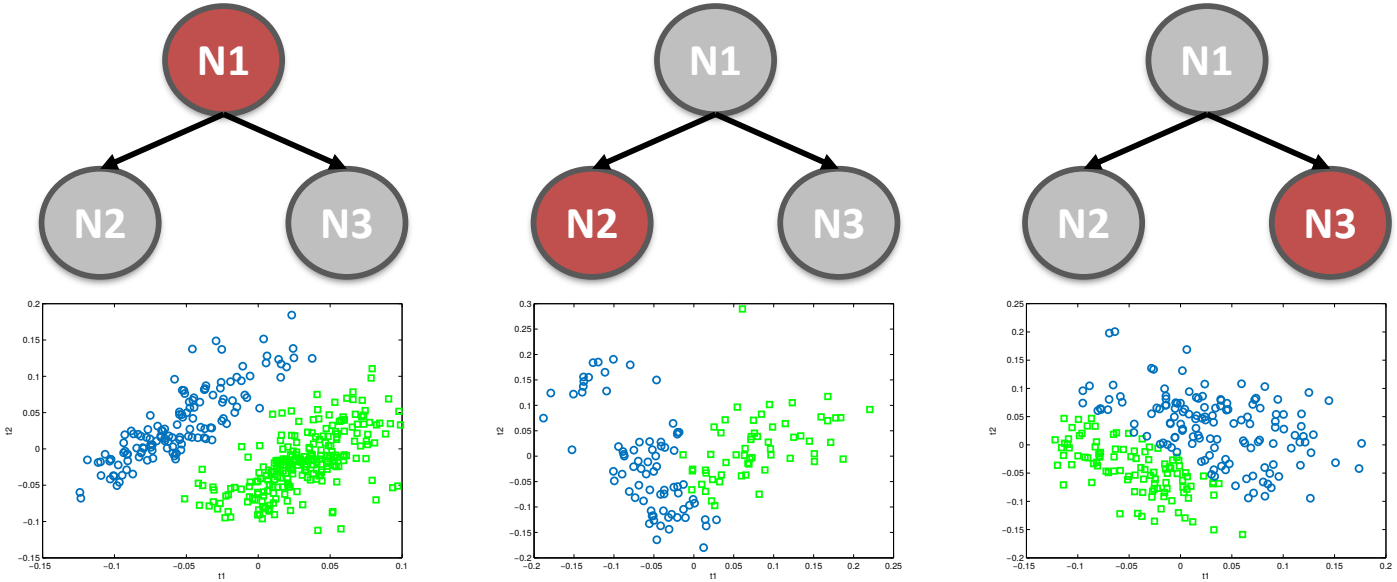
Karlsruhe Institute of Technology

Action Unit Intensity Estimation using Hierarchical Partial Least Squares

Tobias Gehrig*, Ziad Al-Halah*, Hazım Kemal Ekenel, Rainer Stiefelhagen | May 06, 2015

* **Both authors contributed equally to this study.**

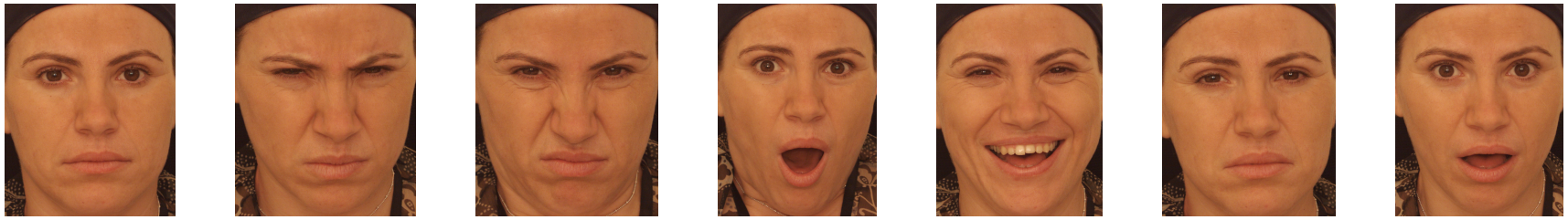
INSTITUTE FOR ANTHROPOMATICS AND ROBOTICS, COMPUTER VISION FOR HUMAN-COMPUTER INTERACTION LAB



Motivation

How can facial expressions be described?

Most popular way: Prototypic expressions from emotions



Neutral | Anger | Disgust | Fear | Happy | Sadness | Surprise

Problems:

- Rarely occur in real-life in their idealized form
- Context is important:
  $\Rightarrow$ Judging only by face, he could be happy, but is actually frustrated

by star5112, on Flickr

Motivation

How can facial expressions be described?

Description without interpretation

- Facial Action Coding System (FACS) (Ekman and Friesen 1978 [1])

Upper Face Action Units

AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
					
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Cheek Raiser	Lid Tightener

Lower Face Action Units

AU 9	AU 10	AU 11	AU 12	AU 15	AU 17	AU 18
						
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Lip Corner Depressor	Chin Raiser	Lip Puckerer

AU 20	AU 23	AU 24	AU 25	AU 26	AU 27	AU 43
						
Lip Stretcher	Lip Tightener	Lip Pressor	Lips Part	Jaw Drop	Mouth Stretch	Eye Closure

Images taken from Tian et al. 2005 [5]

- **Intensity Estimation**

0: not active	A: Trace	B: Slight	C: Marked Pronounced	D: Severe	E: Extreme Maximum

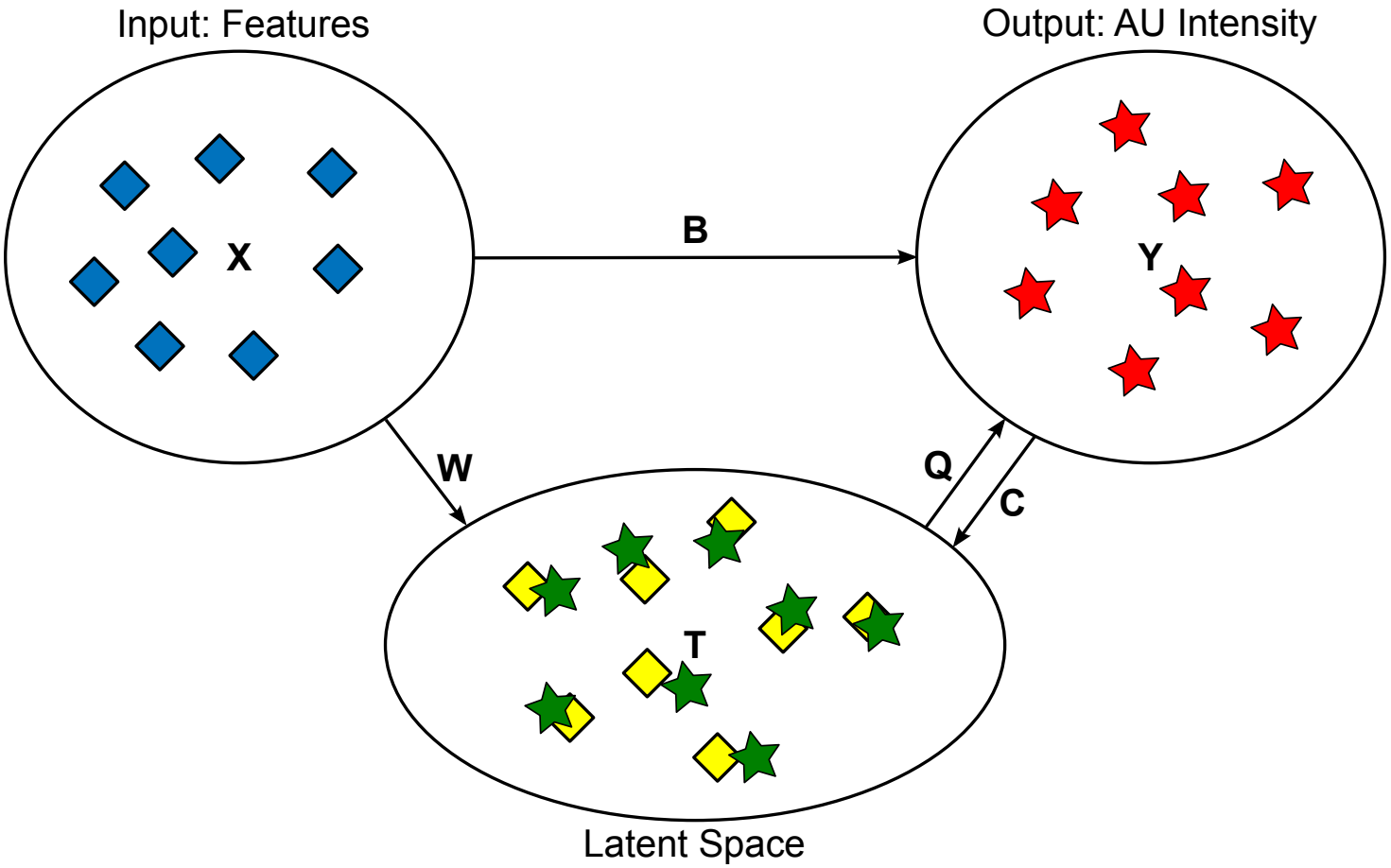
AU Intensity Estimation is Non-linear

- Task is non-linear due to variations in:
  - illumination
  - gender
    - eyebrow shape
  - age
    - skin texture and muscle structure
  - ethnicity
    - face shape

Images taken from Bosphorus database (Savran et al. 2008 [3])

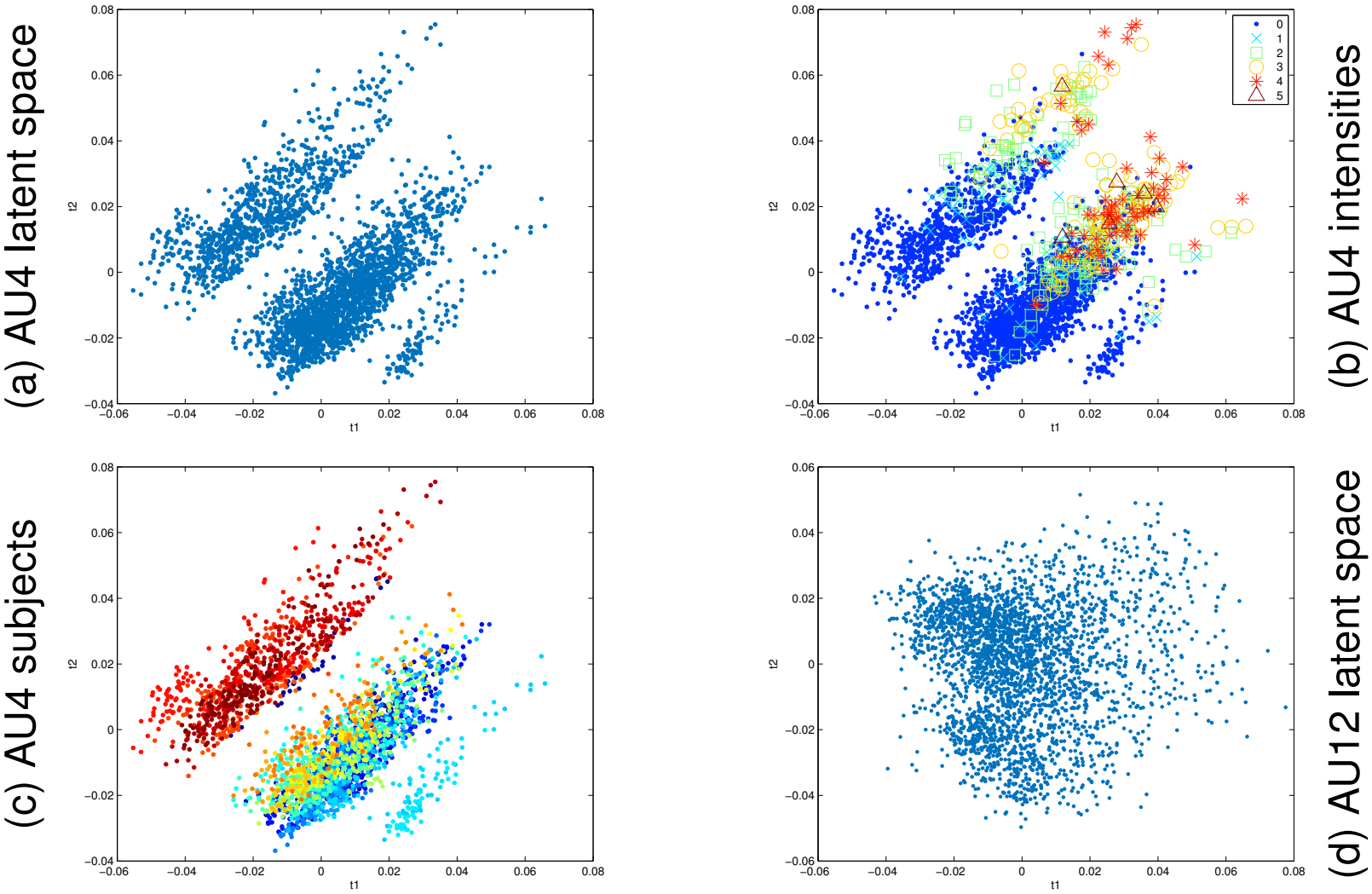
- **Common approach:** specific features or non-linear machine learning
- **Drawback:** non-linear machine learning often overfits and is slow
- **Our solution:** hierarchical approach using locally linear models

Partial Least Square (PLS) Analysis



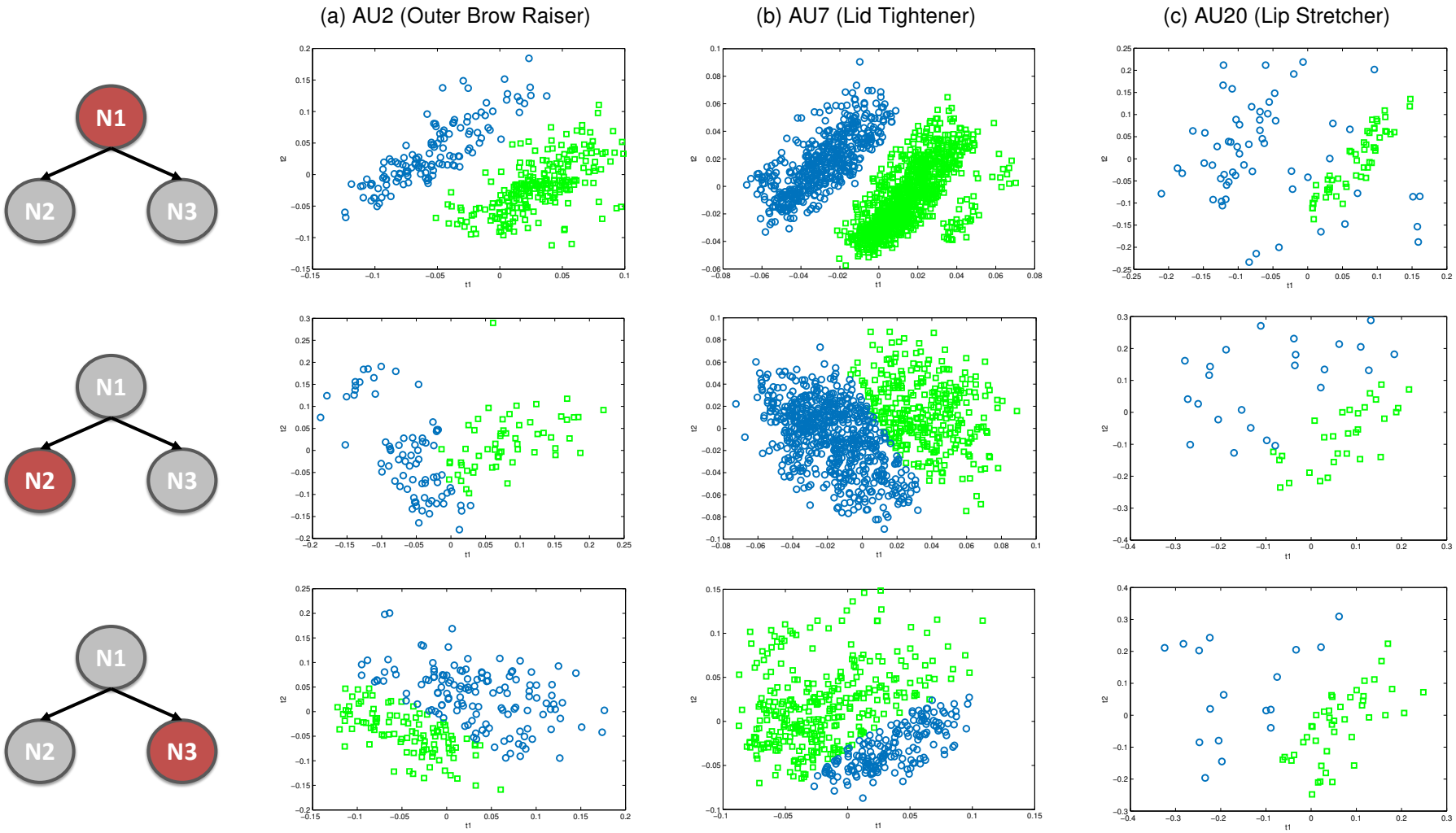
Partial Least Square Analysis

Latent space (first two dimensions)



Local PLS model latent spaces

First two levels of the hierarchy for different AUs

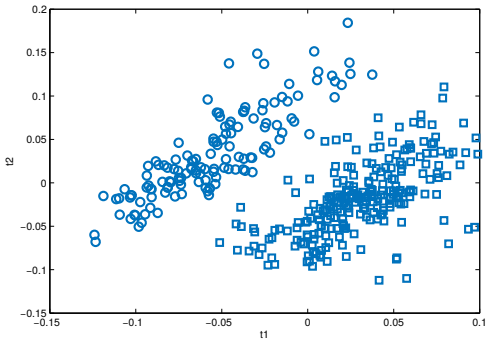


Hierarchical PLS

Algorithm

input : $\mathbf{X}$, $\mathbf{Y}$, (S=2)
output: model
$PLS_{root} \leftarrow \text{PLS\_fit}(\mathbf{X}, \mathbf{Y})$
$(model, root) \leftarrow \text{add\_to\_hierarchy}([\,], PLS_{root})$
$node\_list \leftarrow \text{push}(root, node\_list)$
while *not is_empty(node_list)* **do**
  $n_i \leftarrow \text{pop}(node\_list)$
  $lt_{n_i} \leftarrow \text{get\_latent\_space}(\mathbf{X}_{n_i}, \mathbf{Y}_{n_i}, PLS_{n_i})$
  $C_{n_i} \leftarrow \text{cluster}(lt_{n_i}, S)$
  if *not is_good*(C_{n_i}) **then**
    continue
  end
  for $j \leftarrow 1$ **to** S **do**
    $idx_j \leftarrow \text{get\_cluster}(j, C_{n_i})$
    $PLS_j \leftarrow \text{PLS\_fit}(\mathbf{X}_{n_i}(idx_j), \mathbf{Y}_{n_i}(idx_j))$
    $(model, n_j) \leftarrow \text{add\_to\_hierarchy}(n_i, PLS_j)$
    $node\_list \leftarrow \text{push}(n_j, node\_list)$
  end
end

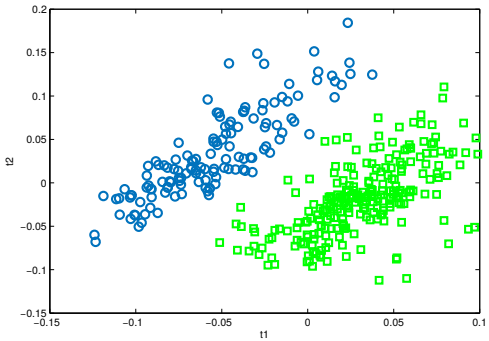
N1



Hierarchical PLS

Algorithm

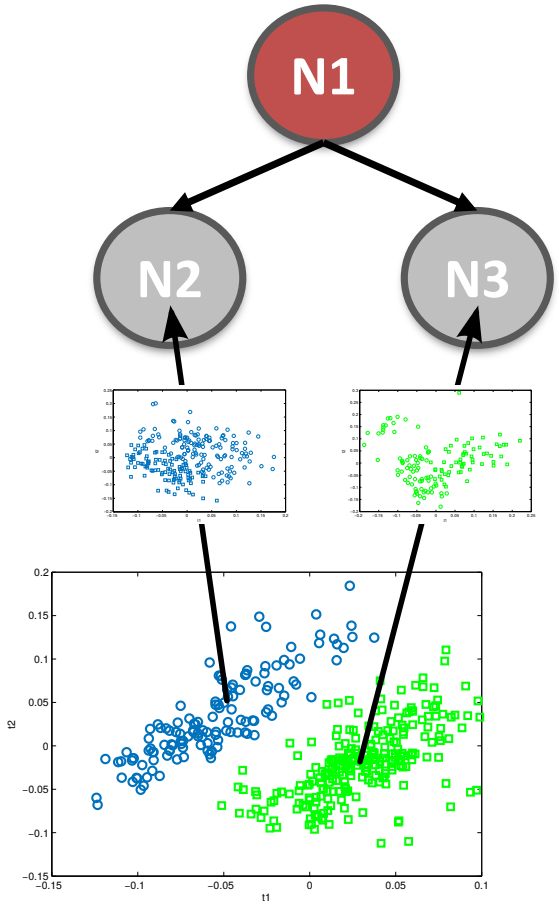
input : $\mathbf{X}$, $\mathbf{Y}$, (S=2)
output: model
$PLS_{root} \leftarrow \text{PLS\_fit}(\mathbf{X}, \mathbf{Y})$
$(model, root) \leftarrow \text{add\_to\_hierarchy}([\,], PLS_{root})$
$node\_list \leftarrow \text{push}(root, node\_list)$
while *not is_empty(node_list)* **do**
  $n_i \leftarrow \text{pop}(node\_list)$
  $lt_{n_i} \leftarrow \text{get\_latent\_space}(\mathbf{X}_{n_i}, \mathbf{Y}_{n_i}, PLS_{n_i})$
  $C_{n_i} \leftarrow \text{cluster}(lt_{n_i}, S)$
  if *not is_good*(C_{n_i}) **then**
    continue
  end
  for $j \leftarrow 1$ **to** S **do**
    $idx_j \leftarrow \text{get\_cluster}(j, C_{n_i})$
    $PLS_j \leftarrow \text{PLS\_fit}(\mathbf{X}_{n_i}(idx_j), \mathbf{Y}_{n_i}(idx_j))$
    $(model, n_j) \leftarrow \text{add\_to\_hierarchy}(n_i, PLS_j)$
    $node\_list \leftarrow \text{push}(n_j, node\_list)$
  end
end



Hierarchical PLS

Algorithm

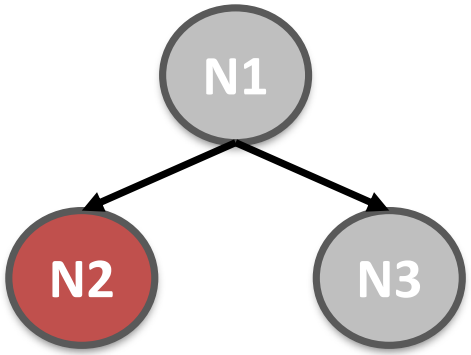
input : $\mathbf{X}$, $\mathbf{Y}$, (S=2)
output: model
$PLS_{root} \leftarrow \mathrm{PLS\_fit}(\mathbf{X}, \mathbf{Y})$
$(model, root) \leftarrow \mathrm{add\_to\_hierarchy}([\,], PLS_{root})$
$node\_list \leftarrow \mathrm{push}(root, node\_list)$
while *not is_empty(node_list)* **do**
  $n_i \leftarrow \mathrm{pop}(node\_list)$
  $lt_{n_i} \leftarrow \mathrm{get\_latent\_space}(\mathbf{X}_{n_i}, \mathbf{Y}_{n_i}, PLS_{n_i})$
  $C_{n_i} \leftarrow \mathrm{cluster}(lt_{n_i}, S)$
  if *not is_good*(C_{n_i}) **then**
    continue
  end
  for $j \leftarrow 1$ **to** S **do**
    $idx_j \leftarrow \mathrm{get\_cluster}(j, C_{n_i})$
    $PLS_j \leftarrow \mathrm{PLS\_fit}(\mathbf{X}_{n_i}(idx_j), \mathbf{Y}_{n_i}(idx_j))$
    $(model, n_j) \leftarrow \mathrm{add\_to\_hierarchy}(n_i, PLS_j)$
    $node\_list \leftarrow \mathrm{push}(n_j, node\_list)$
  end
end

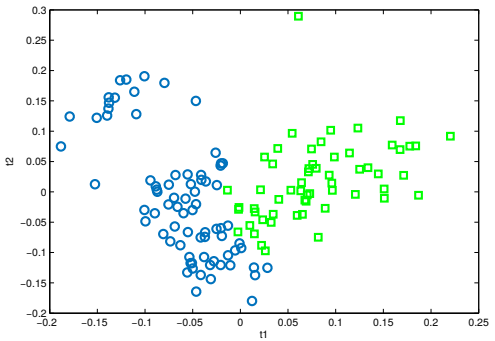


Hierarchical PLS

Algorithm

input : $\mathbf{X}$, $\mathbf{Y}$, (S=2)
output: model
$PLS_{root} \leftarrow \text{PLS\_fit}(\mathbf{X}, \mathbf{Y})$
$(model, root) \leftarrow \text{add\_to\_hierarchy}([\,], PLS_{root})$
$node\_list \leftarrow \text{push}(root, node\_list)$
while *not is_empty(node_list)* **do**
  $n_i \leftarrow \text{pop}(node\_list)$
  $lt_{n_i} \leftarrow \text{get\_latent\_space}(\mathbf{X}_{n_i}, \mathbf{Y}_{n_i}, PLS_{n_i})$
  $C_{n_i} \leftarrow \text{cluster}(lt_{n_i}, S)$
  if *not is_good*(C_{n_i}) **then**
    continue
  end
  for $j \leftarrow 1$ **to** S **do**
    $idx_j \leftarrow \text{get\_cluster}(j, C_{n_i})$
    $PLS_j \leftarrow \text{PLS\_fit}(\mathbf{X}_{n_i}(idx_j), \mathbf{Y}_{n_i}(idx_j))$
    $(model, n_j) \leftarrow \text{add\_to\_hierarchy}(n_i, PLS_j)$
    $node\_list \leftarrow \text{push}(n_j, node\_list)$
  end
end

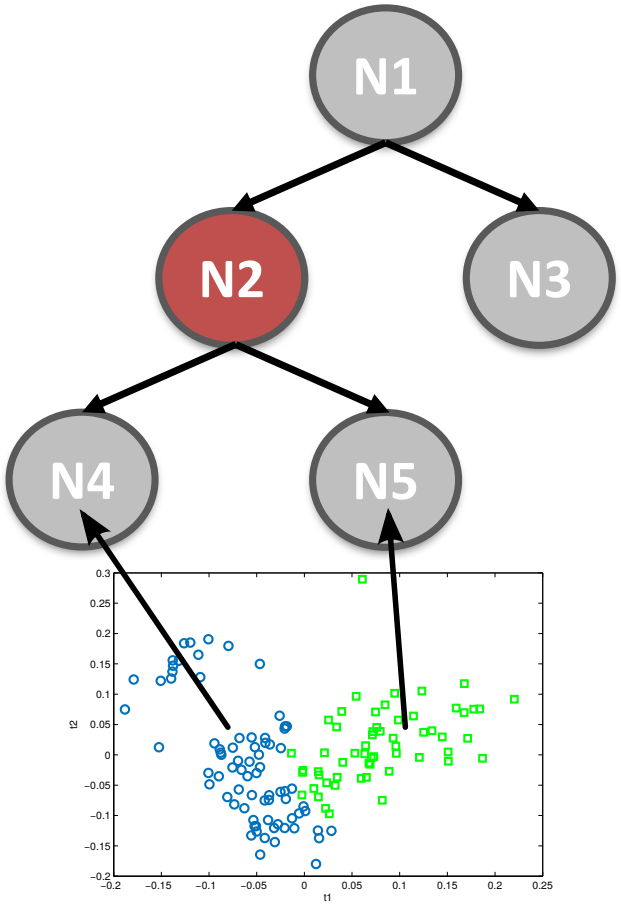




Hierarchical PLS

Algorithm

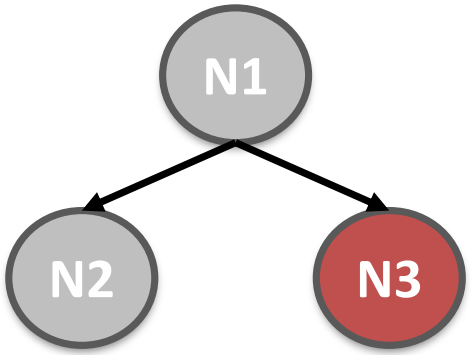
input : $\mathbf{X}$, $\mathbf{Y}$, (S=2)
output: model
$PLS_{root} \leftarrow \text{PLS\_fit}(\mathbf{X}, \mathbf{Y})$
$(model, root) \leftarrow \text{add\_to\_hierarchy}([\,], PLS_{root})$
$node\_list \leftarrow \text{push}(root, node\_list)$
while *not is_empty(node_list)* **do**
  $n_i \leftarrow \text{pop}(node\_list)$
  $lt_{n_i} \leftarrow \text{get\_latent\_space}(\mathbf{X}_{n_i}, \mathbf{Y}_{n_i}, PLS_{n_i})$
  $C_{n_i} \leftarrow \text{cluster}(lt_{n_i}, S)$
  if *not is_good*(C_{n_i}) **then**
    continue
  end
  for $j \leftarrow 1$ **to** S **do**
    $idx_j \leftarrow \text{get\_cluster}(j, C_{n_i})$
    $PLS_j \leftarrow \text{PLS\_fit}(\mathbf{X}_{n_i}(idx_j), \mathbf{Y}_{n_i}(idx_j))$
    $(model, n_j) \leftarrow \text{add\_to\_hierarchy}(n_i, PLS_j)$
    $node\_list \leftarrow \text{push}(n_j, node\_list)$
  end
end

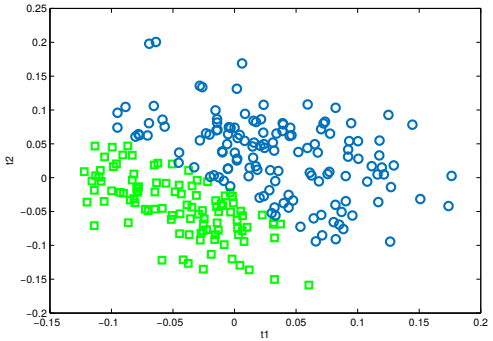


Hierarchical PLS

Algorithm

input : $\mathbf{X}$, $\mathbf{Y}$, (S=2)
output: model
$PLS_{root} \leftarrow \text{PLS\_fit}(\mathbf{X}, \mathbf{Y})$
$(model, root) \leftarrow \text{add\_to\_hierarchy}([\,], PLS_{root})$
$node\_list \leftarrow \text{push}(root, node\_list)$
while *not is_empty(node_list)* **do**
  $n_i \leftarrow \text{pop}(node\_list)$
  $lt_{n_i} \leftarrow \text{get\_latent\_space}(\mathbf{X}_{n_i}, \mathbf{Y}_{n_i}, PLS_{n_i})$
  $C_{n_i} \leftarrow \text{cluster}(lt_{n_i}, S)$
  if *not is_good*(C_{n_i}) **then**
    continue
  end
  for $j \leftarrow 1$ **to** S **do**
    $idx_j \leftarrow \text{get\_cluster}(j, C_{n_i})$
    $PLS_j \leftarrow \text{PLS\_fit}(\mathbf{X}_{n_i}(idx_j), \mathbf{Y}_{n_i}(idx_j))$
    $(model, n_j) \leftarrow \text{add\_to\_hierarchy}(n_i, PLS_j)$
    $node\_list \leftarrow \text{push}(n_j, node\_list)$
  end
end

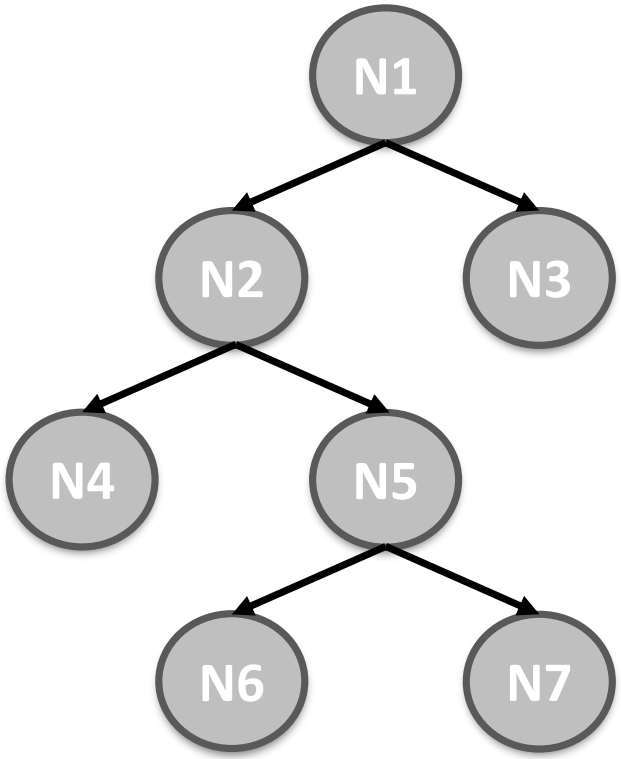




Hierarchical PLS

Algorithm

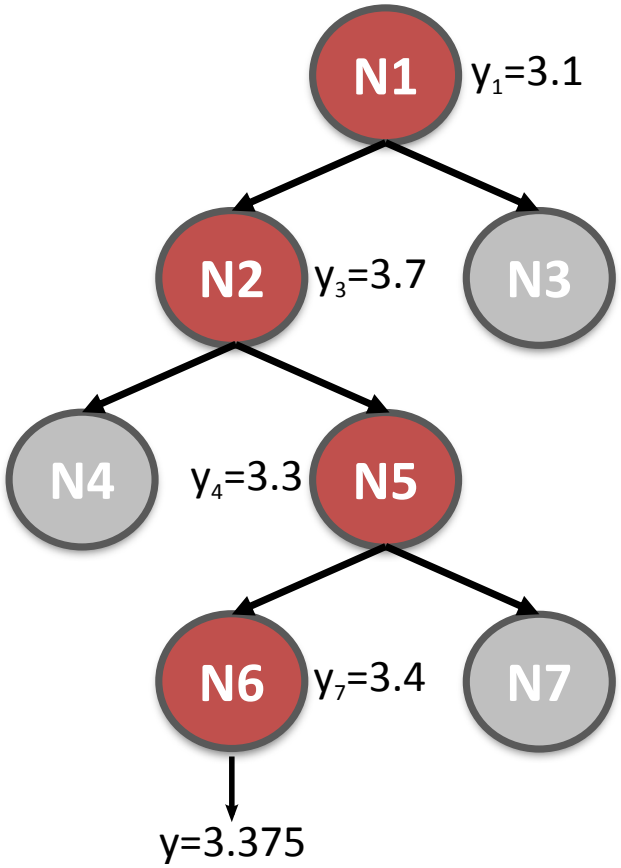
input : $\mathbf{X}$, $\mathbf{Y}$, (S=2)
output: model
$PLS_{root} \leftarrow \text{PLS\_fit}(\mathbf{X}, \mathbf{Y})$
$(model, root) \leftarrow \text{add\_to\_hierarchy}([\,], PLS_{root})$
$node\_list \leftarrow \text{push}(root, node\_list)$
while *not is_empty(node_list)* **do**
  $n_i \leftarrow \text{pop}(node\_list)$
  $lt_{n_i} \leftarrow \text{get\_latent\_space}(\mathbf{X}_{n_i}, \mathbf{Y}_{n_i}, PLS_{n_i})$
  $C_{n_i} \leftarrow \text{cluster}(lt_{n_i}, S)$
  if *not is_good*(C_{n_i}) **then**
    continue
  end
  for $j \leftarrow 1$ **to** S **do**
    $idx_j \leftarrow \text{get\_cluster}(j, C_{n_i})$
    $PLS_j \leftarrow \text{PLS\_fit}(\mathbf{X}_{n_i}(idx_j), \mathbf{Y}_{n_i}(idx_j))$
    $(model, n_j) \leftarrow \text{add\_to\_hierarchy}(n_i, PLS_j)$
    $node\_list \leftarrow \text{push}(n_j, node\_list)$
  end
end



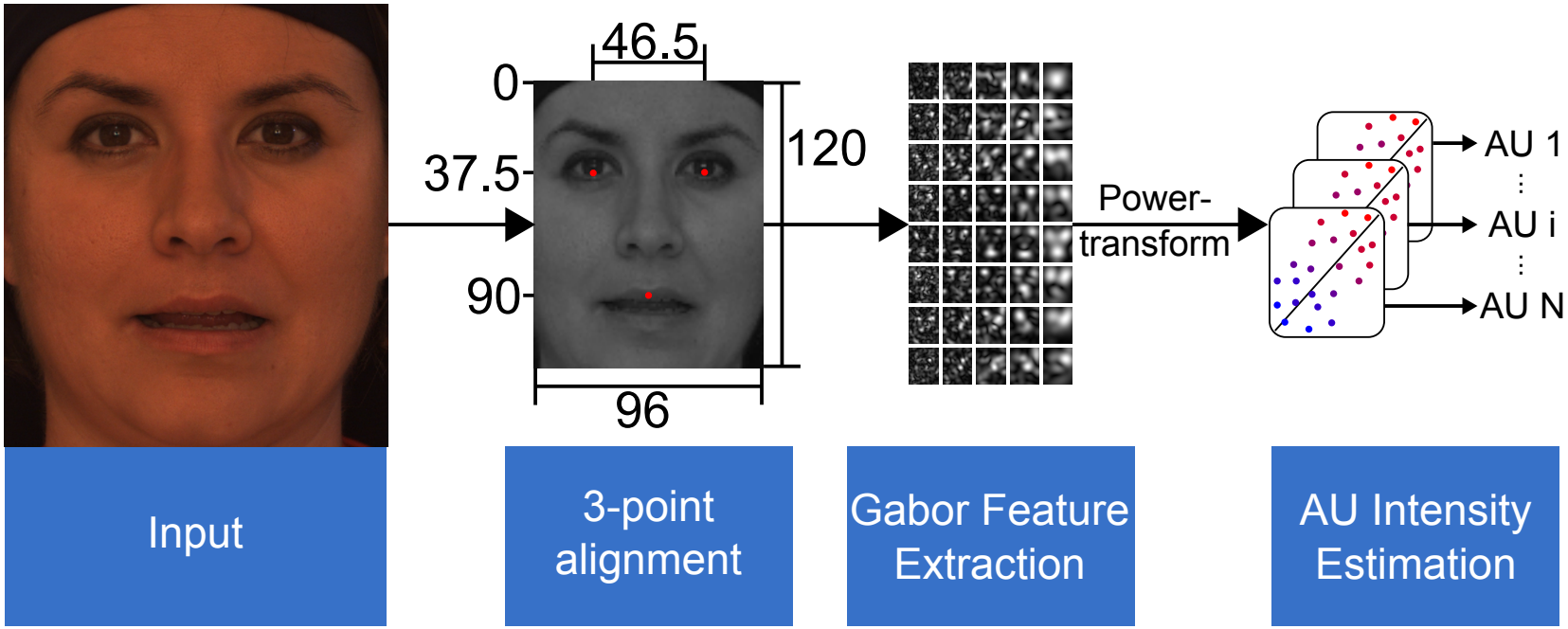
Hierarchical PLS

Model Combination

- Models at different levels of hierarchy are combined
- **Final estimate:** average over AU estimates in tree branch
  $\Rightarrow$ takes advantage of characteristics of models at each level
- Models towards root provide more stable and general estimations
- Models towards leaves are less stable but more accurate



System Overview



Experiments

Datasets

	Bosphorus	**Extended Cohn-Kanade (CK+)**
	Savran et al. 2008 [3]	Lucey et al. 2010 [2]
# Images	2902	117
# Subjects	105	73
# AUs	43	24
Experiment	within-dataset	across-dataset

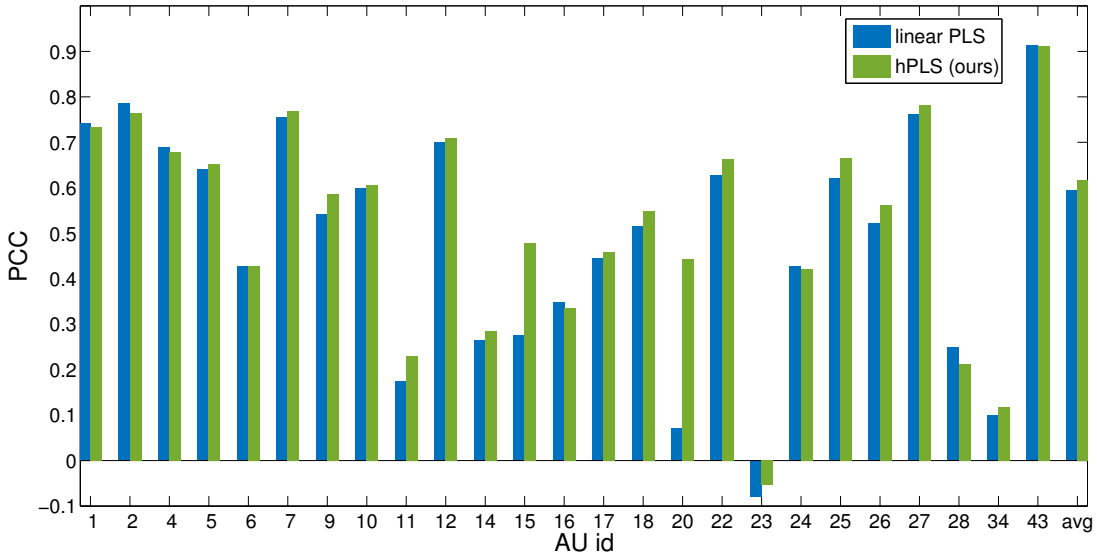
Experiments

Evaluation Setup

- Official Bosphorus Benchmark (Savran et al. 2012) [4]:
  - 10-fold cross validation separately for each of 25 AUs
  - **Metric:** Pearson correlation coefficient (PCC) over all fold estimates
  - **Overall metric:** weighted sum of AU PCC values
  - **Weights:** number of samples for corresponding AU
- **Additional metric:** intraclass correlation coefficient (ICC)
  $\Rightarrow$ better for this task

Within-Dataset Evaluation

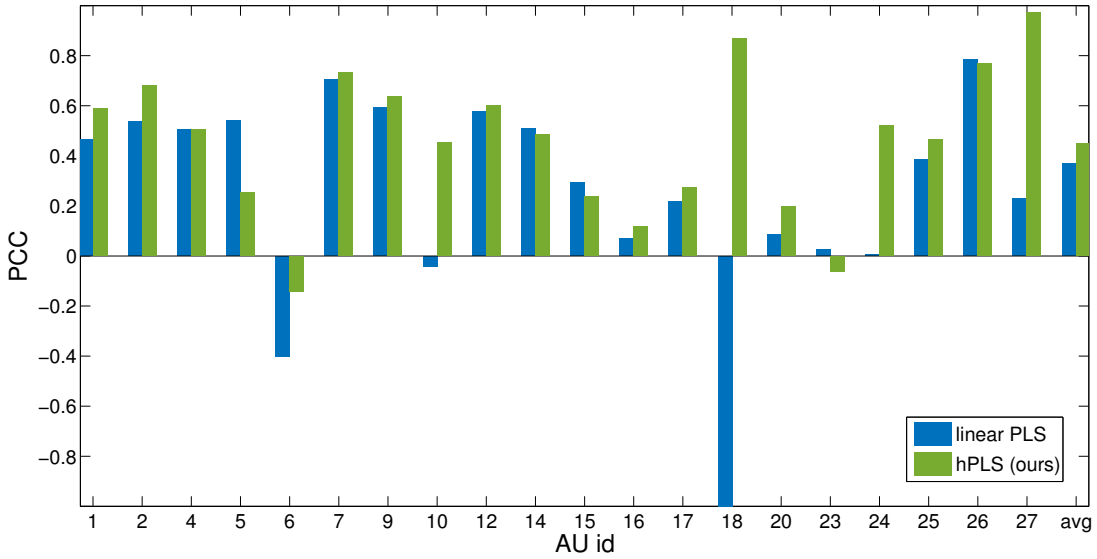
Results on Bosphorus



Metric	hPLS (ours)	linear-PLS	RBF-PLS	Savran et al. 2012 [4]
PCC	**61.7**	59.6	60.5	57.6
ICC	**57.9**	57.3	56.0	-

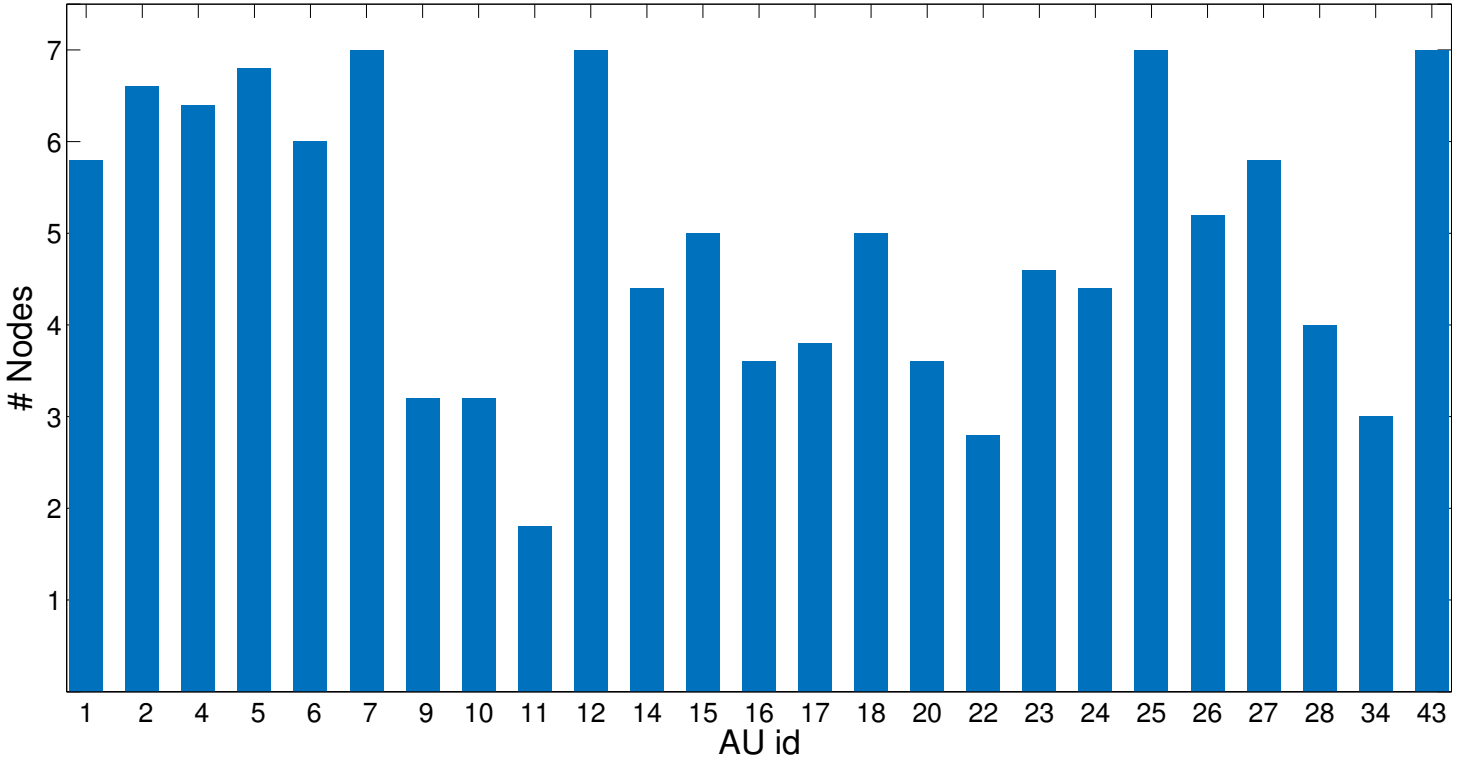
Across-Dataset Evaluation

Results on CK+



Metric	hPLS (ours)	linear-PLS	RBF-PLS
PCC	**44.9**	37.0	41.2
ICC	**41.2**	35.8	26.2

Computational Efficiency



Runtime: Crossvalidation on Bosphorus incl. I/O

- Matlab implementation of hPLS: around 32 minutes
- C++ implementation of RBF kernel PLS: more than 7 hours

Summary

- Novel hierarchical locally-linear regression model for AU intensity estimation
- Automatically captures non-linear relations in data
- Adapts to varying complexity by learning suitable hierarchical structure
- Outperforms kernel-based models with lower computational costs
- Approach is generic $\Rightarrow$ PLS can be replaced

Action Unit Intensity Estimation using Hierarchical Partial Least Squares

Tobias Gehrig*, Ziad Al-Halah*, Hazım Kemal Ekenel, Rainer Stiefelhagen

* Both authors contributed equally to this study.

{tobias.gehrig, ziad.al-halah, rainer.stiefelhagen}@kit.edu, ekenel@itu.edu.tr

http://cvhci.anthropomatik.kit.edu

References I

Paul Ekman and Wallace V. Friesen.
Facial Action Coding System: A Technique for the Measurement of Facial Movement.
Consulting Psychologists Press, Palo Alto, California, 1978.

P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, and Z. Ambadar.
The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression.
In *The 3rd IEEE Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB)*, 2010.

References II

A. Savran, N. Alyüz, H. Dibeklioğlu, O. Çeliktutan, B. Gökberk, B. Sankur, and L. Akarun.
Bosphorus Database for 3D Face Analysis.
In *The First COST 2101 Workshop on Biometrics and Identity Management*, 2008.

A. Savran, B. Sankur, and M. T. Bilge.
Regression-based Intensity Estimation of Facial Action Units.
Image and Vision Computing, 2012.

Y.-L. Tian, T. Kanade, and J. F. Cohn.
Facial Expression Analysis, chapter 11.
Springer, 2005.

Overview

Karlsruhe Institute of Technology

Motivation
- How can facial expressions be described?

Approach
- AU Intensity Estimation is Non-linear
- Partial Least Squares (PLS) Analysis
- Hierarchical PLS (hPLS)

Experiments
- System Overview
- Datasets
- Evaluation Setup
- Within-Dataset Evaluation
- Across-Dataset Evaluation
- Computational Efficiency

Conclusions