Article

Deep Neural Networks Classification via Binary Error-Detecting Output Codes

Martin Klimo , Peter Lukáč and Peter Tarábek

Faculty of Management Sciences and Informatics, University of Žilina, 01026 Žilina, Slovakia;
peter.lukac@fri.uniza.sk (P.L.); peter.tarabek@fri.uniza.sk (P.T.)

* Correspondence: martin.klimo@fri.uniza.sk; Tel.: +421-907-185-782

Abstract: One-hot encoding is the prevalent method used in neural networks to represent multi-class categorical data. Its success stems from its ease of use and interpretability as a probability distribution when accompanied by a softmax activation function. However, one-hot encoding leads to very high dimensional vector representations when the categorical data's cardinality is high. The Hamming distance in one-hot encoding is equal to two from the coding theory perspective, which does not allow detection or error-correcting capabilities. Binary coding provides more possibilities for encoding categorical data into the output codes, which mitigates the limitations of the one-hot encoding mentioned above. We propose a novel method based on Zadeh fuzzy logic to train binary output codes holistically. We study linear block codes for their possibility of separating class information from the checksum part of the codeword, showing their ability not only to detect recognition errors by calculating non-zero syndrome, but also to evaluate the truth-value of the decision. Experimental results show that the proposed approach achieves similar results as one-hot encoding with a softmax function in terms of accuracy, reliability, and out-of-distribution performance. It suggests a good foundation for future applications, mainly classification tasks with a high number of classes.

Keywords: deep neural network; binary coding; error-detecting output code; multi-class classification; Zadeh fuzzy logic; one-hot encoding; output encoding; reliability; out-of-distribution

Citation: Klimo, M.; Lukáč, P.; Tarábek, P. Deep Neural Networks Classification via Binary Error-Detecting Output Codes. *Appl. Sci.* **2021**, *11*, 3563. <https://doi.org/10.3390/app11083563>

Academic Editor: Ugo Vaccaro

Received: 28 March 2021

Accepted: 12 April 2021

Published: 15 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Deep learning methods play a pivotal role in the latest breakthroughs in several tasks, outperforming the previous state-of-the-art approaches. Many tasks require solving a multi-class classification problem where an input space is mapped to an output space with more than two classes (patterns). The multi-class classification problem can be decomposed into several binary classification tasks. The most common approaches are one-versus-all (OVA), one-versus-one (OVO), and error-correcting output coding (ECOC) [1,2]. OVA reduces classifying K classes into K binary classifiers. The i -th binary classifier is trained using all the data of class i as positive examples and all the data from the other classes as negative examples [3]. An unknown example is classified to the class based on the classifier, which produces the maximal output. In the OVO approach, $K(K-1)/2$ binary classifiers are constructed [4]. Each ij -th binary classifier uses the data from class i as positive examples and the data from class j as negative examples. An unknown example is classified to the class based on the simple voting scheme [5] using the Decision Directed Acyclic Graph [6] or probabilistic coupling method [7]. In ECOC [8], coding matrix $M \in \{-1, 1\}$ is utilized. The coding matrix has a size of $K \times N$, where K is the number of classes (codewords), and N is the number of binary classifiers (bits of the codewords). Each row of M corresponds to one class (codeword), and each column of M is used to train a distinct binary classifier. When making inferences about an unknown example, the N binary classifiers' output is compared to the K codewords, usually using the Hamming distance. For example, a codeword with minimal distance is a class label. While OVA and OVO methods divide multi-class classification

problems into a fixed number of binary classification problems, ECOC allows each class to be encoded as an arbitrary number of binary classification problems.

Neural networks can be naturally extended to a multi-class problem using multiple binary neurons. The prevailing approach uses the one-hot (one-per-class) output encoding in combination with the softmax function. In one-hot encoding, each class is represented by a vector with a length equal to the number of categories. One-hot encoding vector for class i consists of all zeros except for the i -th component, which contains the value of 1. It is, therefore, an OVA classification scheme, although the output units share the same hidden layers [9]. The softmax function constrains all one-vs-all output nodes to sum to 1, making them interpretable as a probability distribution and helps training to converge faster.

However, one-hot encoding has several limitations. The number of output neurons in the neural network is equal to the number of classes. Each output neuron requires a unique set of weights which dramatically increases the number of parameters in models. It can be computationally infeasible as the number of classes increases. Furthermore, computing softmax probabilities can be very expensive due to the large sum in the normalizing constant of softmax when the number of classes is large, e.g., tens of thousands or millions [10]. The one-hot encoding schema also assumes a flat label space, ignoring the existence of rich relationships among labels that can be exploited during the training. In large-scale datasets, the data do not span the whole label space, but lie in a low-dimensional output manifold [11].

In this paper, we propose to use binary coding (i.e., coding where an arbitrary number of zeros and ones can be used) as the target coding instead of one-hot encoding. Although our work relates to ECOC [8] in the sense of using error-correcting output coding, it differs in two ways. First, we do not use the divide and conquer approach to partition the multi-class problem into an ensemble of multiple binary classification problems [8,12]. Instead, we train the whole codewords at the same time. Second, contrary to ECOC, we use error detection with a new ability to reject untrustworthy decisions. We propose an approach that replaces the combination of softmax and categorical cross-entropy to train models with binary coding combined with fuzzy Zadeh logic [13,14]. We use linear block codes from coding theory [15,16] as the output coding. Here, codewords are divided into two parts: information bits and check bits. The information part can reflect a data structure while the check part distributes the codewords across the codespace to obtain the error detection performance.

In addition to comparing the accuracy of the proposed method against traditional one-hot output encoding with a softmax activation function, we study the trained models' reliability. Trust and safety is an important topic in machine learning [17,18]. In the case of a neural network, a standard approach to deciding whether or not to trust a classifier's decision is to use the classifiers' reported confidence scores, e.g., probabilities from the softmax layer [19]. We define reliability as a percentage of correctly classified samples among all samples for which the model is confident to make a prediction. It means that we have three types of results: correct, incorrect, and abstain (reject). The typical approach to improve reliability is to use a threshold value on the classifier's confidence score. It allows the classifier to abstain from predicting if the resulting confidence is lower than the predefined threshold. This way, a classification network can indicate when it is likely to be incorrect. Modern deep neural networks are poorly calibrated, i.e., the probability associated with the predicted class label does not reflect its ground truth correctness likelihood [20,21]. It can be improved using calibration methods, such as temperature scaling [20,21], but even if the neural network is calibrated, the ranking of scores may not be reliable [22].

The error detection ability of binary output codes provides an additional tool to improve reliability—a threshold on the Hamming distance between the predicted output code and codewords assigned to classes. For one-hot codes, the Hamming distance between all codes is equal to two, which does not allow correcting errors or improving reliability. We tested the reliability on in-distribution and out-of-distribution data (data far away from the

training samples). We show that our proposed method for both in-distribution and out-of-distribution data achieves approximately the same results in a case of maximum accuracy as the traditional one-hot encoding with softmax when used on a 10-class classification problem. It opens up new possibilities and research directions, especially regarding how to find optimal binary codes (in terms of their size and inner structure) and architectures that use their full potential. We take advantage of the error detection ability to indicate a codeword outside an agreed code subspace without evaluating a confidence metric. In the case of out-of-distribution data, we do not use any advanced techniques to tackle this problem [23,24] as we are interested in the natural properties of the proposed method. We leave this to future research.

The Contribution of This Work Is the Following:

- We study the use of error-detecting linear block codes as a binary output coding (e.g., Cyclic Redundancy Check—CRC).
- We propose a novel method based on the Zadeh fuzzy logic, which enables us to train whole binary output codes at once, and to use error detection ability. We show that the proposed method achieves similar accuracy as one-hot encoding with a softmax function trained using categorical cross-entropy.
- We prove that the error detection in pattern recognition could be achieved directly by rounding the normalized network output and calculating the zero-syndrome. The result is most trustworthy from the Zadeh fuzzy logic perspective, and the truth-value can be given to the user.
- We evaluate the proposed system against other approaches to train binary codes.
- We perform further study to compare the proposed method's performance against the one-hot encoding approach in terms of reliability and out-of-distribution accuracy.

The rest of the paper is organized as follows. In Section 2, we describe our proposed method and compare it to the existing approaches. We also describe the used dataset, metrics, and protocol for training neural network models. Section 3 show the results of in-distribution and out-of-distribution experiments. Finally, the conclusion is drawn in Section 4.

2. Methods and Materials

Pattern recognition is a complex problem that should be decomposed into a chain of more straightforward tasks. In this paper, the chain includes neural network design, output coding, and the classification method. Our paper contributes to the output coding by applying binary error-detecting codes (Section 2.1) and to the classification by proposing an approach based on Zadeh fuzzy logic (Section 2.2). We evaluate the proposed method from a broader perspective of reliability and out-of-distribution data. Used datasets are described in Section 2.3 and metrics, in Section 2.4. The network training protocol and architectures are described in Section 2.5.

2.1. Output Coding

Categorical data at the neural network outputs can divide the space in different ways. Nowadays, most of the classifiers use the one-hot (one-per-class) output coding. Applying the softmax layer, the winning class gives the maximum output, and the output space is divided by hyperplanes into class subspaces. One-hot codewords create an orthogonal standard codespace basis. Bits of the codewords are not independent because giving a value to the one bit changes the possibility of the rest of the bits' values in the codeword. This property provides an advantage to such a holistic approach embodied into one-hot encoding by softmax. The disadvantage of one-hot encoding is a small pairwise Hamming distance equal to two bits.

Another approach is to use a nonstandard orthogonal codespace basis. One possibility is given by Hadamard vectors [25]. The Hadamard matrix provides the following codewords:

$$H_{i+1} = \begin{pmatrix} H_i & H_i \\ H_i & -H_i \end{pmatrix}, H_0 = (1), \quad (1)$$

where the value “−1” is replaced by “0”. Except for the first matrix row, in which all elements are equal to one, other rows contain bits of “1” in the half of bits and “0” in the rest of the codeword. A helpful feature of the Hadamard matrix is an equidistant distribution of the codewords with $2n - 1$ bits Hamming distance, where $2n$ is the number of neural network outputs. Let us notice that number of classes has to be less than or equal to the number of network outputs. Further, ref [26] also argues for the same number of “0” and “1” in the matrix columns (except the first one). They try to preserve this property by using fewer codewords than giving the full codespace (the first bit in all matrix rows is excluded from the codeword).

Binary coding can give a bigger output capacity. While outputs in one-hot or Hadamard coding are limited to basic vectors, binary coding outputs can include all vectors generated by this basis. While the one-hot encoding is vulnerable to adversarial samples [27], the binary coding, e.g., multi-way coding [28], is more resistant.

Assigning codewords to the categories in binary coding is not as direct as in one-hot encoding. A well-studied example is the Error Correcting Output Code (ECOC) introduced in [8]. Recognition of each bit of the output binary code (given by a code matrix) is trained separately on the divided binary pattern or feature training set. The aim is to have the bits independent. Reaching a suitable code matrix remains the main challenge [29–32]. We can interpret the ECOC codeword as an ensemble of independent bits where each bit indicates the appearance of a corresponding feature in the pattern. Our paper focuses on multi-class patterns, where the label is assigned to the whole binary codeword. The dependency of bits allows using inter-bit information in a way similar to softmax.

The main advantage of the binary codes is the excellent knowledge acquired in the coding theory. We use linear block codes for their possibility of the separation class information from the checksum part of the codeword. To compare one-hot versus binary encoding at the 10-class database CIFAR-10, we choose 10-bit binary codes. The essential decision on codeword acceptance is based on syndrome calculation. The error is detected [15] if the syndrome is a non-zero vector or the zero vector (16 codewords), but the codeword is not assigned to any class (e.g., 6 of the 16 codewords). When the rules detect the codeword as an error, we consider the sample untrustworthy, and we do not assign any class to it (we abstain from classification). We use (10, 4) block systematic Cyclic Redundancy Check (CRC) codes as an example. Finding an optimal code from the classification accuracy point of view remains a challenge for the future as the neural network errors are correlated. Table 1 gives generating polynomials [15] of these codes. All of them are irreducible, and the first five are also primitive.

Table 1. The binary form of the generating polynomials of used Cyclic Redundancy Checks (CRCs).

CRC1: 100,0011	CRC2: 101,1011	CRC3: 110,0001	CRC4: 110,0111	CRC5: 111,0011
CRC6: 100,1001	CRC7: 101,0111	CRC8: 110,1101	CRC9: 111,0101	

2.2. Decision Method

How can each training pattern be connected with a suitable class? While the neural network itself implements a nonlinear mapping $\mathcal{R}^m \rightarrow \mathcal{R}^n$, $m > n$, the classification’s goal is to assign a class from the finite set (expressed by the codeword in our case) to each input sample. The essential question to be answered is, which codeword is the most similar to the obtained output? Even though the codeword is binary, we can assume it as a point in the Euclid space. We can find the closest codeword to the network output vector regarding a given distance (e.g., Euclidian). This distance-based approach is sufficient if the most

similar codeword finding is a final step in the classification chain. However, the output assignment to the codeword is a logical statement, and thus we can apply logical rules to find new properties. We can even interpret binary n -tuples as vectors that, together with Galois-field based scalars, create a finite vector space and derive additional properties using the coding theory. This rich source of logical and algebraic tools is available only after assigning neural network outputs to the codewords. Unfortunately, after this flip from the continuous space into the binary space, we lose all similarity information. To tie the benefits of both worlds, we look at the result of an algebraic operation as a statement, and we evaluate the result by the truth value, applying the fuzzy logic. For example, this approach will assess the syndromes' truth values and prove that the truest codeword gives the truest syndrome. As explained later, we decided to use the Zadeh (standard) fuzzy logic and $t \in [0, 1]$ as the truth-values.

Therefore, the first step is the normalization of the individual neural network outputs. Let the response of the neural network to the given sample be $\mathbf{y} = (y_0, \dots, y_{n-1}) \in \mathcal{R}^n$, and $\tilde{\mathbf{y}} = (\tilde{y}_0, \dots, \tilde{y}_{n-1}) \in [0, 1]^n$ after normalization, where n is the number of output bits ($n = 10$ in our case). We use three types of normalization for $i = 0, \dots, n - 1$:

$$\text{sigmoid } \tilde{y}_i = \sigma(y_i) = \frac{1}{1 + e^{-y_i}} \quad (2)$$

$$\text{linear } \tilde{y}_i = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}}, \quad y_{\min} = \min\{y_0, \dots, y_{n-1}\}, y_{\max} = \max\{y_0, \dots, y_{n-1}\} \quad (3)$$

$$\text{vector } \tilde{\mathbf{y}} = \frac{\mathbf{y}}{\|\mathbf{y}\|_2} \quad (4)$$

After normalization, we have to map the individual outputs to a codeword. The simplest way (that is an extension of the one-dimensional task) is to decide for each output independently.

$$c_i = \begin{cases} 0, & \tilde{y}_i < 0.5 \\ 1, & \tilde{y}_i \geq 0.5 \end{cases}, i = 0, 1, \dots, n - 1. \quad (5)$$

We call this method a "bit threshold method". From the other perspective, we can interpret the output value as a truth value of the fuzzy statement that the output gives as a specific bit (see Equation (9) for details). The more complex approaches take the output vector $\tilde{\mathbf{y}}$ as a whole; the most popular is the softmax method. This approach is based on one-hot encoding, where the bit dependence is based on the fact that all bits are equal to zero except one that is equal to one. After an exponential transformation,

$$\tilde{y}_i \leftarrow \frac{e^{\tilde{y}_i}}{\sum_{j=0}^{n-1} e^{\tilde{y}_j}}. \quad (6)$$

applying a probability distribution scale, the expected value one is assigned to the maximum value and the expected zeros, to the others.

In [11], a method which combines ECOC with softmax was proposed. This method does not use error detection. We implement this method using CRC coding instead of ECOC to compare it with the proposed Zadeh decision method.

To explain how general binary codes are used, let us start with two-valued logic. In a Boolean case, let the output $\tilde{\mathbf{y}}$ be mapped into the codeword $\tilde{\mathbf{c}} = (\tilde{c}_0, \dots, \tilde{c}_{n-1}) \in \{0, 1\}^n$, and each class N be associated with a given codeword $\mathbf{c}^N = (c_0^N, \dots, c_{n-1}^N)$, $N \in \{0, 1, \dots, n - 1\}$. The training pattern will be classified in the class N , if $\tilde{\mathbf{c}} = \mathbf{c}^N$, i.e., $\Lambda_{i=0}^{n-1}(\tilde{c}_i = c_i^N)$. In the Boolean logic, the codeword is equal to the class N codeword, if the truth-values of equality in all codeword's bits are equal to one, i.e.,

$$S^N = \Lambda_{i=0}^{n-1}(\tilde{c}_i = c_i^N) \Rightarrow t(S^N) = \prod_{i=0}^{n-1} t(\tilde{c}_i = c_i^N) = 1, N \in \{0, 1, \dots, 2^n - 1\} \quad (7)$$

where $S^N = (\tilde{c} = c^N)$, $\tilde{c}, c^N \in \{0, 1\}^n$, $t(true) = 1$, $t(false) = 0$.

To keep this holistic approach to the codeword decision, we replace the Boolean logic with Zadeh (standard) fuzzy logic and $t \in [0, 1]$ as a truth-value. The true values of the Zadeh fuzzy logic conjunction, disjunction, and negation of the A, B statements are

$$\begin{aligned} t(A \wedge B) &= \min(t(A), t(B)), \\ t(A \vee B) &= \max(t(A), t(B)), \\ t(\overline{A}) &= 1 - t(A). \end{aligned} \quad (8)$$

The i -th output value $\tilde{y}_i \in [0, 1]$ can be interpreted as a truth-value. To identify whether the bit is correct independent of its bit value, we introduce the degree of compliance as a truth-value [33] of the statement $\tilde{y}_i = a$, $a \in \{0, 1\}$. To avoid indeterminate cases (if $\tilde{y}_i = 1/2$), we add or subtract a negligible number $\varepsilon > 0$ to the truth-value, and therefore, $t(\tilde{y}_i = a) \neq 1/2$

$$t(\tilde{y}_i = a) = \begin{cases} \tilde{y}_i, & a = 1, \tilde{y}_i \in [0, 1/2) \cup (1/2, 1] \\ 1 - \tilde{y}_i, & a = 0, \tilde{y}_i \in [0, 1/2) \cup (1/2, 1] \\ 1/2 + \varepsilon, & a = 1, \tilde{y}_i = 1/2, 0 < \varepsilon \ll 1 \\ 1/2 - \varepsilon, & a = 0, \tilde{y}_i = 1/2, 0 < \varepsilon \ll 1 \end{cases} \quad (9)$$

We exclude this value in the implementation because the probability of $\tilde{y}_i = 1/2$ on the unit interval is equal to zero. Then,

$$t(S^N) = t(\tilde{c} = c^N) = \min_{i=0, \dots, n-1} t(\tilde{c}_i = c_i^N), \quad N \in \{0, 1, \dots, 2^n - 1\}. \quad (10)$$

The notation (10) is valid also for Boolean logic because

$$\min_{i=0, \dots, n-1} t(\tilde{c}_i = c_i^N) = \prod_{i=0}^{n-1} t(\tilde{c}_i = c_i^N), \quad \text{for } t(\tilde{c}_i = c_i^N) \in \{0, 1\}.$$

Therefore, in Figure 1, the minimum function is used instead of multiplication, even in the Boolean case.

Figure 1. Proposed classification scheme based on the Zadeh fuzzy logic.

The demand for processing analog channel outputs by digital coding is not new [34]. Our approach is based on fuzzy linear coding [35,36]. In general, a fuzzy codeword is a

subset of the codespace with assigned membership functions. To recognize the pattern of each sample, we evaluate (10), and the function (11)

$$m = \operatorname{argmax}_{c^N \in C} \{t(\tilde{y} = c^N)\} \quad (11)$$

gives the index of the codeword with the highest fuzzy truth-value. The output $\tilde{y}$ and, consequently, sample x , will be assigned to the codeword c^m . We use 10 codewords for training and 1024 codewords for testing to compare one-hot encoding with binary coding. If the codeword c^m is outside of the set of the codewords assigned to the classes, we interpret the case as error detection, and the pattern is not classified (classifier abstains from classification). In this paper, we are using a zero syndrome for error detection of binary codes on $GF(2)^n$ [15]. Zadeh fuzzy logic has a valuable property, defined in Theorem 1, for testing over the complete set of codewords.

Theorem 1. Let $\{c^j = (c_0^j, \dots, c_{n-1}^j) \in \{0, 1\}^n, j \in \{0, 1, \dots, 2^n - 1\}$ be the binary codespace, $\tilde{y} = (\tilde{y}_0, \dots, \tilde{y}_{n-1}) \in [0, 1]^n$ be a normalized output of the neural network, $t(\tilde{y}_i = c_i^j)$ be defined according to (9), and the winning codeword $\tilde{c}$ be taken according to Equation (11).

- (a) Then the truth-value that $\tilde{c}$ is the winning codeword is more than one half.
- (b) There is only one codeword with truth-value higher than one half.
- (c) The winning codeword is obtained by rounding off $\tilde{y}$ i.e., $\tilde{c} = (\approx \tilde{y}_0, \dots, \approx \tilde{y}_{n-1})$, where $\approx \tilde{y}_i = \lfloor \tilde{y}_i + 1/2 \rfloor$.

Proof of Theorem 1. Assume $c^j = \operatorname{argmax}_{c \in \{0,1\}^n} (\mu(\tilde{y} = c))$ and $t(\tilde{y} = c^j) < 1/2$. Then, there exists in the codeword c^j bits $i \in I \subseteq \{0, \dots, n-1\}$ with $t(\tilde{y}_i = c_i^j) < 1/2$. Replacing these bits by their negations, we obtain the codeword c^k with $t(\tilde{y}_i = c_i^k) > 1/2, i = \{0, \dots, n-1\}$. Therefore, $t(\tilde{y} = c^k) > 1/2$, which is in contradiction with the assumption.

- (a) Assume $c^j = \operatorname{argmax}_{c \in \{0,1\}^n} (t(\tilde{y} = c)), c^k = \operatorname{argmax}_{c \in \{0,1\}^n} (t(\tilde{y} = c)) j \neq k$ and $t(\tilde{y} = c^j) > 1/2, t(\tilde{y} = c^k) > 1/2$. Because $c^j \neq c^k$, there exists a bit $i \in \{0, 1, \dots, n-1\}$ that $c_i^j = \overline{c_i^k}$. $t(\tilde{y}_i = c_i^j) = t(\tilde{y}_i = \overline{c_i^k}) > 1/2$ implies $t(\tilde{y}_i = c_i^k) < 1/2, t(\tilde{y} = c^k) < 1/2$, which is in contradiction with the assumption.
- (b) For the rounded output $t(\tilde{y} = (\approx \tilde{y})) > 1/2$. According to b), $\approx \tilde{y}$ is the only winning codeword. $\square$

The idea of the syndrome truth-values is based on truth-value calculations of the algebraic expressions given by their crisp result and fuzzy truth-values. Binary operations modulo 2 are calculated as follows:

$$a \odot b, a, b \in \{0, 1\} \quad a \oplus b, a, b \in \{0, 1\}$$

$\odot$	0	1
0	0	0
1	0	1

$\oplus$	0	1
0	0	1
1	1	0

(12)

According to Equation (8), for $a, b \in [0, 1]$

$$\begin{aligned} t(a \odot b = 1) &= t((a = 1) \wedge (b = 1)), \\ t(a \odot b = 0) &= 1 - t(a \odot b = 1), \\ t(a \oplus b = 1) &= t((a = 0) \wedge (b = 1) \vee (a = 1) \wedge (b = 0)), \\ t(a \oplus b = 0) &= 1 - t(a \oplus b = 1), \end{aligned} \quad (13)$$

We apply Equation (13) also for $t(\tilde{y}_j \odot h_{ij}^T)$, $\tilde{y}_j \in [0, 1]$, $h_{ij}^T \in \{0, 1\}$, where h_{ij}^T is a crisp value in the check matrix H [15]. If the normalized network output $\tilde{y}$ is the truth-value of the codeword $\tilde{c}$ i.e., $\tilde{y} = t(\tilde{c} = c)$ or $\tilde{y}_j = t(\tilde{c}_i = c_i)$, $i = (0, \dots, n-1)$, then the syndrome estimation $\tilde{s}$ is $\tilde{s} = \tilde{c}H^T$, where $H = (P|I)$, and I is a unite matrix for the tested code CRC7

$$P^T = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Applying Equations (9) and (13), the truth-value of the zero syndrome (no error detected) is

$$\begin{aligned} t(\tilde{s}_i = 0) &= t\left(\sum_{\oplus j} \tilde{y}_j \odot h_{ij}^T = 0\right), \\ t(\tilde{s}_i = 0) &= \max_{j=0, \dots, n-1} \left\{ \min \left\{ t(\tilde{y}_j = c_j), h_{ij}^T \right\} \right\} \end{aligned} \quad (14)$$

Note that for the following theorem $t(\tilde{y}_i \odot a) \neq 1/2$ and (n, k) linear block code means codewords with n bits and k information (class) bits.

Theorem 2. Let $\{c^j = (c_0^j, \dots, c_{n-1}^j) \in \{0, 1\}^n\}$, $j \in \{0, 1, \dots, 2^n - 1\}$ be the binary codespace, $\tilde{y} = (\tilde{y}_0, \dots, \tilde{y}_{n-1}) \in [0, 1]^n$ be a normalized output of the neural network, $t(\tilde{y}_i = c_i^j)$ be defined according to Equation (9), then the winning codeword $\tilde{c}$ is taken as

$$\tilde{c} = \underset{\tilde{c} \in \{0, 1\}^n}{\operatorname{argmax}} \{t(\tilde{y} = \tilde{c})\},$$

the code syndrome true values $\mathbf{r} = (r_0, \dots, r_{n-k-1}) \in [0, 1]^{n-k}$ are calculated as $\mathbf{r} = \tilde{y}H^T$, where H^T is the check matrix of a given error-detecting linear (n, k) block code, and the winning syndrome is

$$\tilde{\mathbf{s}} = \underset{\tilde{\mathbf{s}} \in \{0, 1\}^{n-k}}{\operatorname{argmax}} \{t(\mathbf{r} = \tilde{\mathbf{s}})\}.$$

Then $\tilde{c}H^T = \mathbf{0} \Leftrightarrow \tilde{\mathbf{s}} = \mathbf{0}$, i.e., the statement “winning codeword is correct” is equal to the statement “the winning syndrome is equal to zero”.

Proof of Theorem 2. According to Theorem 1, for any $\tilde{y}$ there exists a unique winner codeword $\tilde{c}$ with $t(\tilde{y} = \tilde{c}) > 1/2$, i.e., $t(\tilde{y}_i = \tilde{c}_i) > 1/2$, $i = 0, \dots, n-1$ and for any $\mathbf{r} = \tilde{y}H^T$ there exists a unique winner syndrome $\tilde{\mathbf{s}}$ with $t(\mathbf{r} = \tilde{\mathbf{s}}) > 1/2$ i.e., $t(r_i = \tilde{s}_i) > 1/2$, $i = 0, \dots, n-k-1$. The winner codeword $\tilde{c}$ is correct if $\tilde{c}H^T = \mathbf{0}$, the winner syndrome is zero if $t(\mathbf{r} = \mathbf{0}) > 1/2$. According to Equation (14), $t(\mathbf{r} = \mathbf{0}) = t(\tilde{y}H^T = \mathbf{0})$. Due to the identity submatrix within the check matrix, each column in the check matrix has at least one “1”. Then,

$$t(\tilde{y}_j = c_j) > 1/2 \quad j = 0, \dots, n-1 \Rightarrow t(r_i = 0) = \max_{j=0, \dots, n-1} \left\{ \min \left\{ t(\tilde{y}_j = c_j), h_{ij}^T \right\} \right\} > 1/2$$

$$i = 0, \dots, n-k-1$$

The true value of the zero syndrome is $t(\mathbf{r} = \mathbf{0}) = \min_{i=0, \dots, n-k-1} t(r_i = 0)$. If $t(r_i = 0) > 1/2$ then $t(r_i = 1) < 1/2$, $i = 0, \dots, n-k-1$, and changing any bit in the syndrome will decrease the truth-value of the syndrome.

The complement gives the truth-value of the detected error $t(\mathbf{r} \neq \mathbf{0}) = 1 - t(\mathbf{r} = \mathbf{0})$. Theorem 2 can be easily generalized for error-correcting codes: if the winning codeword has the syndrome $\tilde{\mathbf{s}} = \tilde{c}H^T$, then $\tilde{\mathbf{s}}$ is the winning syndrome. $\square$

$$\tilde{\mathbf{s}} = \underset{\tilde{\mathbf{s}} \in \{0, 1\}^{n-k}}{\operatorname{argmax}} \left\{ t(\mathbf{r} = \tilde{\mathbf{s}}) \right\}, \text{ where } \mathbf{r} = \tilde{y}H^T$$

The direct calculation of the syndrome truth-value is suitable for classification tasks with many classes. In the one-hot encoding/softmax approach, we need network output for each class, while with binary coding, the minimum number of outcomes required to encode n classes will not exceed $\lceil \log_2(n) + 1 \rceil$. This paper tested a fuzzy Zadeh decision on top of the binary codes against the softmax decision in one-hot encoding. To directly compare both methods, we use the same number of outputs (one output per class), and Zadeh fuzzy logic is applied to find the truest class.

To compare softmax with the Zadeh fuzzy logic, we also apply a threshold Δ to accept the decision. In the softmax case, a threshold can be used on the confidence value. In the Zadeh decision, according to Equation (11), we can also use the threshold to accept the selected codeword only if $t(\mathbf{r} = \mathbf{0}) \geq \Delta$, $\Delta \in [0, 1]$, similar to the softmax threshold.

We have also tested the product t-norm, and the probabilistic t-conorm for strong conjunction and disjunction in the fuzzy logic we applied [37]. It seems that it is not suitable for gradient backpropagation. Conversely, in the Zadeh fuzzy logic, the backpropagation is straightforward. The worst neural network output in the best codeword contributes to learning the most (a Zadeh approach from the error viewpoint).

2.3. Dataset

Our goal was to test the performance of CRC coding and the proposed Zadeh classification approach, not to find a state-of-the-art-solution for the classification problem. Our experiments were performed on the CIFAR-10 [38] dataset. Its relative simplicity, small resolution, ten-class classification problem, and usage in many papers as a baseline dataset make it the right choice. For the out-of-distribution statistics, we used four datasets: MNIST [39], Fashion MNIST [40], CIFAR-100 [41] and white noise (independent pixels with uniformly distributed values). Testing and early stopping during the training were performed on the test part of the datasets. We assigned the codewords to the classes, regardless of the class meaning. We used data augmentation with translation 15%, rotation 10°, zoom 10%, and horizontal flipping in our experiments.

2.4. Metrics

We use two primary measures, accuracy and reliability, to compare the performance of different output codings and decision methods. Accuracy, Equation (15), is a standard performance metric used to report results on classification tasks. We are also interested in the reliability of classification that is defined using Equation (16).

We have two primary system states: accepted and rejected. When a decision about an unknown sample is made, it is first either accepted or rejected. When the sample is accepted, the decision about its class is made. It is either positive (true class label) or negative (false class label). Rejected means that the classifier abstains from classification.

For this reason, we have six possible fundamental states formulated in the confusion matrix shown in Table 2. Note that there are only two states: false positive and true rejected, in the case of out-of-distribution data. For this reason, we use rejection (19) to measure out-of-distribution performance.

$$\text{Accuracy } p_{acc} = \frac{TP + TN}{Q} \quad (15)$$

$$\text{Reliability } p_{conf} = \frac{TP + TN}{P + N} \quad (16)$$

$$\text{Precision } p_{conf} = \frac{TP}{P} \quad (17)$$

$$\text{Acceptance } p_A = \frac{P + N}{Q} \quad (18)$$

$$\text{Rejection } p_R = \frac{R}{Q} \quad (19)$$

$$p_{acc} = p_{conf} p_A = p_{conf} (1 - p_R) \quad (20)$$

$$\frac{TP + TN}{Q} = \frac{TP + TN}{P + N} * \frac{P + N}{Q} \quad (21)$$

Table 2. Confusion matrix.

	Accepted (A)		Rejected (R)	Σ
	Positive (P)	Negative (N)		
True (T)	TP	FN	FR	T
False (F)	FP	TN	TR	F
Σ	P	N	R	Q

2.5. Neural Network Architectures and Training Protocol

We selected VGG style [42] convolution neural network with batch normalization and dropout, called CNN1 and CNN2 (shown in Table 3). The difference between these two architectures is in the number of used feature maps. In both CNN1 and CNN2, we used the ELU activation unit. For comparison, we also used the ResNetV2 [43] architecture with depth 20.

Table 3. CNN1 and CNN2 neural network architectures.

Layer	Kernel Size	Kernel Stride	CNN-1	CNN-2
			Feature Maps	Feature Maps
Conv + BN	3	1	32	64
Conv + BN	3	1	32	64
Conv + BN	3	1	32	64
Conv + BN	3	1	32	64
Conv + BN	3	1	32	128
MaxP	2	1	32	128
Dropout 0.2				
Conv + BN	3	1	64	128
Conv + BN	3	1	64	256
MaxP	2	2	64	256
Dropout (0.2)				
Conv + BN	3	1	128	256
Conv + BN	3	1	128	256
MaxP	2	2	128	256
Dense	1	1	10	10
Normalization		Softmax/Sigmoid/Linear		

The number of training parameters in CNN1 is 328K, in CNN2 1837K, and ResNet20v2, 574K. In our experiments, we used several types of normalizations on the output layer. Softmax normalization with categorical cross-entropy is used as a baseline. For the fuzzy Zadeh decision layer, linear (Equation (3)) or sigmoid (Equation (2)) normalizations were applied.

Equation (11) gives the predicted codeword c^m . The loss function for the Zadeh decision is as follows:

$$L(x) = 1 - \max \left\{ t(S^0), \dots, t(S^{n-1}) \right\} \quad (22)$$

The gradient calculated from this error is applied to the worst bit b of the predicted codeword.

$$b = \underset{i=0, \dots, n-1}{\operatorname{argmin}} t(\tilde{y}_i = c_i) \quad (23)$$

In softmax experiments, the categorical cross-entropy loss function was used. The training procedure for all experiments utilized Adam [44] optimizer with an initial learning

rate of 0.001. The learning rate was decreased by a factor of 0.998 until 5×10^{-5} was reached, starting with epoch 250. We used L2 weight regularization with weight decay 1×10^{-4} and mini-batch size 512. We fully trained all models with the following stopping criteria: training is stopped if the testing set results are not improved for 400 epochs or if epoch 6000 is reached. We used weight initialization according to [45] for all networks. In order to train and implement our models, we used Python implementation of TensorFlow [46] and Keras [47] frameworks. For more implementation details, see the reference in Supplementary Materials at the end of the paper.

3. Results and Discussion

This section describes classification accuracy (including out-of-distribution performance) and reliability in various network architecture configurations, output coding, and classification decisions. We were not interested in finding the optimal network architecture for the proposed binary codes. Instead, we used standard architectures optimized for one-hot encoding with the softmax classification described in Section 2.5.

3.1. Effect of CRC Output Coding

First, we experimented with different linear block codes represented by CRC codes (see Table 1). The results for CNN1 architecture with linear normalization of outputs and the Zadeh decision are reported in the accuracy–reliability characteristic in Figure 2. The accuracy is computed according to Equation (12). To measure reliability (Equation (13)), we use the previously mentioned error detection ability of binary output codes. Here, we use the whole codeword space, exploiting 1024 codewords instead of 10. If the output code is outside of the 10 codewords assigned to the classes, the sample is rejected as untrustworthy. One point in the graph shows accuracy vs. reliability for one threshold value applied during the classification decision. Different points can be obtained using different threshold values in the decision process described in Section 2.2. If no samples are rejected as untrustworthy, the accuracy is equal to the reliability. This case corresponds to the dashed line in Figure 2. As the CRC codes have the detection ability to reject untrustworthy patterns, the reliability outperforms the accuracy even with a confidence threshold equal to zero. In all performed experiments, the classification reliability can be improved by trading off reliability for accuracy using higher thresholds.

We run three experiments for each CRC output code (CRC1-9) and report the accuracy vs. reliability characteristic for the highest maximal accuracy. We can see the dependency of the results on the output coding, which suggests further possibilities for finding better output coding. However, finding the best binary output coding is not our goal, and for the rest of the paper, we use the winning CRC7 as a representative of CRC codes. CRC7 will detect all errors e with non-zero syndrome $eH^T \neq 0$, where H^T is the check matrix [15].

3.2. Performance on CIFAR-10

This section describes our main experiments. We evaluated various configurations of network architecture, output coding, and classification decision in terms of classification accuracy and reliability. Experiments were performed on the CIFAR-10 dataset. Accuracy is the most used measure of classification performance. In the overall methodology, the classification procedure assigns each neural network output to the one class (i.e., to the one codeword) during the training and the testing. Introducing the error-detecting codes, we can extend the decision space to the whole codespace. In this paper, we keep assigning network output to one codeword during the training. During the testing, the network output can be assigned to any code from the whole codespace. We use 10 bits output codes for all combinations reported in Table 4, except Hadamard coding (described in Section 2.1), which requires a minimum of 15 bits to cover 10 classes and to ensure the properties described in [26]. It means that the whole codespace is 1024 for CRC7 and 10 for one-hot encoding. We use accuracy to measure performance when only class-codewords (output codes belonging to classes) are used. When we use the whole codespace during the

decision, we can end up with output codes that do not belong to any class. In this case, the error detection capability is used, and the sample is rejected as untrustworthy. We use the term “full codespace accuracy” (FCA) to measure accuracy when full codespace is used.

Figure 2. Accuracy–reliability characteristic for CRC codes (left), CRC7 class-codewords (right).

Although the paper’s primary goal is to examine the properties of binary coding, the decision-making method greatly influences the correctness of the classification and the ability to use full codespace. As shown in Table 4, we compared the performance of bit-by-bit decision (represented by the “bit threshold” approach) and holistic word decisions represented by Euclid distance (applied on Hadamard code), softmax (used on one-hot code), as well as a proposed Zadeh decision (applied for both one-hot and CRC code). We first tested all approaches with CNN1 architecture. For each combination, we ran 10 repetitions of training and reported the result with the best accuracy. Then, we selected four best performing approaches, according to their accuracy, and tested them on CNN2 and ResNet20 architecture. Here, we ran five repetitions of training for each architecture and reported the best result according to best accuracy.

Our primary interest is to examine the possibility of using binary coding (represented by CRC7) with error detection capability. We first tested the bit thresholding approach (Equation (5)) as it is a straightforward decision-making approach. For this approach, the mean squared error was used during the training and bit thresholding during the testing. Then we tested Zadeh decision on both CRC7 and one-hot encoding. Using the Zadeh decision, we approached one-hot encoding as a particular binary coding case where class-codewords have one-hot properties. We used one-hot encoding with softmax as a baseline method for comparison. Inspired by [11], we also tested CRC7 coding with softmax. It allowed to combine the advantages of binary codes and the softmax decision but did not use error detection capabilities. Last, we tried the approach from [26], where the Euclid decision is used on Hadamard coding.

Table 4. Performance of the various output coding and decision strategies.

Network	Code	Decision	Accuracy	FCA	Undetectable Error	Reliability
CNN1	CRC7	Zadeh	92.44	92.01	6.77	93.15
CNN1	one-hot	Zadeh	92.01	91.56	7.29	92.63
CNN1	one-hot	softmax	92.15	92.15	7.85	92.15
CNN1	CRC7	softmax	91.85	91.85	8.15	91.85
CNN1	CRC7	bit threshold	89.56	89.56	6.8	92.94
CNN1	Hadamard	Euclid	89.92	88.6	8.2	91.53
CNN2	CRC7	Zadeh	93.97	93.47	5.44	94.5
CNN2	one-hot	Zadeh	93.4	92.66	5.76	94.14
CNN2	one-hot	softmax	94.07	94.07	5.93	94.07
CNN2	CRC7	softmax	93.88	93.88	6.12	93.88
ResNet20v2	CRC7	Zadeh	91.56	91.27	7.97	91.97
ResNet20v2	one-hot	Zadeh	84.16	83.28	14.5	85.17
ResNet20v2	one-hot	softmax	91.67	91.67	8.33	91.67
ResNet20v2	CRC7	softmax	91.34	91.34	8.66	91.34

The results indicate a predominance of the holistic style of decision (Zadeh, softmax, Euclid) over decisions on individual bits (bit threshold), and we expect increasing advantages with an increasing number of bits. We can see that CRC7 with Zadeh decision achieves similar accuracy to one-hot encoding with softmax. In one-hot encoding with softmax, the code structure and the decision rule support each other. We know that only one is equal to “1” within bits, and the output values are normed and evaluated to keep this knowledge. Even during the network training, it supports the winning bit in its increasing. Decision and training based on Euclid distance improve output values, on average, and we also see average performance results on Hadamard codewords despite using 15 bit-long codewords. According to Zadeh fuzzy logic, the decision tends to decrease the maximum output error, given the class codeword during the training. During the test, the output is assigned to the codeword with a minimum of maximal bit errors. Loosely speaking, while a one-hot softmax pulls the best bit value to “1”, the proposed Zadeh’s decision pushes the worst error to “0”. Table 4 shows similar results for both strategies. We can conclude that what the softmax gives to one-hot encoding, the Zadeh fuzzy logic provides to binary coding.

In the case of the Zadeh decision, we can offer more codewords to the referee. It decreases the FCA compared with the accuracy, and the difference indicates how many samples were assigned to the codewords outside the class-codewords. We interpret the case as error detection, and it bypasses classification. If only class-codewords are applied during the testing, there is no difference between the accuracy and FCA. It is the case of one-hot encoding with softmax where only class-codewords can be used. Therefore, if the classification returns one of the class-codewords, it has to be accepted. The misclassified codeword causes an undetectable error. Table 4 indicates some benefits of error detection. The exclusion of untrustworthy samples increases the classification reliability.

A well-known way to increase reliability is to introduce a level of decision-making. All decision rules are based on the evaluation of a quantitative variable. The usual way to improve the decision reliability is to divide the variable interval by threshold Δ and reject all samples under the threshold, bypassing the decision procedure. This rule can also be applied to the softmax concept in which the variable is in a range $[1/n, 1]$, where n is the number of classes. Therefore, the threshold setting below $1/n$ has no meaning, and it gives no improvement. Simply put, in the softmax decision, the sample is accepted for classification if the highest value of the output exceeds the given threshold. We can apply the same approach in the Zadeh decision and decide if $\max\{S^0, \dots, S^w\} \geq \Delta$. Theorem 1 says that the threshold $\Delta < 0.5$ brings no improvement in the full codespace decision. We show the accuracy–reliability characteristic using different thresholds for proposed CRC7 with the Zadeh decision and baseline one-hot encoding with softmax in Figure 3.

Figure 3. Accuracy–reliability characteristics for different network architectures.

Figure 3 maps values from Table 4 to the left located endpoints of the curves, where no threshold is applied. As the threshold increases, the points are shifted to the right. The error detection property makes the location of the zero threshold points outside the line $y = x$. In practice, we are interested in the accuracy–reliability characteristic mainly for the highest accuracy values (where the system’s accuracy is within a few percentages of the maximal accuracy). Here, we can see that both methods provide similar results.

3.3. Out-of-Distribution Performance

Error detection is an approach to determine whether the senseword (a codeword assigned to the test sample) has agreed with properties that define a particular subspace of the codespace [15]. As we train the network to transform training patterns into the error detecting codewords, we assume that we allocate these samples into a subspace keeping the error detection ability. We hypothesize that if a sample is qualitatively different from the training patterns, the output codeword will have other properties, i.e., it will lie outside of the code subspace generated by the training set. Although the main goal is to show that binary coding can be trained to achieve high accuracy, out-of-distribution (outliers) detection is a side effect; therefore, we also comment on it.

In our case, the in-distribution training set was the CIFAR-10 dataset. As the out-of-distribution dataset, we chose datasets with various complexities, from the simplest (MNIST, Fashion-MNIST) through similar (CIFAR-100) to the most complex (white noise). The natural (and very undesirable) property of the recognition systems that maps all input patterns into one of the output classes is their defenselessness against outliers. It is the case of the softmax decision in our study. On the other side, error-detecting codes have an implicit ability to filter sensewords outside the subspace with class-codewords. This sorting generates rejected sensewords and improves recognition reliability. Table 5 shows the percentage of detected outliers from different out-of-distribution datasets. We can see that outliers are naturally detected by networks trained directly for error detection even without applying the threshold.

Table 5. Rejection on out-of-distribution datasets with no threshold.

Network	Code	Decision	MNIST	Fashion MNIST	CIFAR-100	Noise
CNN1	CRC7	Zadeh	10.29	11.59	11.16	20.6
CNN1	one-hot	softmax	0	0	0	0
CNN2	CRC7	Zadeh	12.37	15.55	11.56	46.05
CNN2	one-hot	softmax	0	0	0	0
ResNet20v2	CRC7	Zadeh	10.77	10.49	5.09	11.61
ResNet20v2	one-hot	softmax	0	0	0	0

Similar to the reliability, the out-of-distribution performance can be improved by applying a threshold during decision-making. Figure 4 shows that in both decision rules, the out-of-distribution rejection significantly increases with the threshold. For CNN1/CNN2 architectures with CIFAR-100/Fashion MNIST datasets, we can see a similar pattern in the accuracy–reliability characteristic. For high in-distribution FCA levels, the performance of both methods is very similar. For lower in-distribution FCA levels, one-hot encoding with softmax provides better out-of-distribution performance. CRC7 with the Zadeh decision performs better on the noise dataset, especially for ResNet20v2 architecture. On the contrary, one-hot encoding with softmax shows better performance in the rest of the ResNet20v2 experiments.

Figure 4. Out-of-distribution rejection (from left to right: CIFAR-100, Fashion MNIST, white noise) versus in-distribution FCA (full codespace accuracy) in studied networks and decision rules.

4. Conclusions

We examined the possibilities to use binary coding as the output coding in neural networks. We do not directly embed training dataset patterns into the coding space. It allows us to use the existing knowledge in the coding theory to design class-codewords instead. We use linear block codes for the possibility of the separation of class information from the checksum part of the codeword. We show that this property allows speeding up the decoding part of the classification. During the decoding phase, the similarity of the predicted output and the class-codewords usually needs to be computed. In the proposed approach, it is enough to take information (class) bits of the truest codeword (a rounded network normalized output) if the codeword gives the zero syndrome. We have proven that this simple procedure returns the truest decision from the Zadeh fuzzy logic point of view. Moreover, the truth-value of the decision can be notified to the user. We studied the error detection capabilities of binary coding. Error detection means that if the network's output does not have the properties of the codewords assigned to the classes, such outcomes are untrustworthy and thus rejected during the classification. It is a different approach to assess the trustworthiness of the prediction compared to the softmax case, where the threshold on the winning output's confidence value is used. Of course, we can combine both approaches, and the paper shows an advantage of such symbiosis. It enables us to reach a higher classification reliability if we are willing to accept more unclassified patterns.

We proposed a novel classification decision method based on the Zadeh fuzzy logic. We can interpret the neural network's output value to measure the statement's truth that the pattern belongs to a given class. During the network training, we aim to maximize the truthiness of the correct answer. In the well-established one-hot encoding with the softmax decision, minimizing the error at the output corresponding to the valid class leads to minimizing deviations in the rest of the outputs. The decision based on Zadeh fuzzy logic has a similar ability. Minimizing the maximum error in the required codeword also tends to suppress the errors in the rest of the outputs. We chose CRC codes as representative of linear block codes. We tested different approaches to the classification decision using the CRC output coding and showed that the proposed Zadeh decision method achieves the best accuracy. Moreover, the Zadeh decision with CRC achieves comparable results to one-hot encoding with softmax while maintaining the error detection capability. Further experiments on reliability and out-of-distribution data confirm that the proposed method achieves similar performance as one-hot encoding with softmax for high accuracy levels.

This paper performed a direct comparison between CRC coding with Zadeh decision and one-hot encoding with softmax. For this reason, we only used 10-bit codewords for 10-class classification. Contrary to the one-hot encoding, linear block codes (and binary codes in general) do not limit the size of the codewords, giving us more freedom in output coding strategies. They open up further possibilities to find better binary outputs codes. One way to use this freedom is to construct more compact output codes. Binary codes can encode n classes to a maximum of $\log_2(n) + 1$ output bits. This feature can be practical, especially for tasks with a large number of classes. Another way is to use a large codespace to improve performance or enhance error detection and/or correction abilities. Authors in [25] show that the longer target Hadamard code achieved higher accuracy than the shorter one. They also argue that longer codes consistently perform better than the shorter codes given the same training epoch. In this paper, we focused on a 10-label classification task represented by CIFAR-10 dataset. Although we consider the results to be very promising, using only a 10-class dataset can be considered a limitation of this study. Therefore, we plan to conduct additional experiments on datasets with a larger number of classes and with different output coding strategies.

Although the article focuses on detection codes, the same fuzzy logic approach can be applied to correction codes. We believe in combining embedding and correction/detection properties of the coding space. ECOC partially covers this idea, but OVO/OVA does not allow a holistic approach to codeword training. Applying general linear block codes speeds up the detection/correction of untrusted results, which is advantageous for tasks

with many classes. We can fit the codes to the obtained error statistics, and the minimum Hamming distance criterion is only one among others. The outlined problems show the direction for further research.

Supplementary Materials: The following are available online at: https://github.com/5erl/DNNs_classification_via_BEDOC.git, accessed on 12 April 2021.

Author Contributions: Conceptualization, M.K. and P.T.; methodology, M.K. and P.T.; software, P.L.; validation, M.K., P.T. and P.L.; formal analysis, P.L.; investigation, M.K., P.T. and P.L.; resources, M.K., P.T. and P.L.; data curation, P.L.; writing—original draft preparation, M.K., P.T. and P.L.; visualization, P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Operational Program Integrated Infrastructure within the project: Research in the SANET network and possibilities of its further use and development, ITMS code: NFP313010W988, co-financed by the European Regional Development Fund.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in this study are publicly available from sources cited in the paper: MNIST [39], Fashion MNIST [40], CIFAR-10 [41], CIFAR-100 [41].

Acknowledgments: Authors thank peer-reviewers for their valuable comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rocha, A.; Goldenstein, S.K. Multiclass from Binary: Expanding One-Versus-All, One-Versus-One and ECOC-Based Approaches. *IEEE Trans. Neural Networks Learn. Syst.* **2014**, *25*, 289–302. [CrossRef] [PubMed]
2. Aly, M. Survey on multiclass classification methods. *Neural Netw.* **2005**, *19*, 1–9.
3. Anand, R.; Mehrotra, K.; Mohan, C.K.; Ranka, S. Efficient classification for multiclass problems using modular neural networks. *IEEE Trans. Neural Netw.* **1995**, *6*, 117–124. [CrossRef] [PubMed]
4. Hastie, T.; Tibshirani, R. Classification by pairwise coupling. *Ann. Stat.* **1998**, *26*. [CrossRef]
5. Friedman, J.H. *Another Approach to Polychotomous Classification*; Department of Statistics, Stanford University: Stanford, CA, USA, 1996.
6. Platt, J.C.; Cristianini, N.; Shawe-Taylor, J. Large margin DAGs for multiclass classification. *NIPS 2000*, 547–553. [CrossRef]
7. Šuch, O.; Barreda, S. Bayes covariant multi-class classification. *Pattern Recognit. Lett.* **2016**, *84*, 99–106. [CrossRef]
8. Dietterich, T.G.; Bakiri, G. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *J. Artif. Intell. Res.* **1995**, *2*, 263–286. [CrossRef]
9. Pawara, P.; Okafor, E.; Groefsema, M.; He, S.; Schomaker, L.R.; Wiering, M.A. One-vs-One classification for deep neural networks. *Pattern Recognit.* **2020**, *108*, 107528. [CrossRef]
10. Titsias, M.K. One-vs-each approximation to softmax for scalable estimation of probabilities. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Ithaca, NY, USA, 2016; Volume 29.
11. Rodríguez, P.; Bautista, M.A.; González, J.; Escalera, S. Beyond one-hot encoding: Lower dimensional target embedding. *Image Vis. Comput.* **2018**. [CrossRef]
12. Zhou, J.; Peng, H.; Suen, C.Y. Data-driven decomposition for multi-class classification. *Pattern Recognit.* **2008**, *41*, 67–76. [CrossRef]
13. Zadeh, L.A. Fuzzy sets. *Inf. Control.* **1965**, *8*, 338–353. [CrossRef]
14. Zadeh, L.A. Soft computing, fuzzy logic and recognition technology. In *IEEE World Congress on Computational Intelligence, Proceedings of the 1998 IEEE International Conference on Fuzzy Systems Proceedings*, Anchorage, AK, USA, 4–9 May 1998; Cat. No.98CH36228. IEEE: New York, NY, USA, 2002.
15. Blahut, R.E. *Algebraic Codes for Data Transmission*, 1st ed.; Cambridge University Press: Cambridge, UK, 2003.
16. Roth, R. *Introduction to Coding Theory*; Cambridge University Press: Cambridge, UK, 2006.
17. Varshney, K.R.; Alemzadeh, H. On the Safety of Machine Learning: Cyber-Physical Systems, Decision Sciences, and Data Products. *Big Data* **2017**, *5*, 246–255. [CrossRef]
18. Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; Mané, D. Concrete Problems in AI Safety. *arXiv* **2004**, arXiv:1606.0656.
19. Jiang, H.; Kim, B.; Gupta, M.; Guan, M.Y. To trust or not to trust a classifier. *arXiv* **2018**, arXiv:1805.11783.
20. Mukhoti, J.; Kulharia, V.; Sanyal, A.; Golodetz, S.; Torr, P.H.S.; Dokania, P.K. Calibrating Deep Neural Networks using Focal Loss. *arXiv* **2020**, arXiv:2002.09437.
21. Guo, C.; Pleiss, G.; Sun, Y.; Weinberger, K.Q. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, Sydney, Australia, 6–11 August 2017.

22. Nguyen, A.; Yosinski, J.; Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 427–436.
23. Hendrycks, D.; Gimpel, K. A Baseline for Detecting Misclassified Out-of-Distribution Examples. In Proceedings of the International Conference on Learning Representations 2017 (ICLR 2017), Toulon, France, 24–26 April 2017.
24. DeVries, T.; Taylor, G.W. Learning confidence for out-of-distribution detection in neural networks. *arXiv* **2018**, arXiv:1802.04865.
25. Grigoryan, A.; Grigoryan, M. Hadamard Transform. In *Brief Notes in Advanced DSP*; Apple Academic Press: Palm Bay, FL, USA, 2009; pp. 185–256.
26. Yang, S.; Luo, P.; Loy, C.C.; Shum, K.W.; Tang, X. Deep representation learning with target coding. In Proceedings of the National Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
27. Akhtar, N.; Mian, A. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* **2018**, *6*, 14410–14430. [\[CrossRef\]](#)
28. Kim, D.; Bargal, S.A.; Zhang, J.; Sclaroff, S. Multi-way Encoding for Robustness. In Proceedings of the 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass Village, CO, USA, 1–5 March 2020; pp. 1341–1349.
29. Qin, J.; Liu, L.; Shao, L.; Shen, F.; Ni, B.; Chen, J.; Wang, Y. Zero-Shot Action Recognition with Error-Correcting Output Codes. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1042–1051.
30. Wolpert, D.H.; Dietterich, T.G. Error-Correcting Output Codes: A General Method for Improving Multiclass Inductive Learning Programs. In *The Mathematics of Generalization*; CRC Press: New York, NY, USA, 2019. [\[CrossRef\]](#)
31. Lachaize, M.; le Hégarat-Masclé, S.; Aldea, E.; Maitrot, A.; Reynaud, R. Evidential framework for Error Correcting Output Code classification. *Eng. Appl. Artif. Intell.* **2018**, *73*, 10–21. [\[CrossRef\]](#)
32. Li, K.-S.; Wang, H.-R.; Liu, K.-H. A novel Error-Correcting Output Codes algorithm based on genetic programming. *Swarm Evol. Comput.* **2019**, *50*. [\[CrossRef\]](#)
33. Lian, S. *Principles of Imprecise-Information Processing: A New Theoretical and Technological System*; Springer: Berlin/Heidelberg, Germany, 2016.
34. Rudolph, L.; Hartmann, C.; Hwang, T.-Y.; Duc, N. Algebraic analog decoding of linear binary codes. *IEEE Trans. Inf. Theory* **1979**, *25*, 430–440. [\[CrossRef\]](#)
35. von Kaenel, A.P. Fuzzy codes and distance properties. *Fuzzy Sets Syst.* **1982**, *8*, 199–204. [\[CrossRef\]](#)
36. Tsafack, S.A.; Ndjeya, S.; Strümgmann, L.; Lele, C. Fuzzy Linear Codes. *Fuzzy Inf. Eng.* **2018**, *10*, 418–434. [\[CrossRef\]](#)
37. Klement, E.P.; Mesiar, R.; Pap, E. Triangular norms: Basic notions and properties. In *Logical, Algebraic, Analytic and Probabilistic Aspects of Triangular Norms*; Elsevier: Amsterdam, The Netherlands, 2005; pp. 17–60.
38. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Technical Report TR-2009; University of Toronto: Toronto, ON, Canada.
39. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [\[CrossRef\]](#)
40. Han, X.; Kashif, R.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
41. Krizhevsky, A.; Nair, V.; Hinton, G. CIFAR-10 and CIFAR-100 Datasets. Available online: <https://www.cs.toronto.edu/~kriz/cifar.html> (accessed on 8 April 2009).
42. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.
43. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity Mappings in Deep Residual Networks. In *Transactions on Petri Nets and Other Models of Concurrency XV*; Springer Science and Business Media: Berlin/Heidelberg, Germany, 2016; pp. 630–645.
44. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference Learn, San Diego, CA, USA, 5–8 May 2015.
45. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision, Piscataway, NJ, USA, 7–13 December 2015; pp. 1026–1034.
46. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M. TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016), Savannah, GA, USA, 2–4 November 2016.
47. François, C. Keras. Github Repository. Available online: <https://github.com/fchollet/keras> (accessed on 5 May 2017).