EECS6322 Reproducibility Challenge Final Report

Paper Title: *CNN-generated images are surprisingly easy to spot... for now*

GitHub Project: https://github.com/SwithinTan/EECS6322_Reproducibility_Challeng_Project

Main Contribution

In the paper, the main contributions are:

1. They introduced a new approach to detect CNN-generated images, by training forensics models on CNN-generated images. This approach shows a surprising amount of generalization compared with other CNN systhesis methods.
2. They introduced a new dataset and evaluation metrics for detecting CNN-generated images.
3. They experimentally analyzed the factors that account for cross-model generalization.

Data sets

Train data: The train data is provided by the author in their publication. The authors used 20 ProGAN models, each trained on a different LSUN category, to sample 18000 CNN generated images per model. Also, they collected 18000 training images for each class. So for each class, there are 18000 real images and 18000 fake images. And there are 720000 images in total.

Test data: The test dataset is consist of images from 11 different synthesis models.

1. GANs: It contains samples from six different Generative Adversarial Networks (GANs): ProGAN, StyleGAN, BigGAN, GauGAN, CycleGAN, and StarGAN.
2. Perceptual loss: It includes samples from generative models which are directly trained to optimize a perceptual loss: Cascaded Refinement Networks (CRN) and Implicit Maximum Likelihood Estimation (IMLE).
3. Image manipulation models: It includes samples from Seeing In The Dark (SITD), and the Second-Order Attention Network (SAN).
4. Deep fakes

Validation data: The validation data is randomly sampled 10% from the training dataset.

About My Reproducibility Attempt

There are two main attempts in my reproducibility project that correspond to the two main contributions of the original paper:

1. The first attempt is to verify if the data augmentations improve generalization.
2. The second attempt is to verify if more diverse datasets improve generalization on unseen architectures.

Claim 1: Data Augmentations Improve Generalization

1. Dataset

   For the training, the training dataset is consist of original images and fake images generated by CNN generator, ProGAN.

   The training dataset used in the paper can be downloaded via this google drive link.

2. Models

   Firstly, I want to verify the first claim - whether data augmentation can improve the generalization or not. I trained a series of different models with four different image augmentation techniques with one baseline.

   - Gaussian blur (Blur): Images from the original dataset have 50% probablity to be blurred.
   - JPEG-compressing (JPEG): Images from the dataset have 50% probablity to be JPEG-compressed by two image libraries, OpenCV and PIL.
   - Gaussian blur + JPEG-compreesing (Blur + JPEG (0.5)): Images from the dataset have 50% probability to be blurred and JPEG-compressed. (The two are independent from each other.)
   - Gaussian blur + JPEG-compressing (Blur + JPEG (0.1)): Images from the dataset have 10% probability to be blurred and JPEG-compressed. (The two are independent from each other.)
   - No Augmentation (No Aug): As a baseline, I also trained one more model on the images without any augmentation.

   There are 5 models that were trained with training datasets of four image augmentation and one more dataset without any augmentation as a baseline model.

   The 5 models that I trained are now on google drive, and can be downloaded.

3. Training Details

   I sampled 10% of the training data as validation data. I trained all models with batch size of 64, and an initial learning rate at 10e-4. The learning rate will drop by 10 times after the validation accuracy stagnates for 5 epochs. I used Google colab for training and the Tesla T4 gpu for acceleration. For each model, it took around 8 hours for training.

4. Model Evaluation

Augmentation	ProGAN	StyleGAN	BigGAN	CycleGAN	StarGAN	GauGAN	CRN	IMLE	SITD	SAN	DeepFake	StyleGAN2	Whichfaceisreal	mAP
Blur	0.879	0.6039893173	0.5335	0.6570779712	0.7423711856	0.5485	0.8560795989	0.8310090881	0.6305555556	0.5114155251	0.5037927845	0.6156109164	0.5145	0.64826
JPEG	0.8175	0.5363044567	0.5425	0.5806207419	0.5477738869	0.5389	0.5735662802	0.562206205	0.6	0.497716895	0.5285846438	0.5087005508	0.5125	0.56514
Blur+JPEG(0.1)	0.848625	0.5405608413	0.546	0.6824375473	0.7108554277	0.556	0.7663741774	0.7433406456	0.575	0.5228310502	0.513413506	0.5210315473	0.505	0.61780
Blur+JPEG(0.5)	0.80425	0.5219495911	0.52675	0.6124148372	0.58004002	0.5449	0.5365089314	0.5680037606	0.575	0.5273972603	0.5097132285	0.5123309965	0.502	0.56317
No_Augmentation	0.950375	0.7053914205	0.5875	0.6937925814	0.8231615808	0.5982	0.9738326543	0.8278752742	0.6833333333	0.6095890411	0.5768732655	0.6371432148	0.598	0.71269

Table 1: **The Accuracy of models trained with different augmentation methods.**

Augmentation	ProGAN	StyleGAN	BigGAN	CycleGAN	StarGAN	GauGAN	CRN	IMLE	SITD	SAN	DeepFake	StyleGAN2	Whichfacei
Blur	0.9900821954	0.893717847	0.6777326608	0.8488602249	0.9716404992	0.7321058271	0.9437832129	0.9184226582	0.9187388981	0.5682211066	0.6948618278	0.8727467873	0.67515706
JPEG	0.9810454486	0.7988384267	0.7454003548	0.8212859274	0.7750122375	0.8612231284	0.9875796059	0.9878788817	0.7641504204	0.538376404	0.6791221034	0.6508763666	0.71511630
Blur+JPEG(0.1)	0.9870773324	0.8051647144	0.6968184852	0.8893030685	0.8762956476	0.8364431426	0.9822766626	0.9776496234	0.7859228343	0.6177323581	0.6772670675	0.7381824488	0.72494373
Blur+JPEG(0.5)	0.9728522685	0.7073263548	0.656408311	0.8559367581	0.8364330301	0.8411573479	0.9061048455	0.9585468321	0.6842257096	0.5621723013	0.6152030867	0.6371878851	0.59012960
No_Augmentation	0.996335379	0.9536072638	0.7243253601	0.8099999361	0.9855824952	0.7501473913	0.9970182397	0.9690621987	0.9052079268	0.793899671	0.9596584558	0.9172850197	0.68850243

Table 2: **The Average Precision of models trained with different augmentation methods.**

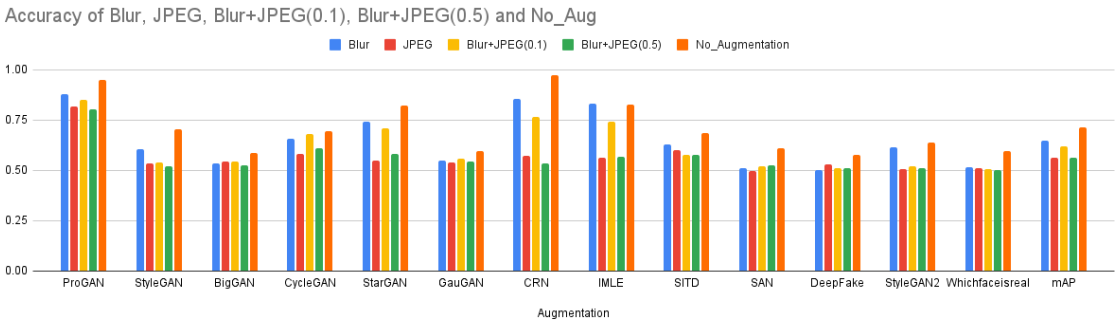

Figure 1: **The effect of different augmentation methods by accuracy.**

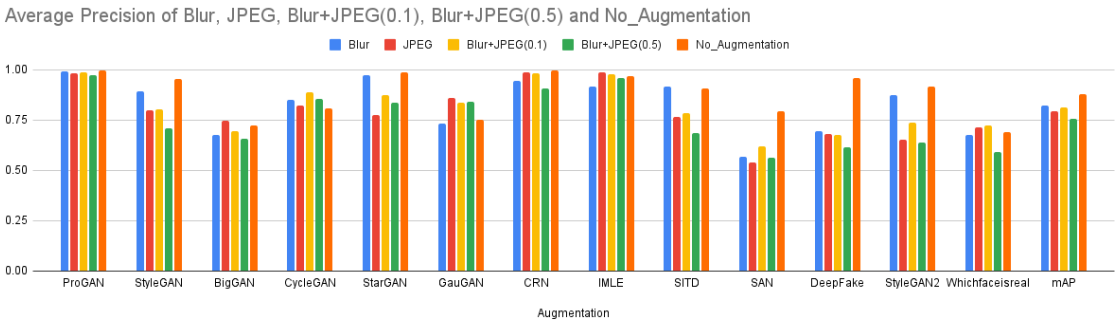

Figure 2: **The effect of different augmentation methods by average precision.** The classifiers trained with CycleGAN, GauGAN, IMLE, whichfaceisreal show better performance with augmentation training data. However, the classifiers trained with StyleGAN, StarGAN, CRN, SITD, SAN, DeepFake and StyleGAN2 shows lower performance with augmentation. This conclusion is contradictory with that in the original paper. This may be because the classifiers are trained with only 1 class of training data, which is not suffcient for model training.

Claim 2: Diverse Datasets Improve Generalization

1. Dataset

   For the training, the training dataset is consist of original images and fake images generated by CNN generator, ProGAN.

   The training dataset used in the paper can be downloaded via this google drive link.

2. Models

   Here, I want to verify the second claim - whether a more diverse training dataset can improve the generalization or not. I trained a series of different models with four different image augmentation techniques with one baseline.

   - 1 class: Airplane image dataset.
   - 3 class: Airplane, person and cat image datasets.
   - 5 class: Airplane, person, cat, potted plant, and tv monitor image datasets.

   There are 3 models that trained with different volume of training datasets, one, three and five datasets.

   The 3 models that I trained are now on google drive, and can be downloaded.

3. Training Details

   I sampled 10% of the training data as validation data. I trained all models with batch size of 64, and an initial learning rate at 10e-4. The learning rate will drop by 10 times after the validation accuracy stagnates for 5 epochs. I used Google colab and the Tesla T4 gpu for acceleration. The training time for the three models are 7 hours, 10 hours, 15 hours respectively.

4. Model Evaluation

Training Classes	ProGAN	StyleGAN	BigGAN	CycleGAN	StarGAN	GauGAN	CRN	IMLE	SITD	SAN	DeepFake	StyleGAN2	Whichfaceisreal	mAP(mean Average Precision)
1 Classes	0.80425	0.5219495911	0.52675	0.6124148372	0.58004002	0.5449	0.5365089314	0.5680037606	0.575	0.5273972603	0.5097132285	0.5123309965	0.502	0.5631737404
3 Classes	0.928875	0.5476548156	0.51725	0.6866010598	0.6625812906	0.622	0.6295832028	0.7817298652	0.7222222222	0.50456621	0.5191489362	0.5248497747	0.6195	0.6358894136
5 Classes	0.977375	0.6023201469	0.54375	0.7108251325	0.7083541771	0.6736	0.6225321216	0.7440457537	0.6833333333	0.497716895	0.513413506	0.526977967	0.607	0.6470187718

Table 3: **The Accuracy of models trained with 1, 3, and 5 classes of data.**

Training Classes	ProGAN	StyleGAN	BigGAN	CycleGAN	StarGAN	GauGAN	CRN	IMLE	SITD	SAN	DeepFake	StyleGAN2	Whichfaceisreal
1 Classes	0.9728522685	0.7073263548	0.656408311	0.8559367581	0.8364330301	0.8411573479	0.9061048455	0.9585468321	0.6842257096	0.5621723013	0.6152030867	0.6371878851	0.5901296041
3 Classes	0.9956857817	0.8697977039	0.7211204318	0.9058420484	0.8994392068	0.9383659032	0.9627024299	0.9894784473	0.7927950189	0.5903214756	0.613959178	0.7877579329	0.8613755047
5 Classes	0.9984525573	0.9526830379	0.8088701782	0.9269493029	0.9208146783	0.9621299622	0.9787728989	0.9947204722	0.8345401325	0.6276014018	0.6058929992	0.850303794	0.886694401

Table 4: **The Average Precision of models trained with 1, 3, and 5 classes of data.**

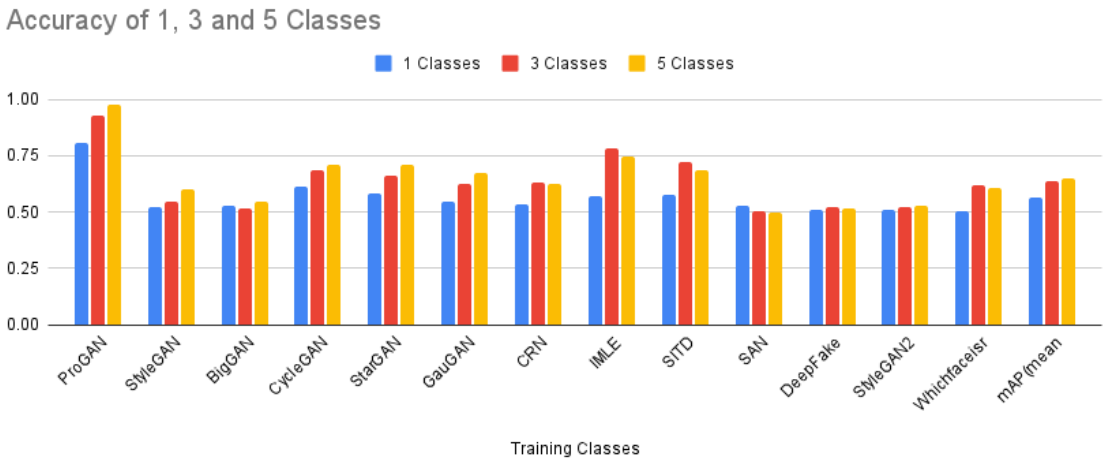

Figure 3: **The Effect of diverse dataset by accuracy.**

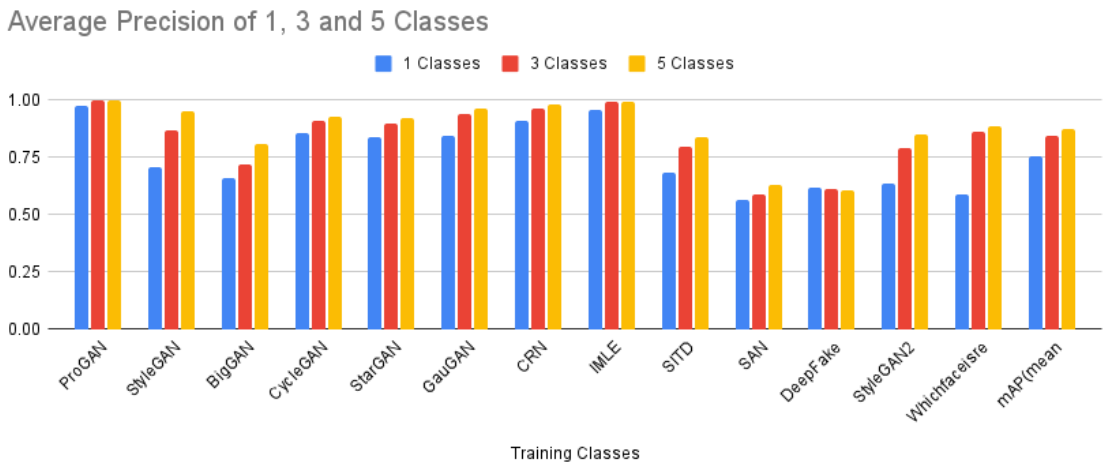

Figure 4: **The Effect of diverse dataset by average precision.** All classifiers are trained on ProGAN dataset, and tested on other generators data (AP shown). Training with more classes generally improves performance. All runs use blur and JPEG augmentation with 50% probability.

Discussion

Difficulties

1. Data Availability: Dataset downloader provided by the author has been deprecated, and the size of dataset is relatively large (~70GB), which causes some trouble when I was trying to ingest dataset. This problem has been solved by downloading original datasets and saving to another drive.
2. Limited Computation Power and long training time: As the training dataset is huge, the training time increases. Also, training on google colab for more than 12 hours is likely to corrupt accidentally. And once the training crashes, the model will not be saved. One solution is to save the best performance model so far onto google drive while training, and once crash happens, reloading the saved checkpoints from google drive again.

Future Direction

1. For now, I have only tested on one classification model, ResNet50. As an result, the conclusion that is drawn from this research could be dependent on the classifier. However, due to the limit of time and computation power, I haven't tried other classifiers. In the future research, other classification models like VGG could be introduced and test if the result is independent from the kind of classification model that used or not.
2. In this reproducibility project, and in the verification of claim 1, I only used one class - airplane for training and comparing 5 trained models. That could be the reason why the verification of claim 1 is not successful compared with the verification of claim 2. Because it's supposed to be using all 20 classes for the 5 models' training. I tried to train one model with all 20 classes, however, the program crashed after 20 hours training. I will try to redesign and implement this reproducibility project if computation power and time allows in the future.