DEEP LEARNING FOR JOINT SOURCE-CHANNEL CODING OF TEXT

Nariman Farsad, Milind Rao*, and Andrea Goldsmith*

Electrical Engineering, Stanford University, Stanford, CA

ABSTRACT

We consider the problem of joint source and channel coding of structured data such as natural language over a noisy channel. The typical approach inspired by information theory to this problem involves performing source coding to first compress the text and then channel coding to add robustness while transmitting across the channel; this approach is optimal with arbitrarily large block lengths for discrete memoryless channels. Given documents of finite length and limitations on the length of the encoding, we achieve lower word error rates by developing a deep learning based encoder and decoder. While the information theoretic approach would minimize bit error rates, our approach preserves semantic information of sentences by first embedding sentences in a semantic space where sentences closer in meaning are located closer together, and then performing joint source and channel coding on these embeddings.

***Index Terms*—** deep learning, natural language processing, Joint source-channel coding

1. INTRODUCTION

In digital communications, data transmission typically entails source coding and channel coding. In source coding the data is mapped to a sequence of symbols where the sequence length is optimized. In channel coding redundant symbols are systematically added to this sequence to detect or correct at the receiver the errors that may be introduced during data transfer. One of the consequences of the source-channel coding theorem by Shannon [1] is that source and channel codes can be designed separately, with no loss in optimality, for memoryless and ergodic channels when infinite block length codes are used. This is known as the separation theorem, and can be extended to a larger class of channels [2].

Optimality of separation in Shannon's theorem assumes no constraint on the complexity of the source and channel code design. However, in practice, having very large block lengths may not be possible due to complexity and delay constraints. Therefore, many communication systems may benefit from designing the source/channel codes jointly. Some examples demonstrating this benefit include: wireless channels [3], video transmission over noisy channels [4], and image transmission over noisy channels [5, 6].

In this work, we consider design of joint source-channel coding for text data with constrained code lengths. Particularly, our ultimate goal is to design a messaging service where sentences are transmitted over an erasure channel. The erasure channel is used here since it can model a broad class of channels where errors are detected but not corrected. One example is timing channels, where information is encoded on the time of release of packets [7]. Our proposed coding technique can be used in this channel to create a covert messaging service over packet-switched networks [8, 9, 10, 11]. In our messaging service, instead of recovering the exact sentence at the receiver, we are interested in recovering the semantic information such as facts or imperatives of the sentence. Therefore, any sentence that conveys the information in the originally transmitted sentence would be considered as an error free output by the decoder even if it differed from the exact sentence. For example, the phrase "the car stopped" and "the automobile stopped" convey the same information.

One of the first works that considered joint source-channel coding using neural networks is [12], where simple neural network architectures were used as encoder and decoder for Gauss-Markov sources over additive white Gaussian noise channel. There are also a number of works that use neural networks for compression without a noisy channel (i.e., only source coding). In particular, in [13, 14] image compression algorithms are developed using RNNs, which outperformed other image compression techniques. Sentence and document encoding is proposed in [15] using neural autoencoders.

Contributions: Inspired by recent success of deep learning in natural language processing for tasks such as machine translation [16], we develop a neural network architecture for joint source-channel coding of text. Our model uses a recurrent neural network (RNN) encoder, a binarization layer, the channel layer, and a decoder based on RNNs. We demonstrate that using this architecture, it is possible to train a joint source-channel encoder and decoder, where the decoder may output a different sentence that preserves its semantic information content.

We compare the performance of our deep learning encoder and decoder with separate source and channel coding design[1]. Since the channel considered here is the erasure channel, we use Reed-Solomon codes for channel coding. For source coding, we consider three different techniques: a universal source coding scheme, a Huffman coding, and 5-bit character encoding. We demonstrate that the proposed deep learning encoder and decoder outperform the traditional approach in term of word error rate (WER), when the bit budget per sentence encoding is low. Moreover, in many cases, although some word may be replaced, dropped, or added to the sentence by the deep learning decoder, the semantic information in the sentence is preserved in a qualitative sense.

2. PROBLEM DESCRIPTION

In this section, we define our system model associated with transmitting sentences from a transmitter to a receiver using limited number of bits.

Let $\mathcal{V}$ be the set of all the words in the vocabulary and let $\mathbf{s} = [w_1, w_2, \cdots, w_m]$ be the sentence to be transmitted where $w_i \in \mathcal{V}$ is the i^{th} word in the sentence. The transmitter converts the sentence into a sequence of bits prior to transmission using source and channel coding. Let $\mathbf{b} = \varphi_\ell(\mathbf{s})$ be a binary vector of length-ℓ, where φ_ℓ is the function representing the combined effect of the source and channel encoder. Let $\mathbf{o}$ be the vector of observations at the receiver corresponding to each of the ℓ-bits in the transmission. Note that $\mathbf{o}$ does not necessarily need to be a binary vector, and it could be a vector of real or natural numbers depending on the channel considered. Let the combined effect of the source and channel decoder function be given by $\nu_\ell(\mathbf{o})$. Then $\hat{\mathbf{s}} = [\hat{w}_1, \hat{w}_2, \cdots, \hat{w}_{m'}] = \nu_\ell(\mathbf{o})$, where $\hat{\mathbf{s}}$ is the recovered sentence. The traditional approach to designing the source and channel coding schemes is to minimize the word error rate while also minimizing the number of transmission bits. However, jointly optimizing the source coding and the channel

*The authors contributed equally.

This work was funded by the TI Stanford Graduate Fellowship, NSF under CPS Synergy grant 1330081, and NSF Center for Science of Information grant NSF-CCF-0939370.

[1]To the best of our knowledge there are no known joint source-channel coding schemes for text data over erasure channels.

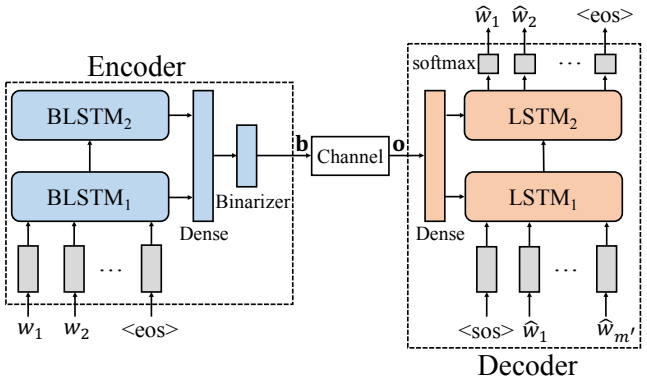


Fig. 1: The encoder-decoder architecture.

coding schemes is a difficult problem and therefore, in practice, they are treated separately.

The problem considered in this work is designing a joint source-channel coding scheme that preserves the meaning between the transmitted sentence $\mathbf{s}$ and the recovered sentence $\hat{\mathbf{s}}$, while the two sentences may have different words and different lengths.

3. DEEP LEARNING ALGORITHM

Our work is motivated by the recent success of the sequence-to-sequence learning framework [17] in different tasks such as machine translation [16, 18]. Our system, which is shown in Fig. 1 has three components: the encoder, the channel, and the decoder. The encoder takes as an input a sentence $\mathbf{s}$, concatenated with the special end of sentence word <eos>, and outputs a bit vector $\mathbf{b}$ of length ℓ. The channel takes an input bit vector $\mathbf{b}$ and produces an output vector $\mathbf{o}$. The effects of this module is random. The channel output $\mathbf{o}$ is the input to the decoder, and the output of the decoder is the estimated sentence $\hat{\mathbf{s}}$. We now describe each of these modules in detail.

3.1. The Encoder

The first step in the encoder uses an embedding vector to represent each word in the vocabulary. In this work, we initialize our embedding vectors using Glove [19]. Let $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_m, \mathbf{e}_{\text{eos}}]$ be the $m+1$ embeddings of words in the sentence. In the second step, the embedded words are the inputs to a stacked bidirectional long short term memory (BLSTM) network [20]. The LSTM cell used in this work is similar to that used in [21]. The j^{th} BLSTM stack is represented by

$$\mathbf{C}_j, \mathbf{H}_j = \text{BLSTM}_j(\mathbf{H}_{j-1}), \tag{1}$$

where $\mathbf{C}_j$ is the cell state matrix and $\mathbf{H}_j$ is output matrix. Each column of $\mathbf{C}_j$ and $\mathbf{H}_j$ represents the cell state vector at each time step, and $\mathbf{H}_0 = \mathbf{E}$. Fig. 1 shows an encoder with two stacked BLSTM layers.

Let k be the total numbers of BLSTM stacks. We concatenate the outputs at the last step and similarly the cell states at the last step of each layer using

$$\mathbf{h} = \mathbf{H}_1[m+1] \oplus \mathbf{H}_2[m+1] \oplus \cdots \oplus \mathbf{H}_k[m+1], \tag{2}$$

$$\mathbf{c} = \mathbf{C}_1[m+1] \oplus \mathbf{C}_2[m+1] \oplus \cdots \oplus \mathbf{C}_k[m+1], \tag{3}$$

where $\oplus$ is the concatenation operator, and $\mathbf{H}_j[m+1]$ and $\mathbf{C}_j[m+1]$ are the $m+1$ column (i.e., the last step) of respectively, the outputs and cell states of the j^{th} stack.

To convert $\mathbf{h}$ and $\mathbf{c}$ into binary vectors of length $\ell/2$ we use the same technique as in [22, 23, 13]. The first step in this process uses two fully connected layers

$$\mathbf{h}^* = \tanh(\mathbf{W}_h \mathbf{h} + \mathbf{a}_h), \tag{4}$$

$$\mathbf{c}^* = \tanh(\mathbf{W}_c \mathbf{c} + \mathbf{a}_c), \tag{5}$$

where $\mathbf{W}_h$ and $\mathbf{W}_c$ are weight matrices each with $\ell/2$ rows, and $\mathbf{a}_h$ and $\mathbf{a}_c$ are the bias vectors. Note that although here we use one fully connected layer, it would be possible to use multiple layers where the size of $\mathbf{h}$ and $\mathbf{c}$ is increased or decreased to $\ell/2$ in multiple steps. However, the last layer's activation function must alway be a tanh, to keep the output value in the interval $[-1, 1]$.

The second step maps the the values in $\mathbf{h}^*$ and $\mathbf{c}^*$ from the interval $[-1, 1]$ to binary values $\{-1, 1\}$. Define a stochastic binarization function as

$$\beta(x) = x + Z_x, \tag{6}$$

where $Z_x \in \{1-x, -x-1\}$ is a random variable distributed according to $P(Z_x = 1-x) = \frac{1+x}{2}$ and $P(Z_x = -x-1) = \frac{1-x}{2}$. Then final binarization step during training is

$$\mathbf{b} = \beta(\mathbf{h}^*) \oplus \beta(\mathbf{c}^*) \tag{7}$$

for the forward pass. During the back-propagation step of the training, the derivative with respect to the expectation $\mathbb{E}[\beta(x)] = x$ is used [24]. Therefore, the gradients pass through the β function unchanged.

After training the network using β, during deployment or testing the stochastic function $\beta(x)$, is replaced with the deterministic function $2u(x) - 1$, where $u(x)$ is the unit step function.

3.2. The Channel

To allow for end-to-end training of the encoder and the decoder, the channel must allow for back-propagation. Fortunately, some communication channels can be formulated using neural network layers. This includes the additive Gaussian noise channel, multiplicative Gaussian noise channel and the erasure channel. In this work, we consider the erasure channel as it could model packets of data being dropped in a packet switched networks, or wireless channels with deep fades or burst errors.

The erasure channel can be represented by a dropout layer [25],

$$\mathbf{o} = \text{dropout}(\mathbf{b}, p_d), \tag{8}$$

where $\mathbf{o}$ is the vector of observations at the receiver, and p_d is the probability that a bit is dropped. The elements of $\mathbf{o}$ are in $\{-1, 0, 1\}$, where 0 indicates erasure (i.e., a dropped bit). Every bit in $\mathbf{b}$ may be dropped independent of other bits with probability p_d.

3.3. The Decoder

At the receiver we use a stack of LSTMs for decoding. The observation vector $\mathbf{o}$ is input to the decoder. Let $\ominus(\mathbf{x}, v)$ be the inverse of the concatenation operator, where the vector $\mathbf{x}$ is broken into v vectors of equal length. Then we have

$$\mathbf{h}', \mathbf{c}' = \ominus(\mathbf{o}, 2), \tag{9}$$

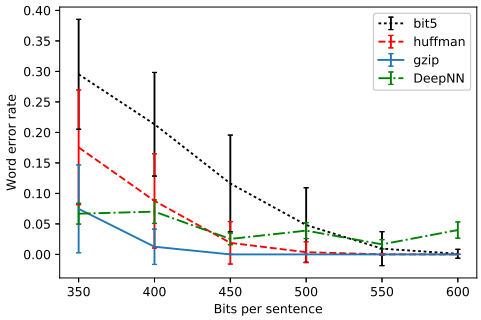
which contribute to the initial $\mathbf{h}_0^{(j)}$ state and $\mathbf{c}_0^{(j)}$ state of the j^{th} LSTM stack. Particularly, these initial states are given by

$$\mathbf{h}_0^{(j)} = \tanh\left(\mathbf{W}_h^{(j)} \mathbf{h}' + \mathbf{a}_h^{(j)}\right), \tag{10}$$

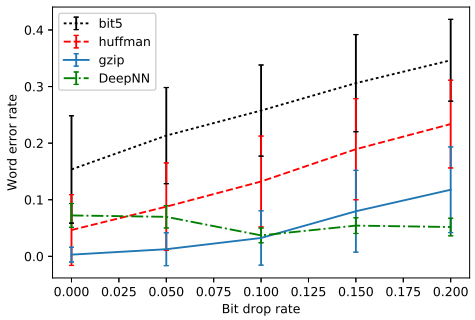
$$\mathbf{c}_0^{(j)} = \mathbf{W}_c^{(j)} \mathbf{c}' + \mathbf{a}_c^{(j)}, \tag{11}$$

where $\mathbf{W}_h^{(j)}$ and $\mathbf{W}_c^{(j)}$ are the weight matrix, and $\mathbf{a}_h^{(j)}$ and $\mathbf{a}_c^{(j)}$ are the bias vectors.

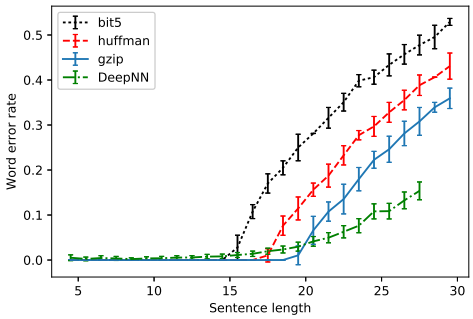
The first input to the LSTM stack is the embedding vector for a special start of the sentence symbol <sos>. Note that after the first word $\hat{w}_1$ is estimated, its embedding vector will be used as the input for the next time step. To speed up the training, during the first few epochs, with probability 1 we use the correct word w_i as the input for the $i+1$ time step at the decoder; we gradually anneal the probability with which we replace the correct word w_i with the estimated word $\hat{w}_i$. During deployment and testing we always use the estimated words and the beam search algorithm to find the most likely sequences of words [26, 16].



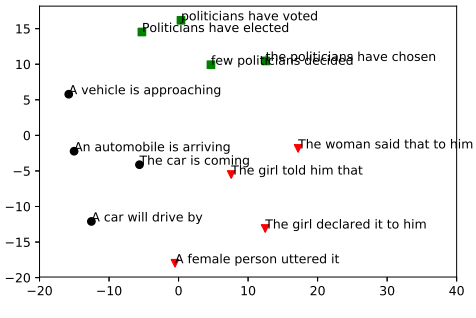
(a) Word error as bits per sentence changes for 0.05 bit erasure probability.



(b) Word error as erasure or bit-drop rate increases for 400 bit encoding.



(c) Effect of sentences of different sizes with 400 bit encoding, 0.05 drop rate.



(d) Sample embeddings mapped to two dimensions using manifold dimensions with hamming distances between codes.

Fig. 2: Performance plots.

Punctuation error	TX: efficiency – what efficiency ? RX: efficiency , what efficiency ?
Rephrasing	TX: tourism serves as a source of income to totalitarian regimes . RX: tourism has become a source of income to totalitarian regimes .
Rephrasing	TX: a few wealthy individuals compared with millions living in hunger . RX: a few wealthy individuals face with millions living in hunger .
Tense Error	TX: a communist country riding roughshod over human rights . RX: a communist country rides roughshod over human rights .
An inexplicable error	TX: i listened to colleagues who mentioned bicycles . RX: i listened to colleagues who mentioned goebbels .
Long sentence 1	TX: there is one salient fact running through these data : the citizens want more information and have chosen television as the best means to receive that information . RX: there is one glaring weaknesses , by the communication : the citizens want more information and hold ' television as the means to receive this information .
Long sentence 2	TX: i hope we will be able to provide part - funding for a renovation programme for energy efficiency as a result of this decision of the eu . RX: i hope we will be able to provide for funding for the renovation programme for energy efficiency as a result of decision by the eu .

Table 1: Sample sentences which were transmitted and received using the deep learning approach.

4. RESULTS

In this section, we compare the deep learning approach with traditional information theoretic baselines for bit erasure channels. The source code used to generate the results is available on github[2].

4.1. The Dataset

We work with the proceedings of the European Parliament [27]. This is a large parallel corpus that is frequently used in statistical machine translation. The English version has around 2.2 million sentences and 53 million words.

We crawl through the corpus to extract the most common words which we call our vocabulary. We pre-process the dataset by selecting sentences of lengths 4-30 where less then 20% of the words in the sentences are unknown words (i.e., they are outside of the selected vocabulary). The corpus is split into a training and test data set, where the training set has more that 1.2 million sentences and the test data set has more than 200 thousand sentences.

[2] https://github.com/milindmrao/nlp_comm

4.2. Deep Learning Approach

We initialize 200-dimensional word embeddings using the Glove pre-trained embeddings [19] for words in our vocabulary as well as a few special words (unknowns, padding, start and end symbols). We batch the sentences from the corpus based on their sentence lengths to increase efficiency of computation - i.e. sentences of similar length are fed in batches of size 128.

Two layered BLSTM of dimension 256 with peepholes are used for the encoder followed by a dense layer thats brings the dimension of the resultant state to the required bit budget. The decoder has two layers of LSTM cells each with the dimensions 512 with peephole connections. Note that one disadvantage of the deep learning approach is the use of a fixed number of bits for encoding all sentences of different lengths.

4.3. Information Theoretic Baselines

We implement separate source and channel coding which we know is optimal in the asymptote of arbitrarily large block lengths and delays. The source coding is done using three approaches:

1. Universal compressors: We use gzip which combines a Lempel-Ziv universal compression [28] scheme with Huffman coding. This method works universally with all kinds of data and theoretically reaches the entropy limit of compression in the asymptote. However, since this technique does not work well for single sentences, we improve its performance by jointly compressing sentences in batches of size 32 or more. Note that this will give this technique an unfair advantage since it will no longer perform source coding on single sentences.
2. Huffman coding: To allow for single sentence source coding, we use Huffman coding on characters in the sentence. Using the training corpus, we compute character frequencies, which are then used to generated the Huffman codebook.
3. Fixed length character encoding: In this approach, we use a fixed 5-bit encoding for characters (the corpus is converted to lower case) and some special symbols. Decoding gzip and Huffman codes when there are errors or corruptions in the output of the channel decoder is not trivial. However, this baseline with 5-bit encoding can be decoded.

After source encoding using the above approaches, we use a Reed-Solomon code [29] that can correct up to the expected number of erasures. In the comparison, we assume the channel code can exactly compensate for erasures that occur. This assumption favors information theoretic baselines as we can expect the number of bit erasures to be larger than the expected number with high probability. If this occurs, the channel decoding process will have errors and this may result in irredeemable corruption for decoding the source codes (gzip or huffman).

Finally, we compare performance by using a fixed bit budget per sentence. However, these schemes inherently produce embeddings of different lengths. If the encoding of a sentence exceeds the bit budget, we re-encode the sentence without its last word (resulting in a word error). We repeat the procedure until the encoding is within the bit limit.

4.4. Performance

There is no better metric than a human judge to establish the similarity between sentences. As a proxy, we measure performance of the deep learning approach as well as the baselines using the edit distance or the Levenshtein distance. This metric is commonly used to measure the dissimilarity of two strings. It is computed using a recursive procedure that establishes the minimum number of word insertion, deletion, or substitution operations that would transform one sentence to another. The edit distance normalized by the length of the sentence is what we refer to as the word error rate. Word error rate is commonly used to evaluate performance in speech recognition and machine translation [30, 31]. A downside of the metric is that it cannot capture the effect of synonyms or other aspects of semantic similarity.

In Fig. 2a, we study the impact of the bit budget or the number of bits per sentence on the word error rate when we have a bit erasure probability of 0.05. Among the traditional baselines, gzip outperforms Huffman codes, and Huffman codes outperform the fixed length encoding. All three approaches result in no error if the bit allocation exceeds the number of bits required. This is because we assume the Reed-Solomon code compensates for all channel erasures. We observe that the deep learning approach is most competitive with limited bit allocations. As we enter the regime of excessive redundancy, the word error rate continually falls.

In Fig. 2b, we look at the impact of the channel on word error rates when we have a bit allocation of 400 bits per sentence. Between the traditional baselines, we observe again that gzip is optimal as it operates on large batches followed by Huffman codes. 400 bits is not enough to completely encode sentences even when the channel is lossless. We make the observation again that in stressed environments (low bit allocations for large bit erasure rates), the deep learning approach outperforms the baselines.

What Fig. 2a and Fig. 2b hide is the impact of varying sentence lengths. If we consider a batch of sentences in random order from the corpus, we will have both large and short sentences. Traditional baselines can allot large encodings to long sentences and short encodings to others leading to an averaged bit allocation that may be short with few errors. However, the deep learning approach has the same bit allocation for sentences regardless of their length. We can improve the performance of the deep learning approach here by varying the length of the embedding based on the sentence length.

Fig. 2c illustrates this very clearly. In this case, instead of having batches with sentences of different lengths, we use homogeneous batches to show the impact of the sentence lengths on word error rates (bit allocation 400, bit erasure rate 0.05). For short sentences, we are in the excess bit allocation regime. As the sentence length increases beyond 20, the deep learning approach significantly outperforms baselines. Another aspect to consider is that word errors of the deep learning approach may not be word errors - that may include substitutions of words using synonyms or rephrasing which does not change the meaning of the word.

4.5. Properties of the encoding

The deep learning approach results in a lossy compression of text. It is able to do this by encoding a semantic embedding of the sentence. We can watch this in action in Fig. 2d. Here, we compute the embeddings of a few sentences, groups of which are thematically linked. One group of sentences is about a girl saying something to a man, another is about a car driving and the last is about politicians voting. We then find the Hamming distance between the embeddings and use this dissimilarity matrix and multidimensional scaling approaches [32] to view it in two dimensions. Sentences that express the same idea have embeddings that are close together in Hamming distance. We do not see such behavior in information theoretic baselines which do not consider the fact that it is text with semantic information that they are encoding.

A few representative errors are shown in Table 1.

5. CONCLUSION

We considered the problem of joint source-channel coding of text data using deep learning techniques from natural language processing. For example, in many applications, recovery of the exact transmitted sentence may not be important as long as the main information within the sentence is conveyed. We demonstrated that our proposed joint source-channel coding scheme outperforms separate source and channel coding, especially in scenarios with a small number of bits to describe each sentence.

One drawback of the current algorithm is that it uses a fixed bit length to encode sentences of different length. As part of future work, we investigate how to resolve this issue. With severe bit restrictions per sentence, we will also look at deep learning based summarization to represent information. Joint source-channel coding of other forms of structured data such as images, audio, and video would also be a relevant future direction.

6. REFERENCES

[1] Claude E Shannon and Warren Weaver, *The mathematical theory of communication*, University of Illinois press, 1998.

[2] Sridhar Vembu, Sergio Verdu, and Yossef Steinberg, "The source-channel separation theorem revisited," *IEEE Transactions on Information Theory*, vol. 41, no. 1, pp. 44–54, 1995.

[3] A. Goldsmith, "Joint source/channel coding for wireless channels," in *IEEE Vehicular Technology Conference. Countdown to the Wireless Twenty-First Century*, Jul 1995, vol. 2, pp. 614–618 vol.2.

[4] Fan Zhai, Yiftach Eisenberg, and Aggelos K Katsaggelos, "Joint source-channel coding for video communications," *Handbook of Image and Video Processing*, 2005.

[5] Geoffrey Davis and John Danskin, "Joint source and channel coding for image transmission over lossy packet networks," in *Conf. Wavelet Applications to Digital Image Processing*, 1996, pp. 376–387.

[6] Ozgun Y Bursalioglu, Giuseppe Caire, and Dariush Divsalar, "Joint source-channel coding for deep-space image transmission using rateless codes," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3448–3461, 2013.

[7] Venkat Anantharam and Sergio Verdu, "Bits through queues," *IEEE Transactions on Information Theory*, vol. 42, no. 1, pp. 4–18, 1996.

[8] Brian P Dunn, Matthieu Bloch, and J Nicholas Laneman, "Secure bits through queues," in *IEEE Information Theory Workshop*. IEEE, 2009, pp. 37–41.

[9] Negar Kiyavash, Farinaz Koushanfar, Todd P Coleman, and Mavis Rodrigues, "A timing channel spyware for the csma/ca protocol," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 477–487, 2013.

[10] Pritam Mukherjee and Sennur Ulukus, "Covert bits through queues," in *IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2016, pp. 626–630.

[11] Arnab Kumar Biswas, Dipak Ghosal, and Shishir Nagaraja, "A survey of timing channels and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 50, no. 1, pp. 6, 2017.

[12] Li Rongwei, Wu Lenan, and Guo Dongliang, "Joint source/channel coding modulation based on bp neural networks," in *Proceedings of the International Conference on Neural Networks and Signal Processing*. IEEE, 2003, vol. 1, pp. 156–159.

[13] George Toderici et al., "Variable rate image compression with recurrent neural networks," in *International Conference on Learning Representations*, 2016.

[14] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell, "Full resolution image compression with recurrent neural networks," *arXiv preprint arXiv:1608.05148*, 2016.

[15] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," *arXiv preprint arXiv:1506.01057*, 2015.

[16] Yonghui Wu et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[17] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[18] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[19] Jeffrey Pennington, Richard Socher, and Christopher Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[20] Alex Graves and Jürgen Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[21] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[22] Ronald J Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[23] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[24] Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh, "Techniques for learning binary stochastic feedforward neural networks," *stat*, vol. 1050, pp. 11, 2014.

[25] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[27] Philipp Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT summit*, 2005, vol. 5, pp. 79–86.

[28] Jacob Ziv and Abraham Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.

[29] Irving S Reed and Gustave Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[30] Chris Quirk, Chris Brockett, and William Dolan, "Monolingual machine translation for paraphrase generation," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[31] Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer, "Sentence simplification by monolingual machine translation," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 2012, pp. 1015–1024.

[32] Ingwer Borg and Patrick JF Groenen, *Modern multidimensional scaling: Theory and applications*, Springer Science & Business Media, 2005.