We would like to thank the reviewers for their constructive comments. Below, we try to respond to their main comments.

R1: **Quantitative evaluation**. Indeed, a quantitative evaluation would improve the significance of the empirical results. Note that each motif exhibits some distinct properties and can be considered as a graph-feature. For instance, a star graph contains a single node with high degree. The caveman graphs contain many triangles, the ladder graphs consist of cycles of size 4, etc. For each dataset, we plan to measure how much these properties are satisfied by the learned hidden graphs and we will report the results in the revised manuscript.

R1: **Comparing against k-step RW**. We have started evaluating the k-step RW kernel (the implementation contained in the graphkernels package) on the 10 datasets. We obtained the following average accuracies on MUTAG, ENZYMES and NCI1: 72.39 (± 6.9), 19.00 (± 4.4) and 54.06 (± 1.6), respectively. We observed that computing the kernel matrix on the larger datasets (DD, REDDIT-BINARY, REDDIT-MULTI-5K, COLLAB) takes more than 1 day or requires large amounts of memory (more than 16GB of RAM). Thus, it seems that the k-step RW kernel is fairly weak and suffers from time/memory issues. With this in mind, we are not sure if it is worth adding this baseline to the paper.

R1: **Large graph datasets**. This is definitely on our agenda for future work. We plan to evaluate the proposed architecture on the QM9 dataset which contains more than 100k samples.

R2: **Fully connected learned graphs**. In our implementation, a hidden graph of order n is associated with $n(n-1)/2$ trainable parameters. We do not directly treat these values as the weights of the edges between the different pairs of nodes, but we first apply the ReLU activation function. Therefore, all the negative values are set equal to 0, and the corresponding edges are essentially removed. Even though the learned graphs can be complete, we empirically observed that in most cases, they are fairly sparse. We will make this clear in the revised manuscript.

R2: **Feature space of kernels vs. that of proposed model**. Indeed, a graph kernel maps graphs to some Hilbert space where each dimension typically corresponds to some substructure (e.g., shortest path of specific length, a specific subtree, etc). This space is different from the one to which our model maps graphs. In our case, each dimension corresponds to the "similarity" of the input graph and some hidden graph. Furthermore, since the proposed model is end-to-end trainable, this space is not fixed, but it depends on the structure of the hidden graphs.

R2, R3: **Empirical performance**. It is true that the empirical results are not very impressive in general. Even though the proposed model does not provide a new state-of-the-art in graph classification, still it outperforms the majority of the GNN baselines on most datasets. We agree that the main strength of the paper is in the novelty of the proposed architecture, not in the empirical performance (though we believe that it is not that bad).

R2: **Papers that first introduced GNNs**. We thank the reviewer for pointing us to these papers. We will rephrase the sentence and cite the above papers.

R3: **Comparing against explainability techniques for GNNs**. We should first mention that our paper is different from these works in that the main focus of the proposed model is not on providing interpretable explanations for its predictions, but on dealing with graph-level supervised learning tasks. Note also that the methods proposed in these 3 papers are mainly applied as a post-processing step: they take a trained MPNN and its predictions, and they return an explanation of these predictions. On the other hand, in the case of the proposed model, these explanations come as a byproduct of the learning process. Comparing against these methods seems thus to be out of the scope of this paper.

Furthermore, note that the explanations generated by the proposed model are similar to those of the method presented in the third paper (both belong to the family of model-level methods). Note that the third paper has not been published yet and was posted on arXiv 2 days before the submission deadline of NeurIPS. Therefore, comparing against this method was also practically infeasible.

R4: **MPNNs ignore edge information**. We agree with the reviewer that it is graph-pooling that treats graphs as sets/multisets of node representations. Since the graph structure (e.g., subtree patterns) is encoded into these representations, MPNNs (message passing along with graph-pooling) do not ignore edge information, but they take it into account. We will rephrase our claim as suggested by the reviewer.

R4: **Experiments with different sizes of hidden graphs**. This is mainly due to computational reasons. In our implementation, the parameters of all hidden graphs are combined into a single trainable matrix. This allows us to perform operations such as batch matrix multiplications that benefit greatly from GPUs. To employ hidden graphs of different sizes, we would need to have more than one trainable matrices, and that would slightly increase the running time of the model.

R4: **Comparing against DDGK**. This paper is indeed related to our work. One major difference is that our model is supervised, while DDGK is unsupervised. We will discuss the difference between this approach and our work in the related work section, as suggested. The source code of DDGK is also publicly available, and we have started evaluating it on the 10 graph classification datasets. We will report the results in the revision.