Adversarial Attacks, DeepFake Detection and Audio Spoofing

Dependable AI Assignment-1

Kwanit Gupta, B19EE046[a]

[a]*Indian Institute of Technology, Jodhpur*

March, 2023

Note

The 3rd Question's Solution Parts were left because of time constraints and other course and project commitments. I recorded 1000 English Sentences and 160 Hindi Sentences (the platform seemed to not work well after 160). In the report itself, I attached the link with respective files and folders. For Question-1 and Question-2, extensive analysis was done for the Overall Accuracy and attacks were performed during inference, i.e Evasion Mode.

---

1. Question-1

- Using CIFAR10 dataset, implement FGSM on your own and use a basic CNN architecture with 3 convolution layers to show the impact of the attack.
- Use 2 deep-learning architectures and any two attacks out of which one has to be either Mask based attack or PGD attack. Using the SVHN dataset, you need to show the impact of the attacks and compare and contrast them.
- Implement a deep learning architecture to detect adversarial attacks. The detection accuracy should be more than 70% of your attack (more the better).

1.1. Part-1 (FGSM and CNN Architecture)

FGSM utilizes the flawed assumption of a neural network possessing the high-dimensional linearity (i.e sum(w_i,x_i)). Due to this, it generates adversarial perturbations by noticing the suitable direction of maximum gradient change for the model and being additive, it can easily be incorporated into the training input.

$$\delta = \epsilon sign(\nabla_x J_\theta(\theta, x, y)) \quad (1)$$

Here, δ represents the perturbation, sign() as the signature function (+ or -), J_θ as loss function associated with model parameters θ, $\in$ as the bound set for perturbations, ∇_x as the gradient function, x and y as input image and output label respectively.

The three layered CNN constituted 3 convolutional layers with input channels equalling to 3 and then hidden layers incrementing feature channels by 1 and decrementing kernel size by 1, thereby outputting 6 feature channels. And then flattening and utilizing 2 hidden dense layers to obtain output probabilities.

Epsilon	*OverallAccuracy*
0.000	37.14%
0.001	23.48%
0.002	13.42%
0.004	3.67%
0.008	0.36%
0.01	0.13%

Table 1: 3-Layer CNN with FGSM Attack on CIFAR10

From the given Table-1, the overall accuracies went down, but in non-uniform manner, especially when epsilon is from 0.004 to 0.01 (3.67% to 0.13%, almost 30 times lower).

1.2. Part-2 (ResNet34 and DenseNet121)

In many of the research papers related to these architectures, they indeed have the role of conveying original feature information to further layers (consecutively for ResNets and layer-by-layer for DenseNets). Since the adversarial attack impacted the original feature maps too (at the 1st layer), the effects of adversary will eventually pass on to future layers, due to which these networks are highly vulnerable to attacks.

1.2.1. PGD

The intuition behind PGD is that by constraining the perturbation to lie within a small Lp ball around the original input, the algorithm ensures that the perturbed example remains perceptually similar to the original input, while still being adversarial (i.e., misclassified by the model). By iteratively updating the perturbation in the direction that increases the loss, the algorithm finds the minimal perturbation needed to cause misclassification.

1.2.2. Jitter

The intuition behind Jitter is that by adding small random noise to the input example, the attacker can cause the machine learning model to make a wrong prediction. The key challenge in this attack is to choose the noise vector in such a way that it is small enough to be imperceptible to a human observer, but large enough to cause misclassification by the machine learning model.

Epsilon	*Acc_ResNet*	*Acc_DenseNet*
0.000	32.24%	78.78%
0.001	28.54%	72.78%
0.002	25.36%	66.07%
0.004	20.11%	53.14%
0.008	12.87%	32.98%
0.01	10.43%	26.08%

Table 2: ResNet and DenseNet with FGSM Attack on CIFAR10

The huge drop for both of the model variants was observed when the epsilon value in FGSM Attack was changed from 0.004 to 0.008 (8% in ResNet and 21% in DenseNet).

Attack_Mode	*ResNet*	*DenseNet*
No Attack	93.46%	93.68%
FGSM	90.58%	90.55%
PGD	62.2%	64.1%
Jitter	62.14%	64.01%

Table 3: ResNet and DenseNet with/without Attack on SVHN

For SVHN dataset, FGSM didn't affect the model prediction that much, compared to PGD or Jitter (which make the accuracy lower down by 2/3rd).

1.3. Part-3 (Detection Pipeline)

The Detection pipeline is same as that of the classification one, just with the difference of output layer, i.e now 2 classes are intact (clean or attacked) instead of classification classes. All the detection experimentations happened with the notion of class balance only with 5 epochs. and considering base model as ResNet34, since DenseNet121 took way too long time to train and inference (that too on SVHN dataset).

Attack	*Dataset*	*Clean_Acc*	*Attack_Acc*
FGSM	SVHN	71.7%	0%
FGSM	CIFAR10	92.4%	7.62%
PGD	SVHN	0.18%	100%
PGD	CIFAR10	3.36%	96.64%
Jitter	SVHN	73.54%	0%
Jitter	CIFAR10	17.11%	82.89%

Table 4: ResNet-based Detector with Cross Entropy Loss

Observing the above table, it is pretty much evident that cross entropy loss fails to address the priorities of clean and attacked images simultaneously. It either lays total emphasis on the clean images or totally on the attacked images.

2. Question-2

Create 100 deepfakes/faceswap from the existing tools of your choice using your own face images. Split this data into fine-tune and test sets (50-50). Finetune your model in Q1(iii) and test on the remaining 50 test samples and report the performance.

2.1. FaceSweep

I utilized my 25 captured pictures and augmented with the horizontal flip to make those 50. In order to mitigate the problem of class imbalance and also counter the constraints of 100 images, I picked 50 different personalities for facesweep images and then mapped with the given FaceSweep Tool (Linked here).

With the saved weights of Q1(iii), I finetuned on the 50-50 train-test split for 10 epochs and observed the following table:-

Attack	*Dataset*	*FaceSweep_Acc*
PGD	SVHN	52%
PGD	CIFAR10	50%
Jitter	SVHN	48%
Jitter	CIFAR10	48%

Table 5: ResNet-based FaceSweep Detector with Cross Entropy Loss

Depending on the nature and inductive biases of SVHN and CIFAR10, the attack's detector models differ in training. PGD performed better in SVHN than in CIFAR10. Whereas it was almost similar behaviour for Jitter.

3. Question-3

- Record 1000 hindi and 1000 english sentences in your voice sampled from the given text files. Now use any TTS system with an Indian speaker and generate audios using the text that you have recorded. (Recorded audio will be considered as real and generated audio will be considered as fake). Please mention the TTS system that you have used.

- Use an ASSIST architecture and train the architecture on the recorded and generated audios for spoof detection i.e real vs fake classification.

- Use pre-trained (trained on ASVSpoof2019 LA) AASIST and finetune it on your dataset for spoof classification.

- Compare the performance for above two experiments using accuracy and EER metric.

- Implement the Decision Error Trade-off (DET) curve from scratch and plot it for above two experiments.

- Share the recorded and generated audios within separate folders respectively, using the drive link. Also provide the sentences that you have used for recordings in an excel file. Combine all these folders and an excel file into one folder with a naming convention. Format of excel file content: Audio number, sentence

3.1. Hardware and Platform Setup

I utilized gTTS (Google's Text to Speech) for generating audio of 1000 english and hindi sentences. For recording the audios (1000 english and 160 hindi), I used MSI's GL63 8SD-632IN Laptop, SANTLR and in the code notebook, I managed to convert JSON to CSV and accordingly label the columns as mentioned in the constraints.