CONVOLUTIONAL MESH AUTOENCODERS FOR 3D FACE REPRESENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional neural networks (CNNs) have achieved state of the art performance on recognizing and representing audio, images, videos and 3D volumes; that is, domains where the input can be characterized by a regular graph structure. However, generalizing CNNs to irregular domains like 3D meshes is challenging. Additionally, training data for 3D meshes is often limited. In this work, we generalize convolutional autoencoders to mesh surfaces. We perform spectral decomposition of meshes and apply convolutions directly in frequency space. In addition, we use max pooling and introduce upsampling within the network to represent meshes in a low dimensional space. We construct a complex dataset of 20,466 high resolution meshes with extreme facial expressions and encode it using our Convolutional Mesh Autoencoder. Despite limited training data, our method outperforms state-of-the-art PCA models of faces with 50% lower error, while using 75% fewer parameters.

1 INTRODUCTION

Convolutional neural networks (LeCun, 1989) have achieved state of the art performance in a large number of problems in computer vision (Krizhevsky et al., 2012; He et al., 2016), natural language processing (Mikolov et al., 2013) and speech processing (Graves et al., 2013). In recent years, CNNs have also emerged as rich models for generating both images (Goodfellow et al., 2014; Oord et al., 2016) and audio (van den Oord et al., 2016). These successes may be attributed to the multi-scale hierarchical structure of CNNs that allows them to learn translational-invariant localized features. Since the learned filters are shared across the global domain, the number of filter parameters is independent of the domain size. We refer the reader to Goodfellow et al. (2016) for a comprehensive overview of deep learning methods and the recent developments in the field.

Despite the recent success, CNNs have mostly been successful in Euclidean domains with grid-based structured data. In particular, most applications of CNNs deal with regular data structures such as images, videos, text and audio, while the generalization of CNNs to irregular structures like graphs and meshes is not trivial. Extending CNNs to graph structures and meshes has only recently drawn significant attention (Bruna et al., 2013; Defferrard et al., 2016; Bronstein et al., 2017). Following the work of Defferrard et al. (2016) on generalizing the CNNs on graphs using fast Chebyshev filters, we introduce a *convolutional mesh autoencoder architecture* for realistically representing high-dimensional meshes of 3D human faces and heads.

The human face is highly variable in shape as it is affected by many factors such as age, gender, ethnicity etc. The face also deforms significantly with expressions. The existing state of the art 3D face representations mostly use linear transformations (Tewari et al., 2017; Li et al., 2017; Thies et al., 2015) or higher-order tensor generalizations (Vlasic et al., 2005; Brunton et al., 2014a). While these linear models achieve state of the art results in terms of realistic appearance and Euclidean reconstruction error, we show that CNNs can perform much better at capturing highly non-linear extreme facial expressions with many fewer model parameters.

One challenge of training CNNs on 3D facial data is the limited size of current datasets. Here we demonstrate that, since these networks have fewer parameters than traditional linear models, they can be effectively learned with limited data. This reduction in parameters is attributed to the locally invariant convolutional filters that can be shared on the surface of the mesh. Recent work has exploited thousands of 3D scans and 4D scan sequences for learning detailed models of 3D faces

(Cosker et al., 2011; Yin et al., 2006; 2008; Savran et al., 2008; Cao et al., 2014). The availability of this data enables us to a learn rich non-linear representation of 3D face meshes that can not be captured easily by existing linear models.

In summary, our work introduces a convolutional mesh autoencoder suitable for 3D mesh processing. Our main contributions are:

- We introduce a mesh convolutional autoencoder consisting of mesh downsampling and mesh upsampling layers with fast localized convolutional filters defined on the mesh surface.
- We use the mesh autoencoder to accurately represent 3D faces in a low-dimensional latent space performing 50% better than a PCA model that is used in state of the art methods (Tewari et al., 2017) for face representation.
- Our autoencoder uses up to 75% fewer parameters than linear PCA models, while being more accurate on the reconstruction error.
- We provide 20,466 frames of highly detailed and complex 3D meshes from 12 different subjects for a range of extreme facial expressions along with our code for research purposes. Our data and code is located at `http://withheld.for.review`.

This work takes a step towards the application of CNNs to problems in graphics involving 3D meshes. Key aspects of such problems are the limited availability of training data and the need for realism. Our work addresses these issues and provides a new tool for 3D mesh modeling.

2 Related work

Mesh Convolutional Networks. Bronstein et al. (2017) give a comprehensive overview of generalizations of CNNs on non-Euclidean domains, including meshes and graphs. Masci et al. (2015) defined the first mesh convolutions by locally parameterizing the surface around each point using geodesic polar coordinates, and defining convolutions on the resulting angular bins. In a follow-up work, Boscaini et al. (2016) parametrized local intrinsic patches around each point using anisotropic heat kernels. Monti et al. (2017) introduced d-dimensional pseudo-coordinates that defined a local system around each point with weight functions. This method resembled the intrinsic mesh convolution of Masci et al. (2015) and Boscaini et al. (2016) for specific choices of the weight functions. In contrast, Monti et al. used Gaussian kernels with trainable mean vector and covariance matrix as weight functions.

In other work, Verma et al. (2017) presented dynamic filtering on graphs where filter weights depend on the input data. The work however did not focus on reducing the dimensionality of graphs or meshes. Yi et al. (2017) also presented a spectral CNN for labeling nodes which did not involve any dimensionality reduction of the meshes. Sinha et al. (2016) and Maron et al. (2017) embedded mesh surfaces into planar images to apply conventional CNNs. Sinha et al. used a robust spherical parametrization to project the surface onto an octahedron, which is then cut and unfolded to form a squared image. Maron et al. (2017) introduced a conformal mapping from the mesh surface into a flat torus.

Although, the above methods presented generalizations of convolutions on meshes, they do not use a structure to reduce the meshes to a low dimensional space. The proposed mesh autoencoder efficiently handles these problems by combining the mesh convolutions with efficient mesh-downsampling and mesh-upsampling operators.

Graph Convolutional Networks. Bruna et al. (2013) proposed the first generalization of CNNs on graphs by exploiting the connection of the graph Laplacian and the Fourier basis (see Section 3 for more details). This lead to spectral filters that generalize graph convolutions. Boscaini et al. (2015) extended this using a windowed Fourier transform to localize in frequency space. Henaff et al. (2015) built upon the work of Bruna et al. by adding a procedure to estimate the structure of the graph. To reduce the computational complexity of the spectral graph convolutions, Defferrard et al. (2016) approximated the spectral filters by truncated Chebyshev poynomials which avoids explicitly computing the Laplacian eigenvectors, and introduced an efficient pooling operator for graphs. Kipf & Welling (2016) simplified this using only first-order Chebyshev polynomials.

However, these graph CNNs are not directly applied to 3D meshes. Our mesh autoencoder is most similar to Defferrard et al. (2016) with truncated Chebyshev polynomials along with the efficient graph pooling. In addition, we define mesh upsampling layer to obtain a complete mesh autoencoder structure and use our model for representation of highly complex 3D faces obtained state of the art results in realistic modeling of 3D faces.

Learning Face Representations. Blanz & Vetter (1999) introduced the first generic representation for 3D faces based on principal component analysis (PCA) to describe facial shape and texture variations. We also refer the reader to Brunton et al. (2014b) for a comprehensive overview of 3D face representations.

Representing facial expressions with linear spaces has given state-of-the-art results till date. The linear expression basis vectors are either computed using PCA (e.g. Amberg et al., 2008; Breidt et al., 2011; Li et al., 2017; Tewari et al., 2017; Yang et al., 2011), or are manually defined using linear blendshapes (e.g. Thies et al., 2015; Li et al., 2010; Bouaziz et al., 2013). Multilinear models (Vlasic et al., 2005), i.e. higher-order generalizations of PCA are also used to model facial identity and expression variations. In such methods, the model parameters globally influence the shape, i.e. each parameter affects all the vertices of the face mesh. To capture localized facial details, Neumann et al. (2013) and Ferrari et al. (2015) used sparse linear models. Brunton et al. (2014a) used a hierarchical multiscale approach by computing localized multilinear models on wavelet coefficients.

Brunton et al. (2014a) also used a hierarchical multi-scale representation, but their method does not use shared parameters across the entire domain. Jackson et al. (2017) use a volumetric face representation in their CNN-based framework. In contrast to existing face representation methods, our mesh autoencoder uses convolutional layers to represent faces with significantly fewer parameters. Since, it is defined completely on the mesh space, we do not have memory constraints which effect volumetric convolutional methods for representing 3D models.

3 Mesh Operators

We define a face mesh as a set of vertices and edges $\mathcal{F} = (\mathcal{V}, \mathcal{E}, A)$, with $|\mathcal{V}| = n$ vertices that lie in 3D Euclidean space, $\mathcal{V} \in \mathbb{R}^{n\times 3}$. $\mathcal{E}$ defines the edge connections between these vertices. The sparse adjacency matrix $A \in \{0,1\}^{n\times n}$ represents the edge connections, where $A_{ij} = 1$ denotes an edge connecting vertices i and j, and $A_{ij} = 0$ otherwise. The non-normalized graph Laplacian is defined as $L = D - A$ (Chung, 1997), with the diagonal matrix D that represents the degree of each vertex in $\mathcal{V}$ as $D_{ii} = \sum_j A_{ij}$.

The Laplacian can be diagonalized by the Fourier basis $U \in \mathbb{R}^{n\times n}$ (as L is a real symmetric matrix) as $L = U\Lambda U^T$, where the columns of $U = [u_0, u_1, ..., u_{n-1}]$ are the orthogonal eigenvectors of L, and $\Lambda = diag([\lambda_0, \lambda_1, ..., \lambda_{n-1}]) \in \mathbb{R}^{n\times n}$ is a diagonal matrix with the associated real, non-negative eigenvalues. The graph Fourier transform (Chung, 1997) of the mesh vertices $x \in \mathbb{R}^{n\times 3}$ is then defined as $x_\omega = U^T x$, and the inverse Fourier transform as $x = U x_\omega$, respectively.

Fast spectral convolutions. The convolution operator $*$ can be defined in Fourier space as a Hadamard product, $x * y = U((U^T x) \odot (U^T y))$. This is computationally expensive with large number of vertices. The problem is addressed by formulating mesh filtering with a kernel g_θ using a recursive Chebyshev polynomial (Hammond et al., 2011; Defferrard et al., 2016). The filter g_θ is parametrized as a Chebyshev polynomial of order K given by

$$g_\theta(L) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}), \tag{1}$$

where $\tilde{L} = 2L/\lambda_{max} - I_n$ is the scaled Laplacian, the parameter $\theta \in \mathbb{R}^K$ is a vector of Chebyshev coefficients, and $T_k \in \mathbb{R}^{n\times n}$ is the Chebyshev polynomial of order k that can be computed recursively as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$. The spectral convolution can then be defined as (Defferrard et al., 2016)

$$y_j = \sum_{i=1}^{F_{in}} g_{\theta_{i,j}}(L) x_i \in \mathbb{R}^n \tag{2}$$

that computes the jth feature of $y \in \mathbb{R}^{n \times F_{out}}$. The input $x \in \mathbb{R}^{n \times F_{in}}$ has F_{in} features. The input face mesh has $F_{in} = 3$ features corresponding to its 3D vertex positions. Each convolutional layer has $F_{in} \times F_{out}$ vectors of Chebyshev coefficients $\theta_{i,j} \in \mathbb{R}^K$ as trainable parameters.

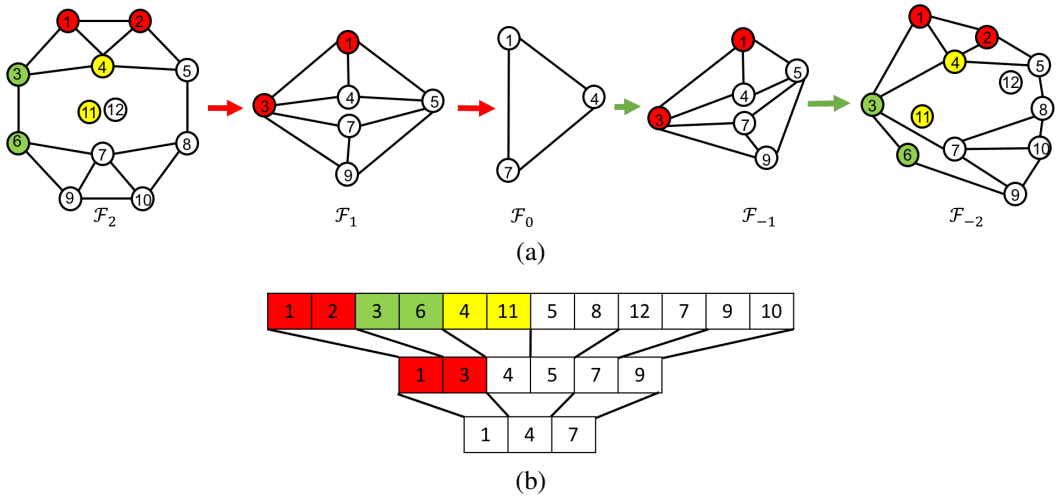


Figure 1: Graph pooling (red arrows) and upsampling (green arrows) operations: (a) The graph $\mathcal{F}_2$ is successively coarsened to $\mathcal{F}_0$ and then upsampled to $\mathcal{F}_{-2}$. A precomputed binary tree (b) is used for both max pooling and upsampling operations. During coarsening, the nodes of the tree are pooled in 1D. Upsampling is done by linear interpolation of the nodes at a level. Fake vertices $\{11,12\}$ are added to balance the tree. During coarsening, adjacent vertices shown with same color are grouped.

Max pooling on a Mesh. The max pooling on the mesh can be performed in two steps. First, the mesh vertices are clustered together using Graclus multilevel clustering algorithm (Dhillon et al., 2007) to obtain multi-level coarser versions of the mesh (Figure 1a). Then, a balanced binary tree (Figure 1(b)) is created and the vertices are rearranged in the tree (Defferrard et al., 2016). The tree is structured such that it allows hierarchical merging of adjacent nodes at coarser levels. The vertices are rearranged such that successive union of adjacent vertices in the tree corresponds to neighboring nodes in the coarsened graph. Fake vertices ($\{11,12\}$ in Figure 1) are added to complete the tree structure as shown in Figure 1.

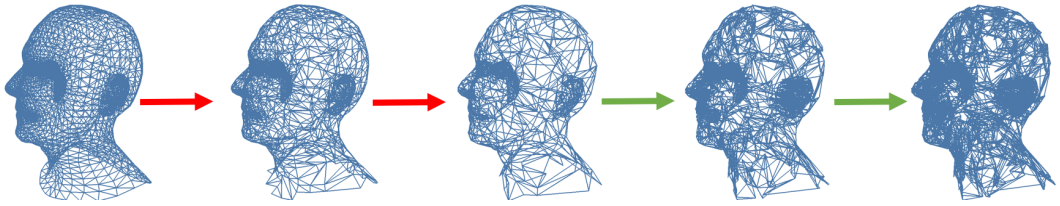


Figure 2: The effect of max pooling (red arrows) and upsampling (green arrows) on 3D face meshes. The reconstructed face after upsampling maintains the overall structure but most of the finer details are lost. The extra vertices that are added in the upsampled mesh are not very consistent.

Mesh Upsampling. For upsampling, we use the same balanced binary tree created for max pooling. To interpolate the vertex positions at finer levels, we linearly interpolate between the values of the adjacent nodes in the tree at a coarser level. As shown in Figure 1, the values of nodes in the mesh $\mathcal{F}_{-1}$ are obtained by linearly interpolating the values of adjacent nodes in $\mathcal{F}_0$. For instance, $\mathcal{F}_{-1}(3) = p(\mathcal{F}_0(1), \mathcal{F}_0(4))$, where p is an interpolating function.

Since, the same tree is used for pooling and upsampling, the mesh structure is preserved during upsampling. However, due to the information loss, the values of the upsampled mesh vertices are not the same. Figure 2 shows the effect of using downsampling and upsampling operations on 3D

Layer	Input Size	Output Size
Convolution	8192×3	8192×16
Max Pool	8192×16	2048×16
Convolution	2048×16	2048×16
Max Pool	2048×16	512×16
Convolution	512×16	512×16
Max Pool	512×16	128×16
Convolution	128×16	128×32
Max Pool	128×32	32×32
Fully Connected	32×32	8

Table 1: Encoder architecture

Layer	Input Size	Output Size
Fully Connected	8	32×32
Upsampling	32×32	128×32
Convolution	128×32	128×32
Upsampling	128×32	512×32
Convolution	512×32	512×16
Upsampling	512×16	2048×16
Convolution	2048×16	2048×16
Max Pool	2048×16	8192×16
Convolution	8192×16	8192×3

Table 2: Decoder architecture

faces. Although, the overall structure is preserved while upsampling, the vertices might rearrange themselves differently due to information loss.

4 Mesh Autoencoder

Now that we have defined the basic operations needed for our neural network in Section 3, we can construct the architecture of the convolutional mesh autoencoder. The structure of the encoder is shown in Table 1. The encoder consists of 4 Chebyshev convolutional filters with $K = 6$ Chebyshev polynomials. Each of the convolutions is followed by a biased ReLU (Glorot et al., 2011). The max pooling layers are interleaved between convolution layers. Each of the max pooling layers reduce the number of mesh vertices by 4 times. The encoder transforms the face mesh from $\mathbb{R}^{n\times 3}$ to an 8 dimensional latent vector using a fully connected layer at the end.

The structure of the decoder is shown in Table 2. The decoder similarly consists of a fully connected layer that transforms the latent vector from $\mathbb{R}^8$ to $\mathbb{R}^{32\times 32}$ that can be further upsampled to reconstruct the mesh. Following the decoder's fully connected layer, 4 convolutional layers with interleaved upsampling layers generated a 3D mesh in $\mathbb{R}^{8192\times 3}$. Each of the convolutions is followed by a biased ReLU similar to the encoder network. Each upsampling layer increases the numbers of vertices by 4x. The Figure 3 shows the complete structure of our mesh autoencoder.

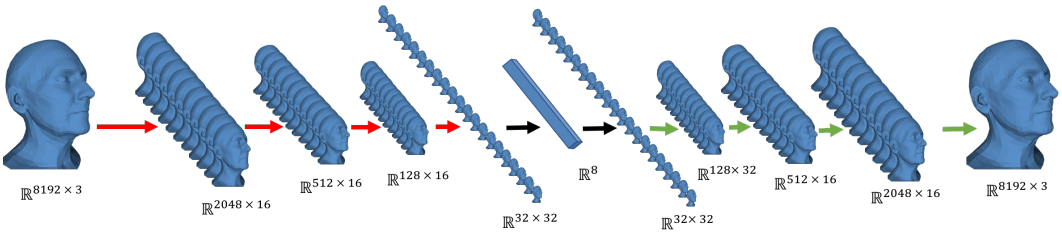


Figure 3: The architecture of the Convolutional Mesh Autoencoder. The red and green arrows indicate downsampling and upsampling layers respectively. The output space of each of the layers in denoted under it. Faces in intermediate layers are only for visualization.

Training. We train our autoencoder for 300 epochs with a learning rate of 8e-3 with a learning rate decay of 0.99 every epoch. We use stochastic gradient descent with a momentum of 0.9 to optimize the L1 loss between predicted mesh vertices and the ground truth samples. We use a regularization on the weights of the network using weight decay of 5e-4. The convolutions use Chebyshev filtering with $K = 6$.

5 Experiments

We evaluate the performance of our model based on its ability to interpolate the training data and extrapolate outside its space. We compare the performance of our model with a PCA model which is

	Mean Error	Median Error	# Parameters
PCA	2.694 ± 2.330	2.002	120,552
Mesh Autoencoder	$\mathbf{1.578} \pm \mathbf{1.546}$	**1.110**	**33,856**

Table 3: Performance comparison with different error metrics on interpolation experiment. Mean error is of the form $[\mu \pm \sigma]$ with mean Euclidean distance μ and standard deviation σ. The median error and number of parameters in each model are also shown. All errors are in millimeters (mm).

used for encoding and reconstruction in state of the art 3D face models (e.g. Tewari et al., 2017). We consistently use an 8-dimensional latent space to encode the face mesh using both the PCA model and the Mesh Autoencoder. Thus, the encoded latent vectors lie in $\mathbb{R}^8$. Meanwhile, the number of parameters in our model is much smaller than PCA model (Table 3).

Facial Expression Dataset. Our dataset consists of 12 classes of extreme expressions from 12 different subjects. These expressions are highly complex and uncorrelated with each other. The expressions in our dataset are – bareteeth, cheeks in, eyebrow, high smile, lips back, lips up, mouth down, mouth extreme, mouth middle, mouth side and mouth up. The number of frames of each sequence is shown in Table 4.

The data is captured at 60fps with a multi-camera active stereo system (3dMD LLC, Atlanta) with six stereo camera pairs, five speckle projectors, and six color cameras. Our dataset contains 20,466 3D Meshes, each with about 120,000 vertices. The data is pre-processed using a sequential mesh registration method (Li et al., 2017) to reduce the data dimensionality to 5023 vertices. We pre-process the data by adding fake vertices to increase the number of vertices to 8192. This enables pooling and upsampling of the mesh across the layers with a constant factor.

Implementation details We use Tensorflow (Abadi et al., 2016) for our network implementation. We use Scikit-learn (Pedregosa et al., 2011) for computing PCA coefficients. Training each network takes about 8 hours on a single Nvidia Tesla P100 GPU. Each of the models is trained for 300 epochs with a batch size of 16.

We design the following two different experiments for evaluating the interpolation and extrapolation capability of our model.

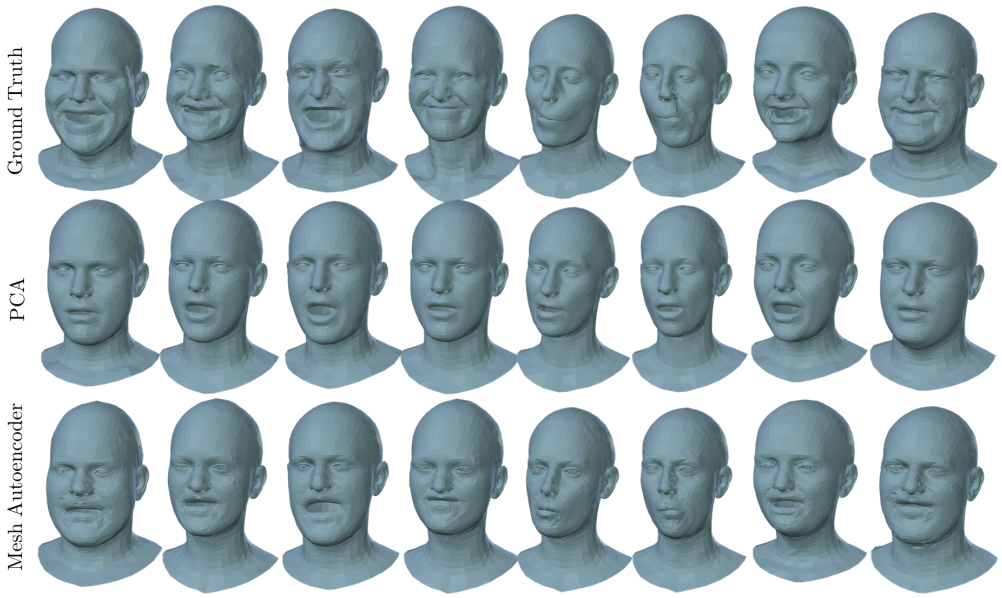
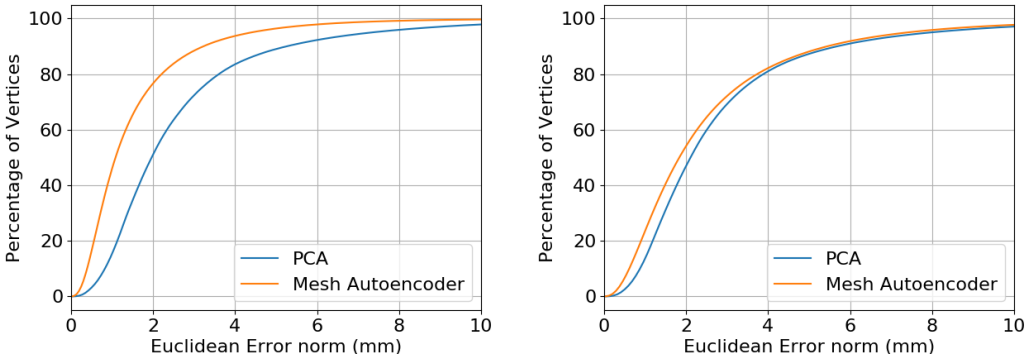


Figure 4: Qualitative results for mesh reconstruction on test set of interpolation experiment.



(a) (b)

Figure 5: Cumulative Euclidean error between PCA model and Mesh Autoencoder for Interpolation(a) and Extrapolation(b) experiments.

5.1 INTERPOLATION EXPERIMENT

In order to evaluate the interpolation capability of the autoencoder, we split the dataset in training and test samples in the ratio of 1:9. The test samples are obtained by picking consecutive frames of length 10 uniformly at random across the sequences. We train our autoencoder for 300 epochs and evaluate it on the test set. We use mean Euclidean distance for comparison with the PCA method. The mean Euclidean distance of N test mesh samples with n vertices each is given by

$$\mu = \frac{1}{nN} \sum_{i=1}^{N} \sum_{j=1}^{n} ||x_{ij} - \hat{x}_{ij}||_2 \tag{3}$$

where $x_{ij}, \hat{x}_{ij} \in \mathbb{R}^3$ are j-th vertex predictions and ground truths respectively corresponding to i-th sample. Table 3 shows the mean Euclidean distance along with standard deviation in the form $[\mu \pm \sigma]$. The median error is also shown in the table. We show a performance improvement, as high as 50% over PCA models for capturing these highly non linear facial expressions. At the same time, the number of parameters in the CNN is about 75% fewer than the PCA model as shown in Table 3. Visual inspection of our qualitative results in Figure 4 show that our reconstructions are more realistic and are effective in capturing extreme facial expressions. We also show the histogram of cumulative errors in Figure 5a. We observe that Mesh Autoencoder has about 76.9% of the vertices within an Euclidean error of 2 mm, as compared to 51.7% for the PCA model.

5.2 EXTRAPOLATION EXPERIMENT

We evaluate the generalization capability of the Mesh Autoencoder by attempting to reconstruct the expressions that are completely unseen by our model. We split our dataset by completely excluding one expression set from all the subjects of the dataset. We test our Mesh Autoencoder on the excluded expression as the test set. Linear models like PCA traditionally show a very good performance on such generalization of meshes. We compare the performance of our model with PCA using the same mean Euclidean distance. We perform 12 cross validation experiments, one for each expression as shown in Table 4. For each experiment, we run our training procedure ten times initializing the weights at random. We pick the best performing network for comparison.

We compare the results using mean Euclidean distance and median error metric in Table 4. Our method performs better than PCA model on 9 out of 12 expression sequence. The sequences where PCA performs better are *eyebrow*, *mouth extreme* and *mouth open*. This could be due to very linear and smooth nature of these sequences that can be easily be approximated by the linear model, or due to shortage of training data. We show the qualitative results in Figure 6. We observe that our Mesh Autoencoder generalizes much better than PCA on these facial expression sequences. Especially, PCA results are very smooth and are unable to capture most of the non-linear deformations. Our model performs much better on these extreme expressions. We show the cumulative euclidean error

Sequence	# Frames	PCA		Mesh Autoencoder	
		Mean Error	Median Error	Mean Error	Median Error
bareteeth	1946	3.508 ± 3.327	2.510	$\mathbf{3.048 \pm 2.909}$	**2.040**
cheeks in	1396	2.947 ± 2.709	2.104	$\mathbf{2.458 \pm 2.333}$	**1.800**
eyebrow	2283	$\mathbf{2.361 \pm 1.826}$	**1.880**	2.736 ± 2.108	2.214
high smile	1878	3.268 ± 2.924	2.311	$\mathbf{2.280 \pm 2.130}$	**1.620**
lips back	1694	2.611 ± 2.575	1.816	$\mathbf{2.442 \pm 2.387}$	**1.741**
lips up	1511	2.652 ± 2.254	1.981	$\mathbf{1.850 \pm 1.765}$	**1.307**
mouth down	2363	2.939 ± 2.258	2.261	$\mathbf{2.634 \pm 2.177}$	**2.023**
mouth extreme	793	$\mathbf{3.523 \pm 3.753}$	**2.449**	4.040 ± 4.251	2.636
mouth middle	1997	2.805 ± 2.057	2.259	$\mathbf{2.372 \pm 2.107}$	**1.722**
mouth open	674	2.960 ± 3.109	2.034	$\mathbf{1.866 \pm 1.773}$	**1.367**
mouth side	1778	$\mathbf{3.288 \pm 3.443}$	**2.305**	3.784 ± 3.823	2.566
mouth up	2153	2.747 ± 2.277	2.036	$\mathbf{2.449 \pm 2.281}$	**1.776**

Table 4: Quantitative evaluation of Extrapolation experiment. The training set consists of the rest of the expressions. Mean error is of the form $[\mu \pm \sigma]$ with mean Euclidean distance μ and standard deviation σ. The median error and number of frames in each expression sequnece is also shown. All errors are in millimeters (mm).

histogram in Figure 5b. For a 2 mm accuracy, Mesh Autoencoder captures 54.6% of the vertices while the PCA model captures 48.1% of it.

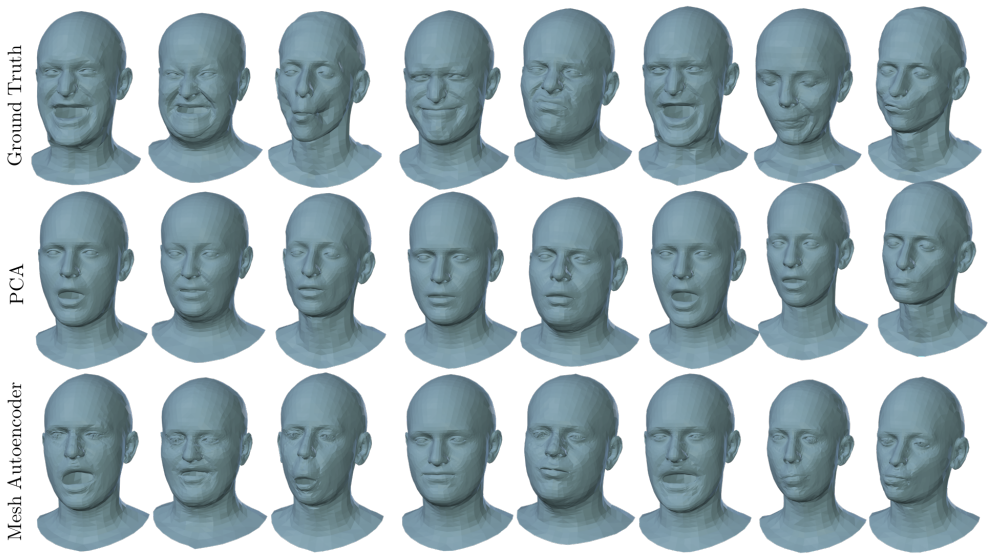


Figure 6: Qualitative results on test set of extrapolation experiments. The expression of the test set was completely excluded from all subjects.

5.3 Discussion

While our convolutional Mesh Autoencoder leads to a representation that generalizes better for unseen 3D faces than PCA with much fewer parameters, our model has several limitations. Our network is restricted to learning face representation for a fixed topology, i.e., all our data samples needs to have the same adjacency matrix, A. The mesh sampling layers are also based on this fixed adjecency matrix A, which defines only the edge connections. The adjacency matrix does not take in to account the vertex positions thus affecting the performance of our sampling operations. In future, we would like to incorporate this information into our learning framework.

The amount of data for high resolution faces is very limited. We believe that generating more of such data with high variability between faces would improve the performance of Mesh Autoencoders for 3D face representations. The data scarcity also limits our ability to learn models that can be trained for superior performance at higher dimensional latent space. The data scarcity also produces noise in some reconstructions.

6 CONCLUSION

We have introduced a generalization of convolutional autoencoders to mesh surfaces with mesh downsampling and upsampling layers combined with fast localized convolutional filters in spectral space. The locally invariant filters that are shared across the surface of the mesh significantly reduce the number of filter parameters in the network. While the autoencoder is applicable to any class of mesh objects, we evaluated its quality on a dataset of realistic extreme facial expressions. The local convolutional filters capture a lot of surface details that are generally missed in linear models like PCA while using 75% fewer parameters. Our Mesh Autoencoder outperforms the linear PCA model by 50% on interpolation experiments and generalizes better on completely unseen facial expressions.

Face models are used in a large number of applications in computer animations, visual avatars and interactions. In recent years, a lot of focus has been given to capturing highly detailed static and dynamic facial expressions. This work introduces a direction in modeling these high dimensional face meshes that can be useful in a range of computer graphics applications.

REFERENCES

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

B. Amberg, R. Knothe, and T. Vetter. Expression invariant 3D face recognition with a morphable model. In *International Conference on Automatic Face Gesture Recognition*, pp. 1–6, 2008.

V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, pp. 187–194, 1999.

D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Eurographics Symposium on Geometry Processing*, pp. 13–23, 2015.

D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 3189–3197, 2016.

S. Bouaziz, Y. Wang, and M. Pauly. Online modeling for realtime facial animation. *ACM Transactions on Graphics*, 32(4):40, 2013.

M. Breidt, H. H. Bülthoff, and C. Curio. Robust semantic analysis by synthesis of 3D facial motion. In *International Conference on Automatic Face and Gesture Recognition and Workshops*, pp. 713–719, 2011.

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *Signal Processing Magazine*, 34(4):18–42, 2017.

J Bruna, W. Zaremba, A. Szlam, and Y LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

A. Brunton, T. Bolkart, and S. Wuhrer. Multilinear wavelets: A statistical shape space for human faces. In *European Conference on Computer Vision*, pp. 297–312, 2014a.

A. Brunton, A. Salazar, T. Bolkart, and S. Wuhrer. Review of statistical shape spaces for 3D data with comparative analysis for human faces. *Computer Vision and Image Understanding*, 128: 1–17, 2014b.

C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: A 3D facial expression database for visual computing. *Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2014.

F. R. K. Chung. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

D. Cosker, E. Krumhuber, and A. Hilton. A FACS valid 3D dynamic action unit database with applications to 3D dynamic morphable facial modeling. In *International Conference on Computer Vision*, pp. 2296–2303, 2011.

M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *Transactions on pattern analysis and machine intelligence*, 29(11), 2007.

C. Ferrari, G. Lisanti, S. Berretti, and A. Del Bimbo. Dictionary learning based 3D morphable model construction for face recognition with varying expression and pose. In *International Conference on 3D Vision*, pp. 509–517, 2015.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing*, pp. 6645–6649, 2013.

D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

M Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.

A. S. Jackson, A. Bulat, V. Argyriou, and G. Tzimiropoulos. Large pose 3D face reconstruction from a single image via direct volumetric CNN regression. *International Conference on Computer Vision*, 2017.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Y LeCun. Generalization and network design strategies. Technical report, University of Toronto, 1989.

H. Li, T. Weise, and M. Pauly. Example-based facial rigging. 29(4):32, 2010.

T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics*, 36(6), 2017.

H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, and Y. Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Transactions on Graphics*, 36(4):71:1–71:10, 2017.

J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. In *International Conference on Computer Vision Workshops*, pp. 37–45, 2015.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

F. Monti, D. Boscaini, J. Masci, E. Rodolà, J Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. 2017.

T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt. Sparse localized deformation components. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 32(6): 179:1–179:10, 2013.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

A. Savran, N. Alyuöz, H. Dibeklioglu, O. Celiktutan, B. Gökberk, B. Sankur, and L. Akarun. Bosphorus database for 3D face analysis. In *Biometrics and Identity Management*, pp. 47–56, 2008.

A. Sinha, J. Bai, and K. Ramani. Deep learning 3D shape surfaces using geometry images. In *European Conference on Computer Vision*, pp. 223–240, 2016.

A. Tewari, M. Zollöfer, H. Kim, P. Garrido, F. Bernard, P. Perez, and Theobalt C. MoFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *International Conference on Computer Vision*, 2017.

J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt. Real-time expression transfer for facial reenactment. *Transactions on Graphics*, 34(6):183:1–183:14, 2015.

A van den Oord, S Dieleman, H Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

N Verma, E. Boyer, and J Verbeek. Dynamic filters in graph convolutional networks. *CoRR*, abs/1706.05206, 2017.

D. Vlasic, M. Brand, H. Pfister, and J. Popović. Face transfer with multilinear models. *Transactions on Graphics*, 24(3):426–433, 2005.

F. Yang, J. Wang, E. Shechtman, L. Bourdev, and D. Metaxas. Expression flow for 3D-aware face component transfer. *Transactions on Graphics*, 30(4):60:1–10, 2011.

L. Yi, H. Su, X. Guo, and L. J. Guibas. SyncSpecCNN: Synchronized spectral CNN for 3d shape segmentation. 2017.

L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato. A 3D facial expression database for facial behavior research. In *International Conference on Automatic Face and Gesture Recognition*, pp. 211–216, 2006.

L. Yin, X. Chen, Y. Sun, T. Worm, and M. Reale. A high-resolution 3D dynamic facial expression database. In *International Conference on Automatic Face and Gesture Recognition*, pp. 1–6, 2008.