ECE 595: Project

Sayeed Shafayet Chowdhury, # 19, Collaborator: Nathan Mensah, #5

(Spring 2020)

Exercise 1

(a)(i) l_2 attack - $\underset{x}{\text{argmin}}\ ||x-x_0||_2^2$, subject to $w^Tx+w_0=0$; $w^Tx+w_0=0$ is a line, we need to find a point x on this line closest to x_0. $\mathcal{L}(x,\lambda)=.5||x-x_0||_2^2-\lambda(w^Tx+w_0)$; $\nabla_x\mathcal{L}=(x-x_0)-\lambda w=0$; $\nabla_\lambda\mathcal{L}=w^Tx+w_0=0$, so, $x=x_0+\lambda w \implies w^Tx+w_0=w^T(x_0+\lambda w)+w_0 \implies 0=w^Tx_0+\lambda||w||^2+w_0 \implies \lambda=\frac{-(w^Tx_0+w_0)}{||w||^2}$; so, $x^*=x_0-\frac{w^Tx_0+w_0}{||w||_2^2}w$, where w and w_0 are given by Eqn. (4) of given prompt.

l_∞ **attack -** $\underset{x}{\text{argmin}}\ ||x-x_0||_\infty$, subject to $w^Tx+w_0=0$; let $r=x-x_0, b_0=-(w^Tx+w_0)$, so the problem becomes $\underset{r}{\text{argmin}}\ ||r||_\infty$, subject to $w^Tr=b_0$; Now using Holder's inequality with p=1 and $q=\infty$, we get $|x^Ty|\le||x||_1||y||_\infty$, so $|b_0|=|w^Tr|\le||w||_1||r||_\infty \implies ||r||_\infty\ge\frac{|b_0|}{||w||_1}$, let us consider $r=\eta sign(w)$, for some const. η. Now $||r||_\infty=\max_i|\eta sign(w_i)|=|\eta|$, so if we let $\eta=\frac{b_0}{||w||_1}$, then we have $||r||_\infty=|\eta|=\frac{|b_0|}{||w||_1}$. So, the lower bound is achieved and soln is $r=\frac{|b_0|}{||w||_1}sign(w)$. So, $x^*=x_0-\frac{w^Tx_0+w_0}{|w||_1}sign(w)$, where w and w_0 are given by Eqn. (4) of given prompt.

(ii) $\underset{x}{\text{argmin}}\ ||x-x_0||_2^2$, subject to $g(x)=0$; To a 1st order approximation, $g(x)=g(x^{(k)})+\nabla_xg(x^{(k)})^T(x-x^{(k)})$, so assuming $x^{(0)}=x_0$, we modify the problem as, $x^{(k+1)}=\underset{x}{\text{argmin}}\ ||x-x^{(k)}||^2$, subject to $g(x^{(k)})+\nabla_xg(x^{(k)})^T(x-x^{(k)})=0$, now we rewrite $g(x^{(k)})+\nabla_xg(x^{(k)})^T(x-x^{(k)})=\nabla_xg(x^{(k)})^Tx+g(x^{(k)})-\nabla_xg(x^{(k)})^Tx^{(k)}$. Let, $w^{(k)}=\nabla_xg(x^{(k)}), w_0^{(k)}=g(x^{(k)})-\nabla_xg(x^{(k)})^Tx^{(k)}$. So, the optimization problem is equivalent to, $x^{(k+1)}=\underset{x}{\text{argmin}}\ ||x-x^{(k)}||^2$ subject to $(w^{(k)})^Tx+w_0^{(k)}=0$, which is a linear problem! So, the solution is $x^{(k+1)}=x^{(k)}-\frac{(w^{(k)})^Tx^{(k)}+w_0^{(k)}}{||w^{(k)}||^2}w^{(k)}=x^{(k)}-\frac{g(x^{(k)})}{||\nabla_xg(x^{(k)})||^2}\nabla_xg(x^{(k)})$, here $g(x)=\frac{1}{2}x^T(W_j-W_t)x+(w_j-w_t)^Tx+(w_{j,0}-w_{t,0})$, where $W_i=\boldsymbol{\Sigma_i^{-1}}, w_i=-\boldsymbol{\Sigma_i^{-1}}\mu_i, w_{i,0}=\frac{1}{2}\mu_i\boldsymbol{\Sigma_i^{-1}}\mu_i+\frac{1}{2}log(|\boldsymbol{\Sigma}_i|)-log(\pi_i)$, so $\boldsymbol{\nabla_xg(x)=(W_j-W_t)x+(w_j-w_t)}$.

(iii) $g(x)=\sqrt{|x|}$ is one such decision boundary.

(b)(i) $\boldsymbol{l_\infty}$ maxloss attack - $\underset{x}{\text{argmin}}\ \boldsymbol{w^Tx+w_0}$, subject to $||\boldsymbol{x-x_0}||_\infty\le\eta$; Let, $\boldsymbol{x=x_0+r}$, then $\boldsymbol{w^Tx}+w_0=\boldsymbol{w^Tr+w^Tx_0}+w_0$ with $b_0=\boldsymbol{w^Tx_0}+w_0$, so the problem becomes $\underset{\boldsymbol{r}}{\text{argmin}}\ \boldsymbol{w^Tr}+b_0$, subject to $||\boldsymbol{r}||_\infty\le\eta$. From Holder's inequality (the negative side): $\boldsymbol{w^Tr}\ge-||\boldsymbol{w}||_\mathbf{1}||\boldsymbol{r}||_\infty\ge-\eta||\boldsymbol{w}||_\mathbf{1}$. We claim, the lower bound of $\boldsymbol{w^Tr}$ is attained when $\boldsymbol{r}=-\eta sign(\boldsymbol{w})$, so $\boldsymbol{w^Tr}=-\eta\boldsymbol{w^T}sign(\boldsymbol{w})=-\eta\sum_{i=1}^d w_isign(w_i)=-\eta\sum_{i=1}^d|w_i|=-\eta||\boldsymbol{w}||_\mathbf{1}$. So the soln is, $\boldsymbol{r}=-\eta sign(\boldsymbol{w})$. So, $\boldsymbol{x^*=x_0-\eta sign(w)}$.

(ii) The loss is $J(\boldsymbol{x,w})=g_t(\boldsymbol{x})-g_j(\boldsymbol{x})$, then max-loss attack is $\underset{x}{\text{maximize}}\ J(\boldsymbol{x,w})$, subject to $||\boldsymbol{x-x_0}||_\infty\le\eta$; We linearize by $J(\boldsymbol{x;w})=J(\boldsymbol{x_0+r;w})\approx J(\boldsymbol{x_0;w})+\nabla_xJ(\boldsymbol{x_0;w})^\boldsymbol{T}\boldsymbol{r}$. Then the problem becomes $\underset{r}{\text{maximize}}\ J(\boldsymbol{x_0;w})+\nabla_xJ(\boldsymbol{x_0;w})^\boldsymbol{T}\boldsymbol{r}$, subject to $||\boldsymbol{r}||_\infty\le\eta$, which is equivalent to $\underset{r}{\text{minimize}}\ -J(\boldsymbol{x_0;w})-\nabla_xJ(\boldsymbol{x_0;w})^\boldsymbol{T}\boldsymbol{r}$, subject to $||\boldsymbol{r}||_\infty\le\eta$. Comparing this with $\underset{r}{\text{minimize}}\ (\boldsymbol{w^Tr})+\boldsymbol{w_0}$, subject to $||\boldsymbol{r}||_\infty\le\eta$ and using result from part (i) above, we get, $\boldsymbol{r}=-\eta sign(-\nabla_xJ(\boldsymbol{x_0;w}))$. So, $\boldsymbol{x=x_0}-\eta sign(-\nabla_xJ(\boldsymbol{x_0;w}))=\boldsymbol{x_0}+\eta sign(\nabla_xJ(\boldsymbol{x_0;w}))$. Now, $J(\boldsymbol{x})=-g(\boldsymbol{x})$, so, $\boldsymbol{\nabla_xJ(x)=(W_t-W_j)x+(w_t-w_j)}$.

Therefore, $\boldsymbol{x} = \boldsymbol{x_0} + \eta sign((\boldsymbol{W_t} - \boldsymbol{W_j})\boldsymbol{x_0} + (\boldsymbol{w_t} - \boldsymbol{w_j}))$, where $\boldsymbol{W_i}$s and $\boldsymbol{w_i}s$ are given by Eqn. (2) of given prompt.

(iii) IFGSM attack We write the following eqn from above- $J(\boldsymbol{x}; \boldsymbol{w}) = J(\boldsymbol{x_0} + \boldsymbol{r}; \boldsymbol{w}) \approx J(\boldsymbol{x_0}; \boldsymbol{w}) + \nabla_x J(\boldsymbol{x_0}; \boldsymbol{w})^{\boldsymbol{T}} \boldsymbol{r} = J(\boldsymbol{x_0}; \boldsymbol{w}) + \nabla_x J(\boldsymbol{x_0}; \boldsymbol{w})^{\boldsymbol{T}} (\boldsymbol{x} - \boldsymbol{x_0}) = J(\boldsymbol{x_0}; \boldsymbol{w}) + \nabla_x J(\boldsymbol{x_0}; \boldsymbol{w})^{\boldsymbol{T}} \boldsymbol{x} - \nabla_x J(\boldsymbol{x_0}; \boldsymbol{w})^{\boldsymbol{T}} \boldsymbol{x_0}$. Now, let us consider the problem, $\underset{x}{\text{maximize}}\ J(\boldsymbol{x_0}; \boldsymbol{w}) + \nabla_x J(\boldsymbol{x_0}; \boldsymbol{w})^{\boldsymbol{T}} \boldsymbol{x} - \nabla_x J(\boldsymbol{x_0}; \boldsymbol{w})^{\boldsymbol{T}} \boldsymbol{x_0}$, subject to $||\boldsymbol{x} - \boldsymbol{x_0}||_\infty \leq \eta, 0 \leq \boldsymbol{x} \leq 1$, the 1st and 3rd term are irrelevant in terms of optimization wrt x, so we introduce iterative linearization problem as, $\boldsymbol{x}^{(k+1)} = \underset{x}{\text{argmax}}\ \nabla_x J(\boldsymbol{x_0}; \boldsymbol{w})^{\boldsymbol{T}} \boldsymbol{x}$, subject to $||\boldsymbol{x} - \boldsymbol{x}^{(k)}||_\infty \leq \eta, 0 \leq \boldsymbol{x} \leq 1$. Using the same soln procedure as (ii), we get, $\boldsymbol{x}^{(k+1)} = \mathcal{P}_{[0,1]}\{\boldsymbol{x}^{(k)} + \eta \cdot sign(\nabla_x J(\boldsymbol{x}^{(\boldsymbol{k})}; \boldsymbol{w}))\}$, where $\nabla_x J(\boldsymbol{x}^{(\boldsymbol{k})}, \boldsymbol{w}) = (\boldsymbol{W_t} - \boldsymbol{W_j})\boldsymbol{x}^{(\boldsymbol{k})} + (\boldsymbol{w_t} - \boldsymbol{w_j})$ and $\boldsymbol{W_i}$s and $\boldsymbol{w_i}s$ are given by Eqn. (2) of given prompt.

(c)(i) The optimization problem can be written as $\underset{x}{\text{argmin}}\ J(x) = \underset{x}{\text{argmin}}\ \frac{1}{2}||\boldsymbol{x} - \boldsymbol{x_0}||^2 + \lambda(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x} + w_0)$. Then, $\nabla_x J(x) = 0 \implies (\boldsymbol{x} - \boldsymbol{x_0}) + \lambda \boldsymbol{w} = 0 \overset{x}{\implies} \boldsymbol{x^*} = \boldsymbol{x_0} - \lambda \boldsymbol{w}$, where w is given by Eqn. (4) of given prompt.

(ii) CW attack Let us consider, $\underset{x}{\text{argmin}}\ ||\boldsymbol{x} - \boldsymbol{x_0}||_1 + \lambda(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x} + w_0)$, which is equivalent to $\underset{r}{\text{argmin}}\ ||\boldsymbol{r}||_1 + \lambda \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{r}$. Here, the optimality condition is (sort of): $sign(r_i) + \lambda w_i = 0$. This requires that-

$$\lambda w_i = \begin{cases} \pm 1, & |r_i| 0 \\ \in (-1, 1), & |r_i| = 0 \end{cases}$$

So, $|\lambda w_i|$ will never exceed 1. Therefore, if $|\lambda w_i| > 1$, the optimization does not have a solution regardless of how we choose r. Note, the function $f(x) = |x| + \alpha x$ goes to $-\infty$ as x $\to -\infty$ if $\alpha > 1$ and goes to $-\infty$ as x $\to +\infty$ if $\alpha < -1$, so it is unbounded below. So the idea of Carlini-Wagner attack is to solve $\underset{x}{\text{argmin}}\ ||\boldsymbol{x} - \boldsymbol{x_0}|| + \lambda \cdot max\{(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x})), 0\}$. The justification is if $g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}) < 0$, the sample is already misclassified, no action is needed. Only if $g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}) > 0$, the sample is not yet misclassified, so action is needed. We use the rectifier function, $\zeta(x) = max(x, 0)$. So the problem can be written as $\underset{x}{\text{argmin}}\ ||\boldsymbol{x} - \boldsymbol{x_0}|| + \lambda \cdot \zeta(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}))$. Now consider the CW attack: $\underset{x}{\text{argmin}}\ ||\boldsymbol{x} - \boldsymbol{x_0}|| + \lambda \cdot max(\frac{1}{2}\boldsymbol{x}^{\boldsymbol{T}}(\boldsymbol{W_j} - \boldsymbol{W_t})\boldsymbol{x} + (\boldsymbol{w_j} - \boldsymbol{w_t})^{\boldsymbol{T}} \boldsymbol{x} + (w_{j,0} - w_{t,0}), 0)$. The objective function is always non-negative: $||\boldsymbol{x} - \boldsymbol{x_0}||_1 \geq 0$ and $\zeta(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x})) \geq 0$. So,we are guaranteed to have a solution.

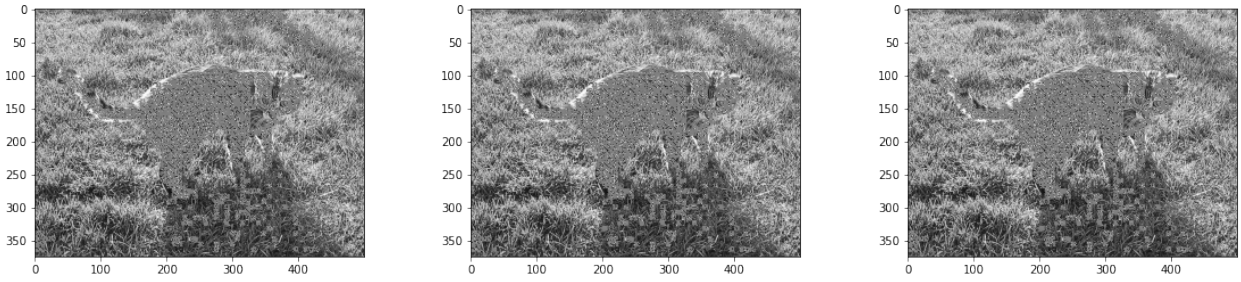
The function $h(x) = max(\phi(x), 0)$ is convex in x if $\phi(x)$ is convex. Now, $h(\alpha x + (1-\alpha)y) = max(\phi(\alpha x + (1-\alpha)y), 0) \leq max(\alpha\phi(x) + (1-\alpha)\phi(y), 0) \leq \alpha max(\phi(x), 0) + (1-\alpha)max(\phi(y), 0) = \alpha h(x) + (1-\alpha)h(y)$. In our case, $\phi(x) = \frac{1}{2}\boldsymbol{x}^{\boldsymbol{T}}(\boldsymbol{W_j} - \boldsymbol{W_t})\boldsymbol{x} + (\boldsymbol{w_j} - \boldsymbol{w_t})^{\boldsymbol{T}} \boldsymbol{x} + (w_{j,0} - w_{t,0})$, here 2nd term is linear in x, 3rd term doesn't have x, hessian of 1st term is $(W_j - W_t)$, now since the covariance matrix $\boldsymbol{\Sigma}_i$ is PSD, so for any non-zero z, $\boldsymbol{z}^{\boldsymbol{T}}(\boldsymbol{W_j} - \boldsymbol{W_t})\boldsymbol{z} \geq 0$, so $\phi(x)$ is convex. So the overall optimization is convex. In general, CW attack solves $\underset{x}{\text{argmin}}\ ||\boldsymbol{x} - \boldsymbol{x_0}||^2 + \lambda \cdot \zeta(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}))$. Now, the gradient of $\zeta(\cdot)$ is $\frac{d}{ds}\zeta(s) = \mathbb{1}\{s > 0\}$. Thus, $\nabla_x \zeta(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x})) = \mathbb{1}\{g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}) > 0\} \cdot (\nabla_x g_j(\boldsymbol{x}) - \nabla_x g_t(\boldsymbol{x}))$. Now letting $\varphi(x)$ be the overall objective function, $\varphi(x) = \underset{x}{\text{argmin}}\ ||\boldsymbol{x} - \boldsymbol{x_0}||^2 + \lambda \cdot \zeta(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}))$. So the gradient is,

$$\nabla_x \varphi(x) = 2(\boldsymbol{x} - \boldsymbol{x_0}) + \lambda \cdot \mathbb{1}\{g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}) > 0\} \cdot (\nabla_x g_j(\boldsymbol{x}) - \nabla_x g_t(\boldsymbol{x}))$$

. So, CW attack is performed iteratively using the update rule as- For iteration k = 1, 2, ..., $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \nabla_x \varphi(\boldsymbol{x}^{(k)})$, where α_k is gradient descent step size and $\nabla_x \varphi(\boldsymbol{x}^{(k)})$ is given by above equation. Also, $\nabla_x(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x})) = (\boldsymbol{W_j} - \boldsymbol{W_t})\boldsymbol{x} + (\boldsymbol{w_j} - \boldsymbol{w_t})$

Exercise 2

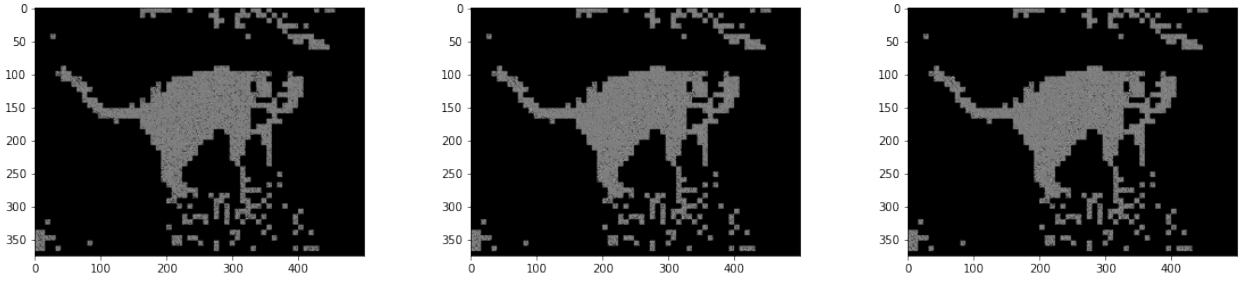
(a) and (b) We implemented the functions assuming the target is to make cat class turned into grass class (to fool the classifier to think the cat patches are grass).
(c)(i) The images are shown in Fig. 1.



(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 10$

Figure 1: The attack, i.e. the final perturbed image

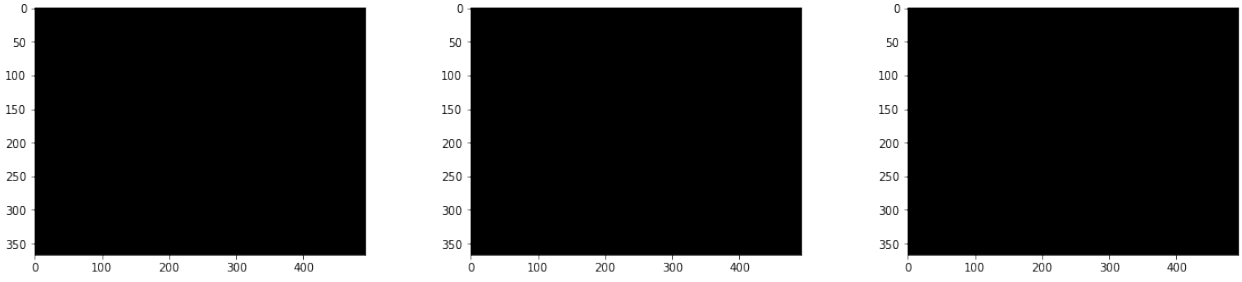
(ii) The perturbations are shown in Fig. 2.



(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 10$

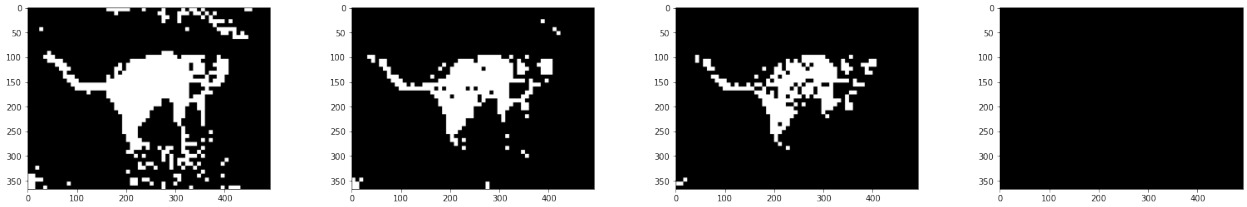
Figure 2: The perturbation added to the original image to obtain the attack

(iii) The frobenius norm of the perturbation for $\lambda = 1, 5, 10$ are 98.8869, 100.7766 and 100.7636 respectively.
(iv) The classifier's output on the final attack images are shown in Fig. 3. We see for all 3 λs, the classifier is completely fooled by the attack, all patches are classified as grass.



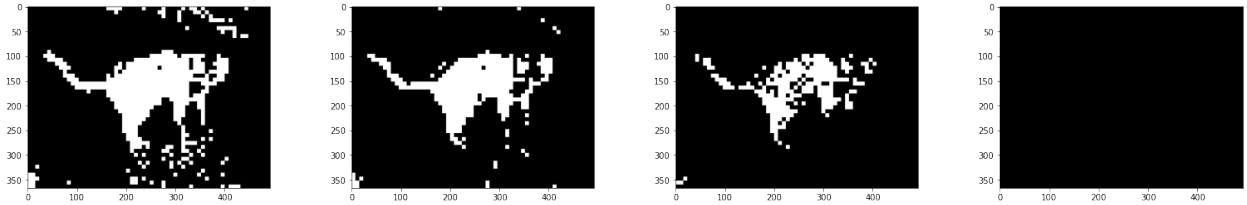
(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 10$

Figure 3: The classifier's output on the attack



(a) iter=1 (b) iter=30 (c) iter=50 (d) iter=300

Figure 4: Evolution of the classifier's output on the attack with $\lambda = 1$



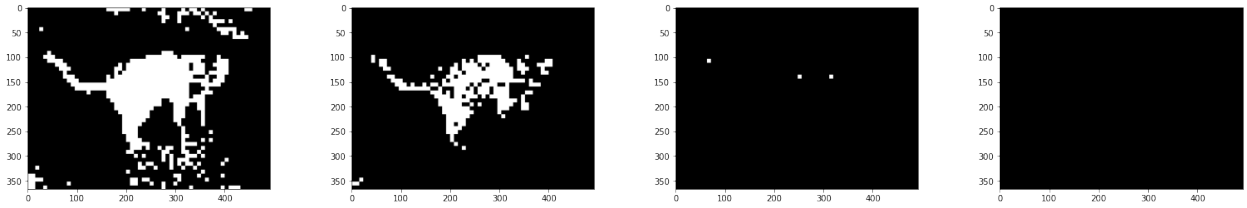
(a) iter=1 (b) iter=5 (c) iter=10 (d) iter=300

Figure 5: Evolution of the classifier's output on the attack with $\lambda = 5$

(v) The evolution of classifier's output on the perturbed image with training iterates are shown in Figs. 4-6. With higher λ, the classifier is fooled with lower iterations, this happens as higher λ means more cost if attack is unsuccessful. Again, for lower λ, since more importance is given on minimizing $||x - x_0||^2$, it takes more iters to fool the classifier. This is evident since for iter=10, with λ=10, the classifier is almost fooled completely, but for λ=1 and λ=5, there is still many patches which are still not misclassified.
(d) Next we increased α from 0.0001 to 0.001, we saw for fixed λ, the attack was able to fool the classifier with much lower no of iterations, this is expected with larger step size, the sample is perturbed faster towards the decision boundary. With λ=1, α=0.0001, it took 101 iterations to get all patches misclassified but with λ=1, α=0.001, it took just 12 iterations to get all patches misclassified.

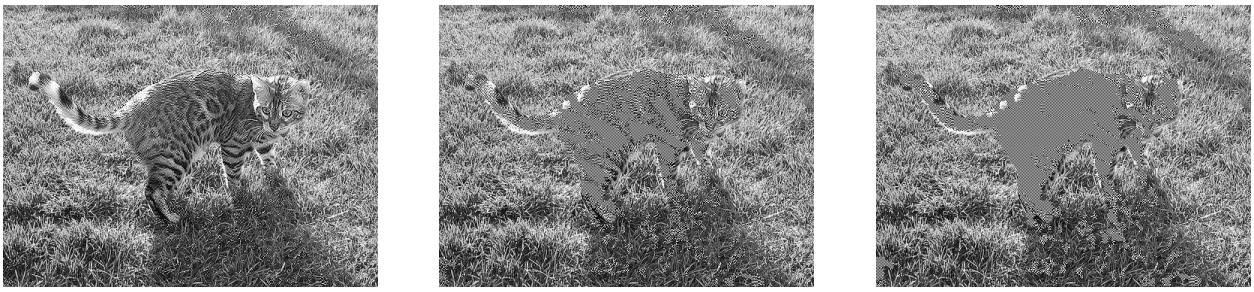
Exercise 3

(a) The gradient of the objective function in the second line of (7) on given prompt with respect to x is, $2||\boldsymbol{x} - \boldsymbol{x_0}|| + \lambda \sum_{i=1}^{d} \mathbb{1}\{g_j(\boldsymbol{P_i x}) - g_t(\boldsymbol{P_i x}) > 0\} \cdot (\nabla_x g_j(\boldsymbol{P_i x}) - \nabla_x g_t(\boldsymbol{P_i x}))$, here $(\nabla_x g_j(\boldsymbol{P_i x}) - \nabla_x g_t(\boldsymbol{P_i x})) = ((\boldsymbol{W_j} - \boldsymbol{W_t})\boldsymbol{P_i x} + (\boldsymbol{w_j} - \boldsymbol{w_t}))\boldsymbol{P}_i^T$.
(b) We implemented the CW attack on the overlapping patches classifier.



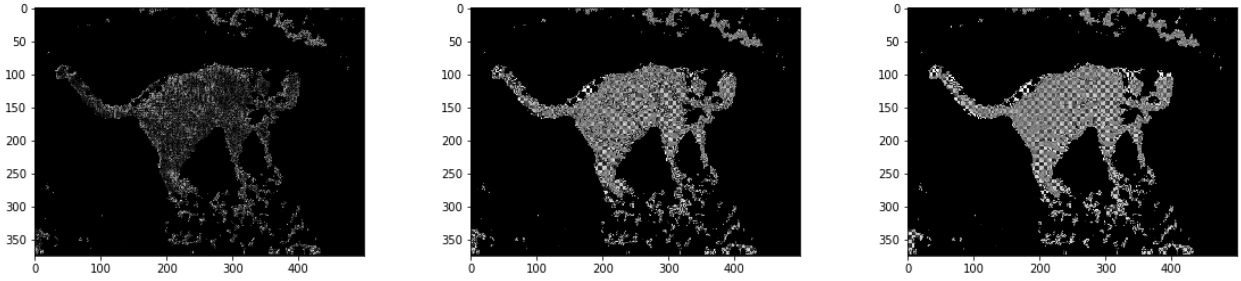
(a) iter=1 (b) iter=5 (c) iter=10 (d) iter=300

Figure 6: Evolution of the classifier's output on the attack with $\lambda = 10$



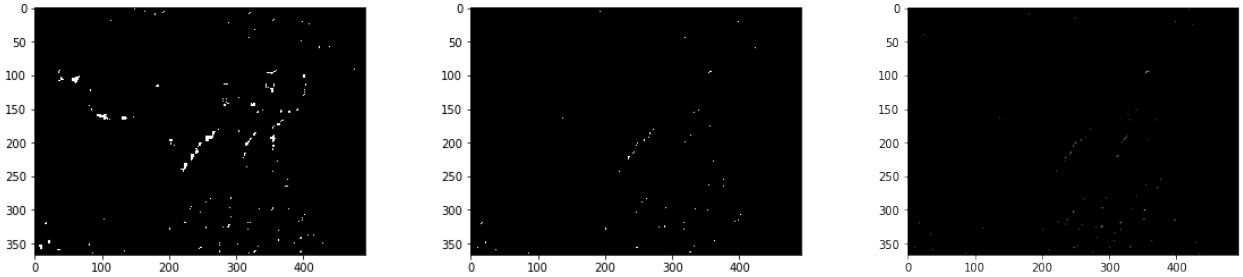
(a) $\lambda = .5$ (b) $\lambda = 1$ (c) $\lambda = 5$

Figure 7: The attack, i.e. the final perturbed image for the overlapping patches classifier



(a) $\lambda = .5$ (b) $\lambda = 1$ (c) $\lambda = 5$

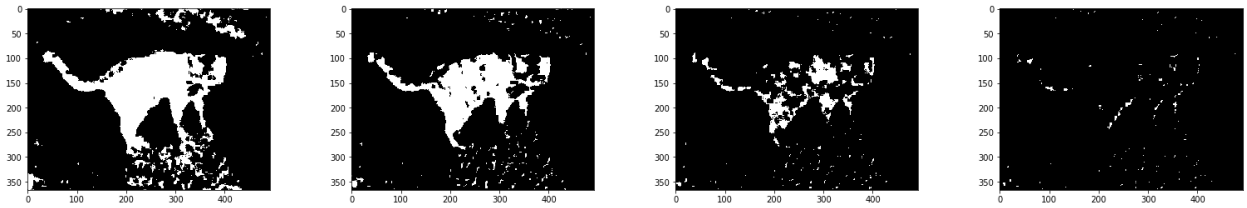
Figure 8: The perturbation added to the original image for the overlapping patches classifier



(a) $\lambda = .5$ (b) $\lambda = 1$ (c) $\lambda = 5$

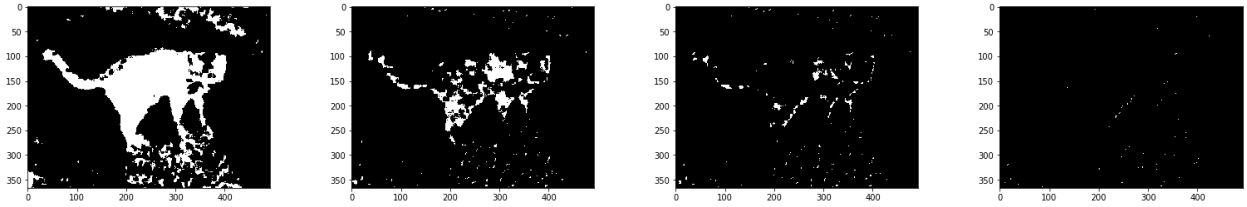
Figure 9: The classifier's output on the attack

(c)(i) The images are shown in Fig. 7. As can be seen, compared to the non-overlapping classifier attack, here the attack is less perceivable, also with smaller λ, the change in the cat pixels are lesser, like for $\lambda = .5$, the cat is changed very slightly. This is expected since here we simultaneously minimize the l_2 distance term for the whole image and the regularization term for every patch, thus we keep the perturbed image looking as "similar" to the original image as allowed, while making every patch in the perturbed image to be as "close" to the target class as possible.
(ii) The perturbations are shown in Fig. 8. Again, with smaller λ, the perturbations are lesser, like for $\lambda = .5$, the perturbation is quite negligible. This happens since for smaller λ, we put less cost on attack success and more emphasis on the attacked image being similar to original.
(iii) The frobenius norm of the perturbation for $\lambda = .5, 1$ and 5 are 44.19, 92.08 and 103.02 respectively. As expected, with smaller λ, the perturbations are lesser.
(iv) The classifier's output on the final attack images are shown in Fig. 9. With higher λs, the classifier is more fooled, more cat patches are classified as grass. This happens since for higher λ, we put more cost on attack failure, so it is able to fool the model more with same no of iterations.



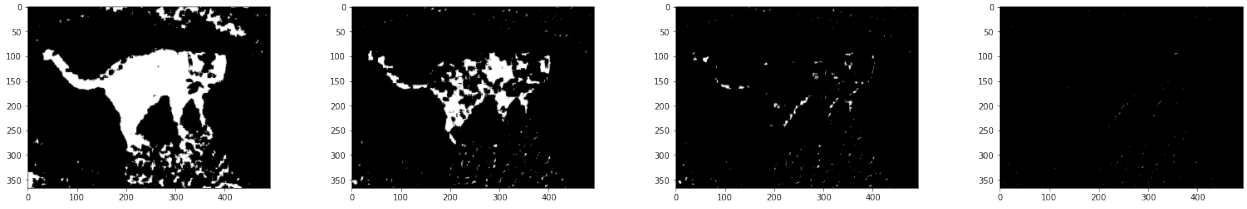
(a) iter=1 (b) iter=120 (c) iter=210 (d) iter=300

Figure 10: Evolution of the classifier's output on the attack with $\lambda = .5$



(a) iter=1 (b) iter=105 (c) iter=135 (d) iter=300

Figure 11: Evolution of the classifier's output on the attack with $\lambda = 1$



(a) iter=1 (b) iter=20 (c) iter=30 (d) iter=300

Figure 12: Evolution of the classifier's output on the attack with $\lambda = 5$

(v) The evolution of classifier's output for the overlapping attack with training iterates are shown in Figs. 10-12. With higher λ, the classifier is fooled with lower iterations, this happens as higher

λ means more cost if attack is unsuccessful. Again, for lower λ, since more importance is given on minimizing $||x - x_0||^2$, it takes more iters to fool the classifier. This is evident since for iter=30, with λ=5, the classifier is almost fooled completely, but there are many patches which are still not misclassified even at iter=210 and 105, for λ=.5 and λ=1, respectively. In terms of the quality of the attack compared to that of the last exercise, in general we may say that the attack is stronger here since given enough iterations, the attack is able to fool the model with almost complete success, however the perturbations caused by the attack is lesser compared to before.

Exercise 4 Defending the Attack

Method Description: To defend against the CW attack, we used the adversarial training [1] method here. The basic idea of the defense is to solve a minimax problem-

$$\underset{\theta}{\text{minimize}} \underset{(\mathbf{x},\mathbf{y})\sim D}{\mathbb{E}}[max_{\delta\in S}\ L(\theta, \mathbf{x}+\delta, \mathbf{y})]$$

Here, $\delta \in S$ is the attack added to the input data x; y is the true label, S defines the set of allowable attacks, e.g., l_∞ ball. First we take the maximum of the loss, $L(\theta, \mathbf{x}+\delta, \mathbf{y})$ by searching for the nastiest attack δ, then expectation over the training set D is taken to compute the (empirical) risk and finally the risk is minimized. Though adversarial training incurs additional

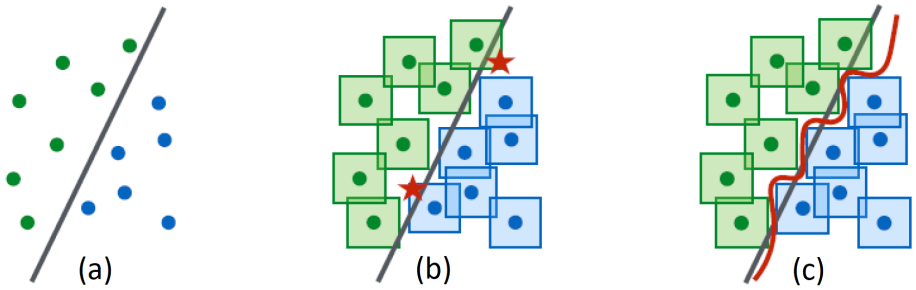


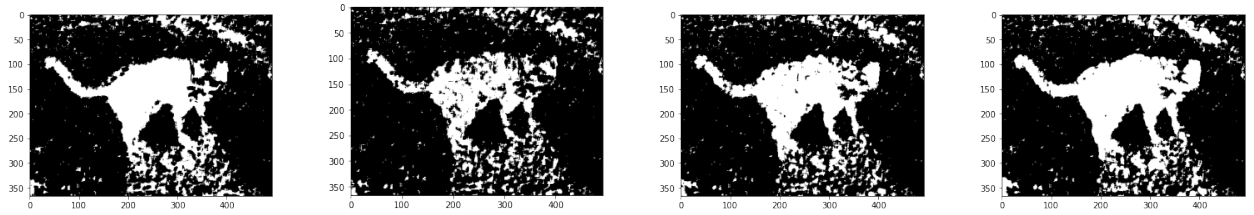
Figure 13: Standard vs adv decision boundaries [image adapted from 1], (a) Points separable by linear boundary, (b) boundary fails to separate l_∞ balls, (c) Separating l_∞ balls requires a complex decision boundary.

computational burden for large datasets, here we have just a single training image, so it's suitable. In our case, we do not have explicit training iterations for the classifier, rather the classifier has prior, mean and covariance of the cat and grass classes. So, we augment our data by obtaining an auxiliary attack image using an off-the-shelf attack generator. Then a fraction of randomly chosen patches (1000 each) from both classes are added to the training set and an updated Gaussian Classifier is obtained. The steps are given in the pseudo-code below-

Data: Clean Image with true labels, Attack Generator, Current Gaussian Classifier
Result: Modified Gaussian Classifier with metric trained from augmented dataset
1. Generate attack image from clean image using the Attack Generator;
2. Augment the training dataset using a fraction of patches from the attack image using their true label;
3. Calculate new mean, prior and covariance of the classes using this augmented dataset;

Algorithm 1: Adversarial Training Methodology for the Gaussian Classifier

Again this is done using a separate attack image and just a fraction of randomly chosen patches from the adversarial image are concatenated to the cat/grass training data according to their true label. This is done during training, so it's valid.



(a) Classifier output on clean image

(b) Classifier output on attack image with $\lambda = .5$

(c) Classifier output on attack image with $\lambda = 1$

(d) Classifier output on attack image with $\lambda = 5$

Figure 14: Modified classifier's output on the clean and attack images

Table 1: Classification Accuracy for the attack images

$\lambda = 0.5$	$\lambda = 1$	$\lambda = 5$
0.8048	0.7595	0.7309

Results and Comparison: First, the output of the augmented classifier on the clean and attack images are shown in Fig. 14. As can be seen, the defense is more or less able to classify the pixels correctly for both the clean as well as attack images for various λs. Next, we calculate the accuracy of the classifier using the truth labels, the results are given in Table 1. As expected, with a stronger attack (higher λ), the accuracy gradually decreases. Here, we calculated accuracy as one minus the absolute mean difference between classifier's output and ground truth mask. We also obtained the classification accuracy for the other baseline defenses, the results are plotted in Fig. 15. It is clear that across the attacked domain, adversarial training seems to be very effective. The high performance of the proposed adversarial training may be partially due to the fact that here the training image that was used to generate the auxiliary attack image used to augment the classifier metrics was the same as the testing image. However, an important thing to note is, we trained with a different λ and the testing λ s were different, also only a small fraction of patches from the auxiliary attack image was used to update the classifier.

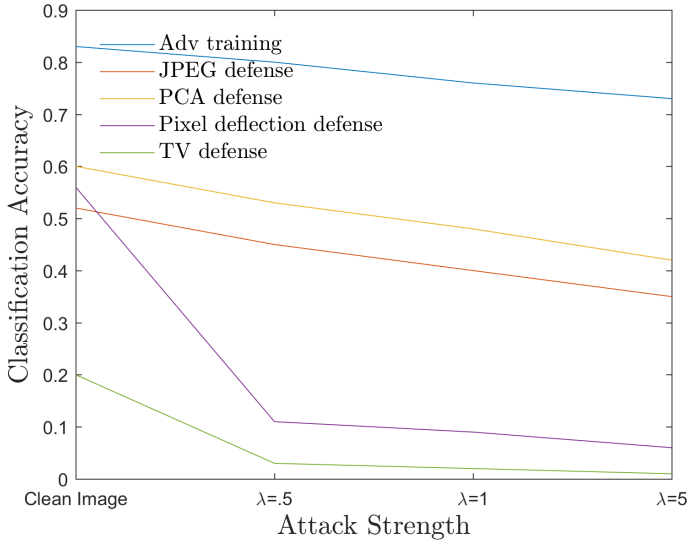
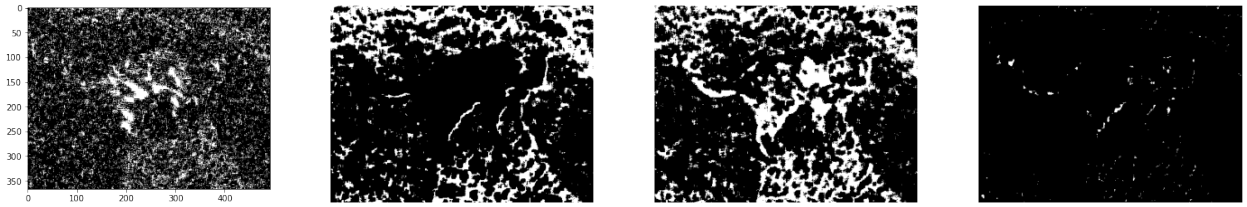


Figure 15: Classification accuracy for various defenses against different attack strengths.



(a) PCA defense (b) TV defense (c) JPEG defense (d) PD defense

Figure 16: Classifier's output on attack images with various defenses in place

In this part, we compare our results with few baselines namely PCA based defense [2], Total Variance Minimization [3], JPEG Compression [3], and Pixel Deflection methods [4]. Dimensionality reduction can be imposed in a variety of ways but, in the context of developing defensive algorithms to adversarial attacks, Principal Component Analysis (PCA) is one of the most common dimensionality reduction defenses. Another method of removing adversarial perturbations is by using pixel dropout with total variance minimization or total variation denoising or simply TV [3]. The idea is that the reconstructed image does not contain the adversarial perturbations since the perturbations are usually small and have a very low chance of being selected. Again, via JPEG Compression, the adversarial perturbation can potentially be removed. This is a by-product of the loss of image quality as some of the adversarial perturbation would be lost too. For pixel deflection (PD) [4], a random pixel is selected and replaced it with another randomly selected pixel from a local neighborhood. The reasoning behind it is most attacks are agnostic to the presence of semantic objects in the image; by picking more pixels outside the regions-of-interest the likelihood of destroying the adversarial perturbation is increased but not much of the content. The output of the classifier on the attack images with corresponding defenses in place are shown in Fig. 16. From this figure, it is clear that these defenses are not very successful in defending against the strong CW attack, the PCA-defense is able to recover some of the cat pixels, JPEG can recover quite a few cat pixels but suffers from a lot of false positives. TV and PD defenses seem to fail mostly.

Ablation Study: For this section, the importance of changing different metrics of the classifier while augmenting the dataset was explored. During training, new metrics based on training data are created. Using these metrics (covariance, mean, and prior), the classifier becomes more robust to adversarial attacks. To see how this affects the output of the algorithm, the adversarial Training was modified to use one or two of these new metrics while keeping the other the same as the original metric. From this study, it was found that covariance has the biggest impact on the defense, followed by mean and prior. Just using covariance and prior performed better than using all three metrics.

L2 Norm Difference of Defence Output Classification and Modified Adv Training				
New Metrics	No Attack	$\lambda = .5$	$\lambda = 1$	$\lambda = 5$
No Training	0.000	187.243	190.205	190.399
Training Prior	**3.162**	187.248	190.205	190.399
Training Covariance	53.656	132.623	112.329	116.982
Training Prior and Covariance	53.600	**132.570**	**112.289**	**116.970**
Training Mean	10.816	187.232	190.234	190.420
Training Mean and Prior	10.770	187.235	190.234	190.420
Training Mean and Covariance	54.488	133.247	112.512	117.145
Complete Training	54.415	133.169	112.463	117.123

Limitations: Our defense based on adversarial training does not depend on causing any obfuscation of gradients, i.e., there is no process of hiding away the gradient or destroying the gradient in order to intentionally make gradient based attackers fail. So, we believe it is able to withstand this attack even if the attacker tries to approximate the gradient or shatter any gradient obfuscation. Again, the obfuscated gradient paper by Athalye *et al.* [5] concluded that only adversarial training can withstand their attack. However, the limitation to such technique is a 10-fold adversarial training (i.e., for each image I generate 10 adversarial examples) will cause 10 times more training data, so scalability is very challenging. Again, if the attack is of a different type than those used during training, then this method may fail. Also, we have access of adversarial examples only on the training data, so if the pattern on the test cases are quite different, then the mean and covariance metric changes corresponding to the training data may not resonate well with the unseen testing ones, which may cause the method to fail again.
Discussion: Defending against the Carlini and Wagner (C & W) attack in this project has proven to be a challenging task. This is also exacerbated by the fact that we have just a single image to play/train with. With smaller λ like the case with $\lambda = .5$, it's a challenge to remove perturbations without also removing non-perturbations. With higher lambda like $\lambda = 5$ since the perturbation becomes more visible, detecting them visually becomes easier per se, but still, the classifier gets fooled totally. As a result, the methods based on input transformation struggle to perform very well. Again, the dimension of each training patch was 64, so it was difficult to implement PCA based methods successfully (we tried reducing up to 15 dimensions per patch) without losing significant image info. By training the classifier on adversarial images, there's no need to remove the perturbation by making the classifier more robust in exchange for accuracy. Other types of defenses that were not explored include ensemble methods, some other transforms like Image Quilting, other randomization based approaches, adversarial detection, denoising autoencoder, etc. We gained some useful insight from this project. We believe adversarial training can easily be added to more complicated machine learning models as it has been shown before [1]. However, such augmentation of dataset causes significant additional computational overhead on the training procedure which may not be feasible in all circumstances. Again, newer and different types of attacks may circumvent this type of adversarial training defense provided attack is strong enough. Also, there is a tradeoff on clean image accuracy with robustness as well. So, building a successful defense must take into account all these factors and as a designer, we must choose the optimal route.

References

[1] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.
[2] Bhagoji, Arjun Nitin, Daniel Cullina, and Prateek Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. arXiv preprint arXiv:1704.02654 2 (2017).
[3] Guo, Chuan, et al. Countering adversarial images using input transformations. arXiv preprint arXiv:1711.00117 (2017).
[4] Prakash, A., Moran, N., Garber, S., DiLillo, A., Storer, J. (2018). Deflecting adversarial attacks with pixel deflection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8571-8580).
[5] Athalye, Anish, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. arXiv preprint arXiv:1802.00420 (2018).