# Integer Programming-based Error-Correcting Output Code Design for Robust Classification

**Samarth Gupta**[1] **Saurabh Amin**[2]

[1]Center for Computational Science and Engineering, Massachusetts Institute of Technology, USA
[2]Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, USA

## Abstract

Error-Correcting Output Codes (ECOCs) offer a principled approach for combining binary classifiers into multiclass classifiers. In this paper, we study the problem of designing optimal ECOCs to achieve both nominal and adversarial accuracy using Support Vector Machines (SVMs) and binary deep neural networks. We develop a scalable Integer Programming (IP) formulation to design minimal codebooks with desirable error correcting properties. Our work leverages the advances in IP solution techniques to generate codebooks with optimality guarantees. To achieve tractability, we exploit the underlying graph-theoretic structure of the constraint set. Particularly, the size of the constraint set can be significantly reduced using edge clique covers. Using this reduction technique along with Plotkin's bound in coding theory, we demonstrate that our approach is scalable to a large number of classes. The resulting codebooks achieve a high nominal accuracy relative to standard codebooks (e.g., one-vs-all, one-vs-one, and dense/sparse codes). Interestingly, our codebooks provide non-trivial robustness to white-box attacks without any adversarial training.

## 1 INTRODUCTION

Error Correcting Output Codes (ECOCs) offer an effective and flexible tool to combine individually trained binary classifiers for multiclass classification. Prior research [Dietterich and Bakiri, 1995, Allwein et al., 2000] showed that ECOCs can provide high multiclass classification accuracy using simple but powerful binary classifiers (e.g., Support Vector Machines and Adaboost). More recently extensive body of work has emerged, showing that when large amount of training data is available, deep learning models [LeCun et al., 2015] outperform most multiclass classifiers. Still, further progress is needed for classification tasks when training data is limited and model robustness and interpretability are preferred.

In this paper, we consider the problem of ECOC-based multiclass classification, when individual binary classifiers are SVMs or deep learning models. We focus on the question of "optimal" design of codebooks based on explicitly designed criteria that contribute to high prediction accuracy, both in nominal and adversarial settings.

Our approach to codebook design is distinct from the prior works that use a continuous relaxation of the inherently discrete optimization problem, and solve the relaxed problem using nonlinear optimization tools Crammer and Singer [2002], Zhao and Xing [2013], Xiao Zhang et al. [2009], Martin et al. [2018]. While this approach is scalable to a large number of classes, it does not provide optimality guarantees. This limitation prevents a systematic evaluation of how accurately the original (discrete) problem is solved. An earlier approach in Dietterich and Bakiri [1995] casts the design problem as a propositional satisfiability problem that can be solved for using off-the-shelf SAT solvers. However, using their approach only a feasible solution may be readily computable. In contrast to both these approaches, we formulate the optimal codebook design problem as a large-scale Integer Program (IP) and exploit the structure of the problem to obtain a tight formulation that can be easily solved with modern IP solvers. Importantly, the resulting codebook has optimality guarantees. This also enables a systematic comparison with respect to several well-known fixed-size codebooks on real-world datasets.

Our flexible IP formulation can account for the following codebook (or coding matrix) design criteria: (i) Sufficiently large Hamming distance between any pair of codewords (*row separation*); (ii) Uncorrelated columns (*column separation*); (iii) Relatively even distribution of data points across two classes (*balanced columns*); and (iv) Larger Hamming distance between pair of codewords whose cor-

*Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence* (UAI 2021), PMLR 161:1724–1734.

responding classes are hard to separate from one another (*data-distribution*). These criteria are important not only for the nominal error correction performance, but they also contribute to adversarial robustness. However, this initial formulation can quickly become intractable for classification problems with more than 10 classes.

To address the above-mentioned computational bottleneck, we exploit the inherent graph-theoretic structure of the constraint set [Padberg, 1973, Atamtürk et al., 2000]. In particular, we prove that the constraints modeling the pair of columns that do not satisfy column separation criterion can be replaced by a much smaller set formed by an *edge clique cover* of the underlying graph. This result enables us to transform our original problem into another IP with a much smaller set of constraints. We further scale to even larger number of classes, while maintaining optimality guarantees using the classical Plotkin's bound.

A distinct advantage of our ECOC based design approach is, that our IP-generated *compact* codebooks achieve high nominal accuracy and outperform well-known codebooks such as one-vs-all, one-vs-one and other dense or sparse designs. To evaluate the robustness of our optimal codebooks to adversarial perturbations (see Szegedy et al. [2013], Goodfellow et al. [2015], Su et al. [2017]), we conduct extensive experiments based on white-box attacks Madry et al. [2018], Tramer et al. [2020]. Remarkably, our codebooks achieve non-trivial robustness even without *any adversarial training of the individual binary classifiers*. Thus, our results suggest a strong potential of ECOCs for training robust classifiers.

Our main contributions are as follows: (i) We propose an IP formulation to the fundamental ECOC design problem and achieve tractability by exploiting the graph-theoretic structure of the constraint set. (ii) We provide optimality guarantees, even for large number of classes using Plotkin's bound. (iii) Our formulation can flexibly incorporate different design criteria while preserving the (integer) linear structure of the optimization problem. (iv) Our IP generated codebooks outperform standard codebooks on various natural classification tasks. (v) Our codebooks provide non-trivial robustness without any adversarial training, which appears to be the first such result in the literature.

## 2 ECOC FOR CLASSIFICATION

In the ECOC-based framework [Dietterich and Bakiri, 1995], for a given $k$-class classification problem, each class is *encoded* with a unique codeword of length $L$, resulting in a codebook (coding matrix) $\mathcal{M} = (m_{ij})$ of size $k \times L$. For binary (resp. ternary) codes, the entries $m_{ij}$ of the coding matrix $\mathcal{M}$ belong to the set $\{+1, -1\}$ (resp. $\{+1, 0, -1\}$). The rows (resp. columns) of $\mathcal{M}$ correspond to distinct classes (resp. binary classifiers or *hypotheses*). Figure 1 shows examples of two standard codebooks.

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $C_1$ | +1 | -1 | -1 | -1 |
| $C_2$ | -1 | +1 | -1 | -1 |
| $C_3$ | -1 | -1 | +1 | -1 |
| $C_4$ | -1 | -1 | -1 | +1 |

(a) one-vs-all (binary)

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ |
|---|---|---|---|---|---|---|
| $C_1$ | +1 | +1 | +1 | 0 | 0 | 0 |
| $C_2$ | -1 | 0 | 0 | +1 | +1 | 0 |
| $C_3$ | 0 | -1 | 0 | -1 | 0 | +1 |
| $C_4$ | 0 | 0 | -1 | 0 | -1 | -1 |

(b) one-vs-one (ternary)

Figure 1: Examples of codebooks for a 4-class problem.

In the learning problem corresponding to every column in $\mathcal{M}$, the set of training examples belonging to different classes $C_1, \ldots, C_k$ is partitioned into two groups: all examples from classes with entry $+1$ represent the positive class, and all examples from classes with entry $-1$ represent the other class. In the case of ternary codes, examples from classes with entry $0$ are not included in the training set and are considered irrelevant.

Let $f_1(\cdot), \ldots, f_L(\cdot)$ denote the learned binary hypotheses for the corresponding columns of $\mathcal{M}$. For a learned hypothesis $s \in \{1, \ldots L\}$ and a test example $\tilde{x}$, let $f_{s_{+1}}(\tilde{x})$ (resp. $f_{s_{-1}}(\tilde{x})$) denote the output/score of the class $+1$ (resp. class $-1$). Then,

$$f_s(\tilde{x}) := \begin{cases} +1 & \text{if } f_{s_{+1}}(\tilde{x}) > f_{s_{-1}}(\tilde{x}) \\ -1 & \text{otherwise} \end{cases} \qquad \forall s \in \{1, \ldots, L\}.$$

After evaluating $\tilde{x}$ on all the $L$ hypotheses, we obtain an encoding $\vec{f}(\tilde{x}) = [f_1(\tilde{x}), \ldots, f_L(\tilde{x})]$. To associate $\vec{f}(\tilde{x})$ with a class (i.e., a row of coding matrix $\mathcal{M}$), we can use a decoding scheme based on a similarity measure such as Hamming distance. Particularly, one can compute the Hamming distance $d_H(\cdot, \cdot)$ between $\vec{f}(\tilde{x})$ and each codeword $\mathcal{M}(r, \cdot)$ and select the class, denoted $\hat{y}$, that corresponds to the minimum distance:

$$d_H(\mathcal{M}(r, \cdot), \vec{f}(\tilde{x})) := \sum_{s=1}^{L} \left( \frac{1 - \mathcal{M}(r, s) \times f_s(\tilde{x})}{2} \right)$$
$$\hat{y} = \underset{r}{\operatorname{argmin}} \; d_H(\mathcal{M}(r, \cdot), \vec{f}(\tilde{x})). \tag{1}$$

## 3 CODEBOOK DESIGN CRITERIA

The final prediction accuracy of ECOC scheme (1) introduced in section 2, crucially depends on the error correction capability of the coding matrix $\mathcal{M}$. To ensure low test error, the coding matrix must be chosen carefully. We now introduce the key criteria that guide our design of binary codes.[1]

**Row Separation:** It is well-known that more separation between pairs of codewords (i.e., rows in the coding matrix

[1]Similar codebook design approach can be developed for ternary codes (not presented here due to space constraints).

$\mathcal{M}$) improves the error correction capability. Particularly, if every pair of distinct codewords has a Hamming distance of at-least $d$, then such a code can correct *at-least* $\lfloor \frac{d-1}{2} \rfloor$ *errors* [Guruswami and Sahai, 1999]. Thus, we seek a coding matrix with high row separation between any pair of codewords. For a code of size $k \times L$, the minimum Hamming distance $d$ between any pair of rows (or codewords) can be analytically upper bounded using the bound provided in Plotkin [1960]:

$$d \leq \Big\lfloor \frac{kL}{2(k-1)} \Big\rfloor. \tag{2}$$

**Column Separation:** Additionally, every pair of distinct columns in $\mathcal{M}$ should be uncorrelated. The benefit of large column separation can be understood by drawing analogy to error correction in communication over a noisy channel. Encoding a signal and transmitting the codeword over a noisy channel is highly effective when the errors introduced during transmission are *random*. By maintaining a sufficiently large encoding, one can recover the original signal at the receiving end with high accuracy. Analogously, in our setup, if any two columns (classifiers) make errors in their predictions on the same inputs (i.e., their outputs are highly correlated), then the effectiveness of encoding in correcting errors will be reduced.

**Balanced Columns:** On the other hand, to prevent overfitting of individual (binary) hypotheses, it is important to prioritize the selection of columns for which the entire $k$-class training data are evenly distributed across the two classes.

**Data Distribution:** Finally in multi-class problems, some class pairs are more difficult to separate than others. Therefore, it is desirable to have larger Hamming distances among pairs of codewords corresponding to hard-to-separate class pairs. Since the underlying data-distribution is unknown, this hardness of separation can be estimated from training data either using the semantics of classes, as done in Zhao and Xing [2013]; or by calculating similarity measures between classes (for small datasets) as shown in Pujol et al. [2006], Griffin and Perona [2008], Gao and Koller [2011], Xiao Zhang et al. [2009], Martin et al. [2018].

We develop a flexible approach to capture the above-mentioned criteria into a discrete optimization formulation.

## 4 INTEGER PROGRAMMING FORMULATION

To begin with, note that for a $k-$class problem a coding matrix can have at most $(2^k - 2)/2 = 2^{k-1} - 1$ columns. However, such an exhaustive coding might be feasible only for a small $k$ (2 to 5). As $k$ increases, the number of binary classifiers that need to be trained for exhaustive coding increase exponentially. Practically, it is desirable to select a small subset (say of $L$ columns) from $2^{(k-1)} - 1$ possible columns. This subset should be selected in accordance with the codebook design criteria described in section 3.

A classical way to formulate the column subset selection problem is to cast it as a propositional satisfiability problem, and solve it using an off-the-shelf SAT solver. For example, Dietterich and Bakiri [1995] considered the following problem for $8 \leq k \leq 11$: For a predefined number of columns $L$ and some value $\rho$, is there a solution such that the Hamming distance between any two columns is between $\rho$ and $L - \rho$? However, this approach only leads to a feasible (not necessarily optimal) solution and does not capture other design criteria. In contrast, we present an Integer Programming (IP) formulation that captures the design criteria in a flexible manner and develop an approach to find an *optimal* codebook.

For sake of simplicity, we first consider the row and column separation criteria; the remaining criteria on balanced columns and data distribution can be addressed in our IP formulation, as discussed subsequently at end of this section. In its basic form, our problem is the following: We want to find a solution which *maximizes the minimum Hamming distance between any two rows (or the error-correcting property) while maintaining high column separation.*

Let $x_i$ denote the binary variable associated with each column $i$ of the exhaustive code for $i \in \{1, \dots 2^{k-1} - 1\}$, i.e. the decision variable whether or not column $i$ is selected in the final solution. Also, let $x_{ij}$ be the binary variable which represents the outcome of AND operation between variables $x_i$ and $x_j$ for all distinct $i, j$ pairs, i.e. $(i, j) \in \{1, \dots, 2^{k-1} - 1\}^2 | i < j$. We can now write our basic IP formulation to generate an optimal codebook as follows:

$$\mathcal{IP}1 : \max_{x_i, x_{ij}} \ \min \ \{d_H^{1,2}(x_i), \dots, \ d_H^{k-1,k}(x_i)\} \tag{3}$$

$$\text{s.t.} \sum_{i=1}^{2^{k-1}-1} x_i \leq L$$

$$\rho \, x_{ij} \leq d_H\big(\mathcal{M}(\cdot, i), \mathcal{M}(\cdot, j)\big) \, x_{ij} \leq (L - \rho) \, x_{ij}$$
$$\forall \ (i, j) \in \{1, \dots, 2^{k-1} - 1\}^2 | \ i < j \tag{4}$$

$$x_{ij} \leq x_i \tag{5}$$

$$x_{ij} \leq x_j \tag{6}$$

$$x_i + x_j - 1 \ \leq x_{ij} \tag{7}$$

$$d_H^{s,t}(x_i) = \sum_{i=1}^{2^{k-1}-1} \Big(\frac{1 - \mathcal{M}(s, i) \times \mathcal{M}(t, i)}{2}\Big) x_i$$
$$\forall \ (s, t) \in \{1, \dots, k\}^2 | \ s < t$$

$$x_i \in \{0, 1\} \quad \forall \ i \in \{1, \dots, 2^{k-1} - 1\}$$

$$x_{ij} \in \{0, 1\} \ \forall \ (i, j) \in \{1, \dots, 2^{k-1} - 1\}^2 | \ i < j$$

In $\mathcal{IP}1$, max-min objective (3) can be simplified by introducing an auxiliary variable $t$, where $t = \min\ \{d_H^{1,2}(x_i),\ d_H^{1,3}(x_i),\ \ldots,\ d_H^{k-1,k}(x_i)\ \}$, and adding the corresponding constraints $t \leq d_H^{1,2}(x_i)$ , $t \leq d_H^{1,3}(x_i)$ , $\ldots$ , $t \leq d_H^{k-1,k}(x_i)$. Eq. (4) ensures large column separation when $x_{ij} = 1$. Constraints (5) and (6) ensure that if $x_{ij} = 1$ then both columns $i$ and $j$ are included in the solution, i.e. $x_i = 1$ and $x_j = 1$. Conversely, Equation (7) ensures that if columns $i$ and $j$ are selected then $x_{ij} = 1$.

We note that in $\mathcal{IP}1$ there are $2^{k-1} - 1 \approx \mathcal{O}(2^{k-1})$ binary variables for each column, and corresponding to every distinct pair of columns $(i, j)$, there are $\binom{2^{k-1}-1}{2} \approx \mathcal{O}(2^{2k-3})$ binary variables. Thus, the total number of binary variables are $\mathcal{O}(2^{2k-3})$. Similarly, the total number of constraints are $\mathcal{O}(2^{2k-1})$. For $k = 10$, this entails solving an IP of approximately $130,000$ variables and $650,000$ constraints. Modern IP solvers like Gurobi and CPLEX can indeed handle such problem instances.

However, for $k > 10$, the above optimization problem quickly becomes intractable. The main reason is that we have a binary variable $x_{ij}$ for each pair of columns to capture the large column separation criterion (4). We now present a second formulation which does not involve a new variable for every pair of columns.

Let $\mathcal{S}_p$ denote the set of all distinct pairs of columns in the exhaustive code $\mathcal{M}$, i.e. $\mathcal{S}_p = \{(i, j) \in \{1, \ldots, 2^{k-1} - 1\}^2 |\ i < j\}$ and $|\mathcal{S}_p| = \binom{2^{k-1}-1}{2}$. We now consider two mutually disjoint subsets $\mathcal{S}_p^{feas}$ and $\mathcal{S}_p^{inf}$, such that $\mathcal{S}_p = \mathcal{S}_p^{feas} \cup \mathcal{S}_p^{inf}$: the set $\mathcal{S}_p^{feas}$ (resp. $\mathcal{S}_p^{inf}$) contains only those $i, j$ pairs that *satisfy* (resp. *do not* satisfy) the column separation criterion (4). Mathematically, we can write:

$$
\begin{aligned}
\mathcal{S}_p &= \Big\{(i,j) \in \{1,\ldots,2^{k-1}-1\}^2 |\ i<j\Big\},\\
\mathcal{S}_p^{feas} &= \Big\{(i,j) \in \{1,\ldots,2^{k-1}-1\}^2 |\ i<j \text{ and }\\
&\qquad \rho \leq d_H\big(\mathcal{M}(\cdot,i), \mathcal{M}(\cdot,j)\big) \leq (L-\rho)\Big\}, \quad (8)\\
\mathcal{S}_p^{inf} &= \mathcal{S}_p \setminus \mathcal{S}_p^{feas}.
\end{aligned}
$$

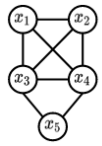
In this new representation, the constraint (4) is captured by the construction of $\mathcal{S}_p^{feas}$ (8), which eliminates the need of variables $x_{ij}$ for column pairs. Similarly, we no longer need the constraints (5), (6) and (7). Note that, for any $(i, j)$ pair of columns in the set $\mathcal{S}_p^{inf}$, *at-most* one of the two columns can be included in the final solution. This can be achieved by setting $x_{ij} = 0$ in (7). Equivalently, for every pair $(i, j) \in \mathcal{S}_p^{inf}$, it is sufficient to impose the constraint $x_i + x_j - 1 \leq 0$.

We can now write $\mathcal{IP}1$ as the following equivalent form:

$$
\begin{aligned}
\mathcal{IP}2: \max_{x_i}\ & \min\ \{d_H^{1,2}(x_i), \ldots, d_H^{k-1,k}(x_i)\ \}\\
\text{s.t.}\ & \sum_{i=1}^{2^{k-1}-1} x_i \leq L\\
& x_i + x_j \leq 1 \qquad \forall\ (i,j) \in \mathcal{S}_p^{inf} \qquad (9)\\
& d_H^{s,t}(x_i) = \sum_{i=1}^{2^{k-1}-1} \Big(\frac{1 - \mathcal{M}(s,i) \times \mathcal{M}(t,i)}{2}\Big) x_i\\
& \qquad\qquad \forall\ (s,t) \in \{1,\ldots,k\}^2 |\ s<t\\
& x_i \in \{0,1\} \qquad \forall\ i \in \{1,\ldots,2^{k-1}-1\}
\end{aligned}
$$

Since $\mathcal{IP}2$ does not contain any $x_{ij}$ variables, this formulation has significantly less number of variables and constraints in comparison to $\mathcal{IP}1$. The computational complexity of this formulation ($\mathcal{IP}2$) is mainly governed by the size of the set $\mathcal{S}_p^{inf}$, which determines the number of constraints in (9). In $\mathcal{IP}2$, there are $\mathcal{O}(2^{k-1})$ variables and the number of constraints are $\mathcal{O}(|\mathcal{S}_p^{inf}|)$. However, even in this new representation, the size of the set $\mathcal{S}_p^{inf}$ becomes prohibitively large as $k$ increases further. Table 1, column 4 shows that $|\mathcal{S}_p^{inf}|$ quickly increases with $k$ for an appropriately chosen $\rho$.

Fortunately, the constraints (9) for the set $\mathcal{S}_p^{inf}$ can be represented on a graph $\mathcal{G}_p^{inf}$, in which each node corresponds to a column $x_i$ and each constraint $x_i + x_j \leq 1$ corresponds to an edge between the node $i$ and node $j$; see Figure 2 for an illustration.

$$
\mathcal{S}_p^{inf} = \left\{
\begin{array}{l}
x_1 + x_2 \leq 1\ ,\ x_1 + x_3 \leq 1\\
x_1 + x_4 \leq 1\ ,\ x_2 + x_3 \leq 1\\
x_2 + x_4 \leq 1\ ,\ x_3 + x_4 \leq 1\\
x_3 + x_5 \leq 1\ ,\ x_4 + x_5 \leq 1
\end{array}
\right\} \Rightarrow
$$


$\mathcal{G}_p^{inf}$

Figure 2: An example of $\mathcal{S}_p^{inf}$ and corresponding $\mathcal{G}_p^{inf}$.

This graph-theoretic representation can be exploited to reduce the number of constraints involving $(i, j)$ column pairs in the set $\mathcal{S}_p^{inf}$. Before presenting this result, we recall that a *clique* (denoted as $\mathcal{C}$) is a subset of nodes in a graph such that there is an edge between any two distinct nodes of this subset. Proofs of upcoming results are provided in the supplementary material (SM).

**Lemma 1.** *The feasible space enclosed by the constraints constituting the edges of any clique $\mathcal{C}$ in $\mathcal{G}_p^{inf}$ is same as that enclosed by the constraint:*

$$\sum_{i \in \mathcal{C}} x_i \leq 1. \qquad (10)$$

From lemma 1, we obtain that for a clique of size $n$, $n(n-1)/2$ constraints of form $x_i + x_j \leq 1$ between all $(i, j)$ node

pairs in the clique can be substituted with a *single* constraint (10). This constraint captures the requirement that out of all the columns in $\mathcal{M}$ forming a clique, at most one can be present in a feasible solution. Before introducing our next result, we recall the following useful definition [Conte et al., 2016].

**Definition 1** (Edge Clique Cover)**.** *An edge clique cover for a graph $\mathcal{G}$, denoted as $\mathcal{ECC}(\mathcal{G})$, is a set of cliques $\mathcal{ECC}(\mathcal{G}) = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ such that:*

1. *No clique $\mathcal{C}_i$ is contained in another clique $\mathcal{C}_j$, i.e $\mathcal{C}_i \nsubseteq \mathcal{C}_j$ for all $i \neq j$, and*
2. *Every edge in the graph $\mathcal{G}$ is included in atleast one clique.*

**Lemma 2.** *The feasible space enclosed by the constraint set $\mathcal{S}_p^{inf}$ (or its graphical equivalent $\mathcal{G}_p^{inf}$) in $\mathcal{IP}2$ is same as that enclosed by a much smaller constraint set formed by the edge-clique-cover of $\mathcal{G}_p^{inf}$, i.e. $\mathcal{ECC}(\mathcal{G}_p^{inf})$.*

Note that a graph can have many possible edge clique covers; see for example Figure 3. To reduce the size of the constraint set $\mathcal{S}_p^{inf}$ as much as possible, one would need to find an edge clique cover of the smallest size. However, the minimum edge cover problem is known to be NP-hard [Garey and Johnson, 1990]. Several heuristics have been proposed to find edge clique cover of a graph such as Kellerman [1973], Kou et al. [1978], Gramm et al. [2009], Conte et al. [2016]. For our purpose, the heuristic proposed in Conte et al. [2016] is particularly well-suited since it shows a *linear runtime* in the number of edges. We therefore use this heuristic in our analysis.

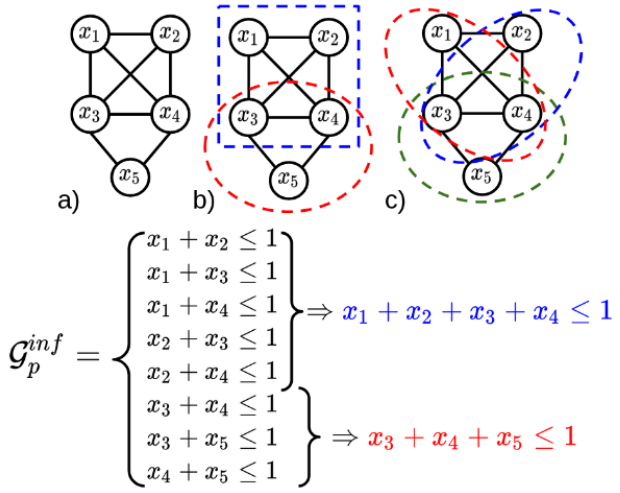



Figure 3: Graphical depiction of an example $\mathcal{S}_p^{inf}$ in (a), with two feasible edge clique covers ((b) and (c)). For edge-cover in (b), we show the reduced set of constraints corresponding to its cliques in blue and red.

Furthermore, we can extend lemma 2 to generate edge-clique-covers of very large graphs in a distributed manner as follows:

**Lemma 3.** *Suppose $\mathcal{G}_1, \ldots \mathcal{G}_m$ are edge-disjoint subgraphs of $\mathcal{G}_p^{inf}$, such that:*

1. $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset \;\; \forall\, i, j \in \{1, \ldots, m\}^2 | i < j$
2. $\bigcup_{i=1}^{m} \mathcal{G}_i = \mathcal{G}_p^{inf}$

*The union of the edge clique covers of individual subgraphs $\mathcal{G}_1, \ldots \mathcal{G}_m$ is a valid edge clique cover of $\mathcal{G}_p^{inf}$ : $\bigcup_{i=1}^{m} \mathcal{ECC}(\mathcal{G}_i) = \mathcal{ECC}(\mathcal{G}_p^{inf})$.*

Finally, using lemma 2 (or its extension lemma 3) we can reduce $\mathcal{IP}2$ to the following integer program:

$$\mathcal{IP}3 : \max_{x_i} \; \min \{ d_H^{1,2}(x_i), \ldots, d_H^{k-1,k}(x_i) \} \tag{11}$$

s.t.

$$\sum_{i=1}^{2^{k-1}-1} x_i \leq L$$

$$\sum_{i:\, i \in \mathcal{C}_t} x_i \leq 1 \qquad \forall\, \mathcal{C}_t \in \mathcal{ECC}(\mathcal{G}_p^{inf}) \tag{12}$$

$$d_H^{s,t}(x_i) = \sum_{i=1}^{2^{k-1}-1} \Big( \frac{1 - \mathcal{M}(s,i) \times \mathcal{M}(t,i)}{2} \Big) x_i \qquad \forall\, (s,t) \in \{1, \ldots, k\}^2 |\; s < t$$

$$x_i \in \{0, 1\} \qquad \forall\, i \in \{1, \ldots, 2^{k-1} - 1\}$$

The size of $\mathcal{IP}3$ is mainly governed by the size of the set $\mathcal{ECC}(\mathcal{G}_p^{inf})$, which determines the number of constraints in (12). This is a major improvement over $\mathcal{IP}2$, as the size of $\mathcal{ECC}(\mathcal{G}_p^{inf})$ is much smaller than $\mathcal{S}_p^{inf}$; to see this one can compare columns 4 and 5 in Table 1.

We now address the remaining criteria of balanced columns and data-distribution in $\mathcal{IP}3$.

The requirement of balanced columns can be simply incorporated by pre-fixing all the $x_i$ violating this criterion to 0 in $\mathcal{IP}3$. Equivalently, since each $x_i \in \{0, 1\}$ corresponds to whether or not a column is selected from the exhaustive code $\mathcal{M}$, we can simply reduce $\mathcal{M}$ by removing the unbalanced columns and then form $\mathcal{IP}3$. In contrast to Xiao Zhang et al. [2009], in our formulation, the requirement for balanced columns further reduces the final problem size. For more details, see appendix C.1 in SM.

The requirement of data-distribution can be easily incorporated by modifying the objective function. Previous works such as Martin et al. [2018], Zhao and Xing [2013], Xiao Zhang et al. [2009], pre-compute a similarity measure between every pair of classes (from training data) and use this measure to estimate the desirable class-pairwise hamming

Table 1: Reducing the size of the constraint set $|\mathcal{S}_p^{inf}|$ in $\mathcal{IP}2$ by finding the Edge Clique Cover of $\mathcal{G}_p^{inf}$.

| No. of classes $k$ | No. of Columns $2^{k-1}-1$ | $\rho$ | No. of constraints $\lvert\mathcal{S}_p^{inf}\rvert$ | No. of constraints $\lvert\mathcal{ECC}(\mathcal{G}_p^{inf})\rvert$ **(Reduced)** | Reduction Factor | Time Taken (in sec.) |
|---|---|---|---|---|---|---|
| 10 | 511 | 3 | 11,475 | 695 | **16** | 0.146 |
| 11 | 1,023 | 3 | 28,105 | 1,404 | **20** | 0.208 |
| 12 | 2,047 | 4 | 236,313 | 8,165 | **28** | 0.991 |
| 13 | 4,095 | 4 | 610,006 | 18,472 | **33** | 2.573 |
| 14 | 8,191 | 4 | 1,543,815 | 41,088 | **37** | 7.390 |
| 15 | 16,383 | 5 | 12,040,770 | 44,916 | **268** | 58.957 |
| 16 | 32,767 | 5 | 31,783,020 | 91,304 | **348** | 249.53 |
| 17 | 65,535 | 5 | 82,441,772 | 185,661 | **444** | 935.76 |
| 18 | 131,071 | 6 | 616,094,535 | 1,073,248 | **574** | 10075.8 |
| 18 | 131,071 | 6 | 616,094,535 | 5,952,906 + 622,604 = 6,575,510 | 93 | 16251.667 + 4977.376 = 21229.04 |

distances (denoted as $\hat{d}^{p,q}$). Using $\hat{d}^{p,q}$, they optimize to obtain codebooks which attain these distance values. This can be easily incorporated in our formulation by changing the objective function (11) in $\mathcal{IP}3$ to the following:

$$\min_{x_i} \sum_{(p,q)\in\{1,\dots,k\}^2 | p<q} |d_H^{p,q}(x_i) - \hat{d}^{p,q}|. \tag{13}$$

For more details regarding the data-distribution criteria, refer to appendix C.2 in SM. Before concluding this section, we establish a connection between the optimal objective function value of $\mathcal{IP}3$ (denoted as $f^*$) and Plotkin's bound as follows:

**Proposition 1.** *Plotkin's Bound is an upper bound to* $\mathcal{IP}3$*:* $f^* \leq \lfloor \frac{kL}{2(K-1)} \rfloor$.

Recall from (2), the minimum Hamming distance between any two rows is analytically upper bounded by Plotkin's bound. Mathematically, we can write:

$$\min\{d_H^{1,2}(x_i),\dots,d_H^{k-1,k}(x_i)\} \leq \left\lfloor \frac{kL}{2(K-1)} \right\rfloor, \text{implying}$$
$$\underbrace{\max_{x_i} \min\{d_H^{1,2}(x_i),\dots,d_H^{k-1,k}(x_i)\}}_{(\text{ obj. function of } \mathcal{IP}3)} \leq \left\lfloor \frac{kL}{2(K-1)} \right\rfloor. \tag{14}$$

This connection enables us to determine the solution quality (optimality gap) for large $k$, when *exactly* solving $\mathcal{IP}3$ becomes challenging, as discussed in the next section.

## 5 EXPERIMENTS

We run all our experiments on a machine with a single 1080Ti Nvidia GPU, Intel Core i7-6800K CPU and 128 GB RAM. We use Gurobi-v9.1 as our IP solver. Our code is available at https://github.com/SamarthGM/IP_ECOC.

Our first set of computational experiments focus on solving $\mathcal{IP}3$ which uses the edge-clique-cover approach to reduce the constraint set $\mathcal{S}_p^{inf}$. Table 1 shows the reduction in size of set $\mathcal{S}_p^{inf}$ as the number of classes $k$ increases. Notably, for $k \geq 15$ we achieve a reduction of more than *two orders of magnitude*, which demonstrates the advantage of using our approach. The last row in Table 1 shows the performance of generating the edge-cover on two different subgraphs obtained after partitioning the original graph, thus validating lemma 3.

Thanks to the reduced constraint set, we can now solve $\mathcal{IP}3$ and obtain the optimality gap for different instances as shown in Table 2. Here $f_{best}$ denotes the objective function value of the best solution and "Best Bound" denotes the tightest upper bound, both obtained by solving $\mathcal{IP}3$ using Gurobi. In most cases, we obtain an optimal solution or a relatively small optimality gap. Thus, our formulation is tight and enables Gurobi to terminate quickly without exploring a large branch-and-bound tree. These results demonstrate that our approach to codebook design indeed provides *low optimality gaps.*

Table 2: $\mathcal{IP}3$ Performance for $k \leq 18$ (max. time 1800s).

| $k$ | $L$ | $f_{\text{best}}$ | Best Bound (BB) | Plotkin's Bound (PB) | Gap $\lvert f_{\text{best}} - \text{BB}\rvert / f_{\text{best}}$ |
|---|---|---|---|---|---|
| 10 | 20 | 10 | 10 | 11 | 0% |
| 11 | 22 | 12 | 12 | 12 | 0% |
| 12 | 24 | 12 | 12 | 13 | 0% |
| 13 | 26 | 13 | 14 | 14 | 7.69% |
| 14 | 28 | 14 | 15 | 15 | 7.14% |
| 15 | 30 | 15 | 16 | 16 | 6.67% |
| 16 | 32 | 16 | 17 | 17 | 6.25% |
| 17 | 34 | 16 | 18 | 18 | 12.2% |
| 18 | 36 | 17 | 19 | 19 | 11.8% |

Further in Table 2, we observe that for almost all cases, the analytical Plotkin's Bound (recall (14)) is same as the "Best Bound" generated by the solver. Therefore, we can use Plotkin's Bound in place of the solver-generated bound to estimate the gap or solution quality.

Our approach is easily scalable to $k > 18$, with a minor modification. Instead of using all the $2^{k-1}-1$ columns from the exhaustive code $\mathcal{M}$, we can use only a subset of randomly sampled columns (around 2000) from $\mathcal{M}$ and then formulate and solve $\mathcal{IP}3$ on this much smaller subset. Our experiments, in Table 3, show that this approach provides a high quality feasible solution for $18 < k \leq 50$. For most

commonly used datasets, particularly in ECOC literature [Martin et al., 2018], $k \sim 50$ should suffice. An upper bound can be obtained using the analytical Plotkin's bound, thus enabling us to compute the gap. We again achieve reasonably *low gap*, thus further validating the tightness and scalability of our overall approach.

Table 3: $\mathcal{IP}3$ Performance for $k > 18$ (max. time 1800s).

| $k$ | $L$ | $f_{\text{best}}$ | Plotkin's Bound (PB) | Gap $\lvert f_{\text{best}} - \text{PB}\rvert / f_{\text{best}}$ |
|---|---|---|---|---|
| 20 | 40 | 19 | 21 | 10.5% |
| 25 | 50 | 23 | 26 | 13% |
| 30 | 60 | 27 | 31 | 14.8% |
| 35 | 70 | 31 | 36 | 16.1% |
| 40 | 75 | 36 | 41 | 13.8% |
| 45 | 80 | 40 | 46 | 15% |
| 50 | 100 | 44 | 51 | 15.9% |

We now evaluate the classification performance of our IP-generated codebooks in both natural and adversarial settings. We compare performance against various standard codebooks: 1-vs-all [Rifkin and Klautau, 2004] and 1-vs-1 as well as Sparse and Dense codes generated using the procedure outlined in Allwein et al. [2000].[2]

## NATURAL CLASSIFICATION PERFORMANCE

**Toy Dataset (2d):** We generate a synthetic dataset of 10 classes where points in each class are sampled from a 2d Gaussian distribution. Here we use SVMs with radial basis function (RBF) kernel as our binary classifier for individual hypotheses in all our codebooks. Figure 4 shows the decision boundaries of all hypotheses for three codebooks along with the training set. The prediction accuracy (average of 50 randomly generated instances) on the test set is reported in Table 4. Our $\mathcal{IP}3$ generated codebook easily outperforms other codebooks, and almost matches the accuracy of 1-vs-1. Note that this codebook only used $L = 20$ columns while 1-vs-1 used $L = 45$ columns. This highlights the benefit of ECOC theory: high accuracy can be achieved with a carefully chosen *compact* code-book.

Table 4: Performance on 2d Toy dataset ($k = 10$).

| $\mathcal{IP}3$ | | Dense | Sparse | 1-vs-All | 1-vs-1 |
|---|---|---|---|---|---|
| $L = 10$ | $L = 20$ | $L = 10$ | $L = 10$ | $L = 10$ | $L = 45$ |
| 91.76% | **92.15%** | 90.85% | 71.3% | 84.04% | **92.5%** |

**Real-world Datasets (Small/Medium):** We evaluate the performance of different codebooks on small to medium sized, real-world datasets. We consider *Glass*, *Ecoli* and *Yeast* datasets taken from UCI repository [Dua and Graff, 2017]. Details such as the number of samples, features and classes for each dataset are provided in SM. For Dense, Sparse, and IP generated codebook we set $L = 2k$. We use SVMs with Rbf kernel as the binary classifier for training different hypotheses in our IP-generated and other codebooks. We randomly set aside 30% of the samples as our test set and use them to evaluate the performance of different codebooks. The final average test set accuracies of 50 random splits of training and test data are reported in Table 5. Our codebook provides best accuracy on all the three datasets with significant improvement on *Glass* and *Ecoli* datasets.

Table 5: Performance of various codebooks on different real-world (small) datasets.

| | $\mathcal{IP}3$ | Dense | Sparse | 1-vs-all | 1-vs-1 |
|---|---|---|---|---|---|
| Glass | **65.13%** | 62.86% | 57.78% | 55.29% | 56.52% |
| Ecoli | **79.4%** | 76.91% | 76.18% | 69.57% | 74.04% |
| Yeast | **53.5%** | 52.88% | 45.27% | 46.62% | 51.34% |

We now evaluate the performance of different codebooks on real-world image datasets: MNIST and CIFAR10.

**MNIST:** We run two set of experiments: In the first set, we use SVMs (with both Linear and Rbf kernel) on PCA-transformed MNIST dataset (using 25 principal components). In the second set, we use binary Convolutional Neural Networks (CNNs) to train different hypotheses in our codebooks. Tables 6 and 7 provide the test set accuracy (averaged over 5 runs) of different codebooks from both sets of experiments.

Table 6: Performance of different codebooks using SVM on PCA transformed MNIST dataset.

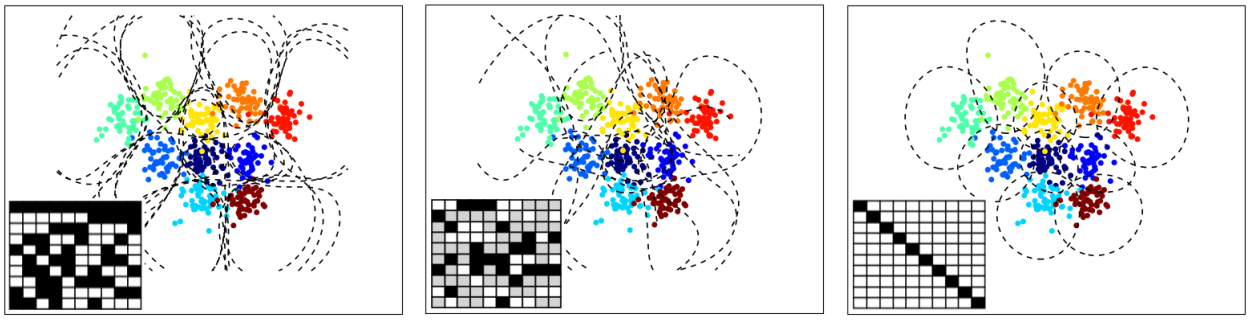
| | $\mathcal{IP}3$ | Dense | Sparse | 1-vs-all | 1-vs-1 |
|---|---|---|---|---|---|
| Linear | 80.37% | 75.74% | 68.87% | 76.82% | 92.01% |
| Rbf | 97.59% | 97.5% | 79.18% | 96.95% | 98.01% |

Table 7: Performance of Different Codebooks with binary CNN on MNIST dataset.

| $\mathcal{IP}3$ | Dense | Sparse | | 1-vs-all | 1-vs-1 |
|---|---|---|---|---|---|
| | | Normalized | Raw | | |
| **98.84%** | 98.8% | 95.05% | 84.17% | 98.65 | 94.51% |

We observe that in the case of *Linear kernel* our $\mathcal{IP}3$ codebook outperforms all other codebooks except for 1-vs-1, which achieves relatively higher accuracy of around 92%. This is due to fact that the individual hypotheses of different codebooks (except 1-vs-1 ) are solving much harder problems with *highly non-linear decision boundaries*. On the contrary, 1-vs-1 solves only natural classification problems, where a linear separator can be expected to do well. In using *non-linear Rbf-kernel*, both $\mathcal{IP}3$ codebook and 1-vs-1 codebook achieve similar accuracy. On the other hand, when

[2]Please see appendix E in SM for more details.



(a) $\mathcal{IP}3$ generated (b) Sparse (c) 1-vs-All

Figure 4: Decision boundaries of different hypotheses in three different codebooks on 2d dataset [Martin et al., 2018].

using CNNs our $\mathcal{IP}3$ codebook provides best performance, indicating the benefit of using powerful binary classifiers in ECOC approach.

**CIFAR10:** Since running SVMs on this dataset is computationally expensive, we resort to CNNs here. In particular, we use ResNet18 [He et al., 2015] as our binary classifier to train the individual hypotheses in different codebooks. As shown in Table 8 (averaged over 5 runs), $\mathcal{IP}3$ achieves the best performance. Note that our experiments on CIFAR10 should be viewed only in terms of evaluating the *relative* performance of different codebooks. We are aware that modern multi-output CNNs have achieved an accuracy of around 95% (or higher) on CIFAR10 dataset. However, recall that in this work our goal is to highlight the benefit of using ECOCs when working with binary classifiers.

Table 8: Performance of Different Codebooks with binary CNN (ResNet18) on CIFAR10 dataset.

| $\mathcal{IP}3$ | Dense | Sparse | | 1-vs-all | 1-vs-1 |
|---|---|---|---|---|---|
| | | Normalized | Raw | | |
| **76.25%** | 75.47% | 68.15% | 61.53% | 71.25% | 68.76% |

## ADVERSARIAL ROBUSTNESS

We now evaluate the robustness of different codebooks against white-box attacks.[3] For further comparison, we also evaluate the robustness of a naturally trained multiclass CNN with our IP-generated codebook in the final layer – this is somewhat similar to the recent approach by Verma and Swami [2019].[4] In contrast to their approach, all our binary hypotheses are naturally trained, i.e. *without any adversarial training*. We first discuss how to obtain the class probability estimates that are necessary to evaluate the adversarial robustness.

Recall from section 2 the procedure of assigning a class to an input $\tilde{x}$ using Hamming decoding. However, this decoding scheme in itself does not provide us with class probability estimates, which are essential for evaluating the robustness of an ECOC-based classifier with respect to white-box attacks [Madry et al., 2018, Goodfellow et al., 2015]. Particularly, we need probability estimates to compute the adversarial loss function. Also, we need to be able to compute the gradients of the loss-function with respect to input $\tilde{x}$.

We adopt the procedure of calculating the class probability estimates for general codebooks, as proposed in Zadrozny [2002], Hastie and Tibshirani [1998]. After evaluating an input $\tilde{x}$ on each binary classifier, we obtain a probability estimate (or score[5]), denoted $r_l(\tilde{x})$, for each column $l$ (i.e., binary classifier) in $\mathcal{M}$. Let $I$ denote the set of classes for which $\mathcal{M}(\cdot, l) = 1$ and $J$ denote the set of classes for which $\mathcal{M}(\cdot, l) = -1$. Then the class probability estimate for $i \in \{1, \dots k\}$ on an input $\tilde{x}$ is given as follows:

$$\hat{p}_i(\tilde{x}) = \sum_{l:\ \mathcal{M}(i,l)=1} r_l(\tilde{x}) + \sum_{l:\ \mathcal{M}(i,l)=-1} (1 - r_l(\tilde{x}))\,, \quad (15)$$

where differentiability with respect to $\tilde{x}$ is maintained. Note that in (15), $r_l(\tilde{x})$ is the class score or logit of the individual binary classifier and therefore represents raw estimates.

Using these estimates, we can compute a loss function (e.g, cross-entropy Loss) and then generate white-box PGD-attacks [Madry et al., 2018] to evaluate the robustness of the overall classifier. Note that we use the same *differentiable* class scores (or decoding scheme) for both prediction and to generate a white-box attack in order to prevent gradient-obfuscation [Athalye et al., 2018, Carlini et al., 2019, Tramer et al., 2020]. For all our experiments, we work with perturbations based on $l_\infty$-norm. In particular, for a given input $x'$, the allowed set of perturbations are given by set:

$$\mathcal{Q}(x') = \{x \in R^d \mid ||x - x'||_\infty \leq \epsilon \ ;\ x'_l \leq x \leq x'_u\},$$

[3] Different attacks including white-box attacks are discussed in appendix H in supplementary material (SM).

[4] Please refer to appendix I in SM for more details.

[5] Class scores can be easily converted into probabilities using sigmoid non-linearity.

where $x'_l$ and $x'_u$ are the domain dependent bounds. For example, for image data $x'_l = 0$ and $x'_u = 255$.

**MNIST:** We run an $l_\infty$-norm based 100-step PGD attack (with 5 different seeds) over multiple values of $\epsilon$ on different codebooks; Table 9 summarizes these results. In terms of the overall performance, our $\mathcal{IP}3$-generated codebook significantly outperforms all other codebooks except the *Dense* codebook. In this codebook, different pairs of codewords have different Hamming distances, ranging from 8-14. On the other hand, in $\mathcal{IP}3$, all codeword pairs have identical Hamming distance of 10 as a result of the max-min objective function (11). This disparity in performance can be addressed by incorporating the underlying data distribution (via class pair similarity measures) using the objective function (13). However, note that efficiently computing similarity measures for large image datasets is in itself a research problem. In the next set of experiments, we discuss that the performance of Dense codebook deteriorates as the data-distribution changes.

Table 9: Adversarial Accuracy of Nominally Trained Codebooks on MNIST.

| | $\epsilon = 0.05$ | $\epsilon = 0.1$ | $\epsilon = 0.15$ | $\epsilon = 0.2$ | $\epsilon = 0.25$ | $\epsilon = 0.3$ |
|---|---|---|---|---|---|---|
| $\mathcal{IP}3$ | 95.46% | 83.6% | 57.67% | 29.96% | 12.99% | 4.81% |
| 1-vs-1 | 84.48% | 59.17% | 25.57 % | 7.91% | 2.36 % | 0.66% |
| 1-vs-All | 93.64% | 70.74% | 30.89% | 6.74% | 1.87% | 0.86% |
| Sparse | 86.12% | 58.67% | 22.65% | 5.4% | 0.63% | 0.01% |
| Dense | 95.17% | 84.08% | 62.95% | 43.54% | 28.6% | 16.34% |
| Multiclass | 94.35% | 70.29% | 21.72% | 2.19 % | 0.04 % | 0.0% |

**CIFAR10:** We evaluate the robustness of different codebooks on CIFAR10 by running 30-step PGD attack (with 5 different seeds); see Table 10. In this case, our $\mathcal{IP}3$ codebook outperforms all other codebooks including Dense codebook. Note that the underlying data-distribution has changed as we moved from MNIST to CIFAR10 dataset, and Dense codebook now shows lower performance than $\mathcal{IP}3$, particularly for larger perturbations of $\epsilon = 4/255$ and $\epsilon = 8/255$.

Table 10: Adversarial Accuracy of Nominally Trained Codebooks on CIFAR10.

| | $\epsilon = 2/255$ | $\epsilon = 4/255$ | $\epsilon = 8/255$ |
|---|---|---|---|
| $\mathcal{IP}3$ | 24.04% | 19.24% | 16.48% |
| 1-vs-1 | 4.65% | 0.11% | 0.0 % |
| 1-vs-All | 2.83% | 0.14% | 0.0% |
| Sparse | 5.05% | 0.08% | 0.0% |
| Dense | 24.2% | 12.79% | 11.63% |
| Multiclass | 15.46% | 2.55% | 0.27% |

Importantly, the adversarial accuracy achieved by our $\mathcal{IP}3$ is by no means trivial as under the exactly same setting other codebooks like 1-vs-1, 1-vs-All, Sparse do not show any robustness. In similar setting, a Multiclass CNN of similar network capacity also does not provide any robustness to adversarial perturbations. This demonstrates a rather impressive capability of ECOCs to handle adversarial perturbations even though the individual binary hypotheses are all nominally trained. Thus, our approach provides *robustness-by-design*, without making any specific assumptions about the adversary model in the design of codebook.

## 6 CONCLUSION AND FUTURE WORK

Our computational results demonstrate the merits of our optimal codebook design approach. Importantly, our IP-based formulation achieves small optimality gaps while maintaining tractability. This is possible mainly due the graph-theoretic structure the we exploited in applying the edge-clique-cover, which substantially reduced the constraint set of original IP formulation. In the nominal setting, our IP-generated *compact* codebooks outperform commonly used large codebooks on most datasets.

Furthermore, in the adversarial setting, our IP-generated codebooks achieve non-trivial robustness. This is surprising due to three reasons: (1) We do not employ any adversarial training; (2) Most other codebooks (except Dense) do not exhibit any robustness even when they use more than twice the number of columns; (3) The robustness that we obtain is not simply because of the large network capacity. To the best of our knowledge, we are the first ones to report that adversarial robustness can be achieved by a careful codebook design approach, while only using nominally trained classifiers.

Our results provide guidance for further research in the use of ECOCs for robust classification. We plan to study the effect of robustifying individual hypotheses with adversarial training. This could involve using a combination of nominally and adversarially trained hypotheses.

We believe that our approach avoids over-estimation of adversarial accuracy following Athalye et al. [2018] and Tramer et al. [2020], however our approach should be evaluated against other attacks such as decision-based attacks [Brendel et al., 2017]. To generate guarantees on adversarial accuracy of our ECOC based approach, extensions of our work within the framework of certifiable defenses [Wong and Kolter, 2018], [Zhang et al., 2019] are also desirable.

### Acknowledgements

We sincerely thank all the reviewers for their constructive and helpful feedback. The second author acknowledges support from AFSOR grant FA9550-19-1-0263.

### References

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1: 113–141, 2000.

Alper Atamtürk, George Nemhauser, and Martin Savelsbergh. Conflict graphs in solving integer programming problems. *European Journal of Operational Research*, 121:40–55, 02 2000.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, July 2018.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. 12 2017.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian J. Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *CoRR*, abs/1902.06705, 2019.

Alessio Conte, Roberto Grossi, and Andrea Marino. Clique Covering of Large Real-World Networks. In *31st Annual ACM Symposium on Applied Computing (SAC 2016)*, pages 1134–1139, Pisa, Italy, April 2016. ACM.

Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2):201–233, May 2002. ISSN 1573-0565.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2(1):263–286, January 1995. ISSN 1076-9757.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

Tianshi Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *International Conference on Computer Vision*, pages 2072–2079, 2011.

Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. 1990.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *ACM J. Exp. Algorithmics*, 13, February 2009. ISSN 1084-6654.

G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

Venkatesan Guruswami and Amit Sahai. Multiclass learning, boosting, and error-correcting codes. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT'99, pages 145–155, 1999.

Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In *Advances in Neural Information Processing Systems 10*, NIPS '97, pages 507–513, Cambridge, MA, USA, 1998. MIT Press. ISBN 0-262-10076-2.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

E. Kellerman. Determination of keyword conflict. *IBM Technical Disclosure Bulletin*, 16(2):544–546, 1973.

L. T. Kou, L. Stockmeyer, and C. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Commun. ACM*, 21:135–139, 1978.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015. ISSN 0028-0836.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2018.

Miguel Ángel Bautista Martin, Oriol Pujol, Fernando De la Torre, and Sergio Escalera. Error-correcting factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2388–2401, Oct 2018.

Manfred W. Padberg. On the facial structure of set packing polyhedra. *Math. Program.*, 5(1):199–215, December 1973. ISSN 0025-5610.

Morris Plotkin. Binary codes with specified minimum distance. *IRE Trans. Inf. Theory*, 6(4):445–450, 1960.

Oriol Pujol, Petia Radeva, and Jordi Vitria. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 28(6): 1007–1012, June 2006. ISSN 0162-8828.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5: 101–141, 2004.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23:828–841, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses, 2020.

Gunjan Verma and Ananthram Swami. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. In *Advances in Neural Information Processing Systems 32*, pages 8646–8656. 2019.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5286–5295. PMLR, July 2018.

Xiao Zhang, Lin Liang, and Heung-Yeung Shum. Spectral error correcting output codes for efficient multiclass recognition. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1111–1118, 2009.

Bianca Zadrozny. Reducing multiclass to binary by coupling probability estimates. In *Advances in Neural Information Processing Systems 14*, pages 1041–1048. MIT Press, 2002.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *CoRR*, abs/1906.06316, 2019.

B. Zhao and E. P. Xing. Sparse output coding for large-scale visual recognition. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3350–3357, 2013.