.340	1.067	0.455	11.752	10.612	174.44
EfficientNet	*Inw2*	*Yes*	*60*	*5e-6*	1.053	1.326	1.227	11.844	11.463	179.55

We can see that our custom sample weight using MSE as loss function was not successful for neither of the models. After these results, we considered adding a third training stage that has the same hyperparameters as the second stage but changes the loss function to MAPE (other runs with MAE and Huber loss were done but achieving similar results). This experiment, in Table 3, was made to see if there is any good by training a same model with two different loss functions. Sample weights chosen were Inw2, hoping to improve the baseline following our custom approach.

Table 3: Adding a third stage.

Model	***Weights***	***Comp. Groups***	***Epoch***	***LR***	***Bg***	***Bf***	***Be***	***Ba***	***MAE***	***L***
ResNet50	*S-kit*	*No*	*12*	*1e-5*	0.540	0.279	0.738	3.804	11.076	30.106
ResNet50	*Inw2*	*Yes*	*80*	*5e-6*	*0.259*	*0.536*	*0.758*	*8.241*	*7.388*	92.431
VGG19	*Inw2*	*Yes*	*80*	*5e-6*	*0.076*	*0.247*	*0.572*	*6.996*	*6.595*	69.119
EfficientNet	*Inw2*	*Yes*	*80*	*5e-6*	0.321	0.214	0.637	4.655	5.744	37.877

We noticed an increase in the performance regarding the bias mitigation for some categories. Also, we can see an improvement on the MAE. However, we are still far away to a proper bias mitigation and more experiments should be conducted. Finally, we ran a last model using the ***L*** bias scores loss function but found several technical problems that we could not solve. In the attached Notebook one can find our attempt to convert the bias scores into a tf.loss function. The main difficulties found are that every operation and all loops need to be in tf format to construct the computational graph.

3. FINAL REMARKS

In this report we tried mainly to mitigate biases by defining a custom sample weight function loss. After reviewing the results we claim that this approach was not enough for bias mitigation. We can see in the tables that the models for which a lower aggregated bias was achieved, did not use the log-normalized sample weights and neither the compound groups. However, VGG19 together with Inw2 achieved an amusingly low gender bias in experiments 2 and 3. We also proposed using the aggregated bias score as a loss function.