GAN Image Detection: Up-Sampling Artifact & GAN Pipeline Emulator

CVPR Workshop on Media Forensics

Xu Zhang and Shih-Fu Chang

06/17/2019

Goals:

- Are there “artifacts” induced in the GAN image generation pipeline?
  - We explore a phenomenon and a theory related to **up-sampling artifact (checkerboard pattern)**.
- Are there ways to relax knowledge about the GAN models when training fake image classifier?
  - We propose a GAN pipeline emulator called AutoGAN.
Introduction

- 3 popular scenarios of image generation using GAN
  - Generating images from Noise
    - DCGAN [2016], ProgGAN [2017], StyleGAN [2018], BigGAN [2018]
    - Lack control of the generated content

[Karras et. al, 2018a] ProgressiveGAN

[Karras et. al, 2018b] StyleGAN

[Brock et. al, 2018] BigGAN
Introduction

- 3 popular scenarios of image generation using GAN
  - Image to Image Translation: transfer images from one category/style to another
    - Pix2Pix [2016], CycleGAN [2017], StarGAN [2018], FaceSwap/DeepFake/FaceApp
  - Provide more control of the generated content

FaceApp by Facebook

DeepFake

[https://www.alanzucconi.com/2018/03/14/introduction-to-deepfakes/](https://www.alanzucconi.com/2018/03/14/introduction-to-deepfakes/)
Introduction

• 3 popular scenarios of image generation using GAN
  
  • Sketch to Image Translation
    • Pix2Pix [2017], CycleGAN [2017], GauGAN [2019]
    • Similar to image to image translation, but give even more controls to the generated content.

[Isola et. al, 2017]
Pix2Pix

[Zhu et. al, 2017]
CycleGAN

[Park et. al, 2019]
GauGAN
A Common Pipeline for Image2Image or Sketch2Image Transfer

\[
\min_G \max_D V(D, G) = E_{b \sim p_b(b)} (\log D(b)) + E_{a \sim p_a(a)} \log (1 - D(G(a)))
\]

- **Real Zebra Images**
- **Discriminator (D)**
- **GAN Generated zebra images from horse images.**
- **Encoder (E)**
- **Generator (G)**
- **Real Horse Images**
- **Low-resolution features**

Category b

Category a
(An Incomplete) Review of Defense Tools

- Statistical Machine Learning + Feature Design
  - [Marra et. al 2018a] Use raw pixel and conventional forensics features. CNN, SVM, CycleGAN data
  - [Yu et. al 2018] Use raw pixel to detect noise2image GAN. CNN, ProGAN, SNGAN, and SAGAN
  - [Nataraj et. al 2019] Train with Co-Occurrence matrix. VGG-like, cycleGAN+StarGAN
  - [Marra et. al 2018b] Extract fingerprint from GAN. Correlation, cycleGAN+StarGAN

- Special Observations:
  - [McCloskey et. al 2018] GAN generated image doesn’t have saturation region. SVM, NIST GAN challenge data
  - [Li et. al 2018] Deepfake video has no blinking eye. LSTM+VGG, Deep Fake

- Attribute Verification of Test Video against Real Video
  - [Agarwal et al 2019] Study the movement of the action unit of the leader from real video and see whether the generated video matches.
A Popular Baseline: Train a Fake/Real Image Classifier

- **Design Issues**
  - How to collect training samples?
  - What features to use?
Data Bias Pitfall

- In order to train a robust classifier we need, [Marra et al. 2018, Nataraj et al. 2019]
  - diverse training image content (avoid bias)
  - diverse generation models
Leave-one-out strategy to avoid data bias

- Collecting real images and GAN generated image from a variety of sematic transfer pairs. [Marra et. al 2018, Nataraj et. al 2019]

- Train with leave-one-out strategy: 10 transfer pairs/folds, leave one fold out for test.

![Diagram of Leave-one-out strategy]
Results (leave one out)

- Leave one out performs pretty well, but need training data from diverse sources.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DenseNet</td>
<td>79.1</td>
<td>95.8</td>
<td>67.7</td>
<td>99.0</td>
<td>93.8</td>
<td>78.3</td>
<td>99.5</td>
<td>97.7</td>
<td>99.9</td>
<td>89.8</td>
<td>90.1</td>
</tr>
<tr>
<td>Steganalysis feature</td>
<td>98.9</td>
<td>98.4</td>
<td>66.2</td>
<td>100.0</td>
<td>97.4</td>
<td>88.1</td>
<td>97.9</td>
<td>99.7</td>
<td>99.8</td>
<td>98.5</td>
<td>94.5</td>
</tr>
<tr>
<td>Cozzalino2017</td>
<td>99.9</td>
<td>100.0</td>
<td>61.2</td>
<td>99.9</td>
<td>97.3</td>
<td>99.6</td>
<td>100.0</td>
<td>99.9</td>
<td>100.0</td>
<td>99.2</td>
<td>95.7</td>
</tr>
<tr>
<td>XceptionNet</td>
<td>95.9</td>
<td>99.2</td>
<td>76.7</td>
<td>100.0</td>
<td>98.6</td>
<td>76.8</td>
<td>100.0</td>
<td>99.9</td>
<td>100.0</td>
<td>95.1</td>
<td>94.2</td>
</tr>
<tr>
<td>Nataraj2019</td>
<td>99.8</td>
<td>99.8</td>
<td>99.7</td>
<td>92.0</td>
<td>80.6</td>
<td>97.5</td>
<td>99.6</td>
<td>100.0</td>
<td>99.6</td>
<td>99.2</td>
<td>96.8</td>
</tr>
</tbody>
</table>
What if we train with one semantic class only?

- Performance downgrades significantly.
- Classifier does not generalize well to other categories

<table>
<thead>
<tr>
<th>Training</th>
<th>Horse</th>
<th>Zebra</th>
<th>Summer</th>
<th>Winter</th>
<th>Apple</th>
<th>Orange</th>
<th>Facades</th>
<th>CityScapes</th>
<th>Map</th>
<th>Ukiyoe</th>
<th>Van Gogh</th>
<th>Cezanne</th>
<th>Monet</th>
<th>Photo</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horse</td>
<td>98.8</td>
<td>75.4</td>
<td>95.4</td>
<td>85.6</td>
<td>87.5</td>
<td>79.8</td>
<td>62.3</td>
<td>67.3</td>
<td>84.7</td>
<td>65.7</td>
<td>95.0</td>
<td>92.1</td>
<td>90.7</td>
<td>90.6</td>
<td>83.6</td>
</tr>
<tr>
<td>Zebra</td>
<td>87.7</td>
<td>98.8</td>
<td>95.4</td>
<td>92.1</td>
<td>57.2</td>
<td>57.8</td>
<td>50.5</td>
<td>53.9</td>
<td>50.1</td>
<td>66.3</td>
<td>89.7</td>
<td>64.5</td>
<td>89.2</td>
<td>90.3</td>
<td>74.5</td>
</tr>
<tr>
<td>Summer</td>
<td>88.8</td>
<td>87.3</td>
<td>98.7</td>
<td>99.8</td>
<td>76.1</td>
<td>76.3</td>
<td>50.9</td>
<td>59.5</td>
<td>77.0</td>
<td>94.5</td>
<td>91.9</td>
<td>93.7</td>
<td>90.5</td>
<td>94.3</td>
<td>84.2</td>
</tr>
<tr>
<td>Winter</td>
<td>84.6</td>
<td>82.7</td>
<td>98.2</td>
<td>98.9</td>
<td>74.7</td>
<td>69.6</td>
<td>50.0</td>
<td>50.4</td>
<td>88.5</td>
<td>96.7</td>
<td>82.5</td>
<td>93.3</td>
<td>87.3</td>
<td>92.9</td>
<td>82.2</td>
</tr>
</tbody>
</table>
Is It Recognizing Real vs. Fake images?

- Or is it recognizing other differences?
  - High-quality horse vs. low-quality horse
  - Horse habitats vs. zebra habitats
What if we change it to the frequency domain?

- Use frequency-domain data as input to the classifier
- Convert 3 RGB channels to the spectrum of each channel as input.
Directly Train with DFT Spectrum, using one class only

- Performance is significantly improved
- The generalization ability is much better than training with RGB images
Explaining the Success of Spectrum Input

- Explore the signal processing model underlying the GAN synthesis pipeline
Revisit the Pipeline in Image2Image Transfer

\[
\min_{G} \max_{D} V(D, G) = E_{b \sim p_b(b)} (\log D(b)) + E_{a \sim p_a(a)} \log \left(1 - D(G(a))\right)
\]

GAN Loss

GAN Generated zebra images from horse images.

Real Zebra Images \(b\)

Real Horse Images \(a\)

64*64*256 features

Encoder (E)

Generator (G)

Discriminator (D)
Inside the GAN Generator

Feature low-resolution 64x64x256

- Convolution layer 1
- Up-sample layer 1
  - Transposed Convolution [Zhu et. al, 2017]
  - Nearest Neighbor Sampling [Karras et. al, 2018a]
  - Bilinear Sampling [Karras et. al, 2018b]
- Convolution layer N
- Up-sample layer N
- Convolution layer N+1

Output Image High Resolution
Convolution vs. Transposed Convolution (Deconvolution)

http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html
Transposed Convolution = Zero Padding Convolution

Transposed Convolution

Zero Padding Convolution

http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html
Stride 2 Transposed Convolution for Up-sampling

Input: 3*3
Output: 6*6
Kernel: 3*3
Stride: 2

http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html
Zero insertion $\rightarrow$ spectrum artifact

Low-resolution image

128

128

128

Spectrum

128

128

128

Zero Inserted image

256

256

256

256

Spectrum

256

256

256

256
Example from CycleGAN- checkerboard pattern

- Latent vector 64*64*256
- Transposed Convolution layer 1 (3*3 stride 2)
- Transposed Convolution layer 1 (3*3 stride 2)
- Convolution layer (7*7)

Output Image 256*256*3
Effect of the Convolution Kernel

1. It has to be low-pass to remove the up-sampling artifacts
2. It can’t cut too much high frequency, otherwise the detail of the image will disappear.
Spectrum of the Fake Image

- Final Output
Goals:

• Are there “artifacts” induced in GAN image generation pipeline?
  • We explore a phenomenon and a theory related to up-sampling artifact.

• Are there ways to relax knowledge about the GAN models when training fake image classifier?
  • We propose a GAN pipeline emulator called AutoGAN.
AutoGAN – a GAN emulator for generating training samples

- Inspired by CycleGAN, we propose AutoGAN, which emulates the pipeline used in most GAN generation processes
AutoGAN

Real Horse Image

AutoGAN Reconstructed Horse image
Benefits of GAN Pipeline Emulator

- High output image quality
- Different components can be easily incorporated (e.g., different up-samplers)
- Can be applied to any semantic class
AutoGAN does not need category transfer pairs and does not require access to the pre-trained model.

**Pairwise Im2Im Transfer**
- Category 1 to Category 1
- Category 2 to Category 2
- Category 3 to Category 3
- Category N to Category N

Needs to consider all possible pairs (infeasible) or smartly chosen pairs

**AutoGAN**
- Category 1 to Category 1
- Category 2 to Category 2
- Category 3 to Category 3
- Category N to Category N

Can be applied to any category
### Leave One Out Performance

- **Result**
  - Leave one out performs pretty well, but need a huge number of training data from diverse sources.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>95.2</td>
<td>90.0</td>
<td>97.9</td>
<td>76.4</td>
<td>85.1</td>
<td>92.7</td>
<td>98.0</td>
<td>91.6</td>
<td>96.3</td>
<td>97.0</td>
<td>92.0</td>
</tr>
<tr>
<td>Spectrum</td>
<td>99.6</td>
<td>99.4</td>
<td>99.7</td>
<td>100.0</td>
<td>100.0</td>
<td>50.0</td>
<td>100.0</td>
<td>98.0</td>
<td>100.0</td>
<td>99.4</td>
<td><strong>94.6</strong></td>
</tr>
<tr>
<td>Auto</td>
<td>92.7</td>
<td>67.4</td>
<td>98.4</td>
<td>94.8</td>
<td>50.6</td>
<td>51.8</td>
<td>68.7</td>
<td>97.1</td>
<td>57.4</td>
<td>92.5</td>
<td>77.1</td>
</tr>
<tr>
<td>Auto Spectrum</td>
<td>98.7</td>
<td>99.3</td>
<td>99.9</td>
<td>100.0</td>
<td>100.0</td>
<td>79.1</td>
<td>100.0</td>
<td>99.7</td>
<td>97.8</td>
<td>98.7</td>
<td><strong>97.3</strong></td>
</tr>
</tbody>
</table>
Trained with One Semantic Class Only

- Train with spectrum and AutoGAN works well for selective classes
- Conjecture: need classes that have sufficient spectrum coverage

<table>
<thead>
<tr>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Horse</td>
</tr>
<tr>
<td>Horse Auto Spec</td>
<td>99.2</td>
</tr>
<tr>
<td>Zebra Auto Spec</td>
<td>61.9</td>
</tr>
<tr>
<td>Summer Auto Spec</td>
<td>96.9</td>
</tr>
<tr>
<td>Winter Auto Spec</td>
<td>47.3</td>
</tr>
<tr>
<td>COCO Auto Spec</td>
<td>93.8</td>
</tr>
</tbody>
</table>
Does It Work for Different Up-sample Modules?

- Nearest neighbor interpolation
  - Widely used nowadays for up-sampling (Prog-GAN, GauGAN)
  - Can be viewed as zero inserting + low pass filter
  - Suffers less from checkerboard patterns [Odena, et al., 2016].

![Zero Inserted Image](image1)

**Zero Inserted Image**

```
1 0 2 0
0 0 0 0
3 0 4 0
0 0 0 0
```

```
1 1 0
1 1 0
1 1 0
1 1 0
```

![Interpolated Image](image2)

**Interpolated Image**

```
1 1 2 2
1 1 2 2
1 1 2 2
1 1 2 2
```

```
3 3 4 4
3 3 4 4
3 3 4 4
3 3 4 4
```

![Picture of Interpolated Image](image3)
CycleGAN with NN Up-Sampler

Latent vector 64*64*256

128*128 -> \text{NN Interpolation} -> 3*3 Conv *2 -> \text{NN Interpolation} -> 3*3 Conv *2 -> Convolution layer (1*1) -> 256*256*64

Output Image 256*256*3

Spectrum
Up-sample Module Comparison

- Nearest neighbor interpolation causes less checkerboard effect
Train with NN up-sampler and Test with NN up-sampler, One Class

- Spectrum based models still work well for NN up-sample, even if trained on one class only

<table>
<thead>
<tr>
<th>Training</th>
<th>Test</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Horse NN</td>
<td>Zebra NN</td>
<td>Summer NN</td>
<td>Winter NN</td>
<td>Apple NN</td>
<td>Orange NN</td>
<td>Avg</td>
<td></td>
</tr>
<tr>
<td>Horse_NN_Spec</td>
<td>99.6</td>
<td>96.9</td>
<td>86.7</td>
<td>97.1</td>
<td>96.1</td>
<td>93.2</td>
<td>94.9</td>
<td></td>
</tr>
<tr>
<td>Zebra_NN_Spec</td>
<td>100.0</td>
<td>99.6</td>
<td>96.5</td>
<td>99.3</td>
<td>92.2</td>
<td>90.9</td>
<td>96.4</td>
<td></td>
</tr>
<tr>
<td>Summer_NN_Spec</td>
<td>96.2</td>
<td>91.2</td>
<td>99.6</td>
<td>99.8</td>
<td>87.2</td>
<td>85.0</td>
<td>93.2</td>
<td></td>
</tr>
<tr>
<td>Winter_NN_Spec</td>
<td>96.2</td>
<td>96.5</td>
<td>100.0</td>
<td>100.0</td>
<td>93.4</td>
<td>90.3</td>
<td>96.1</td>
<td></td>
</tr>
</tbody>
</table>
Generalization: GANs of different upsamplers

Diagram showing the relationships between different upsampling methods:
- Transposed Conv Upsample
- Nearest Neighbor Upsample

Connections:
- Transposed Conv Upsample to Transposed Conv Upsample: Yes
- Transposed Conv Upsample to Nearest Neighbor Upsample: No
- Nearest Neighbor Upsample to Transposed Conv Upsample: No
- Nearest Neighbor Upsample to Nearest Neighbor Upsample: Yes

Overall, the diagram illustrates the generalization properties of GANs with different upsamplers.
Generalization across different models

- [Nataraj et al. 2019] showed model trained with CycleGAN works well for StarGAN
- StarGAN and CycleGAN share the similar generator structure
- But model learned with cycleGAN (2 up-sampling modules) does not generalize well to GauGAN (5 up-sampling modules)

<table>
<thead>
<tr>
<th>Test</th>
<th>Method</th>
<th>Train with CycleGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Image</td>
<td>Spectrum</td>
</tr>
<tr>
<td>StarGAN</td>
<td>65.06</td>
<td>100.0</td>
</tr>
</tbody>
</table>
Conclusions

- Typical up-sampling modules in GAN leave **up-sampling artifacts** in the generated images.

- **Spectrum-based detectors** seem to be able to reveal the artifacts
  - Training with spectrum input generalizes well even if trained with one class only.

- We also propose **GAN pipeline emulator AutoGAN**, while emulates the up-sampling artifacts in GAN generated image.
  - Relax knowledge about GAN model
  - Does not need access to the GAN model or generated images
Conclusions

- Model trained with one up-sampling module does not generalize well to different up-sampling modules
  - But models trained with multiple modules work

- Model learned with similar up-sampling architectures works (CycleGAN vs. StarGAN), but not distinct models (e.g., GauGAN)