A BLOCKWISE PROBABILITY LOSS FUNCTION FOR MULTICLASS CLASSIFICATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Binary loss functions are very popular in multiclass classification learning algorithms due to their low computational complexity. On the other hand, some of them, such as softmax cross-entropy (CE) and binary cross-entropy (BCE) may impose restrictions on class separability or algorithm convergence. In standard multiclass classification, cross-entropy is typically used with one-hot targets, while the BCE reduces to the case where all outputs of the learned function are treated separately. We observe that if binary losses consider the whole output as a collection of separate binary outputs, they do not fully explore the extrinsic information from other components during training. Motivated by that, we present a new blockwise loss function that satisfies three basic properties: it is (i) not separable into binary subproblems, (ii) well defined for associating classes with binary vectors, which we refer to as binary code assignments, and (iii) obtained by comparing the output to the true class representation as well as to all other classes. We show that the proposed loss reduces to CE when one-hot coding is applied, while it reduces to a scaled version of BCE when property (iii) is simplified such that the loss compares the output only with the true class representation. Furthermore, when one-hot coding is replaced with error correcting output codes (ECOCs), the proposed loss achieves accuracy that is equal to or higher than that of classical bitwise losses. Experiments were conducted on the CIFAR-100 and Tiny ImageNet datasets for various losses from the literature and demonstrate the advantages of the new blockwise probability (BP) loss in combination with ECOCs.

1 INTRODUCTION

In multiclass classification problems, it is common to break the task into N binary outputs, where $N > 1$. Each binary output of the learning algorithm used for such a task may represent a particular feature or the class itself. A classical case is the one-hot coding, in which each of the M classes is associated with one binary output of the learned function, which implies that $N = M$. Dietterich & Bakiri (1994) observed that a better performance can be obtained by increasing the Hamming distances between the binary vectors representing the classes. The idea was to increase the separation between class representations by allowing the learned function to have a larger number $N > M$ of binary outputs. By doing that, it was possible to use an error correcting output code (ECOC) decoder at the last stage. As a generalization of this work Allwein et al. (2000) introduced the concept of a *loss-based decoder*, that implements the error correction by using a margin-based loss function for a given codebook matrix. While the focus of these works was in the decoding stage, at inference, our work highlights the role of the *training loss* and how it differs from the *decoding loss* used with ECOCs.

Two of the most popular loss functions used in multiclass classification are the binary cross-entropy (BCE) and the softmax cross-entropy (CE). While the BCE loss is applied to each output separately, the CE considers distances with respect to the whole set of outputs. We can then classify these losses as bitwise and blockwise, respectively. In parallel to the development of ECOCs, some authors investigated how the BCE and CE losses operate with the one-hot encoding. The negative class distraction problem in BCE was identified by Dong et al. (2019) and further investigated by Li et al. (2020) from the backpropagation equations, showing that the CE had advantages when compared to the classical form of the BCE. In the field of ECOCs, however, the CE loss cannot be used since the softmax stage is restricted to only the one-hot case. In this work, we propose a new loss function,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

called blockwise probability (BP), that can apply a "generalized softmax" at the output layer and still be used with any codebook. When the blockwise feature of the BP loss is removed, it is transformed into the BCE loss with a multiplicative factor. Also interestingly, when the one-hot encoding is applied to the BP loss, it reduces to the classical CE loss.

Experiments were run with a strong and a weak model over the CIFAR-100 and Tiny ImageNet datasets. The results demonstrate that the proposed loss performs better than the competitors when the same codebook is used. Furthermore, even randomly picked codebooks with compressed output space, $N < M$, can provide an accuracy close to one-hot, where $N = M$.

As the main contributions of this work we:

- Introduce the concept of bitwise and blockwise training losses;
- Propose a novel blockwise loss that generalizes CE to arbitrary binary codebooks;
- Prove that CE and BCE arise as special cases of the proposed BP loss.

2 Related Work

Error correcting codes have their roots in information theory, where sparse-graph codes like LDPC codes (Gallager, 1963) are capable of approaching Shannon's capacity limit with efficient iterative message passing decoders. We borrow some ideas from this field when designing our blockwise loss function: iterative decoders operate with log-likelihood ratios, which can be used to estimate blockwise probabilities, and they exchange extrinsic information between components (Hagenauer et al., 1996). The application of LDPC codes as ECOCs in learning algorithms has been considered in Marrocco & Tortorella (2016).

Following the initial phase of ECOC research (Dietterich & Bakiri, 1994; Allwein et al., 2000), Rifkin & Klautau (2004) revised several relevant works on ECOCs and argued that, surprisingly, the one-hot encoding is optimal if the model parameters are well-tuned. Some research shifted towards exploring the data for efficient application of ECOCs while other works focused on the use of ECOCs for robustness against adversarial attacks (Wan et al., 2022; Verma & Swami, 2019) and noisy outputs (Yu et al., 2024). The use of data was demonstrated empirically to be crucial for mapping codewords to classes (Evron et al., 2023). In Bautista Martin et al. (2018), the authors propose a problem-aware method for designing efficient ECOCs from the data. Our work, on the other hand, aims at defining a problem-agnostic loss function that performs well for any given codebook.

A known issue when classical bitwise loss functions are applied is that repeated columns in the codebook matrix result in identical score values at the corresponding outputs and the coding gain is limited. Powerful codes work best when N is large and M grows exponentially with N. For this reason, they are well-suited for extreme classification problems. In this domain it is shown in Evron et al. (2018) that ECOCs can be used to reduce the complexity compared to one-hot coding. The authors of Rocha & Goldenstein (2014) relax the independence assumption and take into account correlations in the codebook design. While we do not consider these aspects explicitly in this work, our blockwise loss function does not rely on separate scores at the output but considers them jointly during training. In Mettes et al. (2019) it is shown that the cosine similarity loss has comparable performance to CE loss in their hyperspherical prototype framework. Interestingly, this loss is also blockwise according to our definition, but it only takes into account the true class representation. By using the cosine similarity loss with softmax normalization, Kasarla et al. (2022) computes a loss that considers distances to all class representations. However, that approach is not well defined for binary class representations. In this work, we present a loss function that satisfies these characteristics and is binary in nature, meaning that it can be used within the context of ECOCs. Beyond that scope, we believe that such loss function may be relevant not only for ECOCs but also for compressed and structured output representations in modern neural networks.

3 Background: Error Correcting Output Codes

Consider a classification problem with M classes. A general classifier predicts the label $y \in \{1, \dots, M\}$ of the correct class as $\hat{y} = \mathbf{f}(\mathbf{x})$, where the function $\mathbf{f}$ is produced from a data set $\{\mathbf{x}^{(j)}, y^{(j)}\}$ by some learning algorithm that is suitable for multiclass prediction. In order to reduce

complexity, Dietterich & Bakiri (1994) and Allwein et al. (2000) proposed to decompose the overall problem into N binary subproblems with real valued output vector $\mathbf{z}(\mathbf{x}) \stackrel{\text{def}}{=} [z_1(\mathbf{x}), \ldots, z_N(\mathbf{x})] \in \mathbb{R}^N$. The elements of this output vector are called the scores and are predicted by the functions $z_1, \ldots, z_N$ that are produced from the same data set using a binary learning algorithm with input labels from $\{-1, +1\}$. In the context of neural networks the scores $z_n(\mathbf{x})$, $n = 1, \ldots N$, correspond to the logits, i.e., the raw outputs before, possibly, a non-linear activation function is applied. The assignment between classes and binary predictors is defined by an $M \times N$ codebook matrix $\mathbf{C}$ with elements $c_{mn} \in \{-1, 0, +1\}$ and codewords $\mathbf{c}_m$ as rows, where $m = 1, \ldots, M$. Rows correspond to classes and columns to scores produced by the binary predictors.

For example, consider the following two codebook matrices for $M = 4$ classes:

$$\mathbf{C}_{\text{one-hot}} = \begin{pmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{pmatrix} \qquad \mathbf{C}_{\text{all-pairs}} = \begin{pmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix} . \quad (1)$$

The left corresponds to one-hot coding (one-versus-all) and is defined by $c_{mn} = +1$ for $m = n$ and $c_{mn} = -1$ otherwise. As a consequence, each class is represented by a single score/logit. The right is an example of an ECOC with $N > M$ binary predictors that distinguish all possible pairs of classes (one-versus-one).

In this paper, we limit our experimental results to codebooks with $c_{mn} \in \{-1, 1\}$, while the loss function we propose is valid for general $c_{mn} \in \{-1, 0, +1\}$. Without any zero elements, a binary version of the codebook matrix $\hat{\mathbf{C}}$ with elements $\hat{c}_{mn} \in \{0, 1\}$ can be obtained by converting each element by

$$\hat{c}_{mn} = (1 + c_{mn})/2 \, . \quad (2)$$

After training, an output decoder predicts a class label $\hat{y}$ by comparing the vector of scores $\mathbf{z}(\mathbf{x})$ with all codewords in $\mathbf{C}$. The codeword $\mathbf{c}_m$ that is closest to the scores according to some given loss function $L(m, \mathbf{z}(\mathbf{x}), \mathbf{C})$ determines the final output $\hat{y} = m$.

The error correction capability of a codebook matrix is determined by d_{min}, the minimum Hamming distance $d_{\text{H}}(\mathbf{c}_m, \mathbf{c}_{m'})$ between any pair of codewords, i.e., the smallest number of positions where two codewords differ. The decoding radius $t = \lfloor \frac{d_{\text{min}} - 1}{2} \rfloor$ defines the guaranteed number of correctable errors. Generally, larger values of t can be obtained by increasing the code length N or reducing the code rate $R = (\log_2 M)/N$. For one-hot coding, $d_{\text{min}} = 2$ for any number of classes M, which implies that $t = 0$ (no error correction).

4 Training losses: Bitwise and Blockwise Components

In the most general case, the input to the learning algorithm is $(\{\mathbf{x}^{(j)}, y^{(j)}\}, \mathbf{C})$ where j is the sample index and $y^{(j)}$ is the correct label of sample $\mathbf{x}^{(j)}$. For a given j, the codebook matrix $\mathbf{C}$ is used for comparing the output $\mathbf{z}(\mathbf{x}^{(j)})$ of the learning algorithm with the different classes $m = 1, \ldots, M$. The comparison is based on some loss function $L(m, \mathbf{z}(\mathbf{x}^{(j)}), \mathbf{C})$. We call a loss function *bitwise*, and denote it by L_{bit}, if its derivative with respect to the n^{th} logit depends only on that logit and its corresponding representation in the codebook matrix

$$\frac{\partial L_{\text{bit}}}{\partial z_n(\mathbf{x}^{(j)})} = f(c_{y^{(j)}n}, z_n(\mathbf{x}^{(j)})) \, . \quad (3)$$

Due to that property, a generic bitwise loss will assume the following form

$$L_{\text{bit}} = \sum_{n=1}^{N} \ell_n(c_{y^{(j)}n}, z_n(\mathbf{x}^{(j)})) = \sum_{n=1}^{N} \ell_n(c_{y^{(j)}n} \cdot z_n(\mathbf{x}^{(j)})), \quad (4)$$

where $\ell_n(c_{y^{(j)}n} \cdot z_n(\mathbf{x}^{(j)}))$ is a collection of functions for each logit, not necessarily equal to each other. In the second equality, following Allwein et al. (2000), we considered only the class of margin-based loss functions. Although Allwein's work is focused on the decoding stage, we would

like to make an important distinction between the training and decoding losses. For the purposes of our analysis, the *training loss* for one sample of a margin-based classifier can be expressed as

$$L(y^{(j)}, \mathbf{z}(\mathbf{x}^{(j)}), \mathbf{C}) = \sum_{n=1}^{N} \ell_n(c_{y^{(j)}n} \cdot z_n(\mathbf{x}^{(j)})) + L_{\text{block}}(y^{(j)}, \mathbf{z}(\mathbf{x}^{(j)}), \mathbf{C}) \stackrel{\text{def}}{=} L_{\text{bit}} + L_{\text{block}} \,. \quad (5)$$

Equation 5 separates bitwise and blockwise contributions to the training loss. It is very important to note, however, that the bitwise and blockwise decomposition is not unique, since one can always add and decrease a new bitwise term to the original loss and incorporate the terms to the old bitwise and blockwise components. In addition, constants can be multiplied or added to the bitwise loss, as it still keeps the property from equation 3.

When $L_{\text{block}} = 0$, the derivatives $\partial L / \partial z_n$ depend only on the corresponding output z_n and label component $c_{y^{(j)}n}$, and the learning problem decomposes into independent binary subproblems. In contrast, a nonzero L_{block} couples the outputs through their gradients, making the loss sensitive to the structure of the output block and enabling that dependencies between components are used for error correction. Some losses may not be separable into these two components. For example, the squared cosine (or cosine similarity) loss, used in the experiments later, can be represented by

$$L_{\cos} = (1 - \cos\theta_{\mathbf{z},\mathbf{c}_y})^2 = \left(1 - \sum_{n=1}^{N} c_{yn} \cdot z_n\right)^2 \,, \quad (6)$$

considering that $\mathbf{z}$ and $\mathbf{c}_y$ were already normalized. Due to the squaring, the derivatives $\frac{\partial L_{\cos}}{\partial z_n}$ do not depend on z_n only, so that it is a blockwise loss for which we cannot extract a bitwise component. For ease of notation, like in equation 6, we will omit the sample index in the equations for the rest of the text.

5 Decoding Losses

A loss-based decoder acts at inference stage by picking the class alternative that has the minimum loss, considering the scores at the output of the trained function. The predicted label $\hat{y}$ is then defined as

$$\hat{y} = \arg\min_m L(m, \mathbf{z}(\mathbf{x}), \mathbf{C}). \quad (7)$$

In general one picks the decoding loss to be the same as the training loss, since the model learned how to minimize distances based on that metric. From that, normally it is possible to simplify the decoding loss. Consider, for example, the special case in which, in the training loss, the term L_{block} from equation 5 does not depend on the true label of the class, i.e. $L_{\text{block}} = L_{\text{block}}(\mathbf{z}(\mathbf{x}^{(j)}), \mathbf{C})$. In such case, L_{block} does not affect the inference stage and equation 7 can be simplified as

$$\hat{y} = \arg\min_m L_{\text{bit}}(m, \mathbf{z}(\mathbf{x}), \mathbf{c}_m) \,, \quad (8)$$

The L_{bit}-term, in this case, represents the decoding loss, which is in general more simple than the training loss L. As an example, consider the CE training loss:

$$L_{\text{CE}} = -\ln\left(\frac{e^{z_y}}{\sum_{m=1}^{M} e^{z_m}}\right) = -z_y + \ln\sum_{m=1}^{M} e^{z_m}. \quad (9)$$

From that, one can have a convenient decomposition of blockwise and bitwise terms. At inference:

$$\ell_m = \begin{cases} -z_n, & \text{if } n = m \\ 0, & \text{if } n \neq m \end{cases} \quad , \quad L_{\text{bit}} = -z_m \quad \text{and} \quad L_{\text{block}} = \ln\sum_{m'=1}^{M} e^{z_{m'}} \,. \quad (10)$$

The separation of bitwise and blockwise above is not unique but it is useful in this case to let the bitwise loss be simply the decoding loss. The great advantage of this loss is that it is very easy to decode:

$$\hat{y} = \arg\min_m -z_m = \arg\max_m z_m \quad (11)$$

and the decoding loss is simply $-z_m$.

In essence, an ECOC that uses a simple bitwise decoding loss at inference does not necessarily require that the same bitwise loss is used at training. One can use a more sophisticated blockwise loss at training and remove the blockwise term at inference.

6 THE BLOCKWISE PROBABILITY LOSS FUNCTION

In this section we define the BP loss function within the context of neural networks. We first present three desired properties for loss functions considering the use in error correction, and then, based on those, the BP loss is derived and connected to existing methods.

6.1 DESIRED PROPERTIES

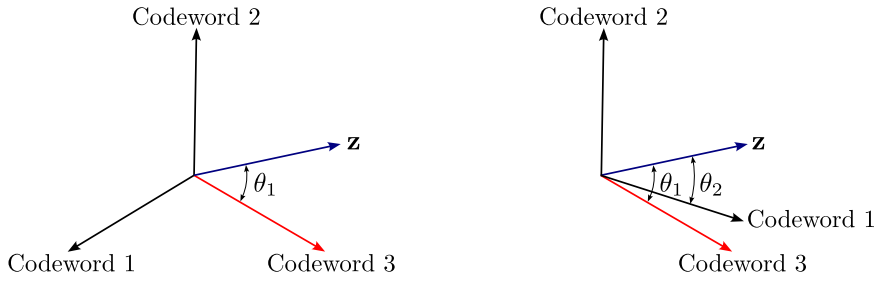
We would like to formulate a training loss function with three desired properties.

It is binary in nature. The loss must be well defined for codebooks in the binary domain $\hat{\mathbf{C}} \in \{0, 1\}^{M,N}$, that converts to $\mathbf{C}$ by equation 2. That is required so that it can be used for ECOCs. Given that, all class representations will be binary.

It is defined blockwise. The intuition is drawn from information theory. If we consider that each logit carries information about itself (intrinsic) and about the rest of the logits (extrinsic), it is beneficial that the loss also makes use of both such information (not only the intrinsic). The advantage of this approach is that equal columns in the codebook matrix will not necessarily generate equal logits, although high correlation is possible.

It evaluates a distance not only to the correct codeword but also to the other candidates. This property is better understood with some examples. Suppose first that $\mathbf{c}_i = [-1, +1, +1, +1, -1]$ and $\mathbf{c}_j = [-1, -1, +1, +1, -1]$. The two codewords are close to each other as they differ only in the second bit. If i is the correct class for $\mathbf{x}$ then the loss must attain a sufficiently small value so that the algorithm is capable of differentiating class i from class j. A codebook with low minimum distance between its codewords requires more accurate inference. On the other hand, if $\mathbf{c}_i$ and $\mathbf{c}_j$ are further away from each other (differ at more elements), the loss function can be more relaxed, it does not need to be very precise in order to differentiate class i and class j.

To illustrate this, Figure 1 shows two different codebooks designed for the same task. Consider the angular displacement θ_1 between the codeword $\mathbf{c}_y$ corresponding to the true class label $y = 3$ (Codeword 3, in red) and the logits $\mathbf{z}$ as a separation measure of a particular loss function. Codebook 1 could be the representation of one-hot encoding, since all codewords are equidistant, while Codebook 2 could be any random encoding in which Codeword 1 and Codeword 3 ended up close to each other. After training the model with the same loss function and only using input samples with $y = 3$, both learned functions provide the same logits $\mathbf{z}$ when some input $\mathbf{x}$ is applied. In the first codebook, the considered loss works properly as θ_1 is smaller than the angles to other classes. The algorithm then converged to a sufficiently good operation point. For Codebook 2, on the other hand, this is not true since $\theta_2 < \theta_1$, causing a wrong class inference by the loss-based decoder. The loss function is not well suited in this case because it is not training the outputs by comparing the distance to Codewords 1 and 2 as well as the ground truth Codeword 3. The loss function lacks the third property and it fails at inference because more precision is required to differentiate Codewords 1 and 3.



(a) Codebook 1 (b) Codebook 2

Figure 1: Codebook 1 is equally distributed with maximal separation, whereas in Codebook 2 the Codewords 1 and 3 are closer to each other than to Codeword 2.

The presence of the third property in a loss function implies that it has a blockwise component, but the opposite is not true. The cosine similarity loss from equation 6, which is a blockwise loss, is an example: the distance metric only looks at row y of the codebook, not to the whole $\mathbf{C}$.

An example of a loss function that satisfies all three conditions is the CE with softmax. Note from equation 9 and equation 10 how the softmax layer creates the blockwise term $L_{\text{block,BP}}$. That loss compares the metric e^{z_y} to all other class representations. The problem with CE is that it is only defined for one-hot coding, which is not generic enough to be used within the context of ECOCs.

Finally, by using the three properties above, we expect that the loss makes the learning algorithm converge to logits that are correctly classified when the loss-based decoder is applied. The bitwise losses are not able to do that since they make each logit converge independently, agnostic to how the loss-based decoder will perform and forcing each binary output to be as close as possible to the final value. From these observations, we expect that the number of bits corrected by the decoder with the blockwise loss is higher in general than any bitwise decoder.

6.2 Blockwise Probability

Let $z_n(\mathbf{x})$, $n = 1, \ldots, N$, denote the raw logits of the learning algorithm. We interpret each logit as a log-likelihood ratio and define the corresponding bit probabilities as $q_n = \sigma(z_n)$ and $\bar{q}_n = \sigma(-z_n)$, where $\sigma(x) = 1/(1 + \exp(-x))$. The logits are then $z_n = \ln(q_n/\bar{q}_n)$. A class score ρ_m can be defined by simply multiplying all bit probabilities associated with the target codeword $\mathbf{c}_m$:

$$\rho_m = \prod_{t:c_{mt}=0} (1/2) \cdot \prod_{r:c_{mr}=1} q_r \cdot \prod_{s:c_{ms}=-1} \bar{q}_s \tag{12}$$

$$= \prod_{t:c_{mt}=0} \sigma(0) \cdot \prod_{r:c_{mr}=1} \sigma(z_r) \cdot \prod_{s:c_{ms}=-1} \sigma(-z_s) \;=\; \prod_{n=1}^{N} \sigma(c_{mn} \cdot z_n)\,. \tag{13}$$

In equation 13, $c_{mn} = 0$ indicates that the corresponding output carries no class information and contributes a factor $\sigma(0) = 1/2$. In addition, ρ_i is not a probability yet, since $\sum_{m=1}^{M} \rho_m \neq 1$, in general. After normalization, the codeword probability is obtained as

$$p_m = \frac{\rho_m}{\sum_{m'=1}^{M} \rho_{m'}}\,. \tag{14}$$

Note that the normalization above guarantees that all codewords will have $p_m \geq 0$ and that at least one of them has probability larger than zero since $\sum_{m=1}^{M} p_m = 1$. During the training, the loss must then account only for the probability p_y that the output logits represent the correct row of the codebook matrix. We can then apply the negative logarithm so that the loss is zero for probability 1 and has no limit on the positive real numbers. From that, the blockwise probability loss L_{BP} is defined as

$$L_{\text{BP}} = -\ln p_y = -\sum_{n=1}^{N} \ln \sigma(c_{yn} \cdot z_n) + \ln \sum_{m=1}^{M} \rho_m\,. \tag{15}$$

Note that equation 15 is in the form of equation 5, so we can define

$$L_{\text{bit,BP}} \stackrel{\text{def}}{=} -\sum_{n=1}^{N} \ln \sigma(c_{yn} \cdot z_n) \quad \text{and} \quad L_{\text{block,BP}} \stackrel{\text{def}}{=} \ln \sum_{m=1}^{M} \rho_m\,. \tag{16}$$

The derivatives with respect to the logits are then

$$\frac{\partial L_{\text{bit,BP}}}{\partial z_n} = -\sigma(-c_{yn} \cdot z_n) \cdot c_{yn} \tag{17}$$

and

$$\frac{\partial L_{\text{block,BP}}}{\partial z_n} = \frac{1}{\sum_{i=1}^{M} \rho_i} \sum_{m=1}^{M} \rho_m \sigma(-c_{mn} \cdot z_n) \cdot c_{mn}\,. \tag{18}$$

Thus,

$$\frac{\partial L_{\text{BP}}}{\partial z_n} = \frac{\partial L_{\text{bit,BP}}}{\partial z_n} + \frac{\partial L_{\text{block,BP}}}{\partial z_n} = \sum_{m=1}^{M} p_m \cdot \sigma(-c_{mn} \cdot z_n) \cdot c_{mn} - \sigma(-c_{yn} \cdot z_n) \cdot c_{yn}\,. \tag{19}$$

It is interesting to observe that, differently from the CE loss, the p_y is the probability that the whole block of output logits represents the class y, instead of the probability that the y^{th} output is correct. We refer to A.6 in the supplementary material for more implications of that loss.

6.3 SPECIAL CASES

The CE loss It is now natural to think that the CE with softmax is a special case of the BP loss function presented. Consider the one-hot $\mathbf{C}^{M\times N}$ codebook matrix with $+1$ elements on the main diagonal and -1 otherwise. In that case, the probability of the i^{th} output bit to be 1 is exactly the probability that the correct class is the i^{th} row of the codebook matrix, since there is only one codeword with +1 in the i^{th} column of $\mathbf{C}$. From that, we can recognize that

$$L_{\text{BP}_{\text{one-hot}}} = L_{\text{CE}} \,. \tag{20}$$

From another point of view, one could say that the BP loss is the same as the CE loss except that the traditional softmax is replaced by a "generalized" softmax activation function as given by equation 14. The formal proof of the result above is given in A.2.

The BCE loss Consider the BCE loss in terms of the codebook notation in equation 1:

$$L_{\text{BCE}} = -\frac{1}{N}\sum_{n=1}^{N}\left[\hat{c}_{yn}\ln\sigma(z_n) + (1-\hat{c}_{yn})\ln(1-\sigma(z_n))\right] \tag{21}$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\left[\frac{c_{yn}+1}{2}\ln\sigma(z_n) + \frac{1-c_{yn}}{2}\ln(\sigma(-z_n))\right]. \tag{22}$$

The term in the summation can be rewritten as

$$\frac{c_{yn}+1}{2}\ln\sigma(z_n) + \frac{1-c_{yn}}{2}\ln(\sigma(-z_n)) = \begin{cases}\ln\sigma(z_n), & \text{if } c_{yn}=+1\\ \ln\sigma(-z_n), & \text{if } c_{yn}=-1\end{cases}$$
$$= \ln\sigma(c_{yn}\cdot z_n)$$

Therefore,

$$L_{\text{BCE}} = -\frac{1}{N}\sum_{n=1}^{N}\ln\sigma(c_{yn}\cdot z_n) = \frac{L_{\text{bit,BP}}}{N} \,. \tag{23}$$

The result above shows that the BCE loss is the bitwise component of the BP loss (except for the $1/N$ factor), when $c_{yn}\in\{-1,+1\}$. Its performance should be comparable to using only the term $L_{\text{bit,BP}}$ from the BP loss.

7 EXPERIMENTS

This section compares the BP loss with other losses commonly used for multiclass classification problems. The comparison is made in terms of accuracy for four types of codebooks: one-hot ($N=M$), a random codebook with $N=M$, a random codebook with $N<M$ and a *degenerate* codebook with $N=M$. All random codebooks were created with each element drawn from the uniform distribution over $\{0,1\}$ and converted by equation 2 when applicable. Equal rows are eliminated and replaced by new randomly generated rows, until the codebook is complete. The *degenerate* codebook is created by first generating two binary random codebook matrices $\hat{\mathbf{C}}_1$ and $\hat{\mathbf{C}}_2$ with elements and dimension $\{0,1\}^{\frac{M}{2}\times\frac{M}{2}}$ and assigning

$$\hat{\mathbf{C}} = \begin{pmatrix}\hat{\mathbf{C}}_1 & \hat{\mathbf{C}}_2\\ \hat{\mathbf{C}}_1\oplus\mathbf{I} & \hat{\mathbf{C}}_2\end{pmatrix}. \tag{24}$$

From that, the $\mathbf{C}$ codebook is obtained by equation 2. For this *degenerate* codebook, each codeword has minimum distance $d_{\text{min}}=1$ to some other codeword. With that we test whether the $L_{\text{bit,BP}}$-term really is of importance for distinguishing the classes that are close to each other.

Two architecture models, the ResNet34 and a custom model called CNN0, were tested on the CIFAR-100 and the Tiny ImageNet datasets. The ResNet34 represents a strong classifier while the CNN0 represents the weak classifier. The CIFAR-100 has $M=100$, whereas Tiny ImageNet has $M=200$. With the random codebooks of $N=75$ (CIFAR-100) and $N=100$ (Tiny ImageNet), we also evaluate whether the output dimensionality can be reduced without a significant loss of accuracy.

All losses, their normalization variants and the models are further detailed in A.1. Each configuration of codebook, dataset and loss was run at least three times, with different seeds.

Table 1: Top-1 accuracy (%) for CNN0 on CIFAR-100 and Tiny ImageNet. Values are reported as mean (max).

Dataset	Loss	one-hot	random ($N = 100$)	random ($N = 75$)	degenerate
CIFAR-100	L_{CE}	40.52 (40.85)	–	–	–
	L_{CE} w/ logit norm	39.16 (39.98)	–	–	–
	L_{BP}	**40.97 (41.10)**	36.07 (37.21)	32.86 (33.35)	32.78 (34.22)
	L_{BP} w/ logit norm	39.01 (39.27)	**39.61 (40.28)**	**35.87 (36.42)**	**34.90 (35.13)**
	$L_{\mathrm{bit,BP}}$	39.11 (39.87)	22.35 (23.88)	22.49 (23.80)	18.95 (19.25)
	$L_{\mathrm{bit,BP}}/N$	38.78 (39.68)	24.12 (24.35)	23.89 (23.97)	19.36 (19.47)
	L_{BCE}	39.99 (40.81)	23.19 (23.49)	23.95 (24.56)	19.13 (19.19)
	L_{MSE}	22.30 (22.57)	21.82 (22.47)	22.77 (23.38)	18.48 (19.65)
	L_{MSE} w/ logit norm	24.04 (25.05)	27.32 (28.04)	26.55 (26.76)	19.07 (19.53)
	$L_{\cos}$ w/ logit norm	33.51 (33.91)	28.29 (28.64)	27.40 (27.86)	18.31 (18.81)
		one-hot	random ($N = 200$)	random ($N = 100$)	degenerate
Tiny ImageNet	L_{CE}	25.89 (26.47)	–	–	–
	L_{CE} w/ logit norm	24.94 (25.31)	–	–	–
	L_{BP}	**26.40 (26.76)**	22.52 (22.81)	16.78 (17.25)	19.35 (20.74)
	L_{BP} w/ logit norm	25.32 (25.66)	**25.95 (26.50)**	**18.43 (18.93)**	**21.19 (21.68)**
	$L_{\mathrm{bit,BP}}$	25.01 (25.45)	11.86 (12.19)	11.38 (11.73)	9.92 (10.03)
	L_{BCE}	24.71 (25.98)	12.50 (13.06)	11.37 (11.62)	9.75 (9.94)
	L_{MSE}	4.45 (5.80)	12.12 (12.26)	11.21 (11.67)	9.09 (9.53)
	L_{MSE} w/ logit norm	8.32 (8.62)	14.76 (15.25)	13.42 (14.17)	10.28 (10.61)
	$L_{\cos}$ w/ logit norm	18.83 (19.13)	15.70 (15.82)	13.43 (13.68)	10.00 (10.02)

7.1 Accuracy

Tables 1 and 2 show the average and maximum accuracies of the different losses for all four codebooks. The differences between the CE loss and the BP loss (and their logit-normalized versions) in one-hot are minimal and only due to numerical implementation. Across all codebooks, except for one-hot, the BP loss with logit normalization provides the highest accuracy among the losses tested, for both datasets. For the one-hot case, the CE and BP losses and their logit-normalized versions are equally good and produce the highest accuracies. Some loss functions have better performance with non one-hot codebooks. For instance, the MSE and BCE with ResNet34 perform better at random codebooks.

The proposed L_{BP} is the loss with the largest accuracy when using the *degenerate* codebook. This is an evidence that the BP loss balances out the weaknesses of the codebook, as mentioned by property 3. Overall, the results of the BP loss change much less for different codebooks than the other losses, but still the *degenerate* codebook produces the lowest scores for that loss.

In Table 2 the $L_{\mathrm{bit,BP}}/N$ was left in the CIFAR-100 for comparison with the L_{BCE}. The comparison supports the equivalence between $L_{\mathrm{bit,BP}}/N$ and L_{BCE}. The way these losses were implemented numerically differs slightly, hence the small differences in accuracy.

For the Tiny ImageNet dataset with ResNet34, the BP loss does not present a big advantage when the random codebooks are used, especially when compared to the MSE loss. For that same dataset, the BP loss looks more useful with the weak model than with the strong model. The MSE and BCE losses do not achieve good performances for the bad model CNN0, in general. Another benefit of the BP loss can be seen by the codebook with $N < M$, in which the number of outputs is reduced and the performance is similar to the other codebooks.

The *degenerate* codebook never performs better than the $N < M$ codebooks for all losses except for the BP loss. This seems to point out that the BP loss can better extract the extrinsic information of every output.

8 Discussions and Limitations

There is still a lot to be investigated about the BP loss and how its error correction capabilities operate when more challenging conditions are required. One of the important limitations of the BP loss is that it does not scale very well with the number of classes, since at each training iteration the loss requires the calculation of all blockwise probabilities, instead of less costly losses that use only the target class representation in the calculation. In order to account for that, the BP loss can be simplified to estimate the blockwise probabilities by considering only the closest neighbors at training. At

Table 2: Top-1 accuracy (%) for ResNet34 on CIFAR-100 and Tiny ImageNet. Values are reported as mean (max).

Dataset	Loss	one-hot	random ($N = 100$)	random ($N = 75$)	degenerate
CIFAR-100	L_{CE}	56.57 (57.04)	–	–	–
	L_{CE} w/ logit norm	56.09 (56.79)	–	–	–
	L_{BP}	**56.91 (57.81)**	47.35 (49.60)	46.72 (48.94)	47.32 (49.05)
	L_{BP} w/ logit norm	56.47 (57.13)	**56.43 (56.86)**	**55.92 (56.50)**	**55.96 (56.61)**
	$L_{\mathrm{bit,BP}}$	45.05 (57.37)	49.93 (51.12)	50.14 (51.82)	42.65 (43.16)
	$L_{\mathrm{bit,BP}}/N$	32.39 (32.94)	47.99 (48.30)	47.15 (47.20)	42.20 (42.33)
	L_{BCE}	32.30 (32.79)	47.66 (48.37)	47.69 (49.13)	42.10 (42.90)
	L_{MSE}	26.08 (28.16)	52.66 (53.11)	52.98 (53.94)	47.55 (47.93)
	L_{MSE} w/ logit norm	25.29 (27.46)	52.87 (53.43)	52.04 (52.82)	51.26 (51.82)
	$L_{\cos}$ w/ logit norm	52.01 (52.55)	48.78 (49.56)	47.71 (48.23)	43.79 (45.55)
		one-hot	random ($N = 200$)	random ($N = 100$)	degenerate
Tiny ImageNet	L_{CE}	45.34 (49.29)	–	–	–
	L_{CE} w/ logit norm	47.01 (49.36)	–	–	–
	L_{BP}	43.68 (48.87)	42.42 (44.17)	39.93 (42.29)	39.96 (40.71)
	L_{BP} w/ logit norm	45.67 (49.25)	**48.58 (49.68)**	**45.47 (49.01)**	**47.95 (49.56)**
	$L_{\mathrm{bit,BP}}$	**48.50 (49.73)**	41.29 (42.38)	40.94 (42.77)	37.30 (38.68)
	L_{BCE}	15.94 (23.95)	46.01 (49.19)	33.04 (48.47)	31.08 (45.55)
	L_{MSE}	17.40 (20.64)	45.10 (45.95)	44.14 (46.42)	37.66 (41.92)
	L_{MSE} w/ logit norm	17.79 (21.00)	45.93 (48.97)	45.31 (48.02)	44.57 (46.61)
	$L_{\cos}$ w/ logit norm	46.46 (48.93)	43.05 (47.50)	41.96 (45.41)	32.83 (34.16)

testing, a more sophisticated binary loss-based decoder can then be used, such as soft-output decoders extensively used in communication systems. In addition, it is still open for investigation whether softmax cross-entropy loss, which is applied to one-hot encoding, is actually optimal or not in terms of accuracy. Still, even if that might be the case, the use of codebooks different from one-hot is essential for many ECOC applications, such as robustness against adversarial attacks, multi-label classification, prototypical learning, or among other areas of machine learning.

9 Conclusion

In this paper, we propose a new binary loss function that is more flexible than CE, which is typically applied to one-hot coding only. This new loss function is formulated with respect to the entire set of logits for a learning algorithm. On the path to this new loss function, we first introduce the concepts of blockwise and bitwise losses, borrowed from communication systems terminology, and the concept of codebooks (Dietterich & Bakiri, 1994; Allwein et al., 2000). Although bitwise losses may be used as loss-based decoders, it is beneficial to apply blockwise during training, making use of extrinsic information from each logit. The resulting loss, which we refer to as blockwise probability loss L_{BP}, can be viewed as a blockwise version of the BCE loss or a generalized CE loss for arbitrary codebook design; or even the CE with a "generalized softmax" layer applied.

By evaluating distances over the entire codebook, the proposed loss function is capable of adapting to different codebook designs. The BP loss can be used under any binary codebook design, which makes it a good candidate to be used in ECOCs.

With the use of the BP loss, one can also compress the highly redundant one-hot coding scheme to less outputs, simplifying the code assignments and storing less data if the number of classes is very high. On the other hand, the BP loss gets more complex computationally as the number of classes increases. Variations of that loss that account for a large number of classes are an open problem for future work in the subject.

AI use statement

In this work, we used LLMs for generating the code that parses and plots the results as well as pointing out grammatical mistakes in the text. In addition, LLMs were used to troubleshoot and add comments to the code that runs both training and inference of the neural networks in this paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

We provide the source code to the reviewers as supplementary material. The code will be available online as soon as the paper is published.

References

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 9–16, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

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, 2018.

T. G. Dietterich and G. Bakiri. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1994.

Qi Dong, Xiatian Zhu, and Shaogang Gong. Single-label multi-class image classification by deep logistic regression. In *AAAI Conference on Artificial Intelligence*, 2019.

Itay Evron, Edward Moroshko, and Koby Crammer. Efficient Loss-Based Decoding on Graphs for Extreme Classification. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Itay Evron, Ophir Onn, Tamar Weiss, Hai Azeroual, and Daniel Soudry. The Role of Codeword-to-Class Assignments in Error-Correcting Codes: An Empirical Study. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pp. 8053–8077. PMLR, 2023.

Robert G. Gallager. *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA, 1963.

J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Transactions on Information Theory*, 42(2):429–445, 1996.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition . In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Tejaswi Kasarla, Gertjan J. Burghouts, Max van Spengler, Elise van der Pol, Rita Cucchiara, and Pascal Mettes. Maximum class separation as inductive bias in one matrix. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario, 2009.

Yann Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS231N*, 2015.

Xiangrui Li, X. Li, Deng Pan, and D. Zhu. On the learning property of logistic and softmax losses for deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2020.

C. Marrocco and F. Tortorella. Exploiting coding theory for classification: An LDPC-based strategy for multiclass-to-binary decomposition. *Information Sciences*, 357:88–107, 2016.

Pascal Mettes, Elise van der Pol, and Cees Snoek. Hyperspherical prototype networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5: 101–141, 2004.

Anderson Rocha and Siome Klein Goldenstein. Multiclass From Binary: Expanding One-Versus-All, One-Versus-One and ECOC-Based Approaches. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):289–302, 2014.

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*, 2019.

Li Wan, Tansu Alpcan, Emanuele Viterbo, and Margreta Kuijper. Efficient error-correcting output codes for adversarial learning robustness. In *ICC 2022 - IEEE International Conference on Communications*, pp. 2345–2350, 2022.

Anlan Yu, Shusen Jing, Ning Lyu, Wujie Wen, and Zhiyuan Yan. Error correction output codes for robust neural networks against weight-errors: A neural tangent kernel point of view. *Advances in Neural Information Processing Systems 37*, 2024.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith (eds.), *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1–87.12. BMVA Press, September 2016.

A Appendix

A.1 Setup

The experiments were conducted on the CIFAR-100 (Krizhevsky & Hinton, 2009) and Tiny ImageNet (Le & Yang, 2015) datasets. CIFAR-100 comprises 60 000 images distributed over 100 classes, where each image is of size $32 \times 32 \times 3$. Of all the images in the dataset, 50 000 were used for training and the remaining 10 000 were used for testing. The Tiny ImageNet comprises 120 000 images distributed over 200 classes, each image of size $64 \times 64 \times 3$. The dataset contains 100 000 training images and 10 000 validation images which were used for testing. We experimented with two different deep neural network architectures: ResNet34 (He et al., 2016) (Zagoruyko & Komodakis, 2016) and a customized CNN, referred to as CNN0. The CNN0 comprises two convolutional layers with 32 and 64 filters, respectively, with kernel size 3×3, ReLU activation and same padding for both. The first convolutional layer is followed by a max pooling layer. The second convolutional layer is followed by a global average pooling layer. Subsequent to the global average pooling layer, there is a final fully connected layer with N outputs. We train the models for 500 epochs using the RMSprop optimizer with a learning rate of 0.001, batch size of 512, no pre-training and no gradient clipping.

Results are presented for the following losses: blockwise probability L_{BP}, squared cosine $L_{\cos}$ equation 6 (Mettes et al., 2019), L_{BCE} equation 22, L_{CE} and the MSE loss applied bitwise: $L_{\mathrm{MSE}} = \sum_{n=1}^{N}(c_{yn} - z_n)^2/N$. The CE loss already includes the softmax layer.

The squared cosine loss is not properly defined for one-hot coding. Nevertheless, in order to provide some basis for comparison, we used the $\hat{\mathbf{C}} \in \{0,1\}^{M,N}$ codebook in that case. The reason is that, if the matrix $\mathbf{C}$ given in equation 1 is used, the angles between class representations tend to zero as N increases, and the performance of that loss gets worse. As for the other codebooks with the squared cosine loss, each row of the codebook matrix $\mathbf{C}$ was normalized to one, so that all rows lie on the unit sphere. The logits are also normalized to one.

The CE does not require any codebook defined in order to be computed, it simply uses equation 9, although the one-hot coding is in fact underlying that loss definition. The BCE loss was computed considering equation 22, meaning that the codebook used is $\hat{c}_{mn} \in \{0,1\}$. For all other combinations of losses and codebooks, the $\mathbf{C} \in \{-1,+1\}^{M,N}$ was used.

In order to avoid a local minimum at the all-zero LLRs (logits), normalization was applied to the BP loss. As a consequence of the results in equation 28, the CE also benefits from logit normalization. Such normalization is implemented so that the output logits have norm $\sqrt{N}$, the same as any row of $\mathbf{C}$, considering that it is filled only with $\{-1,+1\}$. The only exception is the normalization of the squared cosine loss, mentioned above. For a fair comparison, a variation of MSE with this normalization was implemented as well.

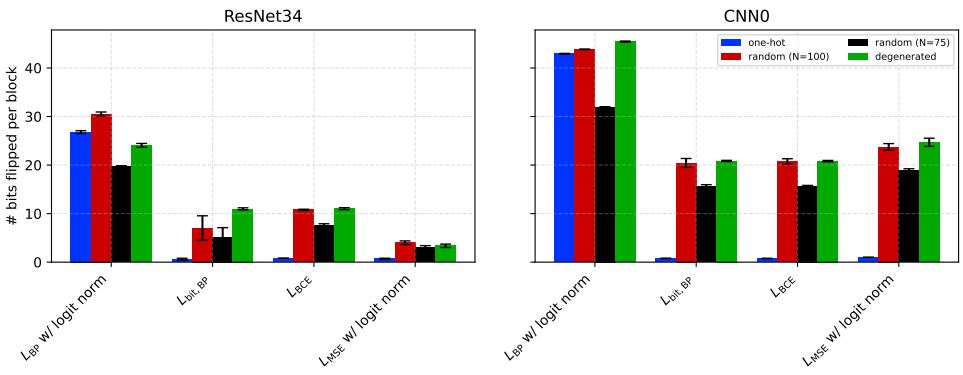


Figure 2: Number of flipped bits per sample by using the loss-based decoder, for different codebooks. From left to right: one-hot, random codebook with $N = 100$, random codebook with $N = 75$ and the degenerate codebook.

A.2 CE LOSS OBTAINED FROM THE BP LOSS

Consider the one-hot coding as mentioned in Section 6.3. Compute ρ_y from equation 13:

$$\rho_y = \sigma(c_{yy} \cdot z_y) \prod_{n \neq y} \sigma(c_{yn} \cdot z_n) = \sigma(z_y) \prod_{n \neq y} \sigma(-z_n) \, . \tag{25}$$

Then p_y is given by

$$p_y = \frac{\sigma(z_y) \prod_{n \neq y} \sigma(-z_n)}{\sum_{m=1}^{M} \sigma(z_m) \prod_{n \neq m} \sigma(-z_n)} \, . \tag{26}$$

We know that $\sigma(-\alpha)/\sigma(\alpha) = e^{-\alpha}$, so the right hand side can be rewritten by dividing numerator and denominator by the numerator

$$p_y = \frac{1}{\sum_{j=1}^{N} e^{z_j - z_y}} = \frac{e^{z_y}}{\sum_{j=1}^{N} e^{z_j}} \, , \tag{27}$$

where we made use of $M = N$ for one-hot coding. The equation above is exactly the application of the softmax activation function. Therefore,

$$L_{\mathrm{BP}_{\mathrm{one-hot}}} = -\ln \frac{e^{z_y}}{\sum_{j=1}^{N} e^{z_j}} = L_{\mathrm{CE}} \, . \tag{28}$$

A.3 ERROR-CORRECTION CAPABILITY

The error correction capability of each loss was put to test for the CIFAR-100 dataset. The number of bits flipped by the loss-based decoder for a particular sample $\mathbf{x}^{(j)}$ can be obtained as $\sum_{n=1}^{N} [\![c_{\hat{y}^{(j)} n} \cdot z_n < 0]\!]$. This quantity counts the number of bits the hard output $\mathrm{sign}(z_n)$ differs from the decoded codeword. Here $[\![\pi]\!] = 1$ if the predicate π holds and zero otherwise. Figure 2 shows the number of flipped bits per sample input for the losses tested. The losses without logit normalization were not presented for clarity and the squared cosine loss has no clear definition of "number of flipped bits", since each logit lies in the real numbers domain and also they cannot be interpreted as LLRs. Note that the use of a decoder is less important if only the bitwise component of the BP loss is considered. This happens because such bitwise loss forces each output, individually, to be as close as possible to the target. This is observed by noting that all bitwise loss-based decoders flip less bits than the blockwise.

Table 3 summarizes the characteristics of each codebook used. Here d_{avg} denotes the average Hamming distance between any pair of codewords in the codebook. This quantity is a reference to the number of bits flipped by the decoder, except for the one-hot case. The reason is that the CE and, consequently, the BP loss applied to one-hot, are invariant to translations of the logits, so it is possible that most logits are positive, for example, and many bits are flipped back to their negative values.

Table 3: Characteristics of each codebook designed

	one-hot	random ($N = 75$)	random ($N = 100$)	degenerate
$d_{\min}$	2	22	34	1
d_{avg}	2	37	50	50

A.4 Further discussions on the bitwise and blockwise loss decomposition

As mentioned in Section 4, the L_{bit} is not uniquely defined. In fact, it is possible to just do

$$\begin{aligned} L &= L_{\text{bit}} + L_{\text{block}} \\ &= L_{\text{bit}} + \sum_{n=1}^{N} \alpha_n f(c_{yn}, z_n) + C + L_{\text{block}} - \sum_{n=1}^{N} \alpha_n f(c_{yn}, z_n) - C \\ &= L'_{\text{bit}} + L'_{\text{block}} \,, \end{aligned}$$

where

$$L'_{\text{bit}} = L_{\text{bit}} + \sum_{n=1}^{N} \alpha_n f(c_{yn}, z_n) + C \text{ and}$$

$$L'_{\text{block}} = L_{\text{block}} - \sum_{n=1}^{N} \alpha_n f(c_{yn}, z_n) - C.$$

That works since L'_{bit} still preserves the gradient property from equation 3. The decoding loss, on the other hand, is well defined. The important observation is that the property of dividing the problem into several subproblems is not required during training. In cases such as the BP loss (and CE loss as consequence), it naturally arises at decoding stage.

What we are interested, in particular, is how the blockwise term of the BP loss, as defined in equation 16, is beneficial to the overall accuracy of the model. According to that definition, the BP loss bitwise term for one hot is

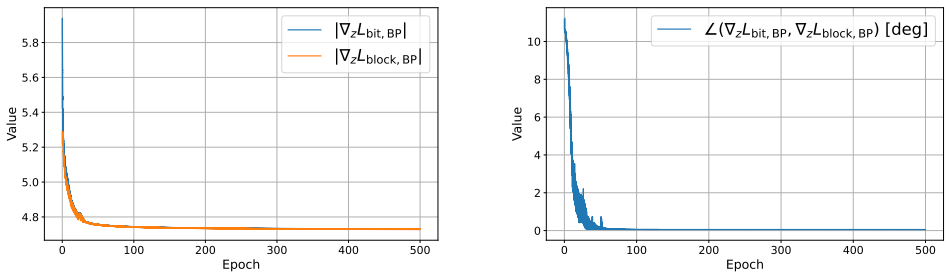
$$L_{\text{bit,BP,one-hot}} = N L_{\text{BCE,one-hot}} = -z_y + \sum_{n=1}^{N} \ln \sigma(-z_n) \,,$$

which is different than the simple $-z_y$, as defined in equation 10.

Table 4 summarizes the losses considered in this work from the perspective introduced in Sections 4 and 6.2.

Table 4: Bitwise, blockwise and decoding components of the losses considered in this work. For the training losses, the decomposition $L = L_{\text{bit}} + L_{\text{block}}$ follows equation 5

Loss	L_{bit}	L_{block}	Decoding loss
BP	$-\sum_{n=1}^{N} \ln \sigma(c_{yn} \cdot z_n)$	$\ln\left(\sum_{m=1}^{M} \rho_m\right)$	$-\sum_{n=1}^{N} \ln \sigma(c_{yn} \cdot z_n)$
BCE	$-\frac{1}{N}\sum_{n=1}^{N} \ln \sigma(c_{yn} \cdot z_n)$	0	$-\sum_{n=1}^{N} \ln \sigma(c_{mn} \cdot z_n)$
CE	$-z_y$	$\ln\left(\sum_{m=1}^{M} e^{z_m}\right)$	$-z_m$
MSE	$\frac{1}{N}\sum_{n=1}^{N}(c_{yn} - z_n)^2$	0	$\sum_{n=1}^{N}(c_{mn} - z_n)^2$
cos similarity	—	$(1 - \cos\theta_{\mathbf{z},\mathbf{c}_y})^2$	$(1 - \cos\theta_{\mathbf{z},\mathbf{c}_m})^2$



(a) Absolute values

(b) Angular displacement

Figure 3: Gradient behavior of the BP loss for the one-hot coding

A.5 Blockwise term contribution in the gradients

To better understand how the bitwise and blockwise terms influence the optimization process, we analyze the gradient of the loss with respect to the vector of logits $\mathbf{z} = [z_1, \ldots, z_N]^T$.

For a trainable parameter θ, the gradient descent update is

$$\theta^{(k+1)} = \theta^{(k)} - \eta \frac{\partial L}{\partial \theta}, \tag{29}$$

where η denotes the learning rate. Applying the chain rule yields

$$\frac{\partial L}{\partial \theta} = (\nabla_{\mathbf{z}} L)^T \frac{\partial \mathbf{z}}{\partial \theta}, \tag{30}$$

where

$$\nabla_{\mathbf{z}} L = \left[\frac{\partial L}{\partial z_1}, \frac{\partial L}{\partial z_2}, \ldots, \frac{\partial L}{\partial z_N} \right]^T. \tag{31}$$

Consequently, the quantity that directly controls how strongly the loss influences the parameter updates is the gradient vector $\nabla_{\mathbf{z}} L$. Large gradient magnitudes imply larger parameter updates and therefore a stronger optimization effort devoted to modifying the logits.

The corresponding gradient on the bitwise and blockwise components are then

$$\nabla_{\mathbf{z}} L = \nabla_{\mathbf{z}} L_{\text{bit}} + \nabla_{\mathbf{z}} L_{\text{block}}. \tag{32}$$

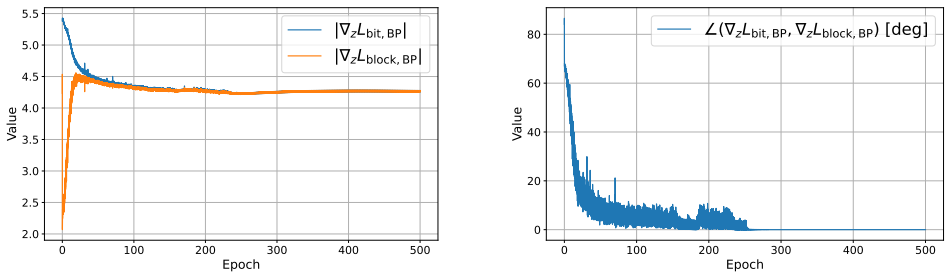
In order to quantify the relative contribution of each term, we consider the Euclidean norm of the gradient vectors,

$$|\nabla_{\mathbf{z}} L_{\text{bit}}| = \sqrt{\sum_{n=1}^{N} \left(\frac{\partial L_{\text{bit}}}{\partial z_n} \right)^2} \quad \text{and} \quad |\nabla_{\mathbf{z}} L_{\text{block}}| = \sqrt{\sum_{n=1}^{N} \left(\frac{\partial L_{\text{block}}}{\partial z_n} \right)^2}. \tag{33}$$

The gradient magnitude alone, however, does not fully characterize the interaction between the bitwise and blockwise terms. It is also important to analyze their relative orientation. For this purpose, we compute the angle between the two gradient vectors,

$$\angle(\nabla_z L_{\text{bit}}, \nabla_z L_{\text{block}}) = \arccos \left(\frac{(\nabla_{\mathbf{z}} L_{\text{bit}})^T \nabla_{\mathbf{z}} L_{\text{block}}}{|\nabla_{\mathbf{z}} L_{\text{bit}}| \cdot |\nabla_{\mathbf{z}} L_{\text{block}}|} \right). \tag{34}$$

Figures 3 and 4 show, for the CIFAR-100 dataset, the norms of the gradients $\nabla_{\mathbf{z}} L_{\text{bit}}$ and $\nabla_{\mathbf{z}} L_{\text{block}}$ together with the angle between them. Figure 3 refers to the one-hot coding while Figure 4 refers



(a) Absolute values

(b) Angular displacement

Figure 4: Gradient behavior of the BP loss for the degenerate codebook. Training over the CIFAR-100 dataset with ResNet34

to the degenerate codebook, both with the use of the BP loss and the ResNet34 model architecture. These plots provide insight into how the BP loss differs from the purely bitwise losse such as BCE. A first observation is that the blockwise gradient has a magnitude comparable to that of the bitwise gradient throughout most of the training process. If the blockwise term were negligible, removing it would not significantly alter the optimization trajectory. The angular displacement plots provide additional evidence of the importance of the blockwise component. If both gradients were always parallel, the blockwise term would merely rescale the update generated by the bitwise component and would not introduce any fundamentally new information into the optimization process. Instead, Figures 3b and 4b show that, particularly during the initial training stages, the angle between $\nabla_{\mathbf{z}} L_{\text{bit}}$ and $\nabla_{\mathbf{z}} L_{\text{block}}$ is nonzero.

The effect is particularly visible for the degenerate codebook shown in Figure 4. Since some pairs of codewords differ by only one bit, successful classification requires additional emphasis on the logits responsible for distinguishing nearby classes.

A.6 Extensions of the BP loss

The BP loss is a cross-entropy loss with respect to codewords and not individual bits. It follows then that the one-hot distribution still exists, but with respect to codewords. That can be further clarified if we consider a more generic case, in which the ground truth is not a deterministic label y but a random variable Y with distribution $p'_m = \text{Prob}\{Y = \text{class } m\}$. Then we can have

$$L_{\text{BP}} = -\sum_{m=1}^{M} p'_m \ln p_m \,. \tag{35}$$

The equation above shows the most generic form of the BP loss. It can potentially be used for any binary class representations applications.