SUPPLEMENTARY MATERIAL FOR "RE-BENCHMARKING OUT-OF-DISTRIBUTION DETECTION IN DEEP NEURAL NETWORKS"

Anonymous authors
Paper under double-blind review

A APPENDIX

In this section, we provide another method of improving OOD detection models. Existing method is to split the ID dataset into training set and validation set. Training set is used to train models while validation set is used to select the model. After the model is established, each image can get a score according to OOD detection algorithms. Hence, we recommend models trained by the data with higher scores and utilize the data with lower scores to form validation set. Models can then get better features from training data. Here is the introduction of our method.

Adaptive Model Selection

We divide datasets into different groups and make ID dataset as well as OOD dataset using methods from our benchmark. Then we divide this dataset into training set and validation set. After we train the model by training set, we hope to select the data which has the lowest algorithm scores as the new validation set. Thus, we calculate the algorithm scores of different batches of data and then sort them. The lowest 20% batches are picked as the validation set while the other batches are used as the training set. The models are then trained again by utilizing these new sets. These new models are used to re-detect OOD data. In order to improve the effectiveness of models, this method can be repeated more than one time. After several times of iteration, the models are fully improved.

In this paper, we only suggest this method. We will make a total experiment and all-round analysis in the future work.

Algorithm 1 Divide training set and validation set manually based on grad_norm score

Selecting datasets to form ID dataset;
Training model with training set and validation set divided from ID dataset;
Initializing an empty dictionary S;
for each batch i in ID dataset **do**
    Calculating the algorithm score S_i;
    $S[S_i]$=i;
end for;
Sorting S according to the algorithm scores;
Initializing an empty list val_set;
for $i = 0; i < 0.2 * num(IDdataset); i++$ **do**
    Adding $S[i][1]$ into val_set;
end for
for each batch i in ID dataset **do**
    if i in val_set **then**
        Validating model
    else
        Training model
    end if
end for
Testing the performance of the model