ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2020
J. Bender and T. Popa
(Guest Editors)

Volume 39 (2020), Number 8

Cloth and Skin Deformation with a Triangle Mesh Based Convolutional Neural Network, Supplementary

Nuttapong Chentanez[1,2], Miles Macklin[1,3], Matthias Müller[1], Stefan Jeschke[1] and Tae-Yong Kim[1]

[1]NVIDIA
[2]Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University
[3]University of Copenhagen

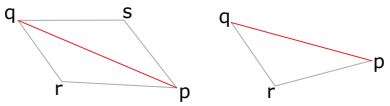

Figure 1: *Neighbors for convolution from vertices onto edge pq. Left) Internal edge, Right) Boundary edge.*

1 Alternative Convolution adapted from [HHF*19]

Prior to the convolution operator presented in the paper, we experimented with a convolution operator inspired by the edge based convolution from [HHF*19]. It does not perform as well as the convolution operators we proposed in this work but we include it here as a baseline. In the result section this is referred to as $V4$. Let's say the mesh consists of V vertices and E edges, where each vertex has c_{in} features and we want to do a convolution where each vertex has c_{out} features. Each convolution performs the following operations:

1. For each edge, compute 4 order invariant features based on its neighboring vertices and produce an intermediate tensor of size $c_{in} \times E \times 4$.
2. Do a 2D convolution, where the kernel size is 1 x 4, no padding, stride 1, with c_{in} input channels and c_{out} output channels, to produce an image of size c_{out} x E x 1, for which we drop the last dimension of the tensor.
3. For each vertex, aggregate the features from the surrounding edges by averaging.

For an edge (p,q), whose opposite vertices are r and s, as shown in Figure 1, we compute 4 order invariant features as follows, $(f_p + f_q, |f_p - f_q|, f_r + f_s, |f_r - f_s|)$, where the operations are done componentwise. For boundary edges, where s is -1 indicating no neighbor, we either use $f_s = 0$, "zero padding" or $f_s = f_r$, "same padding". The convolution is efficiently implemented with General Matrix to Matrix Multiplications (GEMMs).

2 Details about MeshCNN and Spiral++ experiments

We compare MeshCNN [HHF*19] and Spiral++ [GCBZ19] with our network for the Pose to Cloth problems. Our network is an encoder-decoder architecture of the form $D^x D^{2x} D^{4x} D_5 D^{4x}$ for various values of x. The result is shown in Table 1.

We modify the code of MeshCNN [HHF*19] from https://github.com/ranahanocka/MeshCNN and Spiral Net++ [GCBZ19] https://github.com/sw-gong/spiralnet_plus to compare with our work. The details will be discussed in this section.

The encoder-decoder variation of MeshCNN [HHF*19] takes input and produces output on edges, which is not directly compatible with our problem. Therefore, we prepend a first layer that produces edge features from vertex features. Say an edge's endpoints are vertices p and q with features f_p and f_q, the edge features are computed as $(f_p + f_q, |f_p - f_q|)$. We also append a last layer that produces vertex features by averaging from the surrounding edges. We also use L_1 as the loss function. For the pants mesh with 12064 edges, we use the pool sizes of 1800, 1350, 780. From a preliminary test, the last pool size can't be smaller than 780, as otherwise, the code sometimes reports that it can't collapse edges and crashes. We set the number of channels of convolution layers to $[c, 2c, 4c, 8c]$ where c is the smallest value such that

$$U_{MeshCNN}(d) \geq M, \tag{1}$$

where $U_{MeshCNN}(d)$ is the number of learnable weights of the MeshCNN network when the number of channels of the convolution layers are set as above and M is the number of learnable weights of our network that the MeshCNN network is compared against. For each case, we run experiments with the number of residual blocks 1, 2, and 3, each with the number of channels computed as above, and report the lowest training and testing errors. We train for 200 epochs and the other parameters are left as default. We cannot however, run the code on the tank-female mesh with 43308 edges as the program ran out of memory on our 16GB GPU, no matter what parameters we tried for the pool sizes, even when the batch size is 1.

For Spiral Net++ [GCBZ19], we modify the code to use our dataset and output the displacements that minimize the L_1 error. We set the number of channels of the convolution layers to be $[c, c, c, 2c]$ where c is smallest value such that

$$U_{SpiralNet++}(d) \geq M, \tag{2}$$

where $U_{SpiralNet++}(d)$ is the number of learnable weights of the SpiralNet++ network when the number of channels of the convolution layers are set as above and M is the number of learnable

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

weights of our network that the SpiralNet++ network is compared against.

As in their default choice, the number of channels only doubled at the innermost layer. For each case, we run experiments with the number of latent channels of 16, 64, 256, 1024 and report the lowest train and test errors. We also need to reduce the learning rate to 10^{-4} as the default choice of 10^{-3} diverges in some cases. We also need to reduce the batch size to 8 for the cases when $x = 80$, as the default choice of 32 runs out of memory. We then run the training for 200 epochs and other parameters are left at their default values.

References

[BBP*19] BOURITSAS G., BOKHNYAK S., PLOUMPIS S., BRONSTEIN M., ZAFEIRIOU S.: Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *The IEEE International Conference on Computer Vision (ICCV)* (2019). 3

[GCBZ19] GONG S., CHEN L., BRONSTEIN M. M., ZAFEIRIOU S.: Spiralnet++: A fast and highly efficient mesh convolution operator. In *The IEEE International Conference on Computer Vision (ICCV)* (2019). 1, 3

[HHF*19] HANOCKA R., HERTZ A., FISH N., GIRYES R., FLEISHMAN S., COHEN-OR D.: Meshcnn: A network with an edge. *ACM Trans. Graph. 38*, 4 (July 2019), 90:1–90:12. 1, 3

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

Pants							
x	RC_9	EC_9	MeshCNN	Spiral++	V_4	SP_9	SPD_9
10	3.043 / 2.285	3.028 / 2.426	6.5388 / 6.5224	3.4155 / 3.5047	3.312 / 2.562	3.102 / 2.379	3.034 / 2.323
20	2.144 / 1.634	2.149 / 1.755	5.4732 / 5.4819	2.7328 / 2.8645	2.379 / 1.901	2.207 / 1.783	2.15 / 1.673
40	1.511 / 1.183	1.553 / 1.297	4.5729 / 4.6337	2.4536 / 2.6278	1.743 / 1.406	1.554 / 1.407	1.555 / 1.281
80	1.259 / 1.028	1.239 / 1	3.6733 / 3.8082	1.764 / 1.9833	1.322 / 1.122	1.214 / 1.066	1.221 / 0.967
x	RC_{11}	EC_{11}	MeshCNN	Spiral++	V4	SP_{11}	SPD_{11}
10	2.896 / 2.239	2.839 / 2.253	6.4336 / 6.4131	3.2546 / 3.3714	3.077 / 2.287	2.949 / 2.3	2.904 / 2.289
20	1.997 / 1.671	2.009 / 1.639	5.3345 / 5.3604	2.6666 / 2.8185	2.32 / 1.95	2.105 / 1.687	2.056 / 1.601
40	1.418 / 1.149	1.498 / 1.393	4.4487 / 4.4988	3.0562 / 3.2058	1.725 / 1.454	1.454 / 1.23	1.522 / 1.251
80	1.071 / 0.956	1.07 / 0.921	3.5983 / 3.7106	1.6386 / 1.86	1.296 / 1.139	1.104 / 0.971	1.094 / 0.933
x	RC_{13}	EC_{13}	MeshCNN	Spiral++	V4	SP_{13}	SPD_{13}
10	2.762 / 2.176	2.746 / 2.242	6.2417 / 6.2284	3.1931 / 3.3133	3.019 / 2.397	2.834 / 2.252	2.848 / 2.156
20	1.895 / 1.52	1.944 / 1.621	5.1697 / 5.1706	2.6287 / 2.7731	2.166 / 1.779	2.007 / 1.683	1.945 / 1.612
40	1.357 / 1.069	1.344 / 1.169	4.2156 / 4.2555	2.4766 / 2.5865	1.614 / 1.414	1.437 / 1.192	1.431 / 1.194
80	1.047 / 1.002	1.012 / 0.929	3.5141 / 3.6128	1.4909 / 1.712	1.33 / 1.192	1.066 / 0.895	1.102 / 0.972

Tank-Female						
x	RC_9	EC_9	Spiral++	V_4	SP_9	SPD_9
10	2.364 / 2.351	2.368 / 2.564	2.6794 / 2.7246	2.439 / 2.531	2.401 / 2.507	2.337 / 2.373
20	1.819 / 1.872	1.8 / 1.946	2.3799 / 2.4717	2.006 / 2.159	1.882 / 2.077	1.817 / 2.063
40	1.449 / 1.553	1.297 / 1.405	2.1795 / 2.327	1.455 / 1.704	1.322 / 1.449	1.259 / 1.356
80	0.894 / 0.997	0.923 / 1.003	1.8951 / 2.0932	1.15 / 1.317	0.94 / 1.125	0.902 / 1.067
x	RC_{11}	EC_{11}	Spiral++	V4	SP_{11}	SPD_{11}
10	2.261 / 2.433	2.292 / 2.412	2.6694 / 2.7239	2.389 / 2.511	2.331 / 2.656	2.249 / 2.427
20	1.709 / 1.833	1.701 / 1.838	2.3528 / 2.4508	1.9 / 2.107	1.755 / 1.961	1.768 / 2.018
40	1.304 / 1.408	1.21 / 1.307	2.1113 / 2.2708	1.416 / 1.587	1.232 / 1.431	1.179 / 1.3
80	0.928 / 1.053	0.917 / 0.982	1.8529 / 2.0591	1.361 / 1.646	0.975 / 1.131	0.875 / 1.014
x	RC_{13}	EC_{13}	Spiral++	V4	SP_{13}	SPD_{13}
10	2.188 / 2.313	\ / 2.256	2.6294 / 2.6926	2.292 / 2.344	2.275 / 2.235	2.201 / 2.467
20	1.636 / 1.808	1.649 / 1.78	2.3252 / 2.4425	1.823 / 2.103	1.697 / 1.86	1.604 / 1.675
40	1.13 / 1.265	1.079 / 1.265	1.862 / 2.0254	1.304 / 1.521	1.215 / 1.353	1.138 / 1.244
80	0.842 / 0.922	0.879 / 0.967	1.8199 / 2.0312	1.221 / 1.5	0.9 / 1.049	0.891 / 0.995

Table 1: *Errors of various convolutions for the upsampling for the pants and tank-female mesh for various networks. Our proposed ring based (RC) and ellipse based (EC) convolutions (RC) for various filter length, specified by the subscript are shown. We use the encoder-decoder architecture $D^xD^{2x}D^{4x}D_5D^{4x}$ for various x. Error is specified as training / testing average L1 error per vertex per degree of freedom $\times 10^{-3}$. We compare against previous work MeshCNN(MeshcNN) [HHF*19] and SpiralNet++(Spiral++) [GCBZ19] where we choose the network parameters so that the number of learnable weights are as close as possible but greater than ours. We also include the errors when using our network but with the convolution replaced by alternatives. V4 refers to the vertex convolution adapted from [HHF*19] as stated in the appendix, SP and SPD are the spiral convolution [BBP*19, GCBZ19]. The cells highlighted in green/yellow show the configurations with lowest training/testing errors for each x and filter size.*

Architecture	#Params	RC_{13}	EC_{13}	SP_{13}	SPD_{13}	RC^{max}_{13}	EC^{max}_{13}	SP^{max}_{13}	SPD^{max}_{13}
$D^{36}D^{72}D^{144}D^{144}{}_6$	7790055	0.802 / 0.898	0.805 / 0.918	0.8 / 0.989	0.927 / 1.008	0.78 / 0.877	0.798 / 0.986	0.781 / 0.926	0.903 / 1.035
$D^{47}D^{94}D^{188}DD^{188}DD^{188}DD^{188}$	7760058	0.752 / 0.886	0.683 / 0.777	0.792 / 0.93	0.778 / 0.922	0.73 / 0.857	0.712 / 0.791	0.709 / 0.832	0.778 / 0.939
$D^{54}D^{108}D^{216}{}_2D_3D^{216}D_2$	7815813	0.783 / 0.93	0.721 / 0.8	0.739 / 0.865	0.758 / 0.868	0.746 / 0.805	0.715 / 0.879	0.707 / 0.842	0.745 / 0.851
$D^{62}D^{124}D^{248}DD^{248}D_3D^{248}$	7902829	0.711 / 0.83	0.702 / 0.802	0.794 / 0.896	0.751 / 0.849	0.66 / 0.745	0.664 / 0.779	0.719 / 0.832	0.769 / 0.894
$D^{65}D^{130}D^{260}D_3D^{260}D_2$	7806688	0.831 / 0.914	0.78 / 0.917	0.755 / 0.904	0.777 / 0.876	0.688 / 0.805	0.715 / 0.815	0.76 / 0.888	0.807 / 0.954
$D^{80}D^{160}D^{320}D_5D^{320}$	7829163	0.857 / 0.966	0.844 / 0.945	0.84 / 0.975	0.867 / 0.966	0.862 / 0.975	0.808 / 0.895	0.88 / 1.001	0.83 / 0.956
$D^{41}{}_2D^{82}{}_2D^{164}{}_5$	7742481	0.798 / 0.94	0.788 / 0.889	0.901 / 1.032	0.859 / 0.999	0.835 / 0.958	0.77 / 0.898	0.831 / 0.991	0.869 / 1.017
$D^{52}DD^{104}DD^{208}DD^{208}DD^{208}$	7775423	0.739 / 0.845	0.847 / 0.969	0.845 / 0.979	0.948 / 1.026	0.774 / 0.875	0.816 / 0.911	0.829 / 0.881	0.96 / 1.102
$D^{62}DD^{124}{}_2D^{248}DD^{248}D_2$	7902953	0.792 / 0.893	0.794 / 0.878	0.841 / 0.947	0.856 / 0.897	0.683 / 0.749	0.724 / 0.791	0.791 / 0.925	0.899 / 1.001
$D^{64}DD^{128}DD^{256}D_3D^{256}$	7728315	0.813 / 0.948	0.736 / 0.787	0.826 / 0.945	0.808 / 0.89	0.801 / 0.902	0.77 / 0.867	0.782 / 0.881	0.813 / 0.93
$D^{80}DD^{160}DD^{320}{}_2D_3$	7829163	0.896 / 1.085	0.895 / 0.976	1.032 / 1.163	0.982 / 1.081	0.994 / 1.116	0.919 / 1.04	0.934 / 1.091	1.001 / 1.143
$D^{75}DD^{150}DD^{300}DD^{300}D_2$	7759398	0.827 / 0.946	0.822 / 0.925	0.98 / 1.125	0.958 / 1.063	0.82 / 0.875	0.83 / 0.965	0.887 / 1.062	1.068 / 1.18
$D^{79}DD^{158}DD^{316}D_3D^{316}$	7878319	0.78 / 0.893	0.85 / 1.014	0.819 / 0.918	0.862 / 0.895	0.765 / 0.848	0.753 / 0.871	0.914 / 1.087	0.787 / 0.906
$D^{49}{}_3D^{98}{}_3D^{196}{}_3$	7685430	0.835 / 0.95	0.826 / 0.916	0.949 / 1.038	0.936 / 1.084	0.821 / 0.941	0.836 / 0.909	0.884 / 0.996	0.96 / 1.032
$D^{61}DD^{61}D^{122}{}_2DD^{244}DD^{244}$	7843869	0.778 / 0.852	0.798 / 0.861	0.873 / 0.967	0.902 / 1.04	0.747 / 0.855	0.772 / 0.839	0.938 / 0.998	0.881 / 0.974
$D^{88}D_2D^{176}D_2D^{352}D_2$	7861251	1.082 / 1.178	0.964 / 1.112	1.04 / 1.202	1.028 / 1.155	1.041 / 1.138	0.917 / 1.074	0.964 / 1.123	1.063 / 1.161
$D^{73}D_2D^{146}{}_2DD^{292}DD^{292}$	7836527	0.826 / 0.903	0.856 / 0.976	0.951 / 1.078	0.986 / 1.058	0.913 / 0.957	0.792 / 0.871	0.834 / 0.974	1.005 / 1.125
$D^{80}D_2D^{160}DD^{160}D^{320}D_2$	7829323	0.874 / 0.989	0.856 / 0.955	1.001 / 1.138	0.935 / 1.064	0.796 / 0.875	0.87 / 0.973	0.905 / 1.054	0.933 / 1.051
$D^{79}D_3D^{158}D_2D^{316}DD^{316}$	7878319	0.845 / 0.889	0.886 / 0.936	0.886 / 0.976	0.958 / 1.105	0.783 / 0.89	0.785 / 0.886	0.888 / 0.934	0.988 / 1.112

Table 2: *Errors for various architectures of the encoder-decoder network with skip connection using RC_{13}, EC_{13}, SP_{13} and SPD_{13} convolutions with either average or max pooling for the tank-female mesh. Error is specified as training / testing L1 error $\times 10^{-3}$. The cells highlighted in green/yellow show the convolution with lowest training/testing errors for each architecture.*

# Mesh	blouse - asvmmetric	blouse-symmetric	blouse-sym-loose	skirt	dress-asvmmetric	shorts	tank-male	pants	dress-victor	dress2
1	1.613 / 1.585									
2	1.677 / 1.572	1.586 / 1.372								
3	1.887 / 1.823	1.744 / 1.488	1.898 / 1.923							
4	1.925 / 1.824	1.793 / 1.517	1.995 / 1.905	1.479 / 1.536						
5	1.993 / 2.14	1.879 / 1.845	2.071 / 2.182	1.542 / 1.676	1.914 / 1.941					
6	2.075 / 2.037	1.95 / 1.79	2.17 / 2.094	1.636 / 1.705	2.004 / 1.956	1.176 / 1.108				
7	2.048 / 2.086	1.939 / 1.819	2.175 / 2.1	1.61 / 1.685	1.994 / 1.889	1.167 / 1.064	1.389 / 1.284			
8	2.09 / 2.151	1.999 / 2.032	2.204 / 2.13	1.641 / 1.719	2.028 / 1.991	1.152 / 1.12	1.415 / 1.353	1.56 / 1.443		
9	2.272 / 2.351	2.158 / 2.163	2.407 / 2.296	1.752 / 1.814	2.219 / 2.147	1.273 / 1.257	1.529 / 1.511	1.744 / 1.652	2.946 / 2.889	
10	2.293 / 2.524	2.189 / 2.334	2.455 / 2.542	1.756 / 1.899	2.242 / 2.203	1.28 / 1.251	1.577 / 1.575	1.734 / 1.666	2.979 / 2.902	1.673 / 1.589

Table 3: *Errors when training a single network $D^{62}D^{124}D^{248}DD^{248}D_3D^{248}$ with RC^{max}_{13} with data from 1 to 10 meshes. As expected, the error tends to increase as the number of meshes used for training grows, but they still remain relatively low. The result shows that our network architecture allows for the possibility of upsampling multiple meshes with a single network.*

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

Architecture	#Params	RC_{13}	EC_{13}	SP_{13}	SPD_{13}	RC^{max}_{13}	EC^{max}_{13}	SP^{max}_{13}	SPD^{max}_{13}
$U^{168}{}_7U^{84}U^{42}U^3$	7170489	1.974 / 1.914	2.065 / 2.209	2.032 / 1.823	2.048 / 2.336	2.075 / 2.154	2.087 / 1.969	2.022 / 2.107	2.235 / 2.265
$U^{208}UU^{208}UU^{208}UU^{208}U^{104}U^{52}U^3$	7117519	1.763 / 1.781	1.886 / 1.86	1.858 / 2.001	2.016 / 2	1.919 / 2.008	1.881 / 1.91	1.925 / 2.164	1.907 / 2.789
$U^{232}U_2U^{232}U_2U^{232}U^{116}U^{58}U^3$	7185801	1.845 / 2.091	1.862 / 1.894	1.912 / 2.292	1.831 / 2.079	1.865 / 2.492	1.876 / 2.053	1.801 / 2.001	2.014 / 2.043
$U^{264}U_5U^{264}U^{132}U^{66}U^3$	7126209	1.893 / 1.853	1.846 / 1.864	1.849 / 2.143	1.964 / 2.456	1.9 / 2.573	1.869 / 2.116	1.909 / 2.035	1.888 / 2.253
$U^{188}{}_5U^{94}{}_2U^{47}{}_2U^3$	7203061	2.122 / 1.951	2.142 / 1.954	2.229 / 2.242	2.371 / 2.405	2.164 / 2.067	2.2 / 1.818	2.159 / 1.995	2.258 / 2.02
$U^{224}UU^{224}UU^{224}U^{112}U^{112}U^{56}{}_2U^3$	7184419	2.064 / 2.073	2.052 / 2.334	2.179 / 2.326	2.215 / 2.101	2.08 / 2.488	2.038 / 1.91	2.113 / 1.899	2.142 / 2.158
$U^{256}U_2U^{256}UU^{128}{}_2U^{64}UU^3$	7205051	1.848 / 2.013	1.924 / 2.079	1.852 / 2.104	2.038 / 2.152	1.853 / 1.934	1.888 / 2.119	1.871 / 2.045	2 / 2.166
$U^{260}U_3U^{260}U^{130}UU^{65}{}_2U^3$	7061463	1.889 / 2.06	1.89 / 1.776	1.929 / 2.152	2.006 / 2.227	1.892 / 2.06	1.872 / 1.821	1.939 / 2.319	2.007 / 2.017
$U^{304}U_4U^{152}{}_2U^{76}UU^3$	7184023	1.811 / 2	1.872 / 2.224	1.784 / 2.006	2.026 / 2.232	1.783 / 1.769	1.872 / 2.084	1.823 / 2.392	1.8 / 2.307
$U^{316}U_4U^{158}UU^{79}{}_2U^3$	7150097	1.888 / 2.421	1.886 / 2.119	1.923 / 2.612	1.87 / 1.934	1.876 / 2.057	1.876 / 1.796	1.852 / 2.167	1.961 / 1.828
$U^{216}{}_3U^{108}{}_3U^{54}{}_3U^3$	7137357	2.256 / 2.466	2.218 / 2.287	2.723 / 2.624	2.349 / 2.275	2.219 / 2.062	2.256 / 2.071	2.262 / 2.489	2.334 / 2.403
$U^{252}UU^{252}U^{126}{}_2UU^{63}UU^{63}U^3$	7124541	2.062 / 2.062	2.072 / 2.006	2.045 / 2.027	2.142 / 2.182	2.024 / 2.182	2.058 / 2.222	2.04 / 1.943	2.303 / 1.933
$U^{300}U_2U^{150}{}_2UU^{75}UU^{75}U^3$	7182273	1.928 / 2.562	1.899 / 1.961	1.933 / 1.94	1.995 / 1.807	1.912 / 2.101	1.936 / 1.986	1.952 / 2.052	2.051 / 2.174
$U^{304}U_2U^{152}UU^{152}U^{76}U_2U^3$	7184023	1.827 / 2.188	1.795 / 1.722	1.835 / 2.553	2.034 / 2.553	1.844 / 2.092	1.898 / 2.541	1.777 / 1.882	1.903 / 2.356
$U^{316}U_2U^{158}U_2U^{79}UU^{79}U^3$	7150097	1.808 / 1.602	1.933 / 1.755	1.921 / 2.12	1.922 / 1.9	1.911 / 1.813	1.821 / 1.984	1.906 / 2.16	1.862 / 2.316

Table 4: *Errors for various architectures of decoder network using RC_{13}, EC_{13}, SP_{13} and SPD_{13} convolutions with either average or max pooling for the dress2 mesh for the pose to cloth problem. Error is specified as training / testing $L1$ error times 10^{-3}. The cells highlight in green/yellow show the convolution with lowest training/testing errors for each architecture.*

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.