Champagne Taste on a Beer Budget: Better Budget Utilisation in Multi-label Adversarial Attacks

1st Erwin van Thiel
Computer Science
Delft University of Technology
Deflt, Nederland

***Abstract*—Multi-label classification is an important branch of classification problems as in many real world classification scenarios an object can belong to multiple classes simultaneously. Deep learning based classifiers perform well at image classification but their predictions have shown to be unstable when subject to small input distortions, called adversarial perturbations. There are multi-class classifiers, which assign images to a single class, and multi-label classifiers that attribute multiple labels to an image. In multi-class scenarios these adversarial attacks are conventionally constrained by a perturbation magnitude budget in order to enforce visual imperceptibility. In the related studies concerning multi-label attacks there has been no notion of a budget and this results in visible perturbations in the image. In this paper we develop attacks that cause the most severe disruptions in the binary label predictions, i.e. a maximum number of label flips, while adhering to a perturbation budget. To achieve this, we first analyse the applicability of the existing single label attack MI-FGSM on multi label problems. A naive way of using MI-FGSM in a multi-label scenario means using binary cross entropy loss and targeting all labels simultaneously. Our key observations are that targeting all labels simultaneously when restricted to a small budgets leads to inefficient budget use, that all labels have different attackability and also that labels exhibit different correlation structures which influences the combined attackability. Moreover, we show that the loss function determines the optimisation direction through prioritising labels with certain confidence values. We find that there are two different strategies to optimise budget use and propose two distinct methods namely, Smart Loss-function for Attacks on Multi-label models (SLAM) and Classification Landscape Attentive Subset Selection (CLASS). SLAM comprises a loss function that uses an estimate for the potential amount of flips to adapt the shape of the curve, and hence the label prioritisation. CLASS uses binary cross entropy loss but focuses the budget on merely a subset of the labels, which was constructed while considering label attackability and pairwise label correlation. CLASS does have the drawback that it relies on classifier specific heuristics for determining the size of the label subset. We extensively evaluate SLAM and CLASS on three datasets, using two state of the art models, namely Query2Label and ASL. Our evaluation results show that CLASS and SLAM are able to increase the flips given the budget constraint by up to 131% and 61% respectively compared to naive MI-FGSM.**

***Index Terms*—Deep Learning, Multi-label Classification, Adversarial Attacks,**

I. Introduction

In many real world problems we need objects to be labeled. In contrast to multi-class classification in which each object belongs to a single class, there are also often objects that are associated with multiple labels simultaneously. Hence we need to also address multi-label classification. Multi-label classification entails for example detecting different objects in an image [16] or determining the different subjects of a text [14]. An example of multi-label classification in computer vision can be seen in Fig. 1, in which an image shows two objects, namely fire hydrant and person.

Despite the power of deep neural networks in classification tasks, these networks still remain vulnerable to adversarial examples, which are input images that have undergone a slight perturbation such that the output of the classifier on this image is changed according to the malicious intent of the attacker [34]. In multi-class adversarial attacks [26, 29, 10, 21, 23, 22, 27, 1, 46, 44, 45, 25, 41, 37, 22, 38, 2, 28, 15, 39, 12, 35, 17, 33] the goal is to change the prediction of the classifier from one to another class. In multi-label classification however an attack can target any subset of labels, which creates additional attack opportunities. For example an attack that has the goal of altering as many binary class predictions, i.e. flipping as many, labels as possible. This attack is important to understand well because this type of attack could seriously compromise the practical use of a classifier. Furthermore, in order to develop secure and robust models we need to first find and address the greatest vulnerabilities.

The proposed attacks in the related work have drawbacks when used for maximum flip attacks. There is MLA-LP [47], a method that assumes that for a small perturbation the influence on the output is linear and hence uses linear programming for constructing perturbations. There is also ML-DeepFool[32] which relies on the same assumption but uses a pseudo-inverse instead of linear programming to generate a perturbation. Lastly, there is ML-CW[32], which is a gradient descent based optimisation of the perturbation. MLA-LP and ML-Deepfool fail to flip large numbers of labels. On top of that, ML-DeepFool generates large perturbations which are clearly recognizable in the image, so the attack can be detected easily Fig 1 (c). ML-CW[32] generates effective perturbations but grants no explicit control over the perturbation magnitude. The adversarial examples generated by ML-CW for a maximum flip attack thus have large perturbations and consequently show visible artifacts (see Fig 1 (b)). To conclude, the related work has not investigated how to perform maximum flip attacks while adhering to a perturbation budget. In this paper we ad-

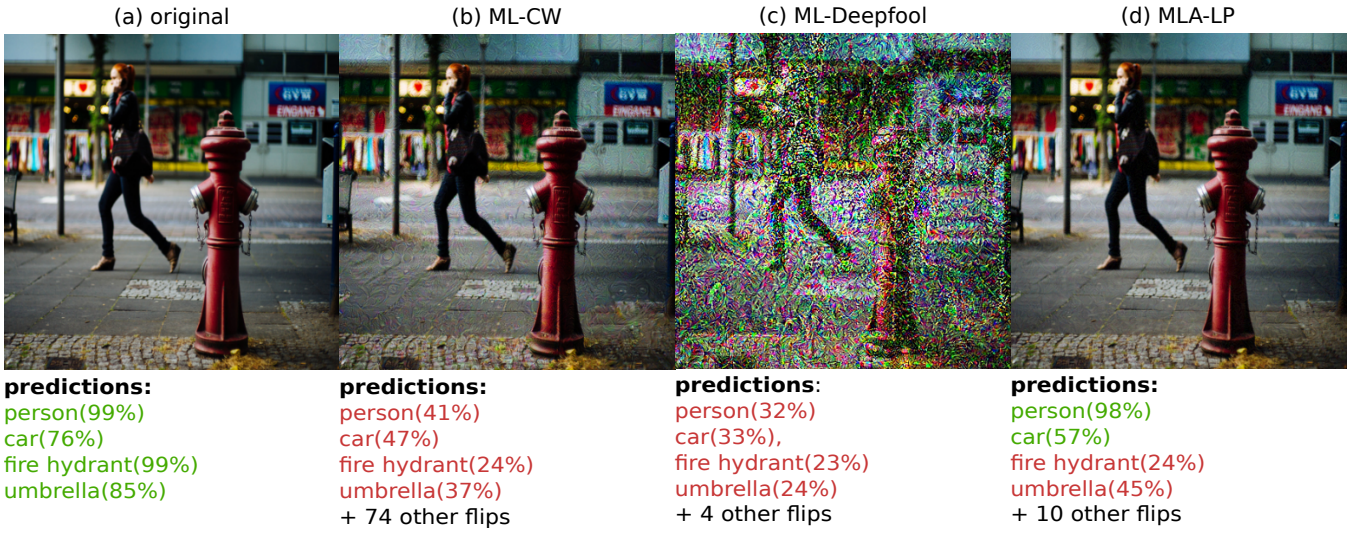

Fig. 1: Adversarial examples of different multi-label attacks from the Query2Label[18] model trained on the MS-COCO dataset [16]. The figure shows the adversarial examples, the confidence values of the originally positively predicted labels before and after the attacks and the number of flips (changed binary predictions). Note that the original image does not contain a car or a umbrella, as the model does not always give a perfect prediction.

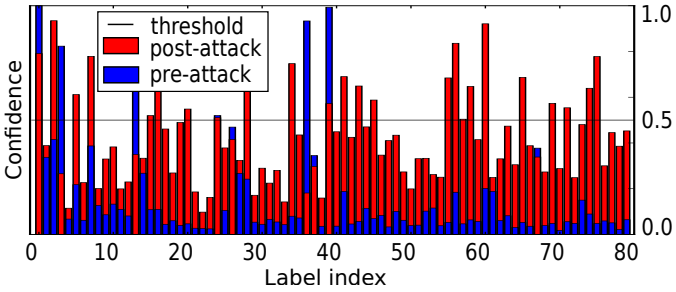

Fig. 2: Example output confidence values of the Q2L model on a MS-COCO data sample before and after attacking with $\epsilon = 0.02$.

dress this unresolved problem and hence focus on perturbation constrained maximum label flip attacks.

In order to tackle the aforementioned problem we use MI-FGSM [6] as the base adversarial attack, because it has a built-in perturbation restriction mechanism. However naive MI-FGSM [9], with binary cross entropy as the default loss function and the entire set of labels as its target, has the problem that it is very inefficient for small budgets. As shown in Fig. 2, we calculate the confidence of each class before and after the attack. After the attack, the confidence of the majority of classes increases significantly, but a large part of the labels ends up just below the threshold. It can be observed that, 14 of the 80 labels end up between 0.4 and 0.5 and cause no flips. Hence the attack does not flip labels effectively when the budget is not large enough to flip all labels.

In this paper, we configure MI-FGSM so that it can use smaller budgets efficiently. We investigate two alternative approaches to better utilize budget in the case of a small budget. Firstly, we propose Classification Landscape Attentive Subset Selection(CLASS). This method uses information about label attackability and pairwise label correlation to select a subset of labels to target. Secondly, we propose SLAM a loss function specifically tailored for the budget constrained multi-label attack optimisation. This method takes the budget and a flip estimator for the target classifier as parameters and accordingly adjusts the shape of the loss function, such that appropriate label prioritisation is used during optimisation. CLASS and SLAM are both methods designed to be efficient when the budget is low. The difference is that CLASS selects a subset of labels to target explicitly and SLAM targets all labels but uses the shape of the loss curve to assign different priority to different labels during attack optimisation. Furthermore, SLAM has the advantage over CLASS that it does not rely on classifier specific heuristics.

We evaluate our proposed methods on two state-of-the art models, namely ASL [3] and Query2Label [18] and three datasets MS-COCO [16], NUS-WIDE [19] and VOC2007 [7]. We show that we improve upon the naive MI-FGSM baseline by up to 131% and 61% with CLASS and SLAM respectively.

Our contributions are as follows:

- As opposed to prior related work we look at individual labels and introduce the notion of label attackability. Moreover, we analyse the pairwise label correlations that are embedded in the classifier and propose a method to extract these correlations.
- We introduce two novel methods, called CLASS and SLAM that are the first methods specifically designed for budget constrained multi-label adversarial attacks.
- We demonstrate the superior performance of our solutions on two state-of-the-art multi-label models, i.e., ASL and Query2Label, on the MS-COCO, NUS-WIDE and VOC2007 datasets.

II. Background

A. Multi-label classification

The objective of a multi-label classifier is to assign the correct set of labels to a data instance, which is in this case an image. Suppose we have a model F that takes d-dimensional input $\boldsymbol{x} \in \mathbb{R}^d$ and transforms it into a l-dimensional output vector $\boldsymbol{o} \in \{0,1\}^l$ in which l denotes the number of possible labels. As shown in Fig. 3, the output values are transformed to the range $[0,1]$ by a sigmoid function σ at the last layer of the DNN and can then be interpreted as class probabilities, i.e. confidence values, which we denote vector $\mathcal{P}$. A threshold τ is used to decide upon presence/absence of a class. The performance of state-of-the-art multi-label classifiers[18, 3, 5, 36] is conventionally measured in mean Average Precision(mAP), which is the precision averaged over a range of thresholds and then averaged over multiple samples.

As target model of our attacks we employ one of the state-of-the-art models proposed by [3]. This model is trained with an asymmetric loss function known as ASL. ASL addresses the problem of imbalance between positive and negative labels in multi-label datasets. It does so by amplifying the impact of positive labels, attenuating the impact of negative labels on the loss and also nullifying the influence of easy negative labels on the loss during training.

In addition to ASL, we also consider Query2Label [18], a transformer based classifier that leverages transformer decoder structures to query the presence of certain labels. This model consists of a backbone network that extracts spacial features from the input image. Then a cascade of transformer-decoder blocks take these spacial features and label representations and perform cross-attention to query the presence of the labels. This model uses a simplified version of the loss function proposed in [3], it however omits the nullifying of the easy negatives.

We choose these two models because they achieve state-of-the-art performance and because they have different architectures, namely convolutional neural network and vision transformer architectures.

B. Adversarial Attack

An adversarial attack entails perturbing an input such that when this input is fed to the classifier it causes the classifier to give whatever prediction the attacker intends. An essential part of the attack is that the adversarial example is indistinguishable from clean images by human eyes. In order to ensure this, the perturbation must be bounded in magnitude. In other words, the adversarial objective is to change the output while limiting the change in the input. In multi-class attacks the objective is simply to change the predicted class. In multi-label classification however the labels are not mutually exclusive so any label can be flipped from 0 to 1 (up) or from 1 to 0 (down). In this work we only concern ourselves with white-box attacks, which are attacks on classifiers of which the model parameters are accessible.

An example of a white-box attack is FGSM [9]. A loss function $\mathcal{L}$ is used to compute the loss of the network by comparing the output to the target t. FGSM then calculates perturbation r by taking the gradient of this loss function with respect to the input image $\boldsymbol{x}$, applying the sign function to it, negating it and then scaling it with a step size factor α (see equation 1).

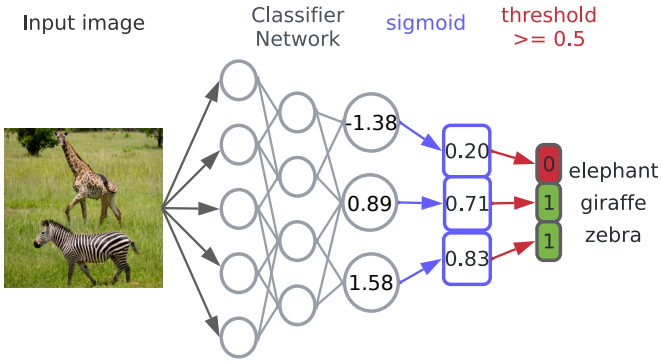

Fig. 3: An example of multi-label classification for computer vision. The pixels of the image are passed into the network, which generates output values for each possible class. Every output is tuned into a confidence value by applying a sigmoid function to it. The confidence values are then turned into binary predictions using a threshold.

$$\boldsymbol{r} = \alpha \cdot -sign(\nabla \mathcal{L}(F(\boldsymbol{x}), \boldsymbol{t})) \tag{1}$$

The resulting perturbation is then added to the image to create the so called adversarial example. This adversarial example causes a decrease of the loss between the model prediction and the target, which the causes prediction to match the target. The authors of [6] proposed MI-FGSM [6], which performs multiple such steps in an iterative fashion. Each iteration the step of the previous iteration is aggregated in the current perturbation step as an exponential moving average.

MI-FGSM originally implements FGSM with binary cross entropy loss (equation 2). This function comprises a logarithm that is applied to the input. The input is the output of a classifier with sigmoid activation at the end. Two different logarithms, which are reflections of each other, are applied for different cases of target t (either 0 or 1).

$$\mathcal{L}_{BCE}(\boldsymbol{y}, \boldsymbol{t}) = \sum_{i=0}^{l} -w_i(t_i \times ln(y_i) + (1-t_i) \times ln(1-y_i)) \tag{2}$$

III. Related Work

A. Multi-label Classifier Attacks

Song et al. [32] propose a framework that consists of a set of constraints and equations that describe the objective of a multi-label adversarial attack. Thereafter they propose two attacks (ML-CW and ML-Deepfool) to solve for these constraints.

a) ML-CW: The first attack, which is called Multi-label-Carlini&Wagner(ML-CW) attack, comprises Adam optimisation [13] of an adversarial objective function that considers the attack-success and also the perturbation magnitude by way of a regularisation term. This attack generates perturbations that effectively cause flips. There is however no explicit control over the perturbation magnitude, which often causes the resulting adversarial examples to have visible artifacts.

b) ML-Deepfool: The second attack is called ML-Deepfool [21] and is based on the assumption that for a small perturbation the effect on the output is linear. Hence, they generate a perturbation by turning the constraints into a set of linear equations and solving them with a pseudo-inverse. This attack generates very large perturbations which are clearly visible in the image.

c) MLA-LP: Inspired by ML-Deepfool, Zhou et al. [47] make the same linearity assumption but solve the linear constraints not with a pseudo-inverse but with a linear programming engine [24]. This attack generates smaller perturbations but does not achieve a lot of label flips.

We aim to overcome the drawbacks of these existing attacks by designing attacks that allow for generating perturbations with small magnitudes so that they remain imperceptible. Because of this we constrain the attack with a perturbation budget, as opposed to these existing attacks. We focus on flipping as many labels.

B. Attackability Assessments

Yang et al. [43, 42] propose exact adversarial attackability estimators for multi-label classifiers. Xu et al. [40, 48] empirically investigate the robustness of multi-label models to adversarial examples. Melacci et al. [20] propose detecting adversarial examples from the resulting predictions. Using knowledge of co-occurrence relations between labels they are able to detect incoherent predictions. In other words, when labels that usually do not occur together are both predicted positive for a certain image, this image is considered suspicious. In this case detecting an adversarial example serves as a way of defending against it.

IV. Empirical Analysis

1) Problem Statement: We want to construct imperceptible perturbations that cause as many label flips as possible. Therefore in this paper we investigate the budget constrained maximum flip attack, as formalised in equation 3. Assume we have a classifier that takes as input an image with d pixels and outputs confidences for l different possible labels. The formal description of our objective is as follows:

$$\begin{aligned} \underset{r}{\textbf{maximize}} \quad & \Sigma_{i=0}^{l}((F_i(\mathbf{x}+\mathbf{r}) \geq \boldsymbol{\tau}) \oplus (F_i(\mathbf{x})) \geq \boldsymbol{\tau}) \\ \textbf{subject to} \quad & ||\mathbf{r}||_\infty \leq \epsilon \end{aligned} \tag{3}$$

Given the classifier output for label i F_i, image x, perturbation r, logical xor operator $\oplus$, threshold $\boldsymbol{\tau}$ and perturbation bound ϵ, flip as many labels as possible within L_∞ perturbation budget ϵ.

This problem can be envisioned as moving through a d-dimensional space containing l decision boundaries, having to traverse as many of these boundaries as possible, while being limited in travel distance in each dimension.

A. Label Attackabilities

We first investigate which individual decision boundaries can be crossed the easiest. To study attackability of different labels, we attack each label separately with a fixed ϵ value and measure in how many out of a 100 instances we are able to achieve a flip. This flip ratio can be interpreted as attackability. We see in Fig. 4(b) that for the ASL model trained on the MS-COCO dataset that labels have different attackabilities. It can also be observed that in the plots the pre-attack confidence values (Fig. 4a) and the attackabilities show a similar structure. Because of this we examine the correlation between the attackabilities and the pre-attack confidence values. Moreover, we investigate how label confidences relate to training dataset occurrence frequencies. We use Pearson's correlation coefficient[31] to perform a correlation analysis. Table II shows the results of this analysis for different classifiers:

Table II shows the correlations between the pre-attack confidence value, the attackability and the training data set occurrence frequency. The rows depict the concerned correlation and each column shows these correlations for a different model-dataset combination. We separate the flip up cases from the flip down cases.

As can be observed there is a positive correlation between the attackabilities and pre-attack confidence values in the case of a flip-up. This suggests that labels with higher confidence are flipped up more easily. There is however one exception to this observation namely the ASL NUS-WIDE model. This classifier shows no correlation between confidence and attackability. This classifier also outputs very low confidences for each label. This means that the absolute difference between the confidence values of the labels is very small. This small difference apparently does not make a significant difference for the attackability hence there is no significant correlation between pre-attack confidence and attackability. The classifiers show a positive correlation between the training set occurrence frequency and the pre-attack confidence value. This indicates that the bias in the confidences of absent labels originates from the label imbalance of the training dataset.

For the case of flip-downs, there is a negative correlation between pre-attack confidence value and the attackabilities. This indicates that a higher confidence value means that it is more difficult to flip a label down. For the flip-downs there are no significant correlations between the pre-attack confidence value and the dataset occurrence frequencies. When analysing the pre-attack confidence values for positive labels it becomes apparent that the classifiers predict all labels with very high probability. This could be caused by the fact that both classifiers were trained with the same asymmetric loss function, that focuses on true positives. Consequently, the classifiers show more confident predictions for true positives than for true negatives. To conclude, we can make two observations:

	ASL × MS-COCO	ASL × NUS-WIDE	Query2Label × MS-COCO	Query2Label × NUS-WIDE
Flip up				
attackability vs confidence correlation	0.76	-0.05*	0.72	0.73
occurrence frequency vs confidence correlation	0.64	0.57	0.53	0.72
Flip down				
attackability vs confidence correlation	-0.47	-0.31	-0.54	-0.22
occurrence frequency vs confidence correlation	0.1*	0.08*	-0.06*	0.17*

TABLE II: Pearson's correlation coefficients among attackability, pre-attack confidence value and training set occurrence frequency. A score of 1 means positive correlation, -1 means negative correlation and 0 means no correlation. Insignificant correlations ($p > 0.05$) are marked with *.

Observation 1: The classifier inherits the statistics of the training data such that the more frequently occurring labels in the training data are likely to receive a higher confidence value from the classifier when they are absent in the image.

This suggests that the classifier is more prone to predicting the presence of certain labels regardless of the nature of the input. This is an undesirable characteristic of a classifier. In an ideal scenario the classifier would make its prediction solely based on features in the input image.

Observation 2: Labels of which the confidence value is closer to the threshold are easier to flip.

This means we can use the confidence value of a label as an indication of its attackability. Also this could be a sign of a decision space exhibiting linearity. It can easily be derived that when a decision space is linear, so is the decision boundary. We mathematically support this claim in appendix G. Linear decision spaces also mean that the confidence value is proportional to the distance to the decision hyperplane, hence we use the difference between the confidence value of a label and the threshold as an indication of distance of the clean image to the decision hyperplane for this specific label.

B. Label Correlations

Besides individual label attackabilities we aim to extract pairwise label correlations to get a better understanding of the decision landscape. Multi-label datasets contain samples belonging to multiple classes, and the labels can be statistically related to each other. These correlations could be learned by the model and influence the attackability. The pseudo-code of the proposed procedure for extracting these correlations is presented in Algorithm 1.

First we obtain the confidence values of the clean prediction (line 4). Then we attack the image and obtain the adversarial example x'_i (line 5). We then calculate the confidence values of the prediction of the adversarial example (line 6). The average change in confidence values, and thus correlations, of all the labels are then calculated by taking the difference between the clean- and adversarial predictions (line 7). The values are averaged over the n samples. This procedure is carried out for each label and also in two-directions ($target \in 0, 1$) to obtain flip-up and flip-down correlation matrices. In our experiments, we set $n = 100$ and for each model we use an ϵ value has the the average label confidence approach the target (0.001, 0.005, 0.005 and 0.008 for ASL MS-COCO, ASL NUS-WIDE, Query2Label MS-COCO and Query2Label NUS-WIDE respectively). Fig. 5(c) represents the positive

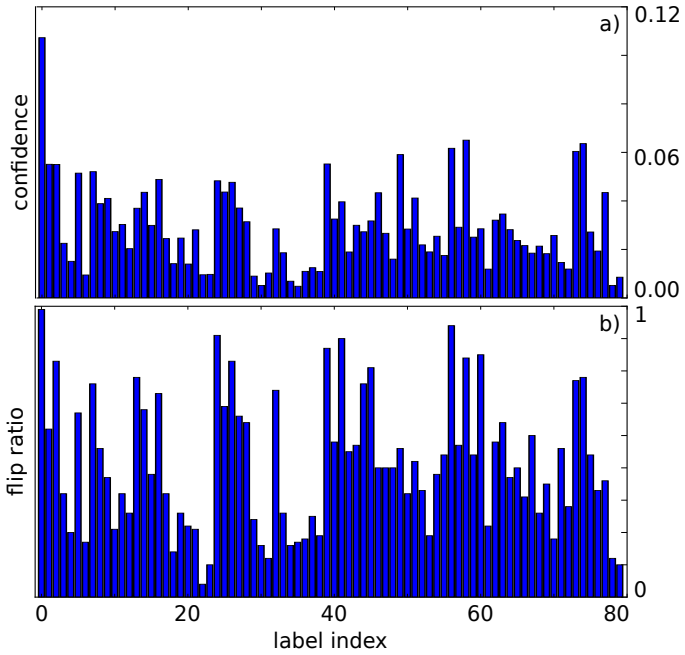

Fig. 4: Per label flip ratio i.e. attackability plot for ASL MS-COCO a) and per label average predicted confidence on image (also from ASL MS-COCO) that does not belong to the concerned class).

Algorithm 1 Correlation extraction

Inputs: Classifier model $F : \{d \times d \rightarrow l \times l\}$, Perturbation bound ϵ and a tensor $x \in \mathbb{R}^{n \times d \times d}$ containing n images with $d \times d$ pixels.

Output: a $l \times l$ correlation matrix $\mathcal{Z}$

1: $\mathcal{Z}$ = zeros(l, l)
2: **for** $target = 1, 2, \ldots\ l$ **do**
3: **for** $i = 1, 2, \ldots$ n **do**
4: $\mathcal{P}_i$ = σ(F($\boldsymbol{x_i}$))
5: $\boldsymbol{x'_i}$ = attack(F, $\boldsymbol{x_i}$, t, ϵ)
6: $\mathcal{P}'_i$ = σ(F($\boldsymbol{x'_i}$))
7: $\mathcal{Z}[target]$ += $\frac{\mathcal{P}'_i - \mathcal{P}_i}{n}$
8: **end for**
9: **end for**
10: return $\mathcal{Z}$

correlations of the Q2L model trained on the MS-COCO dataset. When comparing 5(c) to the co-occurrence frequencies of the dataset (5(a)), it becomes apparent that the model learns the correlations of the training dataset. Hence, we make the following observation:

Observation 3: When a pair of labels occurs often in the training dataset simultaneously, their confidence values are likely to positively correlate during an attack.

Fig. 5(b) illustrates the negative correlations, which are obtained by flipping the target labels down. The inverse patterns visible in between (c) and (b) indicates that the flip-up and flip-down correlations are negatives of each other. Moreover, this allows us to use the inverse of the positive correlations so that we do not have to generate correlation matrices in both directions. Fig. 5(d) shows the same correlations as in Fig. 5(c), these correlations are however obtained with a dataset that was not used for training. The resemblance between (c) and (d) explains that the correlations are ingrained in the model itself. This means the attacker can extract the correlations without knowing the training dataset. The correlation heatmaps of the other models and the cost of generating these matrices are presented in appendix E and appendix F respectively.

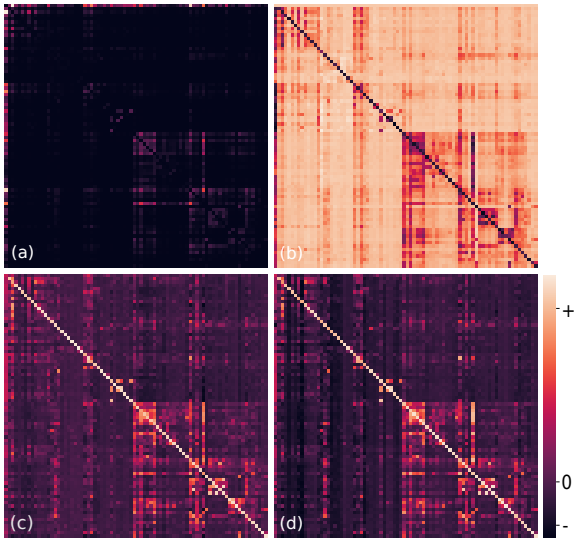

Fig. 5: Co-occurrence heatmap for MS-COCO dataset (a), positive correlation heat map (b), negative correlation heat map (c) and positive correlation heatmap obtained with different dataset (d) for Query2Label model trained on MS-COCO.

Strong correlation between two labels can be interpreted as strong decision hyperplane alignment. This alignment influences the combined attackability of labels. In order to visually explain this we provide a illustration. In Fig. 6(a) we see a point x and decision hyperplanes a,b and c. Adversarial example $x'(a,b)$ is constructed by crossing the hyperplanes of a and b and equivalently $x'(a,c)$ by crossing the hyperplanes of a and c. All three decision hyperplanes are on the same l_∞ distance from x, namely ϵ_1. The difference in alignment of the hyperplanes the pairs (a,b) and (a,b) however causes that $x'(a,b)$ is within l_∞ distance ϵ_2 from x and $x'(a,c)$ is not. This shows that hyperplane alignment is a relevant factor during adversarial attacks.

C. Targeting a Subset

A possible solution to the inefficient budget use is limiting the amount of targets. In Fig. 2 we observe that standard MI-FGSM, which targets all labels simultaneously, wastes its epsilon budget on certain labels. It increases the confidences of certain labels significantly yet insufficiently for causing them to flip. This indicates that distributing the budget over all labels can be sub-optimal when the budget is too limited. This insight raises the question, if we cannot flip all labels, how many and what labels should we target? Intuitively this problem can be solved by targeting only a subset of labels as opposed to targeting them all. In order to gain more insight into this we plot the amount of flipped labels against the amount of targeted labels while attacking in Fig. 7. Targeting labels is counter-intuitively achieved not through the target vector but through the label weight vector in the loss function (see equation 2). By putting 1 as the weight w_i for a targeted label and 0 for one that is not targeted, the optimisation becomes indifferent to the labels that are not targeted. The results in Fig. 7 confirm that targeting too many labels can result in sub-optimal performance.

In order to better understand how this happens we provide a visual example. Fig. 6 (b) shows the perturbation boundaries, the label decision hyperplanes and the attacked image x. If the attack targets both labels ($x'(a,b)$), the image is perturbed towards the cross-section of the hyperplanes but ends up at the border of the L_∞-box. This means that the attack on two labels simultaneously results in 0 flips. If the attack targets either one of the labels individually ($x'(a)$ or $x'(b)$), it will cross the hyperplane perfectly fine and cause 1 flip.

It would be expected that the optimal number of targets is equal to the potential number of flips. It can however be observed that for all four models the maximum number of flips $flips_{opt} \approx 0.6 \times t_{opt}$, in which t_{opt} is the optimal number of targets. We only flip a subset of the labels we target. When increasing our number of targets, some labels we add as target are not causing extra flips. This can be explained by the fact that the confidence is not a 100% accurate indication of distance to the decision hyperplane and also by decision hyperplane alignment between labels. Because of this the peaks of the graphs in Fig. 7 move to the right. Consequently, there is an optimum somewhere between targeting too many and targeting too few labels.

D. Implicitly targeting through the loss function

The loss function that is used for the attack optimisation greatly influences how the attack budget ϵ is distributed over the labels. A key part in the perturbation step calculation is computing the gradient of the loss with respect to the input image $\boldsymbol{x}$. The total loss comprises the sum of individual label losses. This means that when the gradient is calculated, the gradient of the loss is a sum of gradients of individual label

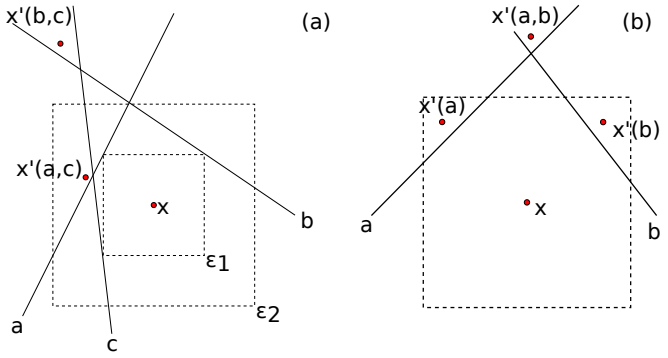

Fig. 6: 2D visualisation with hyperplanes a,b,c, L_∞ perturbation box and attack subject x. (a) shows that the distances to the decision hyperplane do not solely determine the combined attackability but alignment is a factor that contributes to it as well. In (b) it can be observed that trying to flip both a and b simultaneously results in fewer flips than attacking either of them separately. This shows attacking more labels could result in fewer flips.

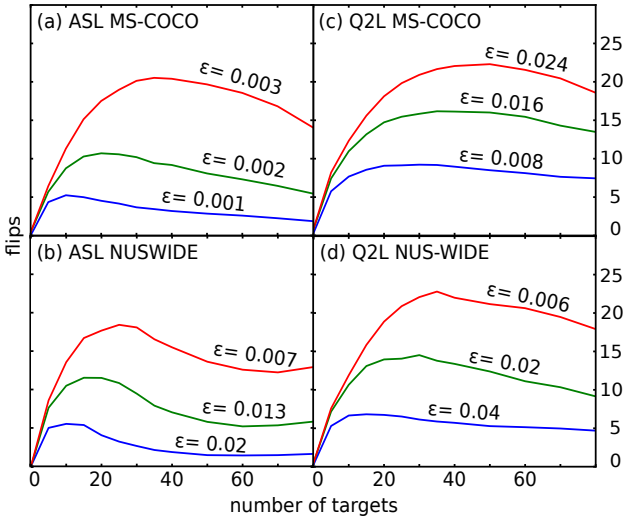

Fig. 7: Measured number of flips versus number of targets for different classifiers and different budgets. for the targets we used the n-targets with confidence closest to the threshold. The results are averaged over 100 samples.

losses. One of these individual label losses comprises the loss function applied to the model output. The derivative of this then according to the chain-rule evaluates to the derivative of the loss function applied to the model output, times the gradient of the model output. As a consequence, if the loss function has a larger derivative at the confidence value of a certain label then this label has a larger influence on the perturbation direction. This idea is formalised in equation 4.

$$\begin{aligned}\mathcal{L}(F(\boldsymbol{x}), \boldsymbol{t}) &= \sum_{i=0}^{l} \mathcal{C}(F_i(\boldsymbol{x}), t_i) \\ \nabla\mathcal{L} &= \sum_{i=0}^{l} \frac{\partial \mathcal{C}}{\partial F_i} \nabla F_i \\ &= \sum_{i=0}^{l} weight_i \cdot \boldsymbol{direction_i}\end{aligned} \tag{4}$$

Given total loss function $\mathcal{L}$, individual label loss function $\mathcal{C}$ and classifier F. The gradient of the loss with respect to input $\boldsymbol{x}$ is a weighted sum of the gradients of all l labels. The weight of the gradient of each label i in the summation is equal to the derivative of the loss function evaluated at the confidence value of that label i, taking into consideration the target t.

It can be derived when $\mathcal{L}_{BCE}$ is used that before the threshold the loss approaches linearity and hence a constant derivative(see appendix K). From equation 4 it then follows that the priority of all labels in this confidence region is equal. This is not what we want when there is little perturbation budget, as distributing the priority and thus budget over too many labels means that there is not enough budget per label too cause the label to flip. Note that the notion of a low or high budget depends on the classifier its attackability/robustness.

An alternative approach for smaller budgets is to focus on the decision hyperplanes that are closer by, and thus act more greedily. This means we adjust the loss curve to prioritise labels with confidences closer to the threshold. We decrease the priority, and hence decrease the derivative, when the confidence is further from the threshold. In other words, we need a function of which the derivative peaks at the threshold. A function that has this property is the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

.

Because we assume a multi-label model has sigmoid activation at the output, we use a linear loss function. As opposed to $\mathcal{L}_{BCE}$, which applies a logarithm to a sigmoid, we apply a linear function to a sigmoid. Our greedy loss is formulated in 5. In this equation y and t denote the label confidence and target respectively.

$$\mathcal{L}(\boldsymbol{y}, \boldsymbol{t}) = \Sigma_{i=0}^{l} \left(1 - t_i\right) \cdot y_i + \left(t_i\right) \cdot \left(y_i - 1\right) \tag{5}$$

Fig. 8 demonstrates the $\mathcal{L}_{linear}$ and the $\mathcal{L}_{BCE}$ approaches work better with a lower budget, and a higher budget, respectively. In appendix H we provide a 2D simulation that visualizes why this happens.

V. Proposed Multi-Label Attacks

In this section we propose, SLAM and CLASS, two distinct approaches for achieving more labels flips with small perturbation budgets. SLAM is a loss function that adapt its shape, and hence its prioritisation per equation 4, to prioritise easier labels when the budget allows for few flips. CLASS

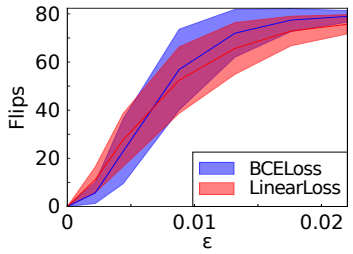

Fig. 8: Mean and std of flips of $\mathcal{L}_{BCE}$ vs $\mathcal{L}_{Linear}$ on the ASL MS-COCO model for different values of ϵ.

comprises selecting a subset of labels based on label attackability and pairwise label correlation. Both these methods rely on estimates of the potential number of flips. Hence lastly, we propose a method for modeling the relationship between the potential flips for a certain budget for a certain classifier.

A. Smart Loss-function for Attacks on Multi-label models (SLAM)

Given an estimate for the potential number of flips, we aim to design a loss function that uses this estimate to calculate a loss curve that exhibits the right amount of prioritisation of nearer decision hyperplanes. We propose a simple approach called Smart Loss-function for Attacks on Multi-label models (SLAM) that comprises a weighted average between the BCE and linear losses as follows:

$$\begin{aligned}\mathcal{L}_{SLAM}(\boldsymbol{y}, \boldsymbol{t}, p, q) = (p \cdot q) \cdot \mathcal{L}_{BCE}(\boldsymbol{y}, \boldsymbol{t}) + \\ (1 - p \cdot q) \cdot \mathcal{L}_{Linear}(\boldsymbol{y}, \boldsymbol{t})\end{aligned} \tag{6}$$

The weight factor consists of $p \times q$, where p (not to be confused with $\mathcal{P}$) is the ratio of labels we can potentially flip. This way, when our expected number of flips is low our curve resembles $\mathcal{L}_{linear}$ and when this number becomes larger our curve starts resembling $\mathcal{L}_{BCE}$ more. Consequently, SLAM always lies between $\mathcal{L}_{BCE}$ and $\mathcal{L}_{linear}$. Furthermore, q is a hyperparameter that regulates the maximum influence of $\mathcal{L}_{BCE}$ in the mix. The use of this parameter and tuning of it is discussed in appendix D. To calculate p, we use the following expression:

$$p = \begin{cases} 1, & \text{if } \epsilon > \epsilon_{max} \\ 0, & \text{if } \epsilon \leq 0 \\ \frac{E(\epsilon)}{C}, & \text{otherwise} \end{cases} \tag{7}$$

where ϵ and ϵ_{max} denote the ϵ-value for the attack and the ϵ-value for which the maximum number of flips is achieved, respectively. C is the number of classes. Also, $E(\epsilon)$ denotes a polynomial function, specifically modeled for the concerned classifier, that returns an estimate of the potential number of flips. This function is elaborated upon in the last part of this section.

B. Classification Landscape Attentive Subset Selection (CLASS)

Given an estimate of how many labels we can flip, we attempt to construct the optimal subset. First we need to determine the length of the subset. According to the experiment in Fig. 7, we find that the optimal number of flips is roughly 0.6 times the number of targets, so we will take $\frac{1}{0.6} = 1.66$ times the number of estimated potential flips as a heuristic. For determining what specific labels we attack we consider two things. Firstly, the confidence values, which are interpreted as distances to their corresponding decision hyperplanes. Secondly, we look at the correlation between the labels, i.e., the alignment of the decision hyperplanes. We aim to solve the optimization problem with the following objective criterion:

$$\mathcal{S} = \gamma \cdot \mathcal{Z} + (1 - \gamma) \cdot \mathcal{P} \tag{8}$$

where $\mathcal{S}$ denotes the score of a label in a certain subset. $\mathcal{Z}$ and $\mathcal{P}$ are the average correlation to the other labels in the set and the confidence value respectively. γ determines the weight between correlation and confidence.

a) Instance correlation matrix: When using label correlations for determining a target subset we need to consider the flip directions for the specific image. For the generating a target subset we thus need to generate a specific attack correlation matrix for each attacked image. Depending on the targets of each label we select rows from the flip-up and flip-down matrices to construct an instance correlation matrix (ICM). We can then obtain the correlations of a certain label to the other labels by looking at the row in the ICM corresponding to that label.

b) Branching Algorithm: The complexity of the problem makes finding an optimal solution infeasible, as this optimization problem is an instance of the NP-hard quadratic knapsack problem [8] (all labels having the confidence as the individual profit, the correlation as the joined profit, the weights are all 1 and the weight limit is the subset length). Because of this we use a branching algorithm. The algorithm entails starting with an empty set and iteratively adding a label until it has reached the pre-defined length. Each iteration we look d steps and b branches ahead to find the optimal label to add in that iteration. This means each iteration we create a tree structure (see Fig. 9) in which the root node holds a set containing the so far added labels. A child node is created by copying the label set of the parent and appending one label. For each parent we compute a ranking of labels using the objective criterion in equation 8 and create b children that append the top b labels in this ranking to their respective sets. The children then become parent nodes and we repeat the same process up until we have constructed a tree of depth d with b children per parent. When the tree has been constructed we apply $\mathcal{O} = \sum_{i=0}^{\mathcal{M}} \mathcal{S}_i$ as the objective function on the label sets in the leaves of the tree to determine the best branch. $\mathcal{M}$ is the length of the subset. In essence, this means looking d steps ahead to calculate what label should be added per iteration. A visual representation of such an iteration is presented in Fig. 9. The leaf node with maximum objective value is regarded as the most lucrative branch. This means that in this example label 30 is added to

the set and in the next iteration the tree will have [0,30] as the root. Note that when there are duplicate combinations the objective values can be the same. In that case the algorithm will choose the first one encountered and consider that as the best branch.

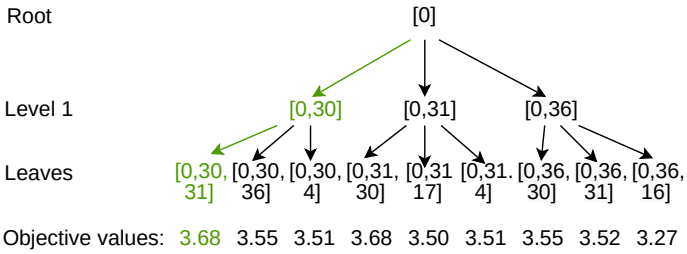

Fig. 9: An example of a tree in a single iteration of the subset generation algorithm with $d = 2, b = 3$. The numbers in the list denote label indices. The green branch represents the most lucrative branch.

C. Creating Model Profiles

As mentioned before, we need to create a function that models the number of potential flips based on the ϵ value. We probe attack and check the number of flips each i-th perturbation step with interval i and stop when the number of flips converges. We used a derivative less than 1% of the derivative in the origin to establish convergence. We attack with both $\mathcal{L}_{Linear}$ and $\mathcal{L}_{BCE}$ and use the maximum of the two as a lower bound estimate. Because we need to query this model we fit a 3rd or 4th-order polynomial to the data points by way of least squares regression. During an attack we can now apply the polynomial function to the attack-ϵ and calculate an estimate for the potential flips. For profile plots and profile cost estimation refer to appendix L.

VI. Evaluation

In this section we evaluate the performance of the proposed attacks. We attack different models trained on different datasets and measure the amount of labels we are able to flip for different budgets.

A. Datasets

MS-COCO: MS-COCO stands for Microsoft Common Objects in Context which is a real-world object detection dataset [16]. The dataset contains 82783 training images, 40504 validation images, and 40775 test images belonging to 80 classes.

NUS-WIDE: This dataset was created by Lab for Media Search [19]. Some of the original samples are however no longer available. For this reason we used the version provided by [3]. This version contains 119103 training images and 50720 validation images. The dataset contains 81 classes.

VOC2007: The Visual Object Classes(VOC2007) [7] dataset consists of 9,963 images from which the test set and training set both take 50%. The dataset includes 20 different categories.

TABLE III: Models with their respective mean average precision performance.

Dataset	Architecture	mAP
MS-COCO	ASL / TResnetL[30]	80.1
MS-COCO	Q2L / Resnet101[11] backbone	84.1
NUS-WIDE	ASL / TResnetL	59.8
NUS-WIDE	Q2L / Resnet101 backbone	65.0
VOC2007	ASL / TResnetXL[30]	65.0

B. Models

The models that are used for the experiments are ASL and Query2Label. For training ASL, we use the default hyper-parameters values provided in [3]. The Query2Label models are downloaded from their original github page. The model architectures and performances are given in Table III. These model architectures are provided in ASL [3] and Query2Label [18], respectively.

C. Experimental Setup

The experiments are all performed with MI-FGSM. The original version of MI-FGSM fixes the number of iterations n and calculates update step size α by $\frac{\epsilon}{n}$. However we analyze wide ranges of epsilon values for different values of α. This lead to poor results when attacking with larger ϵ values, so we fix α and perform more iterations for larger epsilon values. Consequently, we calculate the number of iterations n by $\frac{\epsilon}{\alpha}$. We fix α at $\frac{1}{256*10}$, which is 10 iterations per discrete pixel value. This value is chosen because a higher value will cause that there are too few iterations in the lower epsilon attacks and the lower values would cause the attacks to require a runtime we consider infeasible. For each attack we collect statistics over 100 random samples from the test/validation dataset. For ϵ we take 5, 10, 20, 40, 60, 80 and 100% of ϵ_{max} as determined by the constructed profile. Some examples of these adversarials can be found in appendix M. All classifications are performed by applying a sigmoid to the network output and a 0.5 threshold thereafter to obtain the predictions. We attack while using the different configurations. For the CLASS attacks we will generate target sets with different values of γ namely 0, 0,5 and 1. We also search for the optimal γ value with steps of 0.05 and include the results for the obtained optimal value of γ. For the branching algorithm we will use trees with 4 branches and 4 levels, as larger trees are considered unfeasible. For the SLAM attacks we use $\mathcal{L}_{SLAM}(q = 0.5)$. The baseline attack uses $\mathcal{L}_{BCE}$ and targets all labels. Tables IV, V and VI show the flips. We highlight the best values per column in bold, if they are significant per t-test with $\alpha = 0.05$. Note that * depicts that the subset length for that CLASS attack equals the total number of labels which means that the method and hence also the value do not differ from the baseline. We also evaluate our methods while using L_2 perturbations. The results of this ablation study can be found in appendix. B. We do not compare our results to the related work attacks in the same procedure, simply because

those attacks do not allow for specifying a fixed budget. A separate comparison is provided in table VII.

D. Experiments Results

In Table IV, we present the results of our proposed methods on MS-COCO dataset. The different rows depict different methods and the different columns depict different budget sizes. The values in the cells represent the mean flips and standard deviation. The last column, denoted with "Total" represents the summation of the other columns. It can be observed that SLAM performs better than the baseline for the lower budgets for both ASL and Query2Label. For CLASS applied to the ASL we see that the best version uses $\gamma = 0$ with a performance increase up to 130% over the baseline. This means that for the ASL MS-COCO model using correlation information in the score calculations grants no performance benefit. The Query2Label model does benefit from the correlations, as the optimal γ value is 0.05. This can be explained by the fact that the Query2Label-MS-COCO model exhibits much stronger correlations than the ASL-MS-COCO model, as can be seen in appendix IV-B.

NUS-WIDE is more challenging than MS-COCO and VOC2007 because it does not only contain tangible objects but also more complex classes such as scenes or events. The results are summarized in Table V. As expected, SLAM achieves higher performance than the BCE baseline. For the Query2Label model, this performance benefit is larger than with the other models. This can be explained by the fact that the robustness of this model strongly limits the number of labels we can flip, which is a scenario that is very suitable for SLAM. For the CLASS methods on both ASL and Query2Label we see that no performance benefit is gained from correlation information. This can be explained by the fact that the NUS-WIDE dataset has very few co-occurrences among the classes and thus the models hardly exhibit correlations. For the ASL model, the performance differences between different CLASS methods and the random subset is very small. This can be explained by the fact that the confidences are a poor indication of label attackability, as seen in Table II.

In order to validate our findings we also evaluate our methods on the ASL model trained on a dataset we did not include in our analysis section namely, the PASCAL VOC2007 dataset. The results are presented in Table VI. From these results we conclude that the functionality of the methods transfers to the VOC2007 dataset as well. The CLASS attacks perform best with CLASS ($\gamma = 0$) achieving up to 94% more flips than the baseline. Also for the ASL VOC2007 classifier there is no benefit to including correlation information. This can be explained by the fact that the VOC2007 dataset has very few co-occurrences among labels. As is the case with the other models, SLAM performs better than BCE overall, with the largest difference for lower budgets.

When comparing SLAM and CLASS we see that overall CLASS has better performance. CLASS has the drawback however that it relies on the accuracy heuristic for the subset size, the accuracy of attackabilities and the correlations. We see for example that for the ASL-NUS-WIDE model, of which the confidences are a less accurate indication of attackabilities and of which the correlations are weak, SLAM outperforms CLASS. This can be explained by the following principle. SLAM uses the derivative of the loss to determine label prioritisation, which means that if after an attack iteration the ranking of the labels regarding their distance to the confidence threshold changes, so does the priority. CLASS on the contrary chooses a subset of labels based on the label confidences before the attack and sticks to this subset.

In table VII we compare the results of the related work to our methods. For the budget we use the perturbation magnitude generated by the related work attack. We observe that our methods, and also MI-FGSM, flip more labels than ML-Deepfool and MLA-LP. We achieve competitive results to the results achieved by ML-CW. Appendix M provides the adversarial examples that were generated by SLAM from the image presented in Fig. 1. It can be observed that SLAM, as opposed to ML-Deepfool and ML-CW, is able to produce adversarial examples that are indistinguishable from the original image.

VII. Discussion and Conclusion

To the best of our knowledge, we are the first study that considers maximum flip attacks while adhering to perturbation budgets in multi-label classification. A perturbation budget is necessary as it prevents visual impairment of the image, which is a crucial aspect of an adversarial attack. First we explained how naive MI-FGSM does not efficiently use a small budget and we explained how different budgets require different targeting. We proposed two strategies, CLASS and and SLAM. The first comprises selecting a subset based on attackabilities and correlations called CLASS. The second approach is a budget aware loss function called SLAM.

We evaluate both our approaches on multiple models and datasets. We show that our methods significantly improve upon the naive MI-FGSM baseline. Only for the Query2Label-MS-COCO model correlations provide a significant benefit, as the other models do not exhibit strong enough correlations. The authors of [42] reason that encouraging the alignment of decision hyperplanes for correlated labels increases the classification performance but also increases the adversarial attackability. Consequently, there exists a trade-off between classifier robustness and performance. For datasets in which more there are more label co-occurrences or for models that are better able to learn these correlations, label correlations will play a bigger role in adversarial attacks.

When comparing SLAM to CLASS, CLASS has slightly better performance overall but SLAM has the advantage that it does not rely on heuristics regarding the optimal number of targets, a number that differs per model. For the datasets used in our experiments SLAM and CLASS achieve on average a maximum performance benefit of 45% and 56% respectively compared to naive MI-FGSM. The best results of SLAM and CLASS increase upon this baseline by up to 61% and 131% respectively.

TABLE IV: Mean and std of the obtained flips for different budgets for the MS-COCO dataset. Cells containing * depict that the value is equal to the baseline value for that column.

	Method	ϵ = 0.001	ϵ = 0.003	ϵ = 0.006	ϵ = 0.012	ϵ = 0.018	ϵ = 0.025	ϵ = 0.031	**Total**
ASL	BCE	4.33 ± 3.47	13.29 ± 8.57	41.71 ± 18.41	70.47± 11.20	77.51 ±4.13	79.34 ±1.71	79.76 ± 0.80	366.41
	SLAM	7.01 ± 4.53	17.01 ± 9.61	43.17 ± 17.00	70.08 ± 10.63	77.04 ± 4.29	79.05 ± 1.86	79.65 ± 0.89	373.01
	Random subset	4.53 ± 3.44	13.71 ± 8.68	41.64 ± 17.70	*	*	*	*	366.96
	CLASS($\gamma = 0$)	**10.00** ± 4.38	**20.39** ± 9.22	44.06 ± 17.07	*	*	*	*	**383.53**
	CLASS($\gamma = 0.5$)	7.07 ± 3.77	16.78 ± 7.90	40.99 ± 16.59	*	*	*	*	371.92
	CLASS($\gamma = 1$)	6.06 ± 3.79	17.03 ± 8.14	41.08 ± 16.58	*	*	*	*	371.25
	Method	ϵ = 0.006	ϵ = 0.012	ϵ = 0.024	ϵ = 0.048	ϵ = 0.072	ϵ = 0.096	ϵ = 0.121	**Total**
Q2L	BCE	9.2 ± 4.95	16.47 ± 6.69	29.13 ± 8.79	48.66 ± 8.85	60.04 ± 8.03	66.96 ± 7.60	70.7 ± 6.32	301.16
	SLAM	**11.18** ± 5.89	**19.67** ± 7.42	32.06 ± 7.78	49.56 ± 7.95	59.07 ± 7.25	65.41 ± 7.02	69.23 ± 6.24	306.27
	Random subset	7.88 ± 4.31	15.17 ± 5.73	27.28 ± 7.48	*	*	*	*	296.69
	CLASS ($\gamma = 0$)	9.39 ± 4.17	17.98 ± 5.92	30.91 ± 8.97	*	*	*	*	304.64
	CLASS ($\gamma = 0.05$)	9.15 ± 4.25	18.04 ± 5.97	33.7 ± 7.79	*	*	*	*	**307.25**
	CLASS($\gamma = 0.5$)	8.6 ± 4.32	17.87 ± 6.01	**32.93** ± 7.22	*	*	*	*	305.76
	CLASS($\gamma = 1$)	8.76 ± 4.60	17.86 ± 5.82	32.75 ± 7.25	*	*	*	*	305.73

TABLE V: Mean and std of the obtained flips for different budgets for the NUS-WIDE dataset. Cells containing * depict that the value is equal to the baseline value for that column.

	Method	ϵ = 0.007	ϵ = 0.014	ϵ = 0.028	ϵ = 0.057	ϵ = 0.086	ϵ = 0.115	ϵ = 0.144	**Total**
ASL	BCE	8.51 ± 8.47	28.33 ± 17.48	51.96 ± 13.48	66.13 ± 10.72	69.21 ± 10.25	68.6 ± 11.48	65.02 ± 13.77	357.76
	SLAM	**12.29** ± 10.76	**31.52** ± 16.68	51.99 ± 12.88	65.49 ± 10.35	69.15 ± 9.49	68.63 ± 11.02	65.68 ± 12.82	**364.75**
	Random subset	8.24 ± 3.97	18.88 ± 10.45	38.58 ± 15.16	*	*	*	*	334.66
	CLASS ($\gamma = 0$)	10.81 ± 3.11	22.33 ± 8.23	39.15 ± 13.63	*	*	*	*	341.24
	CLASS ($\gamma = 0.15$)	10.18 ± 2.52	22.32 ± 7.72	40.42 ± 13.23	*	*	*	*	341.88
	CLASS($\gamma = 0.5$)	9.77 ± 2.55	21.34 ± 8.45	38.83 ± 14.33	*	*	*	*	328.36
	CLASS($\gamma = 1$)	9.57 ± 2.77	21.11 ± 8.43	38.63 ± 15.30	*	*	*	*	328.33
	Method	ϵ = 0.011	ϵ = 0.023	ϵ = 0.046	ϵ = 0.093	ϵ = 0.140	ϵ = 0.187	ϵ = 0.234	**Total**
Q2L	BCE	7.09 ± 2.67	11.29 ± 3.58	20.89 ± 5.37	33.46 ± 6.45	39.12 ± 6.91	41.85 ± 7.73	42.91 ± 8.36	196.60
	SLAM	**11.48** ± 3.57	**17.99** ± 4.83	**27.65** ± 5.32	**38.05** ± 6.07	**43.73** ± 7.07	**46.11** ± 7.69	**44.69** ± 8.72	**229.7**
	Random subset	6.98 ± 2.88	12.04 ± 3.90	20.14 ± 5.23	32.1 ± 6.11	37.81 ± 6.91	40.75 ± 8.14	42.28 ± 8.65	192.10
	CLASS($\gamma = 0$)	8.83 ± 2.04	15.69 ± 2.60	24.83 ± 4.28	35.77 ± 6.42	41.26 ± 7.46	43.89 ± 8.22	44.64 ± 8.92	214.90
	CLASS($\gamma = 0.5$)	8.92 ± 2.56	15.62 ± 2.49	23.94 ± 3.97	33.12 ± 4.97	38.84 ± 6.24	40.85 ± 7.34	42.4 ± 7.77	203.69
	CLASS($\gamma = 1$)	8.77 ± 2.62	15.64 ± 2.78	23.5 ± 4.07	33.1 ± 4.56	38.8 ± 6.09	40.75 ± 6.95	41.7 ± 8.00	202.26

TABLE VI: Mean and std of the obtained flips for different budgets for the VOC2007 dataset. Cells containing * depict that the value is equal to the baseline value for that column.

Method	$\epsilon = 0.0041$	$\epsilon = 0.0082$	$\epsilon = 0.0164$	$\epsilon = 0.0328$	$\epsilon = 0.0492$	$\epsilon = 0.0656$	$\epsilon = 0.0820$	**Total**
BCE	1.16 ± 1.59	4.62 ± 4.08	12.15 ± 4.98	17.77 ± 2.56	19.29 ± 1.14	**19.71** ± 0.68	19.83 ± 0.49	94.53
SLAM	1.56 ± 1.97	5.33 ± 4.13	12.54 ± 4.46	17.85 ± 2.38	19.29 ± 1.08	19.68 ± 0.73	19.84 ± 0.44	96.09
Random Subset	1.82 ± 1.4	4.74 ± 3.0	11.84 ± 4.38	*	*	*	*	95.09
CLASS($\gamma = 0$)	2.26 ± 1.32	**6.17** ± 2.97	**13.22** ± 3.97	*	*	*	*	**98.4**
CLASS($\gamma = 0.5$)	**2.36** ± 1.34	5.66 ± 2.28	11.94 ± 3.81	*	*	*	*	96.67
CLASS($\gamma = 1$)	2.17 ± 1.11	5.25 ± 2.36	12.06 ± 3.93	*	*	*	*	96.16

TABLE VII: Results of the existing multi-label attacks on the Query2Label model trained for the MS-COCO dataset averaged over 100 samples. Cells containing * depict that the attack targets all labels and hence the value equals the baseline.

Method	**Flips**	$\epsilon = \lvert p \rvert_\infty$	**MI-FGSM flips**	**SLAM flips**	**CLASS flips**
MLA-LP	12.03 ± 3.66	0.044 ± 0.023	46.51 ± 8.36	47.58 ± 7.69	*
ML-CW	76.62 ± 2.52	0.32 ± 0.055	73.20 ± 6.25	73.0 ± 6.38	*
ML-Deepfool	8.62 ± 3.71	0.99 ± 0.021	40.82 ± 9.46	43.0 ± 9.65	*

When comparing our methods to the related work it becomes apparent that our methods are more effective than ML-Deepfool and MLA-LP. ML-CW achieves slightly more flips but generates large perturbations, which disqualifies it for the concerned problem. To conclude, our methods are most effective for generating adversarial examples with the goal of flipping as many labels as possible, when there is a constraint on the perturbation budget.

VIII. Limitations and Future Work

First of all the attacks proposed in this work require white-box access to the model, which limits the practical applicability of the attacks. Also, the attacks require expensive information about the classifier namely, the profiles for the potential flip estimates and in the case of CLASS, also the correlation matrices. Also, the CLASS attack depends on a heuristic for the optimal length of the targeted subset. This heuristic could be less accurate for other classifiers. For further research it would be interesting to investigate how to create cheaper flip estimation profiles. Also it would be interesting to see how adversarial examples generated by our proposed methods perform in a black-box setting. To investigate this it could be interesting to perform a cross-model transferability analysis. Furthermore, it could be interesting to use the adversarial examples produced by our techniques to increase the robustness of neural classifiers against adversarial attacks.

References

[1] Anish Athalye, Nicholas Carlini, and David Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples". In: *International conference on machine learning*. PMLR. 2018, pp. 274–283.

[2] Shumeet Baluja and Ian Fischer. "Adversarial transformation networks: Learning to generate adversarial examples". In: *arXiv preprint arXiv:1703.09387* (2017).

[3] Emanuel Ben-Baruch et al. "Asymmetric loss for multi-label classification". In: *arXiv preprint arXiv:2009.14119* (2020).

[4] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *2017 ieee symposium on security and privacy (sp)*. IEEE. 2017, pp. 39–57.

[5] Zhao-Min Chen et al. "Multi-label image recognition with graph convolutional networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5177–5186.

[6] Yinpeng Dong et al. "Boosting adversarial attacks with momentum". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9185–9193.

[7] Mark Everingham et al. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[8] Giorgio Gallo, Peter L Hammer, and Bruno Simeone. "Quadratic knapsack problems". In: *Combinatorial optimization*. Springer, 1980, pp. 132–149.

[9] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[10] Jiangfan Han et al. "Once a man: Towards multi-target attack via learning multi-target adversarial network once". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5158–5167.

[11] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[12] Ameya Joshi et al. "Semantic adversarial attacks: Parametric transformations that fool deep classifiers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4773–4783.

[13] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[14] David D Lewis et al. "Rcv1: A new benchmark collection for text categorization research". In: *Journal of machine learning research* 5.Apr (2004), pp. 361–397.

[15] Yifeng Li et al. "Generative Transferable Adversarial Attack". In: *Proceedings of the 3rd International Conference on Video and Image Processing*. 2019, pp. 84–89.

[16] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[17] Aishan Liu et al. "Perceptual-sensitive gan for generating adversarial patches". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 1028–1035.

[18] Shilong Liu et al. "Query2Label: A Simple Transformer Way to Multi-Label Classification". In: *arXiv preprint arXiv:2107.10834* (2021).

[19] Lab for Media Search. 2022. URL: https://lms.comp.nus.edu.sg/ (visited on 04/14/2022).

[20] Stefano Melacci et al. "Domain knowledge alleviates adversarial attacks in multi-label classifiers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[21] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[22] Seyed-Mohsen Moosavi-Dezfooli et al. "Universal adversarial perturbations". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1765–1773.

[23] Konda Reddy Mopuri et al. "Nag: Network for adversary generation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 742–751.

[24] Mosek. 2022. URL: https : / / www . mosek . com / documentation/ (visited on 04/12/2022).

[25] Muzammal Naseer et al. "Cross-domain transferability of adversarial perturbations". In: *arXiv preprint arXiv:1905.11736* (2019).

[26] Muzammal Naseer et al. "On generating transferable targeted perturbations". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 7708–7717.

[27] Nicholas Papernot et al. "Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples (2016)". In: *ArXiv e-prints* ().

[28] Omid Poursaeed et al. "Fine-grained synthesis of unrestricted adversarial examples". In: *arXiv preprint arXiv:1911.09058* (2019).

[29] Omid Poursaeed et al. "Generative adversarial perturbations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4422–4431.

[30] Tal Ridnik et al. "Tresnet: High performance gpu-dedicated architecture". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 1400–1409.

[31] Scipy. *Scipy documentation*. 2022. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html (visited on 08/02/2022).

[32] Qingquan Song et al. “Multi-label adversarial perturbations”. In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2018, pp. 1242–1247.

[33] Yang Song et al. “Constructing unrestricted adversarial examples with generative models”. In: *arXiv preprint arXiv:1805.07894* (2018).

[34] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[35] Sanli Tang et al. “Adversarial attack type i: Cheat classifiers by significant changes”. In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[36] Fadi Thabtah, Peter Cowling, and Yonghong Peng. “MCAR: multi-class classification based on association rule”. In: *The 3rd ACS/IEEE International Conference onComputer Systems and Applications, 2005.* IEEE. 2005, p. 33.

[37] Desheng Wang, Weidong Jin, and Yunpu Wu. “Generating Adversarial Examples with Image-To-Perturbation Network”. In: *2020 39th Chinese Control Conference (CCC)*. IEEE. 2020, pp. 7055–7060.

[38] Chaowei Xiao et al. “Generating adversarial examples with adversarial networks”. In: *arXiv preprint arXiv:1801.02610* (2018).

[39] Jian Xu et al. “Generating universal adversarial perturbation with ResNet”. In: *Information Sciences* 537 (2020), pp. 302–312.

[40] Mengting Xu et al. “Towards evaluating the robustness of deep diagnostic models by adversarial attack”. In: *Medical Image Analysis* 69 (2021), p. 101977.

[41] Qiuling Xu et al. “Towards feature space adversarial attack”. In: *arXiv preprint arXiv:2004.12385* (2020).

[42] Zhuo Yang, Yufei Han, and Xiangliang Zhang. “Attack Transferability Characterization for Adversarially Robust Multi-label Classification”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2021, pp. 397–413.

[43] Zhuo Yang, Yufei Han, and Xiangliang Zhang. “Characterizing the Evasion Attackability of Multi-label Classifiers”. In: *arXiv preprint arXiv:2012.09427* (2020).

[44] Weijia Zhang. “Generating adversarial examples in one shot with image-to-image translation GAN”. In: *IEEE Access* 7 (2019), pp. 151103–151119.

[45] Yanghao Zhang et al. “Generalizing Universal Adversarial Attacks Beyond Additive Perturbations”. In: *arXiv preprint arXiv:2010.07788* (2020).

[46] Zhengli Zhao, Dheeru Dua, and Sameer Singh. “Generating natural adversarial examples”. In: *arXiv preprint arXiv:1710.11342* (2017).

[47] Nan Zhou et al. “Generating Multi-label Adversarial Examples by Linear Programming”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[48] Nan Zhou et al. “Hiding All Labels for Multi-label Images: An Empirical Study of Adversarial Examples”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8.

APPENDIX A
CONFIDENCE HISTOGRAMS FOR DIFFERENT ATTACK STRATEGIES

The distribution of confidence values before and after an attack with naive MI-FGSM is shown in the histograms in Fig. 10. The histograms show the distribution 100 sample confidences before the attack (Fig. 10 (a) and after the attack (Fig. 10 (b). In Fig. 10 all labels in the bins from 0.5 to 1 have passed the threshold and thus achieved a flip. It can be observed that many labels are being carried to the bins $(0.3, 0.4)$ or $(0.4, 0.5)$ and consequently do not cause flips. Fig. 11 show confidence distributions after attacking a subset (a) and attacking with linear loss (b). It can be observed that for both these attack strategies the distributions of outputs values is different from histogram b) in Fig. 10. More labels remain in the pre-attack region (bins 0 - 0.1) (Fig. 10 a)) and less labels end up in the region before the threshold (bins 0.3 - 0.5). This causes more labels to reach the flip region (bins 0.5 - 1).

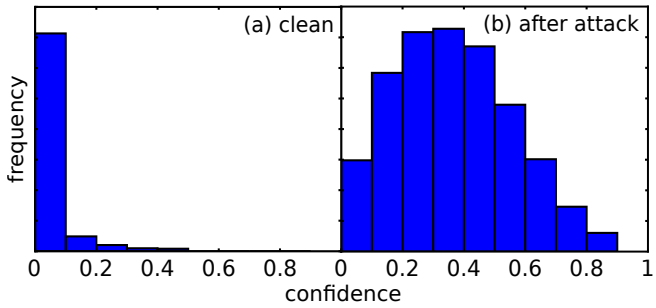

Fig. 10: Histograms of confidence values before attacking a) and after attacking b the ASL MS-COCO model with MI-FGSM BCELoss targeting all labels. Note that this figure regards flipping labels up.

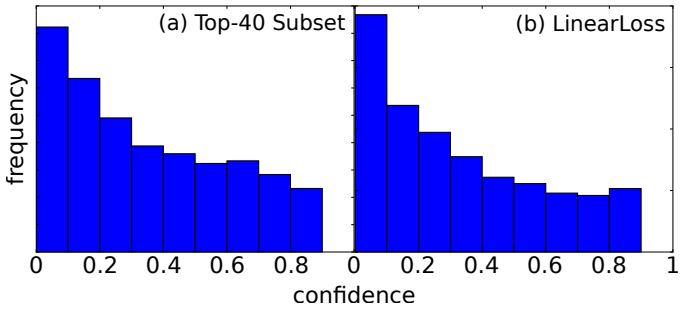

Fig. 11: Histograms of confidence values after attacking ASL MS-COCO model with adapted MI-FGSM. $\epsilon = 0.004$. Both attack approaches yield a similar effect on the confidence distributions.

APPENDIX B
ABLATION STUDY: L_2 PERTURBATIONS

In this ablation study we will use the L_2 norm, alternative to L_∞, to investigate whether our methods work with this type of perturbation bound as well. When using L_∞ norms we search within a hyper-cube around the attacked image. When we use L_2 norms we search within a hyper-sphere around the attacked image. In the L_∞ setting we use the sign of the gradient so that we can enforce the same upper-bound perturbation per dimension. When using L_2 distance however, the allowed perturbation per dimension depends on the other dimensions, as the magnitude is the sum of the squared entries of each dimension. In other words, we can move further in one dimension if we move less in another. Hence, we must abstain from applying the sign function to the gradient, because this forces us to move equal distance in every dimension each step. This creates different opportunities for attackers, since we can now perturb more strongly in smaller regions of the image, as opposed to a smaller perturbation spread out over the entire image. This will intuitively produce stronger perturbations as it allows for focusing on more critical parts of the image, whereas the L_∞-perturbation wastes budget on the less critical parts such as the borders ad corners of the image. These regions contain less objects in practice and the prediction will therefore probably be less affected by perturbations in these areas. In order to investigate these intuitions we perform the same experiment as in Table IV for the Query2Label model, but we change the perturbation metric to L_2 distance. We use steps of the same L_2 magnitude as the attack in Table IV. The results are summarized in Table VIII. CLASS attacks that target all labels, and are hence the same as the baseline are denoted with *.

TABLE VIII: Flips with L_2 perturbations on the Query2Label MS-COCO model.

Method	ϵ = 0.7	ϵ = 1.4	ϵ = 2.8	ϵ = 5.6	ϵ = 8.4	ϵ = 11.2	ϵ = 14.0	total
BCE	5.26 ± 3.68	11.96 ± 5.84	21.02 ± 6.97	38.75 ± 9.54	49.13 ± 8.96	54.15 ± 9.57	55.11 ± 9.46	235.38
SLAM	**6.67** ± 4.14	**13.93** ± 6.17	**24.96** ± 7.61	41.58 ± 7.94	50.28 ± 7.86	55.65 ± 8.49	56.49 ± 8.83	**249.56**
Random subset	4.14 ± 3.34	9.59 ± 4.67	19.29 ± 6.42	36.75 ± 9.11	*	*	*	146.76
CLASS ($\gamma = 0$)	5.62 ± 2.82	11.49 ± 4.36	22.17 ± 6.72	40.15 ± 8.85	*	*	*	237.82
CLASS ($\gamma = 0.05$)	5.48 ± 2.68	11.16 ± 4.26	24.32 ± 5.82	**43.9** ± 8.29	*	*	*	243.25
CLASS ($\gamma = 0.5$)	4.73 ± 3.07	10.73 ± 4.79	24.28 ± 6.27	43.69 ± 8.7	*	*	*	241.82
CLASS ($\gamma = 1$)	4.19 ± 2.72	11.03 ± 4.76	24.14 ± 6.21	43.85 ± 8.03	*	*	*	241.60

Fig. 12: Comparison between an $L_\infty = 0.024$ adversarial (a) and an $L_2 = 9$ adversarial (b) for a Query2Label MS-COCO sample. The L_2 adversarial achieves slightly more flips (52) than the L_∞ adversarial (42) but shows significantly more obvious signs of being tampered with

Also in the L_2 case SLAM outperforms the baseline and as expected achieves most performance increase with the lower budgets. For the CLASS approach we see that the results are not as good as they are with the L_∞ perturbations. This could be explained by the fact that the heuristic used for the subset length was determined for L_∞ optimisations.

We also perform a visual comparison between L_∞ and L_2 perturbations. In order to ensure fairness, we use a hyper-sphere that contains the same volume as the L_∞ hyper-cube. It can be derived (see appendix C) that for an 448×448 RGB the hyper-sphere ϵ (i.e. radius) is approximately 376 times the ϵ of the L_∞ hyper-cube. In the visual comparison we will compare against the adversarial with $L_\infty = 0.024$. The results of the experiment are presented in Fig. 12. Although the L_2 flips slightly more labels, it can be observed that the L_2 perturbation leaves more obvious artefacts in the image. When looking at the adversarials examples in M it becomes apparent that the L_∞ attacks with larger budgets achieve more flips with less visual impairment of the image.

APPENDIX C
HYPER-SPHERE RADIUS DERIVATION

Suppose we have a hyper-cube with half-width ϵ and dimension N, and we want to calculate the radius r of a hyper-sphere that has hyper-volume equal to the hyper-cube. We can calculate this volume V with:

$$V = \frac{\pi^{\frac{N}{2}}}{\Gamma(\frac{n}{2}+1)} r^N = (2\epsilon)^N$$
$$\frac{\pi^{\frac{1}{2}}}{\Gamma(\frac{N}{2}+1)^{\frac{1}{N}}} r = (2\epsilon)$$

in which Γ is Euler's Gamma function. We can accurately approximate the gamma function with Stirling's approximation:

$$\Gamma(n+1) = \sqrt{2\pi n}(\frac{n}{e})^n$$
$$\Gamma(\frac{N}{2}+1) = \sqrt{\pi N}(\frac{N}{2e})^{\frac{N}{2}}$$
$$\Gamma(\frac{N}{2}+1)^{\frac{1}{N}} = \sqrt{\pi N}^{\frac{1}{N}} \times \sqrt{\frac{N}{2e}}$$

This leaves us with:

$$(2\epsilon) = \frac{\pi^{\frac{1}{2}}}{\sqrt{\pi N^{\frac{1}{N}}} * \sqrt{\frac{N}{2e}}} * r$$
$$r = \sqrt{2\frac{(\pi N)^{\frac{1}{N}} * N}{\pi e}} \times \epsilon$$

If $N = 448 \times 448$ pixels $\times$ 3 channels $= 602112$, r evaluates to:

$$r = \sqrt{2\frac{(\pi \times 602112)^{\frac{1}{602112}} \times 602112}{68.31}} \times \epsilon$$
$$= \sqrt{\frac{2 \times 602112}{8.53}} \times \epsilon = 375.73 \times \epsilon$$

APPENDIX D
SLAM Q-PARAMETER TUNING

Parameter q regulates the maximum weight of $\mathcal{L}_{BCE}$ in the mix. This means that q determines the range between $\mathcal{L}_{BCE}$ and $\mathcal{L}_{linear}$ in which SLAM varies, depending on p. Equation 4 tells us that the prioritisation of SLAM is determined by the derivative. Hence we plot the derivative of SLAM in the $t = 0$ case for different weights in 13 (to the $t = 1$ case the same applies, this is the reflected version). In order to analyse the effect of different weights, i.e. different evaluations of $p \times q$, on the prioritisation we fix $q = 1$ and vary p. From this plot it becomes apparent the prioritisation of labels no longer decreases for labels with more distant hyperplanes if the weight is larger than 0.5. Such prioritisation works best when the potential number of flips is high. For this reason it is to be expected that the weight of BCE in the SLAM equation should not become much larger than 0.5. In order to regulate the range of the weight we use hyperparameter q.

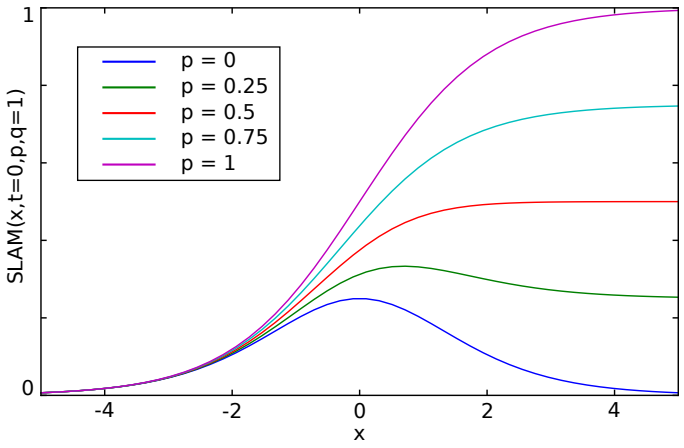

Fig. 13: Derivatives of different parameterisations of SLAM with $t = 0$.

For the tuning of this q parameter we perform the same experiment as in § VI. Table IX shows the flips of the SLAM attack with different q values. It becomes apparent that the overall best value for q is 0.5. The one exception is the Query2Label model on NUS-WIDE. This can be explained by the robustness of the classifier. The model does not allow for flipping many labels which means that a more greedy approach is most effective. The most greedy, and hence most effective, attack is the one with $q = 0.25$ because the weight of the BCE component is restrained the most in this setting.

TABLE IX: Attack results of SLAM with different q values.

Model	SLAM ($q = 0.25$)	SLAM ($q = 0.5$)	SLAM ($q = 0.75$)	SLAM ($q = 1$)
ASL × MS-COCO	372.15	**373.01**	370.99	370.56
ASL × NUS-WIDE	328.84	**364.75**	319.83	318.32
Query2Label × MS-COCO	302.93	**306.27**	305.58	305.31
Query2Label × NUS-WIDE	**245.24**	229.7	233.29	227.25

APPENDIX E
CORRELATION HEATMAPS

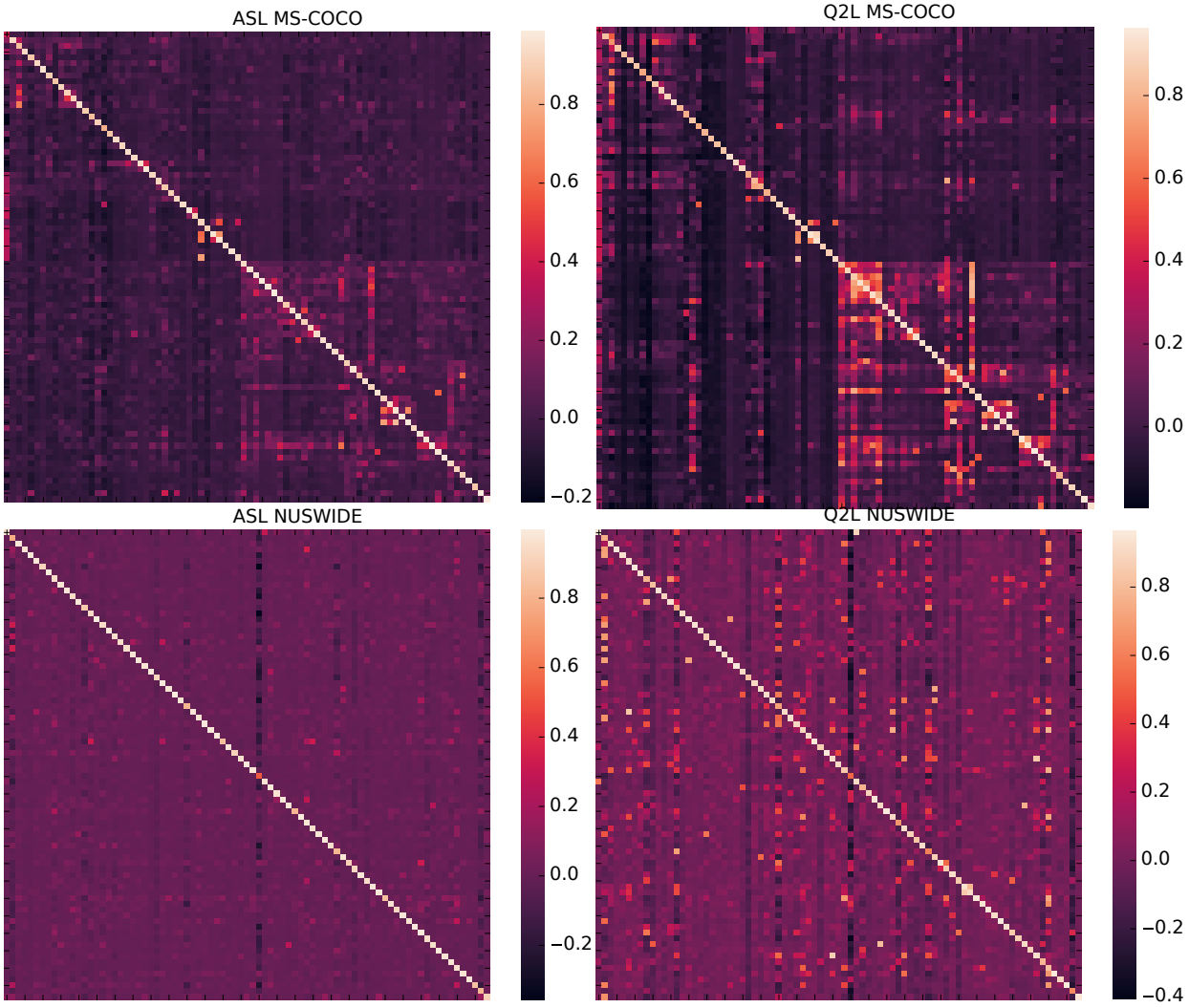

Fig. 14: Positive correlation heatmaps of ASL and Query2Label trained on MS-COCO and NUS-WIDE.

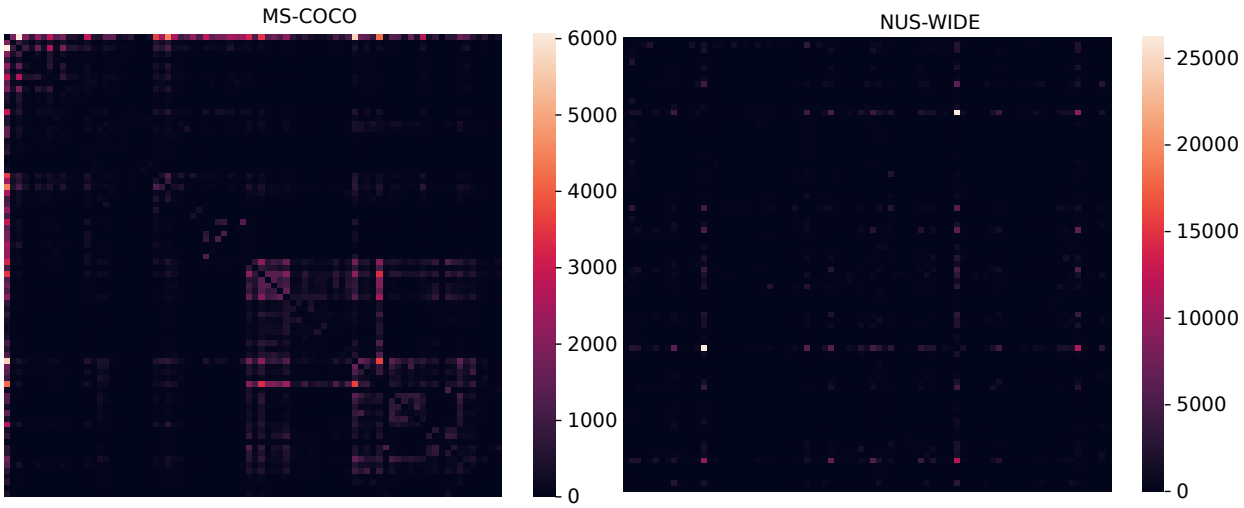

Fig. 15: Co-occurrence frequency heatmaps of MS-COCO and NUS-WIDE.

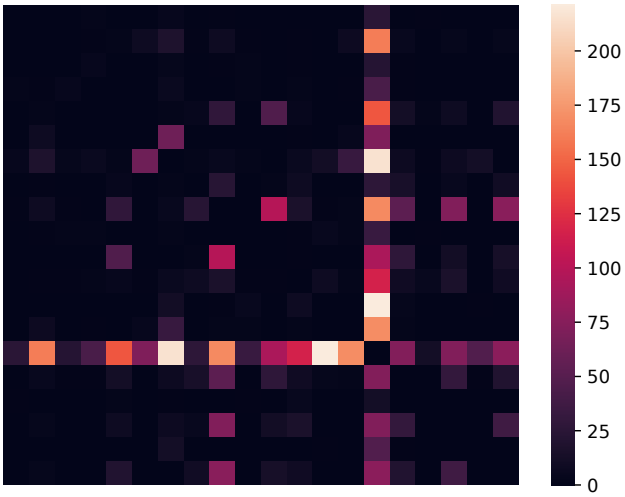

Fig. 16: Co-occurrence frequency heatmaps of VOC2007.

APPENDIX F
CORRELATION MATRIX COST

The cost of generating a correlation matrix can be described by the following equation:

$$Cost = (fp + bp) \times \frac{\epsilon}{\alpha} \times \mathcal{C} \times n \tag{9}$$

In this equation fp and bp denote a forward- and backward pass respectively, ϵ is the perturbation bound used for the attack, α is the attack step-size, $\mathcal{C}$ is the number of classes and n is the number of samples over which the result is averaged.

APPENDIX G
ASSUMPTION OF LINEAR DECISION BOUNDARIES

For building an intuitive understanding and reasoning about solutions for this problem we make use of the belief that the decision space of the classifier is close upon linear. Goodfellow et al. [9] reason that the existence of adversarial examples generated by the one-step FGSM attack serves as evidence that the decision spaces of DNN classifiers are linear. Observation 2 in § IV-A is also in line with this assumption. If a decision space is linear so is the decision boundary, as we will show. A linear decision space can be formalised as a $d-$dimensional hyperplane in a $d+1-$dimensional space in which d dimensions are the input dimensions (pixel values) and there is 1 output dimension $\mathcal{P}$ (label confidence value). We can re-write this in terms of a general hyperplane equation:

$$\mathcal{P} = \sum_{i=0}^{d} a_n \cdot x_n \tag{10}$$

$$0 = \sum_{i=0}^{d} a_n \cdot x_n - \mathcal{P} \tag{11}$$

where a is a scalar weight and x is the input. A decision boundary is a subspace of the decision space where the confidence equals the threshold (in this case 0.5), which means that the confidence becomes a constant:

$$0 = \sum_{i=0}^{d} a_n \cdot x_n - 0.5 \tag{12}$$

This means that we have lost 1 dimension and end up with a $d-1-$dimensional hyperplane. This means that when a decision space is linear, the decision boundary is also linear. This idea is visualised in Fig. 17.

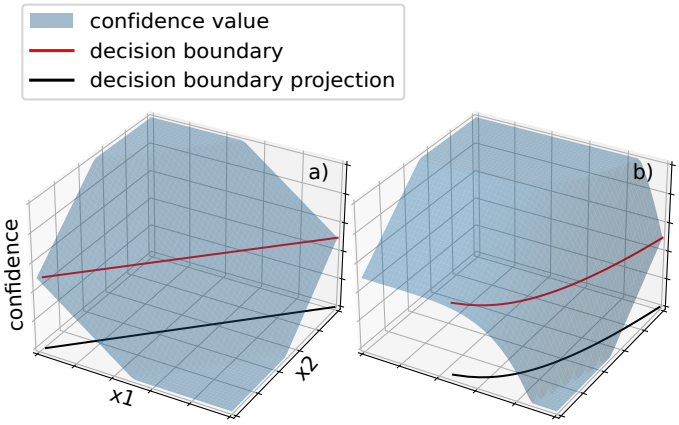

Fig. 17: Visualisation of 3d confidence spaces with 2 decision hyperplanes. a) shows a linear space and b) shows a non-linear space.

These plots both show a decision space with a decision boundary at the threshold. It can be observed in a) that when the decision space is linear, so is the decision boundary. On the contrary, in b) we can see how a decision boundary can be non-linear if the decision space is non-linear. In the 2d visualisations we assume the situation in a) and draw only the projection of the decision boundary in the x,y-plane, which is referred to as decision hyperplane. Using this simplification we can envision the problem as the following problem: In a space containing a set of hyperplanes we want to cross as much of these hyperplanes as possible while staying within a certain distance to the starting point. This way the simplification provides an intuition for better understanding the problem and reasoning about solutions. Note that for solving the problem we make use of MI-FGSM which is a gradient descent based optimisation technique that does not assume linearity.

APPENDIX H
SIMULATION

Fig. 18 shows a 2D simulation of optimisation using both greedy and patient approaches. Note that the optimisation works with L_∞ distance and gradient sign updates. This means that every update step is $\begin{bmatrix}\pm 1\\ \pm 1\end{bmatrix}$. This example can be interpreted as a simulation of an adversarial attack on a 2-pixel image and a decision landscape with 5 decision hyperplanes. The greedy approach enforces that we gravitate more towards the nearest decision hyperplane and the patient approach has us oriented towards the sum of the orthogonal vectors of the decision hyperplanes. As a result we see that in the smaller L_∞ space ϵ_1 the greedy approach gives us more flips and in the larger L_∞ space ϵ_2 the patient approach gives us more flips.

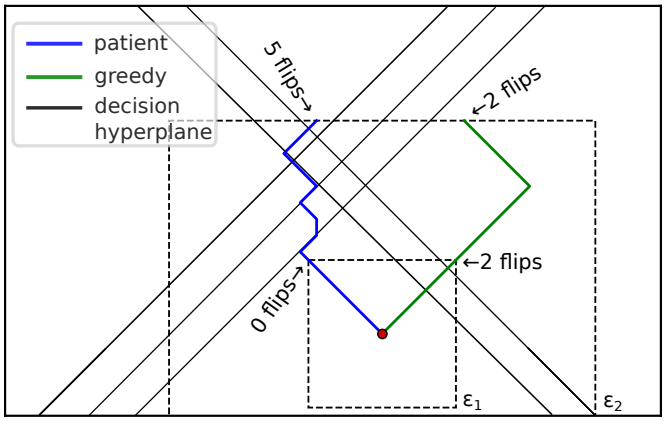

Fig. 18: 2D simulation of optimisation with BCE and sigmoid as loss functions.

APPENDIX I
MI-FGSM (WITH ADAPTATION DISCUSSED IN SECTION VI)

Algorithm 2 MI-FGSM

Inputs: Classifier model $F : \{d \times d \rightarrow l \times l\}$, Perturbation bound ϵ, step size α and a $d \times d$ image.
Outputs: a $d \times d$ image x.

$n = \frac{\epsilon}{\alpha}$
$g = 0$
$\mu = 1$
for $i = 1, 2, \ldots n$ **do**
    $\mathcal{P} = \sigma$**(model(**$\boldsymbol{x}$**))**
    $\mathcal{L} = \mathcal{L}_{BCE}(\mathcal{P}, \boldsymbol{target})$
    $\nabla\mathcal{L} = backpropagation(\mathcal{L}(F))$
    $\nabla\mathcal{L}_{normalised} = \frac{\nabla\mathcal{L}}{\textbf{sum(abs(}\nabla\mathcal{L}\textbf{))}}$
    $\boldsymbol{g} = \mu \cdot \boldsymbol{g} + \nabla\mathcal{L}_{normalised}$
    $\boldsymbol{x} = \boldsymbol{x} - \alpha \cdot g$**.sign()**
end for
$\boldsymbol{x}$ **= clip(**$\boldsymbol{x}$, $\min = 0$, $\max = 1$**)**
return $\boldsymbol{x}$

APPENDIX J
EXISTING MULTI-LABEL ATTACKS

1) ML-CW: This attack is the multi-label version of the well-known Carlini&Wagner-attack [4]. The attack comprises optimisation of the following objective function.

$$\begin{aligned} &Loss = ||r||_\infty + \\ &\lambda\, max(0, (2 \cdot target - 1)(F(\boldsymbol{x} + \boldsymbol{r}) - threshold)) \end{aligned} \tag{13}$$

This objective function penalises for two parts namely, the perturbation magnitude and the loss of the prediction w.r.t. the target. The loss is calculated by the amount the confidence value is larger or smaller than the threshold, depending whether the target is 1 or 0. This optimisation problem can be solved by gradient descent, as we are dealing with a white-box scenario. This means that the problem can be solved using Adam optimiser [13]. The value of lambda is searched for by way of binary search. This means that if no successful attack was performed, the lambda value is multiplied by 10, if success is found, it is divided by 2. The solution that has least hamming distance to the target is decided to be ultimate solution. In case of a tie, the one with the smallest perturbation is picked.

2) MLA-LP: This attack achieves multi-label adversarials through linear programming(MLA-LP). This attack starts with calculating the jacobian matrix, which is the $l \times d$-matrix that holds the partial derivatives of all outputs i.e. labels with respect to all inputs $\boldsymbol{x}$ i.e. pixels. The authors rely upon the fact that a small perturbation has a linear influence on the output. This means that for a perturbation r they can compute the output by Eq. 14:

$$
\begin{aligned}
&given \;\; F(\boldsymbol{x}) = \boldsymbol{y} \\
&\boldsymbol{y'} = \boldsymbol{y} + Jacobian(F, \boldsymbol{x})^{l*d} \quad \boldsymbol{r}^d
\end{aligned}
\tag{14}
$$

In this context an adversarial attack is formulated as the smallest perturbation r for which holds that adversarial prediction y_i' is closer to the target than y_i, for each label i. The authors treat this as a linear programming problem in which the equations are defined as follows:

$$
\begin{aligned}
&\textbf{minimize} \quad \epsilon \\
&\textbf{subject to} \quad \epsilon \geq r_i \;\; \forall \; i \;\; \in [0, d] \\
&\qquad\qquad\qquad\quad \epsilon \leq -r_i \;\; \forall \; i \;\; \in [0, d] \\
&\qquad\qquad\qquad\quad loss(F(\boldsymbol{x} + \boldsymbol{r}), \boldsymbol{y}) + Jacobian(F, \boldsymbol{x})^{l*d} \quad \boldsymbol{r}^d \leq loss(\boldsymbol{\tau}, \boldsymbol{y'})
\end{aligned}
\tag{15}
$$

In this equation loss() denotes the element-wise difference. The equation can be interpreted as follows. Minimise ϵ such that each pixel perturbation r_i is within the budget and such that the computed effect of the perturbation on the output makes the difference between the adversarial output and the target smaller than the difference between the threshold and the target. This linear programming problem is then solved by mosek inner point solver [24].

3) ML-DeepFool: This attack approaches the problem by linearisation of the decision boundaries around x. The authors attempt to solve the following optimisation problem:

$$
\begin{aligned}
&\textbf{minimize} \quad ||\boldsymbol{r}|| \\
&\textbf{subject to} \quad Jacobian(F, \boldsymbol{x})\boldsymbol{r} > 0.5 - F(\boldsymbol{x})
\end{aligned}
\tag{16}
$$

This equations translates to the following: The perturbation on the input multiplied by the Jacobian gives us the change in the output per label. We aim the change to be larger than the difference between the prediction and the threshold. This expression works for when the initial prediction is 0 and the target is 1. Note that depending on the initial values of the initial prediction and target the comparison operator has to be flipped. The authors treat the linear constraints as a system of undetermined linear equations. ML-DeepFool solves this system in a greedy way by using a pseudo-inverse of the Jacobian matrix. This optimization does not necessarily converge so the authors limit the amount of iterations and at the end pick the solution that satisfies the most constraints.

APPENDIX K
LOSS DERIVATIVES

In order to determine label prioritisation during optimisation we analyse the derivative of $\mathcal{L}_{BCE}$. In multi-label classification, the predictions are performed using a threshold preceded by a sigmoid. This sigmoid function influences the loss derivative and for this reason we need to analyse the application of the loss function on the sigmoid. We know that the threshold after applying the sigmoid is 0.5, so we know that the network outputs have an effective threshold of $\sigma^{-1}(0.5) = 0$. Because of this we want to analyse the derivative on either sides of the y-axis. We also need to take into account that the target can be either 0 or 1. In the case of $t = 1$ the loss evaluates to:

$$
\begin{aligned}
&\mathcal{L} = -ln(\frac{1}{1 + e^{-x}}) = ln(1 + e^{-x}) \\
&\lim_{x \to \infty} = ln(1) = 0 \\
&\lim_{x \to -\infty} = ln(e^{-x}) = -x
\end{aligned}
\tag{17}
$$

When $t = 0$:

$$
\begin{aligned}
&\mathcal{L} = -ln(1 - \frac{1}{1+e^{-x}}) \\
&1 - \frac{1}{1+e^{-x}} \\
&= \frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}} \\
&= \frac{e^{-x}}{1+e^{-x}} \\
&= (\frac{1+e^{-x}}{e^{-x}})^{-1} = (\frac{1}{e^{-x}} + 1)^{-1} = \frac{1}{1+e^{x}} \\
&\mathcal{L} = -ln(\frac{1}{1+e^{x}}) = ln(1+e^{x}) \\
&\lim_{x \to \infty} \mathcal{L} = ln(e^{x}) = x \\
&\lim_{x \to -\infty} \mathcal{L} = ln(1) = 0
\end{aligned}
\tag{18}
$$

Fig. 19 confirms these limits. When the loss decreases the derivative goes from 1 if $t = 0$ or -1 if $t = 1$ to 0.

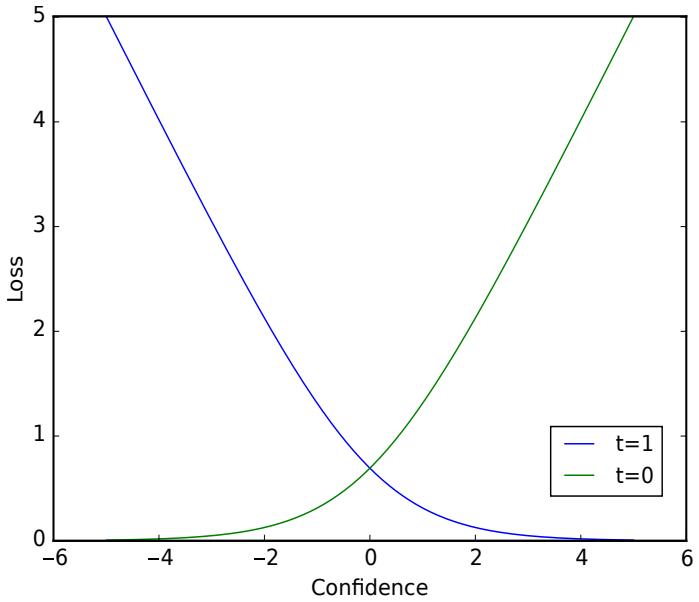

Fig. 19: $\mathcal{L}_{BCE}$ curves for $target \in 0, 1$.

APPENDIX L
PROFILES

A. Profile Plots

All the results and profiles are displayed in Fig. 20. Note that for d) a larger interval was used because it would stop before convergence due to stochastic fluctuations.

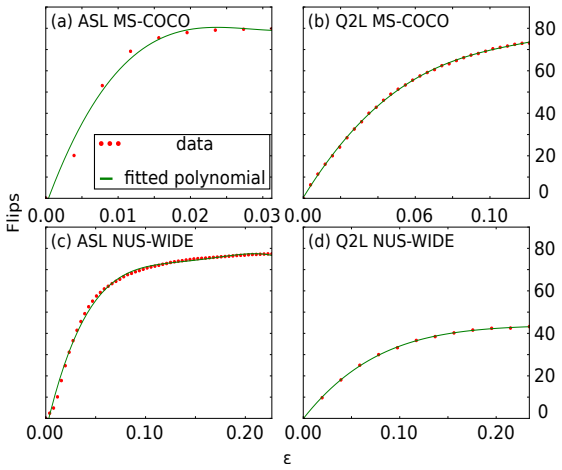

Fig. 20: The profiles of the models. The red dots represent the values that are measured during the probe attacks. The green line represents the fitted polynomial, which is queried to estimate the potential amount of flips.

B. Profile Cost

The profiling cost can be explained in terms of the amount of attack iterations. The number of iterations depends on the ϵ required for convergence of flips. The profiling can be described by the following equation:

$$Cost = \frac{\epsilon_{max}}{\alpha}((1 + \frac{1}{i})fp + bp) \tag{19}$$

In this equation fp and bp denote forward- and backward-pass respectively. α is the step size of the perturbation. We perform the amount of iterations times a forward pass followed by a backward pass. Then there is another forward pass once every interval i for measuring the flips. For the Query2Label model on MS-COCO dataset for example this amounts to $\frac{0.12}{\frac{1}{2560}} \times (1\frac{1}{50}fp + bp) = 337fp + 302bp$. It is worth mentioning that the cost of a forward/backward pass relies on the model architecture.

APPENDIX M
VISUALS

Fig. 21: Examples of adversarial examples generated with different epsilon values for Query2Label MS-COCO with SLAM, the ϵ values are the ones used in Table IV.

APPENDIX N
LABEL SUBSET GENERATION ALGORITHM

Algorithm 3 n-label subset generation algorithm

Inputs: Model inference outputs, instance correlation matrix $\boldsymbol{icm}$, number of labels num_l, γ, number of branches num_b and branch depth bd.
Output: a subset of labels of length num_l [1]

$\mathcal{P}_{rankings} =$ argsort$(outputs)$
$root\_label = \mathcal{P}_{rankings}[\mathcal{P}_{rankings}.size() - 1].item()$
$\boldsymbol{base\_label\_set}$ = $[root\_label]$
for $l = 0, 1, 2, \ldots,$ $subset\_length$ **do**
    root = Node(base_label_set)
    $\boldsymbol{parents}$ = [root]
    $\boldsymbol{children}$ = []
    depth = $min(bd, num\_l - (base\_label\_set).size())$
    for $d = 0, 1, 2, \ldots$ depth **do**
        for parent in $\boldsymbol{parents}$ **do**
            current_label_set = parent.get_labels()
            $\mathcal{Z}_{to\_set} = \boldsymbol{icm}[:, \boldsymbol{current\_label\_set}].sum(axis = 1)$
            $\mathcal{Z}_{from\_set} = \boldsymbol{icm}[\boldsymbol{current\_label\_set}, :].sum(axis = 0)$
            $\mathcal{Z} = \mathcal{Z}_{to\_set} + \mathcal{Z} from\_set$
            $\mathcal{Z}_{normalised}$ =Z / Z_{max})
            $\mathcal{S} = \gamma \times \mathcal{Z}_{normalised} + (1 - \gamma) \times \mathcal{P}_{normalised}$
            $\mathcal{S}_{ranking} = argsort(\mathcal{S}) - \boldsymbol{current\_label\_set}$
            for $b = 0, 1, 2, \ldots$ num_b **do**
                $added\_label = \mathcal{S}_{ranking}[(\mathcal{S}_{ranking}).size() - 1 - b]$
                $\boldsymbol{parent}$.add_child(added_label)
            end for
            $\boldsymbol{children}$.add($\boldsymbol{parent.children}$)
        end for
        $\boldsymbol{parents}$ = $\boldsymbol{children}$
        $\boldsymbol{children}$ = []
    end for
    max_obj_value = 0
    best_option = None
    for parent in $\boldsymbol{parents}$ **do**
        $obj\_value = \mathcal{O}(p.get\_list(), \boldsymbol{icm}, normalized\_confidences, gamma)$
        if obj_value $>$ max_obj_value **then**
            max_obj_value = obj_value
            best_option = p
        end if
    end for
    base_label_set.add(($\boldsymbol{base\_label\_set}$ - $\boldsymbol{best\_option.list}$)[0])
end for
return $\boldsymbol{base\_label\_set}$