Operation Embeddings for Neural Architecture Search - Supplementary Material -

Michail Chatzianastasis[*†]
National Technical University of Athens
mixalisx97@gmail.com

George Dasoulas[†]
École Polytechnique
george.dasoulas1@gmail.com

Georgios Siolas
National Technical University of Athens
gsiolas@islab.ntua.gr

Michalis Vazirgiannis
Ecole Polytechnique
mvazirg@lix.polytechnique.fr

1. Model Details

In this section we provide more details about the baseline model D-VAE [4], We also further describe the incorporation of our proposed method into the D-VAE, that produces DVAE-EMB model, and finally we prove that DVAE-EMB can injectively encode the computations on DAGs.

D-VAE. D-VAE is a graph-based variational autoencoder for Directed Acyclic Graphs (DAGs). It uses a message-passing graph neural network to encode the graphs using an asynchronous message passing schema, following the topological ordering of the DAG. Firstly, for every node u, it aggregates the messages from its neighbors, using an aggregation function A

$$h_u^{in} = A(\{Cat(h_v, x_v) : (v \to u)\}) \tag{1}$$

, where x_v is the **one-hot vector** of node $v's$ type, and Cat is a concatenation operation. Secondly, it update the representation of every node u, based on the incoming aggregated message from its neighbors and the one-hot vector x_u of node $u's$ type,

$$h_u = U(h_u^{in}, x_u). \tag{2}$$

In contrast with simultaneous message passing schemas, D-VAE update the hidden states of the nodes following the topological ordering of the DAG. This asynchronous message passing scheme can effectively encode the computations on DAGs, but the one-hot vector representation of the operations limits the expressivity of the model.

As the goal is to perform optimization in the continuous learned space, the encoder must map different architectures to different representations h_G. To achieve this, *the aggregation function A and the update function U must be injective*, as noted in Theorem 2 in [4]. Also, the encoder should be invariant to node permutations, such that isomorphic graphs, that represent the same architecture, be mapped in the same representation. To achieve this, *the aggregation function must be permutation invariant*, as noted in Theorem 1 in [4]. To model these two functions, they used a gated sum as an aggregation function:

$$h_u^{in} = \sum_{u \to v} g(Cat(h_v, x_v)) \odot m(Cat(h_v, x_v)), \tag{3}$$

where m is a mapping network and g is a gating network and a gated recurrent unit (GRU)[1] as an update function:

$$h_u = GRU(x_u, h_u^{in}) \tag{4}$$

.

DVAE-EMB The model DVAE-EMB replaces the one-hot vectors in D-VAE, with our proposed operation embeddings approach. The aggregation function 1 and the update function 2 are transformed as follows:

$$h_u^{in} = A(\{Cat(h_v, O(x_v)) : (v \to u)\}) \tag{5}$$

,

$$h_u = U(h_u^{in}, O(x_u)), \tag{6}$$

where $O(x_u)$ is the operation embedding of node's u type. The equations 3,4 are now modified as follows:

$$h_u^{in} = \sum_{u \to v} g(Cat(h_v, O(x_v))) \odot m(Cat(h_v, O(x_v))) \tag{7}$$

$$h_u = GRU(O(x_u), h_u^{in}) \tag{8}$$

.

[*]Work done as intern at École Polytechnique in Data Science and Mining (DaSciM) team group.
[†]Equal Contribution

Since our operation embedding function O is injective, the update and the aggregation functions remain injective, as the composition of injective functions is injective. Therefore Theorems 1,2 still holds for DVAE-EMB and consequently our encoder can injectively encode the computations on DAGs.

2. Training Details

In order to have a fair comparison, we use the same settings from Zhang et al. [4] to train our models (DVAE-EMB, GCN-EMB). For the baselines models, we use the reported results from Zhang et al. [4]. We set the dimensionality of the operation embeddings to be 3 for both models.

For DVAE-EMB we employ the strategy described in Section 3.3. Specifically, we fully-train the model for 4 iterations, for 300 epochs in each iteration. In the first iteration we initialize the operation embeddings from a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. In the next iterations, we initialize the operation embeddings, using the output of the last epoch in the previous iteration. Using this strategy, we observe an increasing performance of the autoencoder, as the operation embeddings are trained for more epochs and capture more effectively the relations between the operations.

For GCN-EMB we just initialize the operation embeddings from a normal distribution, and train the model for 300 epochs without extra iterations. Note that both DVAE-EMB and GCN-EMB achieve better results from their counterparts (DVAE,GCN) even from the first iteration of the operation embeddings.

All models are implemented using Pytorch library [3]. The code is available at https://anonymous.4open.science/r/75a8dbc2-7fe3-4d07-8f55-7856b8a67829/.

3. Architecture Performance and Graph Properties - Experiment Details

In this experiment, we investigate the relation between the performance of the architecture and its corresponding graph structure as described in Section 3.1. Specifically, two graph properties, the *average path length* and the *clustering coefficient* reveal a correlation with the architecture performance.

Average path length is defined as the average shortest path distance between all possible pairs of network nodes. We calculate average path length of a graph G using the following formula:

$$L_G = \frac{1}{n \cdot (n-1)} \cdot \sum_{i \neq j} d(u_i, u_j), \tag{9}$$

where $d(u_i, u_j)$ is the shortest distance between the nodes u_i and u_j. Assume that $d(u_1, u_2) = 0$ if there is no path from u_i to u_j.

Clustering coefficient measures the probability that the adjacent vertices of a vertex are connected. We calculate this property using the igraph software [2]. Specifically, we measure the global clustering coefficient which is the the ratio of the triangles and the connected triples in the graph. Because we have directed graphs the direction of the edges is ignored.

References

[1] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. 1

[2] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. 2

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2

[4] Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-vae: A variational autoencoder for directed acyclic graphs. volume 32, 2019. 1, 2