On Deep-learning Based Channel Decoding

Anirudh J- EE16BTECH11013

Vishnu SSN-EE16BTECH11033

Rajeshwar ch-EE16BTECH11003

INTRODUCTION

- The aim of this paper is to one shot-decoding of structural and random codes using deep learning techniques.

Structural codes

Polar codes:

$$G_N = F^{\otimes n}, \qquad F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Where F⊗n denotes the nth Kronecker power ofF. The codewords are now obtained by x=uGN

Hamming codes:

$$x = a \otimes \overset{\text{Generator matrix}}{G} = \underbrace{\boxed{\begin{bmatrix}1 & 0 & 1\end{bmatrix}}}_{k = 3} \otimes \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right] = \underbrace{[\boxed{1 \quad 0 \quad 1}\,\boxed{0}]}_{n = 4}$$

$$x = a \otimes G = \underbrace{\boxed{\begin{bmatrix}1 & 0 & 1\end{bmatrix}}}_{k = 3} \otimes \left[\begin{array}{ccc|cc} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{array}\right] = \underbrace{[\boxed{1 \quad 0 \quad 1}\,\boxed{0\;1}]}_{n = 5}$$

- **Structural codes are easy to learn compared to random codes although Maximum aposterior performance(MAP)bit error rate can be achieved.**
- **Motivation for using Deep learning for decoding is neural networks were able to generalize structural codes well with less training data and epochs.This implies that neural networks were able to learn some kind of decoding algorithms.**
- **This paper introduced a metric called normalized validation error(NVE) in order to further investigate the potential and limitations of Deep learning methods for decoding.**

- Deep learning based channel decoding is doomed by curse of dimensionality i.e,as the size of the code words increases we need more training data for providing generalisation capability for neural networks.
- For a short code length of N=100 and with a code rate =0.5 there are 2^{50} code words which is huge number of data set for fully training a neural network.
- As the length of the codewords increases the computational complexity increases exponentially.
- Thus neural network decoding(NND) is currently not competitive with the state of the art decoding.

- But we can extend this NND for certain codes which has a proper structure by training on less epochs and on small subset of the codewords.

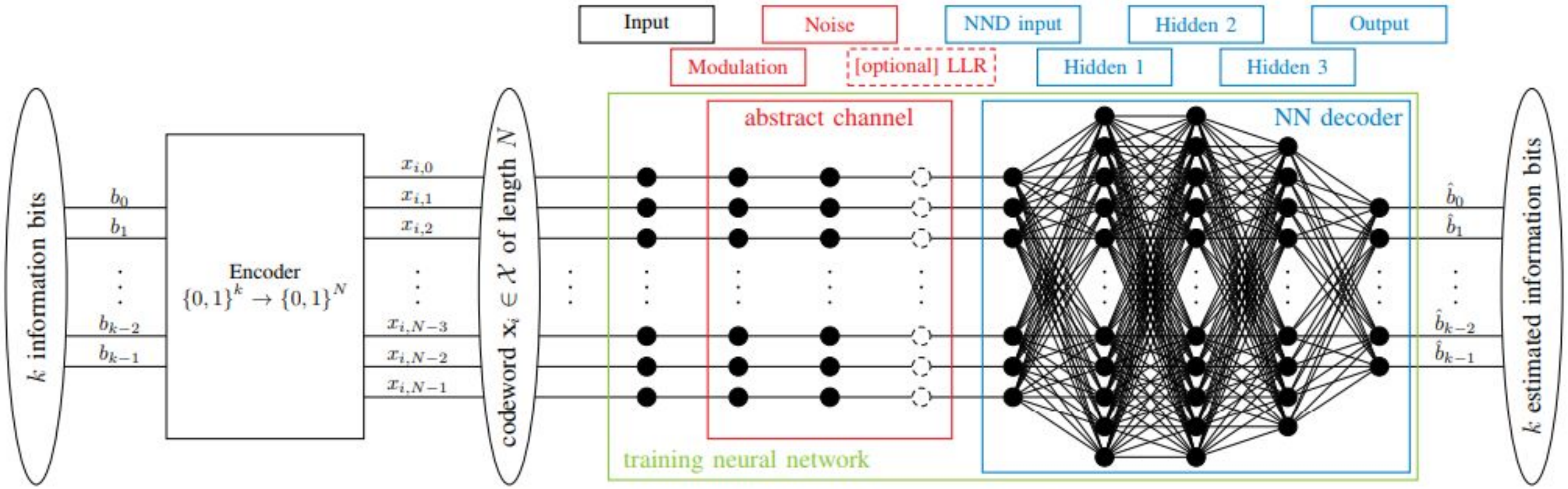


Fig. 1: Deep learning setup for channel coding.

Related works

- In 1943, McCulloch and Pitts published the idea of a NN that models the architecture of the human brain in order to solve problems . But it took about 45 years until the backpropagation algorithm.
- Hopfield nets-One early form of a NN is a Hopfield net .Hopfield nets serve as content-addressable ("associative") memory systems with binary threshold nodes.They are guaranteed to converge to a local minimum and, therefore, may converge to a false pattern (wrong local minimum) rather than the stored pattern (expected local minimum).
- Due to its low storage capacity, Hopfield nets were soon replaced by feed-forward NNs which can learn an appropriate mapping between noisy input patterns and codewords

- A naive implementation of MLD means correlating the received vector of modulated symbols with all possible codewords which makes it infeasible for most practicble codeword lengths, as the decoding complexity is O(2^k) with k denoting the number of information bits in the codeword.
- The parallel computing capabilities of NNs allow us to solve or, at least, approximate the MLD problem in polynomial time . Moreover, the weights of the NN are precomputed during training and the decoding step itself is then relatively simple
- No assumption has to be made about the statistics of the channel noise because the NN is able to learn the mapping or to extract the channel statistics during the learning process

- ❖ In 2006, a new training technique, called layer-by-layer unsupervised pre-training followed by gradient descent finetuning , led to the renaissance of NNs because it made training of NNs with more layers feasible. NNs with many hidden layers are called deep. Nowadays, powerful new hardware such as graphical processing units (GPUs) are available to speed up learning as well as inference

Deep learning for channel coding

❖ **A NN consists of many connected neurons.**

In such a neuron all of its weighted inputs are added up, a bias is optionally added, and the result is propagated through a nonlinear activation function, e.g., a sigmoid function or a rectified linear unit (ReLU), which are respectively defined as

g_sigmoid (z) = 1/(1 + e^{-z}) , g_relu (z) = max {0, z} .

- Then we forward propagate the input through this layers.Then we back propagate the error using gradient descent method and backpropagation algorithm to obtain the desired output for a particular input.
- The goal of training is to enable the NN to find the correct outputs for unseen inputs. This is called generalization.

- In this paper they want to use NND for noisy codewords.
- At the transmitter, k information bits are encoded into a codeword of length N.
- The coded bits are modulated and transmitted over a noisy channel. At the receiver, a noisy version of the codeword is received and the task of the decoder is to recover the corresponding information bits.
- In comparison to iterative decoding, the NN finds its estimate by passing each layer only once.
- This principle enables low-latency implementations, we term it one-shot decoding.

- we deal with manmade signals obtaining labeled training data.
- For the sake of simplicity, binary phase shift keying (BPSK) modulation and an additive white Gaussian noise (AWGN) channel is used.
-
- In order to keep the training set small it is possible to extend the NN with additional layers for modulating and adding noise.
- These additional layers have no trainable parameters, i.e., they perform a certain action such as adding noise and propagate this value only to the node of the next layer with the same index.

- Instead of creating, and thus storing, many noisy versions of the same codeword, working on the noiseless codeword is sufficient.
- Each hidden layer employs a ReLU activation function because it is nonlinear and at the same time very close to linear which helps during optimization.
- Since the output layer represents the information bits, a sigmoid function forces the output neurons to be in between zero and one.

- If the probability is close to the bit of the label, the loss should be incremented only slightly whereas large errors should result in a very large loss.

- Examples for such loss functions are the mean squared error (MSE) and the binary cross-entropy (BCE).

$$L_{\mathrm{MSE}} = \frac{1}{k}\sum_{i}\left(b_i - \hat{b}_i\right)^2$$

$$L_{\mathrm{BCE}} = -\frac{1}{k}\sum_{i}\left[b_i \ln\left(\hat{b}_i\right) + (1 - b_i)\ln\left(1 - \hat{b}_i\right)\right]$$

- There are some alternatives for this setup. First, loglikelihood ratio (LLR) values could be used instead of channel values. For BPSK modulation over an AWGN channel, these are obtained by

$$\mathrm{LLR}\left(y\right) = \ln \frac{P\left(x=0|y\right)}{P\left(x=1|y\right)} = \frac{2}{\sigma^2} y$$

  where σ^2 is the noise power and y the received channel value.

- This processing step can be also implemented as an additional layer without any trainable parameters. Note, that the noise variance must be known in this case and provided as an additional input to the NN.
- The paper used Keras as a convenient high level abstraction.
- It allows to quickly deploy NNs from a very abstract point of view in the Python programming language that hides away a lot of the underlying complexity.

LEARN TO DECODE

- ❖ we will consider two different code families: random codes and structured codes, namely polar codes and random codes.Both have codeword length N = 16 and code rate r = 0.5.
- ❖ random codes are generated by randomly picking codewords from the codeword space with a Hamming distance larger than two, the generator matrix of polar codes of block size N = 2^n is given by

$$\mathbf{G}_N = \mathbf{F}^{\otimes n}, \quad \mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

where F ⊗n denotes the nth Kronecker power of F. The codewords are now obtained by x = uGN , where u contains k information bits and N − k frozen positions.

PARAMETERS OF NEURAL NETWORK DECODER

❖ Our starting point is a NN as described in the previous slides. We introduce the notation 128-64-32 which describes the design of the NN decoder employing three hidden layers with 128, 64, and 32 nodes, respectively.

❖ Other design parameters which can impact the performance :

* Best training signal-to-noise-ratio (SNR).
* Number of training samples.
* Type of loss function.
* Number of layers and nodes in Neural Network
* Type of regularization
* LLR Channel output or Direct Channel Output.

NORMALIZED VALIDATION ERROR (NVE)

Since the performance of NND depends not only on the SNR of the validation data set (for which the bit error rate (BER) is computed) but also on the SNR of the training data set , we define below a new performance metric, the normalized validation error (NVE). Denote by ρt and ρv the SNR (measured as Eb/N0) of the training and validation data sets, respectively, and let BER(NND)(ρt , ρv) be the BER achieved by a NN trained at ρt on data with ρv. Similarly, let BER(MAP)(ρv) be the BER of MAP decoding at SNR ρv. For a set of S different validation data sets with SNRs ρv,1, . . . , ρv,S, the NVE is defined as:

$$\mathrm{NVE}(\rho_t) = \frac{1}{S}\sum_{s=1}^{S} \frac{\mathrm{BER}_{\mathrm{NND}}(\rho_t, \rho_{v,s})}{\mathrm{BER}_{\mathrm{MAP}}(\rho_{v,s})}.$$

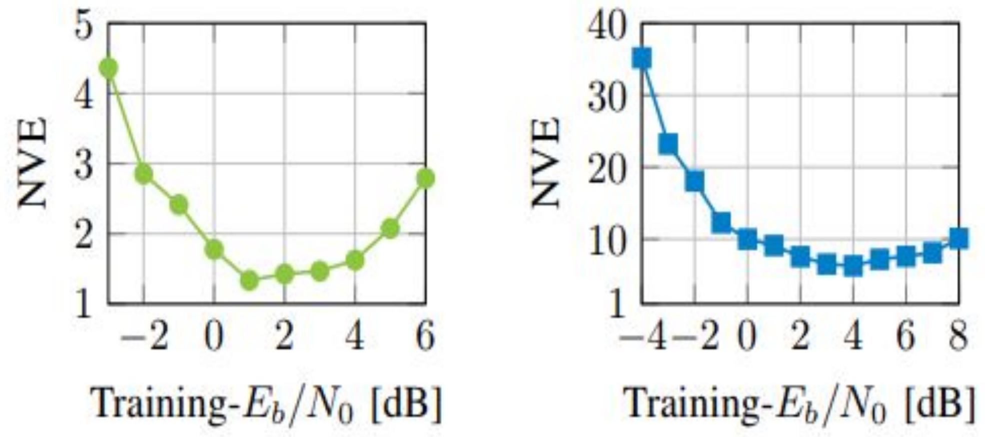
NVE TO MEASURE PERFORMANCE OF NND

- ❖ **The NVE measures how good a NND, trained at a particular SNR, is compared to MAP decoding over a range of different SNRs. Obviously, for NVE = 1, the NN achieves MAP performance, but is generally greater. In the sequel, we compute the NVE over S = 20 different SNR points from 0 dB to 5 dB with a validation set size of 20000 examples for each SNR.**

- ❖ **The training of the NND is done by training the network on a limited set of the data out of the 2^k codewords as it is unnecessary to train the model on the whole data since noise if added to every input and the NND doesn’t see the same input twice.Also good accuracy is acheived by Increasing the number of epochs.**

NVE ON TRAINING SET WITH VARYING SNR

Starting with a NN decoder architecture of 128->64->32 and Mep = 2^{22} learning epochs, we train the NN with datasets of different training SNRs and evaluate the resulting NVE. The result is shown in figure below, from which it can be seen that there is an "optimal" training Eb/N0.

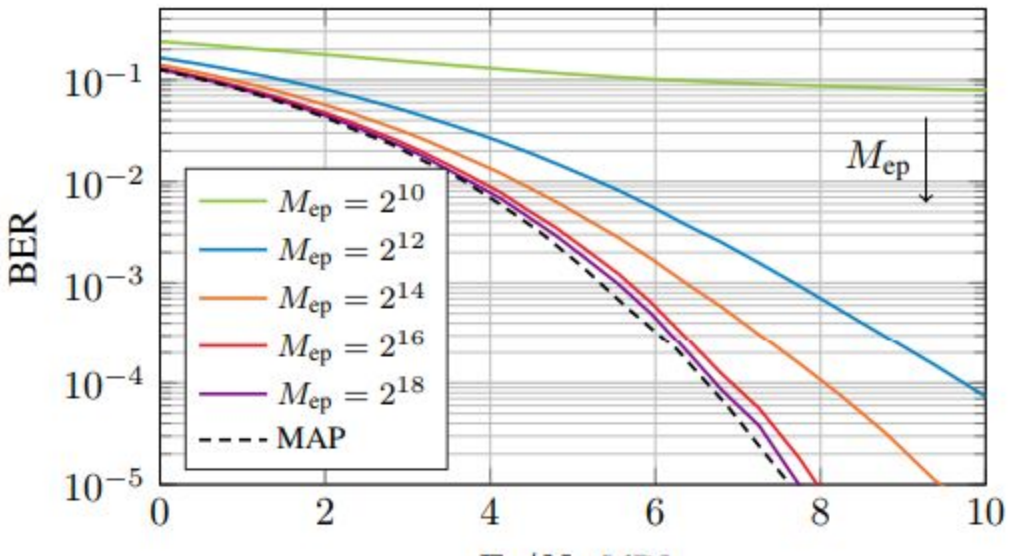


(a) Polar Code (b) Random Code

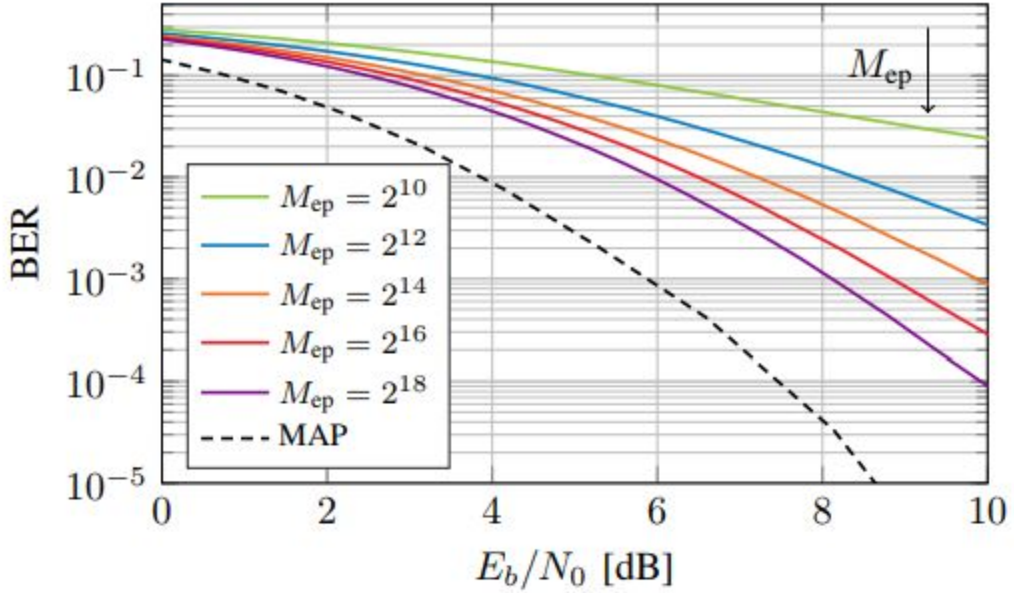
Fig. 2: NVE versus training-E_b/N_0 for 16 bit-length codes for a 128-64-32 NN trained with $M_{\text{ep}} = 2^{16}$ training epochs.

EPOCHS INFLUENCE ON BER OF CODES

- The figures in next two slides shows the BER(Bit Error rate) achieved by a very small NN of dimensions 128-64-32 as a function of the number of training epochs ranging from Mep = 2^{10} , . . . , 2^{18}. For BER simulations, we use 1 million codewords per SNR point. For both code families, the larger the number of training epochs, the closer is the gap between MAP and NND performance.


BER
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
2
4
6
8
10
E_b/N_0 [dB]
M_{ep}
$M_{\text{ep}} = 2^{10}$
$M_{\text{ep}} = 2^{12}$
$M_{\text{ep}} = 2^{14}$
$M_{\text{ep}} = 2^{16}$
$M_{\text{ep}} = 2^{18}$
MAP

(a) Polar Code



(b) Random Code

INFLUENCE OF LLR ON NVE

- We now illustrate the influence of direct channel values versus channel LLR values as decoder input in combination with two loss functions, MSE and BCE. The NVE for all combinations is plotted as a function of the number of training epochs. Such a curve is also called “learning curve” since it shows the process of learning. Although it is usually recommended to normalize the NN inputs to have zero mean and unit variance, we train the NN without any normalization which seems to be sufficient for our setup.

$$\mathrm{LLR}\left(y\right) = \ln \frac{P\left(x = 0|y\right)}{P\left(x = 1|y\right)} = \frac{2}{\sigma^2} y$$

- For a few training epochs, the LLR input improves the learning process; however, this advantage disappears for a larger Mep. The same holds for BCE against MSE. For polar codes with LLR values and BCE the learning appears not to converge for the applied number of epochs. In summary, for training the NN with a large number of training epochs it does not matter if LLR or channel values are used as inputs and which loss function is employed. Moreover, normalization is not required.

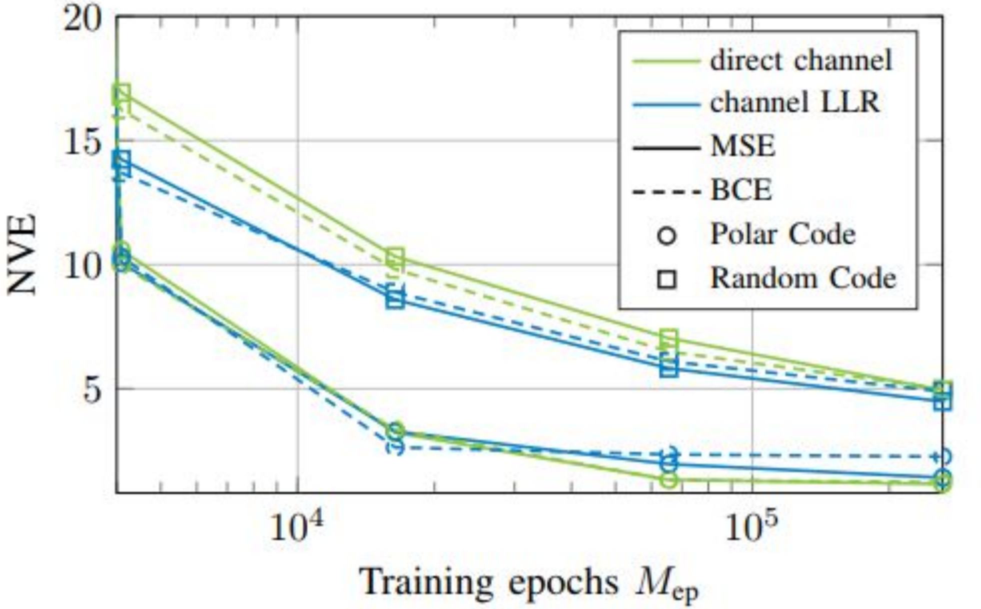


Fig. 4: Learning curve for 16 bit-length codes with code rate $r = 0.5$ for a 128-64-32 NN.

EFFECT OF LARGE NND ON NVE

- In order to answer the question how large the NN should be, we trained NNs with different sizes and structures. From the findings, we can conclude that, for both polar and random codes, it is possible to achieve MAP performance. Moreover, and somewhat surprisingly, the larger the net, the less training epochs are necessary. In general, the larger the number of layers and neurons, the larger is the expressive power or capacity of the NN. Contrary to what is common in classic machine learning tasks, increasing the network size does not lead to overfitting since the network never sees the same input twice.

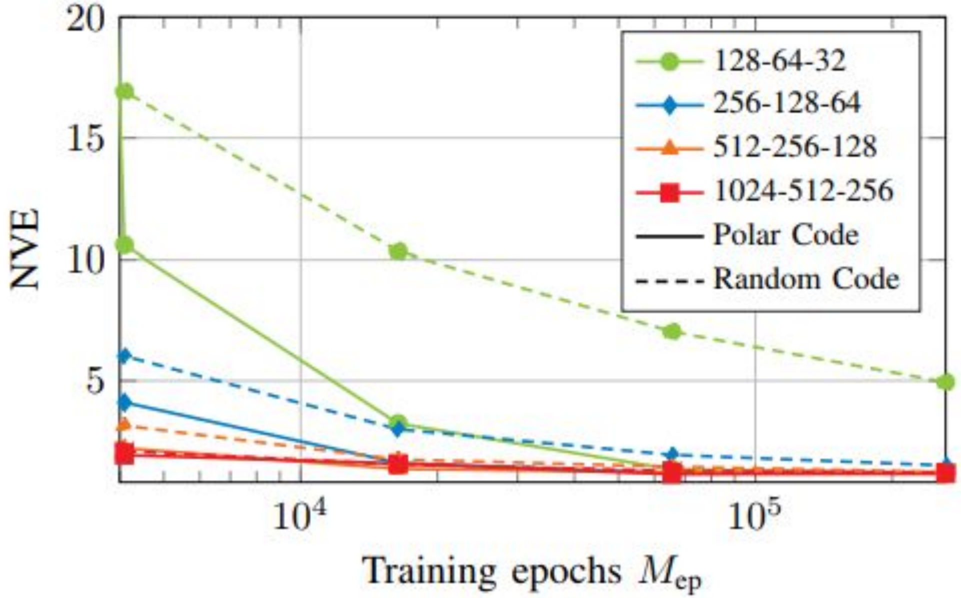


Fig. 5: Learning curve for different NN sizes for 16 bit-length codes with code rate $r = 0.5$.

PERFORMANCE OF NND ON LARGE CODELENGTHS.

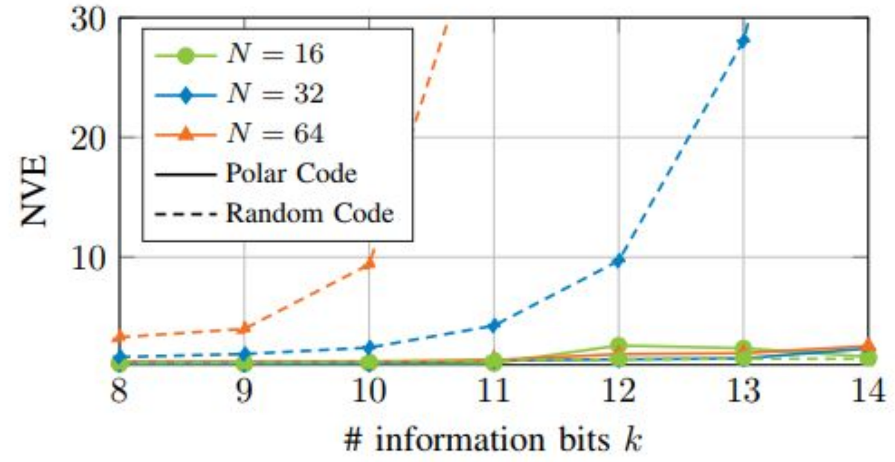
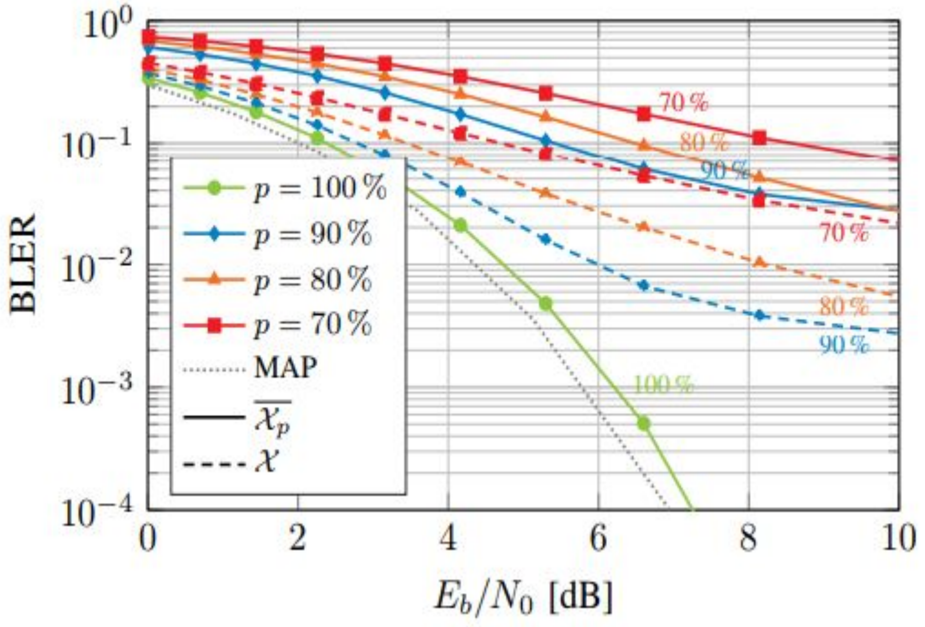


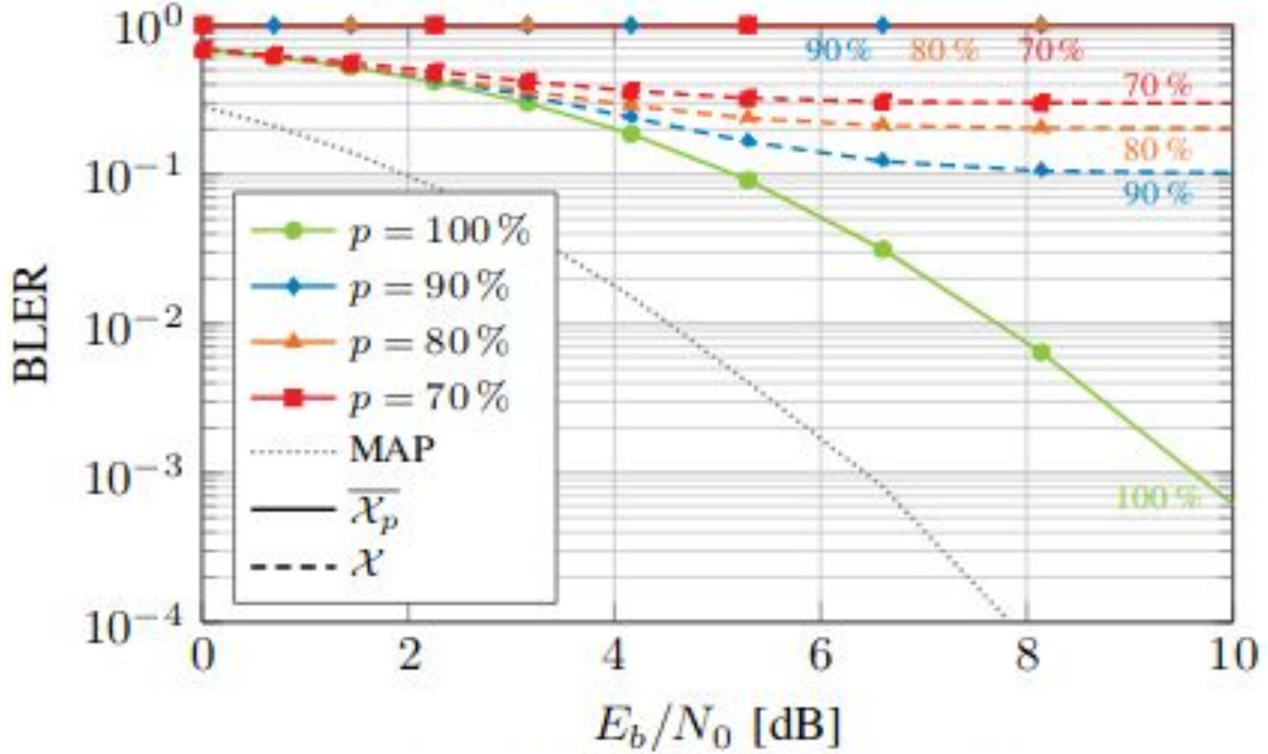
Fig. 6: Scalability shown by NVE for a 1024-512-256 NN for 16/32/64 bit-length codes with different code rates and $M_{\text{ep}} = 2^{16}$ training epochs.

Capability of Generalization

- We have analysed the performance of NND based on BER(Bit error rate) in the previous slides.For better analysis of the accuracy BLER(Block error rate) is used,which takes values only 0 or 1.Hence it is possible to know how exactly the NND is able to predict the entire codeword correctly.



(a) 16 bit-length Polar Code ($r = 0.5$)



(b) 16 bit-length Random Code ($r = 0.5$)

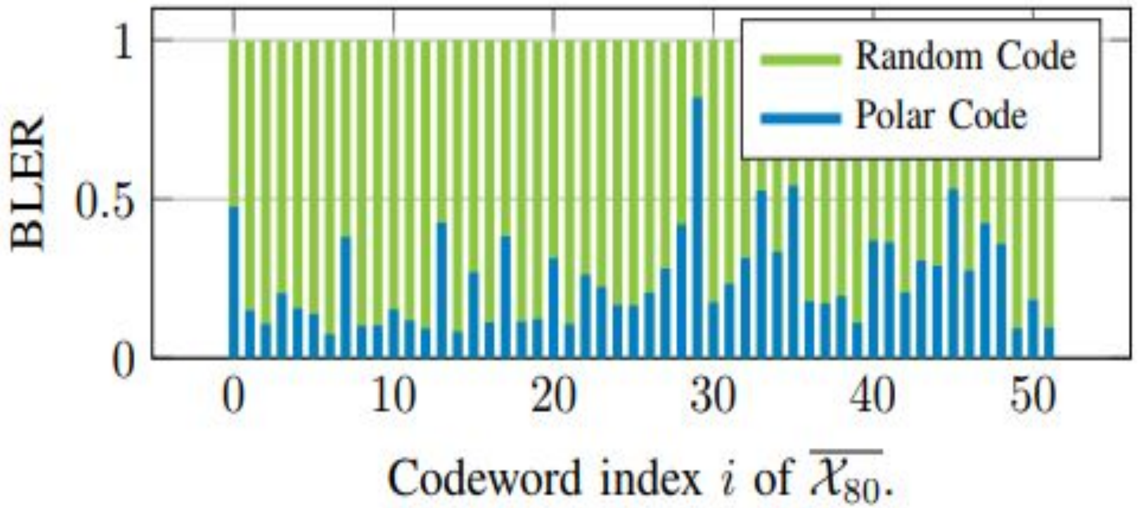


Fig. 8: Single-word BLER for $\mathbf{x}_i \in \overline{\mathcal{X}_{80}}$ at $E_b/N_0 = 4.16\,\mathrm{dB}$ and $M_{\mathrm{ep}} = 2^{18}$ learning epochs.

- From the above figure we can see that the NN fails for almost every unseen random codeword which is plausible. But for a structured code, such as a polar codes, the NN is able to generalize even for unseen codewords. Unfortunately, the NN architecture considered here is not able to achieve MAP performance if it is not trained on the entire codebook. However, finding a network architecture that generalizes best is topic of our current investigations.

- The generalisations that can be made are that the NN can generalize from input channel values with a certain training SNR to input channel values with arbitrary SNR. Second, the NN is able to generalize from a subset Xp of codewords to an unseen subset Xp

Conclusions

- For small block lengths, we achieved to decode random codes as well as polar codes with MAP performance.
- But the exponential complexity increase with the increase in number of information bits in codewords is a limitation.
- State-of-the-art polar decoding currently suffers from high decoding complexity, a lack of possible parallelization and, thus, critical decoding latency.

- NND inherently describes a highly parallelizable structure,enabling one-shot decoding.
- This renders deep learning-based decoding a promising alternative channel decoding approaches it avoids sequential algorithms.
- Future investigations will be based on the exploration of regularization techniques as well as recurrent and memory-augmented neural networks.