Compositional Targeted Multi-Label Universal Perturbations

Supplementary Material

7. Derivation of CMLU$_\alpha$ and CMLU$_\beta$

In this section, we provide more details of the derivation from the main paper. In particular, we focus on deriving Eq. (7). We begin by considering Eq. (5)

$$\max_{\boldsymbol{U}} \sum_{\Omega\in\mathcal{P}(\mathcal{C})} \sum_{k\in\Omega} \Big(\mathcal{H}_k(\boldsymbol{u}_k + \boldsymbol{U}\boldsymbol{\lambda}_{\Omega\setminus k}) - \frac{\gamma}{|\Omega|} \sum_{q\in\bar{\Omega}} \mathcal{R}_q(\boldsymbol{u}_k + \boldsymbol{U}\boldsymbol{\lambda}_{\Omega\setminus k}) \Big), \quad \text{s.t. } \hat{\boldsymbol{U}}^\top \hat{\boldsymbol{U}} = \boldsymbol{I},\ \|\boldsymbol{U}\|_{1,p} \le \epsilon \tag{12}$$

where $\hat{\boldsymbol{U}}$ is column-normalized $\boldsymbol{U}$ and $\|\boldsymbol{U}\|_{1,p} = \max_{1\le i\le|\mathcal{C}|} \|\boldsymbol{u}_i\|_p$, is the maximum norm of the columns of $\boldsymbol{U}$. Notice that we wrote $\boldsymbol{u}_\Omega = \boldsymbol{u}_k + \boldsymbol{U}\boldsymbol{\lambda}_{\Omega\setminus k}$ for a k that belongs to Ω, where $\Omega\setminus k$ denotes removing k from Ω. The sum in Eq. (12) is across the power set of $\mathcal{C}$.

7.1. Powerset Reformulation

In our derivation, we have two key steps (corresponding to two observations) that involve expressing the powerset of a set in a specific way. We explain those steps below.

Observation 1: *For any $k \in \mathcal{C}$, the subsets of powerset $\mathcal{P}(\mathcal{C})$ that include k can be written as:*

$$\mathcal{P}_k \triangleq \bigcup_{\Omega\in\mathcal{P}(\mathcal{C}\setminus k)} \{\, \{k\}\cup\Omega \,\} \triangleq \{\, \{k\}\cup\Omega \mid \Omega \in \mathcal{P}(\mathcal{C}\setminus\{k\}) \,\} \tag{13}$$

$\mathcal{P}_k$ in Eq. (13) gives all subsets of $\mathcal{C}$ that include k, constructed by taking all the possible subsets of $\mathcal{C}\setminus\{k\}$ and adding k to them. *Based on this, the key observation is that we can combine these contributions ($\mathcal{P}_k$) from all $k \in \mathcal{C}$ to reconstruct the entire powerset $\mathcal{P}(\mathcal{C})$ (excluding the empty set) as*

$$\mathcal{P}(\mathcal{C}) \triangleq \bigcup_{k\in\mathcal{C}} \mathcal{P}_k \triangleq \bigcup_{k\in\mathcal{C}} \bigcup_{\Omega\in\mathcal{P}(\mathcal{C}\setminus k)} \{\{k\}\cup\Omega\} \tag{14}$$

Eq. (14) allows us to leverage the property of decomposing the powerset into a union of sets. This structure reveals the inherent symmetry and compositional nature of the powerset. Therefore, using Eq. (14), we can reconstruct the summation across the power set in Eq. (12). However, (14) introduced redundancy, as each subset is repeated $|\Omega| + 1$ time in the formulation. To account for this, one could multiply the loss expression by a factor of $1/(|\Omega|+1)$, which just scales the loss. So instead, we directly optimize

$$\max_{\boldsymbol{U}} \sum_{k\in\mathcal{C}} \sum_{\Omega\in\mathcal{P}(\mathcal{C}\setminus k)} \Big(\mathcal{H}_k(\boldsymbol{u}_k + \boldsymbol{U}\boldsymbol{\lambda}_{\Omega\setminus k}) - \eta \sum_{q\in\{\bar{\Omega}\setminus k\}} \mathcal{R}_q(\boldsymbol{u}_k + \boldsymbol{U}\boldsymbol{\lambda}_{\Omega\setminus k}) \Big), \quad \text{s.t. } \hat{\boldsymbol{U}}^\top \hat{\boldsymbol{U}} = \boldsymbol{I},\ \|\boldsymbol{U}\|_{1,p} \le \epsilon, \tag{15}$$

where $\eta = \gamma/|\Omega|$. Note the change in the indexing of the second loss term in (15). This adjustment ensures consistency with (12) by enforcing that the sum of non-targeted loss excludes k in our new formulation.

Observation 2: *The power set of $\mathcal{C}$ can be written as the union of subsets of $\mathcal{C}$ of fixed size:*

$$\mathcal{P}(\mathcal{C}) \triangleq \bigcup_{t=0}^{|\mathcal{C}|} \binom{\mathcal{C}}{t} \triangleq \bigcup_{t=0}^{|\mathcal{C}|} \bigcup_{S\in\binom{\mathcal{C}}{t}} \{S\} \tag{16}$$

where, each $S \in \binom{\mathcal{C}}{t}$ is a subset of $\mathcal{C}$ of size t.

In our case, any $S \in \binom{\mathcal{C}}{t}$ would represent the indices of the columns of $\boldsymbol{U}$. Using (16), we write the summation across the power set $\mathcal{P}(\mathcal{C}\setminus k)$ in (15) as

$$\max_{\boldsymbol{U}} \sum_{k\in\mathcal{C}} \sum_{t=0}^{|\mathcal{C}|-1} \sum_{S\in\binom{\mathcal{C}\setminus k}{t}} \Big(\mathcal{H}_k(\boldsymbol{u}_k + \boldsymbol{U}\boldsymbol{\lambda}_S) - \eta \sum_{q\in\{\bar{S}\setminus k\}} \mathcal{R}_q(\boldsymbol{u}_k + \boldsymbol{U}\boldsymbol{\lambda}_S) \Big), \quad \text{s.t. } \hat{\boldsymbol{U}}^\top \hat{\boldsymbol{U}} = \boldsymbol{I},\ \|\boldsymbol{U}\|_{1,p} \le \epsilon, \tag{17}$$

Please refer to the main paper for the rest of the derivation.

8. Analysis of Universal Perturbations

In Fig. 6, we visualize label-wise universal perturbations generated using the Oracle, CMLU$_\alpha$, and CMLU$_\beta$ methods on the NUS-WIDE dataset for the ASL model, along with some of their compositions. Note that the perturbations produced by the Oracle method exhibit significant texture variations across labels, and combining these perturbations often causes the key adversarial features of individual classes to diminish or fade away. In contrast, CMLU$_\alpha$ and CMLU$_\beta$ learn label-wise perturbations with subtle variations, enabling their composition to preserve shared features across labels while de-emphasizing conflicting textures. This demonstrates the effectiveness of CMLU$_\beta$ for attacking multiple classes.

In Tab. 7, we show the pairwise dot products of universals computed using Oracle and CMLU$_\beta$. The purpose of this experiment is to evaluate the difference between the vulnerability directions found through each of the methods. From the table, we can see that CMLU$_\beta$ finds unique directions, different from Oracle.

In Tab. 4 in the main paper, we showed the fooling success rate to evaluate the transferability of universal attacks across model. In Tab. 5, we show non-target flip rate (NT$_R$) values. From both tables, we can see that when CMLU$_\beta$

Table 5. Non-target flip rate (NT_R) of Transferability Experiments using the ASL model.

Transferability of Universal Attacks							
		NUS-WIDE			MS-COCO		
Models	**Method**	**1**	**3**	**5**	**1**	**3**	**5**
A → D	**Oracle**	3.81	3.88	3.65	5.29	8.03	9.45
	CMLU$_\beta$	4.21	3.80	3.29	7.39	8.39	9.04
D → A	**Oracle**	4.22	4.86	-	7.54	9.07	10.07
	CMLU$_\beta$	6.02	0.85	-	10.6	10.9	13.3

achieves better FR than Oracle, it also gets higher NT_R than Oracle. In the case, when it performs worse than Oracle, it has lower NT_R.

9. Wall Clock Time

In Tab. 6, we present the computational complexity and wall-clock times for each method, illustrating their practical efficiency. It is important to highlight that the total time column (T) in the table reflects only a few target sizes (1, 3, 5) and accounts for only a small subset $\mathcal{S}(|\Omega|)$ of all possible combinations of a fixed size $|\Omega|$. For methods like Oracle and Or-C, these times would increase significantly with larger target sizes.

From the table, we make the following observations:

- The computational time of Oracle and Or-C continues to increases as additional target sizes are considered, making them less practical for large datasets and target sizes. In contrast, our methods, CMLU$_\alpha$ and CMLU$_\beta$, require training only for $|\Omega| = 1$, and their computational costs remain constant regardless of the number of target sizes.
- Note that CMLU$_\alpha$ is more computationally expensive than CMLU$_\beta$. This cost is due to the inner minimization step involving N_α iterations.
- Or-S and CMLU$_\beta$ have the same time complexity and wall clock time. But as we have shown in the main paper, CMLU$_\beta$ achieves performance close to exponential time Oracle method while Or-S scales poorly to large target sizes.

10. Hyperparameters and Experiment Setting

In algorithm 1, we mainly have five hyperparameters: $\epsilon, \eta = \frac{\kappa}{|\Omega|}$, α, N_α, and ξ. In our experiments, we set $\kappa = 20$ for NUS-WIDE and OpenImages experiments and $\kappa = 10$ for MS-COCO. In all of our main experiments, we use $N_\alpha = 10$, $\alpha = 1e2$, $\epsilon = 0.05$, and $\xi = 0.002$. We set the random seed for pytorch and numpy to 999. We use the officially provided train/val split to train the models and test split to evaluate the attacks. We used batch size of 40 in our experiments. All images were resized and center-cropped to 448 x 448.

11. Generating Convex Subcones

The purpose of Fig. 5 in the main paper is to demonstrate why our assumption (4) works with CMLU but fails with other methods. The figure compares the input regions that Oracle and CMLU$_\beta$ have learnt. In the figure, Oracle plot shows the class-specific universals learnt using Oracle for $|\Omega| = 1$.

Consider a multi-label classifier $\mathcal{F} : \mathbb{R}^d \rightarrow \mathbb{R}^{|\mathcal{C}|}$ where $\mathcal{C}$ is the set of all labels. The classifier $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{|\mathcal{C}|}\}$ consists of $|\mathcal{C}|$ binary classifiers (one for each label), where $\mathcal{F}_c(\boldsymbol{x}) \in (-\infty, +\infty)$ represents the *logit* of label c.

- Assume $\boldsymbol{U}_{\text{Or}}$ represents $|\mathcal{C}|$ universals learnt using Oracle Eq. (1), where each universal vector $\boldsymbol{u}_k \in \boldsymbol{U}_{\text{Or}}$ attacks the specific class k.
- Assume $\boldsymbol{U}_\beta$ represents $|\mathcal{C}|$ universals learnt using CMLU$_\beta$ Eq. (10), where each universal vector $\boldsymbol{u}_k \in \boldsymbol{U}_\beta$ attacks the specific class k.

First, we plot the average model confidence for all possible pairs of vectors from $\boldsymbol{U}_{\text{Or}}$ using (18) and then we plot the average model confidence for all possible pairs of vectors from $\boldsymbol{U}_\beta$. We consider a set $\mathcal{Q} = \binom{\mathcal{C}}{2}$ i.e., all possible pairs of classes. Assume $\mathcal{I}_\Omega$ are the clean images that contain classes Ω. Then, the value at each point (i, j) in the 2D plot in Fig. 5 is computed as:

$$\mathbb{C}(i,j) = \frac{1}{|\mathcal{Q}|}\sum_{\Omega\in\mathcal{Q}}\frac{1}{|\mathcal{I}_\Omega|}\sum_{x\in\mathcal{I}_\Omega}\frac{1}{|\Omega|}\sum_{k\in\Omega}\sigma(\mathcal{F}_k(x + v_\Omega(i,j)))$$
$$\text{where, } v_\Omega(i,j) = \Pi_\epsilon(i * u_{\Omega_1} + j * u_{\Omega_2}) \tag{18}$$

for $i, j \in [0, 1]$. Π_ϵ projects the input on ϵ infinity norm ball and σ is the sigmoid function. Note that the value $\mathbb{C}(i, j)$ is computed across all target classes $\mathcal{Q}$ and all images $\mathcal{I}_\Omega$ for a point (i, j). This renders the cross-section of the input space to visualize the universal adversarial regions. Note that the vectors are not mutually sparse. We clip the cone at 0.05 infinity norm.

While summing universals learnt using (1) fails to successfully attack (no red regions), CMLU$_\beta$ learns universals whose sums effectively attack target labels (red regions). This validates that our framework can learn compositional universal perturbations. Furthermore, the figure provides intuitive 2D insights into MLL model behavior.

12. Limitation

A limitation of our work is the assumption of label independence. In MLL, several labels can be correlated, such as "Cat and Animal" or "Beach and Ocean". This correlation results in a tightly coupled FR and NT_R (reducing NT_R would also reduce FR e.g., if we want to attack Cat but keep Animal unchanged). This was also shown in our ablation experiments. Similarly, for high FR, we see an increased NT_R for larger target sizes. To address this, future

Table 6. Complexity and wall clock time of the methods. We show the approximate average time (in hours) for a single training iteration of the methods on an NVIDIA-V100 32GB GPU. The values are rounded off to the nearest integer. For a given target set size $|\Omega|$, $\mathcal{S}(|\Omega|)$ is the number of selected sets or label combinations of size $|\Omega|$. The column **T** is the sum of the columns ($|\Omega| = 1, 3, 5$) to show the total compute time across different targets.

	Complexity		NUS-WIDE				MS-COCO				OpenImages			
$\|\Omega\|$	Space	Time	1	3	5	**T**	1	3	5	**T**	1	3	5	**T**
$\mathcal{S}(\|\Omega\|)$			25	40	40	105	40	50	50	140	100	100	100	300
Oracle	$\mathcal{O}(d\,2^{\|C\|})$	$\mathcal{O}(E\,2^{\|C\|})$	11	18	18	47	18	23	23	64	47	47	47	141
Or-S	$\mathcal{O}(d\,\|C\|)$	$\mathcal{O}(E\,\|C\|)$	11	-	-	11	18	-	-	18	47	-	-	47
Or-C	$\mathcal{O}(d\|C\|+2^{\|C\|})$	$\mathcal{O}(E\,2^{\|C\|})$	11	17	17	45	18	22	22	62	47	43	43	133
SGA	$\mathcal{O}(d\,\|C\|)$	$\mathcal{O}(E\,\|C\|)$	23	-	-	23	31	-	-	31	112	-	-	112
NAG	$\mathcal{O}(G_p)$	$\mathcal{O}(E\,\|C\|)$	23	-	-	23	37	-	-	37	92	-	-	92
CMLU$_\alpha$	$\mathcal{O}(d\,\|C\|)$	$\mathcal{O}(EN_\alpha\|C\|)$	28	-	-	28	38	-	-	38	137	-	-	137
CMLU$_\beta$	$\mathcal{O}(d\,\|C\|)$	$\mathcal{O}(E\,\|C\|)$	11	-	-	11	18	-	-	18	48	-	-	48

Table 7. The table shows mean dot product values of the normalized perturbation vectors computed across all target class combinations for each target size $|\Omega|$. The perturbations are learnt on NUS-WIDE, using ASL and ML-Decoder models.

		Target 1				**Target 3**				**Target 5**			
		ASL		**ML-Dec**		**ASL**		**ML-Dec**		**ASL**		**ML-Dec**	
		Oracle	**CMLU**$_\beta$	**Oracle**	**CMLU**$_\beta$	**Oracle**	**CMLU**$_\beta$	**Oracle**	**CMLU**$_\beta$	**Oracle**	**CMLU**$_\beta$	**Oracle**	**CMLU**$_\beta$
ASL	**Oracle**	1.0	0.005	0.001	0.0	1.0	0.004	0.001	0.0	1.0	0.004	0.002	0.0
	CMLU$_\beta$	0.005	1.0	0.001	0.0	0.004	1.0	0.0	0.0	0.004	1.0	0.001	0.0
ML-Dec	**Oracle**	0.002	0.001	1.0	0.005	0.001	0.0	1.0	0.003	0.002	0.001	1.0	0.004
	CMLU$_\beta$	0.0	0.0	0.005	1.0	0.0	0.0	0.003	1.0	0.0	0.0	0.004	1.0

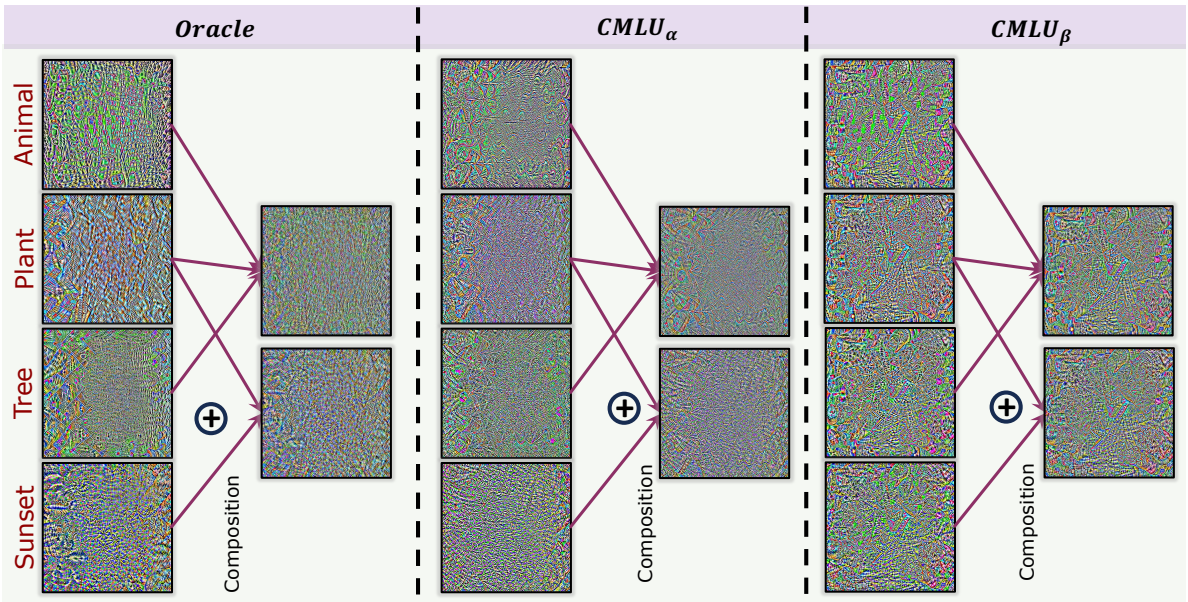

Figure 6. Visualizing the universal perturbations and their compositions, generated using various methods on the NUS-WIDE dataset for the ASL model. The arrows show which label-wise perturbations have been composed together to construct a new universal vector.

works could investigate the integration of label correlation and using a soft loss penalty on non-targeted labels.