CS 7150: Deep Learning — Summer-Full 2020 — Paul Hand

HW 2
Due: Wednesday June 24, 2020 at 11:59 PM Eastern time via Gradescope

Name: Saurabh Vaidya
Collaborators: [Put Your Collaborators Here]

You may consult any and all resources. Note that these questions are somewhat vague by design. Part of your task is to make reasonable decisions in interpreting the questions. Your responses should convey understanding, be written with an appropriate amount of precision, and be succinct. Where possible, you should make precise statements. For questions that require coding, you may either type your results with figures into this tex file, or you may append a pdf of output of a Jupyter notebook that is organized similarly. You may use PyTorch, TensorFlow, or any other packages you like. You may use code available on the internet as a starting point.

Question 1. *Image denoising by ResNets*

Consider the following denoising problem. Let x be a color image whose values are scaled to be within [0,1]. Let y be a noisy version of x where each color channel of each pixel is subject to additive Gaussian noise with mean 0 and variance σ^2. You will need to clip the values of y in order to ensure it is a valid image. The denoising problem is to estimate x given y.

(a) Look up the definition of Peak Signal-to-Noise Ratio (PSNR). Determine what value of σ corresponds to an expected PSNR between x and y of approximately 20 dB.

Response:

Answer 1. *Since max pixel value is 1 and MSE is between clean image and noisy image*

$$PSNR = 20.\log_{10}(MAX_I) - 10.\log_{10}(MSE) \tag{1}$$

$$PSNR = 20.\log_{10}(1) - 10.\log_{10}\left(\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}(x[i,j]-y[i,j])^2}{mn}\right) \tag{2}$$

$$PSNR = 0 - 10\log_{10}\left(\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}(x[i,j]-(x[i,j]-\eta[i,j]))^2}{mn}\right) \tag{3}$$

$$PSNR = -10\log_{10}\left(\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}(\eta[i,j])^2}{mn}\right) \tag{4}$$

$$\mathbb{E}[PSNR] = 20 \tag{5}$$

$$\mathbb{E}\left[-10\log_{10}\left(\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}(\eta[i,j])^2}{mn}\right)\right] = 20 \tag{6}$$

$$\mathbb{E}\left[-\log_{10}\left(\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}(\eta[i,j])^2}{mn}\right)\right] = 2 \tag{7}$$

$$\tag{8}$$

By Jensen's inequality

$$\mathbb{E}\left[-\log_{10}\left(\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}(\eta[i,j])^2}{mn}\right)\right] \geq -\log_{10}\mathbb{E}\left[\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}(\eta[i,j])^2}{mn}\right] \tag{9}$$

$$2 \geq -\log_{10}\frac{1}{mn}\mathbb{E}\left[\sum_{i=0}^{m}\sum_{j=0}^{n}(\eta[i,j])^2\right] \tag{10}$$

Let $(\eta[i,j])$ *be a random variable* $\mathcal{N}$ *which is a zero mean Gaussian*

$$2 \geq -\log_{10} \frac{1}{mn} \mathbb{E}\left[\sum_{i=0}^{m}\sum_{j=0}^{n} \mathcal{N}^2\right] \tag{11}$$

$$2 \geq -\log_{10} \frac{1}{mn} mn.\mathbb{E}\left[\mathcal{N}^2\right] \tag{12}$$

$$2 \geq -\log_{10} \mathbb{E}\left[\mathcal{N}^2\right] \tag{13}$$

$$\mathbb{E}[\mathcal{N}^2] = \text{Var}[\mathcal{N}] - \mathbb{E}[\mathcal{N}]^2 \tag{14}$$

$$\mathbb{E}[\mathcal{N}]^2 = 0 \tag{15}$$

Since noise is a 0 mean Gaussian (16)

$$2 \geq -\log_{10} \text{Var}\left[\mathcal{N}\right] \tag{17}$$

$$10^2 \geq \frac{1}{\sigma^2} \tag{18}$$

$$\sigma^2 \geq \frac{1}{100} \tag{19}$$

$$\sigma \geq 0.1 \tag{20}$$

(b) Create a noisy version of the CIFAR-10 training and test dataset, such that it has additive Gaussian white noise with PSNR approximately 20 dB. Show several pairs of images and their noisy version.

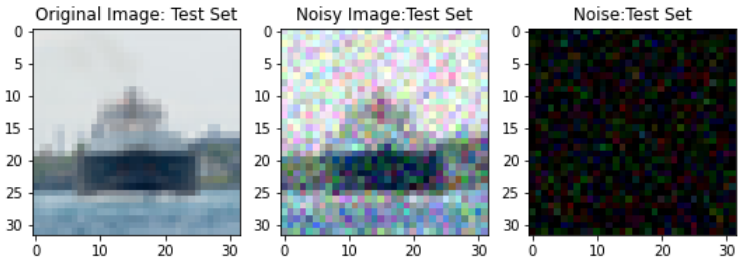
Response:

Answer 2. *As per the above equation, I generated gaussian noise with* $\sigma = 0.1$.

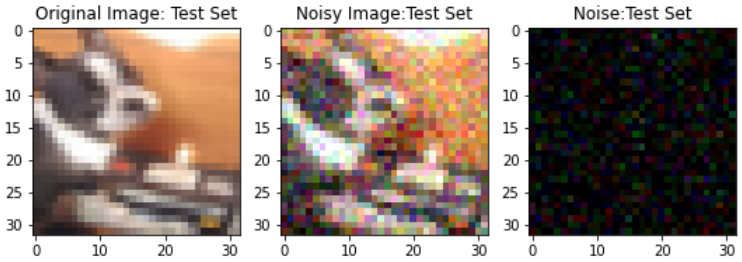
After the noisy images were generated, the PSNR of my noisy dataset with clean dataset was:

PSNR mean 19.997 and std dev 0.11

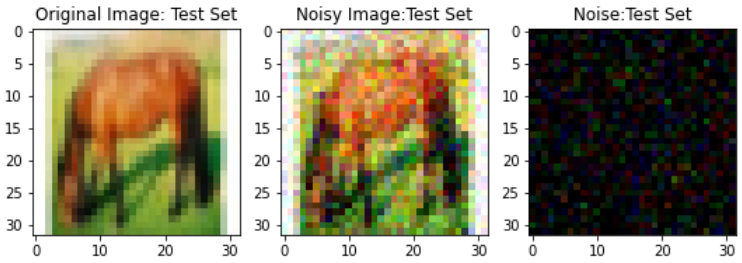
Noisy Image examples:



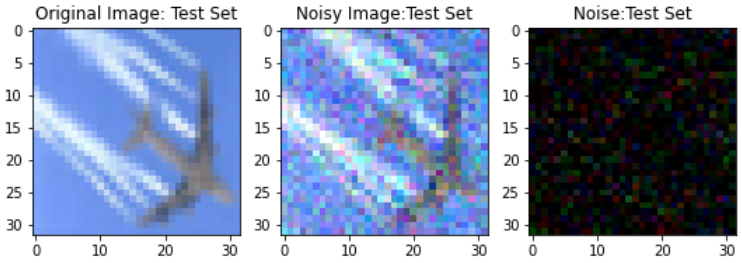
(a) Example 1:



(b) Example 2



(c) Example 3



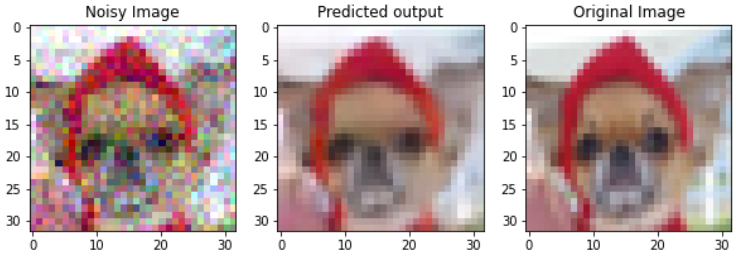
(d) Example 4

Figure 1: Original Image, Noisy Image, Noise

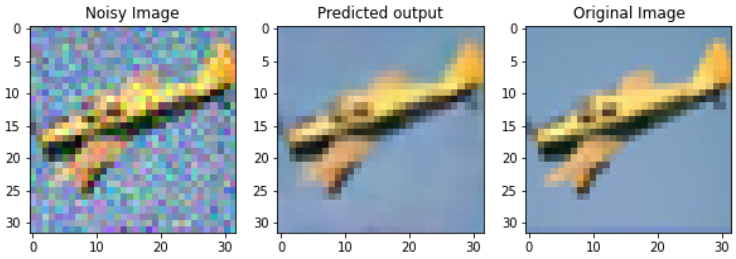
(c) Train a ResNet to denoise noisy CIFAR-10 images. Your net should take a noisy 32x32 px image as an input, and it should output a denoised 32x32 px image. Determine the mean and standard deviation of the recovery PSNRs over the noisy test set. Visually show the performance on several noisy images.

Response:

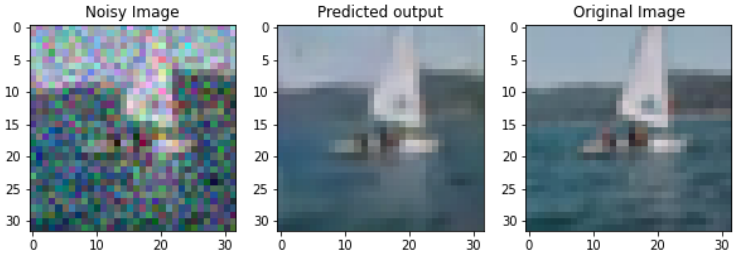
Answer 3. *Mean and standard deviation PSNR of test set are 28.43 and 1.27*



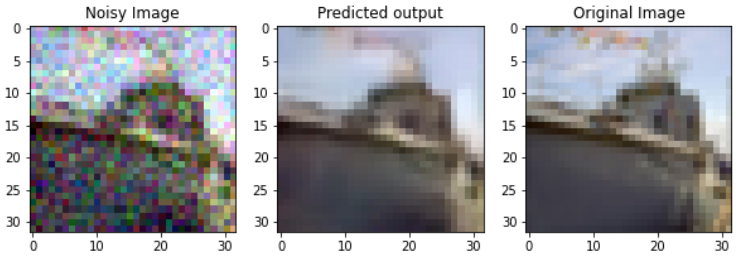
(a) Example 1:



(b) Example 2



(c) Example 6



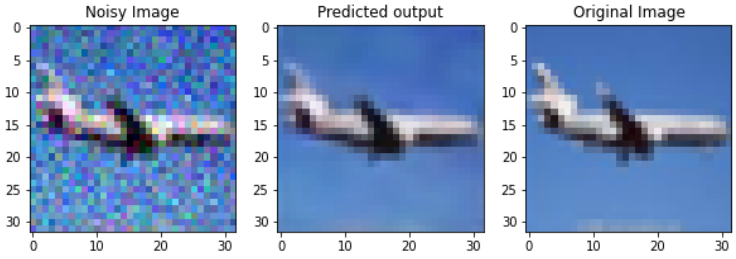
(d) Example 4

Figure 2: Noisy Image, Recovery Image, Original Image

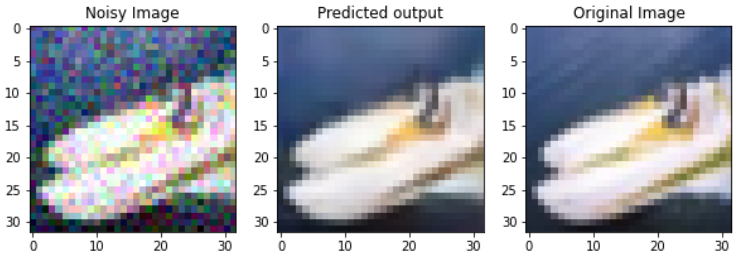
(d) Repeat the previous task but without the skip connections in your model.

Response:

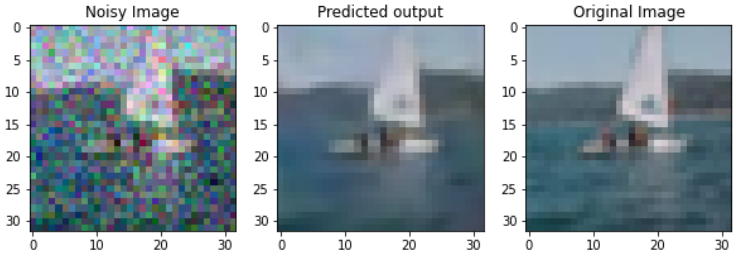
Answer 4. ***Mean and standard deviation recovery PSNR of test set are 28.17 and 1.23***
Examples:



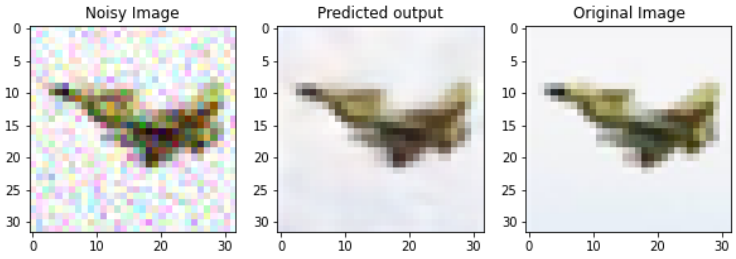
(a) Example 1:



(b) Example 2



(c) Example 6



(d) Example 4

Figure 3: Noisy Image, Recovery Image, Original Image

Question 2. *Adversarial examples*

Obtain an image classifier for CIFAR-10. You may use the one you trained in HW 1.

(a) Select a couple images from the test set. For each image, select a target class that is different from the image's true class. For each image, compute an adversarial perturbation that is barely perceptible to the human eye and that results in the image being classified as the target class. Clearly state the algorithm that you used to generate the perturbation. Show the underlying images, the perturbed image, the perturbation, and the classifier's output.

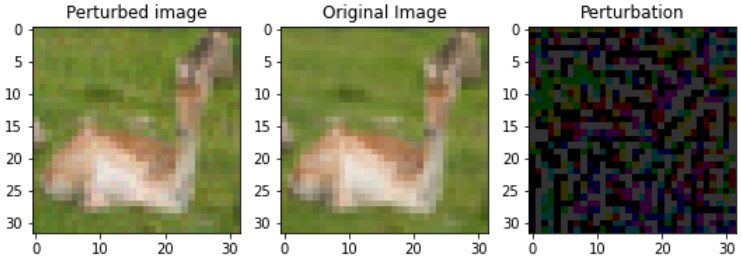
Response:

Answer 5. *The algorithm used to compute was Fast Gradient Sign Method by Goodfellow et al. I implemented the targeted attack where the crossentropy loss function was with attack target label and model output. The algorithm proceeds as follows:*

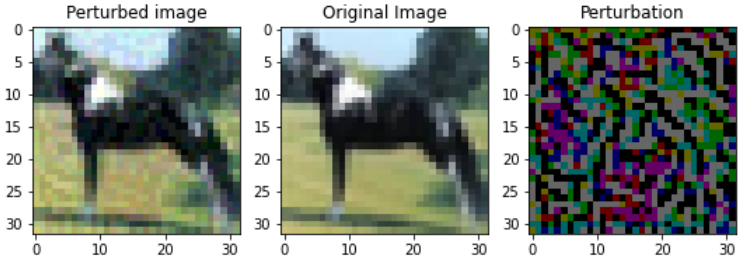
Algorithm 1: Fast Gradient Sign Method targeted attack

Input: *pretrained_model, attack_label, image*
Output: *perturbed_image*
$\epsilon = 0.02$
*loss = -1*categorical_crossentropy(attack_label, model.output)*
gradients = tf.gradient(loss, image)
sign = tf.sign(gradients)
*perturbed_image = image + ϵ*sign*
Return *perturbed_image*

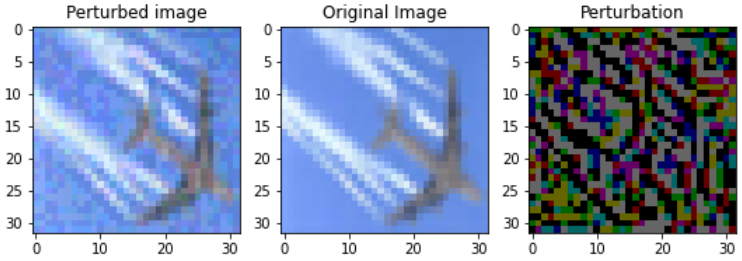
Examples:



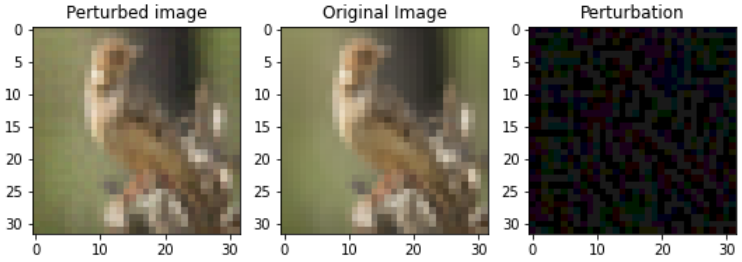
(a) True class Deer, predicted output Dog



(b) True class Horse, predicted output Bird



(c) True class Airplane, predicted output Automobile



(d) True class Bird, predicted output Deer

Figure 4: Examples where before perturbation model predicted true label but after targeted perturbation predict the target class. Noise is scaled(x*10) for visualization

(b) Compute a universal adversarial perturbation for the CIFAR-10 training dataset. Your goal is to compute a single perturbation that can be added to any CIFAR-10 image, resulting in misclassification. Your misclassification does not have to be targeted toward a specific class. Clearly state the algorithm that you used to generate the perturbation. Demonstrate the performance of your perturbation and compute the fraction of the training dataset for which the perturbation results in misclassification.

Response:

Answer 6. *The algorithm for universal perturbation is taken from Universal Adversarial Perturbations Moosavi et al*

Fooling rate was 65.17

Algorithm 2: Universal Perturbations on Cifar10 train set with labels

Input: *Dataset* $\mathcal{X}$*, classifier* $\widehat{k}$*, desired* l_p *norm of perturbations* ξ*, desired accuracy on perturbed samples* δ

Output: *Universal Perturbation vector* v

$v \leftarrow 0$;

While $Err(\mathcal{X}_v) \leq 1 - \delta$

  For $x_i \in \mathcal{X}$

    If $\widehat{k}(x_i + v) == \widehat{k}(x_i)$ **:**

      $r \leftarrow deepfool(x_i + v, \widehat{k})$;

      $v \leftarrow \mathcal{P}_{\xi,p}(r)$ ▷ *Projection on* l_p *ball ;*

Return v

The Deepfool algorithm finds $\longrightarrow \arg\min_r \|r\|_2$ *s.t.* $\widehat{k}(x_i + v) \neq \widehat{k}(x_i)$

For a multiclass classifier a datapoint x gets the label of that class whose hyperplane was closest to x. Deepfool basically just tries to push x beyond that hyperplane causing it to be misclassified.

In a gist, we find a label who has minimum difference of its gradient with input with gradient of true label.. Using the minimum gradients and the difference in the class labels of the chosen label an true label, we calculate the closest hyperplane and a minimal vector that is closest to that

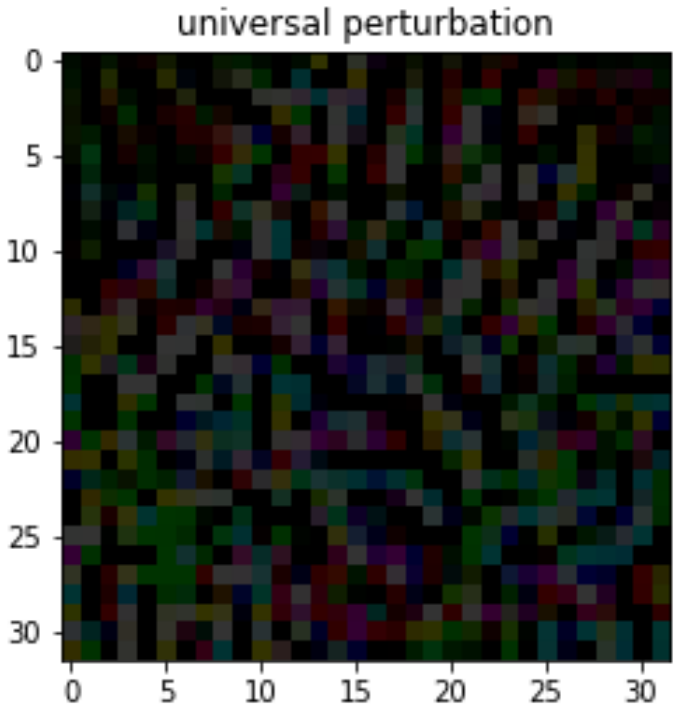


Figure 5: Universal Perturbation (scaled x2 for visualization purpose)

hyperplane. Then we add minimal perturbation to the input and push it beyond that hyperplane.

Algorithm 3: DeepFool adversarial perturbation algorithm

Input: *Image x, a pretrained classifier f such that $\hat{k}(x) = \arg\max_k f_k(x)$ where $f_k(x)$ is the output of $f(x)$ that corresponds to the k^{th} class*

Output: *Perturbation image $\hat{r}$*

$x_0 \leftarrow x$;

$i \leftarrow 0$;

While $\hat{k}(x_0) == \hat{k}(x_i)$

  For $k \neq \hat{k}(x_0)$

    $w_k^{'} \leftarrow \nabla f_k(x_i) - \nabla f_{\hat{k}(x_0)}(x_i)$;

    $f_k^{'} \leftarrow f_k(x_i) - f_{\hat{k}(x_0)}(x_i)$;

  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(x_0)} \frac{|f_k^{'}|}{\|w_k^{'}\|_2}$;

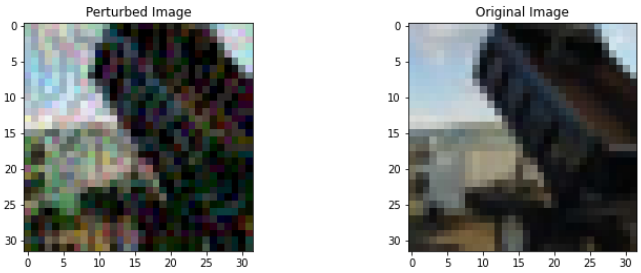
  $r_i \leftarrow \frac{|f^{'}\hat{l}|}{\|w^{'}\hat{l}\|_2^2} w_{\hat{l}}^{'}$;

  $x_{i+1} \leftarrow x_i + r_i$;

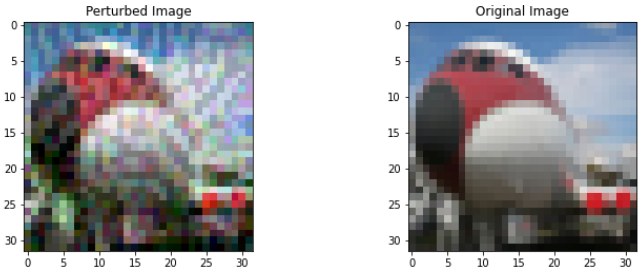
  $i \leftarrow i + 1$;

Return $\widehat{r} = \sum_i r_i$

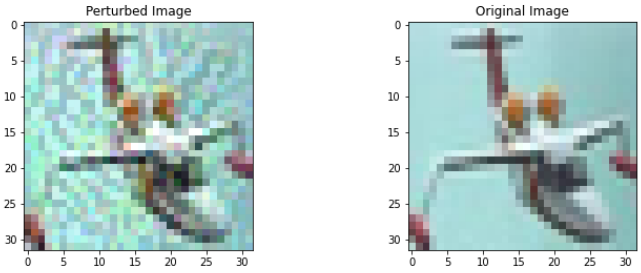
Examples:



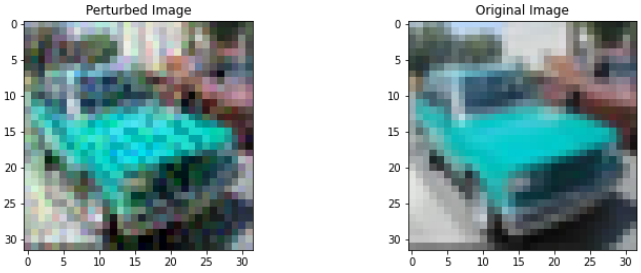
(a) True class Truck, predicted output Frog



(b) True class Airplane, predicted output Bird



(c) True class Airplane, predicted output Deer



(d) True class Automobile, predicted output Frog

Figure 6: Universally computed adversarial examples and original images