# Hyper-Graph-Network Decoders for Block Codes

Eliya Nachmani, Lior Wolf

Tel Aviv University
Facebook AI Research

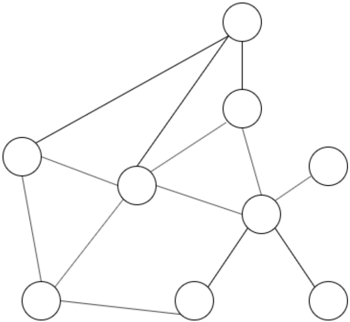
November 2019

# Table Of Contents

- Graph Neural Network (GNN)
- Hyper Network
- Error Correcting Codes
  - Neural Belief Propagation
- Hyper-Graph-Network Decoders
  - Architecture
  - Symmetry Conditions
- Experiments and Results

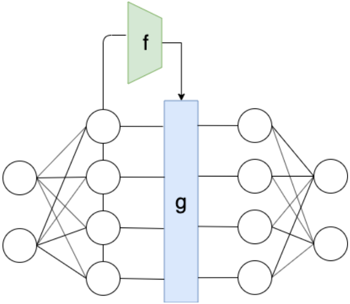
# Graph Neural Network

- A deep learning architecture that operate on graphs structure
- Every node in the graph is a neural network
- The connection between nodes obtained from the graph adjacency matrix:



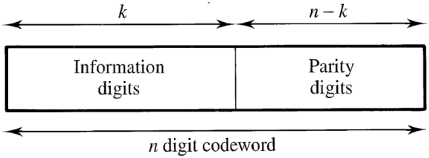
# Hypernetwork

- A neural architecture that has adaptive capabilities
- A network $f$ is trained to predict the weights $\theta_g$ of another network $g$:



# Error Correcting Codes

- Techniques to deliver reliable digital data over unreliable communication channels
- Linear block code:
  - A $(n, k)$ block code, $n > k$
  - Block - block in, block out
  - Linear - addition of two codeword is a codeword



# Error Correcting Codes

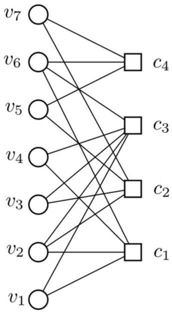
- Parity check matrix $H_{(n-k)\times n}$ - each row is a linear relations that the components of a codeword must satisfy
- For example - $(n, k) = (7, 4)$:

$$\boldsymbol{H} = \begin{pmatrix} 1\,0\,1\,1\,1\,0\,0 \\ 1\,1\,1\,0\,0\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1 \end{pmatrix}$$

- $c$ is codeword if and only if - $c_{1\times n}H^T_{n\times(n-k)} = 0$
- Can be used to:
  - Decide whether a particular vector is a codeword
  - Decode in decoding algorithm

# Belief Propagation Algorithm

- Algorithm for decoding linear block codes
- Subclass of message passing algorithm
- The messages passed along the edges are probabilities, or beliefs
- BP decoder can be constructed from the Tanner graph:



# Belief Propagation Algorithm

- Input - LLR

$$l_v = \log \frac{\Pr(C_v = 1 | y_v)}{\Pr(C_v = 0 | y_v)}$$

  $y_v$ is the channel output corresponding to the $v$th codebit, $C_v$.

- For odd $i$ and $e = (v, c)$ -

$$x_{i,e=(v,c)} = l_v + \sum_{e'=(v,c'),\, c' \neq c} x_{i-1,e'}$$

- For even $i$ and $e = (v, c)$ -

$$x_{i,e=(v,c)} = 2\tanh^{-1}\left(\prod_{e'=(v',c),\, v' \neq v} \tanh\left(\frac{x_{i-1,e'}}{2}\right)\right)$$

- The final $v$th output -

$$o_v = l_v + \sum_{e'=(v,c')} x_{2L,e'}$$

# Neural Network Decoder - Y. Be'ery, D. Burshtein

Re-formalize the BP algorithm as deep neural network

- Input - LLR
- For odd $i$ and $e = (v, c)$ -

$$x_{i,e=(v,c)} = 2\tanh^{-1}\left(\prod_{e'=(v',c),\, v'\neq v} x_{i-1,e'}\right)$$

- For even $i$ and $e = (v, c)$ -

$$x_{i,e=(v,c)} = \tanh\left(\frac{1}{2}\left(w_{i,v}l_v + \sum_{e'=(v,c'),\, c'\neq c} w_{i,e,e'}x_{i-1,e'}\right)\right)$$

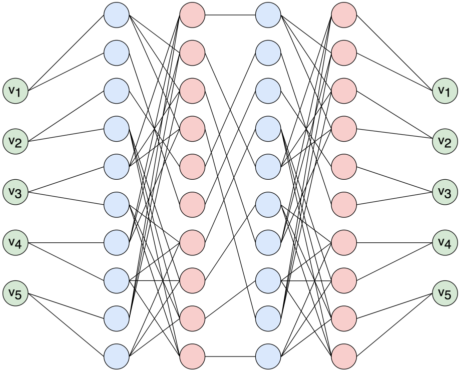
- The final $v$th output -

$$o_v = \sigma\left(w_{2L+1,v}l_v + \sum_{e'=(v,c')} w_{2L+1,v,e'}x_{2L,e'}\right)$$

where $\sigma(x) \equiv (1 + e^{-x})^{-1}$

# Deep Neural Network Architecture

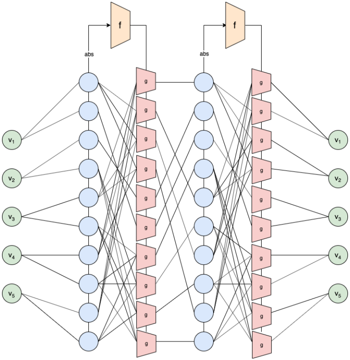
- Unfolding the BP iterations
- Block code with $n = 5$ (correspond to 2 BP iterations)



# Hyper-Graph-Network Decoder

# Hyper-Graph-Network Decoder

- We suggest adding learned components:
  - Graph neural network - replace each variable neuron with neural network
  - Hypernetwork - adding network $f$ to predict the weights of the variables nodes network



# Hyper-Graph-Network Decoder

- Replace odd $j$ equation with the following equations:

$$\theta_g^j = f(|x^{j-1}|, \theta_f) \tag{1}$$

$$x_e^j = x_{(c,v)}^j = g(l_v, x_{N(v,c)}^{j-1}, \theta_g^j), \tag{2}$$

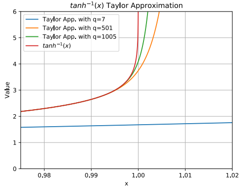
- $\theta_g^j$ is the weights of network $g$ at iteration $j$. $\theta_f$ are the learned weights of network $f$
- The absolute value of the message can be seen as measure for the correctness and the sign corresponding bit value
- In order to focus on the correctness of the message and not the information bits, the input to $f$ is in absolute value

# Hyper-Graph-Network Decoder

- In order to regularize training, we replace the $arctanh$ in the updating equation of even $j$ with Taylor approximation:

$$x_e^j = x_{(c,v)}^j = 2\sum_{m=0}^{q} \frac{1}{2m+1} \left( \prod_{e' \in N(c) \setminus \{(c,v)\}} x_{e'}^{j-1} \right)^{2m+1} \quad (3)$$

- Where $q$ is the Taylor approximation of degree q
- The $arctanh$ activation, has asymptotes in $x = 1, -1$, and training with it often explodes. Its Taylor approximation is a well-behaved polynomial:



# Hyper-Graph-Network Decoder - Symmetry Conditions

- For block codes that maintain certain symmetry conditions, the decoding error is independent of the transmitted codeword
- A direct implication is that we can train our network to decode only the zero codeword
- There are two symmetry conditions:
  - $\Phi\left(b^{\top} x^{j-1}_{N(v,c)}\right) = \left(\prod_1^K b_k\right) \Phi\left(x^{j-1}_{N(v,c)}\right)$
  - $\Psi\left(-l_v, -x^{j-1}_{N(v,c)}\right) = -\Psi\left(l_v, x^{j-1}_{N(v,c)}\right)$
  - $\Phi$ is the check node function and $\Psi$ is the variable node function

# Hyper-Graph-Network Decoder - Symmetry Conditions

Our method, by design, maintains the symmetry condition on both the variable and the check nodes. This is verified in the following lemmas:

**Lemma**

*Assuming that the check node calculation is given by Eq.* (3) *then the proposed architecture satisfies the first symmetry condition.*

**Lemma**

*Assuming that the variable node calculation is given by Eq.* (2) *and Eq.* (1)*,* $g$ *does not contain bias terms and employs the* $tanh$ *activation, then the proposed architecture satisfies the variable symmetry condition.*

# Experiments

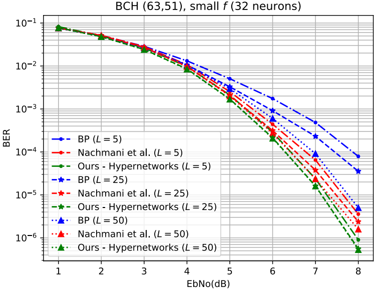
- We train the proposed architecture with three classes of linear block codes: Low Density Parity Check (LDPC) codes, Polar codes and Bose-Chaudhuri-Hocquenghem (BCH) codes
- Training examples are generated as a zero codeword transmitted over an additive white Gaussian noise
- The learning rate was $1e-4$ for all type of codes
- The decoding network has ten layers which simulates $L = 5$ iterations of a modified BP algorithm

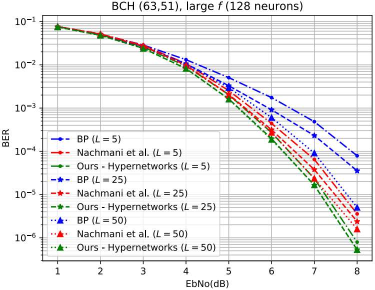
# Experiments

- Each training batch contains examples with different Signal-To-Noise (SNR) values
- The order of the Taylor series of $arctanh$ is set to $q = 1005$
- The network $f$ has four layers with $32$ neurons at each layer. The network $g$ has two layer with $16$ neurons at each layer
- For BCH codes, we also tested a deeper configuration in which the network $f$ has four layers with $128$ neurons at each layer

# Results

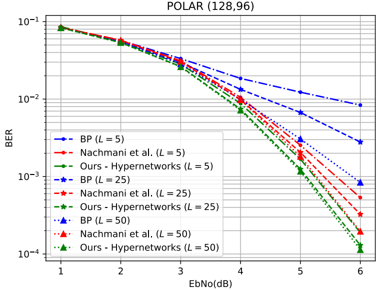
- We present the BER for BCH(63,51) with small and large $f$. As can be seen, we achieve improvements of $0.45$dB, $0.43$dB respectively:

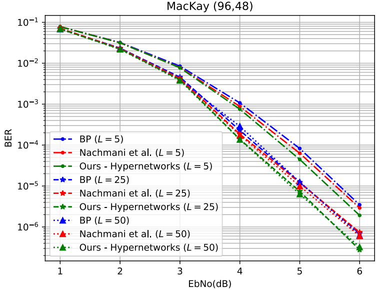


# Results

- We present the BER for Polar(128,96) and LDPC MacKay(96,48). As can be seen, we achieve improvements of $0.48$dB, $0.15$dB respectively:



# Conclusions

- We presents graph networks decoder in which the weights are a function of the node's input
- We present a method to avoid gradient explosion
- By carefully designing our networks, important symmetry conditions are met and we can train efficiently
- Our method introduce a new learnable component to neural decoders

# Gated HyperNet Decoder for Polar Codes

- Gated HyperNet decoding for Polar codes: