ECE 595: Final project(Spring 2020)

Frank Lin (ID:54)

Exercise 1

(a) We study the minimum-distance attack

(i) Consider general l_2-distance attack with linear decision boundary as the problem

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2 \quad \text{subject to} \quad \boldsymbol{w}^T\boldsymbol{x}+w_0=\boldsymbol{0}$$

by Lagrange multiplier such that the solution is

$$\boldsymbol{x}^* = \boldsymbol{x}_0 - \left(\frac{\boldsymbol{w}^T\boldsymbol{x}_0+w_0}{\|\boldsymbol{w}\|_2^2}\right)\boldsymbol{w} \tag{1}$$

we obtain

$$\begin{aligned}\boldsymbol{x}^* &= \boldsymbol{x}_0 - \left(\frac{\boldsymbol{w}^T\boldsymbol{x}_0+w_0}{\|\boldsymbol{w}\|_2^2}\right)\boldsymbol{w}\\ &= \boldsymbol{x}_0 - \left(\frac{(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t)^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_0 - \frac{1}{2}(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j+\boldsymbol{\mu}_t)+\log(\pi_j/\pi_t)}{\left\|\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t)\right\|_2^2}\right)\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t)\\ &= \boldsymbol{x}_0 - \left(\frac{(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t)^T\boldsymbol{\Sigma}^{-1}[\boldsymbol{x}_0 - \frac{1}{2}(\boldsymbol{\mu}_j+\boldsymbol{\mu}_t)]+\log(\pi_j/\pi_t)}{\left\|\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t)\right\|_2^2}\right)\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t).\end{aligned}$$

Consider general l_∞-distance attack with linear decision boundary as the problem

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{x}-\boldsymbol{x}_0\|_\infty \quad \text{subject to} \quad \boldsymbol{w}^T\boldsymbol{x}+w_0=\boldsymbol{0}$$

we let $\boldsymbol{r}=\boldsymbol{x}-\boldsymbol{x}_0$ and $b_0=-(\boldsymbol{w}^T\boldsymbol{x}_0+w_0)$ to form the equivalent problem: minimize$_{\boldsymbol{r}}\|\boldsymbol{r}\|_\infty$ subject to $\boldsymbol{w}^T\boldsymbol{r}=b_0$. We then apply Holder's inequality to have $|b_0|=|\boldsymbol{w}^T\boldsymbol{r}|\le\|\boldsymbol{w}\|_1\|\boldsymbol{r}\|_\infty$ such that the optimal solution is obtained as the equality holds, i.e.,

$$\boldsymbol{r}^* = \frac{b_0}{\|\boldsymbol{w}\|_1}\operatorname{sign}(\boldsymbol{w}) \quad \Rightarrow \quad \boldsymbol{x}^* = \boldsymbol{x}_0 - \left(\frac{\boldsymbol{w}^T\boldsymbol{x}_0+w_0}{\|\boldsymbol{w}\|_1}\right)\operatorname{sign}(\boldsymbol{w}). \tag{2}$$

We substitute into (2) to obtain

$$\begin{aligned}\boldsymbol{x}^* &= \boldsymbol{x}_0 - \left(\frac{\boldsymbol{w}^T\boldsymbol{x}_0+w_0}{\|\boldsymbol{w}\|_1}\right)\operatorname{sign}(\boldsymbol{w})\\ &= \boldsymbol{x}_0 - \left(\frac{(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t)^T\boldsymbol{\Sigma}^{-1}[\boldsymbol{x}_0 - \frac{1}{2}(\boldsymbol{\mu}_j+\boldsymbol{\mu}_t)]+\log(\pi_j/\pi_t)}{\left\|\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t)\right\|_1}\right)\operatorname{sign}\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j-\boldsymbol{\mu}_t)\right).\end{aligned}$$

(ii) Given the quadratic decision boundary $g(\boldsymbol{x})$, we have the gradient

$$\nabla g(\boldsymbol{x}) = (\boldsymbol{W}_j-\boldsymbol{W}_t)\boldsymbol{x}+(\boldsymbol{w}_j-\boldsymbol{w}_t)$$

such that the iterates of solving the DeepFool attack is

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \frac{g(\boldsymbol{x}^{(k)})}{\left\|\nabla g(\boldsymbol{x}^{(k)})\right\|_2^2}\nabla g(\boldsymbol{x}^{(k)}). \tag{3}$$

(iii) Consider a decision boundary $g(x) = 1/x$ such that $dg(x)/dx = -1/x^2$. If we apply the decision boundary to the iterates of DeepFool attack, we have

$$x^{(k+1)} = x^{(k)} - \frac{1/x^{(k)}}{1/(x^{(k)})^4}\left(-\frac{1}{(x^{(k)})^2}\right) = 2x^{(k)},$$

which obviously never converge. The intuitive reason is that the decision boundary has zero $g(x) = 1/x = 0$ when $x \to \infty$.

(b) Then we study the maximum-loss attacks

(i) Consider the maximum loss l_∞ attack of linear case as

$$\underset{\boldsymbol{x}}{\operatorname{argmin}}\, \boldsymbol{w}^T\boldsymbol{x} + w_0 \quad \text{subject to} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|_\infty < \eta. \tag{4}$$

Let $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{r}$ and $b_0 = \boldsymbol{w}^T\boldsymbol{x}_0 + w_0$. Then the original problem (4) becomes

$$\underset{\boldsymbol{r}}{\operatorname{argmin}}\, \boldsymbol{w}^T\boldsymbol{x} + w_0 = \boldsymbol{w}^T(\boldsymbol{x}_0 + \boldsymbol{r}) + w_0 = \boldsymbol{w}^T\boldsymbol{r} + b_0 \quad \text{subject to} \quad \|\boldsymbol{r}\|_\infty < \eta. \tag{5}$$

We apply the Holder's inequality

$$\boldsymbol{w}^T\boldsymbol{r} \geq -\|\boldsymbol{r}\|_\infty \|\boldsymbol{w}\|_1 \geq -\eta \|\boldsymbol{w}\|_1$$

where the equality holds when $\boldsymbol{r} = -\eta \cdot \operatorname{sign}(\boldsymbol{w})$ such that the optimal solution of (5) is $\boldsymbol{r} = -\eta \cdot \operatorname{sign}(\boldsymbol{w})$ and hence the optimal solution of (4) is $\boldsymbol{x} = \boldsymbol{x}_0 - \eta\, \operatorname{sign}(\boldsymbol{w})$. Replacing $\boldsymbol{w}$, we then have $\boldsymbol{x} = \boldsymbol{x}_0 - \eta\, \operatorname{sign}(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_t))$.

(ii) Consider the FGSM attack with Gaussian decision boundary $J(\boldsymbol{x}) = -g(\boldsymbol{x})$, we have $\nabla_{\boldsymbol{x}} J(\boldsymbol{x}) = -(\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x} - (\boldsymbol{w}_j - \boldsymbol{w}_t)$ and hence formula becomes

$$\boldsymbol{x}^* = \boldsymbol{x}_0 + \eta\, \operatorname{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0)) = \boldsymbol{x}_0 - \eta\, \operatorname{sign}((\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x}_0 + (\boldsymbol{w}_j - \boldsymbol{w}_t)).$$

(iii) Consider the FGSM attack with Gaussian decision boundary $J(\boldsymbol{x}) = -g(\boldsymbol{x})$, we have $\nabla_{\boldsymbol{x}} J(\boldsymbol{x}) = -(\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x} - (\boldsymbol{w}_j - \boldsymbol{w}_t)$ and hence formula becomes

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \mathcal{P}_{[0,1]}\left\{\boldsymbol{x}_0 + \eta\, \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}^{(k)})\right)\right\} \\ &= \mathcal{P}_{[0,1]}\left\{\boldsymbol{x}_0 + \eta\, \operatorname{sign}\left((\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x}^{(k)} + (\boldsymbol{w}_j - \boldsymbol{w}_t)\right)\right\}\end{aligned}$$

(c) We finally examine the regularization based attacks

(i) Consider the regularization-based attack in the linear case as

$$\underset{\boldsymbol{x}}{\operatorname{argmin}}\, \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 + \lambda(\boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{w}_0), \tag{6}$$

we apply the first-order necessary condition to have

$$\frac{\partial}{\partial \boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 + \lambda(\boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{w}_0) = \boldsymbol{x} - \boldsymbol{x}_0 + \lambda\boldsymbol{w} = \boldsymbol{0}$$

such that the optimal solution is

$$\boldsymbol{x}^* = \boldsymbol{x}_0 - \lambda\boldsymbol{w} = \boldsymbol{x}_0 - \lambda\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_t).$$

(ii) Consider the Carlini-Wagner attack with Gaussian decision boundary as

$$\underset{\boldsymbol{x}}{\operatorname{argmin}}\,\varphi(\boldsymbol{x}) \quad \varphi(\boldsymbol{x}) = \|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2 + \lambda \cdot \max(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}), 0) = \|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2 + \lambda \cdot \max(-g(\boldsymbol{x}), 0).$$

where $g(\boldsymbol{x}) = g_t(\boldsymbol{x}) - g_j(\boldsymbol{x})$. Since $\nabla_{\boldsymbol{x}} g(\boldsymbol{x}) = (\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x} + (\boldsymbol{w}_j - \boldsymbol{w}_t)$, we have the gradient

$$\begin{aligned}\nabla_{\boldsymbol{x}}\varphi(\boldsymbol{x}) &= 2(\boldsymbol{x}-\boldsymbol{x}_0) - \lambda \cdot \mathbb{I}_{g(\boldsymbol{x})<0} \cdot \nabla_{\boldsymbol{x}} g(\boldsymbol{x}) \\ &= 2(\boldsymbol{x}-\boldsymbol{x}_0) - \lambda \cdot \mathbb{I}_{g(\boldsymbol{x})<0} \cdot [(\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x} + (\boldsymbol{w}_j - \boldsymbol{w}_t)] \\ &= 2(\boldsymbol{x}-\boldsymbol{x}_0) - \lambda \cdot \mathbb{I}_{g(\boldsymbol{x})<0} \cdot \left[\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j) - \boldsymbol{\Sigma}_t^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_t)\right].\end{aligned}$$

Therefore the gradient descent iterate is given by

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - \alpha_k \nabla_{\boldsymbol{x}}\varphi(\boldsymbol{x}^{(k)}) \\ &= \boldsymbol{x}^{(k)} - 2\alpha_k(\boldsymbol{x}^{(k)} - \boldsymbol{x}_0) + \alpha_k\lambda \cdot \mathbb{I}_{g(\boldsymbol{x}^{(k)})<0} \cdot \left[\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}^{(k)} - \boldsymbol{\mu}_j) - \boldsymbol{\Sigma}_t^{-1}(\boldsymbol{x}^{(k)} - \boldsymbol{\mu}_t)\right]\end{aligned} \tag{7}$$

Exercise 2

(a) We solve the CW optimization problem by gradient descent obtained in (7) with the regularization coefficient $\lambda = 5$.

```
def gradient_nonoverlapping(pert_vec, img_vec, target_index, lam):
    # Calculate g_j, g_t, determine if patch_vec is already in target class
    grad = 0
    tc = 1-2*target_index
    tg = 2*target_index-1
    g_cat = dsc_fun(pert_vec, mean_cat, inv_cat, res_cat)
    g_grass = dsc_fun(pert_vec, mean_grass, inv_grass, res_grass)
    if (tc*g_cat+tg*g_grass > 0):
        grad = 2*(pert_vec-img_vec)-lam*(tc*inv_cat*(pert_vec-mean_cat)+tg*inv_grass*(pert
    return grad
```

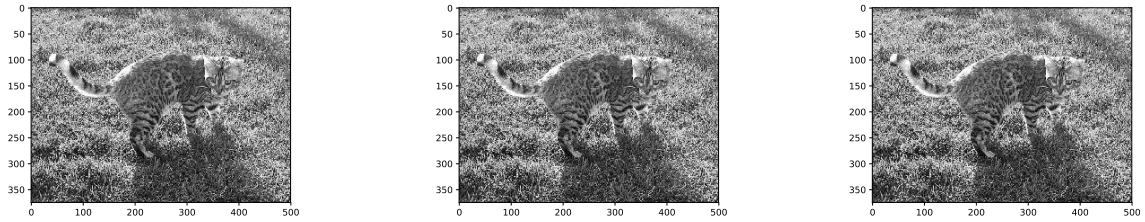
(b) We implement the gradient descent algorithm on every non-overlapping patch in the image. The step size is set as $\alpha = 0.0001$ and the stopping criterion is the maximum number of iterations 300 or the norm $\|\boldsymbol{X}^{(k+1)} - \boldsymbol{X}^{(k)}\|_2 < 0.001$.

```
def CW_attack_nonoverlapping(img_matrix, target_index, alpha, lam):
    # Start gradient descent
    while itr_num <= 300 and change >= 0.001:
        pert_img_prev = pert_img_curr.copy()
        for i in range(M):
            for j in range(N):
                if i + 8 > M or j + 8 > N or i % 8 != 0 or j % 8 != 0:
                    continue
                # Perturb the patch with top left pixel (i,j) in img_matrix by
                x = img_matrix[i:i+8, j:j+8]
                z = pert_img_prev[i:i+8, j:j+8]
```

```
                img_vec = x.flatten('F').reshape((d,1))
                pert_vec = z.flatten('F').reshape((d,1))
                grad = gradient_nonoverlapping(pert_vec, img_vec, target_index, lam)
                pert_vec = np.clip(pert_vec - alpha*grad, 0.0, 1.0)
                pert_img_curr[i:i+8, j:j+8] = np.reshape(pert_vec, (8, 8), order = 'F')
        # Process the perturbed image matrix, display perturbed image, etc.
        change = np.linalg.norm(pert_img_curr - pert_img_prev)
        itr_num += 1
    print(itr_num)
    return pert_img_curr
```

(c) Consider $\lambda = 1, 5, 10$ we implement the attack as follows.

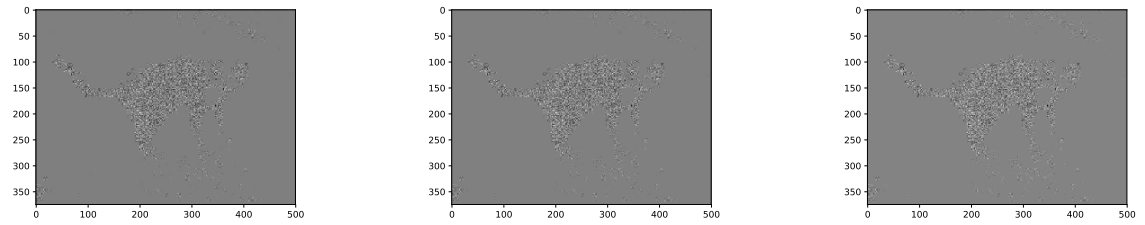
(i) The final perturbed images for $\lambda = 1, 5, 10$ and target index 0 are given by Fig. 1. The perturbed image looks very similar to the original image. It is hard to point out the differences by eyes.



(a) The perturbed image for $\lambda = 1$. (b) The perturbed image for $\lambda = 5$. (c) The perturbed image for $\lambda = 10$.

Figure 1: The final perturbed images for $\lambda = 1, 5, 10$, change cat to grass.

(ii) The perturbations added to the original images for $\lambda = 1, 5, 10$, change cat to grass are given in Fig. 2. It can be observed that if we aim to change cat into grass, we add perturbation on the cat part of the classifier's output of the original image.



(a) Added perturbation for $\lambda = 1$. (b) Added perturbation for $\lambda = 5$. (c) Added perturbation for $\lambda = 10$.

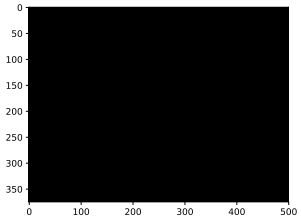
Figure 2: Added perturbation for $\lambda = 1, 5, 10$ change cat to grass.

(iii) The Frobenius norm of the perturbations for $\lambda = 1, 5, 10$, change cat to grass are given by Table 1. We can observe that when λ increases, we allow more distance (Frobenius norm) between the perturbed image and original image.

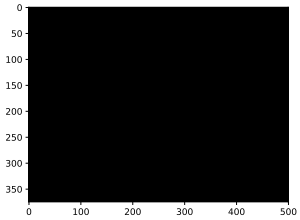
(iv) The classifier's output on the attack for $\lambda = 1, 5, 10$, change cat to grass are given in Fig. 3. It can be observed that after sufficient large (≤ 300) iterates, all the patches ($46 \times 62 = 2852$) of the perturbed image will be classified as grass.

Table 1: The Frobenius norm of the perturbations for $\lambda = 1, 5, 10$, change cat to grass.

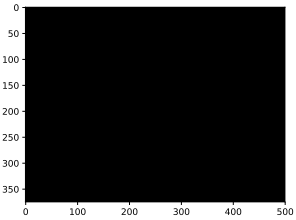
	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$
cat to grass	8.2470	8.9304	9.6988



(a) The classifier's output on the attack for $\lambda = 1$. Total $46 \times 62 = 2852$ patches classified as grass.



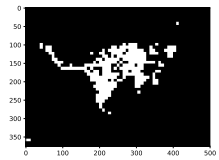
(b) The classifier's output on the attack for $\lambda = 5$. Total $46 \times 62 = 2852$ patches classified as grass.



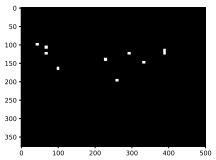
(c) The classifier's output on the attack for $\lambda = 10$. Total $46 \times 62 = 2852$ patches classified as grass.

Figure 3: The classifier's output on the attack for $\lambda = 1, 5, 10$, when change cat to grass.

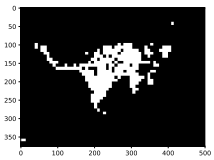
(v) The classifier's output on the perturbed images iterates per 50, 10, 5 iterations for $\lambda = 1, 5, 10$ when changing cat to grass are given by Fig. 4. We can observe that during the gradient descend algorithm, as the number of iterates increases, more and more patches of the perturbed image will be classified from cat to grass.



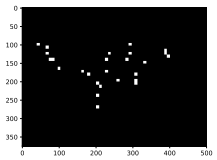
(a) The classifier's output on the perturbed images of 50 iterates for $\lambda = 1$.



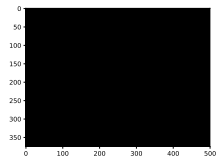
(b) The classifier's output on the perturbed images of 100 iterates for $\lambda = 1$.



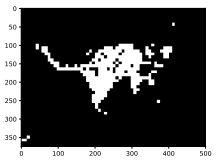
(c) The classifier's output on the perturbed images of 10 iterates for $\lambda = 5$.



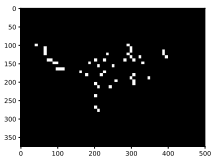
(d) The classifier's output on the perturbed images of 20 iterates for $\lambda = 5$.



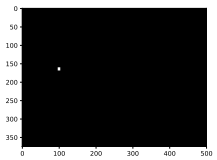
(e) The classifier's output on the perturbed images of 30 iterates for $\lambda = 5$.



(f) The classifier's output on the perturbed images of 5 iterates for $\lambda = 10$.



(g) The classifier's output on the perturbed images of 10 iterates for $\lambda = 10$.



(h) The classifier's output on the perturbed images of 15 iterates for $\lambda = 10$ and target index 0 (to black).

Figure 4: The classifier's output on the perturbed images iterates per 50, 10, 5 iterations for $\lambda = 1, 5, 10$, change cat to grass.

(d) When we decrease the step size α, we search from a smaller neighborhood of the original image such that we have smaller initial change but a lower convergence rate. With a smaller step size, the algorithm is more likely to converge.

Exercise 3

(a) Let $h(\boldsymbol{P}_i\boldsymbol{X}) = g_j(\boldsymbol{P}_i\boldsymbol{X}) - g_t(\boldsymbol{P}_i\boldsymbol{X})$ such that the gradient of $h(\boldsymbol{P}_i\boldsymbol{X})$ with respect to $\boldsymbol{X}$ is

$$\begin{aligned}\nabla_{\boldsymbol{X}} h(\boldsymbol{P}_i\boldsymbol{X}) &= \boldsymbol{P}_i^T \nabla_{\boldsymbol{x}} h(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{P}_i\boldsymbol{X}} = \boldsymbol{P}_i^T \left(\nabla_{\boldsymbol{x}} g_j(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} g_t(\boldsymbol{x})\right)_{\boldsymbol{x}=\boldsymbol{P}_i\boldsymbol{X}} \\ &= \boldsymbol{P}_i^T[(\boldsymbol{W}_t - \boldsymbol{W}_j)\boldsymbol{P}_i\boldsymbol{X} + (\boldsymbol{w}_t - \boldsymbol{w}_j)] = \boldsymbol{P}_i^T \left[\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{P}_i\boldsymbol{X} - \boldsymbol{\mu}_t) - \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{P}_i\boldsymbol{X} - \boldsymbol{\mu}_j)\right].\end{aligned}$$

Therefore the gradient with respect to $\boldsymbol{X}$ is

$$\nabla_{\boldsymbol{X}} \varphi(\boldsymbol{X}) = 2(\boldsymbol{X} - \boldsymbol{X}_0) + \lambda \sum_{i=1}^{L} \mathbb{I}_{h(\boldsymbol{P}_i\boldsymbol{X})>0} \cdot \boldsymbol{P}_i^T \left[\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{P}_i\boldsymbol{X} - \boldsymbol{\mu}_t) - \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{P}_i\boldsymbol{X} - \boldsymbol{\mu}_j)\right]$$

(b) We implement the gradient descent algorithm on the overlapping patch classifier with the iterates

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} - \alpha \nabla_{\boldsymbol{X}} \varphi(\boldsymbol{X}^{(k)}) = \boldsymbol{X}^{(k)} - 2\alpha(\boldsymbol{X}^{(k)} - \boldsymbol{X}_0) - \alpha\lambda \sum_{i=1}^{L} \mathbb{I}_{h(\boldsymbol{P}_i\boldsymbol{X}^{(k)})>0} \cdot \nabla_{\boldsymbol{X}} h(\boldsymbol{P}_i\boldsymbol{X}^{(k)})$$

The stopping criterion is the maximum number of iterations 300 or the norm $\|\boldsymbol{X}^{(k+1)} - \boldsymbol{X}^{(k)}\|_2 < 0.01$.

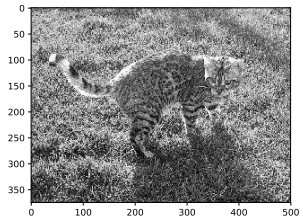
```
def CW_attack_overlapping(img_matrix, target_index, alpha, lam):
    # Start gradient descent
    while itr_num <= 300 and change >= 0.01:
        pert_img_prev = pert_img_curr.copy()
        pert_img_curr = pert_img_curr-2*alpha*(pert_img_prev-img_matrix)
        for i in range(M-8):
            for j in range(N-8):
                # Perturb the patch with top left pixel (i,j) in img_matrix by
                # pert_patch = patch_vec + alpha*grad
                z = pert_img_prev[i:i+8, j:j+8]
                pert_vec = z.flatten('F').reshape((d,1))
                grad = gradient_overlapping(pert_vec, target_index)
                pert_img_curr[i:i+8, j:j+8] += np.reshape(-alpha*lam*grad, (8, 8), order =
        # Process the perturbed image matrix, display perturbed image, etc.
        pert_img_curr = np.clip(pert_img_curr, 0.0, 1.0)
        change = np.linalg.norm(pert_img_curr - pert_img_prev)
        itr_num += 1
    return pert_img_curr
```

(c) Consider $\lambda = 0.5, 1, 5$ we implement the attack as follows.

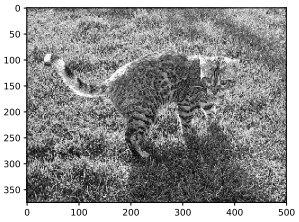
(i) The final perturbed images for $\lambda = 0.5, 1, 5$ and target index 0 are given by Fig. 5. The perturbed image looks very similar to the original image.

(ii) The perturbations added to the original images for $\lambda = 0.5, 1, 5$ when changing cat to grass are given in Fig. 6.

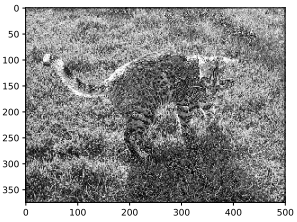
(iii) The Frobenius norm of the perturbations for $\lambda = 0.5, 1, 5$ when changing cat to grass are 14.0593, 16.1586, and 33.6171, respectively. We can observe that when λ increases, we allow more distance (Frobenius norm) between the perturbed image and original image.



(a) The perturbed image for $\lambda = 0.5$.

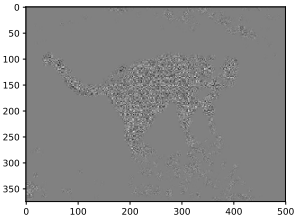


(b) The perturbed image for $\lambda = 1$.

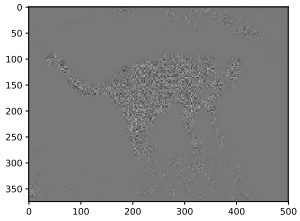


(c) The perturbed image for $\lambda = 5$.

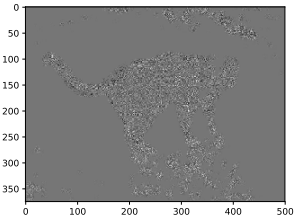
Figure 5: The final perturbed images for $\lambda = 0.5, 1, 5$, change cat to grass.



(a) Added perturbation for $\lambda = 0.5$.



(b) Added perturbation for $\lambda = 1$.



(c) Added perturbation for $\lambda = 5$.

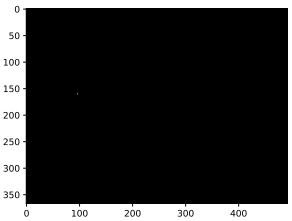
Figure 6: Added perturbation for $\lambda = 0.5, 1, 5$, change cat to grass.

(iv) The classifier's output on the attack for $\lambda = 0.5, 1, 5$ when changing cat to grass are given in Fig. 7. All the $367 \times 492 = 180564$ patches of the perturbed image will be classified as grass.

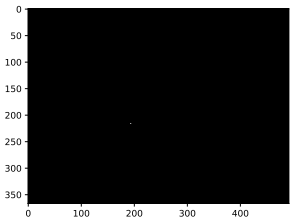
(v) The classifier's output on the perturbed images iterates per 5, 5, 2 iterations for $\lambda = 0.5, 1, 5$ when changing cat to grass are given by Fig. 8, Fig. 9, and Fig. 10, respectively. As iterations continues, more patches of the perturbed image will be classified to grass.

Exercise 4

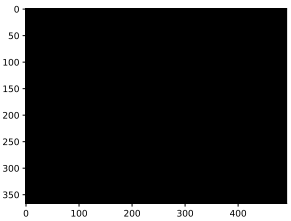
- **Method Description**
  To prevent attackers from launching an adversarial attack to effect the performance of an classifier, we develop a training data preprocessing method to strengthen the data robustness against any malicious data modifications. Different from previous works, instead of trying to



(a) The classifier's output on the attack for $\lambda = 0.5$.

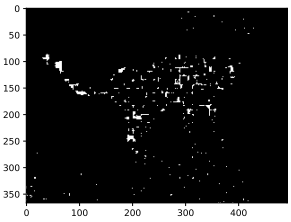


(b) The classifier's output on the attack for $\lambda = 1$.

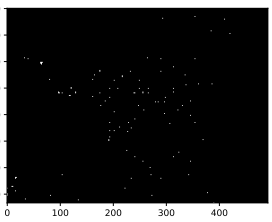


(c) The classifier's output on the attack for $\lambda = 5$.

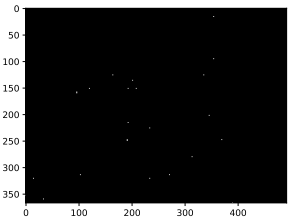
Figure 7: The classifier's output on the attack for $\lambda = 0.5, 1, 5$.



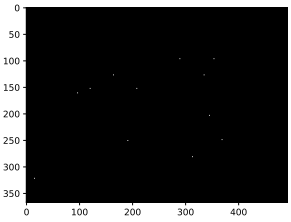
(a) The classifier's output on the perturbed images of 5 iterates for $\lambda = 0.5$.



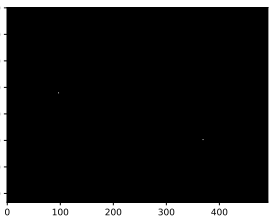
(b) The classifier's output on the perturbed images of 10 iterates for $\lambda = 0.5$.



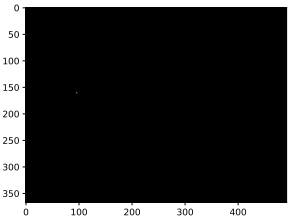
(c) The classifier's output on the perturbed images of 15 iterates for $\lambda = 0.5$.



(d) The classifier's output on the perturbed images of 20 iterates for $\lambda = 0.5$.

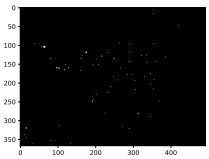


(e) The classifier's output on the perturbed images of 25 iterates for $\lambda = 0.5$.

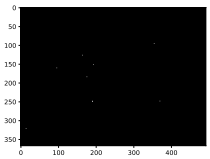


(f) The classifier's output on the perturbed images of 30 iterates for $\lambda = 0.5$.

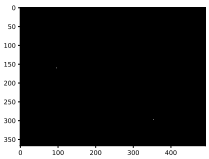
Figure 8: The classifier's output on the perturbed images per 5 iterates for $\lambda = 0.5$, change cat to grass.



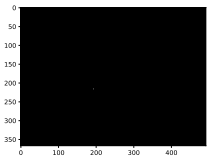
(a) The classifier's output on the perturbed images of 5 iterates for $\lambda = 1$.



(b) The classifier's output on the perturbed images of 10 iterates for $\lambda = 1$.

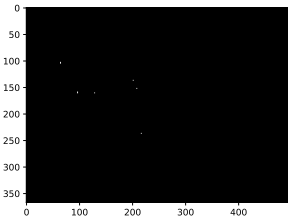


(c) The classifier's output on the perturbed images of 15 iterates for $\lambda = 1$.

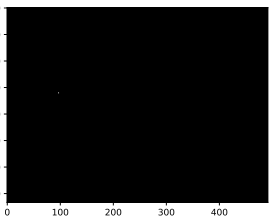


(d) The classifier's output on the perturbed images of 20 iterates for $\lambda = 1$.

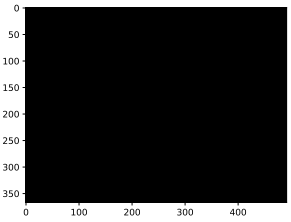
Figure 9: The classifier's output on the perturbed images per 5 iterates for $\lambda = 1$, change cat to grass.



(a) The classifier's output on the perturbed images of 2 iterates for $\lambda = 5$.



(b) The classifier's output on the perturbed images of 4 iterates for $\lambda = 5$.



(c) The classifier's output on the perturbed images of 6 iterates for $\lambda = 5$.

Figure 10: The classifier's output on the perturbed images per 5 iterates for $\lambda = 1$, change cat to grass.

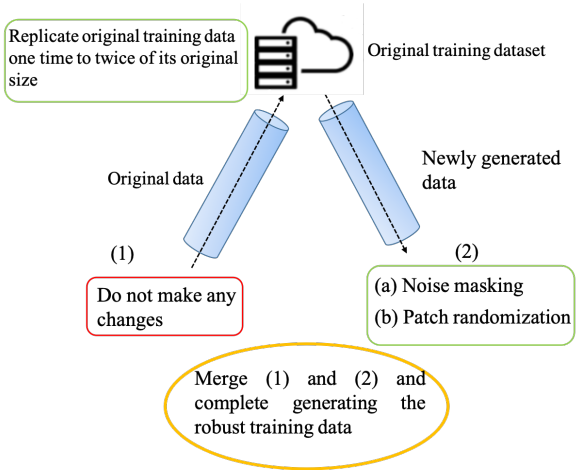


Figure 11: The noise patch-randomization scheme.

recover the damaged data after an attack, we aim to explore a way to effectively intensify the robustness of training data such that the effect of an attack on learning performance in minimized before one is launched.

Consider that the CW attack is a soft-version of the minimum-distance attack, it inherits the idea that, given an input data point x_0, the attackers want to send x_0 to a targeted class $\mathcal{C}_t$ by heading along a specific direction towards and across the decision boundary. In order to effectively prevent this attack, we want to enlarge the minimum distance between x_0 and any point x in the targeted class $\mathcal{C}_t$ so that the probability for x_0 to be moved to $\mathcal{C}_t$ can be guaranteed to decrease. Therefore, we need to investigate the characteristics of the decision boundary for a Gaussian classifier.

Since we know that the decision boundary of a Gaussian classifier is nonlinear when the covariance matrix of data in different groups is nonequivalent, we can expect that the decision boundary becomes more similar to a linear decision boundary and moves away from any input x_0 as the amount of training data becomes larger. One of the way to do so is by image augmentation. However, it is very important that we preserve the features of data as we increase the training samples. Therefore, we developed a two-step noise patch-randomization scheme to effectively augment the training image as depicted in Fig.11. In the first step, a small randomness of zero mean in added to every pixel of the image. In the second step, a larger but feature-preservative randomness is added to every patch in the image. We first replicate the original data samples (e.g. cat and grass) one time then preform the noise patch-randomization scheme on the newly generated data after replication as follows:

(a) **Noise masking**
A zero-mean Gaussian white noise with a standard deviation of 0.2 is added to the image patches for each epoch, so that each epoch uses a slightly different data set.

(b) **Patch randomization**
Every pixel in the same patch is randomly re-ordered. By doing so, we preserves the feature of the data while provide a large randomness to newly generated training data samples.

- **Comparisons**
To show that our scheme provides a more robust training dataset, we train the classifier

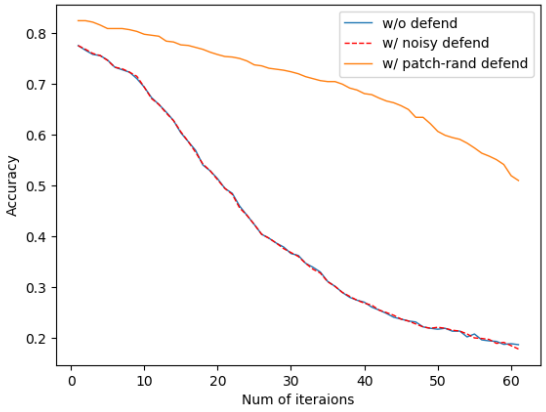


Figure 12: Accuracy w.r.t various defending strategies against CW attack.

(i) with dataset generated by the noise patch-randomization scheme and compare it with the classifier (ii) trained with the original dataset. After showing the robustness of our generated dataset against the original data, we want to further demonstrate the power of patch-randomization. Therefore, we train a classifier (iii) with dataset preprocessed only with noise masking and compare it with classifier (i), which dataset are preprocessed with both noise masking and patch randomization. We compare the performance of the schemes based on accuracy.

The result is demonstrated in Fig.12 with $\alpha = 0.001$ and $\lambda = 0.007$. There are two observations worth noticing. First, it can be observed that the accuracy of our proposed scheme is higher than the other two before the attack is launched. This shows that the patch randomization technique efficiently enlarges the data variability so that the classifier trained with these data has better performance. Second, the decrease rate of accuracy of our proposed scheme is much lower (approximately one-half) than the other two. After 60 iterations of attack, the accuracy obtained by our approach is 50% while the accuracy obtained by the other two remains only 20%. Therefore, we demonstrate the robustness and effectiveness of training data preprocessing of our noise patch-randomization scheme.

- **Limitations**
  In our analysis, the attacker has all the information to our model and data so that they can correctly compute the gradient (White box attack). Even though the classifier generated by our method are robust to the attack compared to the classifier generated from the original model, the attacker can still successfully modify the class of a particular data when enough iterations of attack is performed. For black box attack, when the attacker does not have access to the model parameters, the attacker can still approximate the gradient information by averaging over the data using the its distribution. However, since the patches for every image is highly randomized, we expect that the average approximation of gradient may not be a good estimate such that it takes many iterations for the attacker to finish the update. However, under that condition, the perturbation of the image may be quite obvious. Therefore, this idea of designing a robust training data can be extended to nonlinear neural-network-based classifiers for that the gradient information becomes less useful compared to linear models.