PROJECT REPORT ON IMPLEMENTED CODE

Detecting CNN-Generated Images

Team Tensor

ABSTRACT:

In this research project, we train and test a universal detector to differentiate real and fake images which are generated by CNN models. The model is trained on images generated by a single CNN model (StyleGAN) with careful preprocessing, post-processing, and data augmentation process. And tested on the ForenSynths dataset which consists of fake images generated by 10 different CNN models which include ProGAN, Big-GAN, CycleGAN, StarGAN, GauGAN, DeepFakes, CRN, IMLE, SITD, SAN. FaceForensics to learn the generalization. This model generalizes well on different architectures and unseen datasets. The Augmentation process improved the robustness of the model to different architectures, datasets and different image-processing techniques (JPEG compression, resizing, and blurring). The main focus of this model is to detect CNN fingerprints of CNN-generated images by training the model on a very large diverse dataset to learn generalization factors. The results are promising and far more improved than previous models. And the trained model even had very good robustness to blurriness and JPEG compression as well. But this model cannot detect the images generated by some deep-learning models where the CNN is a small part of large deep neural network architectures(eg: deep-fakes).

INTRODUCTION:

With the advancement in image synthesis using different models of GAN such as progressive growth of GANs (ProGAN), StyleGAN, and BigGAN, etc. It's easy to synthesize highly photorealistic images and videos, which is highly impossible to recognize real and fake by the naked eye. This led to Global issues mainly due to deepfake face replacement and synthesis of photorealistic synthetic humans.

With GANs, creating a speech video with the synthesized fake face of a famous politician caused huge problems in society. So, effectively detecting fake images has become one of the leading research topics in today's world.

To detect fake images generated by CNN, We built a model and trained on a dataset generated by a single CNN model(StyleGAN). Rather than using ProGAN, we used StyleGAN images for training as it provides the upgraded version of the ProGAN generator. The diversity of images while training a model is important, as it outperforms well than the model trained on small datasets. Binary classifiers are robust with the help of data augmentation operations such as JPEG compression, blurring, and resizing of images. Earlier works studied generalization as a prominent problem in the image forensics field.To focus on this field Sheng Yu wang used many steps to approach the problem.The main contributions of implemented code -

1) The forensics model trained a binary classifier [Resnet152] on a single CNN (used Style which is a high-performing unconditional GAN model than ProGAN) as negative examples. Though we used a smaller dataset to train the model, it exhibited a remarkable amount of generalization to other CNN architectures.
2) Model will be evaluated on the ForenSynths dataset which consists of images generated by 10 different CNN models.
3) Analyzed factors robust the cross-model generalization with the augmentation process.

PROBLEM DEFINITION:

Image Forensics has become the major challenging field for researchers as different technical image editing methods keep emerging in today's world. Fake images are being generated by convolutional neural network models. Many Deep Neural Networks architectures and efficient techniques are emerging fastly. We need a model that can generalize over architectures, datasets and predict the result with good precision.

OBJECTIVE:

The main idea of our experiments is to train a "real-or fake" classifier[Resnet152] on the StyleGAN dataset, and evaluate how well the model generalizes to other CNN-synthesized images.

TECHNOLOGY USED:

Pytorch(Deep Learning Framework based on Python programming language)
For the choice of classifier, we used ResNet101, ResNet-152 pre-trained with ImageNet and trained it as a binary classification setting (Transfer Learning).
Other python libraries(dependencies):

- SciPy
- Scikit learn
- Numpy
- Opencv_python
- Pillow
- Torch>=1.2.0
- Torchvision

Trained the model on Google COLAB.

PROBLEMS FACED:

- Limited GPU computation power
- Cannot find large and diverse dataset for training

As these models failed to generalize, our model was trained on a diverse dataset which increased performance. But due to limited GPU acceleration, we trained the model on approximately 3 classes but tested it across 8 to 10 different classes. And used the data augmentation process Blur+JPEG(0.5) for all models to increase the generalization ability while detecting fake images.

DATASETS USED:

Training dataset:
https://drive.google.com/file/d/132WMLKn65vgYGjUEGOF2x34-Q17dw-Wv/view?usp=sharing
Validation dataset:
https://drive.google.com/drive/folders/1FlgtItrY9pbhNRBcPaKZDHf3NNI8_Bth?usp=sharing
Testing Dataset: https://drive.google.com/file/d/1z_fD3UKgWQyOTZIBbYSaQ-hz4AzUrLC1/view

Training data consists of only StyleGAN generated images and real images. We use 20 models each trained on a different LSUN(Large-scale Scene Understanding) object category, and generate 36K train images and 200 validation images, each with equal numbers of real and fake

images for each model. In total there are 720K images for training and 4K images for validation.

To study generalization of the model, we used a dataset of fake images generated by 10 different CNN models such as ProGAN, Big-GAN, CycleGAN, StarGAN, GauGAN, DeepFakes, CRN, IMLE, SITD, SAN.

Family	Method	Image Source	# Images
Unconditional GAN	ProGAN [19]	LSUN	8.0k
	StyleGAN [20]	LSUN	12.0k
	BigGAN [7]	ImageNet	4.0k
Conditional GAN	CycleGAN [48]	Style/object transfer	2.6k
	StarGAN [10]	CelebA	4.0k
	GauGAN [29]	COCO	10.0k
Perceptual loss	CRN [9]	GTA	12.8k
	IMLE [23]	GTA	12.8k
Low-level vision	SITD [8]	Raw camera	360
	SAN [13]	Standard SR benchmark	440
Deepfake	FaceForensics++ [33]	Videos of faces	5.4k

Figure 1: Different CNN models used for testing the model

Preprocessing steps used for the data:

- 256×256 resolution is the most commonly shared output size. This is the Image resolution for our dataset.
- For models that produce images at lower resolutions(DeepFake), we rescaled the images using bilinear interpolation to 256 on the shorter side with the same aspect ratio.
- For models that produce images at higher resolution (ProGAN, StyleGAN, SAN, SITD) we keep the images at the same resolution.
- For all datasets, we make our real/fake prediction from 224 × 224 crops (random-crop at training time and center-crop at the testing time).

MODELS USED:

We build two classifiers based on resnet101 and resnet152 models.
For training dataset collection, we used Stylegan.

IMPLEMENTATION:

Previously our model was trained on a **Progan dataset**(180GB zipped, 20 categories, with a validation dataset size of 800GB with 200 images from each of all categories) with a Resnet50 based classifier.
In our up-gradation part of the project we used **Stylegan dataset**:

1. 2gb size, 3 categories(bedroom, car, cat)
2. For validation, we used 100 images(50 real and 50 fake) in each category from stylegan generated images.

This time we trained two classifiers, one based on Resnet101 and the other based on Resnet152 architectures.

Training procedure:

We applied blur and jpeg compression for over 50% data. Our training process was completely done on Google colab. We trained it till 7 epochs with each epoch containing 366 iterations) . Took almost 2hrs for each model to train. During the training, checkpoints will be created.

Checkpoints is the folder where we save our Tensorboard files and models(.pth format), which we continuously save multiple times in the training time. We maintain the best model which gets saved every time when there is an increase in the validation accuracy. And at every 2000th step we save the latest model(here step is iteration, after every iteration in each epoch, we increment step by one)
Checkpoints folder links for:

1. Classifier with Resnet101 and trained on stylegan trained images:
   https://drive.google.com/file/d/1edY7stzT3fVqG0ajGv6w4IQHbgo5mp0H/view?usp=sharing
   and its tested results folder:
   https://drive.google.com/file/d/1--IY2-8_mozYuB34NZn5Tci_FHsotfaQ/view?usp=sharing
2. Classifier with Resnet152 and trained on stylegan trained images:
   https://drive.google.com/file/d/1--iERpFm1Hz2rQtykyYcE9pps7jNCvc8/view?usp=sharing
   and its tested results folder:
   https://drive.google.com/file/d/1-2ulRFE2e_mVSOK8EiXnwV9iXhAgDLHE/view?usp=sharing

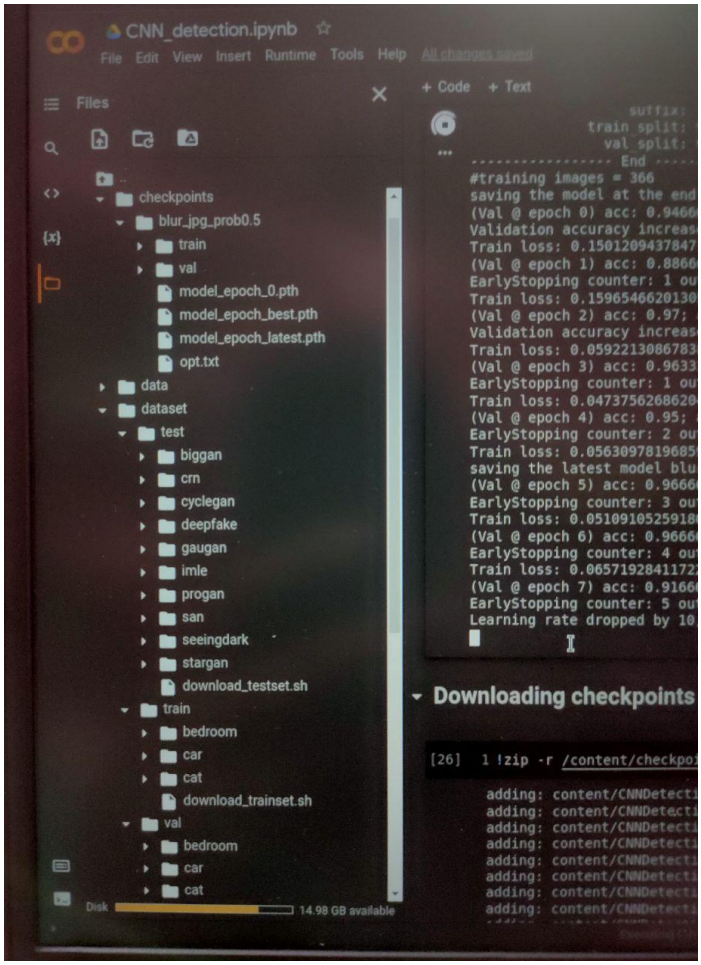

Figure2: Our datasets structure

Testing:
We tested our models on biggan, crn, cyclegan, deepfake, gaugan, imle, progan, san, seeingdark and stargan images. In Figure1, test datasets details are given.

RESULTS:
Model based on Resnet101 testing results:

```
model_epoch_best model testing on ...
(progan) acc: 0.51225; ap: 0.6759500440878026
(biggan) acc: 0.53625; ap: 0.6119325414641495
(cyclegan) acc: 0.5280090840272521; ap: 0.5820448611434252
(stargan) acc: 0.49899949974987495; ap: 0.5345935768965911
(gaugan) acc: 0.5512; ap: 0.693845750823106
(crn) acc: 0.5356471325603259; ap: 0.6826150726254504
(imle) acc: 0.5704324663115011; ap: 0.7374186556418209
(seeingdark) acc: 0.5166666666666667; ap: 0.518665789878996
(san) acc: 0.4931506849315068; ap: 0.4556791265135237
(deepfake) acc: 0.5013876040703052; ap: 0.434107030085052
```

Figure9: Model based on Resnet101 test results

Model based on Resnet152 testing results:

```
model_epoch_best model testing on ...
(progan) acc: 0.52675; ap: 0.711231794297909
(biggan) acc: 0.5505; ap: 0.6512569038196017
(cyclegan) acc: 0.5246025738077215; ap: 0.574665044528365
(stargan) acc: 0.5002501250625313; ap: 0.5434575576160824
(gaugan) acc: 0.5606; ap: 0.6999176862106167
(crn) acc: 0.5269507991225322; ap: 0.6904538903873825
(imle) acc: 0.554763397054215; ap: 0.7836424926139569
(seeingdark) acc: 0.5888888888888889; ap: 0.6022603188315547
(san) acc: 0.4977168949771689; ap: 0.5086181374778624
(deepfake) acc: 0.5008325624421832; ap: 0.49733598326701994
```

Figure10: Model based on Resnet152 test results

Model trained on Resnet152 is giving slightly better results than the model trained on Resnet101.

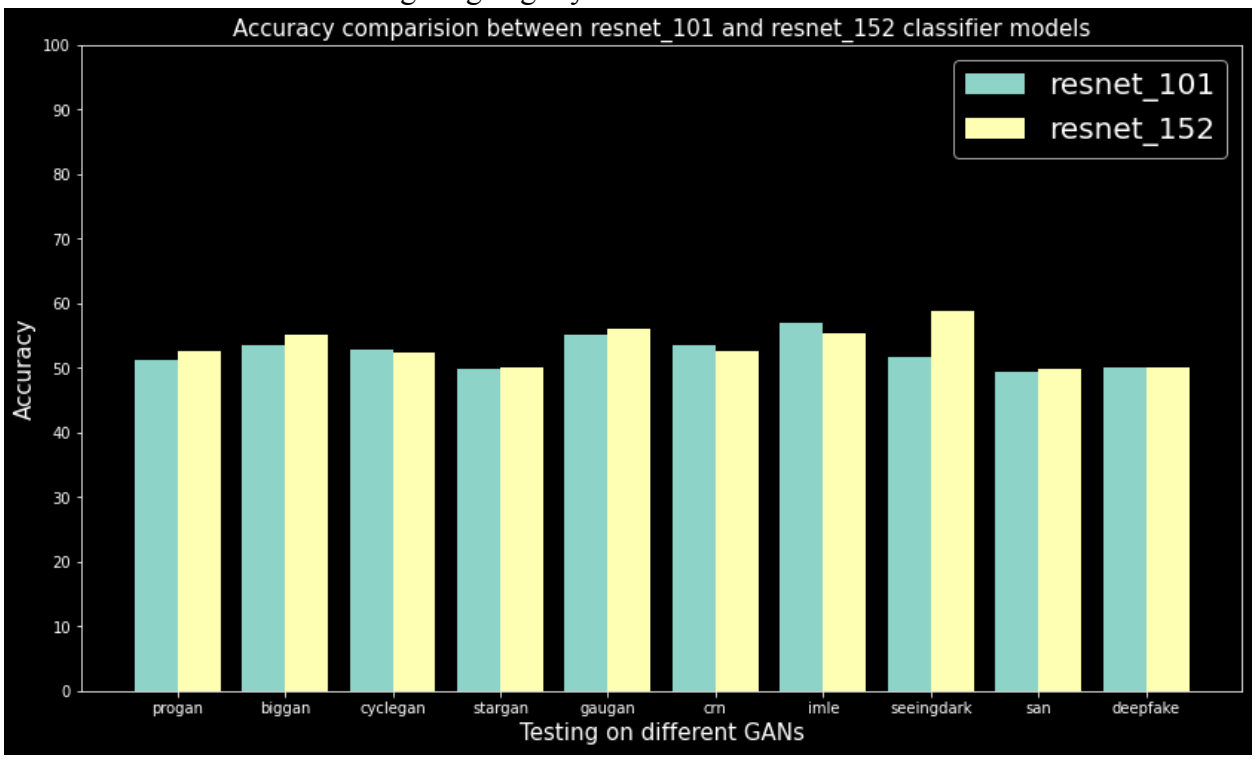

Figure11: Comparison between models based on resnet101 and resnet152

CONCLUSION:

1. Since we have trained our model on a very small dataset ,we didn't get higher accuracy.
2. We have got our results close to randomness.
3. Training with large data results in higher accuracy
4. Also the data set we have trained consists of only 3 classes.
   Increasing the diversity of the dataset also increases the model accuracy.

Reference links:

1. https://www.linkedin.com/pulse/nutshell-gan-progan-stylegan-stylegan2-ibrahim-sobh-phd
2. https://pytorch.org/vision/0.8/_modules/torchvision/models/resnet.html