File Type Recognition and Error Correction for NVMs with Deep Learning

Pulakesh Upadhyaya and Anxiao (Andrew) Jiang
Computer Science and Engineering Department, Texas A&M University
pulakesh@tamu.edu, ajiang@cse.tamu.edu

I. Introduction

Storage systems have a strong need for substantially improving their error correction capabilities, especially for long-term storage where the accumulating errors can exceed the decoding threshold of error-correcting codes (ECCs). For non-volatile memories (NVM), this is especially important because noise mechanisms such as charge leakage, read/write disturbs, and cell-quality degradation due to P/E cycling result in accumulating errors, especially in long-term storage. In this work, a new scheme is presented that uses deep learning to perform soft decoding for noisy files based on their natural redundancy (which refers to the redundancy in uncompressed or imperfectly compressed data). The soft decoding result is then combined with ECCs for substantially better error correction performance.

The system we propose is illustrated in Fig. 1. Each file is partitioned into k bit segments, and each file segment is encoded by a systematic (n, k) ECC into a codeword of n bits. Then each ECC codeword passes through a noisy channel, which is a model for the errors in the device. The NR decoding has two processes: first, a *Deep Neural Network* (DNN) uses the k noisy information bits to recognize the file type (e.g. HTML, LaTeX, PDF or JPEG) of the file segment. Then, a second DNN for that file type performs soft decoding on the k noisy information bits based on natural redundancy, and outputs k probabilities, for $i = 1, 2, \cdots, k$, the i-th output is the probability for the i-th information bit to be 1. The k probabilities are given as additional information to the ECC's decoder. The ECC decoder then performs its decoding and outputs the final result.

Different from previous works [2], [3], [4] that used NR for error correction, the scheme is representation-oblivious: it requires no prior knowledge on how data are represented (e.g., mapped from symbols to bits, compressed, and combined with meta data) in different types of files, which makes the solution more convenient to use for storage systems. Experimental results confirm that the scheme can substantially improve the ability to recover data for different types of files even when the bit error rates in the files have significantly exceeded the decoding threshold of the ECC. In the following we introduce the key steps in the scheme. First, for data of unknown file types, we use deep learning to recognize their file types. Then, we design a novel DNN architecture based on portfolio-theory that can perform soft decoding for noisy bits using NR. Finally, we combine the soft-decoding results with the ECC decoder for enhanced error correction performance. For full details of the paper, please refer to [5].

TABLE I
Bit error rate (BER) vs Test Accuracy(Acc.) for File Type Recognition (FTR).

Bit Error Rate (BER)	Overall Acc.	HTML Acc.	JPEG Acc.	PDF Acc.	LaTeX Acc.
0.2%	99.61%	99.98%	99.52%	99.17%	99.77%
0.4%	99.69%	99.96%	99.60%	99.25%	99.96%
0.8%	99.69%	99.98%	99.50%	99.35%	99.92%
1.6%	99.58%	99.96%	99.60%	98.83%	99.92%

II. File Type Recognition using Deep Learning

The first step in NR decoding is to design a DNN for file type recognition (FTR) from noisy file segments. The DNN takes a noisy file segment of k bits, $(y_1, y_2, \cdots, y_k)$, as input, and outputs one of T file types (e.g., HTML, LaTeX, PDF or JPEG). The errors in the file segment come from a binary-symmetric channel (BSC) of bit-error rate (BER) p.

We use a Convolutional Neural Network (CNN) that takes the k bits of a noisy file segment as input. In our experiments, we let $k = 4095$. The CNN has T outputs that correspond to the T possible file types, namely, the T classification results. We consider four file types: HTML, LaTeX, PDF and JPEG. So $T = 4$. There are $L = 9$ convolution layers $\{C_1, C_2, ...C_L\}$ where each layer C_d (for $d = 1, 2, \cdots, L$) is followed by a max pooling layer M_d. The last max pooling layer M_L is followed by a dense layer D. For each of the $T = 4$ file types, 24,000, 4,000 and 4,800 noiseless file segments are used to generate training, validation, and test data respectively. The $(4376, 4095)$ LDPC code used in our experiments can correct errors of BER up to 0.2% *by itself*. Therefore, we have selected the target BER p with substantially higher values, ranging from 0.2% to 1.6%. The CNN's performance, as shown in Table I compares favorably with existing results on FTR [5].

III. Portfolio Theory based Soft Decoding

After FTR, the next step in NR decoding is to design a DNN to perform soft decoding for noisy file segments of the recognized file type. The input to the DNN is a noisy file segment of k bits $Y = (y_1, y_2, \cdots, y_k)$. As before, the errors in the noisy file segment come from a binary-symmetric channel (BSC) of bit-error rate (BER) p. The output of the DNN is a vector $Q = (q_1, q_2, \cdots, q_k)$, where for $1 \le i \le k$, $q_i \in [0, 1]$ represents the DNN's belief that for the i-th bit in the file segment, the probability that its correct value should be 1 is q_i. In other words, if we use $X = (x_1, x_2, \cdots, x_k)$ to denote an error-free file segment,

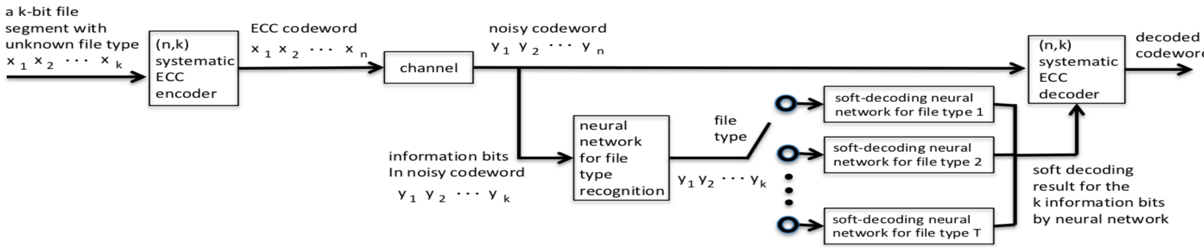

Fig. 1. Encoding and decoding scheme for a noisy file segment of an initially unknown file type.

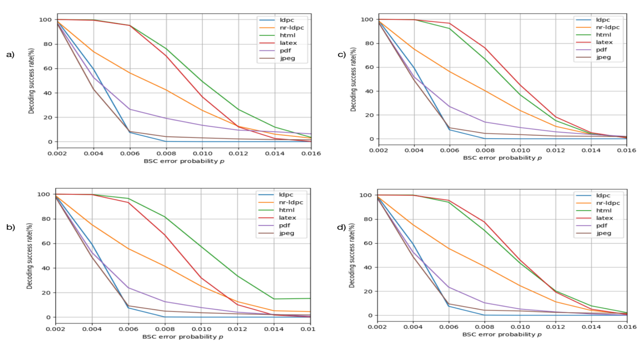

Fig. 2. Decoding success rate vs bit error rate for (a) p_{DNN} = 1.0% , (b) p_{DNN} = 1.2%, (c) p_{DNN} = 1.4%, (d) p_{DNN} = 1.6%

and let it pass through a BSC of BER p to obtain a noisy file segment $Y = (y_1, y_2, \cdots, y_k)$, then q_i is the DNN's estimation for $Pr\{x_i = 1 \mid Y, p\}$.

For soft decoding, our DNN needs to predict probabilities for bits with high accuracy. Many applications of deep learning use *sigmoid* or *softmax* functions, which only mimic probabilities to some very limited extent, and their accuracy is far from sufficient for the task of error correction. Therefore, we present below a new technique based on portfolio theory that trains a DNN to learn *soft decoding* with high accuracy. Define R_i for the ith bit as follows: if $x_i = 1$, we let $R_i = \log_2 q_i$; otherwise, we let $R_i = \log_2(1 - q_i)$. Define the average loss function for the network as $\Lambda = -\frac{1}{k}\sum_{i=1}^{k} R_i$, which is a form of cross entropy. By portfolio theory [1], it can be proved that when the k information bits are independent, the loss function's value is minimized when the DNN's predicted probabilities equal the true probabilities. Experimentally, we show that it can work well even if the bits are correlated. For details of the DNN architecture, please refer to [5].

IV. Error Correction for Noisy File Segments

In this section, we combine the soft decoding output of the DNN – which was presented in the previous section – with an LDPC code for enhanced error correction performance. We adopt a robust scheme here: the DNNs for file-type recognition and for soft decoding have been trained with a constant BER p_{DNN} , but they are tested for a wide range of BERs p for the BSC channel. (For example, the DNNs may be trained just for $p_{DNN} = 1.2\%$, but are tested for any BER p from 0.2% to 1.6% in experiments here.) We choose this robust scheme because when DNNs are designed, the future BER in data can be highly unpredictable. That is exactly why errors may exceed ECC's thresholds for longterm storage, and why NR can become useful for error correction.

We measure the performance of our error correction scheme by the percentage of codewords that are decoded correctly, which we call Decoding Success Rate. (Let us call the scheme the NR-LDPC decoder, since it combines decoding based on natural redundancy and the systematic (n, k) LDPC code). For $i = 1, 2, \cdots, k$, the i-th output of the DNN p_i represents the estimated probability for the i-th information bit to be 1. Those k probabilities can be readily turned into LLRs (log-likelihood ratios) for the information bits using the formula $LLR_i^{DNN} = \log(\frac{1-p_i}{p_i})$. For $i = 1, 2, \cdots, n$, let $LLR_i^{channel}$ be the LLR for the i-th codeword bit obtained from the channel transition probability. We let the *initial LLR* for the i-th codeword bit be $LLR_i^{int} = LLR_i^{channel} + LLR_i^{DNN}$ for information bits $(1 \le i \le k)$, and $LLR_i^{int} = LLR_i^{channel}$ for parity bits $(k+1 \le i \le n)$. We then do belief-propagation (BP) decoding using the initial LLRs, and get the final result. The experimental results are presented in Fig. 2. Here the x-axis is the channel error probability p, and the y-axis is the decoding success rate. The network is trained for a target error probability p_{DNN}. The curve for "ldpc" is the performance of the LDPC decoder alone, and the curve for "nr-lpdc" is for the NR-LDPC decoder. The figure also shows the performance for each of the 4 file types. The NR-LDPC decoder achieves *significantly better* performance. For example, at $p = 0.6\%$, the decoding success rate of the NR-LDPC decoder is almost 4 times as high as the LDPC decoder. For details, please refer to [5].

ACKNOWLEDGMENT: This work was supported in part by NSF Grant CCF-1718886.

References

[1] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd Edition, Wiley 2006.
[2] A. Jiang, P. Upadhyaya, E. F. Haratsch and J. Bruck, " Error Correction by Natural Redundancy for Long Term Storage", in *Proc. Non- Volatile Memories Workshop (NVMW)*, La Jolla, California, March 2017.
[3] J. Luo, Q. Huang, S. Wang and Z. Wang, "Error Control Coding Combined with Content Recognition", in *Proc. 8th International Conference on Wireless Communications and Signal Processing*, pp. 1-5, 2016
[4] P. Upadhyaya and A. Jiang, "LDPC Decoding with Natural Redundancy", in *Proc. Non- Volatile Memories Workshop (NVMW)*, La Jolla, California, March 2017.
[5] P. Upadhyaya and A. Jiang, "Representation-Oblivious Error Correction by Natural Redundancy," in *arXiv:1811.04032 [cs.IT]*, 2018.