VARIATIONAL AUTO-ENCODERS WITHOUT GRAPH COARSENING FOR FINE MESH LEARNING

Nicolas Vercheval[1,2]*, Hendrik De Bie*[2]*, and Aleksandra Pižurica*[1]

[1]Department of Telecommunications and Information Processing, TELIN-GAIM,
Faculty of Engineering and Architecture, Ghent University, Belgium
[2]Department of Electronics and Information Systems, Clifford Research Group,
Faculty of Engineering and Architecture, Ghent University, Belgium

ABSTRACT

In this paper, we propose a Variational Auto-Encoder able to correctly reconstruct a fine mesh from a very low-dimensional latent space. The architecture avoids the usual coarsening of the graph and relies on pooling layers for the decoding phase and on the mean values of the training set for the up-sampling phase. We select new operators compared to previous work, and in particular, we define a new Dirac operator which can be extended to different types of graph structured data. We show the improvements over the previous operators and compare the results with the current benchmark on the Coma Dataset.

***Index Terms*—Variational Autoencoder, Geometric Deep Learning, Mesh processing**

I. INTRODUCTION

The increasing amount of data has been a crucial factor for the recent success of deep learning and the availability of new types of data offers now many challenges and opportunities for research. For example, Geometric Deep Learning [2] extends the traditional deep learning framework to non-Euclidean domains such as graphs, and in particular, meshes. Meshes provide a compact graphical representation of surfaces collected from 3D-scans. The node connectivity reflects the surface geometry while their embedding (i.e. the coordinate values of the nodes) describes the shape of objects.

As meshes convey visual information, a natural use of machine learning is to produce new credible samples by reconstructing the joint distribution of all the variables (the embedding). Most of the time a direct calculation is computationally untreatable and thus, in deep learning, generative models are used to approximate the distribution. Variational Auto-Encoders (VAE) [8] are a class of probabilistic graphical models able to learn complicated sample distributions in an unsupervised manner. This is accomplished by inferring a latent representation that is assumed to govern the distribution of the observed data. The latent variables are modeled with a prior on a small latent space, and can be seen as a robust compression of the samples' information.

A recent benchmark has been set by Rajam et al. [11] where they successfully encode the Coma Dataset, introduced in the same paper, in just 8 latent variables. They did so by using Chebyshev polynomials [5] of the Laplacian and combining it with graph down-sampling for the encoder and up-sampling for the decoder. The progressive down-sampling and up-sampling of the nodes is the current standard to naturally leverage the structure of the graph. The dataset is defined on a common graph which is critical when using spectral methods [2], and their work was competitive in reconstruction against the linear 3D Morphable Models [1] normally used there, while allowing for non-linear shape generation and interpolation.

In Surface Networks [10], Kostrikov et al. introduce a network influenced by PointNet [16] for a task of temporal prediction where they only rely on differential operators to enforce the local structure of the graphs and 1×1 Convolution to mix the graph features. The selected operators are the Laplacian with cotangent weights and the Dirac operator introduced in [3] and are calculated from the samples during the preprocessing phase. This type of architecture comes with some benefits due to its versatility: as it is closer to a standard multilayer perceptron, it is very natural to adapt standard practice like Batch normalization [7] and ResNets [6] and this allows for deeper networks compared to standard spectral approaches. In the same paper they also introduce a VAE version of the network which was used to reconstruct meshes sampled from the MNIST dataset and prove the stability of the operators given different samplings.

In spirit of the PointNet approach, Surface Networks use global pooling on the graph features to reduce the dimensionality and create the latent variables while, symmetrically, recreating the features by expanding the latent variables onto the graph. While adequate on a toy dataset, this architecture struggles to reproduce finer meshes with a larger amount of nodes. Furthermore, storing data dependent operators (the Dirac in particular) poses huge memory requirements.

In this article we show how to exploit the fixed graph structure of a dataset by initiating the decoder with the mean values of the training set. Starting with an initial shape provides spatial context without having to parametrize the deformation in the preprocessing phase [13, 14] nor its degrees of freedom [12]. The resulting output is also free from artifacts that typically need post-processing such as in [4]. The initialised decoder is able to generate fine smooth meshes which were unachievable with the previous architecture from [10], and we demonstrate its effectiveness by accurately encoding the Coma dataset.

Additionally, the proposed operators only depend on the common adjacency matrix: we replace the original Laplacian with an adjacency version, generalizing the approach of [9], and introduce a new version of the Dirac operator which squares to the normalized Laplacian and can be used for any chordal graph. As the adjacency matrix is shared, we can then use the same operator for all the samples. This not only completely solves the memory burden but prevents the network from overfitting.

Hence, we develop a viable approach for reconstructing fine meshes from a very low dimensional latent space without intermediate graph coarsening. The results are encouraging, in comparison with the best reported ones based on alternative architectures with graph coarsening.

In Section II, we review the previous architecture from [10] and its previously used operators. In Section III, we present the proposed decoder and define the adjacency Dirac Operator. We evaluate the performance on the Coma Dataset in Section IV and conclude the paper in Section V.

II. PRELIMINARIES

II-A. Operators

A mesh is a data structure comprising the embedding of the nodes $\phi\colon \mathbb{V} \to \mathbb{R}^3$, and a list of triangular faces $\mathbb{F}$. Given a node v, its dual area $a(v)$ is a third of the areas A of the surrounding triangles. Let W be the symmetric matrix of cotangent weights and a be the diagonal matrix of the dual areas. The degree matrix D is a diagonal matrix such that $D_{i,i} = \sum_j W_{i,j}$. The normalized graph Laplacian with cotangent weights, commonly used in mesh processing, is defined as: $\Delta = a^{-1}(D - W)$.

The symmetric Laplacian is defined instead as $\tilde{\Delta} = 1 - \tilde{L} = 1 - D^{1/2}\tilde{W}D^{1/2}$, where $\tilde{W}$ is the adjacency matrix and $\tilde{L}$ will be called the adjacency Laplacian.

While most of the spectral methods rely on some version of the Laplacian, the Dirac operator was shown to be a valid alternative for high curvature surfaces [10].

The Dirac operator introduced by [3] is a discrete differential operator on quaternion-valued functions (the embedding is immersed in their imaginary part) between the graph and the dual graph. The dual graph is constructed from the faces of the original graph, where two connected (dual) nodes correspond to two adjacent faces. Note that the dual graph of the dual graph is (isomorphic to) the graph itself.

On an oriented triangular mesh, the Dirac operator is defined on a quaternionic function $\lambda\colon \mathbb{V} \to \mathbb{H}$ as follows:

$$(Di_\phi\lambda)_F = -\frac{e_{(1,0)} \cdot \lambda(v_2) + e_{(2,1)} \cdot \lambda(v_0) + e_{(0,2)} \cdot \lambda(v_1)}{2A(F)},$$

where $e_{(i,j)} = \phi(v_i) - \phi(v_j)$ is an edge of $F = (v_0, v_1, v_2) \in \mathbb{F}$ and $\cdot$ is the quaternionic multiplication.

In the neural network it is used in combination with its adjoint operator $Di^A = a^{-1}Di^tA$ which maps the values on the dual nodes to the original graph.

It can be shown that $\Re(Di_\phi^A Di_\phi) = \Delta$.

II-B. General architecture and Encoder

The Variational Auto-Encoders are composed of an encoder and of a decoder, both approximated with neural networks. The Decoder takes a noisy version of the encoded samples during training which makes it more robust while respecting the probabilistic interpretation of the model (the reparameterization trick) [8].

The networks are mainly composed of "1×1 convolution" layers, concatenation and matrix multiplication with the operators, so that we get the following combined output:

$$x_j^{t+1} = \sum_i S_{j,i}^t x_i^t + R_{j,i}^t \Delta x_i^t \tag{1}$$

when we use the Laplacian acting on the feature vector x_i^t and

$$\begin{aligned} x_j^{t+1} &= \sum_i S_{j,i}^t x_i^t + R_{j,i}^t Di \cdot y_i^t \\ y_j^{t+1} &= \sum_i T_{j,i}^t y_i^t + V_{j,i}^t Di^A \cdot x_i^t \end{aligned} \tag{2}$$

when using the Dirac, where y_j^{t+1} are the values of the dual nodes and S, R, T, V are learned weight matrices.

The combined layers in Eq (2) form a ResNet block. To compare the two structures and have the same number of parameters, the Laplacian layers are repeated twice in each block. The net is trained using Batch normalization as preactivation, ELU activation and the Adam optimizer.

The encoder starts by expanding the (x, y, z) coordinates into 32 graph features with 1×1 convolutions, and is composed of a sequence of two blocks which are collapsed into real values through global average pooling. Finally the means and the variances of the posteriors are approximated with dense layers.

Variational Auto-Encoders maximize the (marginal) likelihood $p_\theta(x)$ and minimize the KL divergence between the posterior $p_\theta(z|x)$ and its approximation $q_\phi(z|x)$, which is inferred by the encoder. This is done by maximizing the *evidence lower bond* ELBO [8]. The ELBO is calculated as follows:

$$\text{ELBO} = \underbrace{\log p_\theta(x|z)}_{\text{reconstruction loss}} + \underbrace{\log p_\theta(z) - \log q_\phi(z|x)}_{\text{regularization terms}}.$$

III. METHOD

Surface Networks proved to be stable on different samplings [10] but it was only able to reconstruct the gray level of the Mnist Dataset. Reconstructing a fine mesh is a much harder challenge because in order to obtain good global performances (on a node-wise statistics) the network has to learn how to exploit all the information of the surrounding nodes. Visually, this translates into smooth reconstructions, where the neighbourhood of a node is encouraged to be as flat as possible.

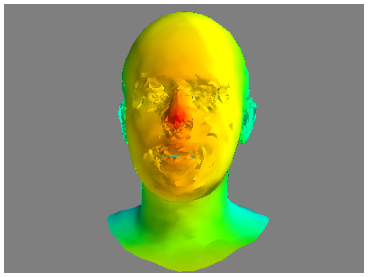

Fig. 1. Even after a long training and an added dense layer to enable a direct influence of the latent variables to the nodes, the reconstruction of the original architecture is not smooth.

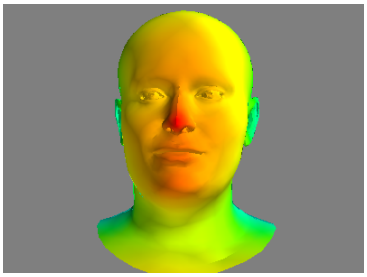

Fig. 2. A smooth recontruction of a mesh using the new decoder and $\widetilde{Di}$ as operator.

Our proposed Decoder has a simple and effective solution, which naturally extends the previous architecture. Furthermore, our proposed operators are common to all the samples, avoiding preprocessing and high memory requirements, making for a model which is very easy to deploy.

III-A. Decoder

We observe that Surface Networks fail to converge on high-resolution meshes and are attracted to the barycenter of the sample. As the graph structure is only communicated through the use of the operators as matrix multiplication we infer that the residual networks (which allow for a slow activation of the inner layers) struggle to activate the operators and thus enforce local information. On the other hand upsampling using an initial dense layer, helps recover the shape but also highlights that the local property of smoothness cannot be easily learned (Fig. 1). To overcome this, we propose to initialize the decoder with a smooth mesh so that smoothness could easily be preserved. To take full advantage of the fixed structure we decide to use the mean shape, the mean embedding of the nodes over the entire training set. This is possible when meshes are consistently sampled and ordered, even if they have different shapes.

The mean shape and the latent variables are given to the decoder as a multimodal input. The proposed architecture joins the two initial inputs with a tensor multiplication as in Fig. 3. The intuitive idea is that the latent variables would have a linear impact on learned features of the mean shape, giving more or less importance to different representations. When decoding with the original operators we use the mean operator which is similarly calculated. In this way, the decoder is independent from the latent variables and allows us to generate samples directly from random noise.

The decoder is then composed of a 1×1 convolution for the mean shape and a dense layer for the latent variables, joined with tensor multiplication and followed by three blocks. The final 1×1 convolution reconstructs the (x, y, z) coordinates. The number of features is 32.

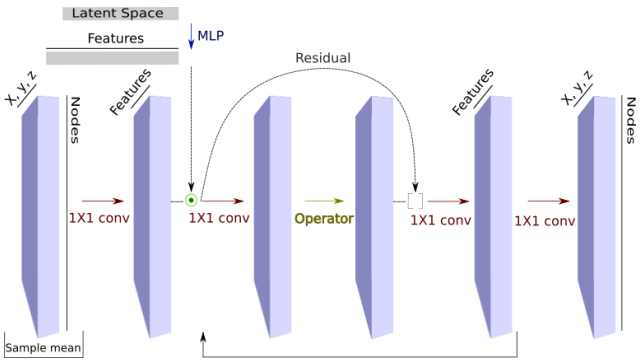

Fig. 3. Proposed architecture of the Decoder. The 1×1 Convolution can be seen as dense layer on the feature graphs and is preceded by batch normalization. The arrow in the bottom represents half a block, and we denote with $\odot$ and [ ] respectively, the element-wise product and feature concatenation.

III-B. Adjacency operators

As an alternative to the extrinsic Dirac operator, we introduce the adjacency Dirac operator and we define it only through the adjacency matrix as $\widetilde{Di} = 2D^{1/2}Di_{\tilde{\phi}}$ where $\tilde{\phi}$ maps each cycle (triangle) to $\{v_0, v_1, v_2\}$ and where

$$v_0 = (\frac{1}{\sqrt{2}}, 0, 0),\ v_1 = (0, \frac{1}{\sqrt{2}}, 0),\ v_2 = (0, 0, \frac{1}{\sqrt{2}}).$$

Like the extrinsic version, defining $\widetilde{Di}^A = \widetilde{Di}^T$ it can be shown that

$$\Re(\widetilde{Di}^A \widetilde{Di}) = \tilde{\Delta}.$$

This operator can also be applied to more general chordal graphs. Thus, we replace Di in Eq (2) with the adjacency Dirac operator $\widetilde{Di}$ and compare the results. Similarly, we replace Δ in Eq (1) with $\tilde{L} = \tilde{\Delta} + 1$. This leads to a more flexible version of Graph Convolutional Networks [9], which uses a renormalized Laplacian in a similar way, but with the added benefits coming from Resnets.

IV. EXPERIMENTAL RESULTS

Here we report the results on the Coma Dataset [11] which is composed of temporal sequences of extreme facial expressions by 12 subjects, recorded as meshes. The test dataset has been extracted uniformly from all the subjects as explained in the interpolation experiment [11].

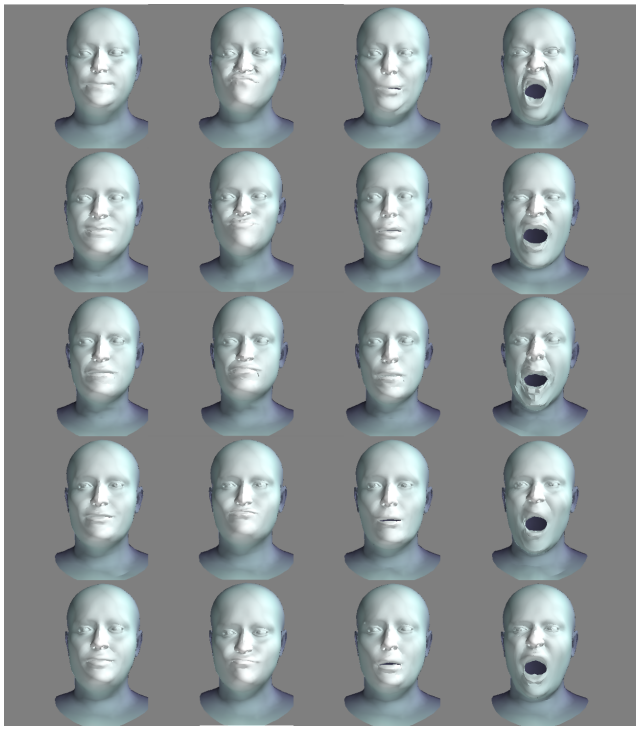

Fig. 4. Reconstructions from extreme expressions of one of the subjects. From top to bottom: Inputs, Δ, Di, $\widetilde{Di}$, $\tilde{L}$. The reconstruction is generally harder the more extreme the expressions.

This dataset shares the same graph structure, is consistently sampled and, at the same time, particularly challenging. Extreme

expressions often have high curvatures which are harder to approximate and the unnatural stretching creates more variability compared to other datasets of recorded faces.

We report experimental results that compare the reconstruction error, using the original Dirac and Laplacian operators and their adjacency versions, and we visually test their ability to generate new samples.

In general, all the operators correctly reconstruct the subject and expressions (Fig. 4), with some difficulties for the Dirac Operator. The original Laplacian is able to correctly recreate many details of the more extreme expression but its performance is severely affected from the errors in the neck (Fig. 5).

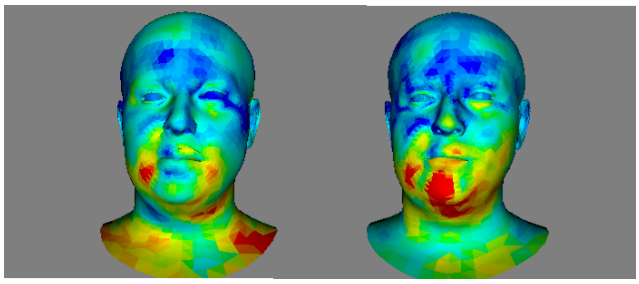

Fig. 5. The same mesh reconstructed by the Δ and $\tilde{L}$. The warmer colors highlight the larger errors.

The Laplacian networks have been trained for 1000 epochs, while the Dirac networks, whose layers are harder to compute, have been trained for an equivalent amount of time (around 200 epochs). The memory cost of the experiment with the original operator (120 Gigabyte in case of the Dirac operator) was considerable. This is not a problem with the adjacency versions.

We refer to the method of Ranjan et al. [11] as Coma, and we use the same performance metrics as in that work: the percentage of reconstructed nodes within a millimeter of euclidean error and the average euclidean error of the nodes. Table I shows that the adjacency versions of the operators offer a clear improvement over the ones previously used. In particular the adjacency Laplacian approaches the performance of Coma, while having less weights. It also compares favourably to all the other approaches listed in a recent paper [15].

Since the original operators are calculated from the samples, they are able to express more information, but this also leads to some instability. This might be the cause of the gap between the training and validation losses when using the original operators, which, with the adjacency versions, present hardly any discrepancy. One explanation is that the nets are harder to optimize when their layers use operators who depend on the samples themselves.

When using the adjacency Laplacian, the latent space is robust and able to represent the semantic information of the meshes. This can be shown by sampling the latent space (Fig. 6) and by interpolating two different subjects and expressions (Fig. 7).

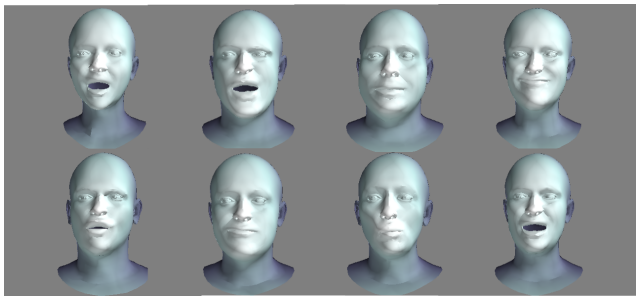

Fig. 6. Generated meshes using $\tilde{L}$. The expressions above are sampled in the latent space from a normal distribution centered at 0 with $\sigma = 0.3$. All meshes are realistic and mix traits and expressions from various subjects.

TABLE I

EUCLIDEAN ERROR AND PERCENTAGE OF CORRECT (<1 MM) NODE RECONSTRUCTIONS BY OPERATOR. THE ADJACENCY OPERATORS OUTPERFORM THE ONES USED PREVIOUSLY AND APPROACH THE BENCHMARK. THE VANILLA SURFACE NETWORKS DID NOT CONVERGE.

Networks	***Operator***	***Error (mm)***	***% correct***	***# Weights***
Proposed	Di	1.90 ± 0.15	21.9	28,660
	$\widetilde{Di}$	1.47 ± 0.10	42.3	28,660
	Δ	2.08 ± 0.36	21.5	28,660
	$\tilde{L}$	1.10 ± 0.05	57.7	28,660
DEMEA[a]	$\tilde{\Delta}$	1.49	/	/
Coma[b]	$\tilde{\Delta}$	0.845 ± 0.99	72.6	33,856

[a] As reported in [15] (no confidence interval was provided).

[b] As reported in [11].

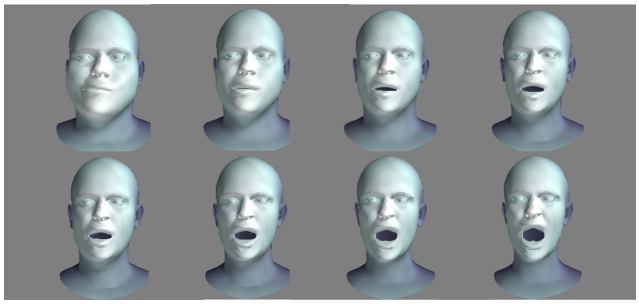

Fig. 7. The upper left and lower right are a reconstruction of two subjects, the other meshes have been generated by linearly interpolating them in the latent space. The generated interpolation is gradual and without any deformities.

V. CONCLUSIONS

In this paper we introduced a new decoder for a Variational Auto-Encoder which is able to learn fine meshes without intermediate graph coarsening, by combining spectral methods, residual nets and global pooling. New operators do not need preprocessing or high memory requirements and show a tangible improvement over the previous work, approaching recent benchmarks that employ graph down- and up-sampling. The resulting model is light, easy to train and deploy. Future work might determine more effective methods to combine the starting smooth mesh with the latent variables.

6. REFERENCES

[1] B. Amberg, R. Knothe, and T. Vetter, "Expression invariant 3d face recognition with a morphable model" In Conference on Automatic Face and Gesture Recognition 2008.

[2] M. Bronstein, J. Bruna,Y. LeCun,A. Szlam , P. Vandergheynst, "Geometric deep learning: going beyond euclidean data" IEEE SIG PROC MAG 2016.

[3] K. Crane, U. Pinkall, P. Schröder, "Spin transformations of discrete surfaces" ACM Transactions on Graphics (TOG) 2011.

[4] T. Groueix, M. Fisher, V. Kim, B. Russell, M. Aubry, "AtlasNet: A Papier-Mch Approach to Learning 3D Surface Generation" CVPR 2016.

[5] M. Defferrard, X. Bresson, P. Vandergheynst "Convolutional neural networks on graphs with fast localized spectral filtering" Advances in Neural Information Processing Systems 2016.

[6] K. He ,X. Zhang, S. Ren, J. Sun "Deep Residual Learning for Image Recognition" arXiv:1512.03385 2015

[7] S. Ioffe, C. Szegedy "Batch normalization: accelerating deep network training by reducing internalcovariate shift" arXiv:1502.03167 2015.

[8] D. P. Kingma, M. Welling, "An Introduction to Variational Autoencoders" arXiv:1906.02691 2019.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks". ICLR 2017.

[10] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, J. Bruna, "Surface Networks" 2018 IEEE Conference on Computer Vision and Pattern Recognition 2018.

[11] A. Ranjan, T. Bolkart, S. Sanyal, M. J. Black, "Generating 3D faces using Convolutional Mesh Autoencoders" ECCV 2018.

[12] A. Sinha, A. Unmesh, Q. Huang and K. Ramani, "SurfNet: Generating 3D Shape Surfaces Using Deep Residual Networks" CVPR 2017.

[13] Q. Tan, L. Gao, Y. K. Lai, J. Yang, S. Xia, "Mesh-based Autoencoders for Localized Deformation Component Analysis" AAAI Conference on Artificial Intelligence 2018.

[14] Q. Tan, L. Gao, Y. K. Lai, S. Xia, "Variational Autoencoders for Deforming 3D Mesh Models" CVPR 2018.

[15] E. Tretschk, A. Tewari, M. Zollhfer, V. Golyanik, C. Theobalt "DEMEA: Deep Mesh Autoencoders for Non-Rigidly Deforming Objects" unpublished 2019.

[16] C. R. Qi, H. Su, K. Mo, L. J. Guibas "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation" arXiv:1612.00593 2016