Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Deep Representation Learning with Target Coding

Shuo Yang[1], **Ping Luo**[2,1], **Chen Change Loy**[1,2], **Kenneth W. Shum**[1], and **Xiaoou Tang**[1,2]
[1]Department of Information Engineering, The Chinese University of Hong Kong
[2]Shenzhen Key Lab of CVPR, Shenzhen Institutes of Advanced Technology,
Chinese Academy of Sciences, Shenzhen, China
{ys014,pluo,ccloy,xtang}@ie.cuhk.edu.hk, {wkshum}@inc.cuhk.edu.hk

Abstract

We consider the problem of learning deep representation when target labels are available. In this paper, we show that there exists intrinsic relationship between target coding and feature representation learning in deep networks. Specifically, we found that distributed binary code with error correcting capability is more capable of encouraging discriminative features, in comparison to the 1-of-K coding that is typically used in supervised deep learning. This new finding reveals additional benefit of using error-correcting code for deep model learning, apart from its well-known error correcting property. Extensive experiments are conducted on popular visual benchmark datasets.

Introduction

Learning robust and invariant representation has been a long-standing goal in computer vision. In comparison to hand-crafted visual features, such as SIFT or HoG, features learned by deep models have recently been shown more capable of capturing abstract concepts invariant to various phenomenon in visual world, e.g. viewpoint, illumination, and clutter (Girshick et al. 2014; Krizhevsky, Sutskever, and Hinton 2012; Ouyang et al. 2014; Sun, Wang, and Tang 2014; Sun et al. 2014; Zhu et al. 2014; 2013; Luo, Wang, and Tang 2012). Hence, an increasing number of studies are now exploring the use of deep representation (Razavian et al. 2014) on vision problems, particularly on classification tasks.

Existing studies first learn a deep model, e.g. convolutional neural network, in a supervised manner. The 1-of-K coding, containing vectors of length K, with the k-th element as one and the remaining zeros, is typically used along with a softmax function for classification. Each element in a 1-of-K code represents a probability of a specific class. Subsequently, the features of a raw image are extracted from the penultimate layer (the layer before the output layer) or shallower layers, to form a high-dimensional feature vector as input to classifiers such as Support Vector Machine (SVM).

Is 1-of-K the best target coding method? The goal of this paper is to investigate an alternative target coding approach for the aforementioned deep feature learning pipeline. There have been intensive efforts to improve supervised learning of deep models, e.g. joint learning of embedding task using unlabeled data (Weston, Ratle, and Collobert 2008), multi-task learning (Zhang et al. 2014), introducing intermediate training phase (Goh et al. 2013), and replacing softmax layer with SVM (Tang 2013). Nevertheless, the influence of target coding is little explored in deep representation learning. We believe that feature representation can be influenced by the training signals encoded in the target formation. We therefore hypothesize that better features can be induced if we design the target code appropriately through coding theory.

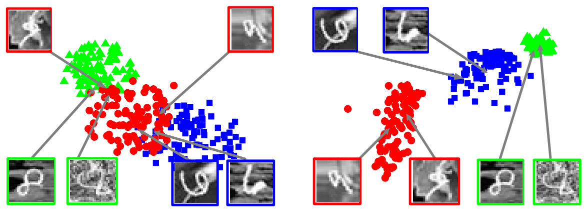


Figure 1: Is 1-of-K the best target coding? To examine the feature representation when a deep model is trained with different coding schemes, we project the features extracted from a deep convolutional neural network's penultimate layer to a two-dimensional embedding using multi-dimensional scaling. Hadamard coding is capable of separating all visually ambiguous classes {9, 8, 6}, which are confused in 1-of-K coding.

Target coding is gaining popularity in machine learning community for the multi-label prediction problem (Hsu et al. 2009; Cisse et al. 2013). The studies that are closer to our work are (Dietterich and Bakiri 1994; Langford and Beygelzimer 2005), which solve multi-class learning problems via error-correcting output codes. The key idea of these studies is to exploit error-correcting codes as target vectors, so that a classifier can recover from constant fraction of binary errors in target prediction. Our work has a different focus: *we wish to investigate how a target coding would influence the feature representation in a deep network, beyond its current use for error correcting*. This perspective is new.

We present our interesting observation in Figure 1 to demonstrate what one could achieve by replacing 1-of-K

with the Hadamard code (Langford and Beygelzimer 2005)[1] as target vectors in deep feature learning. It is observed that even on just a two-dimensional embedding space, the features induced by the Hadamard code-based learning can already be easily separable. In contrast, the feature clusters induced by 1-of-K are overlapping. The separation of such clusters may only be possible at higher dimensions.

In this paper we make the following contributions: (i) We present the first attempt to analyse systematically the influences of target coding to feature representation; (ii) We show that the error-correcting Hadamard code encompasses some desired properties that are beneficial for deep feature learning. (iii) We validate the coding scheme with detailed examination of feature representation in each hidden layer of a deep model. Extensive experiments demonstrate that error correcting codes are beneficial for representation learning, leading to state-of-the-art results on MNIST, STL-10, CIFAR-100, and ILSVRC-2012, in comparison to the typically used 1-of-K coding scheme.

Preliminaries

A Definition of Target Coding

We first provide a general definition of target coding (or target code), which has 1-of-K code as a special case.

Definition 1. Let $\mathcal{T}$ be a set of integers, called the *alphabet set*. An element in $\mathcal{T}$ is called a *symbol*. For example, $\mathcal{T} = \{0, 1\}$ is the binary alphabet set. A *target code* $\mathcal{S}$ is a matrix $\mathcal{S} \in \mathcal{T}^{n \times l}$. Each row of a target code is called a *codeword*. Here, l denotes the number of symbols in each codeword and n denotes the total number of codewords. For a target code $\mathcal{S}$, we denote $\mathcal{A} = \{\boldsymbol{\alpha}_i\}_{i=1}^{n}$ be the set of empirical distributions of symbols in the rows of $\mathcal{S}$, i.e. for $i = 1, 2, \ldots, n$, $\boldsymbol{\alpha}_i$ is a vector of length $|\mathcal{T}|$, with the t-th component of $\boldsymbol{\alpha}_i$ counting the number of occurrence of the t-th symbol in the i-th row of $\mathcal{S}$. Similarly, we let $\mathcal{B} = \{\boldsymbol{\beta}_j\}_{j=1}^{l}$ be the set of empirical distributions of symbols in the columns of $\mathcal{S}$. Given two distinct row indices i and i', the *Hamming distance* between row i and row i' of a target code $\mathcal{S}$ is defined as $|\{j : \mathcal{S}_{ij} \neq \mathcal{S}_{i'j}\}|$, i.e. it counts the number of column indices such that the corresponding symbols in row i and row i' are not equal. For simplicity, we call it *pairwise Hamming distance*.

The second column of Table 1 shows an example of 1-of-K binary target code, which is typically used in deep learning for representing K classes. Each of the K symbols, either '0' or '1', indicates the probability of a specific class. The target code here can be written as $\mathcal{S} = I$, where $I \in \mathcal{T}^{K \times K}$ is an identity matrix. It is easy to attain some properties of the 1-of-K coding. For instance, let $\boldsymbol{\alpha}_{it}$ represents the counts of the t-th symbol in the i-th row, for $i = 1, 2, ..., K$, we have $\boldsymbol{\alpha}_{i1} = \frac{K-1}{K}$ and $\boldsymbol{\alpha}_{i2} = \frac{1}{K}$, since only one symbol in each codeword has a value '1'. Similarly, we have $\boldsymbol{\beta}_{j1} = \frac{K-1}{K}$ and $\boldsymbol{\beta}_{j2} = \frac{1}{K}$. The pairwise Hamming distance is two.

[1]A well-known error-correcting coding method based on Hadamard matrices.

Table 1: Target codes for representing different visual classes.

Classes	1-of-K							Hadamard code of length 7						
bird	1	0	0	0	0	0	0	1	0	1	0	1	0	1
cat	0	1	0	0	0	0	0	0	1	1	0	0	1	1
dog	0	0	1	0	0	0	0	1	1	0	0	1	1	0
human	0	0	0	1	0	0	0	0	0	0	1	1	1	1
horse	0	0	0	0	1	0	0	1	0	1	1	0	1	0
bench	0	0	0	0	0	1	0	0	1	1	1	1	0	0
deer	0	0	0	0	0	0	1	1	1	0	1	0	0	1

Constructing Class Label from Hadamard Code

We wish to show that apart from representing classes, the target coding can play additional roles. Error-correcting in target prediction is a good example that makes good use of target coding (Dietterich and Bakiri 1994; Langford and Beygelzimer 2005). We show that apart from error correcting, target coding is capable of facilitating the learning of better feature representation.

In this paper, we study a popular error-correcting code, the Hadamard code, which has been used as target coding for classifier for its capability of error-correcting (Langford and Beygelzimer 2005). A Hadamard code can be generated from the Hadamard matrix (Hadamard 1893; Hedayat and Wallis 1978). A matrix $\mathcal{H} \in \{+1, -1\}^{m \times m}$, whose entries are either '$+1$' or '-1', is called the Hadamard matrix if $\mathcal{H}\mathcal{H}^{\mathsf{T}} = mI$, where I is an identity matrix. The definition of the Hadamard matrix requires that any pair of distinct rows and columns are orthogonal, respectively.

A possible way to generate the Hadamard matrix is by the Sylvester's method (Hedayat and Wallis 1978), where a new Hadamard matrix is produced from the old one by the Kronecker (or tensor) product. For instance, given a Hadamard matrix $\mathcal{H}^2 = [+ +; + -]$, we can produce $\mathcal{H}^4$ by $\mathcal{H}^4 = \mathcal{H}^2 \otimes \mathcal{H}^2$ as below, where $\otimes$ denotes the Kronecker product. Similarly, $\mathcal{H}^8$ is computed by $\mathcal{H}^4 \otimes \mathcal{H}^2$.

$$\mathcal{H}^4 = \begin{bmatrix} + & + \\ + & - \end{bmatrix} \otimes \begin{bmatrix} + & + \\ + & - \end{bmatrix} = \begin{bmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{bmatrix},$$

$$\mathcal{H}^8 = \begin{bmatrix} + & + & + & + & + & + & + & + \\ + & - & + & - & + & - & + & - \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ + & + & + & + & - & - & - & - \\ + & - & + & - & - & + & - & + \\ + & + & - & - & - & - & + & + \\ + & - & - & + & - & + & + & - \end{bmatrix}.$$

Therefore, the size of the Hadamard matrix is a power of 2. If we look at any two distinct rows (or columns) of a Hadamard matrix of length m, there are exactly $m/2$ pairs of entries whose signs agree with each other, and exactly $m/2$ pairs whose signs are opposite. Thus, the distance between any two rows is a constant and equals $m/2$.

Now, we can construct the target code $\mathcal{S}$ from the Hadamard matrix. Let $\mathcal{H} \in \{+1, -1\}^{m \times m}$ be a Hadamard matrix. $\mathcal{S} \in \mathcal{T}^{(m-1)\times(m-1)}$ is obtained by first removing the first row and the first column of $\mathcal{H}$, and then mapping '$+1$' to '0' and '-1' to '1'. According to the definition of $\mathcal{H}$, there are exactly $m/2$ ones in each row and each column of $\mathcal{S}$. Since any two rows of a Hadamard matrix are orthogonal, the pairwise Hamming distance of $\mathcal{S}$ is equal to $m/2$. The right-most column of Table 1 gives an example of $\mathcal{S}$ of size 7×7, obtained from $\mathcal{H}^8$ as above.

Note that for most of the time, we cannot achieve a target code that contains exactly K codewords for K number of classes, since the length of the Hadamard matrix, m is power of 2. Without loss of generality, we let $\mathcal{C} \in \mathcal{T}^{K \times (m-1)}$ be a matrix of labels of K classes, i.e. for $i = 1, 2, ..., K$, the i-th class is labeled by the i-th row in $\mathcal{C}$. Here, $\mathcal{C}$ is constructed by choosing K codewords from $\mathcal{S} \in \mathcal{T}^{(m-1) \times (m-1)}$.

Relevance of Target Coding in Deep Learning

In this section we discuss the properties of the Hadamard code, which is well-established in previous works (Hadamard 1893; Hedayat and Wallis 1978), and relate these properties to representation learning in deep model.

Property 1. Uniformness in each row of $\mathcal{S}$ – Each row of a Hadamard code has $\frac{m}{2}$ symbols that equals one.

The row uniformness introduces redundancy to target prediction. Specifically, each row of $\mathcal{S}$ represents a class label. Rather than having exactly a single symbol '1' as in the conventional 1-of-K code, each codeword in a Hadamard code has the same number of redundant symbols that are equal to '1'. This property suggests the error-correcting property, which allows for a stronger resilience in target responses against noise-corrupted raw imagery inputs. Specifically, if a limited number of symbols is corrupted, recognising the true object class may still be possible. More symbols '1' (subject to other properties as follows) translate to a larger error-correcting capacity, which enables the responses at the target layer to be used directly for robust nearest neighbour classification.

Property 2. Uniformness in each column of $\mathcal{S}$ – Similarly, each column of a Hadamard code has $\frac{m}{2}$ symbols that equals one.

This property is little explored but desired in representation learning. In essence, it shares a similar spirit advocated by (Bengio 2009), which can be further traced back to the Perceptron algorithm by (Rosenblatt 1958). Specifically, each column of $\mathcal{S}$ can be treated as a 'decision function', which encourages the partitioning of the input space into two balanced regions, i.e. approximately half of the classes will lie on a region, and the remaining classes will be grouped to another region. Given multiple columns, different partitions can be induced and thus combined to give rise to a potentially exponential number of region intersections in the input space. With this property, a Hadamard based target code essentially promotes such formation of complex regions to induce features that separate the data better, and consequently provide discriminative and robust description for the data. We will show this phenomenon by examining layer-wise feature representation in the experiment section.

Property 3. Constant pairwise Hamming distance of $\mathcal{S}$ – As the Hadamard code is both row and column uniform, the pairwise Hamming distance is also $\frac{m}{2}$, implying that the Hamming distance between any two codewords is half of the codeword's length.

This property naturally demands each class to have its own unique codeword equally distanced away from those of all other classes. Samples from the same class are forced to map to the same codeword. This property is thus useful to enforce such mapping and encourage invariant feature representation.

Discussion: Recall that the class labels $\mathcal{C} \in \mathcal{T}^{K \times (m-1)}$ is constructed by choosing K codewords from $\mathcal{S} \in \mathcal{T}^{(m-1) \times (m-1)}$. However, since m is power of 2, $\mathcal{C} \in \mathcal{T}^{K \times (m-1)}$ violates the second property when the class number is smaller than $m-1$, that is $K < m-1$. We propose a greedy search method to select $\mathcal{C}$ from $\mathcal{S}$ to approximate the second property. This method works well in practice because the search space is actually small (i.e. $m-1$). Specifically, a codeword from $\mathcal{S}$ is chosen, iff appending this codeword to $\mathcal{C}$ does not disrupt the uniformness of each column. For example, we construct $\mathcal{C}$ when $K = 10$ and $m = 128$ or 256, using both greedy search and random selection. The distributions of the numbers of '1' in columns are illustrated in Fig.2, which shows that greedy search can better fulfil the column-uniformness (i.e. ideally, all columns contain 5 symbols '1'. Specifically, for the greedy search when $m = 256$, 95% columns contains $4 \sim 6$ symbols '1'), while random search leads to trivial columns.

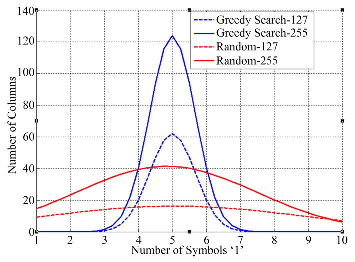


Figure 2: Greedy search better preserves column-uniformness, while random search violates.

Using Target Code in Convolutional Network

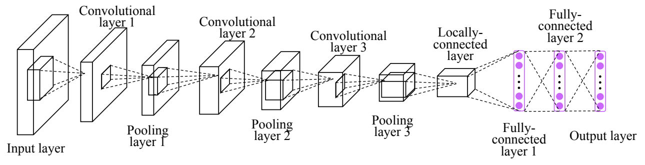


Figure 3: The CNN structure used in our experiments.

Given a data set with N training samples, $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, where $\mathbf{x}_i$ denote the i-th sample and $\forall_i \mathbf{y}_i \in \mathcal{C}$ is the corresponding label. We adopt the deep convolutional network (CNN) to learn feature representation by minimizing the Eu-

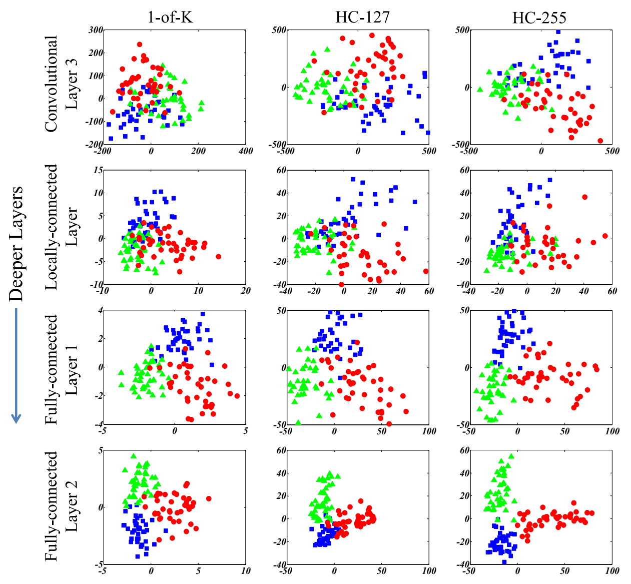


Figure 4: The change of features observed when applying 1-of-K and the Hadamard coding: We deliberately choose three visually ambiguous classes {tiger, lion, leopard} from CIFAR-100, indicated by red, green, blue, respectively. We extracted the features from different CNN layers depicted in Figure 3, namely convolution layer 3, locally-connected layer, and fully-connected layers 1 and 2. To examine the feature representation given different coding schemes, we project the features to a two-dimensional embedding using multi-dimensional scaling. We compare 1-of-K against the Hadamard code with length 127 and 255 (HC-127 and HC-255). It is observed that the Hadamard code yields better data factorization than 1-of-K coding. Longer Hadamard code leads to more separable and distinct feature clusters.

clidean loss,

$$\underset{\{\mathbf{W}_\ell,\mathbf{b}_\ell\}_{\ell=1}^{L}}{\operatorname{argmin}} \sum_{i=1}^{N} \| \mathbf{y}_i - f(\mathbf{x}_i; \{\mathbf{W}_\ell, \mathbf{b}_\ell\}_{\ell=1}^{L}) \|_2^2, \qquad (1)$$

where $\mathbf{W}_\ell$ and $\mathbf{b}_\ell$ denote the filters and the biases at the ℓ-th layer, respectively. As the Hadamard code is independent to a network structure, any other deep models can be employed as well. Exploring an optimal structure is out of the scope of this paper. Instead, we focus on the effectiveness of the target code based on a common CNN model. The CNN architecture is shown in Figure 3, which follows the structure in (Srivastava and Salakhutdinov 2013). The details of the network parameters are provided in the supplementary material. For all experiments, we apply the same network structure in order to verify whether the performance gain is attained by target coding.

Experiments

Influence of Target Coding at Deep Layers

We are interested to examine the influence of target coding on feature representation in each deep layer. We use CIFAR-100 dataset in this experiment. To visualize the feature representation in each layer, we first extract features from the respective layers[2], and then project the features to a two-dimensional embedding space by using multi-dimensional scaling. Figure 4 visualizes the feature subspace of different layers of three CNNs. All CNNs have the same network structure as depicted in Figure 3, but trained using different target codings. It is observed that the Hadamard coding yields more discriminative sub-space clusters than the 1-of-K coding. The influence of the Hadamard code is less obvious in shallower layers, e.g. convolutional layer 3 and locally-connected layer, whilst greater influence is exerted at the top fully connected layers. It is also observed that using longer Hadamard code is beneficial since a longer code implies more 'decision functions', leading to more complex regions formation for better space factorization (see Property 2).

Evaluating the Effectiveness of Target Coding

We performed two sets of experiments[3] to quantitatively evaluate the effectiveness of target coding. Three popular benchmark datasets were used, i.e. variant of the MNIST dataset with irrelevant backgrounds and rotation[4], STL-10, and CIFAR-100. We followed the standard testing protocol and training/test partitions for each dataset. The details of each dataset are provided in the supplementary material.

In the first experiment, we examined how well the features extracted from a CNN's penultimate layer could perform. It is hard to measure the performance of features directly. A viable way practiced by existing studies is to apply them on a standard predictive model, e.g. k-NN or SVM, and measure the classification performance. In the second experiment, we evaluated the classification performance based directly on the CNN's outputs.

For the network design, we trained the same CNN structure using the Hadamard code and 1-of-K code, respectively. The structure is shown in Figure 3. Due to space constraint, the detailed settings of the CNN for each dataset are provided in the supplementary material. For the hyper-parameters, such as learning rates, we tried searching for optimal values for different coding schemes but observed no significant difference in performance. From our experiments, we observed that learning rate mainly affects the convergence speed. Different learning rate would still lead to similar and stable results when we allow sufficient training time for convergence. To have a fair comparison, we set the hyper-parameters the same and optimally for all methods. The only difference is thus on the target coding scheme.

Experiment I: Hidden-layer feature performance We applied PCA on the features extracted from the CNN model

[2]We used the toolbox implemented by (Krizhevsky, Sutskever, and Hinton 2012) to extract feature maps of different layers. The deep representation used in our experiment is simply the concatenation of feature maps from the same layer into feature vectors.

[3]http://mmlab.ie.cuhk.edu.hk/projects/TargetCoding/ (For more technical details of this work, please contact the corresponding author Ping Luo via pluo.lhi@gmail.com)

[4]This MNIST variant corresponds to the challenging *mnist-rot-back-image* reported in (Rifai et al. 2011; Sohn et al. 2013; Vincent et al. 2010).

Table 2: Classification accuracy (%) on MNIST, STL-10, and CIFAR-100. The best performers in each column are in bold. The Hadamard code and 1-of-K schemes employed the same CNN structure.

Methods	Hidden-layer features			Direct classification			k-NN on target response		
	MNIST	STL-10	CIFAR-100	MNIST	STL-10	CIFAR-100	MNIST	STL-10	CIFAR-100
Hadamard code, HC-63	78.44	67.04	–	81.46	71.51	–	77.38	71.56	–
Hadamard code, HC-127	82.03	68.97	58.96	84.02	72.56	63.17	81.89	72.15	62.21
Hadamard code, HC-255	**84.08**	**70.66**	**60.98**	**85.47**	**73.15**	**64.77**	**84.85**	**72.46**	**63.07**
1-of-K	72.83	60.01	59.55	79.25	69.82	61.00			
(Sohn et al. 2013): supervised PGBM	55.33	–	–	–	–	–			
(Sohn et al. 2013): PGBM+DN-1	–	–	–	63.24	–	–			
(Rifai et al. 2011): CAE-2	–	–	–	54.77	–	–			
(Vincent et al. 2010): SDAE-3	60.93	–	–	56.24	–	–			
(Bo, Ren, and Fox 2013): HMP	–	–	–	–	64.50	–			
(Gens and Domingos 2012): SPN	–	–	–	–	62.30	–			
(Zou et al. 2012):Simulated Fixations	–	–	–	–	61.00	–			
(Coates and Ng 2011): RF+L2-SVM	–	60.01	–	–	–	–			
(Krizhevsky, Sutskever, and Hinton 2012): CNN	77.24^a	68.30^a	–	78.06^a	70.80^a	–			
(Goodfellow et al. 2013): Maxout networks	–	–	–	–	–	61.43			
(Srivastava and Salakhutdinov 2013): CNN	–	–	57.26^b	–	–	62.88			
(Srivastava and Salakhutdinov 2013): Tree-based priors	–	–	–	–	–	63.15			
(Lin, Chen, and Yan 2013): NIN	–	–	–	–	–	64.32			

a. Results are obtained using the published code and network structures (Krizhevsky, Sutskever, and Hinton 2012). b. Result is obtained using the structure in (Srivastava and Salakhutdinov 2013).

(penultimate layer) and then performed k-NN classification. We chose k-NN instead of SVM to have a more direct evaluation on the feature performance. The 2nd-4th columns of Table 2 summarise the performance comparisons between the features guided by the Hadamard code and the typical 1-of-K coding schemes. We also list the state-of-the art results reported in the literature. For completeness, the list also covers unsupervised representation learning approaches (e.g. (Vincent et al. 2010)). Note that our key comparison is still between the Hadamard and typical 1-of-K codings.

The Hadamard code performs the best among different methods, even if we consider only a short code with a length of 63. In particular, a large margin of improvement is observed against 1-of-K coding, even on the challenging MNIST dataset with irrelevant patterns as background. The results suggest the benefits of using Hadamard code for learning robust and invariant features. Since the only change we made was replacing the 1-of-K coding with the Hadamard coding, the better performance is thus attributed to the more effective target coding. When the code length is increased to 255, we obtain even better results, with an average relative improvement over **11%** in comparison to 1-of-K coding across all three datasets. The results are expected since HC-255 contains more bits than HC-63 and HC-127, which helps inducing more complex region intersections for richer input description.

Figure 5 shows some classification examples. It can be observed that despite 1-of-K performs well on easy instances, e.g. apple and butterfly, its performance is inferior than the Hadamard coding-based CNN, when applied to difficult classes like raccoon and chimpanzee.

Experiment II: Direct classification performance With improved features learned in hidden layers, we expect better direct classification performance from the Hadamard coding. In this experiment, the baseline 1-of-K was used along with softmax for class prediction. For our approach, we evaluated two variants: (i) we pre-trained the CNN guided by the Hadamard code and subsequently applied logistic regression at the top layer for fine-tuning and class prediction; (ii) we

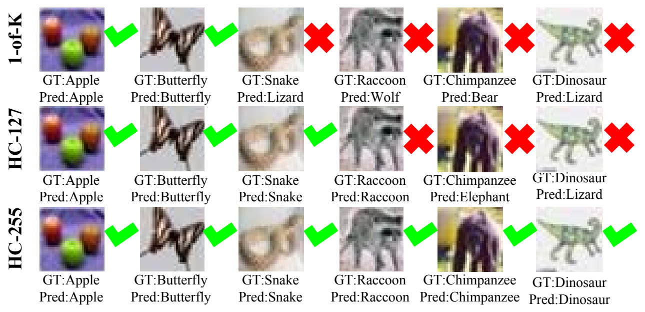


Figure 5: We show some CIFAR-100 classification results by CNN trained with 1-of-K, HC-127, and HC-255.

performed nearest neighbour class assignment by computing the Hamming distance between target responses (predicted codewords) and class codewords.

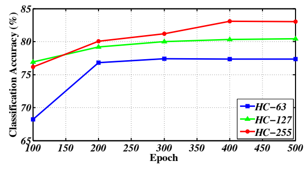
Columns 5th-7th summarise the results of variant-(i). The Balanced Code approach achieves the best accuracy reported so far on both MNIST and STL-10, whilst compares favourably to the state-of-the-art approaches (Goodfellow et al. 2013; Srivastava and Salakhutdinov 2013; Lin, Chen, and Yan 2013) on CIFAR-100. Note that (Goodfellow et al. 2013; Srivastava and Salakhutdinov 2013; Lin, Chen, and Yan 2013) involve elaborate strategies in their learning steps, e.g. multiview inference[5], sliding a micro network over the input, or dropping weight to achieve better regularisation. On the contrary, the Hadamard code method is evaluated on a standard CNN structure. Nevertheless, this does not prohibit one to adopt similar strategies as in the aforementioned studies to boost the CNN performance further. Columns 8th-10th provide the results of variant-(ii). This variant records comparable results to variant-(i) but achieves better performance than using hidden-layer features alone, thanks to the additional error-correcting capabil-

[5]Test error is computed as an average over 10 patches of the original images (4 corner patches, center patch, and their horizontal reflections). This will usually produce better results than testing on the center patch alone.

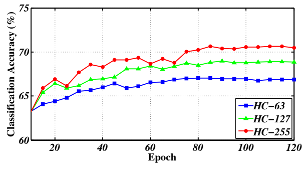
ity in the Hadamard target code. It is worth mentioning that variant-(ii) yields better results than 1-of-K and some contemporary methods even with just simple nearest neighbor classification.

Further Analysis

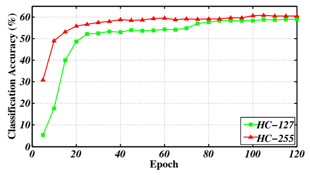
Performance convergence: Figure 6 shows that CNNs trained on (a) MNIST, (b) STL-10, and (c) CIFAR-100, guided by the Hadamard code of different lengths take almost the same number of iterations to converge to their best performance. It is observed that longer target code consistently performs better than the shorter code given the same training epoch. Again, longer code facilitates more complex region intersections (see Property 2), thus yielding better performance.



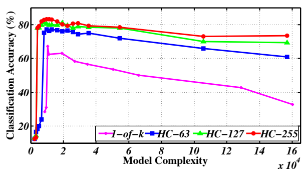
(a) MNIST



(b) STL-10



(c) CIFAR-100



(d) Model Complexities

Figure 6: (a)-(c) Feature performance of a CNN when trained with the Hadamard code (HC) of different lengths. The x-axis represents the number of training epochs. (d) Here we vary the number of filters in the last two layers of the CNN to obtain different model complexities. 1-of-K performs poorer than the Hadamard code under all complexities considered. MNIST dataset is used.

Performance under different model complexities: In this experiment, we trained the same CNN model on MNIST, but varying the number of filters in the last two convolutional layers. At different model complexities, we measure the test performance of hidden-layer features by using k-NN as a classifier. As can been seen in Figure 6 (d), the Hadamard code achieves the best classification accuracy rate when the number of parameters is set around 1×10^4, and its performance degrades gracefully with increasing model complexity due to over-fitting. The results suggest that by simply adjusting the model complexity, we are not able to bring 1-of-K's performance any close to the best performance of the Hadamard code.

Greedy search vs. random search: Our experiments on MNIST show that selecting class labels $\mathcal{C}$ from target code $\mathcal{S}$ by greedy search yields better performance than random search (84.85% vs. 84.38%) for HC-255. This suggests the importance of the second property, i.e. the column uniformness.

Table 3: Top-1 classification accuracy (%) on ImageNet-2012 validation set (predicting on the center patch).

	1-of-K	HC-1023	HC-2047
AlexNet (Krizhevsky, Sutskever, and Hinton 2012)	56.9	58.4	**58.9**
Clarifai (Zeiler and Fergus 2014)	59.4	61.4	**62.4**

Scalability to large number of classes: This part shows that the proposed method scales well to the 1000-category ImageNet-2012 dataset, which contains roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images. We constructed two baseline models, the AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and Clarifai (Zeiler and Fergus 2014). To demonstrate the effectiveness of target code, we first pre-train the networks by following the practices in (Krizhevsky, Sutskever, and Hinton 2012; Zeiler and Fergus 2014). However, we found that Hadamard code is difficult to fit when large number of categories and data are presented. To improve efficiency, the target code is equally divided into ten segments[6], each of which is used to fine-tune the above networks. Then the predictions of the segments are concatenated as the complete prediction of Hadamard code. These networks can be fine-tuned in parallel.

Specifically, the Clarifai is an extension of AlexNet, where the group connections used in AlexNet's layers 3, 4, 5 are replaced with dense connections in Clarifai and the filter size of the first layer in Clarifai is reduced from 11×11 of AlexNet to 7×7. For both networks, the other parameters are similar. Our implementation is based on Caffe (Jia 2013). For 1000 categories, the Hadamard matrix has a size at least 1024×1024. Thus, we evaluate HC-1023, HC-2047, and 1-of-K using the validation images and report the top-1 accuracies in Table 3. The accuracy of 1-of-K code is calculated based on the softmax function, while the accuracy of the Hadamard code is computed based on the Hamming distance between predicted codewords and class codewords. This experiment shows that the Hadamard code achieves better performance than 1-of-K code and longer code tends to improve the accuracies further.

Discussions

We have shown that the Hadamard code naturally comes with some useful characteristics for representation learning, apart from its well-know error correcting property. Experimental results have revealed its potential over the typical 1-of-K coding scheme. We experimented with the standard CNN (Srivastava and Salakhutdinov 2013), AlexNet (Krizhevsky, Sutskever, and Hinton 2012), and Clarifai (Zeiler and Fergus 2014) deep networks. All networks gained improved performance with the Hadamard coding.

Our study opens some interesting and new directions to pursue. Firstly, which property of the Hadamard code contribute the most to the feature representation learning? We found examining individual property separately not feasible, since these properties are strongly related. Nonetheless, we

[6]For example, for a target code with 1023 bits, the first nine segments have 100 bits, while the last one has 123 bits.

observed that fulfilling all properties are critical for good performance. For instance, 1-of-K coding only fulfils the constant Hamming distance property. It yielded poorer performance than the Hadamard coding. Without column uniformness, poorer results are observed. Without row uniformness would jeopardise the error-correcting capability. Secondly, the properties are by no means complete. We believe there exist other coding principles that worth further exploration. For instance, one could manipulate the inter-code distance based on the prior information about the superclasses of objects, i.e. the target code of orchid class should have a closer distance to the rose class than the dog class, which belongs to a different superclass. Finally, it will be interesting to find out if target coding works equally well for shallow classifiers.

References

Bengio, Y. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning* 2(1):1–127.

Bo, L.; Ren, X.; and Fox, D. 2013. Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics*, 387–402.

Cisse, M. M.; Usunier, N.; Artières, T.; and Gallinari, P. 2013. Robust bloom filters for large multilabel classification tasks. In *NIPS*.

Coates, A., and Ng, A. Y. 2011. Selecting receptive fields in deep networks. In *NIPS*.

Dietterich, T. G., and Bakiri, G. 1994. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2(1):263–286.

Gens, R., and Domingos, P. 2012. Discriminative learning of sum-product networks. In *NIPS*, 3248–3256.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Goh, H.; Thome, N.; Cord, M.; and Lim, J.-H. 2013. Top-down regularization of deep belief networks. In *NIPS*.

Goodfellow, I. J.; Warde-Farley, D.; Mirza, M.; Courville, A.; and Bengio, Y. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Hadamard, J. 1893. Résolution d'une question relative aux déterminants. *Bull. Sci. Math.* 17:30–31.

Hedayat, A., and Wallis, W. D. 1978. Hadamard matrices and their applications. *The Annals of Statistics* 6:1184–1238.

Hsu, D.; Kakade, S.; Langford, J.; and Zhang, T. 2009. Multi-label prediction via compressed sensing. In *NIPS*.

Jia, Y. 2013. Caffe: An open source convolutional architecture for fast feature embedding.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*.

Langford, J., and Beygelzimer, A. 2005. Sensitive error correcting output codes. In *COLT*.

Lin, M.; Chen, Q.; and Yan, S. 2013. Network in network. *arXiv preprint arXiv:1312.4400*.

Luo, P.; Wang, X.; and Tang, X. 2012. Hierarchical face parsing via deep learning. In *CVPR*.

Ouyang, W.; Luo, P.; Zeng, X.; Qiu, S.; Tian, Y.; Li, H.; Yang, S.; Wang, Z.; Xiong, Y.; Qian, C.; Zhu, Z.; Wang, R.; Loy, C. C.; Wang, X.; and Tang, X. 2014. Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. *arXiv preprint arXiv:1409.3505*.

Razavian, A. S.; Azizpour, H.; Sullivan, J.; and Carlsson, S. 2014. CNN features off-the-shelf: an astounding baseline for recognition. *arXiv preprint arXiv:1403.6382*.

Rifai, S.; Vincent, P.; Muller, X.; Glorot, X.; and Bengio, Y. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, 833–840.

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* 65(6):386.

Sohn, K.; Zhou, G.; Lee, C.; and Lee, H. 2013. Learning and selecting features jointly with point-wise gated Boltzmann machines. In *ICML*, 217–225.

Srivastava, N., and Salakhutdinov, R. 2013. Discriminative transfer learning with tree-based priors. In *NIPS*.

Sun, Y.; Chen, Y.; Wang, X.; and Tang, X. 2014. Deep learning face representation by joint identification-verification. In *NIPS*.

Sun, Y.; Wang, X.; and Tang, X. 2014. Deep learning face representation from predicting 10,000 classes. In *CVPR*, 1891–1898.

Tang, Y. 2013. Deep learning using linear support vector machines. In *Workshop on Challenges in Representation Learning, ICML*.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR* 3371–3408.

Weston, J.; Ratle, F.; and Collobert, R. 2008. Deep learning via semi-supervised embedding. In *ICML*.

Zeiler, M., and Fergus, R. 2014. Visualizing and understanding convolutional neural networks. In *ECCV*.

Zhang, Z.; Luo, P.; Loy, C. C.; and Tang, X. 2014. Facial landmark detection by deep multi-task learning. In *ECCV*. 94–108.

Zhu, Z.; Luo, P.; Wang, X.; and Tang, X. 2013. Deep learning identity-preserving face space. In *ICCV*.

Zhu, Z.; Luo, P.; Wang, X.; and Tang, X. 2014. Multi-view perceptron: a deep model for learning face identity and view representations. In *NIPS*.

Zou, W. Y.; Ng, A. Y.; Zhu, S.; and Yu, K. 2012. Deep learning of invariant features via simulated fixations in video. In *NIPS*.