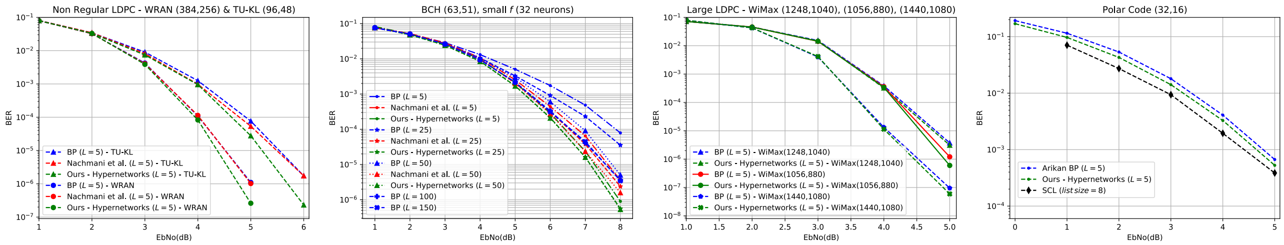

(a) (b) (c) (d)

Figure 1: (a) Non-regular LDPC, (b) BP at convergence, (c) Large LDPC code, (d) Polar code with SCL

We thank our colleagues for their mostly supportive and very constructive and detailed feedback.

Reviewer #1: We will refer to the **Taylor approximation** in terms of numerical stability (line 6) and remove the unfortunate term "regularize training" (l.134). Following the request, we tried to train our method while clipping the arctanh at multiple threshold values (TH $= 0.5, 1, 2, 4, 5$, applied to both the positive and negative sides, multiple block codes BCH(31,16), BCH(63,45), BCH(63,51), LDPC (49,24), LDPC (121,80), POLAR(64,32), POLAR(128,96), $L = 5$ iterations). In all cases, the training exploded, similar to the no-threshold vanilla arctanh (l.209,214-215). In order to understand this, we observe the values when arctanh is applied at initialization for our method and for [17,18]. In [17,18], which are initialized to mimic the vanilla BP, the activations are such that the maximal arctanh value at initialization is 3.45. However in our case, in many of the units, the value explodes at infinity. Clipping does not help, since for any threshold value, the number of units that are above the threshold (and receive no gradient) is large. Since we employ hypernetworks, the weights θ_g^j of the network g are dynamically determined by the network f and vary between samples, making it challenging to control the activations g produces. This highlights the critical importance of the Taylor approximation for the usage of hypernetworks in our setting. **Non-regular LDPC**: Following the request, we ran experiments with WARN$(384, 256)$ and TU-KL$(96, 48)$. As can be seen in Fig. 1(a), our method improves the results, even in non-regular codes where the degree varies. Note that we learned just one hypernetwork g, which corresponds to the maximal degree and we discard irrelevant outputs for nodes with lower degrees. **BP at convergence**: In the paper (l.203), we ran BP experiments until $L = 50$ considering it a convergence point. However, following the review, we ran BP for many more iterations ($L = 150$, which is identical to $L = 100$, showing convergence) and verified that our method indeed obtains considerably better performance. See Fig. 1(b) for a single result that reflects the situation in all tested codes. **Large Codes:** Following the request, we tested WiMax$(1248, 1040)$, WiMax$(1056, 880)$ and WiMax$(1440, 1080)$. Fig. 1(c) shows an improvement for large SNR ($> 4dB$) despite (as R1 mentions) BP being close to optimal in such lengths. **"Minor issues":** We will clarify the sentence regarding training with the zero codeword. The description of the second minor issue seems to have been cut in the middle and we are not sure that we understand it. Running without the absolute value is part of the ablation.

Reviewer #2: The main idea behind the paper is that by employing decoders that are based on hypernetworks, we are able to adapt dynamically to the received noisy codeword and improve the performance. However, applying hypernetworks to such graph networks is uncharted territory, with many challenges that arise from the lack of control over the weights at initialization, the dynamic nature of the weights, and the need to conform with specific symmetry conditions (Lemmas 1,2). Please see the ablation analysis (l.205-216) and note that: (i) hypernetworks allow us to adapt dynamically (l.120), (ii) abs(x) focuses on the reliability of the signal (l.122-125), (iii) Taylor approximation, please see reply to R1, (iv) symmetry conditions, allow to train only on the zero codeword (l.146-148), (v) marginalization at every iteration leads to a better gradient flow (l.140). **State of the art in polar codes**: In the paper, we incorporate our method into the vanilla BP decoder. This decoder does not exploit the special structure of polar codes, leading to results that are below the state of the art. As far as we know, no learning method obtains results on polar codes that are better than the list-successive-cancellation (SCL) method. Following the review, we implemented our method on the BP decoder by Arikan (E. Arikan, "Polar codes: A pipelined implementation"), which makes use of the structure of polar codes. This is done by replacing the f function in Arikan BP with a neural network g, whose weights are obtained from another function f. The input to f is the absolute value of the input LLRs. As Fig. 1(d) shows, our method improves the Arikan BP decoding and is close to the performance of an SCL decoder, which is very close to the ML bound.

Reviewer #4: Thank you for pointing "Learning to decode LDPC codes with finite-alphabet message passing" by Vasic et al, which we will cite. In their work, they learn the node activation based on components from existing decoders (BF, GallagerB, MSA, SPA), while we learn the node activations from scratch. They publish results on Tanner(155,64) and QC-LDPC(1296,72). In both codes, our performance is better across all SNR. For example, for the first code, for both SNR=5.5 and SNR=6, we obtain a third of their bit error rate (5×10^{-7} and 7×10^{-8}, respectively). For the second code, for SNR=2.5 (SNR=3), we obtain half of their bit error rate 1.3×10^{-2} (4.5×10^{-3}). **Additional codes**: See comments to R1 regarding non-regular LDPC codes and large codes.