Tal Kadosh – 319126546
Yonatan Pisman - 316262997

Final Course Project 2021

Enhancing The Reliability of Out-of-Distribution Image Detection In Neural Networks

Stage 1: Selecting the algorithm for evaluation

The algorithm we chose, ODIN (**O**ut of **DI**stribution detector for **N**eural networks) is dealing with the problem of out of distribution samples that appear in real world applications because of very little control over testing data distribution. There is significantly high importance for classifiers to be aware of uncertainty when shows out of distribution inputs. The algorithm is based on observations shown that neural network tends to assign higher softmax scores to in-distribution inputs rather than out-of-distribution, so in order to enlarge the gap in the softmax scores between in and out distribution the algorithm is using temperature scaling in the softmax function and adding small perturbations to inputs.

Except of the results of the algorithm that significantly shows improvements in some metrices like FPR and Detection error, the algorithm has many more advantages:
- The main advantage of the algorithm is its flexibility, it can be applied on different pre-trained neural networks with different architectures.
- Furthermore, this algorithm is a very efficient solution for the described problem. Instead of enlarging the training set of the in and out distribution examples or ensure that the neural network accurately detecting out of distribution, which both solutions are very expensive in terms of time, complexity and other resources. The algorithm solves the problem without re-training the neural network.

- One major disadvantage of the proposed algorithm is the lack of explanation regarding its decision making.
- Another drawback of the algorithm is that it can be applied on a specific classifier model (neural network), and it cannot be applied on simple classification tasks that use other classification models like decision trees and boosting algorithms.
- In addition, because of the use of characteristics of the softmax function, this algorithm is mostly relevant to classification problems rather than regression problem when linear functions work better.

Stage 2: Suggesting an improvement

In order to enlarge the gap between in and out of distribution samples we need to use a regularization technique that will address the problem of the overconfidence of the model predictions. Our suggestion is to use <u>label smoothing</u> in order to panelize the model about its overconfidence.

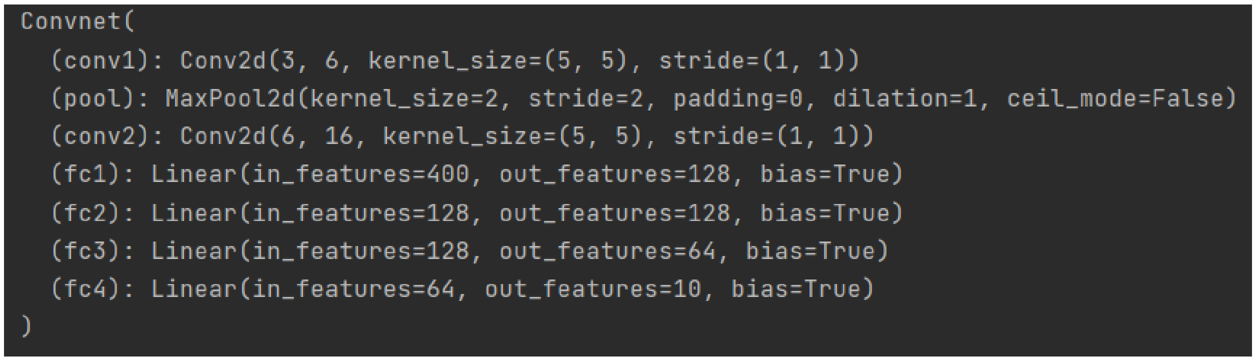
In a real-world scenario, state-of-the-art neural networks architecture that trained on many samples, tend to generalize the problem and even be useful in solving other tasks. But in our case, we should punish the model at uncertainty predictions.
We noticed that the algorithm presented in the article is using cross entropy loss function, which using one-hot encoded labels that encourages largest possible logit gaps be fed into the softmax function, in other words, it will make the model less adaptive and too confident about its predictions. So, in order to deal with this problem, we suggest using smoother labels that

encourages small logit gaps and it will prevent overconfident predictions as mentioned in this article that describes the benefits of label smoothing.

Stage 3: Select a well-known algorithm for comparison.

The baseline algorithm we will compare against our algorithm will be a neural network with the same architecture that trained on the same training set, but without the additions proposed in the article. This way, we can compare the gap in softmax score between a model without the proposed additions to the model with the proposed additions (temperature scaling and the small perturbations).

The article shows to softmax score gap with state-of-the-art neural networks architecture like densenet. In our project we implemented a simple version of convolutional neural network so we can handle the processing power required to train the model on 20 datasets.
Our model architecture:



```
Convnet(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=128, bias=True)
  (fc3): Linear(in_features=128, out_features=64, bias=True)
  (fc4): Linear(in_features=64, out_features=10, bias=True)
)
```

For case of 10 classes classification, the model will contain two layers of convolution and four fully connected layers with Relu activation function.

Stage 4: Evaluating the algorithms you selected in stages 1, 2 and 3

Hyperparameter Optimization

The algorithm in the article shows that by adding temperature scaling in softmax function and small perturbations to the input the gap in softmax score is increasing the gap between in\out of distribution images. These features are not related to any neural network, i.e., they are not hyperparameters of models, and they are not taking part in the training of the models.

The features mentioned in the article are applied on pre-trained models and they are not learned from the internal train in the cross-validation, so in our project, we will not implement the internal CV. Instead, in the internal CV phase, we will train each model on 9 out of the 10 folds, and the other fold will be used for evaluation of softmax score for in-distribution samples, and the Imagenet dataset will be used for evaluation of softmax score for out-of-distribution samples. This evaluation will be made by taking 50 different combinations of temperatures (for the temperature scaling) and magnitudes (for the input perturbations). As a result, the algorithm will return the best combination, the combination with the highest accuracy. This way, we will find the temperature and magnitude that generates the largest gap in softmax score between in\out of distribution images for a given model.

Tal Kadosh – 319126546
Yonatan Pisman - 316262997

Notice, we interpret the accuracy of the model to be the rate of the algorithm classified correctly as out-of-distribution when TPR is 95%, rather than the prediction of the model for the given input.
The Out of distribution works according to this function:

$$g(x,T,\varepsilon,\delta)=\begin{cases}1 & if\ softmax(\tilde{x},T)\geq\delta\\0 & if\ softmax(\tilde{x},T)<\delta\end{cases}$$

for input x, $\tilde{x}$ will be the input after the small perturbation with ε magnitude. If the softmax score with temperature scaling T is greater or equal to a threshold δ it will be classified to in-distribution image, and otherwise it be classified to out-of-distribution image.
The parameters $T, \varepsilon, and\ \delta$ are chosen so that the TPR (the fraction of in-distribution images correctly classified as in-distribution images) is 95%.

<u>explanation for the other metrics for evaluation:</u>
1. TPR- will be interpreted as the probability that a positive (in-distribution) is classified as positive. This metrics will be a constant of 95% that according to him the threshold δ is chosen.
2. FPR- will be interpreted as the probability that a negative (out-of-distribution) is misclassified as positive, at 95% TPR.
3. Precision- will be calculated as $\frac{TP}{TP+FP}$.
4. AUC- will be interpreted as the probability that a positive example is assigned a higher detection score than negative example. This metric is threshold independent.
Area under the precision-recall- another threshold independent metrics. Describe the ratio between precision and recall.
5. Inference time- the time takes to the model to infer in or out of distribution for 1000 instances.

Tal Kadosh – 319126546
Yonatan Pisman - 316262997

Performance metrics for evaluation

The CSVs with the statistical results can be found in file “results”.
Each CSV contains the statistical results for the three models for certain dataset.

Datasets details:

1. All CIFAR excels are partitions of CIFAR100 dataset, each contains 10 classes. The dataset was loaded from torchvision.dataset module.

CIFAR0- apple, aquarium fish, baby, bear, beaver, bed, bee, beetle, bicycle, bottle.

CIFAR1- bowl, boy, bridge, bus, butterfly, camel, can, castle, caterpillar, cattle.

CIFAR2- chair, chimpanzee, clock, cloud, cockroach, couch, crab, crocodile, cup, dinosaur.

CIFAR3- dolphin, elephant, flatfish, forest, fox, girl, hamster, house, kangaroo, keyboard.

CIFAR4- lamp, lawn mower, leopard, lion, lizard, lobster, man, maple tree, motorcycle, mountain.

CIFAR5- mouse, mushroom, oak tree, orange, orchid, otter, palm tree, pear, pickup truck, pine tree.

CIFAR6- plain, plate, poppy, porcupine, possum, rabbit, raccoon, ray, road, rocket.

CIFAR7- rose, sea, seal, shark, shrew, skunk, skyscraper, snail, snake, spider.

CIFAR8- squirrel, streetcar, sunflower, sweet pepper, table, tank, telephone, television, tiger, tractor.

CIFAR9- train, trout, tulip, turtle, wardrobe, whale, willow tree, wolf, woman, worm.

2. All BIRD excels are partitions of BIRDS dataset, each contains 10 classes. The dataset was loaded from- https://www.kaggle.com/gpiosenka/100-bird-species.

Bird1 - african crowned crane, african firefinch, albatross, alexandrine parakeet, american avocet, american bittern, american coot, american goldfinch, american kestrel, american pipit.

Bird2 - american redstart, anhinga, annas hummingbird, antbird, araripe manakin, asian crested ibis, bald eagle, bali starling, baltimore oriole, bananaquit.

Bird3 - banded broadbill, bar-tailed godwit, barn owl, barn swallow, barred puffbird, bay-breasted warbler, bearded barbet, bearded reedling, belted kingfisher, bird of paradise.

Bird5 – black-throated sparrow, blackburniam warbler, blue grouse, blue heron, bobolink, bornean bristlehead, bornean leafbird, brown noody, brown thrasher, bulwers pheasant.

Bird6 - cactus wren, california condor, california gull, california quail, canary, cape may warbler, capuchinbird, carmine bee-eater, caspian tern, cassowary.

Bird7 – cedar waxwing, chara de collar, chipping sparrow, chukar partridge, cinnamon teal, clarks nutcracker, cock of the rock, cockatoo, common firecrest, common grackle.

Tal Kadosh – 319126546
Yonatan Pisman - 316262997

Bird8 - common house martin, common loon, common poorwill, common starling, couchs kingbird, crested auklet, crested caracara, crested nuthatch, crow, crowned pigeon.

Bird9 - cuban tody, curl crested aracuri, d-arnauds barbet, dark eyed junco, double barred finch, downy woodpecker, eastern bluebird, eastern meadowlark, eastern rosella, eastern towee.

Bird10 – elegant trogon, elliots pheasant, emperor penguin, emu, enggano myna, eurasian golden oriole, eurasian magpie, evening grosbeak, fire tailed myzornis, flame tanager.

Tal Kadosh – 319126546
Yonatan Pisman - 316262997

Stage 5: Statistical significance testing of the results

For the statistical significance testing we chose the AUC metric which represents

the probability that a positive example is assigned a higher detection score than negative example.

For each algorithm we collected the average AUC of the 10 cross validation iterations per dataset and have received the following data:

DATASET	ALG1	ALG2	ALG3
Bird1	60.72815	59.59564	50.2202
Bird2	59.42556	57.15567	49.64532
Bird3	49.90963	49.69029	49.99458
Bird4	53.12256	52.44647	50.13516
Bird5	52.8809	52.49998	50.07667
Bird6	62.23932	63.03675	49.85736
Bird7	59.63721	60.0382	49.91259
Bird8	52.18878	53.01564	49.80385
Bird9	60.49432	58.40506	50.03478
Bird10	62.83234	65.55171	49.84682
CIFAR0	62.60569	64.4236	49.85601
CIFAR1	69.99894	73.21573	50.12819
CIFAR2	74.22191	76.85823	49.93572
CIFAR3	63.83102	64.79443	49.97509
CIFAR4	67.92021	69.15879	49.86727
CIFAR5	73.58194	78.60249	50.18544
CIFAR6	71.19091	73.05166	50.05134
CIFAR7	66.40504	67.56942	50.28704
CIFAR8	70.33314	73.04207	50.08649
CIFAR9	72.2122	74.49701	49.9556

After creating the table above, we used the fridmanchisquare function provided by the scipy module by providing it the lists of average AUCs of every algorithm on all the datasets. (The code is in code folder in fridmanNemeyni.py)

Let H0 (our null hypothesis) = There is no difference between the three algorithms.

Let H1 (Alternative hypothesis) = There is a difference between the three algorithms.

Tal Kadosh – 319126546
Yonatan Pisman - 316262997

let’s choose our alpha to be 0.05.

After conducting the fridman test we received the following result:

statistic=25.900000000000034, p-value=2.3762189738479023e-06

As we can see, our p-value is significantly lower than alpha thus we are going to reject the null hypothesis and perform a post hoc test to see where lies the difference between the algorithms.

We have chosen the nemenyi test as our post hoc test and have conducted it using the function posthoc_nemenyi_fridman provided by the scikit_posthocs module it the lists of average AUCs of every algorithm on all the datasets from the table that’s shown above.
(The code is in code folder in fridmanNemeyni.py)

The results were the following:

1. P-value of the algorithm from stage 1 vs algorithm from stage 2: **<u>0.41689</u>**
2. P-value of the algorithm from stage 1 vs algorithm from stage 3: **<u>0.001</u>**
3. P-value of the algorithm from stage 2 vs algorithm from stage 3: **<u>0.001</u>**

Since we chose our alpha to be 0.05 we can infer from the results that the statistically significant differences lie between the algorithm from stage 3 compared to the algorithms from stage 1 and stage 2 due to the p-value being lower than alpha in both cases (lines 2 and 3 in the results) and that the p-value of stage 1 algorithm vs stage 2 algorithm is higher than our alpha which means that they’re most likely **not** statistically significantly different.

We can conclude that the improvement suggested at stage 2 for the algorithm from the article that although it did cause the model to be less overconfident its impact was not statistically different compared to the algorithm from the article. Furthermore, regarding the significant difference between the well-known algorithm from stage 3 compared to the other two algorithms, this significant difference occurs due to the algorithm from the article(stage 1) and the algorithm from the article combined with the improvement from stage 2 cause the model to be less overconfident and as a result to enlarge the gap of the softmax score and provide more accurate classification of in and out of distribution inputs while the algorithm from stage 3 doesn’t pay attention to the overconfidence of the model prediction.

Tal Kadosh – 319126546
Yonatan Pisman - 316262997

Stage 6: Reporting your conclusions

Pseudo code stage 1:

$* \, The\ model\ was\ trained\ with\ cross\ entropy\ loss$

$n = 10 - Number\ of\ classes\ that\ we\ classify$

$f = (f_1, \dots, f_n) - our\ neural\ network$

$i = index\ between\ 1\ and\ n$

$x - input\ to\ neural\ network$

$\tilde{x} - pre\ proccesed\ input$

$T - The\ temperature\ scaling$

$\epsilon - The\ small\ pertrubation\ magnitude$

$S_i(x,T) - Softmax\ score\ for\ class\ i\ with\ input\ x\ and\ temperature\ scaling\ parameter\ T$

```
1. $expSum \leftarrow 0$
2. $For\ i = 1\ to\ n$
3. $For\ j = 1\ to\ n$
4. $expSum \leftarrow expSum + e^{f_j(x)/T}$
5. $S_i(x,T) \leftarrow \frac{e^{f_i(x)/T}}{expSum}$
6. $S_{\hat{y}}(x,T) \leftarrow max_i\ S_i(x,T)$
7. $\tilde{x} = x - \epsilon * sign(-\nabla_x * logS_{\hat{y}}(x,T))$  *The sign function returns 1 if input is positive or -1 if negative and 0 if zero.
8. $return\ S(\tilde{x},T)$
```

- In the following pseudo-code we calculate the softmax score for each class depending on the input and the temperature scaling parameter (lines 1-5) we then select the best softmax score in line 6. We then take the original input to the neural network and decrease the small perturbation and calculate $S(\tilde{x},T)$ which is the calibrated softmax score.

Tal Kadosh – 319126546
Yonatan Pisman - 316262997

Pseudo code stage 2:

$* \, The\ model\ was\ trained\ with\ label\ smoothing\ rather\ than\ cross\ entropy\ loss$

$n = 10 - Number\ of\ classes\ that\ we\ classify$

$f = (f_1, \dots, f_n) - our\ neural\ network$

$i = index\ between\ 1\ and\ n$

$x - input\ to\ neural\ network$

$\tilde{x} - pre\ proccesed\ input$

$T - The\ temperature\ scaling$

$\epsilon - The\ small\ pertrubation\ magnitude$

$S_i(x, T) - Softmax\ score\ for\ class\ i\ with\ input\ x\ and\ temperature\ scaling\ parameter\ T$

```
1. $expSum \leftarrow 0$
2. $For\ i = 1\ to\ n$
3. $For\ j = 1\ to\ n$
4. $expSum \leftarrow expSum + e^{f_j(x)/T}$
5. $S_i(x,T) \leftarrow \frac{e^{f_i(x)/T}}{expSum}$
6. $S_{\hat{y}}(x,T) \leftarrow max_i\ S_i(x,T)$
7. $\tilde{x} = x - \epsilon * sign(-\nabla_x * logS_{\hat{y}}(x,T))$  *The sign function returns 1 if input is positive or -1 if negative and 0 if zero.
8. $return\ S(\tilde{x},T)$
```

- The following pseudo code describes a way to make the model less overconfident by using label smoothing instead of cross entropy loss, other than that the algorithm performs the same steps as stated in stage1.

Tal Kadosh – 319126546
Yonatan Pisman - 316262997

Pseudo code stage 3:

$* \textit{The model was trained with cross entropy loss}$

$n = 10 - \textit{Number of classes that we classify}$

$f = (f_1, \ldots, f_n) - \textit{our neural network}$

$i = \textit{index between } 1 \textit{ and } n$

$x - \textit{input to neural network}$

$S_i(x) - \textit{Softmax score for class } i$

```
1. $expSum \leftarrow 0$
2. $For\ i = 1\ to\ n$
3. $For\ j = 1\ to\ n$
4. $expSum \leftarrow expSum + e^{f_j(x)}$
5. $S_i(x) = \frac{e^{f_i(x)}}{expSum}$
```

In the following pseudo-code, we perform stage 3 which is the calculation of the softmax score of each class but without the temperature scaling in the article and without a small perturbation.

In conclusion, we saw in the article a way to deal with detection of out-of-distribution images by rely on the over confidentiality of the models. The article introduced two ways to deal with the over confidentiality- temperature scaling and small perturbation to inputs, and from the statistical analysis we showed that these changes cause significantly less overconfidence.

Except from the suggested changes, we modified the loss function in the code (cross entropy loss) to be label smoothing loss in order to deal with the overconfidence of the models. Unfortunately, our suggestion did not show statistically significant change compared to the algorithm described in the article.