Paper review

Introduction

I will summarise my thoughts and insights on paper **“Disrupting Deepfakes: Adversarial Attacks Against Conditional Image Translation Networks and Facial Manipulation Systems”** which can be found at this page.

Overview

The mentioned paper is concerned in the field of image manipulation detection - facial manipulation detection, to be more concrete - which is about using deep learning in some way to recognise if the image had been manipulated by some sort of attack (for example in our case, if color hair of a person had been changed or a smile is put on a face). The authors propose generating new modified images of a person that a human eye cannot see difference in order to disrupt the output image of a DeepFake network (see Fig. 1). They apply:

1) so-called class transferable adversarial attacks that generalise to many classes of deepfakes (for example, such attacks disrupt both models that changes color hair and put a smile on a face)

2) adversarial training that enable deepfake networks to be robust to such kind of attack

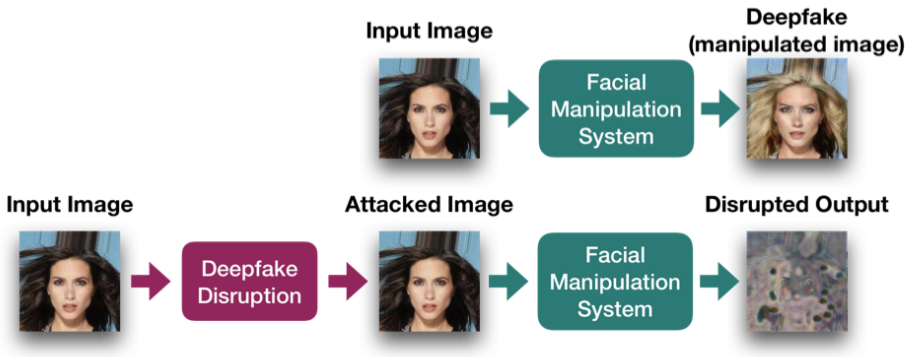

Fig. 1. Illustration of deepfake disruption with a real example. After applying an imperceptible filter on the image using our I-FGSM disruption the output of the face manipulation system (StarGAN) is successfully disrupted. Taken from appropriate paper.

Idea

Image Translation Disruption

We want to generate a disruption by adding a human-imperceptible perturbation η to the input image:

$$x_d = x + \eta,$$

where x_d is the generated disrupted input image and x is the input image. By feeding the original image or the disrupted input image to a generator we have the mappings $G(x) = y$ and $G(x_d) = y_d$, respectively, where y and y_d are the translated output images and G is the generator of the image translation GAN.

Authors consider a disruption successful when it introduces perceptible corruptions or modifications onto the output y_d of the network leading a human observer to notice that the image has been altered and therefore distrust its source.

Therefore, authors seek to $maximise$ the distortion (i.e. distance metrics using the L^0 ,L^2 or L^∞ norms) of output $w.r.t$ a well-chosen reference r**:**

$$\max_{\eta} L(G(x+\eta), r), \text{ w.r.t } ||\eta||_\infty < \epsilon$$

where ϵ is the maximum magnitude of the perturbation and L is a distance function. If we pick r to be the ground-truth output, $r = G(x)$, we get the ideal disruption which aims to $maximise$ the distortion of the output.

Authors also formulate a $targeted$ disruption, which pushes the output y_d to be close to r (it depends on what reference you are choosing in order to set the optimisation of the objective disrupt deepfake network):

$$\eta = \arg\min_{\eta} L(G(x+\eta), r), \text{ w.r.t } ||\eta||_\infty < \epsilon$$

We can thus disrupt an image $towards$ a target or away from a target.

Existing methods

Fast Gradient Sign Method (FGSM) generates an attack in one forward-backward step, and is adapted as follows:

$$\eta = \epsilon \; sign[\nabla_x L(G(x), r)],$$

where ϵ is the size of the **FGSM** step. **Iterative Fast Gradient Sign Method (I- FGSM)** generates a stronger adversarial attack in multiple forward-backward steps. Authors adapt this method for the targeted disruption scenario as follows:

$$x_d{}^{(t)} = clip(x_d{}^{(t-1)} \; - \; a \; sign[\nabla_{x_d} L(G(x_d{}^{(t-1)}), r)]),$$

where a is the step size, t is the time step and the constraint $||x_d - x||_\infty \leq \epsilon$ is enforced by the $clip$ function. For disruptions $away \; from$ the target r instead of $towards \; r$, using the negative gradient of the loss in the equations above is sufficient. For an adapted **Projected Gradient Descent (PGD)**, authors initialise the disrupted image $x_d{}^0$ randomly inside the ϵ-ball around x and use the **I-FGSM** update function.

Conditional Image Translation Disruption

Many image translation systems are conditioned not only on the input image, but on a target class as well:

$$y = G(x, c),$$

where x is the input image, c is the target class and y is the output image. A target class can be an attribute of a dataset, for example blond or brown-haired. So the previous methods should be reformulated to conceive target classes. It is also worth mentioning that disruption for the input image is not guaranteed to transfer it into another semantic class after generator network work, we just want the user to notice attacking because of generating something without a consent (Fig. 1)

Iterative Class Transferable Disruption

A modified **I-FGSM** that increases the transferability of disruption to different classes:

$$x_d{}^{(t)} = clip(x_d{}^{(t-1)} \; - \; a \; sign[\nabla_{x_d} L(G(x_d{}^{(t-1)}, c_k), r)]),$$

Authors initialise $k = 1$ and increment k at every iteration, until $k = K$ reached, where K is the number of classes. Then k resets to 1.

```
import torch

a = "step size"
epsilon = 'amplitude'
K = "number of classes"
model = "pytorch Model"
loss_fn = "loss function"


def mix_att_color_real(att, reg, x_real):
    """Mixes attention, color and real images"""
    return (1 - att) * reg + att * x_real


def perturb_iter_class(X_nat, y, class_target):
        """
        Iterative Class Conditional Attack
        """
        X = X_nat.clone().detach_()

        j = 0
        J = class_target.size(0)

        for i in range(K):
            X.requires_grad = True
            output_att, output_img = model(X, class_target[j,:].unsqueeze(0))

            out = mix_att_color_real(output_att, output_img, X)

            model.zero_grad()

            loss = loss_fn(out, y)
            loss.backward()
            grad = X.grad

            X_adv = X + a * grad.sign()

            eta = torch.clamp(X_adv - X_nat, min=-epsilon, max=epsilon)
            X = torch.clamp(X_nat + eta, min=-1, max=1).detach_()

            j += 1
            if j == J:
                j = 0

        return X, eta
```

Joint Class Transferable Disruption

Authors also propose a disruption which seeks to minimise the expected value of the distance to the target r at every step t. They compute this loss term at every step of an

I-FGSM disruption and use it to inform update step:

$$x_d{}^{(t)} = clip(x_d{}^{(t-1)} \ - \ a \ sign[\nabla_{x_d} \sum_c L(G(x_d{}^{(t-1)}, c), r)]),$$

```
def perturb_joint_class(X_nat, y, class_target):
    """
    Joint Class Conditional Attack
    """
    X = X_nat.clone().detach_()

    J = class_target.size(0)

    for i in range(K):
        full_loss = 0.0
        X.requires_grad = True
        model.zero_grad()

        for j in range(J):
            output_att, output_img = model(X, class_target[j,:].unsqueeze(0))

            out = mix_att_color_real(output_att, output_img, X)

            loss = loss_fn(out, y)
            full_loss += loss

        full_loss.backward()
        grad = X.grad

        X_adv = X + a * grad.sign()

        eta = torch.clamp(X_adv - X_nat, min=-epsilon, max=epsilon)
        X = torch.clamp(X_nat + eta, min=-1, max=1).detach_()

    return X, eta
```

GAN Adversarial Training

As mentioned above, besides disrupting deepfake networks, authors also propose a method to train such networks to be robust against these attacks.

Generator Adversarial Training. A conditional image translation GAN uses the following adversarial loss:

$$V = \mathbb{E}_x[logD(x)] + \mathbb{E}_{x,c}[log(1 - D(G(x, c)))],$$

where D is the discriminator. In order to make the generator resistant to adversarial examples, authors train the GAN using the modified loss:

$$V = \mathbb{E}_x[logD(x)] + \mathbb{E}_{x,c,\eta}[log(1 - D(G(x + \eta, c)))],$$

Generator+Discriminator (G+D) Adversarial Training

Instead of only training the generator to be indifferent to adversarial examples, the discriminator on adversarial examples is also being trained:

$$V = \mathbb{E}_{x,\eta_1}[logD(x + \eta_1)] + \mathbb{E}_{x,c,\eta_2,\eta_3}[log(1 - D(G(x + \eta_2, c) + \eta_3))],$$

Results

As I'm concerned about improving potential disruption of deepfake networks, I will put results associated with this task.

The results of disrupting generated images can be seen below:

Table 1: Comparison of L^1 and L^2 pixel-wise errors, as well as the percentage of disrupted images (% dis.) for different disruption methods on different facial manipulation architectures and datasets. All disruptions use $\epsilon = 0.05$ unless noted. We notice that strong disruptions are successful on all tested architectures.

	FGSM			I-FGSM			PGD		
Architecture (Dataset)	L^1	L^2	% dis.	L^1	L^2	% dis.	L^1	L^2	% dis.
StarGAN (CelebA)	0.462	0.332	100%	1.134	1.525	100%	1.119	1.479	100%
GANimation (CelebA)	0.090	0.017	0%	0.142	0.046	34.9%	0.139	0.044	30.4%
GANimation (CelebA, $\epsilon = 0.1$)	0.121	0.024	1.5%	0.212	0.098	93.9%	0.190	0.077	83.7%
pix2pixHD (Cityscapes)	0.240	0.118	96%	0.935	1.110	100%	0.922	1.084	100%
CycleGAN (Horse)	0.133	0.040	21%	0.385	0.242	100%	0.402	0.253	100%
CycleGAN (Monet)	0.155	0.039	22%	0.817	0.802	100%	0.881	0.898	100%

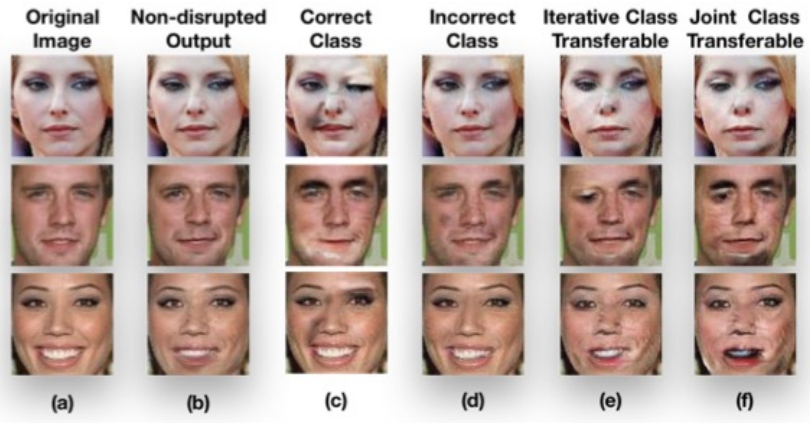

Fig. 4: Examples of our class transferable disruptions. (a) Input image. (b) The ground truth GANimation output without disruption. (c) A disruption using the correct action unit correctly is successful. (d) A disruption with a incorrect target AU is not successful. (e) Our iterative class transferable disruption and (f) joint class transferable disruption are able to transfer across different action units and successfully disrupt the deepfake generation.

Table 4: Class transferability results for our proposed disruptions. This disruption seeks maximal disruption in the output image. We present the distance between the ground-truth non-disrupted output and the disrupted output images, *higher distance* is better.

	L^1	L^2	% dis.
Incorrect Class	0.144	0.053	45.7%
Iterative Class Transferable	**0.171**	**0.075**	**75.6%**
Joint Class Transferable	0.157	0.062	53.8%
Correct Class	0.166	0.071	68.7%

Table 5: Class transferability results for our proposed disruptions. This disruption seeks minimal change in the input image. We present the distance between the input and output images, *lower distance* is better.

	L^1	L^2
Incorrect Class	1.69×10^{-3}	3.09×10^{-4}
Iterative Class Transferable	6.07×10^{-4}	8.02×10^{-5}
Joint Class Transferable	3.86×10^{-4}	$\mathbf{1.67 \times 10^{-5}}$
Correct Class	$\mathbf{9.88 \times 10^{-5}}$	4.73×10^{-5}
No Disruption	9.10×10^{-2}	2.15×10^{-2}

As we can see, such methods deal successfully with given generators

Ideas for improvement

As far as these models were trained on predefined Generative Models (GM), I can conclude they can generalise poorly to GM unseen in training.

To mitigate this problem:

1) a number of GMs used in training should be increased and

2) there should be another way of learning an imperceptible η (without getting access to the gradient flow of the deepfake networks)

3) furthermore, it is better to learn a set of these imperceptible signals η_i ($i = 1, 2, \dots n$) so it would be harder malicious actors to reverse engineer them

4) finally, as it is a demand for η_i to be imperceptible for humans, we can put certain constraints (for example, removing from signals high-frequency components so human eye won't probably notice difference with the original image) in order to have a more stable training procedure.

Reformulating the problem

More formally, the goal is to find η_i for which there is no visual significant difference between x and $x + \eta_i$. More importantly, if $x + \eta_i$ is modified by any GM, this would improve the performance for face manipulation detection.

Let r have a binary nature, indicating $r = 1$ - the image is not being manipulated and $r = 0$indicating image is a deepfake. The objective function can be rewritten as:

$$L_b = -\sum_j [r_j log(net(x + \eta_i; \theta)) + (1 - r_j)(log(1 - net(x + \eta_i; \theta))]$$

where net - binary classification network and θ - its weights.

$$\min_{\eta_i, \theta} L_b$$

Unsupervised Training of signals set

Since there is no ground truth for supervision, we can define various constraints to guide the learning process. Let η_i be the signal selected from the set of signals η. Some loss functions can be defined as shown below:

Magnitude loss. The real image and the disrupted input image should be as similar as possible visually as the user does not want the image quality to deteriorate. Therefore, a regularisation must be used:

$$L_m = ||\eta_i||_2^2$$

Content independent loss. The main aim is to learn a set of universal signals which can be used for detecting manipulated images from unseen GMs. These signals, despite being trained on one dataset, can be applied to images from a different domain (not only to faces domain, in the best case scenario). Therefore, the high frequency information in the signal can be encouraged to be data independent:

$$L_c = ||Filter(\mathbb{F}(\eta_i), k)||_2^2$$

where $Filter$ is the low pass filter selecting the $k \times k$ region in the center of the 2D Fourier spectrum, while assigning the high frequency region to zero. $\mathbb{F}$ is the Fourier transform.

Separation loss. We want the recovered signal η_i from manipulated images $G(x + \eta_i)$ to be different than all the signals in set η. Thus, we optimise η_i to be orthogonal to all the signals in the set η. Therefore, we can take the signal for which the cosine similarity between η_i and the signal is maximum, and minimise its respective cosine similarity:

$$L_s = \max_{k=1..n}(cos(MinMax(\eta_k), MinMax(\eta_i)))$$

where:

$$MinMax(x) = (x - min(x))/(max(x) - min(x))$$

Pair-wise set distribution loss. A signals set would ensure that if the attacker is somehow able to get access to some of the signals, it would still be difficult to reverse engineer other signals. Therefore, a constraint to minimise the inter-signal cosine similarity to push the diversity of the signals in η can be provided:

$$L_p = \sum_{i=1}^{n} \sum_{j=i+1}^{n} cos(MinMax(\eta_i), MinMax(\eta_j))$$

The overall loss function for signals estimation is thus:

$$L = \lambda_1 L_b + \lambda_2 L_m + \lambda_3 L_c + \lambda_4 L_s + \lambda_5 L_p$$

where $\lambda_i (i = 1..5)$ are the loss weights for each term.

Thank you for your attention :)