Appendix

A. Proofs

A.1. Proof of Lemma 1

Proof. We know $\sum_{j=1}^{k} f_{[j]}$ is the solution of

$$\max_{\mathbf{p}} \mathbf{p}^\top F, \text{ s.t. } \mathbf{p}^\top \mathbf{1} = k, \mathbf{0} \le \mathbf{p} \le \mathbf{1}.$$

We apply Lagrangian to this equation and get

$$L = -\mathbf{p}^\top F - \mathbf{v}^\top \mathbf{p} + \mathbf{u}^\top (\mathbf{p} - 1) + \lambda(\mathbf{p}^\top \mathbf{1} - k)$$

where $\mathbf{u} \ge \mathbf{0}$, $\mathbf{v} \ge \mathbf{0}$ and $\lambda \in \mathbb{R}$ are Lagrangian multipliers. Taking its derivative w.r.t. $\mathbf{p}$ and set it to 0, we have $\mathbf{v} = \mathbf{u} - F + \lambda\mathbf{1}$. Substituting it back into the Lagrangian, we get

$$\min_{\mathbf{u},\lambda} \mathbf{u}^\top \mathbf{1} + k\lambda, \text{ s.t. } \mathbf{u} \ge \mathbf{0}, \mathbf{u} + \lambda\mathbf{1} - F \ge 0.$$

This means

$$\sum_{j=1}^{k} f_{[j]} = \min_{\lambda} \Big\{ k\lambda + \sum_{j=1}^{m} [f_j - \lambda]_+ \Big\}.$$

Therefore,

$$\phi_k(F) = \frac{1}{k} \min_{\lambda} \Big\{ k\lambda + \sum_{j=1}^{m} [f_j - \lambda]_+ \Big\}. \tag{16}$$

Furthermore, we can see that $\lambda = f_{[k]}$ is always one optimal solution for Eq.(16). So

$$f_{[k]} \in \arg\min_{\lambda} \Big\{ k\lambda + \sum_{j=1}^{m} [f_j - \lambda]_+ \Big\}.$$

□

A.2. Proof of Lemma 2

Proof. Denote $g(x) = [[a - x]_+ - \lambda]_+$. For $\lambda \ge 0$, we have $g(x) = 0 = [a - x - \lambda]_+$ if $x \ge a$. On the other hand, if $x < a$, we have $g(x) = [a - x - \lambda]_+$. Therefore, $g(x) = [[a - x]_+ - \lambda]_+ = [a - x - \lambda]_+$ for any $\lambda \ge 0$. □

B. Additional Experimental Details

B.1. Source Code

For the purpose of review, the source code is accessible in the supplementary file.

B.2. Computing Infrastructure Description

All algorithms are implemented in Python 3.6 and trained and tested on an Intel(R) Xeon(R) CPU W5590 @3.33GHz with 48GB of RAM and an NVIDIA Quadro RTX 6000 GPU with 24GB memory.

B.3. The Algorithm for The Universal Untargeted Attack

First, we define the universal attack success rate (UASR) as

$$\text{UASR} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{[E(Y(\mathbf{x}_i), \hat{Y}(\mathbf{x}_i + \mathbf{z})) = 0]}. \tag{17}$$

We apply Algorithm 1 (without using projection in it) iteratively over all samples from $\mathbf{X}$. At each iteration, Algorithm 1 finds a perturbation $\Delta\mathbf{z}_i$ for a given data point $\mathbf{x}_i + \mathbf{z}$, which success attack all top-k labels for the current data x_i. Then we update the universal adversarial perturbation $\mathbf{z}$ by using a projection operation to it. The overall procedure is described in detail in Algorithm 3. The algorithm terminates when the predefined attack success rate ξ is reached.

Algorithm 3: T$_k$ML-AP-Uv

```
   Input: $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, predictor $F$, $k$, $\eta_l$, $\xi$, $\beta$
   Output: perturbation $\mathbf{z}^*$
1  Initialization: $\mathbf{z}$
2  while $UASR < \xi$ do
3      for $\mathbf{x}_i \in X$ do
4          if $E(Y(\mathbf{x}_i), \hat{Y}(\mathbf{x}_i + \mathbf{z})) \neq 0$ then
5              $\_, \Delta\mathbf{z}_i =$ T$_k$ML-AP-U($\mathbf{x}_i + \mathbf{z}, F, k, \eta_l, \beta$) ▷ Algorithm 1
6              $\mathbf{z} = \mathcal{P}_\epsilon(\mathbf{z} + \Delta\mathbf{z}_i)$
7          end
8      end
9  end
10 $\mathbf{z}^* = \mathbf{z}$
11 return $\mathbf{z}^*$
```

B.4. Baseline Model Settings

Specifically, we tune the model with Adam optimizer with an initial learning rate of 0.001 and batch size of 64 on PASCAL VOC 2012 and MS COCO 2014 for 25 and 100 epochs respectively.

For the PASCAL VOC 2012 dataset, following the protocol of [22, 27] for a fair comparison, which trained on the training set (5,717 images) and tests on the validation set (5,823 images). The MS COCO 2014 dataset is a larger dataset comparing with the PASCAL VOC 2012 in terms of both numbers of classes and images. It does not provide the ground truth labels for the testing images either. Similarly, we do the training on the training set (82,081 images) and testing on the validation set (40,137 images). The images are normalized into the range of $[-1, 1]$.

k	Methods	PASCAL VOC 2012						MS COCO 2014					
		k'=3		k'=5		k'=10		k'=3		k'=5		k'=10	
		Pert($\times 10^{-2}$)	ASR	Pert($\times 10^{-2}$)	ASR	Pert($\times 10^{-2}$)	ASR	Pert($\times 10^{-2}$)	ASR	Pert($\times 10^{-2}$)	ASR	Pert($\times 10^{-2}$)	ASR
3	kFool	1.64	93.7	0.95	15.7	0.91	3.2	5.48	61.4	0.99	5.5	0.65	0.4
	T$_k$ML-AP-U	**0.51**	**99.6**	0.24	3.6	0.18	0.3	**0.24**	**100**	0.43	26.5	0.35	3.9
5	kFool	-	-	2.39	93.5	1.59	20.4	-	-	9.92	65.2	1.10	6.8
	T$_k$ML-AP-U	-	-	**0.56**	**99.3**	0.18	1	-	-	**0.53**	**100**	0.39	9.8
10	kFool	-	-	-	-	4.87	88.7	-	-	-	-	1.66	68.1
	T$_k$ML-AP-U	-	-	-	-	**0.63**	**98.3**	-	-	-	-	**0.59**	**100**

Table 6: *Comparison of* Pert *and* ASR *(%) of the untargeted attack methods with $k = 3, 5, 10$ on two datasets. The best results are shown in bold. '-' represents the current results in the $k' < k$ setting are the same as the results in the $k' = k$ setting.*

Methods	k	MS COCO 2014	
		Pert($\times 10^{-2}$)	ASR
kFool	10	16.45	68.1
	15	23.49	66.7
	20	26.01	65.7
	25	57.38	63
	30	97.67	62.7
	40	103.53	51.2

Table 7: Pert *and* ASR *(%) of kFool method in different k settings on MS COCO 2014 dataset.*

B.5. Settings of Attacking Methods

We use the same learning rate 0.01 for T$_k$ML-AP-U, T$_k$ML-AP-T, and ML-AP methods. We set 1000 as the maximum iterations in all untargeted and targeted attack methods. we only test the ℓ_2 norm for the perturbations. However, our algorithms can also work on other ℓ_p norms, *i.e.*, ℓ_1 and ℓ_∞. Since the optimization may get stuck in extreme spots, we follow similar image processing and variable transformation methods in the algorithms based on [3] to avoid this problem.

Instead of taking a long time to find a good trade-off hyper-parameter β in all algorithms, we use a projection method [14, 9] on $\mathbf{z}$. After each iteration, we apply projection on the $\mathbf{z}$ with a projector operation $\mathcal{P}_\epsilon$ controls the criteria $\|\mathbf{z}\|_2 \le \epsilon$, where ϵ is a predefined robustness threshold.

In the untargeted attack experiments, we set the projection threshold $\epsilon = 10$ in our algorithm. Since our algorithm and the kFool method have no terminate conditions, they will success attack all images in the test set of data. This means both of them can achieve a 100% ASR score in the final performance even take a long time in some specific images. To avoid this situation, we set the maximum iteration equals 1000 as we mentioned before. After both algorithms finish attack 1000 images in each dataset, we report the final performance.

We set $\epsilon = 2$ in the targeted attack experiment and report the final performance in the main paper. However, we also test other ϵ settings in the following Section.

C. Additional Experimental Results

C.1. Additional Untargeted Attacking Performance

First, we report the complete results with the same setting from Table 2 in Table 6. Second, we analysis some results displayed in Table 2. For the kFool method, we can see

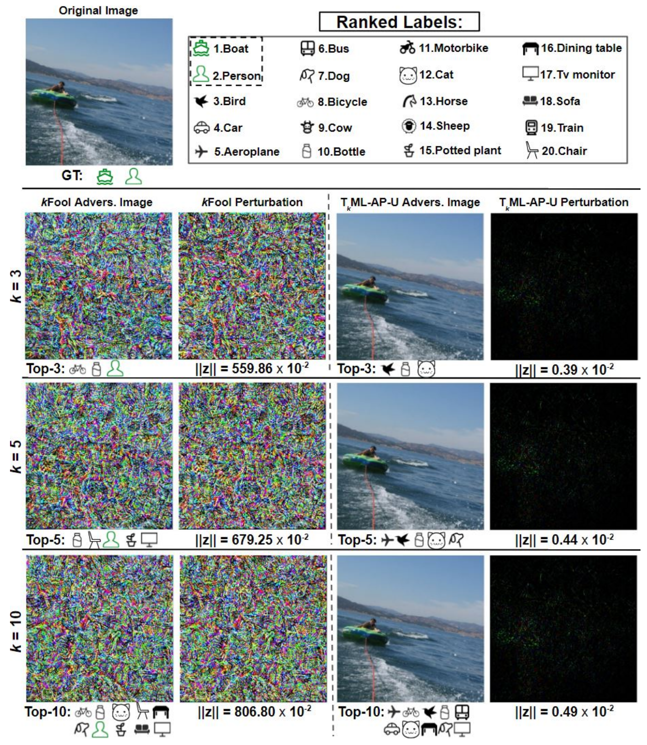

Figure 5: *Examples of kFool and T$_k$ML-AP-U adversarial perturbations with $k = 3, 5, 10$ on PASCAL VOC 2012. To better show perturbations, we have multiplied the intensity of all perturbation images by 20. GT means the ground truth labels. Top-k (k=3, 5, 10) means the Top-k predicted labels from the corresponding image. Advers. means adversarial. The green icons correspond to the ground truth labels.*

that ASR scores from 93.7% to 88.7%, which are decreasing with increasing the k value in the PASCAL VOC 2012 dataset. This is because the top-k attack becomes hard when the k is increasing. The algorithm needs to take more effort to push ground truth labels outside the top-k position. However, we find an opposite trend that ASR score is increasing when the k value is increasing in the COCO dataset. The reason is that the number of labels in the COCO dataset is four times that in VOC. In other words, the number of non-ground truth labels is very large in the COCO dataset. When the k value is small, the performance of the kFool method is not influenced much by the k settings. However, when the k value is large and continues increasing, the ASR score will be decreased because the k value has more impact on the al-

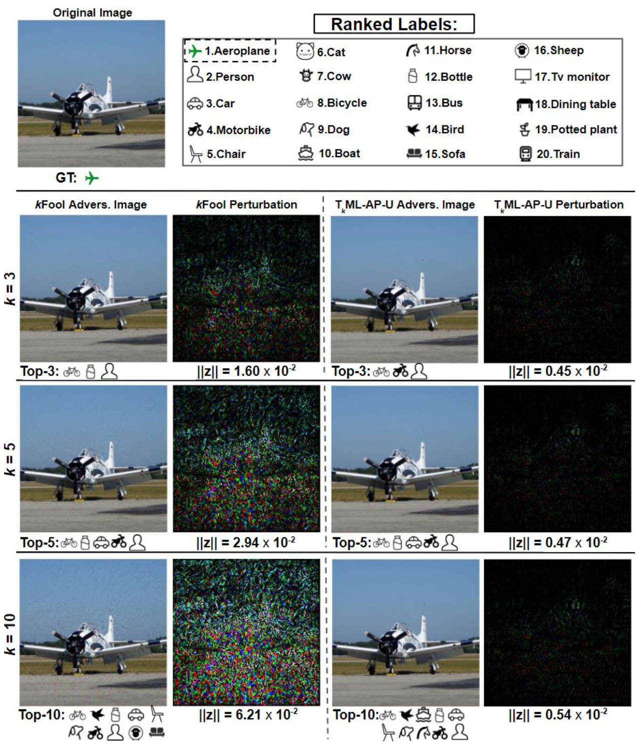

Figure 6: *Examples of kFool and T_kML-AP-U adversarial perturbations with $k = 3, 5, 10$ on PASCAL VOC 2012. To better show perturbations, we have multiplied the intensity of all perturbation images by 20. GT means the ground truth labels. Top-k (k=3, 5, 10) means the Top-k predicted labels from the corresponding image. Advers. means adversarial. The green icons correspond to the ground truth labels.*

gorithm. We have verified this statement through additional experiments in Table 7.

C.2. Additional Untargeted Attacking Image Results

We show a failed example from the kFool method in Figure 5. Since the original kFool method is for attacking the top-k multi-class classifier, we show the results, which an image has only one ground-truth label in Figure 6. From Figure 6, we can see that our method outperforms the kFool method even our method is reduced to a multi-class version.

C.3. Additional Universal Untargeted Attacking Performance

We also compare the performance of our T_kML-AP-Uv method and the kUAPs method. In the training of both algorithms, when UASR larger than 0.7, we output the universal perturbation $||\mathbf{z}||$ and use it to evaluate the attacking methods and report the performance in Table 8. Note that the performance in Table 4 is a partial results of the performance in Table 8. To evaluate the performance efficiently and avoid the algorithm still get stuck in the loop, we set 20

ξ	k	Methods	PASCAL VOC 2012				MS COCO 2014			
			k'=1	k'=2	k'=3	Pert	k'=1	k'=2	k'=3	Pert
0.7	1	kUAPs	×	×	×	×	63.9	51.4	45	0.51
		T_kML-AP-Uv	**72.3**	60	50.2	**0.15**	**86.5**	68.5	62.9	**0.13**
	2	kUAPs	-	×	×	×	-	74.6	65.9	0.51
		T_kML-AP-Uv	-	**68**	59.5	**0.16**	-	**82**	82	**0.15**
	3	kUAPs	-	-	×	×	-	-	73.2	0.51
		T_kML-AP-Uv	-	-	**65**	**0.17**	-	-	**80.5**	**0.16**
0.8	1	kUAPs	×	×	×	×	66.2	56.2	49.4	0.51
		T_kML-AP-Uv	**74.2**	60.7	50.4	**0.14**	**84.5**	70.4	63.7	**0.13**
	2	kUAPs	-	×	×	×	-	×	×	×
		T_kML-AP-Uv	-	**70.5**	61.7	**0.16**	-	**81.2**	75.3	**0.15**
	3	kUAPs	-	-	×	×	-	-	73.2	0.51
		T_kML-AP-Uv	-	-	**69**	**0.18**	-	-	**78.9**	**0.16**

Table 8: *Comparison of `Pert` and `ASR` (%) of the universal untargeted attack methods for two datasets. '×' represents the current algorithm cannot get results. '-' represents the current results in the $k' < k$ setting are the same as the results in the $k' = k$ setting. $\epsilon = 100$. The best results are shown in bold.*

ξ	k	Methods	PASCAL VOC 2012				MS COCO 2014			
			k'=1	k'=2	k'=3	Pert	k'=1	k'=2	k'=3	Pert
0.7	1	kUAPs	×	×	×	×	**97.6**	91.5	85.4	1.24
		T_kML-AP-Uv	**72.3**	60	50.2	**0.14**	86.5	68.5	62.9	**0.13**
	2	kUAPs	-	**78.5**	70.6	0.85	-	**99.4**	98.8	10.16
		T_kML-AP-Uv	-	68	59.5	**0.16**	-	82	82	**0.15**
	3	kUAPs	-	-	**79.5**	2.09	-	-	**73.2**	10.16
		T_kML-AP-Uv	-	-	65	**0.17**	-	-	80.5	**0.16**
0.8	1	kUAPs	**82.9**	70.6	60.7	0.66	**96.7**	78.6	59.4	2.60
		T_kML-AP-Uv	74.2	60.7	50.4	**0.14**	84.5	70.4	63.7	**0.13**
	2	kUAPs	-	**84.2**	77	0.98	-	**99.3**	98.1	1.62
		T_kML-AP-Uv	-	70.5	61.7	**0.16**	-	81.2	75.3	**0.15**
	3	kUAPs	-	-	×	×	-	-	**98.6**	3.95
		T_kML-AP-Uv	-	-	**69**	**0.18**	-	-	78.9	**0.16**

Table 9: *Comparison of `Pert` and `ASR` (%) of the universal untargeted attack methods for two datasets. '×' represents the current algorithm cannot get results. '-' represents the current results in the $k' < k$ setting are the same as the results in the $k' = k$ setting. $\epsilon = 2000$. The best results are shown in bold.*

as the maximum iteration for the outer loop in both algorithms. We use '×' to represent the algorithm that cannot satisfy the terminate conditions after the maximum iteration. We also report the results when $\xi = 0.8$.

In MS COCO 2014 dataset, the ASR scores from our method are higher than the scores from the kUAPs. There is a huge gap (12.6%) between two methods when $k = 1$ and $\xi = 0.7$. Second, we can find that the Pert from our method is smaller than it from the kUAPs method in all k value settings. On the other hand, we find the ASR scores from our method are decreasing when increasing the k value. However, there is no same trend in the kUAPs. The maximum score (74.6%) can be achieved when $k = 2$. A potential explanation is that the kUAPs is based on the modified kFool comparative method, which has no guarantee of the optimal solution in the optimization procedures. But our method is based on T_kML-AP-U, which uses the sum of top-k (a convex relaxation) in the optimization.

When we compare the results between two datasets in our method, we can find that the ASR scores from the MS COCO 2014 dataset are larger than ones from the PASCAL VOC 2012 dataset. The reason is that MS COCO 2014 has 80 labels. There are many labels (exclude the ground truth labels) that can be pushed to the top-k positions. However, there are only 20 labels in the PASCAL VOC 2012 dataset. Therefore, the attacking methods are easy to attack the baseline models with a dataset that contains more labels.

Cases	k	Methods	PASCAL VOC 2012						MS COCO 2014					
			k'=3		k'=5		k'=10		k'=3		k'=5		k'=10	
			Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR
Best	3	ML-AP	3.04	64.7	2.63	3.6	1.14	0.1	4.96	97.5	4.65	11.2	4.35	1.1
		T$_k$ML-AP-T	3.04	**64.7**	2.70	2.80	1.14	0.1	5.10	97.5	4.51	9.1	4.73	0.7
	5	ML-AP	-	-	3.20	51.5	2.02	0.5	-	-	5.55	94	4.70	3
		T$_k$ML-AP-T	-	-	3.20	**51.8**	1.96	0.7	-	-	5.68	**94.5**	5.14	2.5
	10	ML-AP	-	-	-	-	3.33	34.4	-	-	-	-	6.10	86.5
		T$_k$ML-AP-T	-	-	-	-	3.37	**35.5**	-	-	-	-	6.22	**87.8**
Random	3	ML-AP	3.58	34.3	3.44	12.9	3.31	1.6	6.58	84.5	6.50	45.2	6.27	14
		T$_k$ML-AP-T	3.60	**38.2**	3.46	12.8	3.49	1.1	6.63	**86.3**	6.55	42.6	6.52	11.5
	5	ML-AP	-	-	3.67	23.7	3.82	4.7	-	-	6.72	60.1	6.61	22.9
		T$_k$ML-AP-T	-	-	3.76	**28.2**	3.54	3.7	-	-	6.81	**68.1**	6.65	21.1
	10	ML-AP	-	-	-	-	3.75	17.6	-	-	-	-	6.95	26.4
		T$_k$ML-AP-T	-	-	-	-	3.80	**20.5**	-	-	-	-	6.97	**42.6**
Worst	3	ML-AP	3.88	13.7	3.85	9	3.82	2.2	6.71	61.2	6.68	42.4	6.78	18.5
		T$_k$ML-AP-T	3.85	**17.5**	3.79	10.2	3.74	2.6	6.76	**65.5**	6.74	42	6.75	17.4
	5	ML-AP	-	-	3.96	8.3	4.03	3.4	-	-	6.75	39.9	6.70	23.9
		T$_k$ML-AP-T	-	-	3.98	**11.3**	4.01	3.3	-	-	6.80	**47.2**	6.76	23.2
	10	ML-AP	-	-	-	-	3.95	6.2	-	-	-	-	6.90	14.7
		T$_k$ML-AP-T	-	-	-	-	4.04	**10.2**	-	-	-	-	6.88	**24.4**

Table 10: *Comparison of* `Pert` *and* `ASR` *(%) of the targeted attack methods with k=3, 5, 10 in the Best, Random, and Worst cases on two datasets. The best* `ASR` *results are shown in bold.* $\epsilon = 1$.

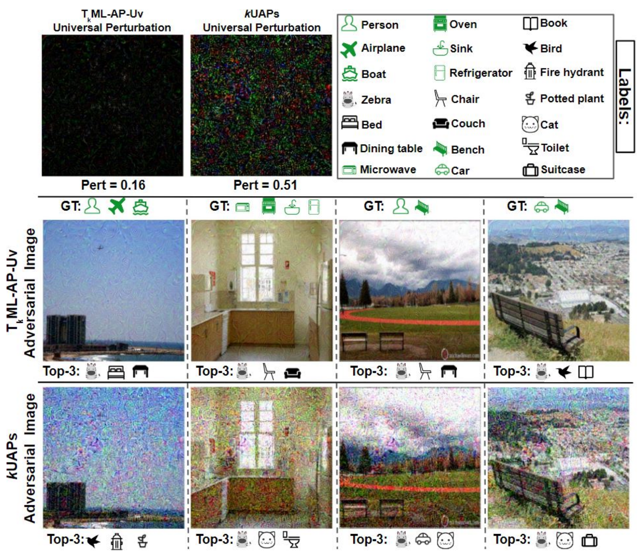

Figure 7: *Examples of kUAPs and T$_k$ML-AP-Uv adversarial perturbations with $k = 3$ on MS COCO 2014. GT means the ground truth labels. Top-3 means the Top-3 predicted labels from the corresponding image. The green icons correspond to the ground truth labels.*

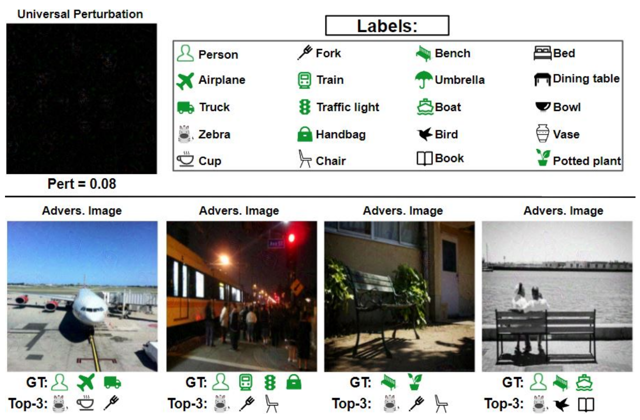

Figure 8: *Examples of T$_k$ML-AP-Uv adversarial perturbations with $k = 3$ on MS COCO 2014. GT means the ground truth labels. Top-3 means the Top-3 predicted labels from the corresponding image. Advers. means adversarial. The green icons correspond to the ground truth labels.* $\epsilon = 15$.

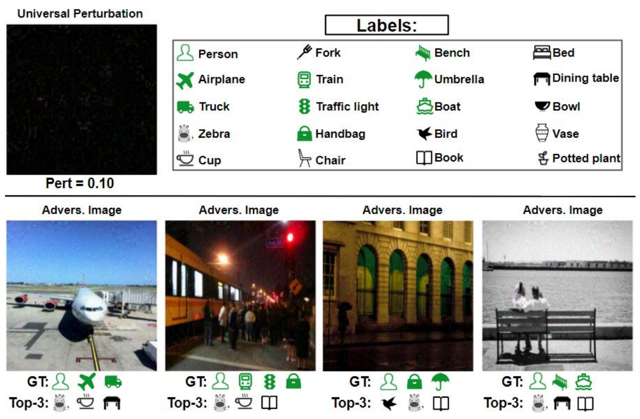

Figure 9: *Examples of T$_k$ML-AP-Uv adversarial perturbations with $k = 3$ on MS COCO 2014. GT means the ground truth labels. Top-3 means the Top-3 predicted labels from the corresponding image. Advers. means adversarial. The green icons correspond to the ground truth labels.* $\epsilon = 20$.

Since we set 100 as a projection threshold in the algorithm, the norm of $\mathbf{z}$ from the kUAPs method in all k value settings can reach the maximum threshold, which means the perturbed images from kUAPs will distortion. When we increase the projection threshold, these values from the kUAPs will become larger. But the values from the T$_k$ML-AP-Uv method are stable. We show the results in Table 9 with setting the projection threshold $\epsilon = 2000$.

C.4. Additional Universal Untargeted Attacking Image results

We set $\epsilon = 100$ and $\xi = 0.7$, then show the image results from both methods in Figure 7. However, we can set a smaller ϵ value and get a smaller perturbation. Here, we show more perturbed image results and report their top-3 predictions based on our T$_k$ML-AP-Uv method with setting different ϵ. We set the projection threshold $\epsilon = 15$ in Figure 8 and $\epsilon = 20$ in Figure 9.

Cases	k	Methods	PASCAL VOC 2012						MS COCO 2014					
			k'=3		k'=5		k'=10		k'=3		k'=5		k'=10	
			Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR
Best	3	ML-AP	4.44	96.2	3.70	15	2.42	0.2	5.49	100	5.13	11.4	5.09	1.3
		T_kML-AP-T	4.45	**96.6**	3.57	4.10	1.14	0.1	5.71	**100**	5.55	11.4	5.68	0.9
	5	ML-AP	-	-	5.01	92	2.59	0.6	-	-	6.57	99.9	5.94	3.9
		T_kML-AP-T	-	-	5.02	**92.8**	1.18	0.1	-	-	6.86	**99.9**	5.69	2.9
	10	ML-AP	-	-	-	-	5.53	84.2	-	-	-	-	8.06	99.8
		T_kML-AP-T	-	-	-	-	5.59	**86.4**	-	-	-	-	8.52	**99.8**
Random	3	ML-AP	5.90	86	5.69	51.7	4.88	3.1	9.43	99.8	9.35	60.7	9.17	24.9
		T_kML-AP-T	5.90	**89.8**	5.39	25.3	4.57	2.4	9.89	**99.9**	9.79	58.4	9.72	22.7
	5	ML-AP	-	-	6.22	77.9	5.78	16.3	-	-	11.10	96.5	11.00	43.2
		T_kML-AP-T	-	-	6.27	**83.7**	5.66	10.5	-	-	11.80	**97.8**	11.70	42.7
	10	ML-AP	-	-	-	-	6.27	67.7	-	-	-	-	12.20	84.2
		T_kML-AP-T	-	-	-	-	6.41	**76.4**	-	-	-	-	12.80	**94.5**
Worst	3	ML-AP	6.59	68	6.43	42.6	5.86	7.7	10.80	90	10.90	72.2	11.10	43.3
		T_kML-AP-T	6.64	**75.8**	6.35	36.2	5.79	7.3	11.40	**91.4**	11.50	71.6	11.70	43
	5	ML-AP	-	-	6.75	53.3	6.47	18.8	-	-	11.90	81.8	11.90	56.6
		T_kML-AP-T	-	-	6.90	**66.6**	6.41	16.6	-	-	12.50	**87.2**	12.60	58.9
	10	ML-AP	-	-	-	-	6.68	39.1	-	-	-	-	12.50	59
		T_kML-AP-T	-	-	-	-	6.91	**57**	-	-	-	-	13.00	**73.1**

Table 11: *Comparison of* `Pert` *and* `ASR` *(%) of the targeted attack methods with k=3, 5, 10 in the Best, Random, and Worst cases on two datasets. The best* `ASR` *results are shown in bold.* $\epsilon = 2$.

Cases	k	Methods	PASCAL VOC 2012						MS COCO 2014					
			k'=3		k'=5		k'=10		k'=3		k'=5		k'=10	
			Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR
Best	3	ML-AP	4.74	98.9	4.12	5.6	1.14	0.1	5.51	100	5.31	12.2	4.57	1
		T_kML-AP-T	4.76	**99.2**	3.91	4.20	1.14	0.1	5.72	**100**	5.48	11.1	4.91	0.7
	5	ML-AP	-	-	5.53	97.2	5.29	1.1	-	-	6.60	99.9	5.57	3.3
		T_kML-AP-T	-	-	5.54	**97.8**	4.71	1.1	-	-	6.91	**99.9**	6.75	3.9
	10	ML-AP	-	-	-	-	6.35	94	-	-	-	-	8.18	100
		T_kML-AP-T	-	-	-	-	6.43	**95.7**	-	-	-	-	8.69	**100**
Random	3	ML-AP	6.77	95.5	6.39	40.2	5.39	2.8	9.62	100	9.47	64.6	9.36	25
		T_kML-AP-T	6.72	**97.7**	6.06	29	5.03	1.9	10.10	**100**	10.00	61.6	9.94	23.4
	5	ML-AP	-	-	7.32	90.7	6.71	17.2	-	-	11.90	98.5	11.50	45
		T_kML-AP-T	-	-	7.34	**94.8**	6.30	12	-	-	13.00	**99.3**	12.90	44.4
	10	ML-AP	-	-	-	-	7.55	85	-	-	-	-	14.40	95
		T_kML-AP-T	-	-	-	-	7.74	**91.8**	-	-	-	-	16.50	**99.1**
Worst	3	ML-AP	7.92	86.1	7.75	54.8	7.09	10.9	11.90	96.6	12.10	77.3	12.60	49.3
		T_kML-AP-T	8.04	**92.9**	7.67	48.5	6.83	9.3	12.70	**98**	13.00	77.7	13.50	50.7
	5	ML-AP	-	-	8.30	75.7	7.67	27.8	-	-	13.50	92.8	13.80	67.8
		T_kML-AP-T	-	-	8.58	**89.2**	7.74	21.5	-	-	15.00	**95.8**	15.30	68.3
	10	ML-AP	-	-	-	-	8.37	64.3	-	-	-	-	15.50	75.6
		T_kML-AP-T	-	-	-	-	8.84	**83.6**	-	-	-	-	17.90	**90**

Table 12: *Comparison of* `Pert` *and* `ASR` *(%) of the targeted attack methods with k=3, 5, 10 in the Best, Random, and Worst cases on two datasets. The best* `ASR` *results are shown in bold.* $\epsilon = 3$.

Cases	k	Methods	PASCAL VOC 2012						MS COCO 2014					
			k'=3		k'=5		k'=10		k'=3		k'=5		k'=10	
			Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR	Pert($\times 10^{-3}$)	ASR
Best	3	ML-AP	4.83	99.6	3.62	4.7	1.14	0.1	5.50	100	5.42	12.3	4.57	1.2
		T_kML-AP-T	4.86	**99.8**	3.64	4.3	1.14	0.1	5.74	**100**	5.66	11.4	6.84	1.1
	5	ML-AP	-	-	5.68	98.4	3.65	1	-	-	6.60	99.9	5.98	3.8
		T_kML-AP-T	-	-	5.73	**99.1**	2.92	0.9	-	-	6.91	**99.9**	5.96	3.3
	10	ML-AP	-	-	-	-	6.62	96.7	-	-	-	-	8.19	100
		T_kML-AP-T	-	-	-	-	6.75	**98.2**	-	-	-	-	8.70	**100**
Random	3	ML-AP	7.00	97.4	6.66	40.6	5.48	4	9.61	100	9.51	63.2	9.22	24.6
		T_kML-AP-T	6.96	**99**	6.31	29.5	5.04	3.1	10.10	**100**	9.98	61.7	9.91	23.4
	5	ML-AP	-	-	7.61	93.6	6.64	16.3	-	-	12.00	99	11.70	44.7
		T_kML-AP-T	-	-	7.84	**98.6**	6.33	12.3	-	-	13.10	**99.7**	13.10	44.6
	10	ML-AP	-	-	-	-	7.93	89.1	-	-	-	-	14.70	96.8
		T_kML-AP-T	-	-	-	-	8.33	**96.4**	-	-	-	-	17.20	**99.9**
Worst	3	ML-AP	8.35	90.9	8.07	57.4	7.24	11.7	12.10	97.9	12.30	80.1	13.10	52.3
		T_kML-AP-T	8.66	**97.8**	8.09	50.6	7.09	10.9	12.90	**98.9**	13.30	77.4	13.90	50.9
	5	ML-AP	-	-	8.75	80.7	8.08	29.4	-	-	13.80	94.1	14.20	68.9
		T_kML-AP-T	-	-	9.47	**96.2**	8.62	25	-	-	15.60	**98.3**	16.00	72.2
	10	ML-AP	-	-	-	-	8.81	69.2	-	-	-	-	16.00	78.7
		T_kML-AP-T	-	-	-	-	10.10	**94.7**	-	-	-	-	19.90	**95.7**

Table 13: *Comparison of* `Pert` *and* `ASR` *(%) of the targeted attack methods with k=3, 5, 10 in the Best, Random, and Worst cases on two datasets. The best* `ASR` *results are shown in bold.* $\epsilon = 10$.

C.5. Additional Targeted Attacking Performance

In the main paper, we set the projection threshold $\epsilon = 2$ and report partial results. Here, we set more different projection thresholds such as $\epsilon = 1$ in both T_kML-AP-T and ML-AP algorithms and show the performance in the Table 10. We also show the complete results in Table 11, 12 and 13 with setting $\epsilon = 2, 3, 10$, respectively. From these Tables, we can see that the performance is increasing when ϵ is increasing.

In the level of the cases, we find that the `ASR` score is decreasing when the selected labels are hard to attack in the same k values. For example, for $k' = 3$ in the PASCAL VOC 2014 dataset with $\epsilon = 2$, the `ASR` score of the ML-AP method is decreased from 96.2% (Best) to 86% (Random), which has a 10.2% difference. This difference becomes large from 86% (Random) to 68% (Worst), which has an 18% difference. On the other hand, for our method, we can find the difference is 6.8% from 96.6% (Best) to 89.8% (Random), and the difference becomes 14% from the 89.8% (Random) to 75.8% (Worst). Comparing these difference values between the two methods in the same scenario, we can see that the difference from T_kML-AP-T is smaller than the difference of ML-AP. This means our method is more robust than the ML-AP method. For the `Pert` values of perturbation norms, we find that it is increased when the selected labels are hard to attack in the same k values. This is very intuitive because both methods need to add more perturbations to handle the hard tasks.