Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, Michael M. Bronstein

The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017

held by Tobias Valinski
Supervisor: Benjamin Busam

28.01.2018

Technische Universität München

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

- Introduction
- Existing Approaches
  - Graphs
    - Spectral CNN
    - ChebNet
    - DCNN
  - Manifolds
    - GCNN
    - ACNN
- MoNet
  - Framework
  - How it works
  - Results
    - Images
    - Graphs
    - Manifolds
- Conclusion

Introduction

What can CNNs handle?

→ Cat

Image: [Cat]

→ Cat

Image: [Sound]

Introduction

What can’t CNNs handle?

Graphs

Image: [SocialGraph]

Manifolds

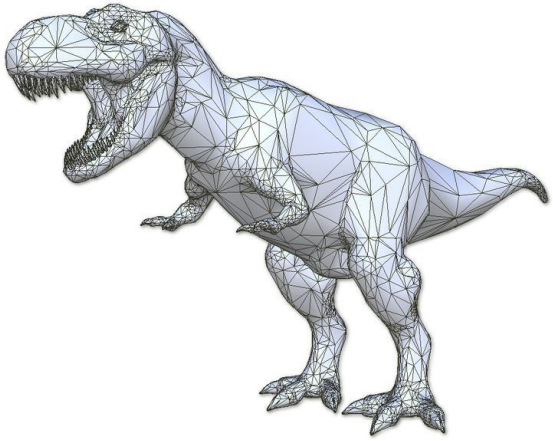


Image: [TRex]

Introduction

Why don’t classical CNNs work?

Kernels can only be applied in **Euclidean** space

1	2	1
2	4	2
1	2	1

*

= ???

Image: [SocialGraph]

Existing Approaches (Graphs) - Spectral CNN

(Bruna et al., ICLR 2013)

$$\mathrm{f}^{out} = \xi\Big(\sum_{l'=1}^{p} \Phi_k \hat{G} \Phi_k^{\top} \mathrm{f}_{l'}^{in}\Big)$$

Formula: [MoNet_Paper]

learnable "filter"

Spectral transformation

Vertex signals

Problem: Computation of the Fourier transforms is Expensive ($O(n^2)$)

Existing Approaches (Graphs) - ChebNet

(Defferrard et al., NIPS 2016)

New definition of the filter:

Formula: [MoNet_Paper]

$$g_\alpha(\Delta) = \sum_{j=0}^{r-1} \alpha_j T_j(\hat{\Delta}) = \sum_{j=0}^{r-1} \alpha_j \Phi T_j(\hat{\Lambda}) \Phi^\top$$

polynomial coefficients

Chebyshev polynomial

Laplacian

$$T_j(\lambda) = 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda)$$

Formula: [MoNet_Paper]

→ Applying g means applying the Laplacian r times
→ O(rn)

Existing Approaches (Graphs)

Big problem:

This spectral approach can’t generalize between different graphs!

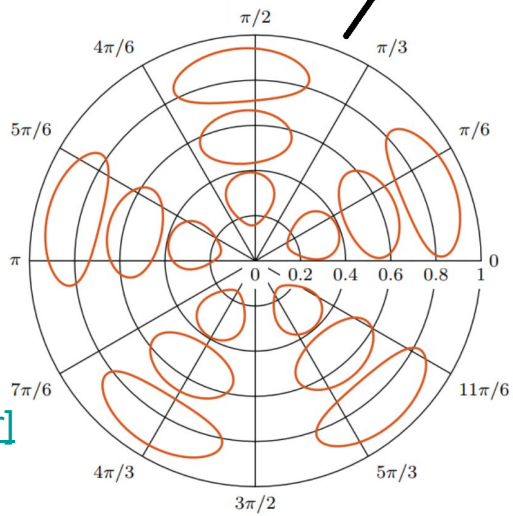
Existing Approaches (Manifolds) - GCNN

(Masci et al., 3DRR 2015)

Extracting Patches:

$$(D(x)f)(p,\theta) = \int_{\chi} w_{p,\theta}(x,y) f(y) dy$$

Formula: [MoNet_Paper]



GCNN

Image: [MoNet_Paper]

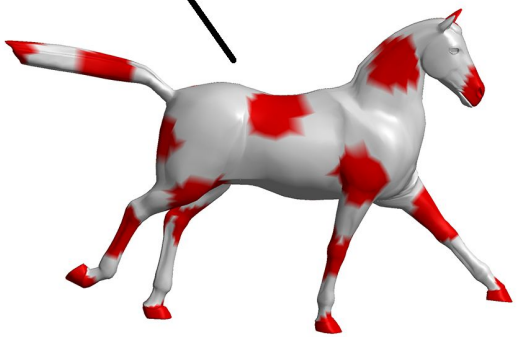


Image: [M.Bronstein]

Existing Approaches (Manifolds) - GCNN

Convolution with Patches:

$$(f \star g)(x) = \max_{\Delta\theta \in [0,2\pi)} \int_0^{2\pi} \int_0^{p_{max}} g(p, \theta + \Delta\theta)(D(x)f)(p, \theta) dp d\theta$$

Formula: [MoNet_Paper]

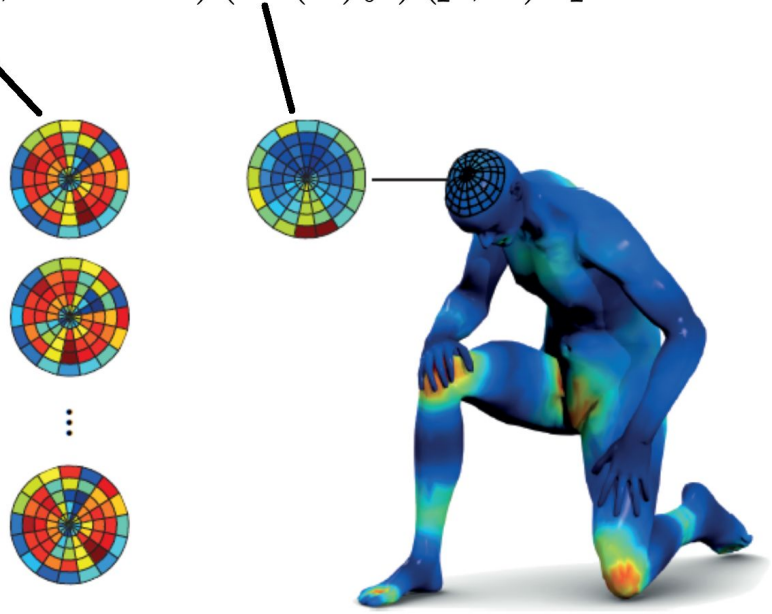


Image: [M.Bronstein]

Existing Approaches (Manifolds) - ACNN

(Boscaini et al., NIPS 2016)

Basic idea: Anisotropic patches

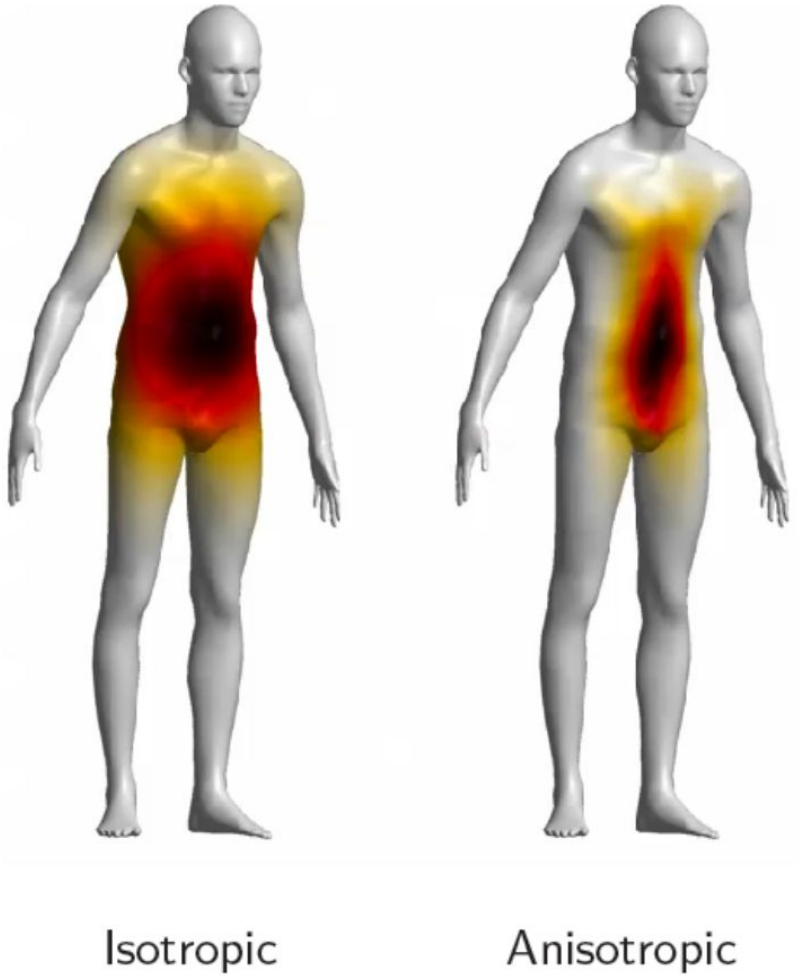


Image: [M.Bronstein]

Existing Approaches (Manifolds) - ACNN

New weighting function:

$$f_t(x,t) = -\mathrm{div}_\chi(A(x)\nabla_\chi f(x,t))$$

$$A_{\alpha\theta}(x) = R_\theta(x)\begin{pmatrix}\alpha & \\ & 1\end{pmatrix}R_\theta^\top(x)$$

rotation

elongation

Formula: [MoNet_Paper]

Formula: [MoNet_Paper]

State of the art so far:

$$(f \star g)(x) = \sum_{j=1}^{J} g_j D_j(x) f$$

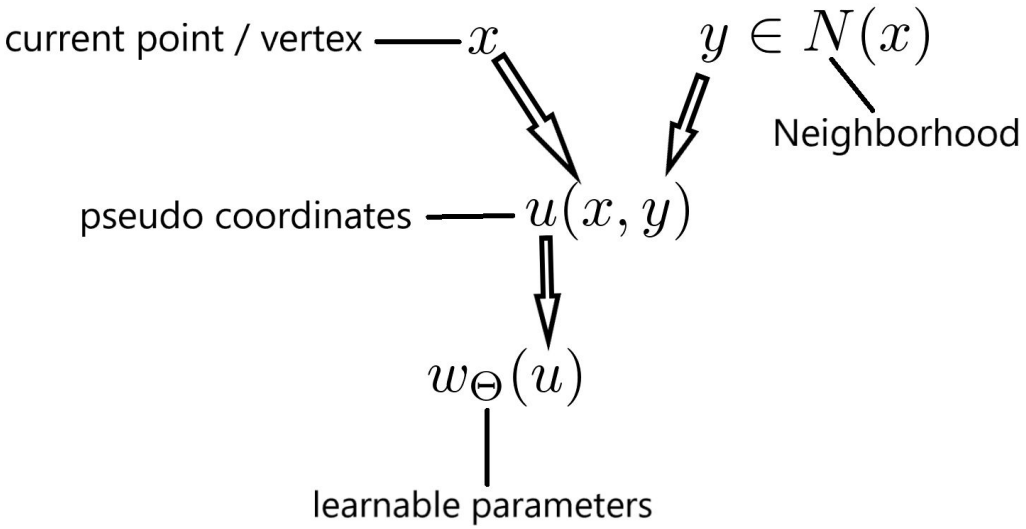
learned

not learned

→ Can’t patch operators be learned?

Monti's Framework

Approach to get machine learning into weight function:



Monti's Framework

Use the new weighting function in a patch operator:

$$D_j(x)f = \sum_{y \in N(x)} w_j(u(x,y))f(y)$$

j'th patch

j'th parameter

Formula: [MoNet_Paper]

Apply the convolution:

# of parameters

$$(f \star g)(x) = \sum_{j=1}^{J} g_j D_j(x) f$$

kernel

Patch

Formula: [MoNet_Paper]

Monti's Framework

Method	Pseudo-coordinates	$\mathbf{u}(x,y)$	Weight function $w_j(\mathbf{u}), j = 1, \dots, J$
CNN [23]	Local Euclidean	$\mathbf{x}(x,y) = \mathbf{x}(y) - \mathbf{x}(x)$	$\delta(\mathbf{u} - \bar{\mathbf{u}}_j)$
GCNN [26]	Local polar geodesic	$\rho(x,y), \theta(x,y)$	$\exp(-\frac{1}{2}(\mathbf{u} - \bar{\mathbf{u}}_j)^\top \begin{pmatrix} \bar{\sigma}_\rho^2 & \\ & \bar{\sigma}_\theta^2 \end{pmatrix}^{-1} (\mathbf{u} - \bar{\mathbf{u}}_j))$
ACNN [7]	Local polar geodesic	$\rho(x,y), \theta(x,y)$	$\exp(-\frac{1}{2}\mathbf{u}^\top \mathbf{R}_{\bar{\theta}_j} \begin{pmatrix} \bar{\alpha} & \\ & 1 \end{pmatrix} \mathbf{R}_{\bar{\theta}_j}^\top \mathbf{u})$
GCN [21]	Vertex degree	$\deg(x), \deg(y)$	$\left(1 - \lvert 1 - \frac{1}{\sqrt{u_1}} \rvert\right)\left(1 - \lvert 1 - \frac{1}{\sqrt{u_2}} \rvert\right)$
DCNN [3]	Transition probability in r hops	$p^0(x,y), \dots, p^{r-1}(x,y)$	$\mathrm{id}(u_j)$

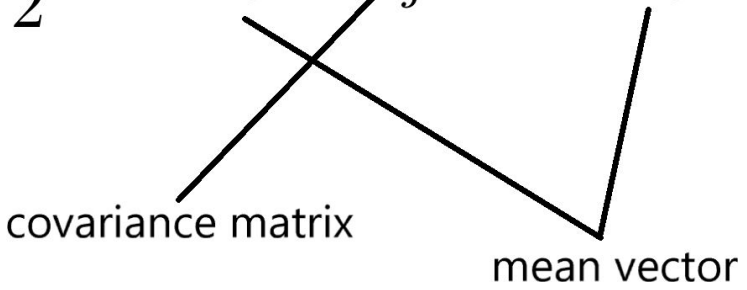
Table: [MoNet_Paper]

Existing approaches for geometric deep learning are special cases of Monti's framework!

MoNet - How it works

Formula: [MoNet_Paper]

$$w_j(u) = exp(-\frac{1}{2}(u - \mu_j)^\top \Sigma_j^{-1}(u - \mu_j))$$



→ Because of this weighting function (f ★ g)(x) can be interpreted as a **Gaussian Mixture Model**!

MoNet - Results (MNIST)

Comparing MoNet to state-of-the-art approaches for euclidean (LeNet) and graph (ChebNet) space

Dataset	LeNet5	ChebNet	**MoNet**
Full grid	99.33%	99.14%	99.19%
$\frac{1}{4}$ grid	98.59%	97.70%	98.16%
300 Superpixels	-	88.05%	**97.30%**
150 Superpixels	-	80.94%	**96.75%**
75 Superpixels	-	75.62%	**91.11%**

Table: [MoNet_Paper]

→ ChebNet fails at generalizing between different graphs

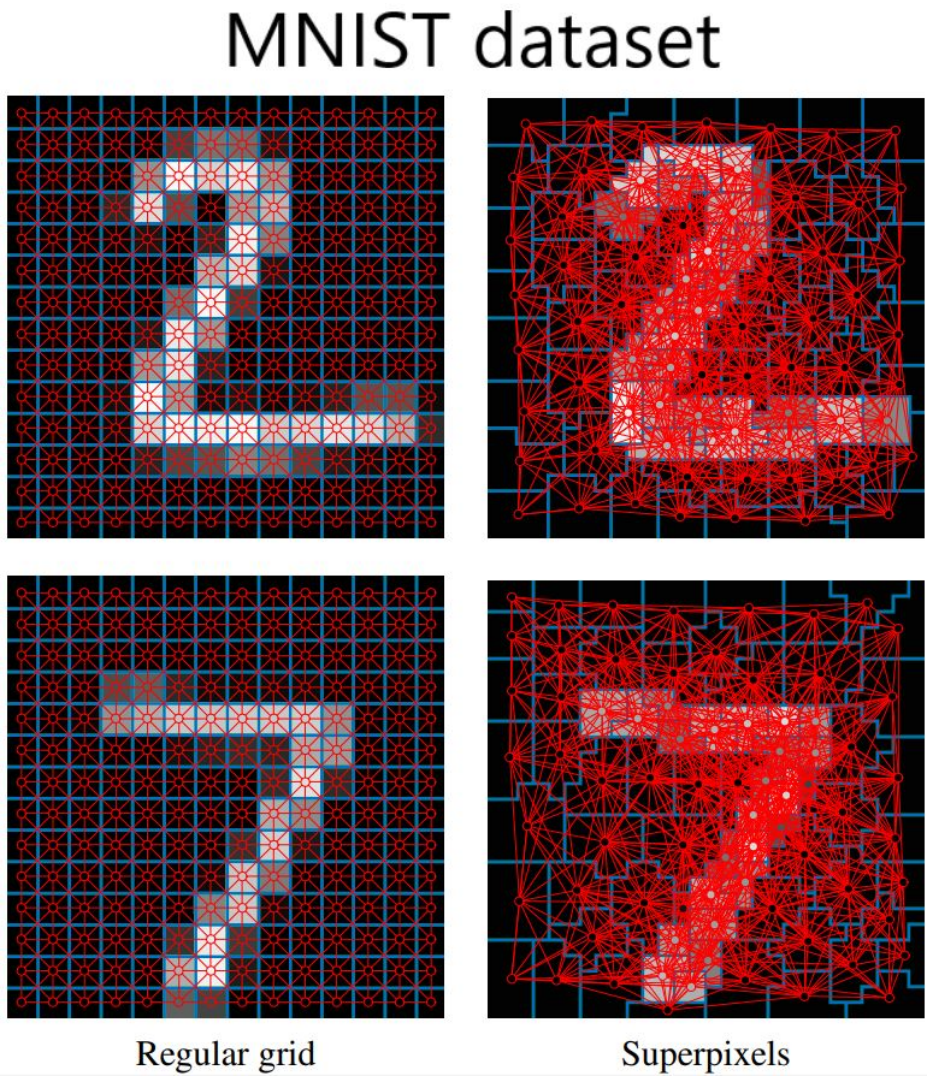


Image: [MoNet_Paper]

MoNet - Results (Cora & PubMed)

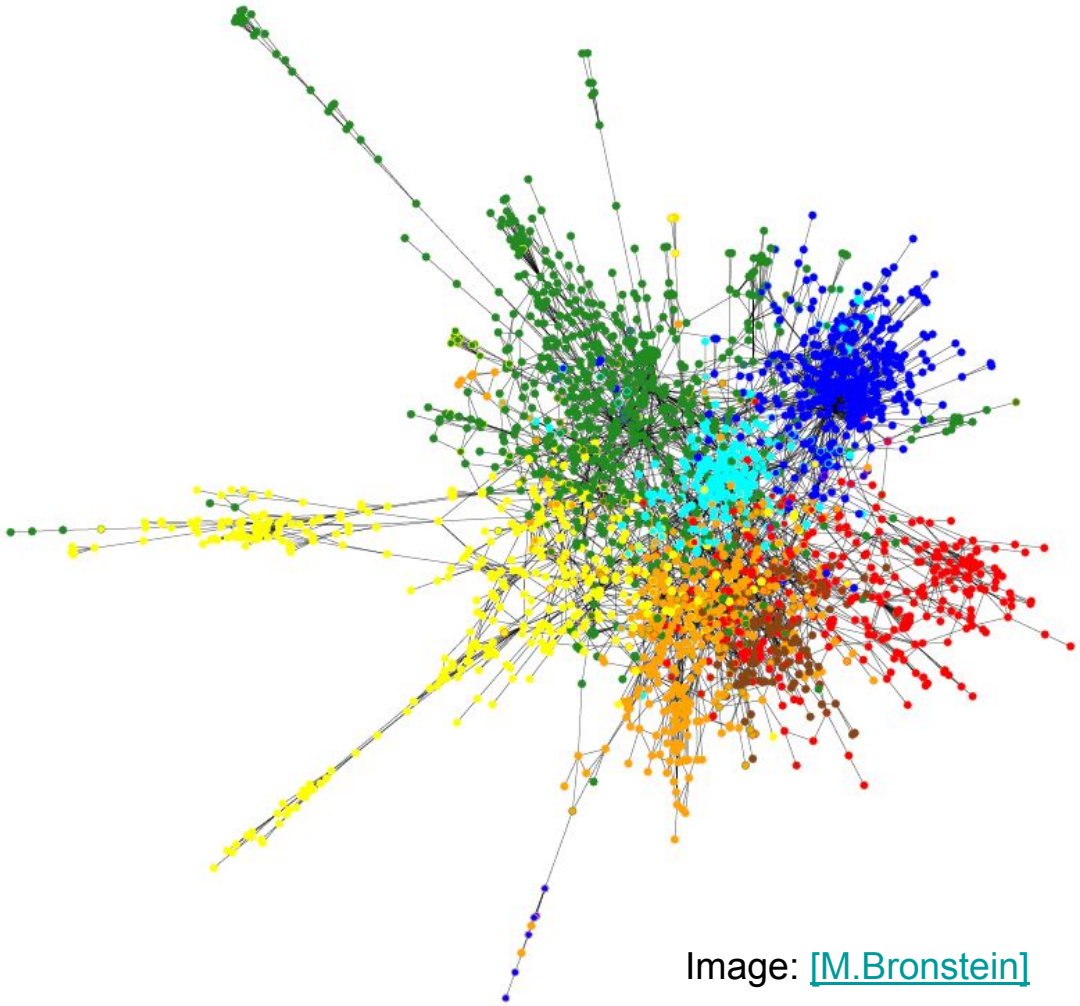


Image: [M.Bronstein]

MoNet - Results (Cora & PubMed)

MoNet was compared to GCN (simplified ChebNet) and DCNN

Task: classify papers into one of 3 (Cora) or 7 (PubMed) groundtruth classes

All approaches were trained in the same way with **20 samples** per class, **500 vertices** for validation and **1000 vertices** for testing.

	Cora	PubMed
Learning Algorithm	Adam	Adam
Number of epochs	3000	1000
Validation frequency	0.01	0.04
Learning rate	0.1	0.1
Decay rate	10^{-1}	-
Decay epochs	1500, 2500	-
Early stopping	No	No

Table: [MoNet_Paper]

MoNet - Results (Cora & PubMed)

$$u(x, y) = \left(\frac{1}{\sqrt{deg(x)}}, \frac{1}{\sqrt{deg(y)}}\right)^\top$$

Formula: [MoNet_Paper]

Patch operator was defined with their standard GMM approach

MoNet - Results (Cora & PubMed)

Table: [MoNet_Paper]

Method	Cora	PubMed
ManiReg [4]	59.5%	70.7%
SemiEmb [42]	59.0%	71.1%
LP [45]	68.0%	63.0%
DeepWalk [28]	67.2%	65.3%
Planetoid [44]	75.7%	77.2%
DCNN [3]	$76.80 \pm 0.60\%$	$73.00 \pm 0.52\%$
GCN [21]	$81.59 \pm 0.42\%$	$78.72 \pm 0.25\%$
MoNet	$\mathbf{81.69} \pm 0.48\%$	$\mathbf{78.81} \pm 0.44\%$

DCNN, GCN and MoNet were trained and tested 50 times. Shown results are average accuracy

MoNet - Results (FAUST)

FAUST dataset = 100 human bodyscans of 10 different persons

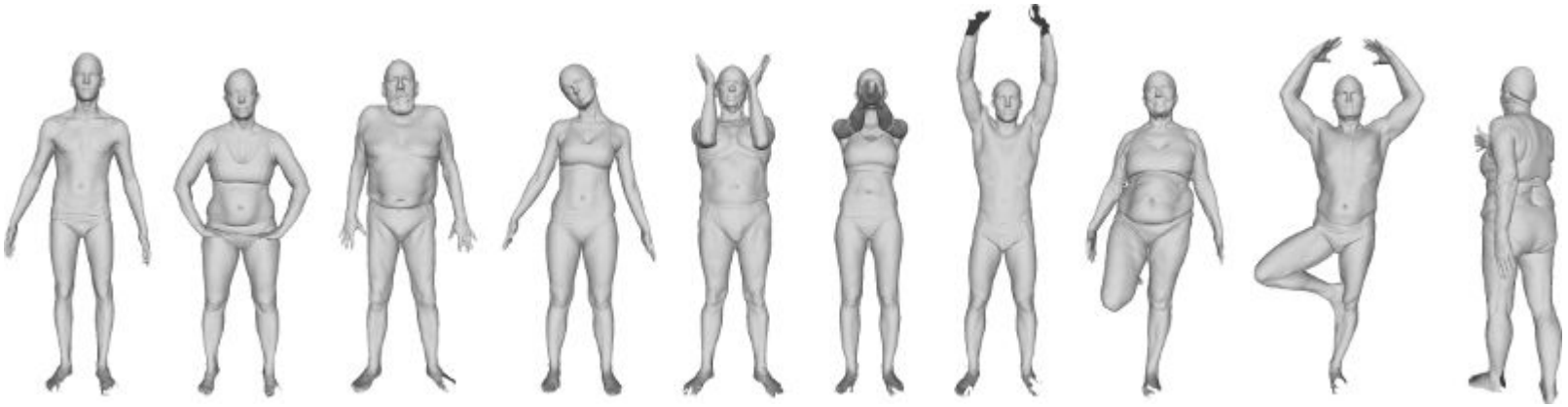


Image: [FAUST]

MoNet - Results (FAUST)

Task: find corresponding points between 3D models

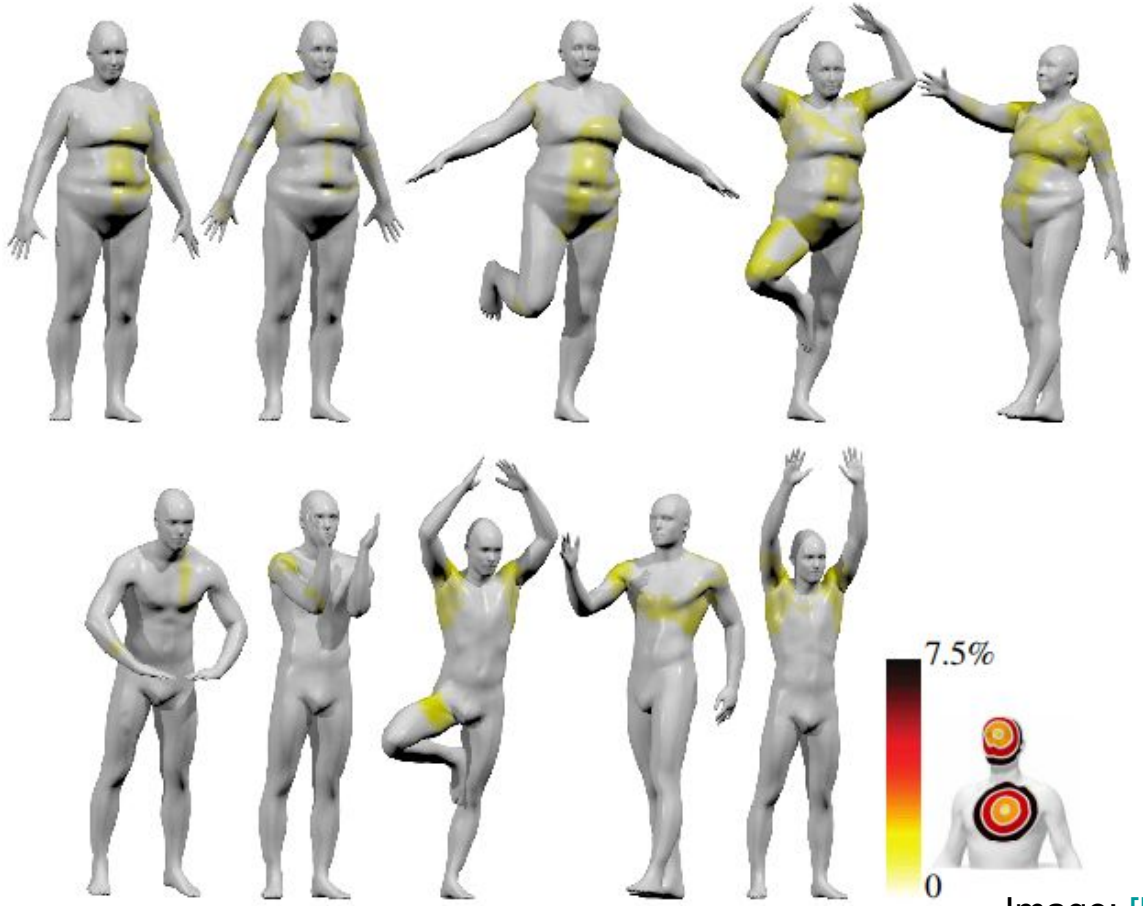


Image: [MoNet_Paper]

MoNet - Results (FAUST)

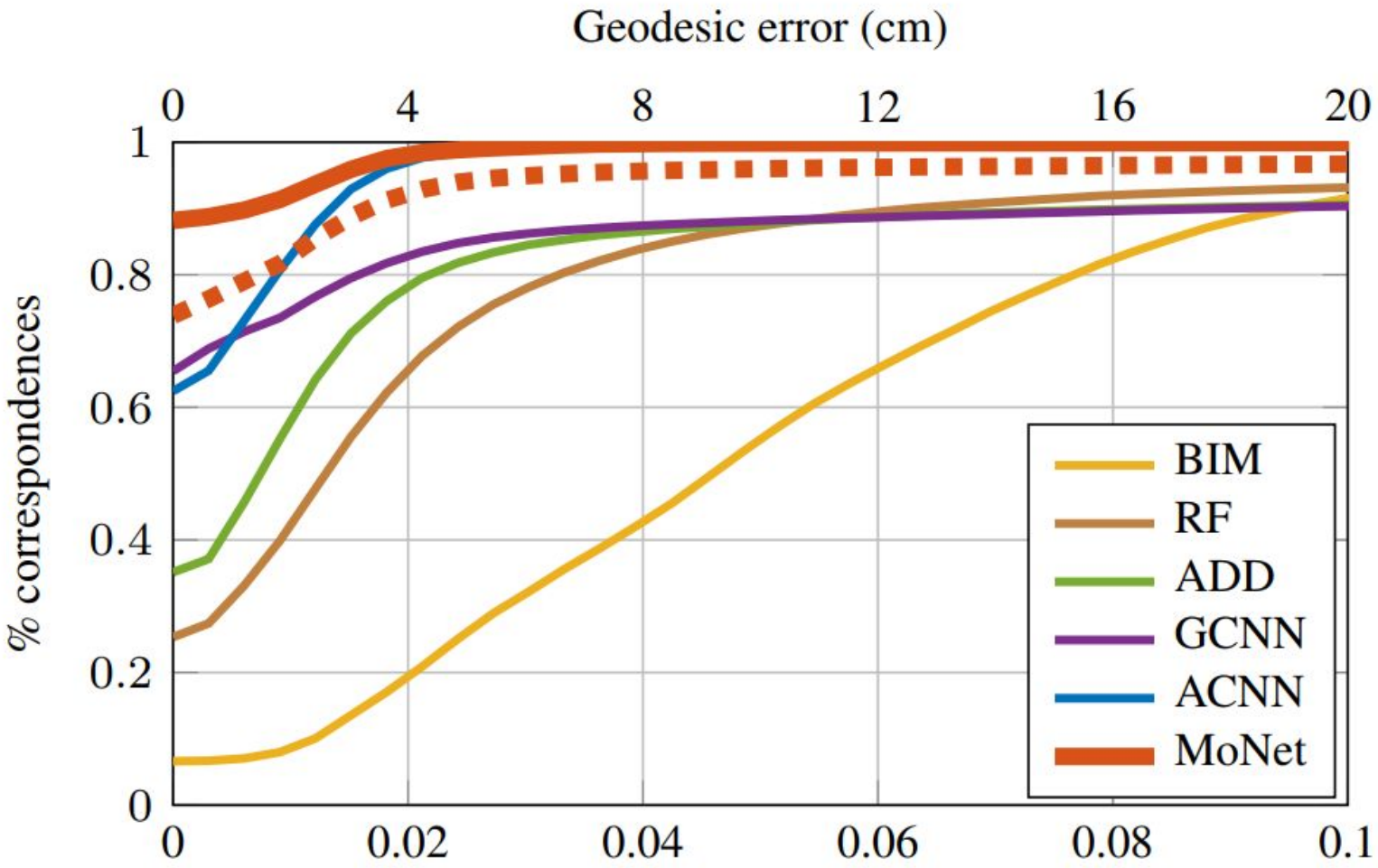


Image: [MoNet_Paper]

MoNet - Results (FAUST)

Conclusion

Positive:

+ Really powerful framework: **generic but simple**
+ Michael Bronstein held a interesting lecture at TUM last summer about geometric deep learning

Possible improvements:

- "w(u) is a convenient choice"

→ other functions might even work better!

- They could have explained more "why"

References

[Cat]: https://www.eastcottvets.co.uk/77/Kitten-Vaccination-Prices/ (26.12.2017)
[Sound]: https://pixabay.com/de/audio-musik-sfa-jazz-ton-welle-1293262/ (26.12.2017)
[SocialGraph]: http://483eclass.com/fall16/build_1/Hung/index.html (26.12.2017)
[TRex]: https://i.pinimg.com/736x/1d/b8/1b/1db81b7773c46bf3b3631a0c84b47385--quadrilateral-scripts.jpg (26.12.2017)
[M.Bronstein] https://www.dropbox.com/s/r4bzfux4ter8902/TUM%20Tutorial.pdf?dl=0 (07.01.2018)
[FAUST] http://faust.is.tue.mpg.de/overview (08.01.2018)
[MoNet_Paper] http://openaccess.thecvf.com/content_cvpr_2017/papers/Monti_Geometric_Deep_Learning_CVPR_2017_paper.pdf (16.12.2018)

Thank you

(Q&A now)