# MOLECULE PROPERTY PREDICTION AND CLASSIFICATION WITH GRAPH HYPERNETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Graph neural networks are currently leading the performance charts in learning-based molecule property prediction and classification. Computational chemistry has, therefore, become the a prominent testbed for generic graph neural networks, as well as for specialized message passing methods. In this work, we demonstrate that the replacement of the underlying networks with hypernetworks leads to a boost in performance, obtaining state of the art results in various benchmarks.

A major difficulty in the application of hypernetworks is their lack of stability. We tackle this by combining the current message and the first message. A recent work has tackled the training instability of hypernetworks in the context of error correcting codes, by replacing the activation function of the message passing network with a low-order Taylor approximation of it. We demonstrate that our generic solution can replace this domain-specific solution.

## 1 INTRODUCTION

The field of learning-based prediction of molecule properties holds the promise of delivering accurate predictions at a fraction of the complexity that is required by the Density Functional Theory (DFT) models, while not being tied to the assumptions and approximations of this theory. This order of magnitude reduction in runtime supports not only the rapid screening of molecule banks, with important applications in medicine, manufacturing, and environmental science, but also the automatic design of new materials.

Molecules are often represented as graphs. Similarly to other application fields, such as computer vision, computational chemistry has benefited both from the development of powerful generic (graph) neural networks, as well as the development of specialized methods that are developed for the specific prediction tasks. For example, the state of the art NMP-Edge method of Jørgensen et al. (2018) is a sophisticated domain-specific method, with many significant algorithmic choices, which generalizes the method of Schütt et al. (2017a) and incorporates ideas from the work of Gilmer et al. (2017) and Kearnes et al. (2016).

In this work, we propose a generic way to improve graph neural networks and demonstrate that it is able to improve molecule property prediction and classification in both specialized networks and in more generic methods that are applied to computational chemistry datasets. The value of our scheme stems from its ability to improve upon a diverse set of already optimized state of the art methods.

Our method employs hypernetworks, also known as dynamic networks. In such neural networks, the weights of at least some of the layers vary dynamically based on the input. A hypernetwork can be seen as a composite network in which one network predicts the weights of another network. In our case, both networks receive the messages that are passed in the graph neural network as inputs.

Since the weights of the message generating network in the hypernetworks change dynamically during inference, training it is a challenge. We tackle this with a specific way of incorporating the incoming messages into the hypernetwork. Instead of passing the current message, we pass a linear combination of the current message and the first message. This simple modification is enough to ensure an improvement in performance. Without it, the hypernetwork would not typically outperform the original network.

Our experiments show that the same scheme is able to improve the predictions provided by three state of the art methods: the NMP-Edge network, the Invariant Graph Network of Maron et al. (2019a), and the Graph Isomorphism Network of Xu et al. (2019). In addition, we evaluate our method in the domain of error correcting codes, in which a very recent contribution by Nachmani & Wolf (2019) employed hypernetworks to improve the accuracy of a message passing scheme. We are able to show that our modification of the input messages is able to replace the method used there to stabilize the network. Taking both methods together, the results further improve.

## 2 Related Work

**Graph Networks** The topic of graph neural networks has drawn considerable attention, both in the context of specific applications, such as text analysis (Socher et al., 2013) and computer vision (Johnson et al., 2018) and as a generic tool. Earlier work employed recursive neural networks (Goller & Kuchler, 1996; Gori et al., 2005; Scarselli et al., 2008; Li et al., 2016), where the information flows once in a network that is generated based on the graph. Most current methods, including graph spectra methods (Bruna et al., 2013; Defferrard et al., 2016; Kipf & Welling, 2016) can be cast as message passing algorithms (Gilmer et al., 2017; Duvenaud et al., 2015; Li et al., 2016; Battaglia et al., 2016; Kearnes et al., 2016; Schütt et al., 2017b; Xu et al., 2019).

The generic message passing methods employ three networks: one that pools the hidden states from the neighborhood graph vertices, another that updates the hidden states based on the aggregated representation of the neighbouring vertices, and one that reads the information from the entire graph in order to generate the final classification. In such a network, the messages are the hidden states of the nodes. In molecule prediction, conditioning the messages also on the receiving graph node and storing a hidden state for the linking edge improves performance (Kearnes et al., 2016; Gilmer et al., 2017; Jørgensen et al., 2018).

An alternative to message passing techniques, is presented by permutation equivariant operators on the tensors that represent $k$-order interactions between graph nodes (Kondor et al., 2018; Maron et al., 2018; Murphy et al., 2019; Maron et al., 2019a). We are able to demonstrate that applying our method to both message passing methods and equivariant operator methods improved performance.

**Molecule property prediction** While in the past, feature engineering was the main route to applying machine learning in chemistry (Rogers & Hahn, 2010; Rupp et al., 2012; Montavon et al., 2012; Hansen et al., 2015; Huang & Von Lilienfeld, 2016), neural networks have become increasingly popular. The NMP-Edge model of Jørgensen et al. (2018) described in Sec. 3.1 is an example of a specialized model. It follows the basic architecture of Gilmer et al. (2017), in which the messages are conditioned on the nodes across both sides of the edge and on the hidden representation of the edge. In the NMP-Edge model, however, similar to SchNet (Schütt et al., 2017a), the message is an elementwise product of a network that encodes the sending node and a network that encodes the edge (not encoding the receiving edge directly). Also similar to SchNet, an RBF initialization and a soft-plus activation are used. Unlike SchNet, the edge embedding is being updated in time, following the Weave network proposed by Kearnes et al. (2016).

An example of a generic graph network solution that also excels on the popular QM9 benchmark (Ramakrishnan et al., 2014; Ruddigkeit et al., 2012) is the Invariant Graph Network (IGN) of Maron et al. (2019a), presented in Sec. 3.2.

**Hypernetworks** Dynamic layers, also known as gating layers, are layers in which the weights are determined by a separate neural network. Such networks were introduced by Klein et al. (2015); Riegler et al. (2015) for visual tasks that require an adaptation of the input image. More recently, the term *hypernetworks* was coined to refer to a composite neural network in which a network $f$ is trained to predict the weights $\theta_g$ of another network $g$. The shift from specific layers to entire networks was presented by Jia et al. (2016), who employed hypernetworks for video frames and stereo views prediction. The usage of hypernetworks for recurrent neural networks was presented by Ha et al. (2016). Krueger et al. (2017) have presented a Bayesian formulation of hypernetworks, and such networks have become prominent in meta-learning following Bertinetto et al. (2016), who studied transfer learning between multiple few-shot learning tasks.

Since the weights of network $g$ are generated instantaneously by network $f$, Brock et al. (2018) have used hypernetworks for searching over the space of possible network architectures. In this

case, a lengthy backpropagation optimization is replaced by the feed forward prediction of network $f$. More related to our work is that of Zhang et al. (2019), who use hypernetworks on graphs, also in the domain of network architecture search. In this work, the weight generating network $f$ is a graph network that operates on the graph that captures the generated architecture.

Another recent application of hypernetworks to graphs is the work of Nachmani & Wolf (2019), where an MLP generates the weights of a message passing network that decodes error correcting codes. This generalizes earlier attempts in the domain of network decoders including (Nachmani et al., 2016; Kim et al., 2018; Gruber et al., 2017; Teng et al., 2018; Cammerer et al., 2017; Vasić et al., 2018) and is shown to improve performance. The input to both the weight generating network $f$ and the message generation network $g$ is the incoming message, where for the first network, the absolute value is used. It is shown that training hypernetworks suffers from severe initialization challenges and would often lead to the explosion of the weights. Nachmani & Wolf (2019), therefore, present a new activation function that is more stable than the $arctanh$ activation typically used in message passing decoders. In our work, we employ conventional activations, and do not employ the absolute value for molecule prediction. We demonstrate that a combination of the initial message (from the first iteration) with the last message is an effective way to stabalize the training of the graph hypernetwork and do not employ dedicated activation functions.

## 3 GRAPH HYPERNETWORKS

We extend three leading architectures for graph neural networks that were either designed for the molecule inference task or shown to excel on it. In each case, we add a hypernetwork scheme in which the input is a linear combination of the first message passed in the network and the current message. As our experiments show, in all three cases, sizable gains in performance are obtained, in comparison to the underlying method. In addition, in order to compare ourselves with a recent hypernetwork message passing scheme, we modify the decoding method of Nachmani & Wolf (2019).

### 3.1 EXTENDING THE NMP-EDGE NETWORK BY JØRGENSEN ET AL. (2018)

In order to describe how hypernetworks are applied to the NMP-Edge network, we rely on the original notation of Jørgensen et al. (2018). Let $h_v^t$ be the hidden state of a node associated with a specific atoms at iteration $t$, and $e_{vw}^t$ be the hidden state representation of an edge, which denotes either a chemical link between atom or spatial proximity. The hidden states of the atoms $h_v^0$ are initialized using a look-up table and the hidden state of the edges $e_{vw}^0$ are initialized using an RBF function with multiple scales, following Jørgensen et al. (2018); Schütt et al. (2017a).

The message passing scheme of the original NMP-Edge network takes the form:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_w^t, e_{vw}^t), \tag{1}$$

$$h_v^{t+1} = S_t\left(h_v^t, m_v^{t+1}\right), \tag{2}$$

where $m_v^t$ are the messages aggregated at node $v$ at time $t$, and $M_t$, $S_t$ are the message and transition networks for iteration $t$. These networks are dynamic (vary between iterations) but are independent of the inputs. The earlier work by Schütt et al. (2017b) uses a similar set of networks which do not change between the iterations.

In our modified network, we replace the state transition function with the hypernetwork $f$ and $g$ as follows:

$$\theta_g^t = f\left(c \cdot h_v^0 + (1-c) \cdot h_v^t\right) \tag{3}$$

$$h_v^{t+1} = h_v^t + g_{\theta_g^t}\left(m_v^{t+1}\right) \tag{4}$$

where $c$ is a learned damping factor, which is clipped to be in the range [0,1] and is initialized with a uniform distribution, and the weights of network $g$ are given by $\theta_g^t$. For $t = 0$ , Eq. 3 becomes $\theta_g^0 = f\left(h_v^0\right)$. Note that $f$ is a fixed function. However, $g_{\theta_g^t}$ vary in time, since the set of weights $\theta_g^t$ change as the input to $f$ changes.

The readout function is the same as in Jørgensen et al. (2018), which is two layer neural network that pools from all of the network atoms. The network $M_t$, the readout network, and network $S_t$ of

the original architecture employ a shifted-soft-plus network, following Schütt et al. (2017a), while $f, g$ employ $tanh$. Bias terms are not used. The number of layers is two in both $g$ and $S_t$. $f$ has four layers (we believe that the architecture of $S_t$ is locally optimal and adding layers to it did not improve the accuracy in our experiments).

### 3.2 Extending the Invariant Graph Network of Maron et al. (2019a)

We follow the original notations of IGN, in order to enable a quick reference to the original IGN work. The original model with $d$ blocks has the form $F = m \circ h \circ B_d \circ ... \circ B_2 \circ B_1$ where, $h$ is an invariant layer (Maron et al., 2019b), $m$ is a MLP, and the blocks $B_i$ are defined as:

$$Y_i = B_i(X_i) = [m_3(X_i), m_1(X_i) \odot m_2(X_i)] \tag{5}$$

where $X_i \in \mathbb{R}^{n \times n \times a}$ denote the input tensor to the block, $\odot$ denotes element-wise multiplication, the square brackets denote concatenation along the last tensor dimension, and the three MLPs $m_1, m_2 : \mathbb{R}^a \to \mathbb{R}^b$ and $m_3 : \mathbb{R}^a \to \mathbb{R}^{b'}$ are applied to each of the $n \times n$ elements of the input tensor individually along the third tensor dimension. The dimension of the block's output $Y_i$ is, therefore, $n \times n \times (b' + b)$.

The modified IGN network has the form:

$$F = m \circ h \circ B_d \circ H_{d-1} \circ ... \circ B_2 \circ H_1 \circ B_1, \tag{6}$$

where $H_1, ..., H_{d-1}$ are hyper blocks. Let $Y_i = B_i(X_i)$ be the input tensor to the hyper block $H_i$. The hyper block $H_i$ performs the following computation:

$$\theta_g^i = f\left(c \cdot Y_0 + (1 - c) \cdot Y_i\right) \tag{7}$$

$$H_i(Y_i) = g_{\theta_g^i}\left(c \cdot Y_0 + (1 - c) \cdot Y_i\right) \tag{8}$$

where the damping factor $c$ is a learned parameter, initialized from the uniform [0,1] distribution and clipped to remain in this range. The input tensors for $f$ and $g$ are an aggregation of $Y_0$ and $Y_i$ with the damping factor $c$. Each layer in $g$ is applied to each feature of the input tensor independently along the third dimension. As before, $f$ and $g$ are neural networks with the $tanh$ activation with $f$ having four layers and $g$ two.

We use the same suffix networks per benchmark as Maron et al. (2019a). For QM9 $h$ of Eq. 6 is an invariant max pooling, which is followed by a MLP $m$ with three layers. For the classification datasets, $h$ is an invariant max pooling layer from every block $B_i$ output $H_i(Y_i)$ ($Y_i$ in the original work) followed by a single layer. These outputs are then summed to produce the network output.

### 3.3 Extending the Graph Isomorphism Network of (Xu et al., 2019)

We now turn to the notation used in (Xu et al., 2019) to introduce the modified GIN model. $G(V, E)$ is the graph with node feature vector $X_v$ for vertices $v \in V$ and edges $E$. In the graph classification problem, one is given a set of graphs with matching labels $\{(G_1, y_1), (G_2, y_2), ..., (G_N, y_N)\}$. The GNN model calculates representation vectors $h_v$ for each node $v$ in an iterative manner. After convergence, a readout function calculates the global graph embedding $h_G$, from which the label is predicted $\bar{y}_G = M(h_G)$ for a MLP $M$. In GIN, the readout takes the form of a summation followed by concatenation:

$$h_G = \left[\sum_{v \in G} h_v^{(1)}, \sum_{v \in G} h_v^{(2)}, \ldots, \sum_{v \in G} h_v^{(K)}\right] \tag{9}$$

where $K$ is the final iteration used for prediction. The update function of the hidden node representation for iteration $k$ is given by a MLP that is specific to this iteration:

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum\nolimits_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right) \tag{10}$$

where $\epsilon^{(k)}$ is either a learned parameter or a fixed scalar, depending on the experiment.

The modified GIN model we propose modifies the update step, without changing the final readout:

$$\theta_g^k = f\left(c \cdot h_v^0 + (1 - c) \cdot \left(h_v^{(k-1)} + \sum\nolimits_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right)\right) \tag{11}$$

$$h_v^{(k)} = g_{\theta_g^k}\left(c \cdot h_v^0 + (1-c) \cdot \left(h_v^{(k-1)} + \sum\nolimits_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right)\right) \tag{12}$$

where $h_v^0$ is calculated from Eq. 10 with $k=0$ as $h_v^{(0)} = \text{MLP}^{(0)}\left(\left(1+\epsilon^{(0)}\right) \cdot x_v + \sum_{u \in \mathcal{N}(v)} x_u\right)$, where $x_v$ is the vector of input features of node $v$. $c$ is a damping factor that is learned during training. $f$ and $g$ are neural networks with three and two hidden layers respectively with the $tanh$ activation. Note that the network $g$ changes between iterations and across nodes, depending on the input to $f$. However, the entire hypernetwork is fixed between the iterations.

### 3.4 Extending the Decoding Hypernetwork of Nachmani & Wolf (2019)

Since Nachmani & Wolf (2019) have proposed to extend an existing network using a hypernetwork, we modify their work in order to compare our way of converting a graph network to a hypernetwork with theirs. Specifically, it is reported that hypernetworks cannot train for the task of decoding error correcting codes, unless a dedicated activation function is used, since any other activation function attempted in their experiments leads to a divergence of the weights.

Nachmani & Wolf (2019) modify the belief propagation algorithm of Nachmani et al. (2016), which is given by:

$$x_e^j = x_{(c,v)}^j = \begin{cases} \tanh\left(\frac{1}{2}\left(l_v + \sum_{e' \in N(v)\setminus\{e\}} w_{e'} x_{e'}^{j-1}\right)\right), & \text{j is odd} \\ 2arctanh\left(\prod_{e' \in N(c)\setminus\{e\}} x_{e'}^{j-1}\right), & \text{j is even} \end{cases} \tag{13}$$

where $l_v$ is the log likelihood ratios of the input bits, $x_e^j$ is the computed edge message for the edge $e=(c,v)$ in a Tanner graph, which is a bidirectional graph that has variable nodes $v$ on one side and check nodes $c$ on the other. Let $H$ be the parity check matrix. Each variable node is indexed by an edge $e=(c,v)$ on the Tanner graph and $N(v) = \{(c,v)|H(c,v)=1\}$, i.e, the set of all edges in which $v$ participates. $x_e^{j-1}$ is the message from the previous iteration.

The hypernetwork model of Nachmani & Wolf (2019) has the following update equations for odd $j$:

$$\theta_g^j = f(|x^{j-1}|, \theta_f) \tag{14}$$

$$x_e^j = x_{(c,v)}^j = g(l_v, x_{N(v,\setminus c)}^{j-1}, \theta_g^j), \tag{15}$$

where $N(v,\setminus c)$ is a vector that contains the elements of $x^j$ that correspond to the indices $N(v) \setminus \{(c,v)\}$. For even $j$ the update equation takes the form:

$$x_e^j = x_{(c,v)}^j = 2\sum_{m=0}^{q} \frac{1}{2m+1}\left(\prod_{e' \in N(c)\setminus\{(c,v)\}} x_{e'}^{j-1}\right)^{2m+1} \tag{16}$$

where $q$ is the degree of the Taylor approximation of $arctanh$.

We modified the model of Nachmani & Wolf (2019) with the following update equations. For odd $j$:

$$\theta_g^j = f(|c \cdot x^0 + (1-c) \cdot x^{j-1}|, \theta_f) \tag{17}$$

$$x_e^j = x_{(c,v)}^j = g(l_v, c \cdot x^0 + (1-c) \cdot x_{N(v,\setminus c)}^{j-1}, \theta_g^j), \tag{18}$$

where $x^0$ is the output of one iteration from Eq. 13, and $c$ is the damping factor which is learned during training.

For an even $j$ we either use Eq. 16 (Taylor approximated arctanh), or consider the conventional $arctanh$ activation, as in Eq. 13. The readout function is not modified.

## 4 Experiments

We evaluate our model on regression and classification for predicting molecule proprieties and for decoding linear block codes. For regression we use the Quantum Machines 9 (QM9) dataset (Ramakrishnan et al., 2014; Ruddigkeit et al., 2012) and Open Quantum Materials Database (OQMD)

Saal et al. (2013); Kirklin et al. (2015). The QM9 dataset has $133,885$ molecules. Each molecule has $12$ properties for predicting. When comparing our results to NMP-Edge and IGN methods, we use the same train-validation-test split as Jørgensen et al. (2018) or Maron et al. (2019a), respectively. While based on the same dataset, these two benchmarks cannot be directly compared for many of the properties. The OQMD dataset contain $435,582$ inorganic structures. We use the same train-validation-test split as Jørgensen et al. (2018).

For classification we employ the four bioinformatics dataset of Yanardag & Vishwanathan (2015), which contains protein structures or chemical compounds: MUTAG, PROTEINS, PTC and NCI1. We use the original train folds, which are also used by Maron et al. (2019a) and Xu et al. (2019).

For decoding error correcting codes, we use the parity check metrics of Helmling et al. (2019). We use three classes of linear block codes: Low Density Parity Check (LDPC) codes (Gallager, 1962), Polar codes (Arikan, 2008) and Bose-Chaudhuri-Hocquenghem (BCH) codes (Bose & Ray-Chaudhuri, 1960).

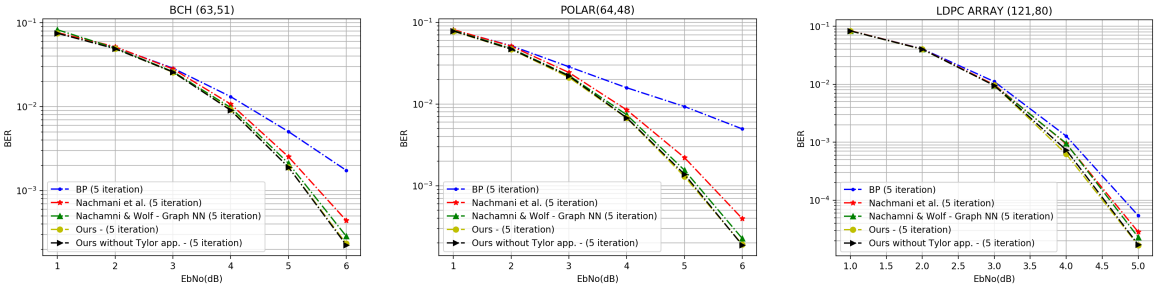
We compare with various baseline method on top of the methods that we modify. For the QM9 and OQMD datasets we compare with V-RF (Ward et al., 2017), SchNet (Schütt et al., 2017a), enn-s2s (Gilmer et al., 2017), Cormorant (Anderson et al., 2019), Incidence (Albooyeh et al., 2019), 123-gnn (Morris et al., 2019) and the two methods by Wu et al. (2018): DTNN and MPNN. The baseline method for the classification datasets are WL subtree (Shervashidze et al., 2011), DCNN (Atwood & Towsley, 2016), PATCHY-SAN(Niepert et al., 2016), DGCNN (Zhang et al., 2018), AWL (Ivanov & Burnaev, 2018), GCN (Kipf & Welling, 2016), and GraphSAGE (Hamilton et al., 2017).

**Implementation details** The various hyperparameters were selected based on the validation set (where vary between experiments) or set arbitrarily based on the underlying architecture (where fixed). **NMP-edge network** For QM9, we trained the models with the ADAM optimizer, with a learning rate set to $1e-4$. The number of iterations was $4$. The number of neurons in network $f$ was $64$ and the number of neurons in network $g$ was $128$. The learning rate decreases by a factor of $0.96$ every $100,000$ gradient steps. The minibatch size was $32$. The node embedding size was $256$ to all the parameters. For OQMD, we use an ADAM optimizer, with a learning rate set to $1e-4$. The learning rate decreases by a factor of $0.96$ every $400,000$ gradient steps. We use a minibatch of $32$ examples. The number of iterations was $3$. The number of neurons per layer in network $f$ was $64$ and that number in network $g$ was $128$. **Invariant graph network** For QM9, we use the same configuration as Maron et al. (2019a), except for the following hyper parameters. When training one model to predict all molecule parameters $f$ has four layers with $128$ neurons, whereas $g$ has two layers with $128$ neurons. When we trained a separate model for each molecule parameter, which calls for a smaller capacity, $f$ has four layers with $64$ neurons and $g$ has two layers with $64$ neurons. For the classification datasets we trained the models with the following hyperparameters, learning rate was $5e-5$ for MUTAG, PTC and NCI1, and was $1e-3$ for PROTEINS. The number of channels in blocks $B_i$ was $400$ for all datasets except for PROTEINS which has $128$ channels. The number of layers in the MLP $m$ was 2 for all datasets. For all datasets the number of neurons in network $f$ was $64$ and the number of neurons in network $g$ was $64$, except for PROTEINS which has $32$ neurons for $f$ and $g$. **Graph isomorphism network** For the classification datasets, we use the same training procedure as Xu et al. (2019), who train the model for 10 folds and choose the number of epochs based on the cross validation accuracy over the folds. We train the models with the following hyper-parameters. Learning rate was $5e-3$, the models run for $5$ iterations and the number of epochs was $180$ for all datasets. The minibatch sizes were $512$, $256$, $32$ and $16$ for NCI1, PTC, MUTAG and PROTEINS, respectively. In all datasets, $f$ has three layers and $64$ neurons, $g$ has two layers with $64$ neurons each. **Decoding hyper-network** We use the same hyper-parameters as in Nachmani & Wolf (2019).

### 4.1 RESULTS

**Graph regression** The results for the QM9 dataset are reported in Tab. 1 for the NMP-Edge compatible splits and units and in Tab. 3 for the benchmark version used by IGN. Our model based on the NMP-Edge architecture achieves state of the art performance on 9 out of 12 parameters, and in only one parameter it is outperformed by the original NMP-Edge model (and another tie).

The result for formation energy predictions OQMD, based on the NMP-Edges architecture, is provided in Tab. 2. We obtain state of the art performance in this benchmark as well.




$(a)$ $(b)$ $(c)$

Figure 1: BER for various values of SNR for various codes. (a) BCH (63,51), (b) POLAR(64,48), (c) LDPC ARRAY (121,80).

The QM9 model that is based on IGN obtains state of the art performance on 7 out of 12 parameters when training the model for each parameter. Furthermore, we improve the results of 9 out of 12 parameters when comparing to IGN model that is trained for each parameter separately. When training one model to predict all the parameters, we improve 12 out of 12 parameters, compared to the IGN model.

**Graph classification** The results for the classification datasets are provided in Tab. 4. As can be observed, our modified versions of IGN and GIN improve the baseline IGN and GIN models in almost all cases (in one case we tie). Note that the GIN model has many variants and our modification is based on the GIN-$\epsilon$ model. There is no Graph Neural Network model that outperforms our results, and for the PTC and PROTEINS datasets, our method outperforms all literature baselines.

**Error Correcting Codes results** In Fig. 1 we provide the BER-SNR results for multiple linear block codes. Our method improves on Nachmani & Wolf (2019) across codes. We get an improvement range between $0.09dB$ and $0.12dB$ for large SNR. Moreover, in all three cases, we are able to improve the baseline results, even without the Taylor approximation of Nachmani & Wolf (2019). Since Nachmani & Wolf (2019) fail to train without the approximation (using $arctanh$ their runs always diverge), this shows that our method stabilizes the training process for error correcting codes. We can also observe that our results with and without this approximation are almost identical.

### 4.2 Ablation Analysis

In Tab. 5 we provide an ablation analysis on the QM9 benchmark. For the NMP-Edge model, we can observe degradation of 11 out of 12 parameters when training without $h_v^0$ in Eq. 3 and the associated damping factor. Moreover, when training without the hypernetwork, but with $h_v^0$ (Eq. 2 becomes $h_v^{t+1} = c \cdot h_v^0 + (1-c) \cdot S_t\left(h_v^t, m_v^{t+1}\right)$) we get a degradation of 7 out of 12 parameters.

For the IGN model, we can observe degradation in 9 out of the 12 parameters when training without the damping factor and $Y_0$ in Eq. 7, 8. Moreover, when training without the hypernetwork but with the added first message ($X_i$ become $c \cdot X_0 + (1-c) \cdot X_i$ in Eq. 5), we get a degradation of 8 out of the 12 parameters.

## 5 Conclusions

Graph neural networks are becoming the dominant tool in molecule prediction and classification tasks. Here we show that by employing hypernetworks with a stabilization mechanism, significant performance gains are obtained. In order to demonstrate the advantage of our stabilizing mechanism over a recently proposed hypernetwork scheme, we also show improved performance in the field of decoding linear block codes.

Table 1: Mean absolute error for QM9 molecule parameters prediction, using the NMP-Edge splits, units and our modified architecture. The lowest error is in bold. The results of the Incidence and Cormorant networks are from Albooyeh et al. (2019) and Anderson et al. (2019), respectively. The rest of the results from Jørgensen et al. (2018).

| Target | Unit | SchNet | enn-s2s | NMP-Edge | Cormorant | Incidence | Ours |
|---|---|---|---|---|---|---|---|
| $\varepsilon_{HOMO}$ | meV | 41 | 43 | 36.7 | 36 | 89 | **26.56** |
| $\varepsilon_{LUMO}$ | meV | 34 | 37 | 30.8 | 36 | 49 | **23.47** |
| $\Delta\varepsilon$ | meV | 63 | 69 | 58.0 | 60 | 68 | **41.91** |
| ZPVE | meV | 1.7 | 1.5 | **1.49** | 1.982 | 8 | **1.49** |
| $\mu$ | Debye | 0.033 | 0.030 | **0.029** | 0.130 | 0.04 | 0.031 |
| $\alpha$ | $\text{Bohr}^3$ | 0.235 | 0.092 | 0.077 | 0.092 | **0.03** | 0.066 |
| $\langle R^2 \rangle$ | $\text{Bohr}^2$ | 0.073 | 0.180 | 0.072 | 0.673 | **0.017** | 0.057 |
| $U_0$ | meV | 14 | 19 | 10.5 | 28 | 8 | **7.27** |
| $U$ | meV | 19 | 19 | 10.6 | - | 7 | **6.99** |
| $H$ | meV | 14 | 17 | 11.3 | - | 8 | **7.17** |
| $G$ | meV | 14 | 19 | 12.2 | - | 8 | **7.99** |
| $C_v$ | cal/molK | 0.033 | 0.040 | 0.032 | 0.031 | 0.028 | **0.026** |

Table 2: OQMD - NMP-Edge. Mean absolute error for formation energy predictions. The results of the various baselines are from Jørgensen et al. (2018)

| Method: | V-RF | SchNet | NMP-Edge | Ours |
|---|---|---|---|---|
| meV/atom | 74.5 | 27.5 | 14.9 | **13.5** |

Table 3: Mean absolute error for QM9 molecule parameters prediction for the IGN splits, units, and our modified architecture of it. The results of the various datasets are taken from (Maron et al., 2019a). For IGN and our modifications, we show results of a single network predicting all values and of dedicated networks.

| Target | DTNN | MPNN | 123-gnn | IGN - all | IGN - single | Ours - all | Ours - single |
|---|---|---|---|---|---|---|---|
| $\mu$ | 0.244 | 0.358 | 0.476 | 0.231 | 0.0934 | **0.157** | **0.0883** |
| $\alpha$ | 0.95 | 0.89 | **0.27** | 0.382 | 0.318 | 0.325 | 0.303 |
| $\epsilon_{homo}$ | 0.00388 | 0.00541 | 0.00337 | 0.00276 | **0.00174** | 0.00203 | 0.00178 |
| $\epsilon_{lumo}$ | 0.00512 | 0.00623 | 0.00351 | 0.00287 | 0.0021 | 0.00228 | **0.0020** |
| $\Delta_\epsilon$ | 0.0112 | 0.0066 | 0.0048 | 0.00406 | 0.0029 | 0.00306 | **0.0027** |
| $\langle R^2 \rangle$ | 17 | 28.5 | 22.9 | 16.07 | **3.78** | 13.9 | 7.56 |
| $ZPVE$ | 0.00172 | 0.00216 | **0.00019** | 0.00064 | 0.000399 | 0.00049 | 0.000396 |
| $U_0$ | 2.43 | 2.05 | 0.0427 | 0.234 | 0.022 | 0.093 | **0.018** |
| $U$ | 2.43 | 2 | 0.111 | 0.234 | 0.0504 | 0.092 | **0.0174** |
| $H$ | 2.43 | 2.02 | 0.0419 | 0.229 | 0.0294 | 0.093 | **0.0193** |
| $G$ | 2.43 | 2.02 | 0.0469 | 0.238 | 0.024 | 0.093 | **0.017** |
| $C_v$ | 0.27 | 0.42 | **0.0944** | 0.184 | 0.144 | 0.180 | 0.146 |

Table 4: Test set classification accuracies (%) for MUTAG, PROTEINS, PTC, NCI1 datasets. The highest results on bold. The results of IGN are by (Maron et al., 2019a), the rest of the results are from Xu et al. (2019).

| | Datasets | MUTAG | PROTEINS | PTC | NCI1 |
|---|---|---|---|---|---|
| Datasets | # graphs | 188 | 1113 | 344 | 4110 |
| | # classes | 2 | 2 | 2 | 2 |
| | Avg # nodes | 17.9 | 39.1 | 25.5 | 29.8 |
| Baselines | WL subtree | $90.4 \pm 5.7$ | $75.0 \pm 3.1$ | $59.9 \pm 4.3$ | $\mathbf{86.0 \pm 1.8}$ * |
| | DCNN | 67.0 | 61.3 | 56.6 | 62.6 |
| | PATCHYSAN | $\mathbf{92.6 \pm 4.2}$ * | $75.9 \pm 2.8$ | $60.0 \pm 4.8$ | $78.6 \pm 1.9$ |
| | DGCNN | 85.8 | 75.5 | 58.6 | 74.4 |
| | AWL | $87.9 \pm 9.8$ | – | – | – |
| GNN variants | OURS BASED ON GIN | $\mathbf{90.55 \pm 5.4}$ | $\mathbf{76.90 \pm 2.24}$ | $\mathbf{69.68 \pm 5.5}$ | $\mathbf{82.7 \pm 2.0}$ |
| | GIN-0 | $89.4 \pm 5.6$ | $76.2 \pm 2.8$ | $64.6 \pm 7.0$ | $\mathbf{82.7 \pm 1.7}$ |
| | GIN-$\epsilon$ | $89.0 \pm 6.0$ | $75.9 \pm 3.8$ | $63.7 \pm 8.2$ | $82.7 \pm 1.6$ |
| | GIN-SUM–1-LAYER | $90.0 \pm 8.8$ | $76.2 \pm 2.6$ | $63.1 \pm 5.7$ | $82.0 \pm 1.5$ |
| | GCN-MEAN–1-LAYER | $85.6 \pm 5.8$ | $76.0 \pm 3.2$ | $64.2 \pm 4.3$ | $80.2 \pm 2.0$ |
| | GRAPHSAGE-MAX–1-LAYER | $85.1 \pm 7.6$ | $75.9 \pm 3.2$ | $63.9 \pm 7.7$ | $77.7 \pm 1.5$ |
| | OURS BASED ON IGN | $\mathbf{91.66 \pm 6.54}$ | $\mathbf{77.8 \pm 5.93}$ | $\mathbf{68.23 \pm 10.07}$ | $81.99 \pm 2.08$ |
| | IGN | $90.55 \pm 8.7$ | $77.2 \pm 4.73$ | $66.17 \pm 6.54$ | $\mathbf{83.19 \pm 1.11}$ |

Table 5: Ablation analysis on the two QM9 dataset views. Mean absolute error is reported in both.

| | NMP-Edge split,units, and architecture | | | IGN split, units, and architecture | | |
|---|---|---|---|---|---|---|
| Target | Full | No $h_v^0$ | No hypernetwork | Full | No $X_0$ | No hypernetwork |
| $\mu$ | 0.031 | 0.033 | **0.024** | **0.157** | 0.163 | 0.171 |
| $\alpha$ | 0.066 | 0.455 | **0.065** | 0.325 | 0.332 | **0.318** |
| $\varepsilon_{HOMO}$ | 26.5 | 26.7 | **25.9** | **0.00203** | 0.00223 | 0.00218 |
| $\varepsilon_{LUMO}$ | 23.4 | 23.99 | **22.5** | **0.00228** | 0.00229 | 0.00244 |
| $\Delta\varepsilon$ | 41.916 | **34.44** | 42.934 | **0.00306** | 0.00311 | 0.00341 |
| $\langle R^2 \rangle$ | **0.057** | 0.183 | 0.197 | 13.9 | **13.6** | 13.8 |
| ZPVE | **1.49** | 2.16 | 1.51 | 0.00049 | 0.00045 | **0.00045** |
| $U_0$ | **7.27** | 7.71 | 9.09 | **0.093** | 0.099 | 0.102 |
| $U$ | **6.99** | 7.73 | 9.64 | **0.092** | 0.0986 | 0.103 |
| $H$ | **7.17** | 7.40 | 9.36 | **0.093** | 0.096 | 0.104 |
| $G$ | **7.99** | 9.05 | 9.58 | **0.093** | 0.0997 | 0.102 |
| $C_v$ | 0.026 | 0.043 | **0.025** | 0.180 | **0.163** | 0.180 |

## REFERENCES


Marjan Albooyeh, Daniele Bertolini, and Siamak Ravanbakhsh. Incidence networks for geometric deep learning. *arXiv preprint arXiv:1905.11460*, 2019.

Brandon Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *arXiv preprint arXiv:1906.04015*, 2019.

Erdal Arikan. Channel polarization: A method for constructing capacity-achieving codes. In *2008 IEEE International Symposium on Information Theory*, pp. 1173–1177. IEEE, 2008.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1993–2001, 2016.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, pp. 523–531, 2016.

Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

Andrew Brock, Theo Lim, J.M. Ritchie, and Nick Weston. SMASH: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rydeCEhs-`.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Sebastian Cammerer, Tobias Gruber, Jakob Hoydis, and Stephan ten Brink. Scaling deep learning-based decoding of polar codes via partitioning. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6. IEEE, 2017.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pp. 3844–3852, 2016.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1): 21–28, 1962.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pp. 347–352. IEEE, 1996.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Tobias Gruber, Sebastian Cammerer, Jakob Hoydis, and Stephan ten Brink. On deep learning-based channel decoding. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6. IEEE, 2017.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Katja Hansen, Franziska Biegler, Raghunathan Ramakrishnan, Wiktor Pronobis, O Anatole Von Lilienfeld, Klaus-Robert Muller, and Alexandre Tkatchenko. Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space. *The journal of physical chemistry letters*, 6(12):2326–2331, 2015.

Michael Helmling, Stefan Scholl, Florian Gensheimer, Tobias Dietz, Kira Kraft, Stefan Ruzika, and Norbert Wehn. Database of Channel Codes and ML Simulation Results. `www.uni-kl.de/channel-codes`, 2019.

Bing Huang and O Anatole Von Lilienfeld. Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity, 2016.

Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2186–2195, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/ivanov18a.html`.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pp. 667–675, 2016.

Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1219–1228, 2018.

Peter Bjørn Jørgensen, Karsten Wedel Jacobsen, and Mikkel N Schmidt. Neural message passing with edge updates for predicting properties of molecules and materials. *arXiv preprint arXiv:1806.03146*, 2018.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8): 595–608, 2016.

Hyeji Kim, Yihan Jiang, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Deepcode: Feedback codes via deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 9436–9446, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Scott Kirklin, James E Saal, Bryce Meredig, Alex Thompson, Jeff W Doak, Muratahan Aykol, Stephan Rühl, and Chris Wolverton. The open quantum materials database (oqmd): assessing the accuracy of dft formation energies. *npj Computational Materials*, 1:15010, 2015.

Benjamin Klein, Lior Wolf, and Yehuda Afek. A dynamic convolutional layer for short range weather prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4840–4848, 2015.

Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *arXiv preprint arXiv:1905.11136*, 2019a.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International Conference on Machine Learning*, pp. 4363–4371, 2019b.

Grégoire Montavon, Katja Hansen, Siamac Fazli, Matthias Rupp, Franziska Biegler, Andreas Ziehe, Alexandre Tkatchenko, Anatole V Lilienfeld, and Klaus-Robert Müller. Learning invariant representations of molecules for atomization energy prediction. In *Advances in Neural Information Processing Systems*, pp. 440–448, 2012.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. *arXiv preprint arXiv:1903.02541*, 2019.

Eliya Nachmani and Lior Wolf. Hyper-graph-network decoders for block codes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Eliya Nachmani, Yair Be'ery, and David Burshtein. Learning to decode linear codes using deep learning. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 341–346. IEEE, 2016.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023, 2016.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1:140022, 2014.

G. Riegler, S. Schulter, M. Rther, and H. Bischof. Conditioned regression models for non-blind single image super-resolution. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 522–530, Dec 2015. doi: 10.1109/ICCV.2015.67.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

James E Saal, Scott Kirklin, Muratahan Aykol, Bryce Meredig, and Christopher Wolverton. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (oqmd). *Jom*, 65(11):1501–1509, 2013.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pp. 991–1001, 2017a.

Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8:13890, 2017b.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Chieh-Fang Teng, Ching-Chun Liao, Chun-Hsiang Chen, and An-Yeu Andy Wu. Polar feature based deep architectures for automatic modulation classification considering channel fading. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 554–558. IEEE, 2018.

Bane Vasić, Xin Xiao, and Shu Lin. Learning to decode ldpc codes with finite-alphabet message passing. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9. IEEE, 2018.

Logan Ward, Ruoqian Liu, Amar Krishna, Vinay I Hegde, Ankit Agrawal, Alok Choudhary, and Chris Wolverton. Including crystal structure attributes in machine learning models of formation energies via voronoi tessellations. *Physical Review B*, 96(2):024104, 2017.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374. ACM, 2015.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rkgW0oA9FX.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.