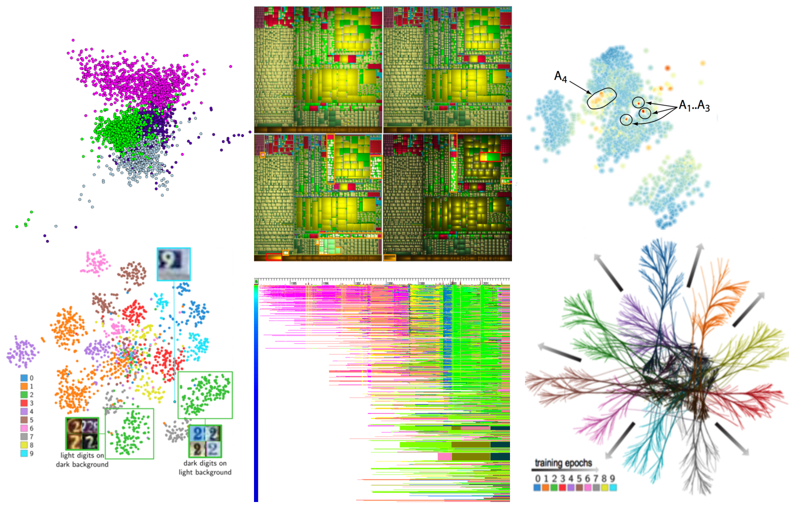
Visual Analytics for Opening the Black Box of Classifier Design



prof. dr. Alexandru (Alex) Telea

Department of Mathematics and Computer Science
University of Groningen, the Netherlands

Introduction

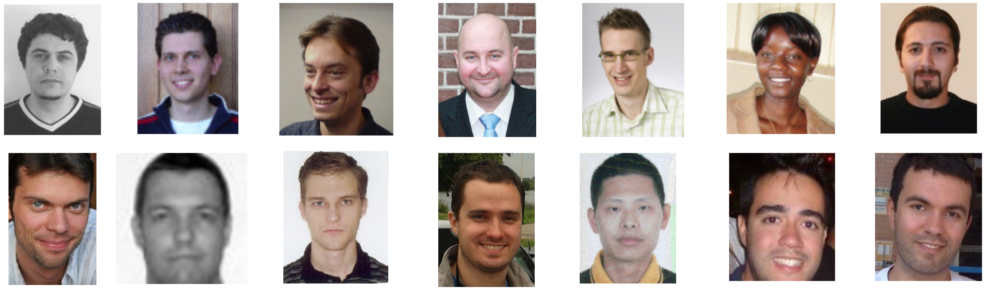
Who am I?

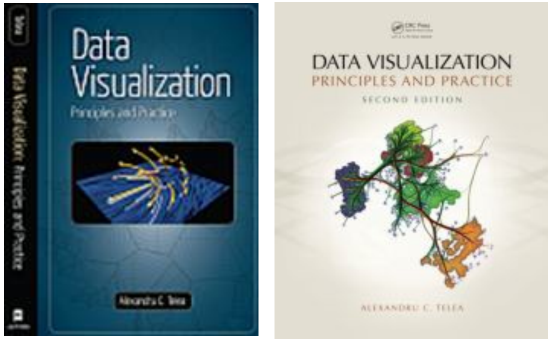
- professor in computer science / multiscale analytics @ RuG (since 2007)
- chair/steering committee ACM SOFTVIS / IEEE VISSOFT (since 2007)
- 14 PhD students, 60+ MSc students
- 200 international publications in visual data analytics
- co-founder SolidSource BV

www.cs.rug.nl/~alext

SolidSource IT

www.solidsourceit.com

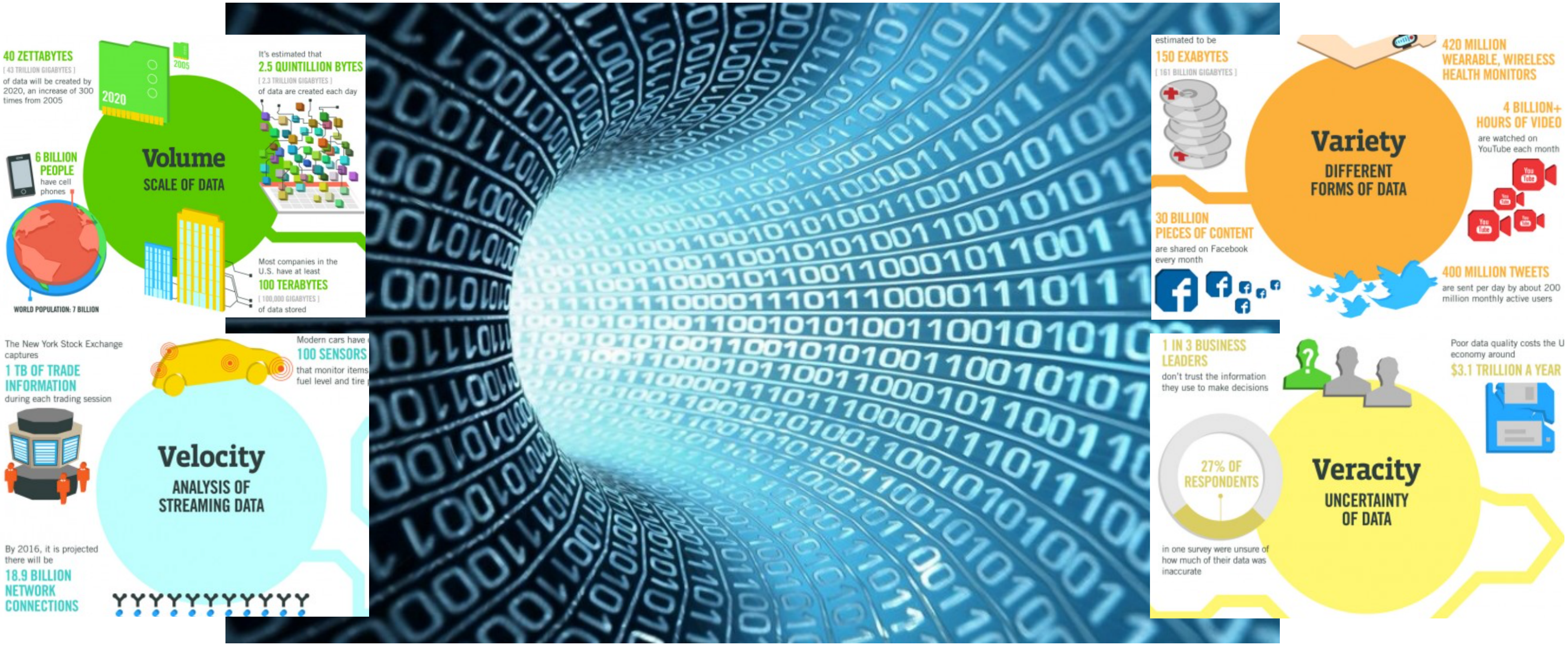




Data Visualization: Principles and Practice
A. K. Peters, 2008 / 2014

Why is Visualization Needed for Big Data?

The 'four V' challenges of big data



Volume: in 2010-2012, the humanity has created more data than it has previously in its history*

Velocity: the speed of generating data already exceeds storage capacities and processing power

Variety: data is numbers, text, images, maps, sounds, video, networks, relations, … anything

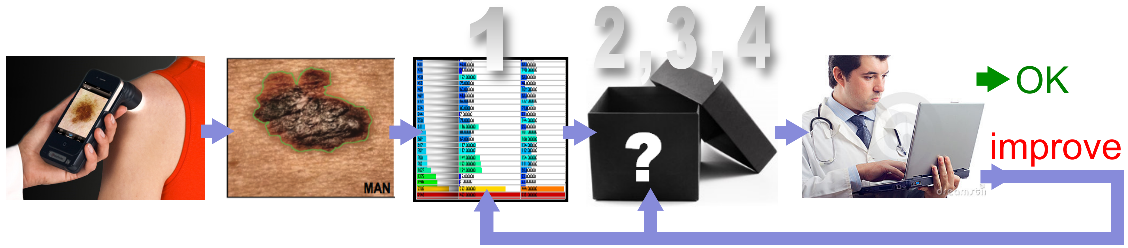
Veracity: more data = more noise = more trouble: How do we know we found all is in it?

If data is the modern-age oil…
visualization is an exploitation engine

* www.emc.com/leadership/programs/digital-universe.htm

** A. Kirk, Visualization: A success design story, Packt Publ., 2012

Why is Visualization Needed for Classifier Design?



1. Domain exploration

- how do we know which **features** we can extract?
- how to tell the **quality** of the data?

2. Classifier diagnosis

- typical aggregate metrics (accuracy / area under ROC curve / discriminative power)
- if this value is high, all good
- but what if not? **What** has gone wrong?

3. Classifier comparison

- typical: compare aggregate metrics
- how to tell **where** and **why** behave classifiers differently?

4. Classifier improvement

- typical: black art (change some parameters, hope for the best, …)
- how to tell **what** and **why** causes problems?
- how to find **best/cheapest** direction for improvement?

1. Domain exploration

Question

1000 samples x 1 attribute
100 samples x 100 attributes

And why?

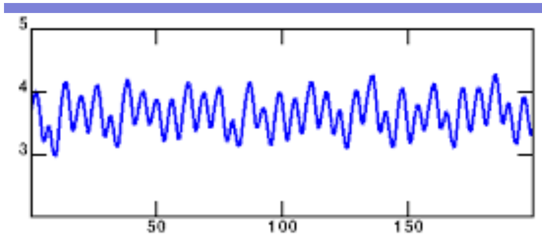
* www.emc.com/leadership/programs/digital-universe.htm
** A. Kirk, Visualization: A success design story, Packt Publ., 2012

1. Domain exploration

Problem: We deal with multivariate, non-spatial, abstract data

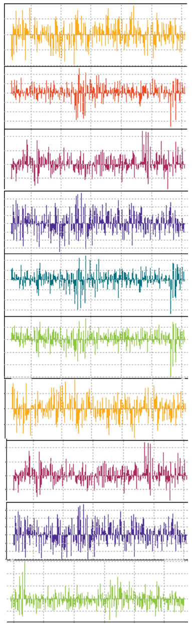
- univariate data: typically we compare a pair of patterns
- *m*-variate data: we have $\boldsymbol{m^2/2}$ **pairs** to compare!

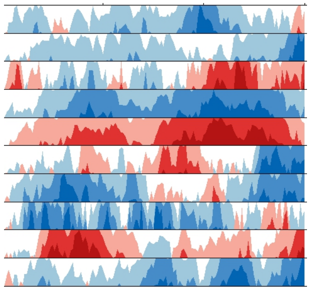
1000 samples x 1 attribute

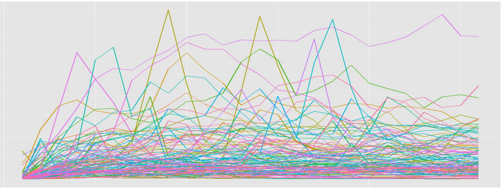


1D graphs/charts
work pretty well ☺

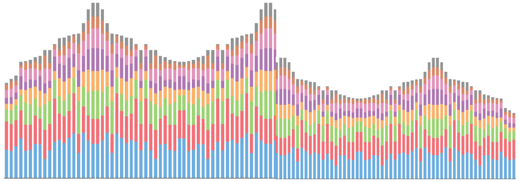
100 samples x 10 attributes

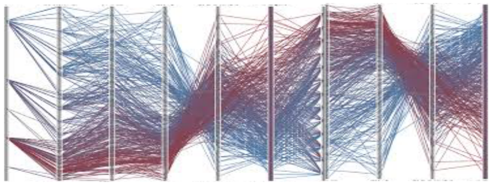






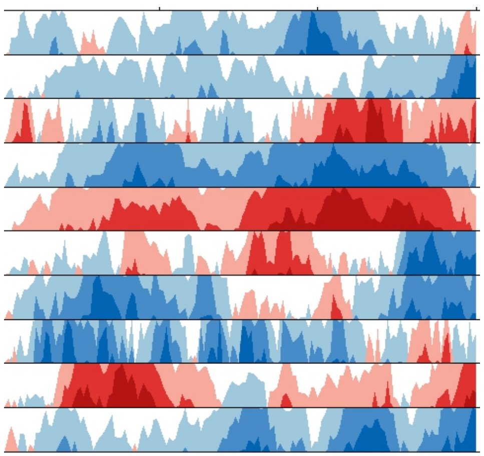
many chart kinds, many problems
(not scalable, cluttered, abstract, …)



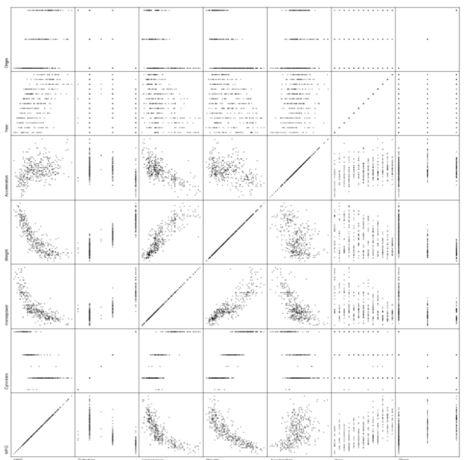


1. Domain exploration

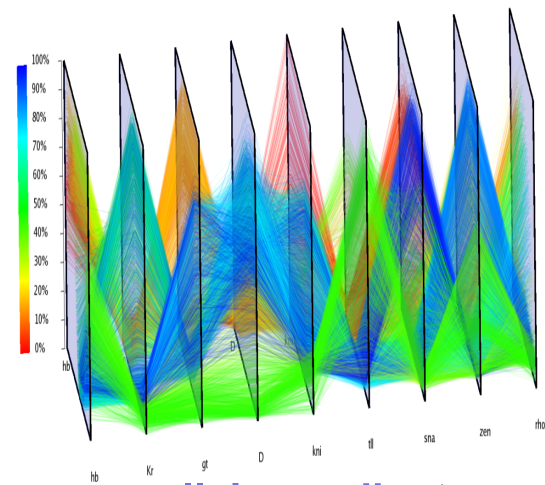
Current solutions: Very limited!



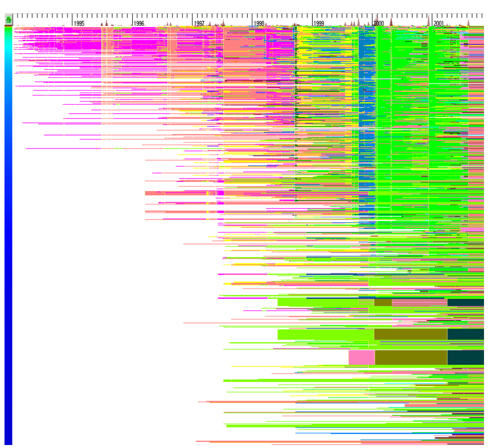
small multiples



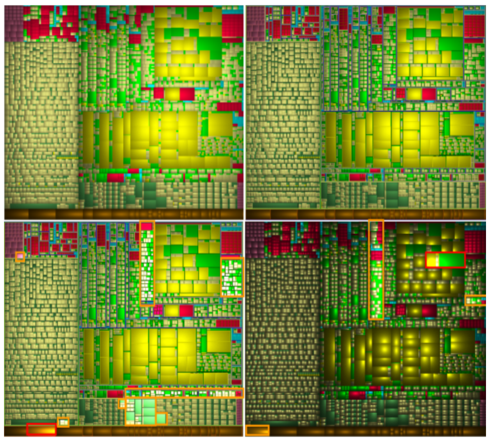
scatterplot matrices



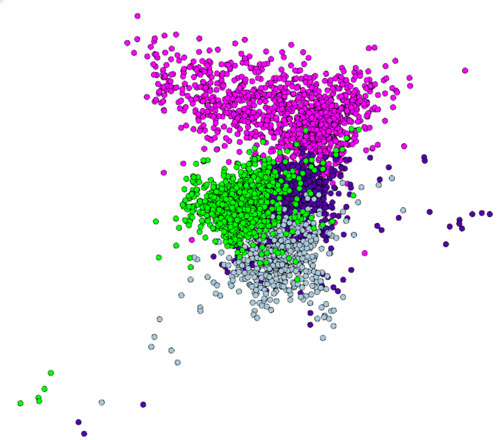
parallel coordinates



dense pixel techniques



treemaps



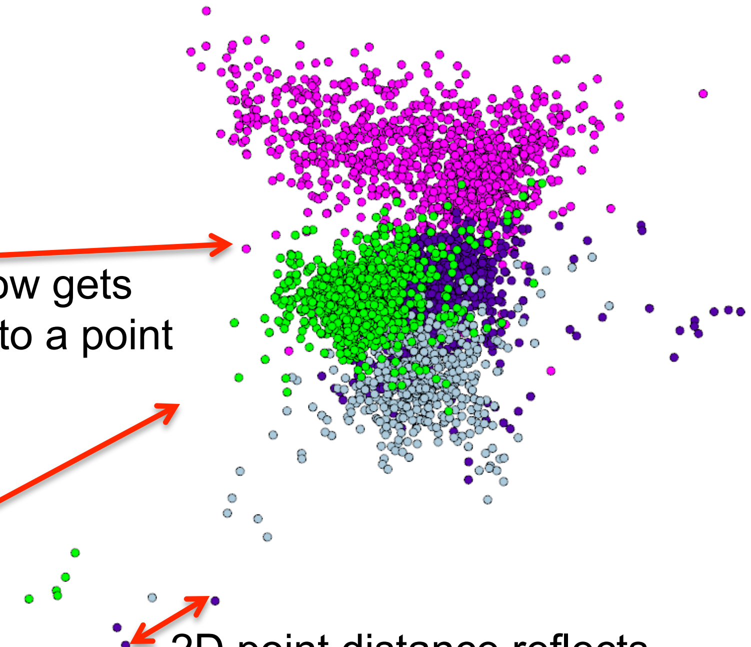
projections

Projections

Table

id	category	name	date	time	open	high	low	close
636	sif	SIF1	2004-11-29	13:00	.800000	0.800000	0.800000	0.800000
635	sif	SIF1	2004-11-29	14:00	.800000	0.800000	0.800000	0.800000
633	sif	SIF1	2004-11-29	16:00	.795000	0.795000	0.795000	0.795000
630	sif	SIF1	2004-11-30	14:00	.795000	0.795000	0.795000	0.795000
632	sif	SIF1	2004-11-30	12:00	.800000	0.800000	0.795000	0.795000
631	sif	SIF1	2004-11-30	13:00	.795000	0.795000	0.795000	0.795000
628	sif	SIF1	2004-11-30	16:00	.795000	0.795000	0.795000	0.795000
629	sif	SIF1	2004-11-30	15:00	.795000	0.795000	0.795000	0.795000
627	sif	SIF1	2005-00-02	12:00	.785000	0.790000	0.785000	0.790000
626	sif	SIF1	2005-00-02	13:00	.790000	0.795000	0.790000	0.795000
624	sif	SIF1	2005-00-02	15:00	.800000	0.800000	0.800000	0.800000
623	sif	SIF1	2005-00-03	12:00	.795000	0.795000	0.795000	0.795000
622	sif	SIF1	2005-00-03	13:00	.795000	0.795000	0.795000	0.795000
621	sif	SIF1	2005-00-03	14:00	.795000	0.795000	0.795000	0.795000
619	sif	SIF1	2005-00-03	16:00	.795000	0.795000	0.795000	0.795000
618	sif	SIF1	2005-00-06	11:00	.790000	0.790000	0.790000	0.790000
614	sif	SIF1	2005-00-06	15:00	.795000	0.795000	0.795000	0.795000
617	sif	SIF1	2005-00-06	12:00	.795000	0.795000	0.795000	0.795000
616	sif	SIF1	2005-00-06	13:00	.795000	0.795000	0.795000	0.795000
615	sif	SIF1	2005-00-06	14:00	.795000	0.795000	0.795000	0.795000
613	sif	SIF1	2005-00-06	16:00	.795000	0.795000	0.795000	0.795000
609	sif	SIF1	2005-00-07	14:00	.790000	0.795000	0.790000	0.795000
612	sif	SIF1	2005-00-07	11:00	.795000	0.795000	0.795000	0.795000
611	sif	SIF1	2005-00-07	12:00	.795000	0.795000	0.795000	0.795000
610	sif	SIF1	2005-00-07	13:00	.790000	0.790000	0.790000	0.790000
608	sif	SIF1	2005-00-07	15:00	.790000	0.790000	0.790000	0.790000
606	sif	SIF1	2005-00-08	13:00	.795000	0.795000	0.795000	0.795000
607	sif	SIF1	2005-00-08	12:00	.790000	0.790000	0.790000	0.790000
605	sif	SIF1	2005-00-08	14:00	.795000	0.795000	0.795000	0.795000

2D projection



a table row gets mapped to a point

color map values of a selected column

2D point distance reflects *n*D row distance

- extremely compact: one *n*-dimensional point = 1 pixel
- fast to compute (on GPU: 500K 100-dim points: <1 second)
- show underlying data grouping in classes
- can be shown by well-known scatterplot visualization

Projections

How to construct them?

1. Principal component analysis

- compute *n* eigenvectors e_i and eigenvalues w_i of the *m* *n*D points (table rows)
- select the two eigenvectors e_i for the two largest eigenvalues w_i
- project the nD points on the 2D plane spanned by the two largest eigenvectors
- pro's: simple to compute, many tools support this (linear) method
- con's: 2D distances typically **do not accurately reflect** *n*D distances

2. Nonlinear/local methods

- find *n*'<<*n* most representative points from the total of *m* *n*D points
- use a linear method to project the *n*' points in 2D
- fit remaining *n*-*n*' points around the projected points so they best preserve distances
- pro's: **accurately preserve distances** from *n*D to 2D
- con's: much more complex to implement, few(er) packages support such methods
- examples: MDS, t-SNE, LAMP, LSP, Glimmer, PLMP

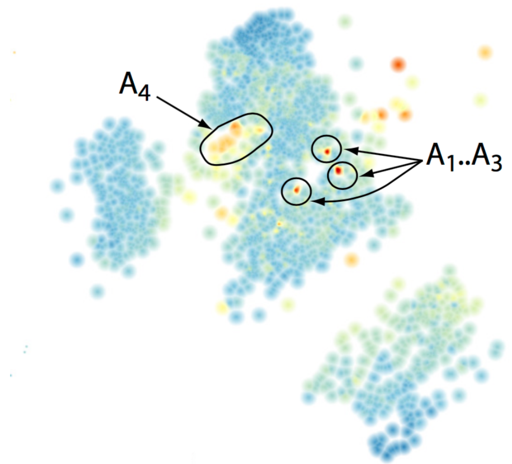
A good projection software toolkit: ProjectionExplorer (http://infoserver.lcad.icmc.usp.br/infovis2/Pex)

Projection Challenges (1/4)

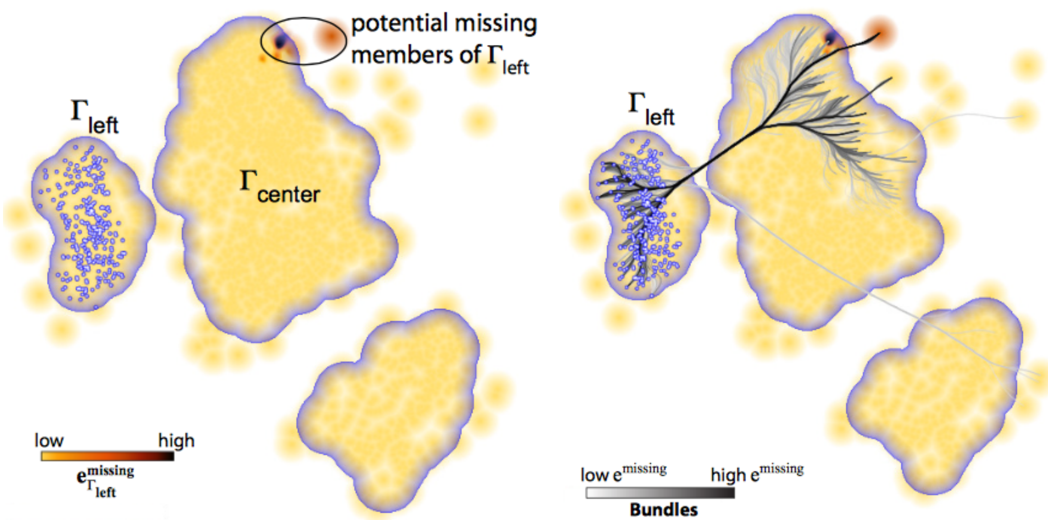
How to understand their veracity?

- false positives: points close in 2D but far in *n*D
- false negatives: points close in nD but far in 2D

False positives map
(false neighbors)



False negatives map
(missing neighbors)

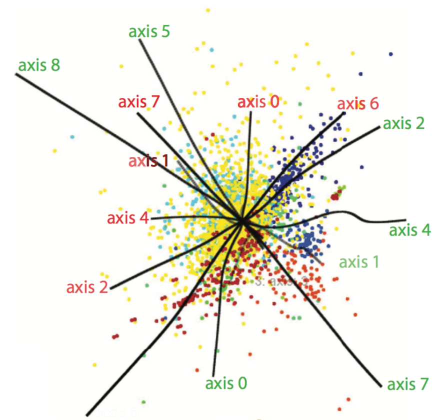


[Martins et al'14]

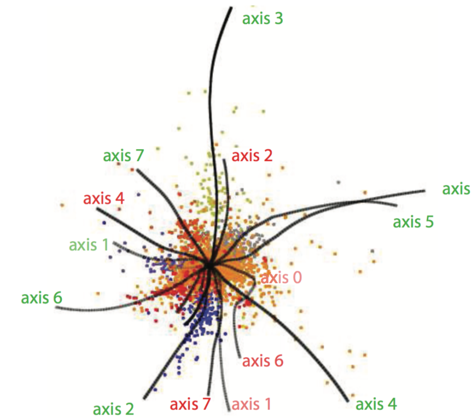
Projection Challenges (2/4)

How to see the *n*D variables?

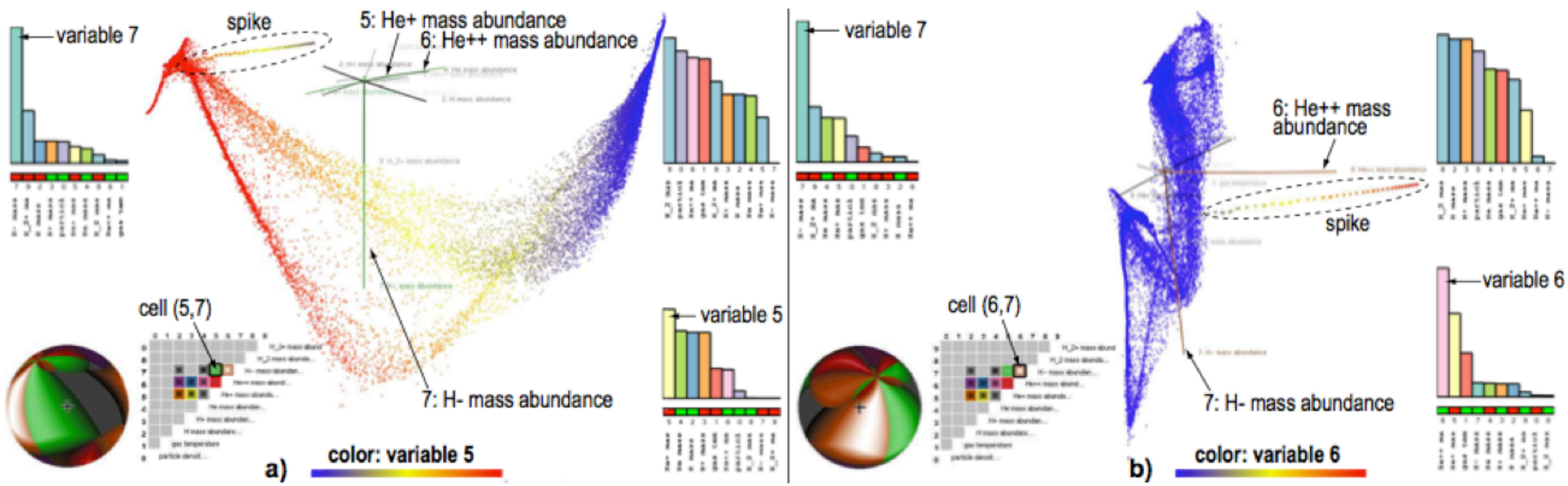
unannotated projection



biplot axes (good projection)



biplot axes (bad projection)



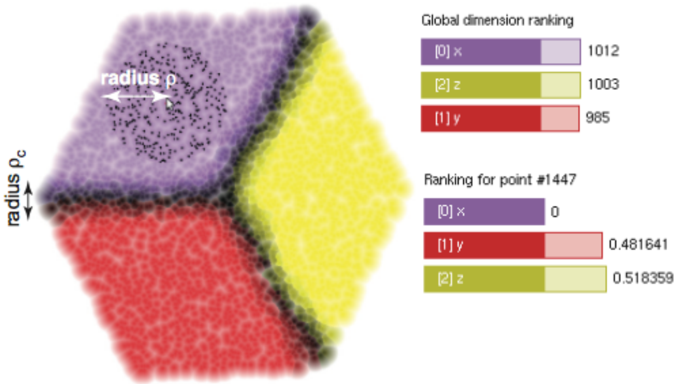
two viewpoints for a 3D projection showing usefulness of **axis legends**

[Coimbra et al'15]

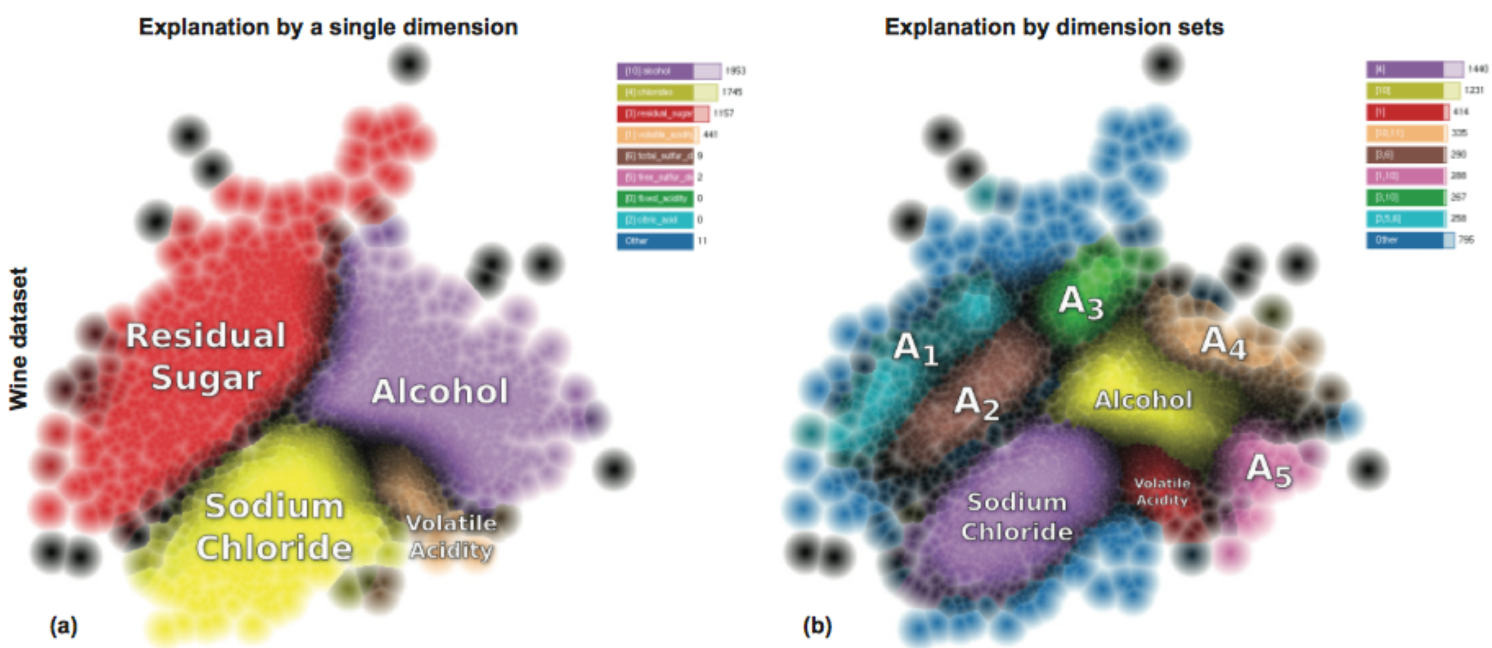
Projection Challenges (3/4)

How to see why observations are similar?

- visually detect and explain **groups** in a projection



Data: 2400 wine samples, 12 attributes/sample
Goal: see why wine sorts resemble each other



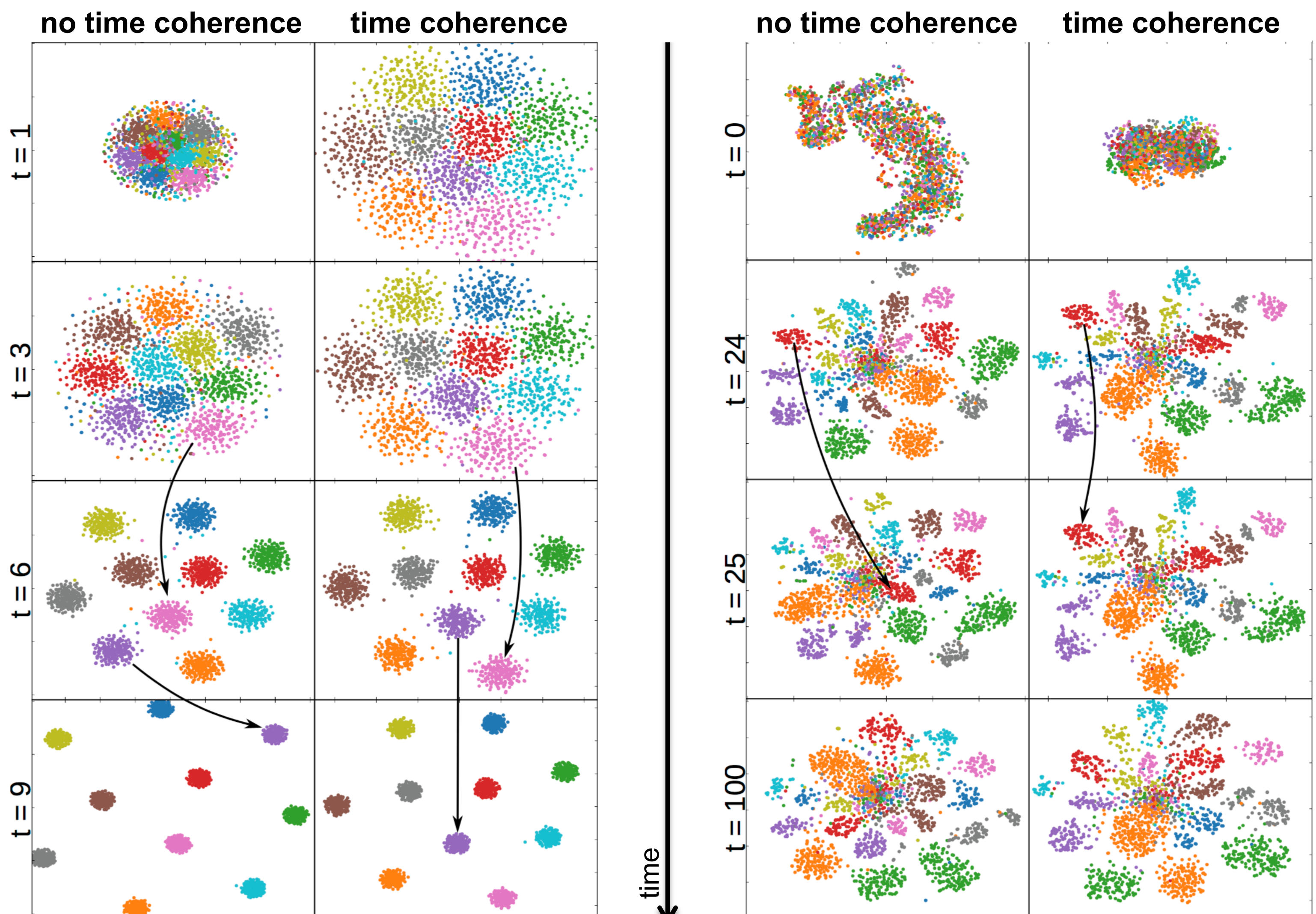
[da Silva et al'15]

Projection Challenges (4/4)

How to project time-dependent multivariate data?

- extend t-SNE to handle time-dependent data

[Rauber et al'16]



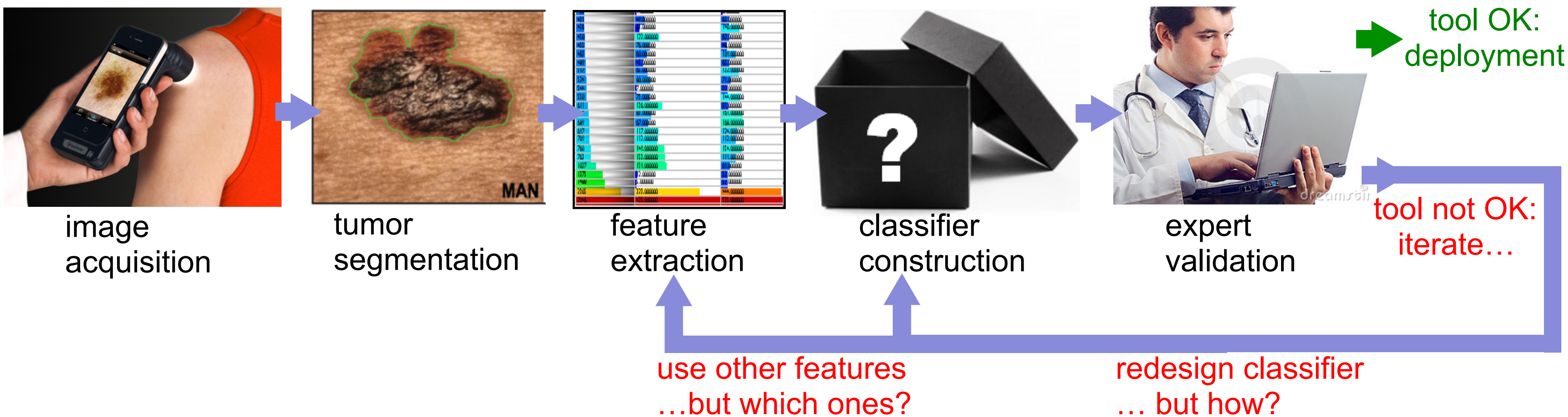
100-variate Gaussian (2000 points, 10 classes)

CNN 512-dim activations (2000 images, 10 classes), SVHN dataset

2. Feature selection for medical classifier design

- want to build an efficient and effective classifier for **skin lesion images**
- to be used for automatic melanoma (skin cancer) pre-detection
- skin cancer: most common worldwide; survival rate=25% if **diagnosed late**

Automated diagnosis pipeline



Challenges

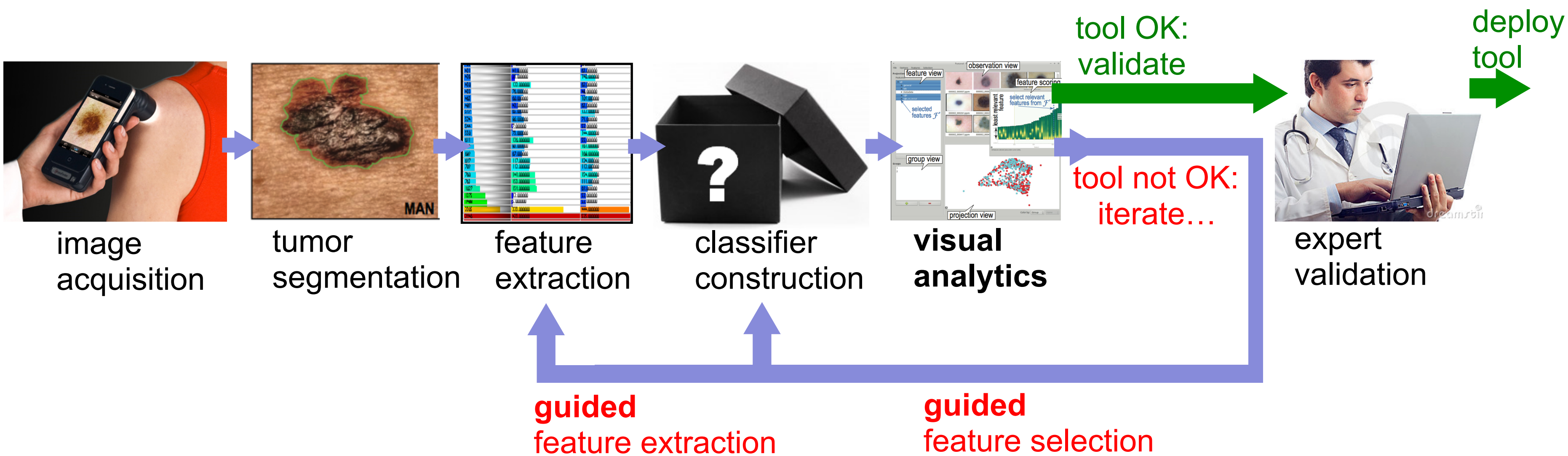
- classifier design is a **black-box**, magic-art science
- we can extract an **infinite** number of features – which are the good ones?
- how to design an effective **classifier** of skin images?

Visual analytics pipeline for classifier design

Application

- want to build an efficient and effective classifier for **skin lesion images**
- to be used for automatic melanoma (skin cancer) detection

Proposed automated pipeline



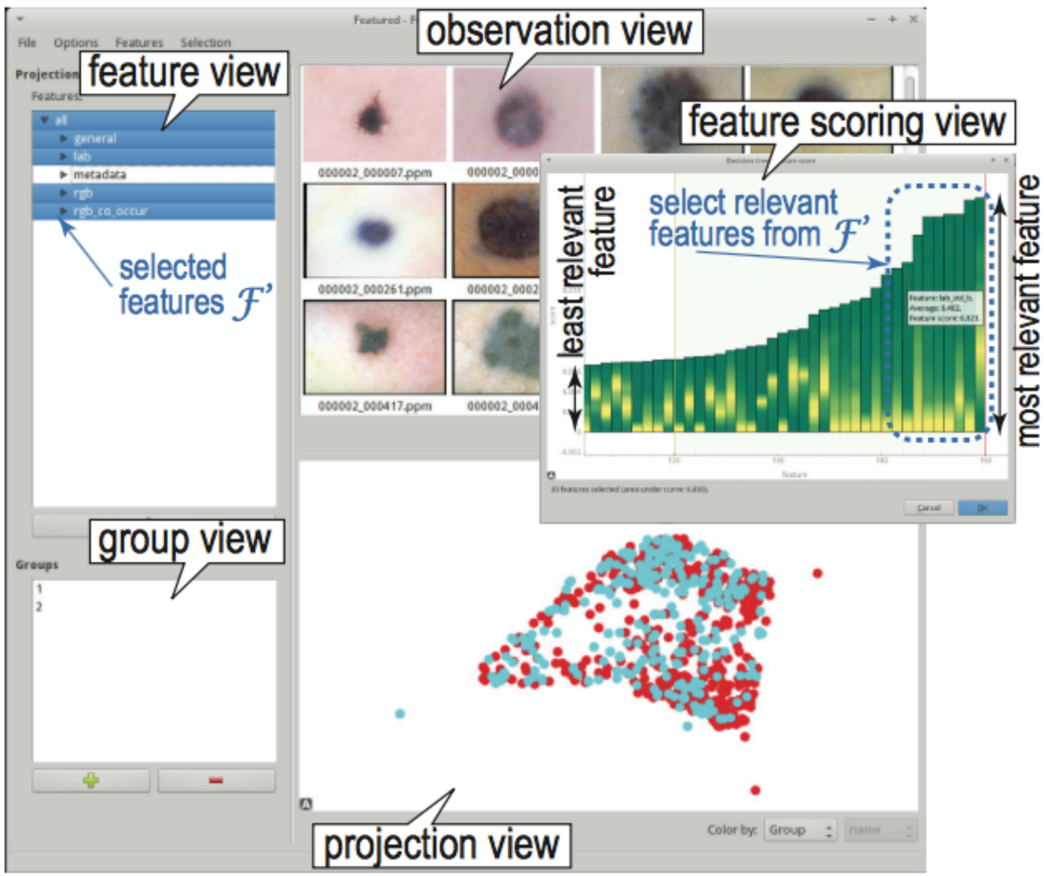
Advantages

- we **see** why classifier works (or not)
- we **see** which features are good (or not)
- visual analytics **guides** us towards improvement
- open the 'black box magic' of classifier design

Visual analytics for classifier design

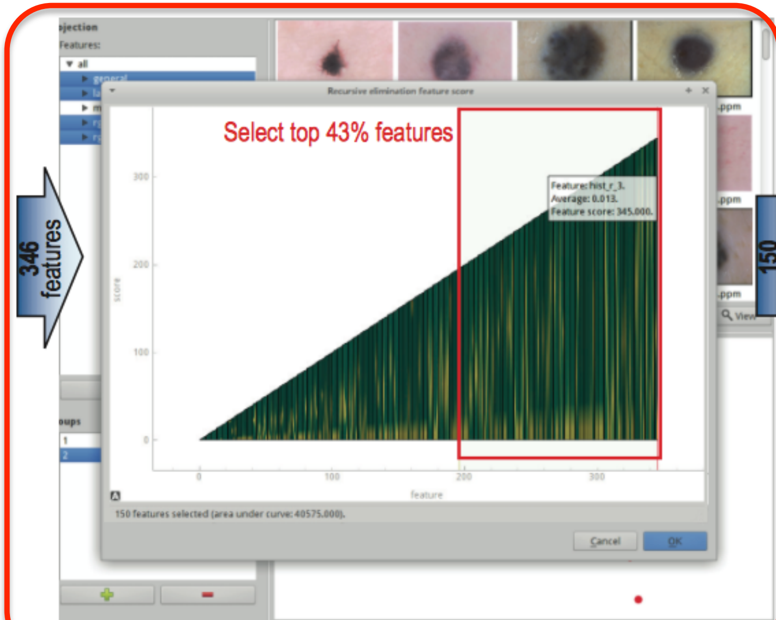
Visual tool design

- linked views showing
  - all images (acquired with dermatoscopes)
  - all features (extracted from images)
  - selected features for classifier construction
  - feature-vector similarities (using 2D multidimensional projection)
  - feature relevance (scoring) for image similarity

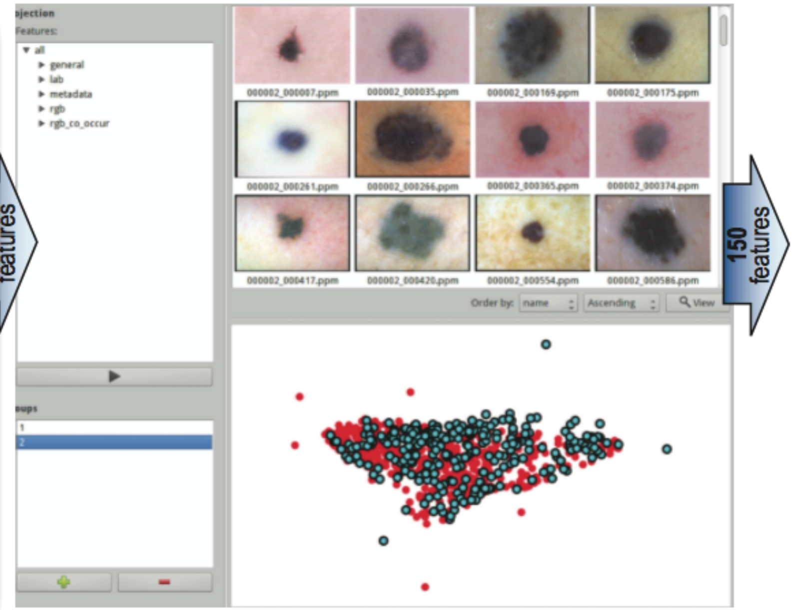


[Rauber et al'15]

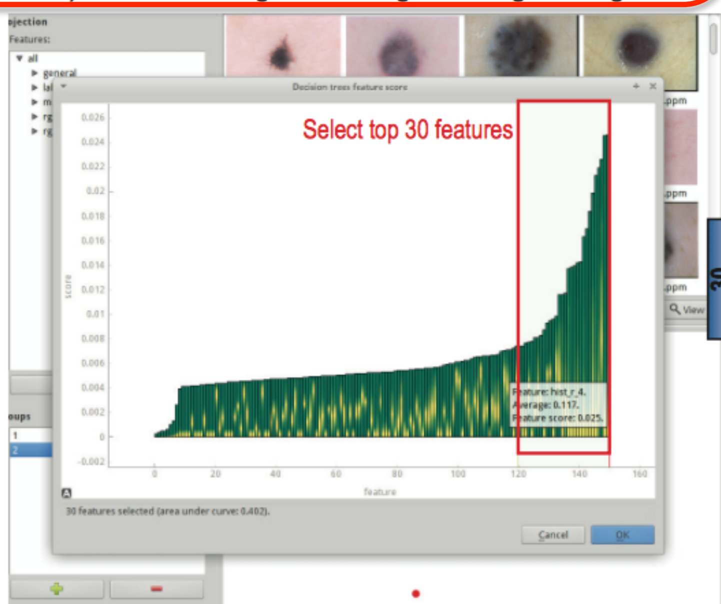
Way of working (1/7): Start with 346 features…



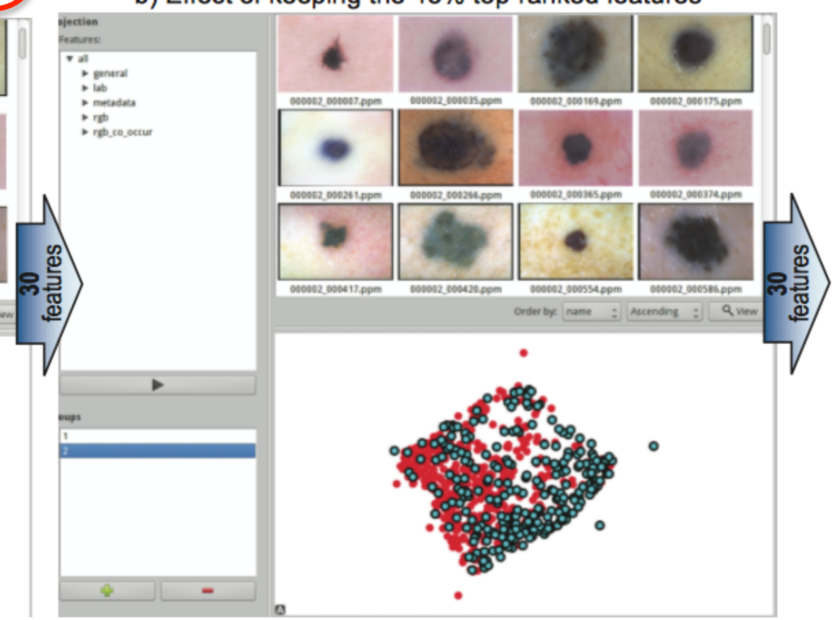
a) Feature ranking on all malignant images using RFE



b) Effect of keeping the 43% top-ranked features

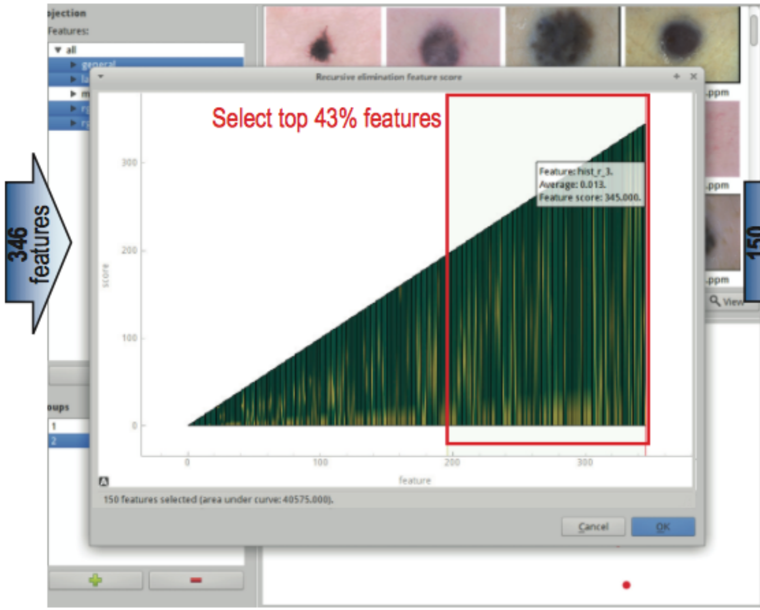


c) Feature ranking using decision trees (DT)

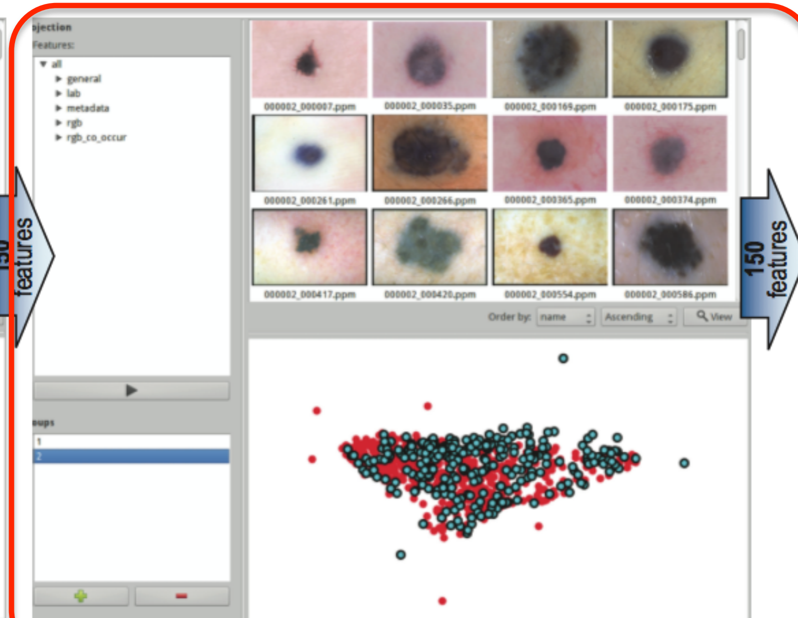


d) Effect of keeping the 30 top-ranked features

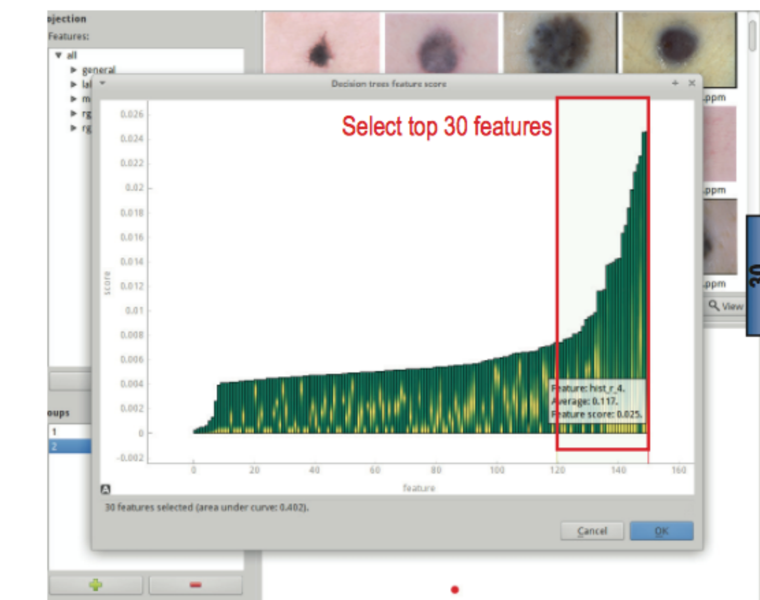
Way of working (2/7): Reduce to 150 features…



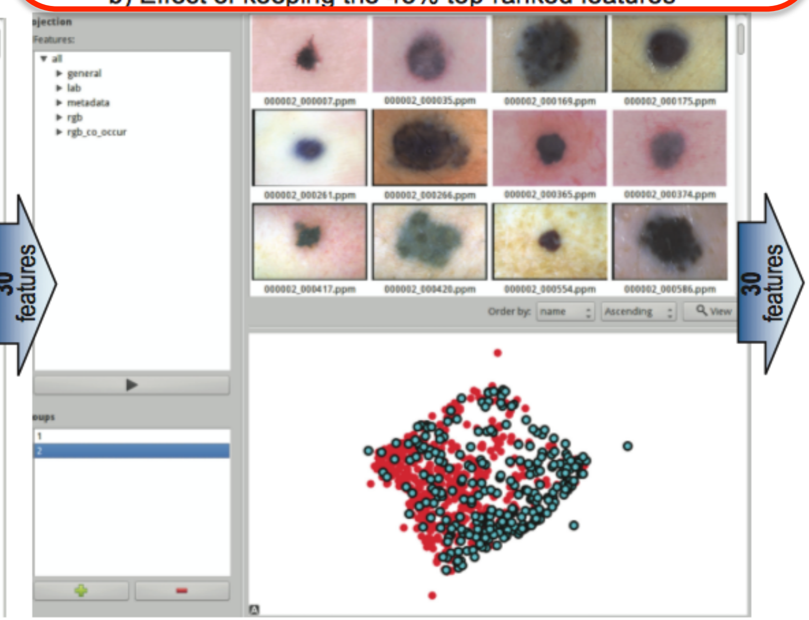
a) Feature ranking on all malignant images using RFE



b) Effect of keeping the 43% top-ranked features

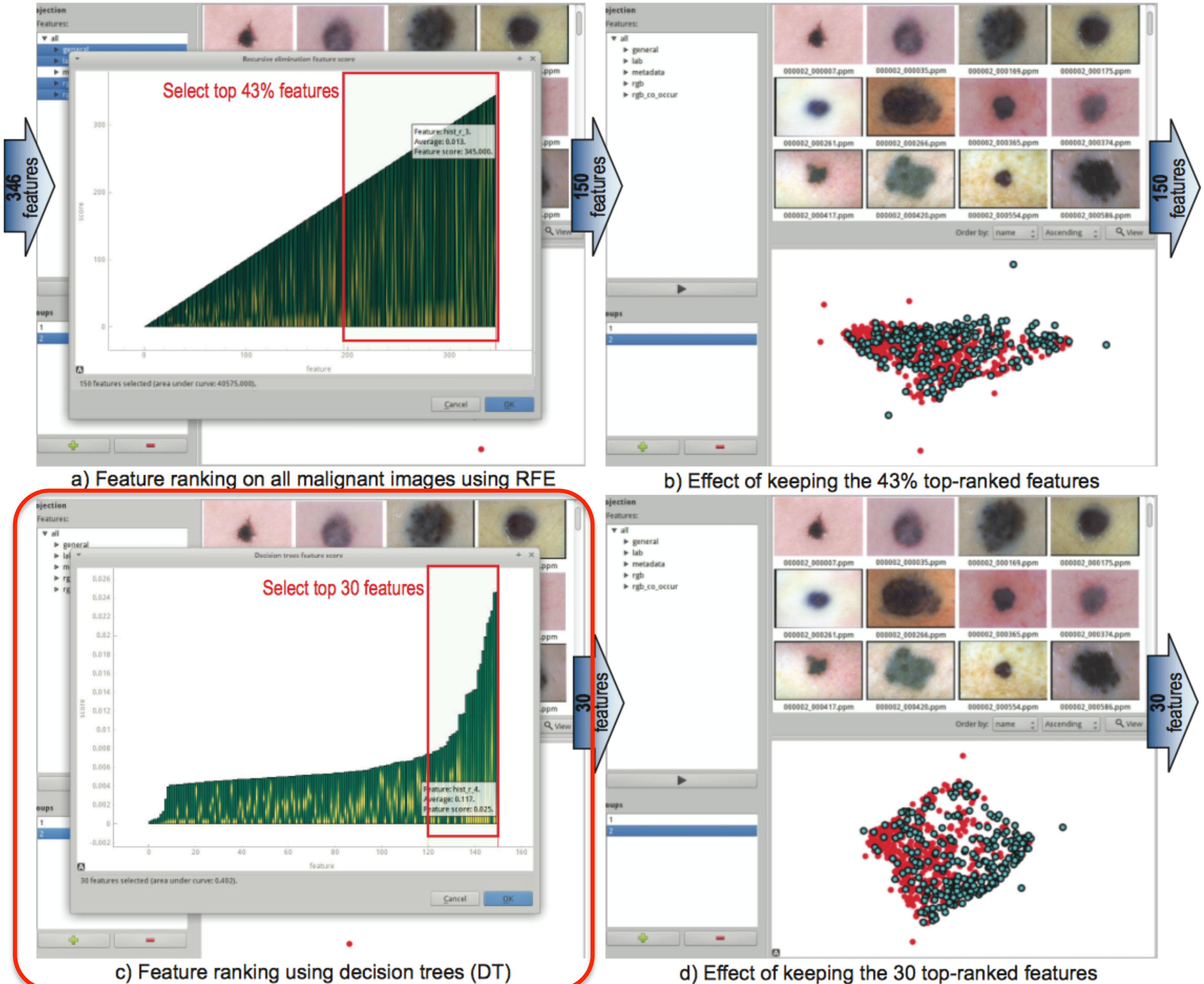


c) Feature ranking using decision trees (DT)



d) Effect of keeping the 30 top-ranked features

Way of working (3/7): Select most relevant 30 features…



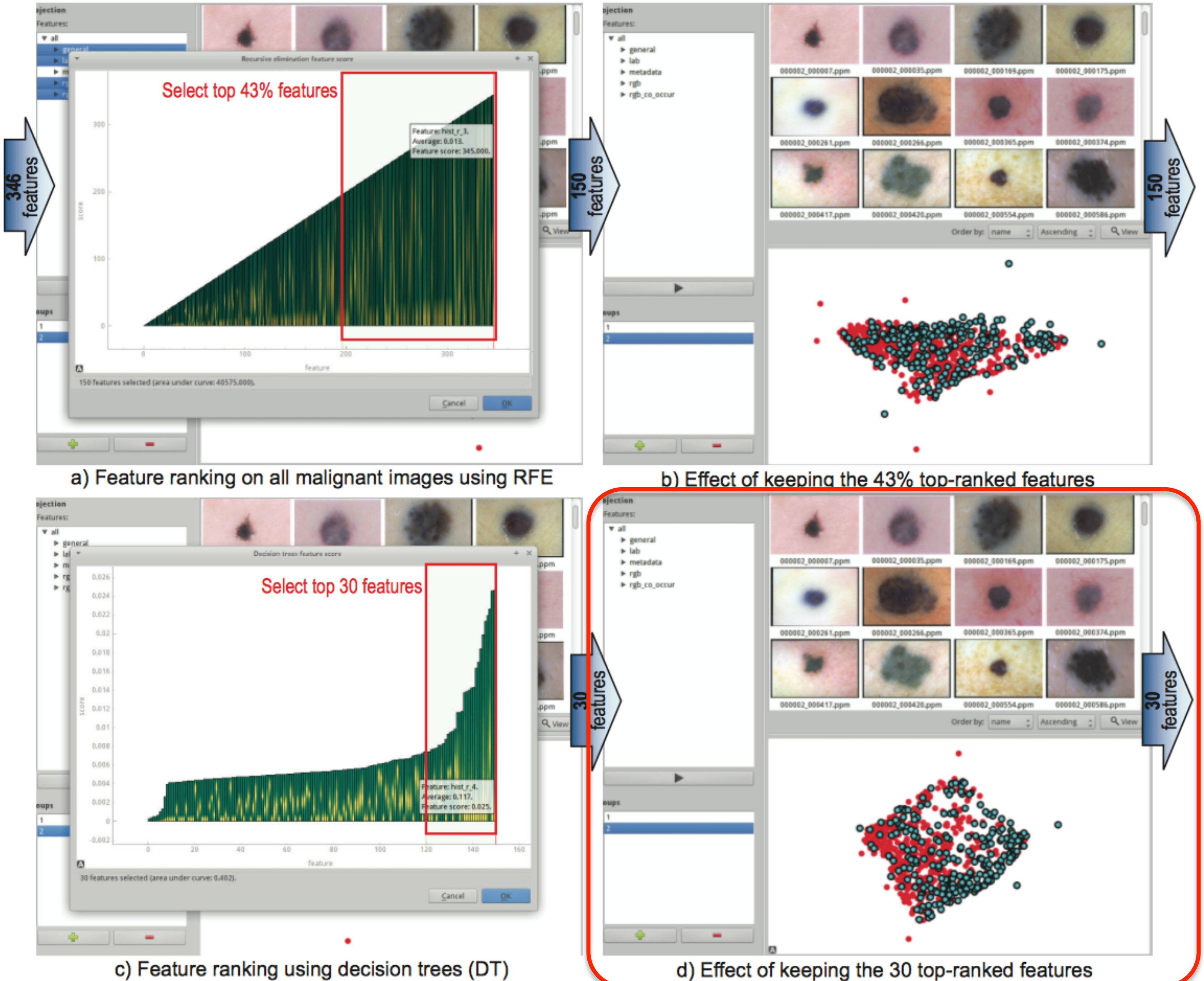
a) Feature ranking on all malignant images using RFE

b) Effect of keeping the 43% top-ranked features

c) Feature ranking using decision trees (DT)

d) Effect of keeping the 30 top-ranked features

Way of working (4/7): Reduce to 30 features...



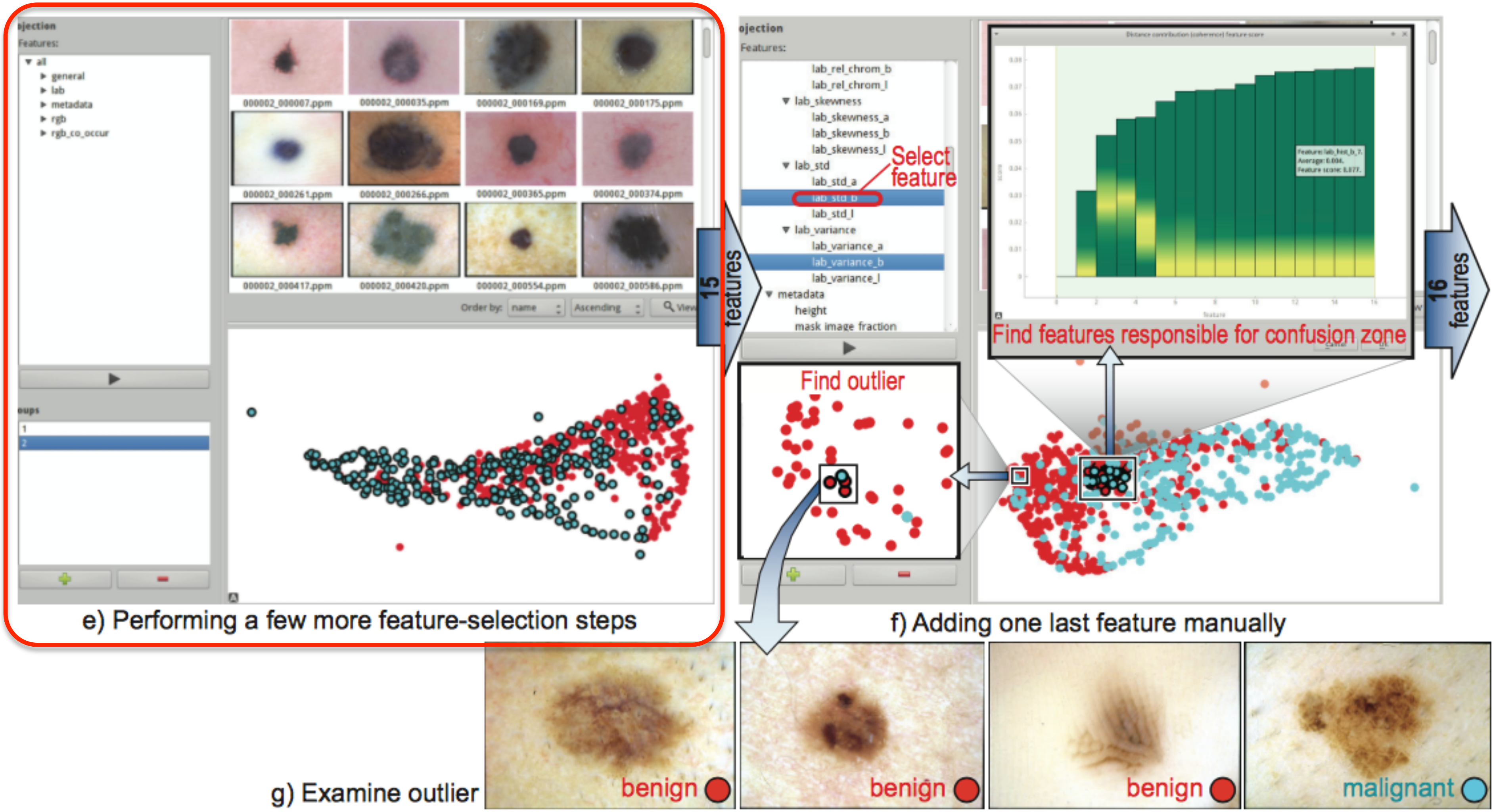
a) Feature ranking on all malignant images using RFE

b) Effect of keeping the 43% top-ranked features

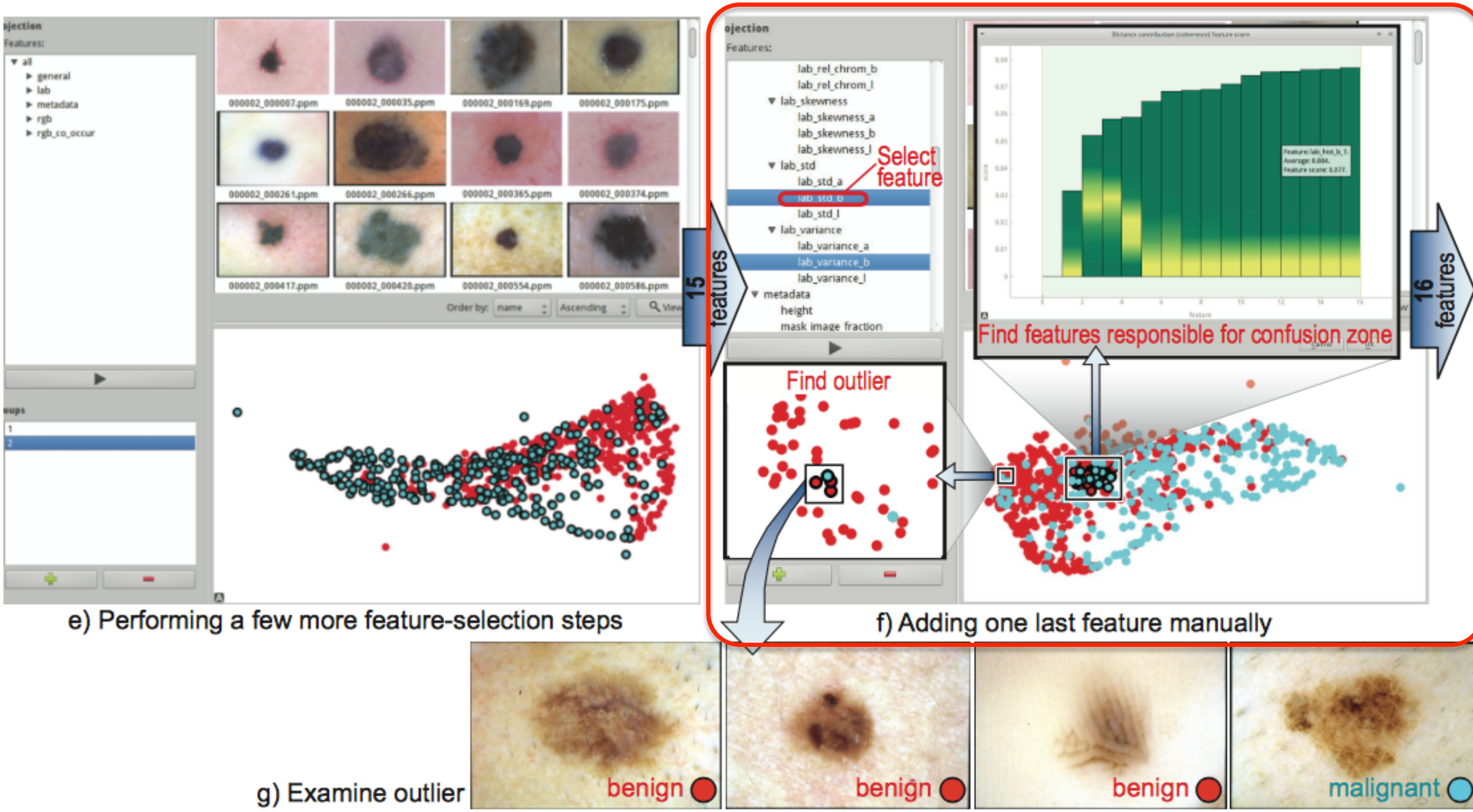
c) Feature ranking using decision trees (DT)

d) Effect of keeping the 30 top-ranked features

Way of working (5/7): Reduce to 15 features…



Way of working (6/7): Solve confusions by adding 1 feature…

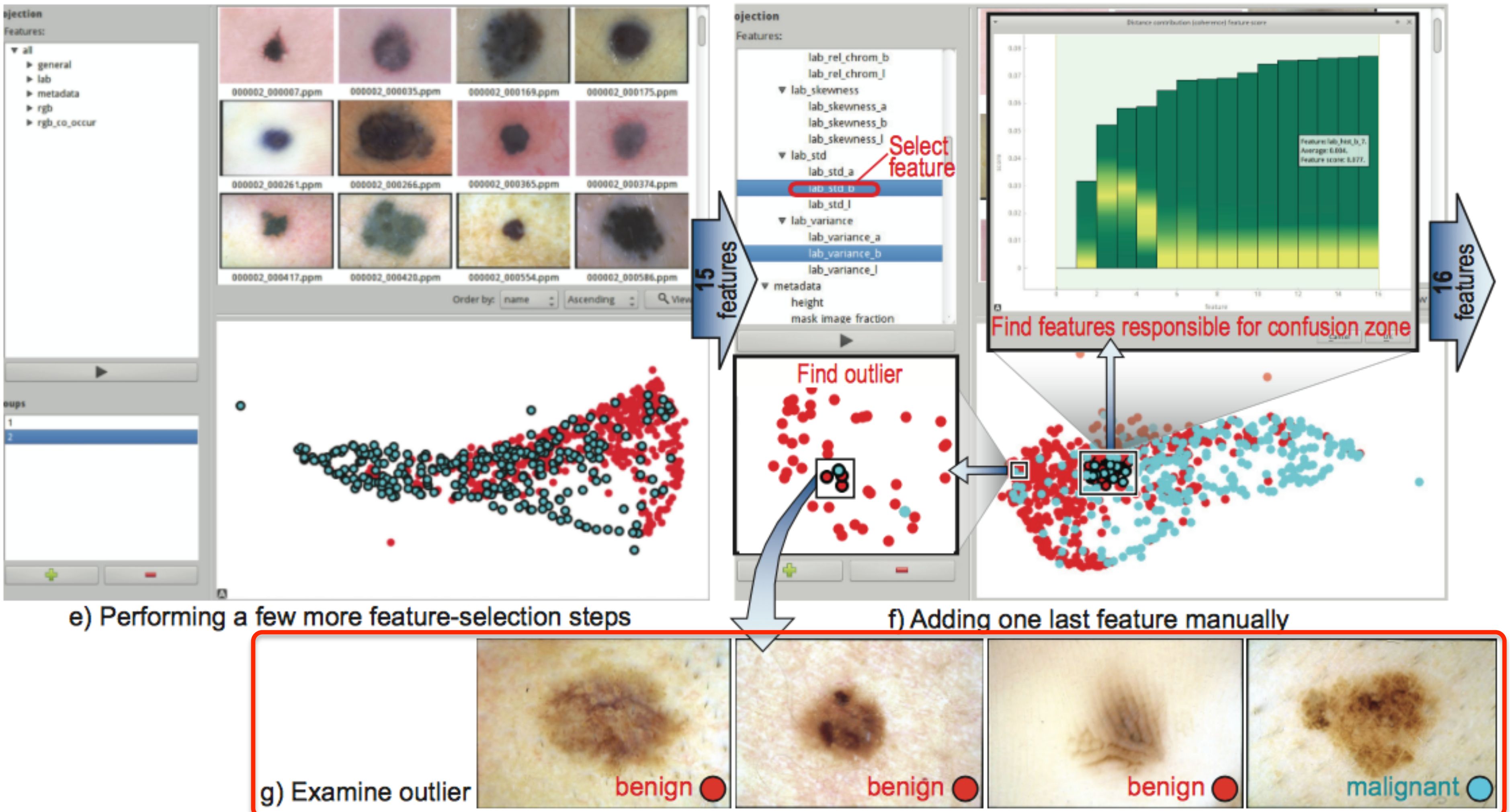


e) Performing a few more feature-selection steps

f) Adding one last feature manually

g) Examine outlier

Way of working (7/7): Explain remaining confusion zones



Results

- reduced 346 features to **16**, keeping good classification **accuracy** (~75%)
- found which images are wrongly classified, got insights in what **new features** we need
- our tool: classification accuracy **82%**, better than state-of-the-art commercial tools

3. Projections for improving classifier-construction

Problem

- say we want to construct a classifier for some problem/data
- typical way of working
  - select a classifier technique
  - find an implementation
  - fine-tune implementation
  - run/test implementation
  - assess accuracy
  - repeat from step 3 until satisfaction
- this is **very costly!**

Proposal

- shorten cycle by assessing
  - discriminative power of computed features
  - types of problems they will induce

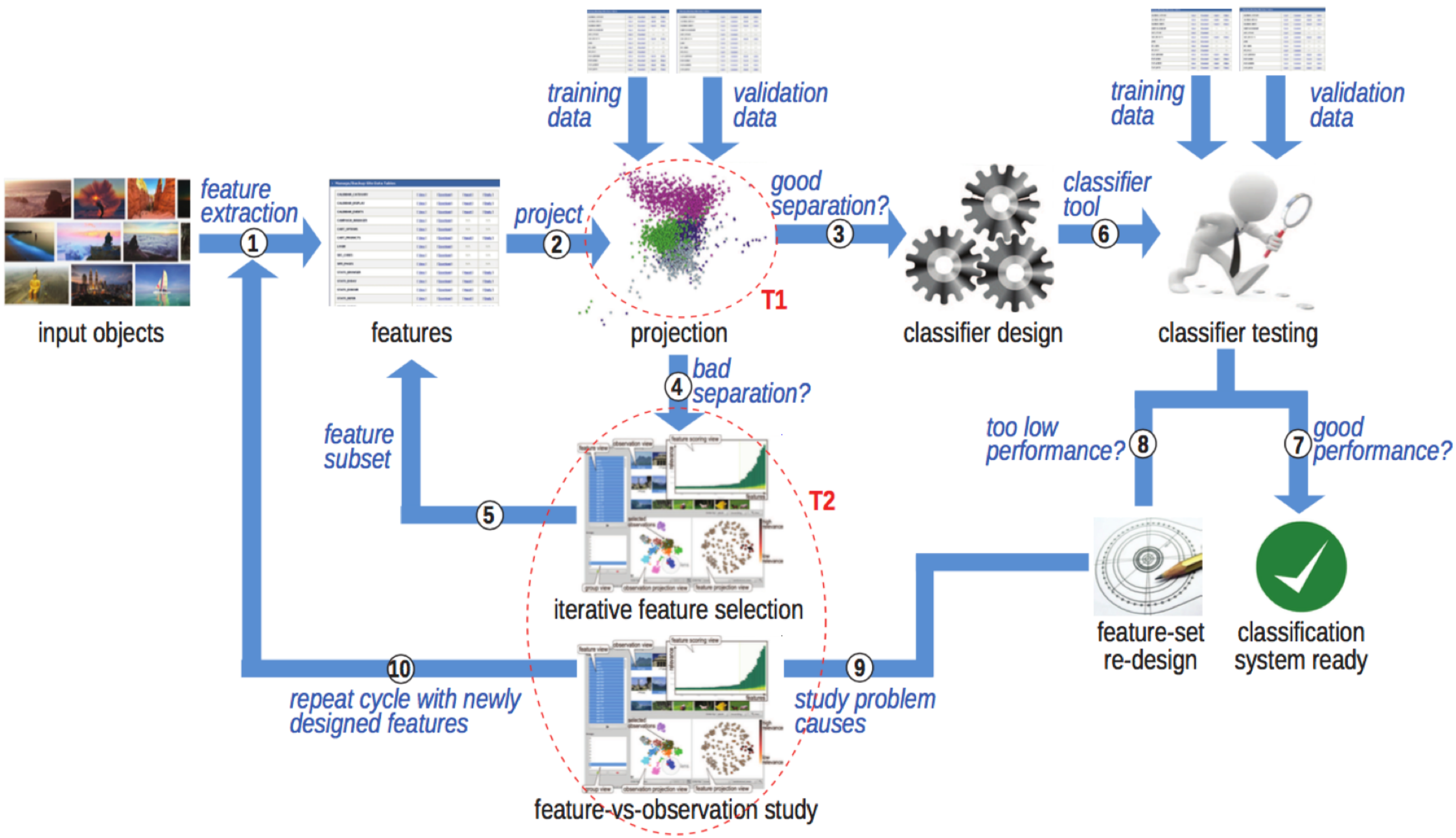
  before selecting/building/testing classifier!

Advantages

- get feedback on problem complexity and feature quality **early on and cheaply**
- improve input of classifier is **easier** than improving a classifier itself

Projections: Central tools in our solution

Workflow

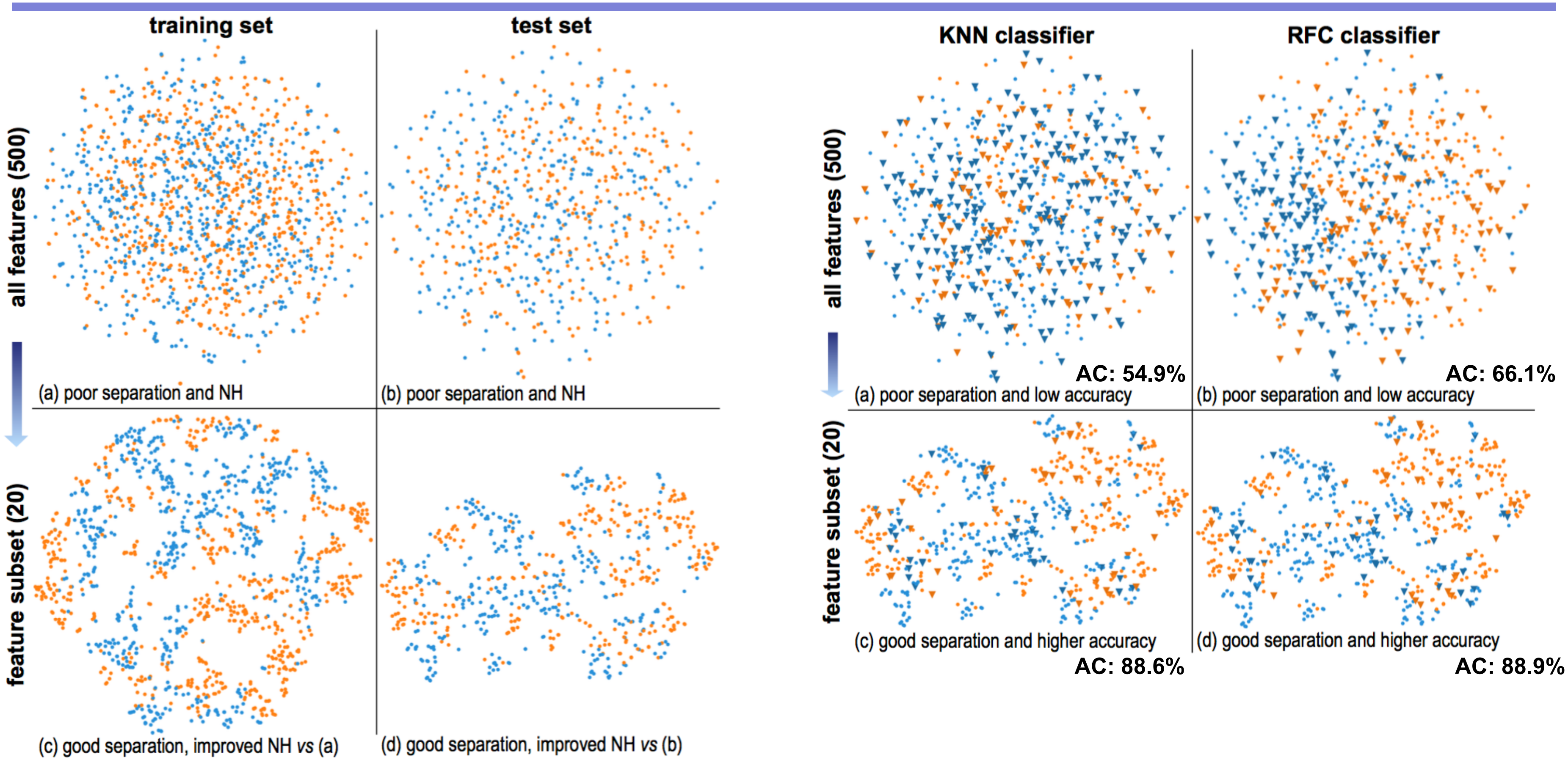


T1: Predict classification efficacy
T2: Improve classification efficacy

T1: Predict classification efficacy

Extensive set of experiments proved that separation in a (good) projection *predicts* accuracy of a (good) classifier

Top row: Hard classification problem → **Validation: many misclassifications**



Bottom row: Easy classification (after feature selection) → **Validation: few misclassifications**

Bottom row: Select 20 of 550 features on their discriminative power on training set, using extremely randomized trees

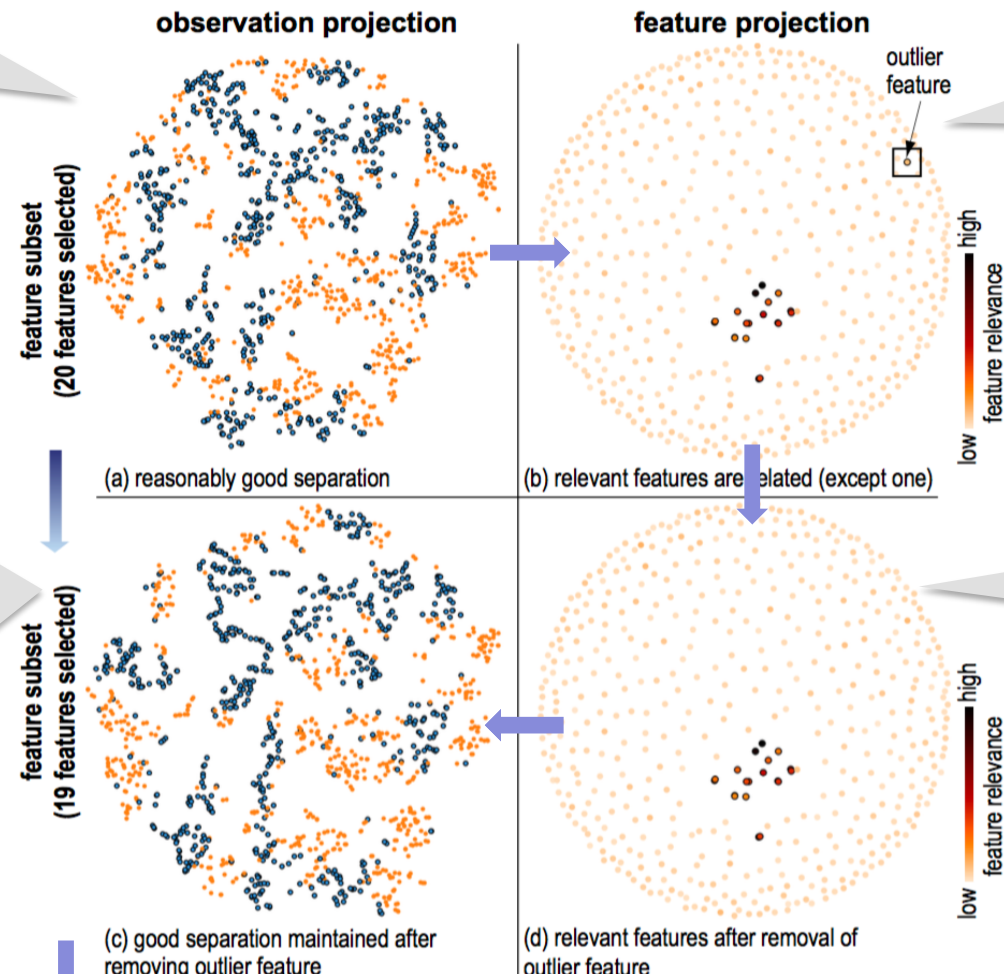
Dataset: Madelon (200 points, 500 dims, 2 classes)

T2: Improve classification efficacy

Visually analyze and reason about observations and features to improve classifier efficacy

1. Start with this good projection (20 features)…

2. Examine similarity and discriminative power of features. We find an outlier feature…



3. Remove outlier feature, since it is unrelated to the other discriminative ones

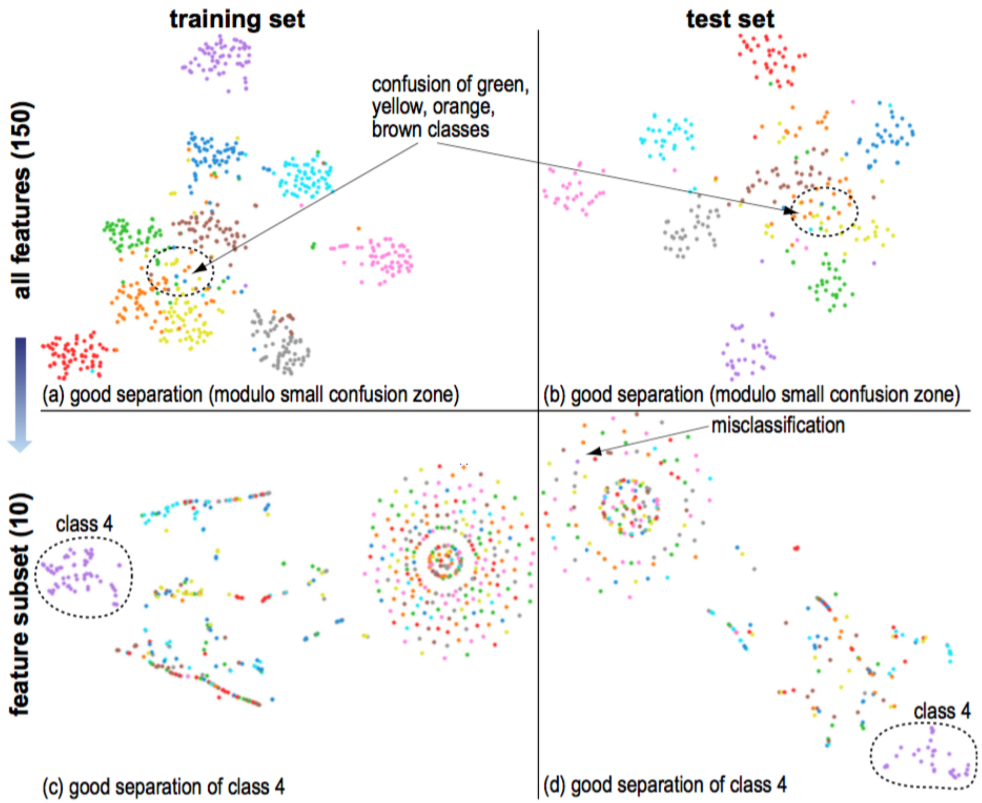
4. Resulting projection still shows high separation. Also, classification accuracy is now higher (see table)

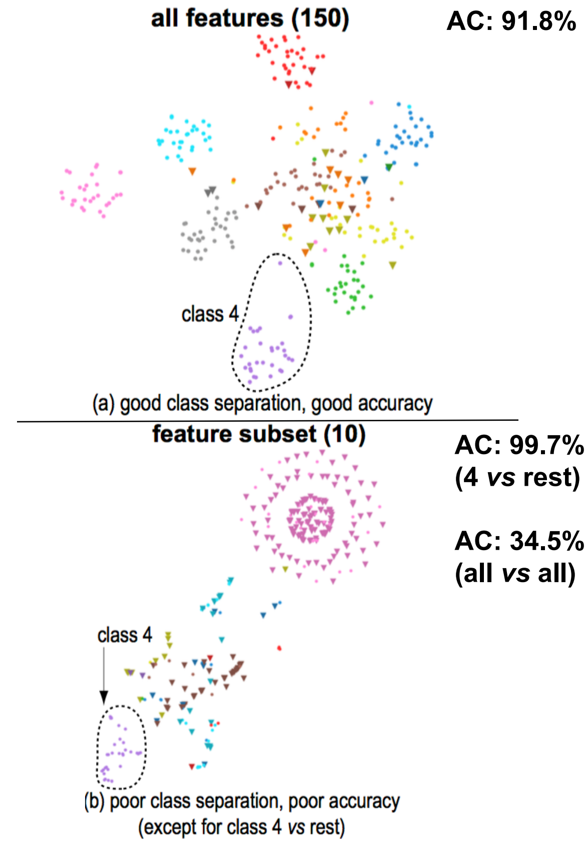
Design a (slightly) better classifier

Features/Algorithm	KNN	RFC	SVM
20 features	88.62%	88.92%	86.68%
19 features	88.92%	88.92%	89.22%

T1: Predict classification efficacy

Top row: 10-class separation is easy → **Validation: good classification results**





Bottom row: Separating class 4 from rest is easy (using only 10 features) → **Validation: good classification results**

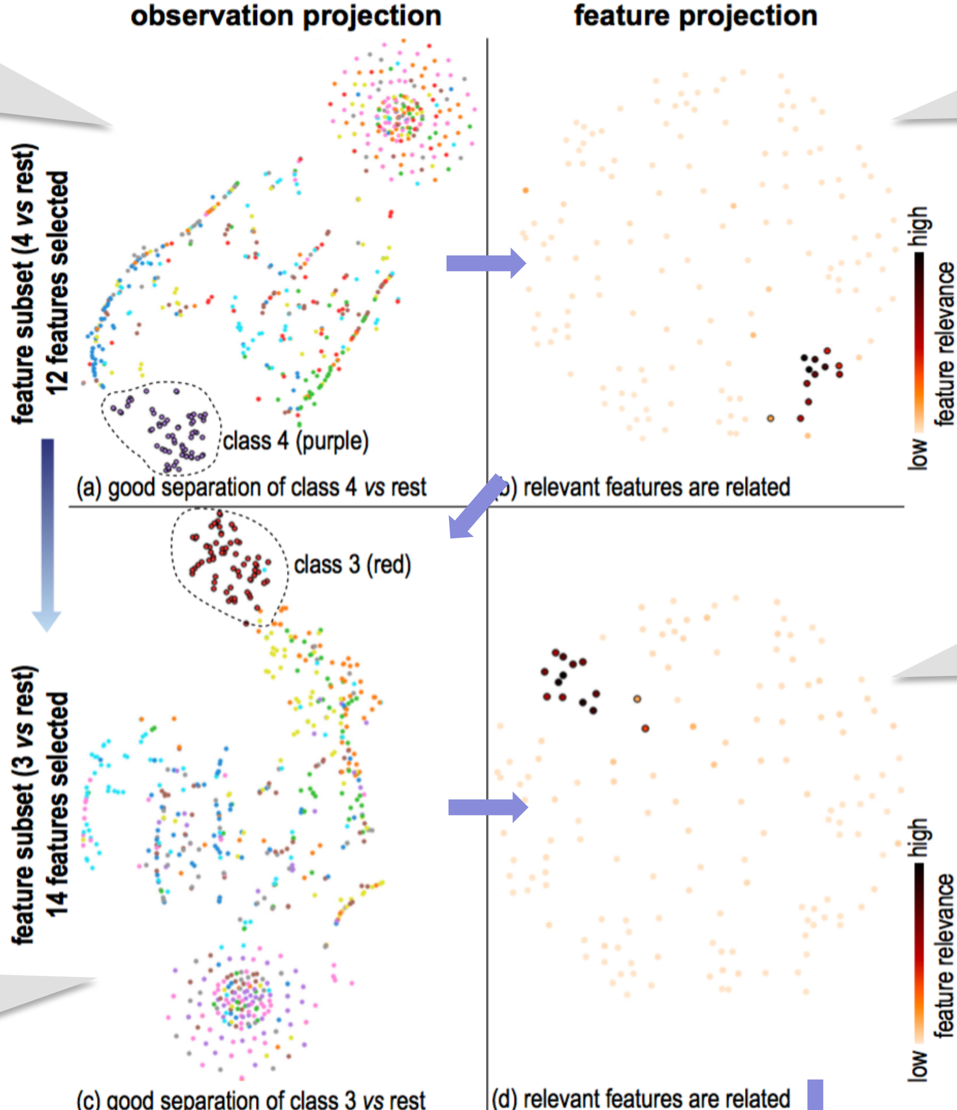
- projections help designing very high-quality more specific classifiers
- confusion zones indicate type and extent of classification problems

Dataset: Corel (1000 points, 150 SIFT features, 10 classes)

T2: Improve classifier

1. Start with this good projection (12 features) designed to separate class 4 *vs* rest…

2. We find that features discriminating class 4 are highly related *and* different from the rest



3. Now do the same for another class (3), which gives us 14 features…

4. We find that features discriminating class 3 are highly related *and* different from features discriminating class 4

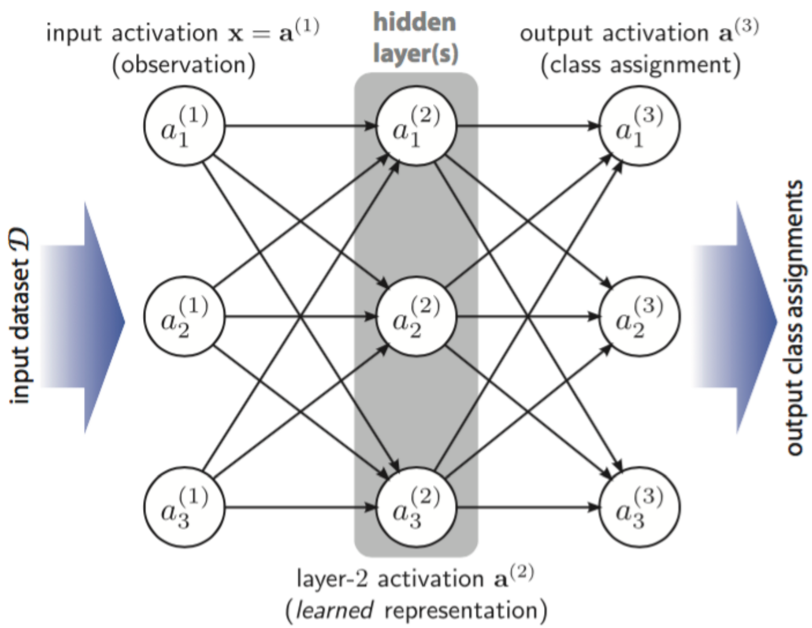
Features/Algorithm	KNN	RFC	SVM
All (150) features	98.18%	98.79%	98.48%
26 features	98.48%	98.79%	98.79%

Design a (slightly) better classifier with far less features

4. Projections for understanding deep neural networks

Artificial Neural Networks (ANNs)

- increasingly popular for classification, pattern recognition
- good results in cases where other methods are suboptimal (e.g. feature selection)
- different types (multilayer perceptrons (MLPs), convolutional neural networks (CNNs))



Problems

- way of working of an ANN is a true 'black box'
- when results are not optimal, how to
  - understand **what** has gone wrong, and **where**?
  - **improve** the classifier?

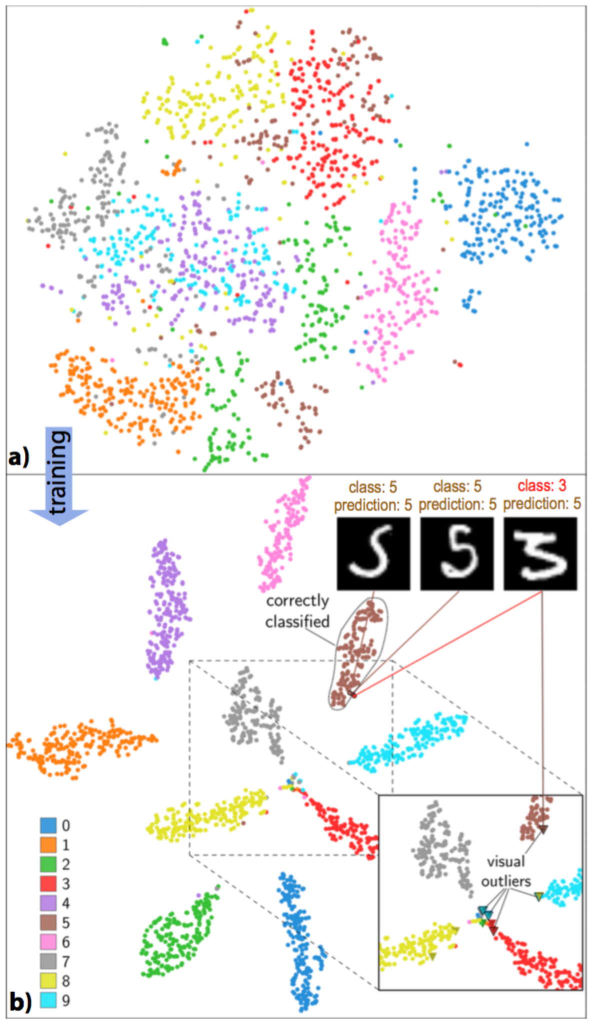
T1: Explore learned representations (activations)

Method

- project input observations (images) having all activations in a layer as dimensions
- we see how the learned info is created by **training** and the **layer** structure

MNIST dataset / MLP

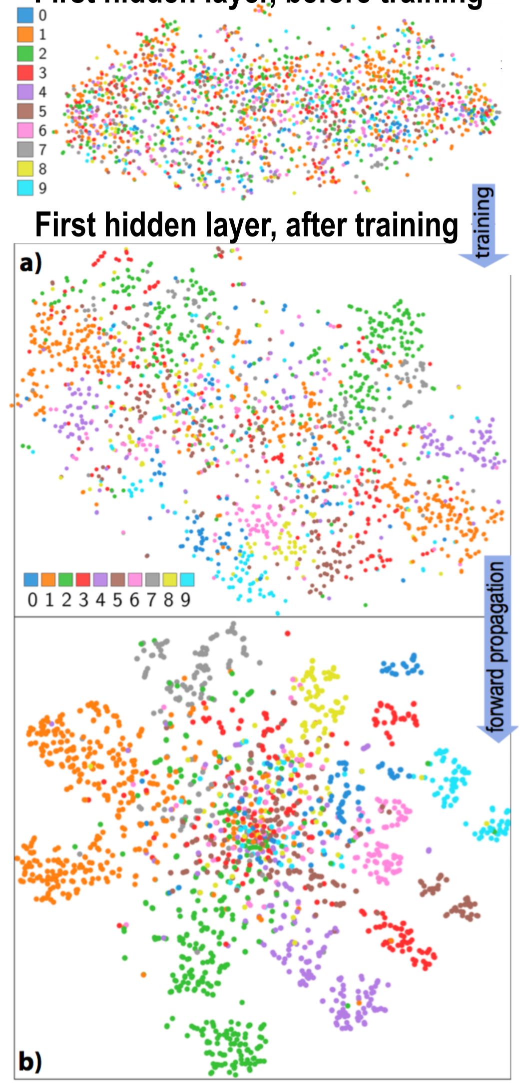
Last hidden-layer activations, before training



Last hidden-layer activations, after training

SVHN dataset / MLP

First hidden layer, before training



Last hidden layer, after training

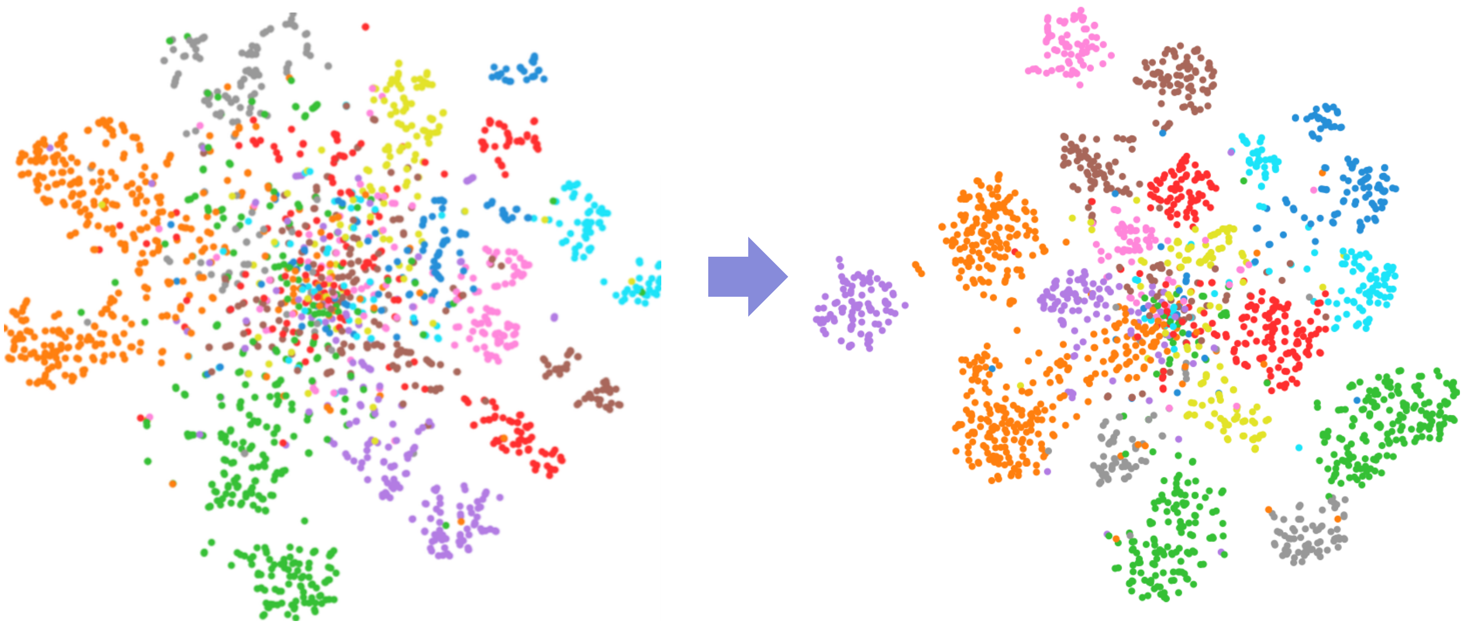
T2: Explore learned representations to improve classification

First step

- try another network (CNN instead of MLP)

We next notice something strange in this image. Can you see what?

MLP network **CNN network**



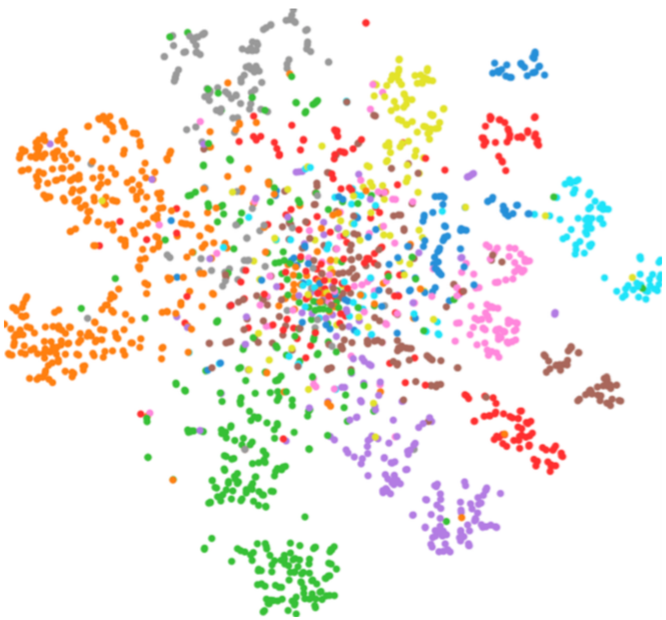
- reasonable visual separation
- AC: 77.3%

- much better visual separation
- AC: 93.8%

T2: Explore learned representations to improve classification

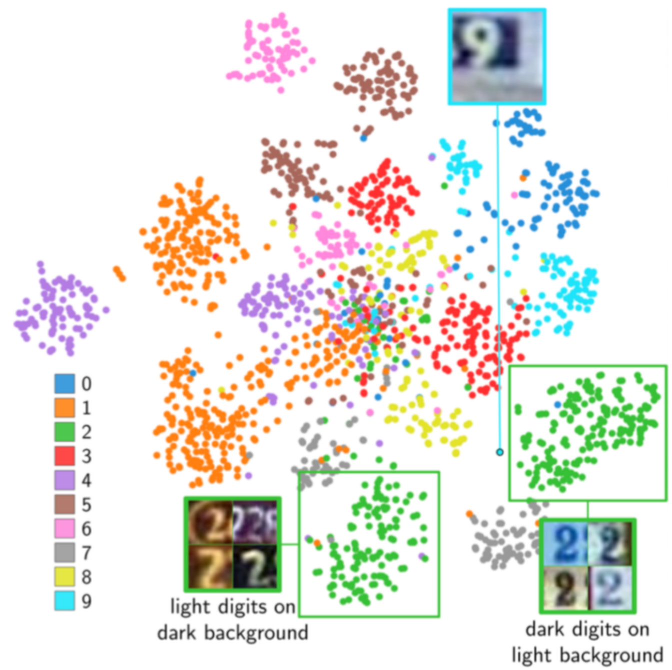
Each class is formed by two balanced but **distinct clusters!**

MLP network



- reasonable visual separation
- AC: 77.3%

CNN network



- much better visual separation
- AC: 93.8%

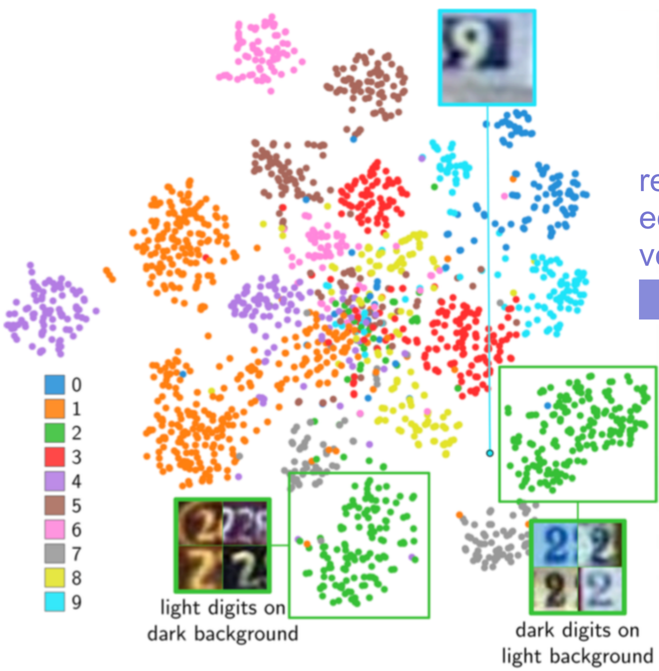
T2: Explore learned representations to improve classification

What is going on?

- visually explore clusters by brushing
- we find that each cluster-pair contains
  - a cluster for light images on **dark** background
  - a cluster for **dark** images on light background

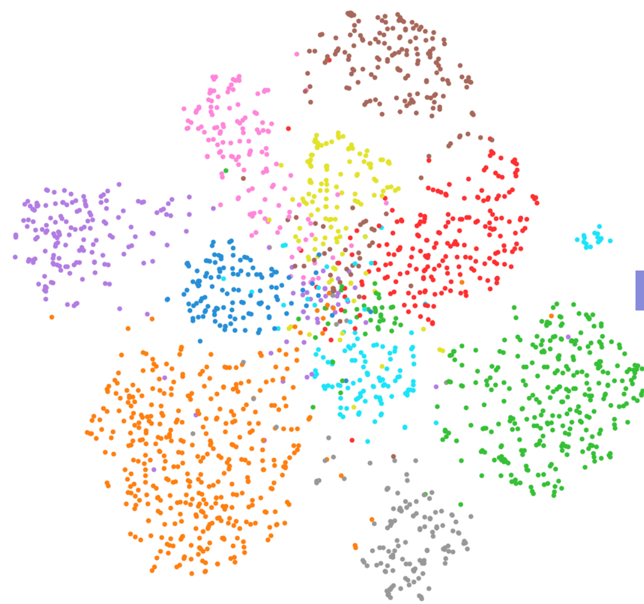
Let’s use this insight to improve the classification:

Current situation



replace images by edge-detection versions

After preprocessing



Accuracy increases

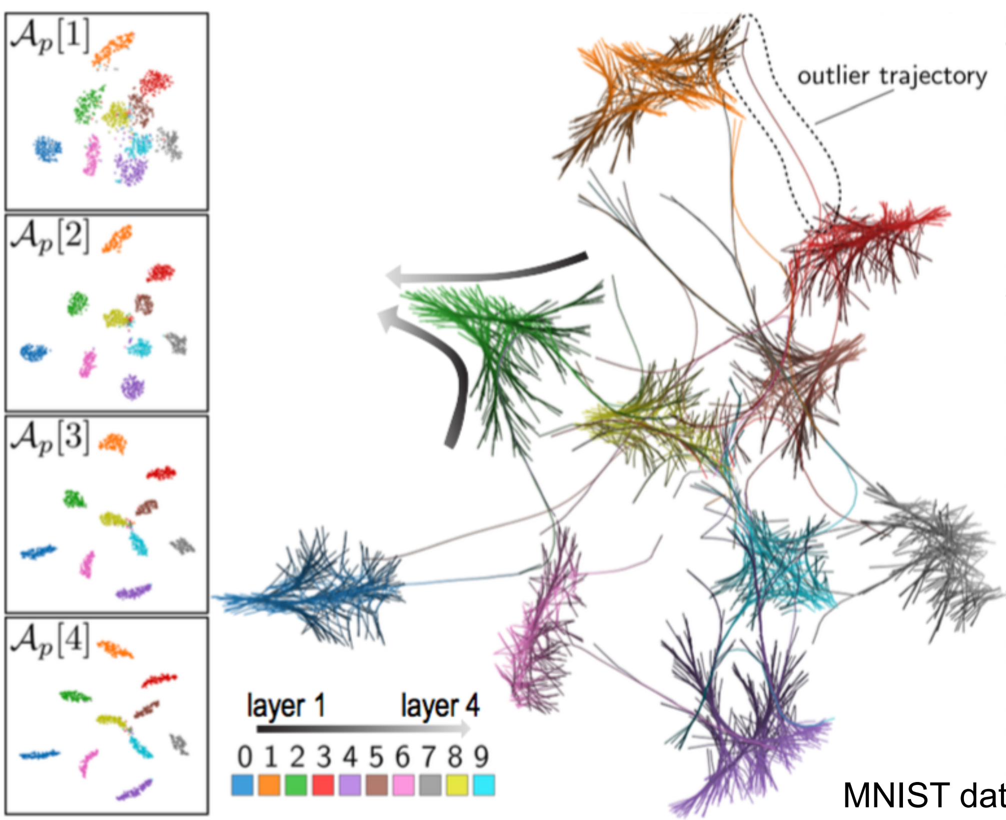
MLP: +4%
CNN: +0.7%

T2: Explore evolution of learned representations

Context

- activation in an ANN change in time in two ways
  - as data flows from the 1st to the last network, during operation (**inter-layer** evolution)
  - as different datasets are used, during training (**inter-epoch** evolution)
- we want to explore both so as to
  - understand how different layers contribute to learning
  - understand if training is effective

Inter-layer evolution



MNIST dataset, MLP classifier

Bundled observation paths
(built using our dynamic t-SNE)

We observe how

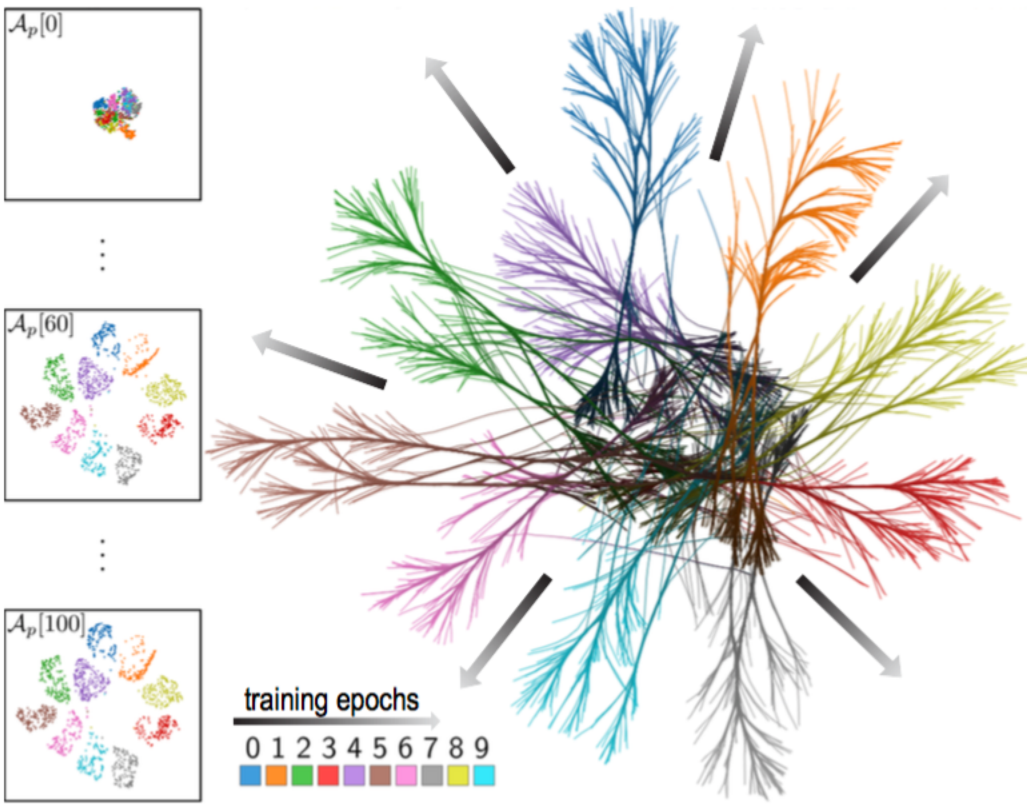
- group separation increases
- group size decreases
- groups increasingly diverge
- few trails connect different groups (classification decisions are stable)

Conclusions

Network performs (very) well in practice!

T2: Explore evolution of learned representations

Inter-epoch evolution



MNIST dataset, last CNN hidden layer, 100 training epochs

Bundled observation paths

We observe how

- group separation increases (from complete clutter to perfect separation)
- groups increasingly diverge
- paths are quite straight/smooth (no canceling of learning)
- paths don't link different-color groups

Conclusions

- Learning is very effective
- Knowledge accumulates as desired
- Few/no 'hesitations' during learning

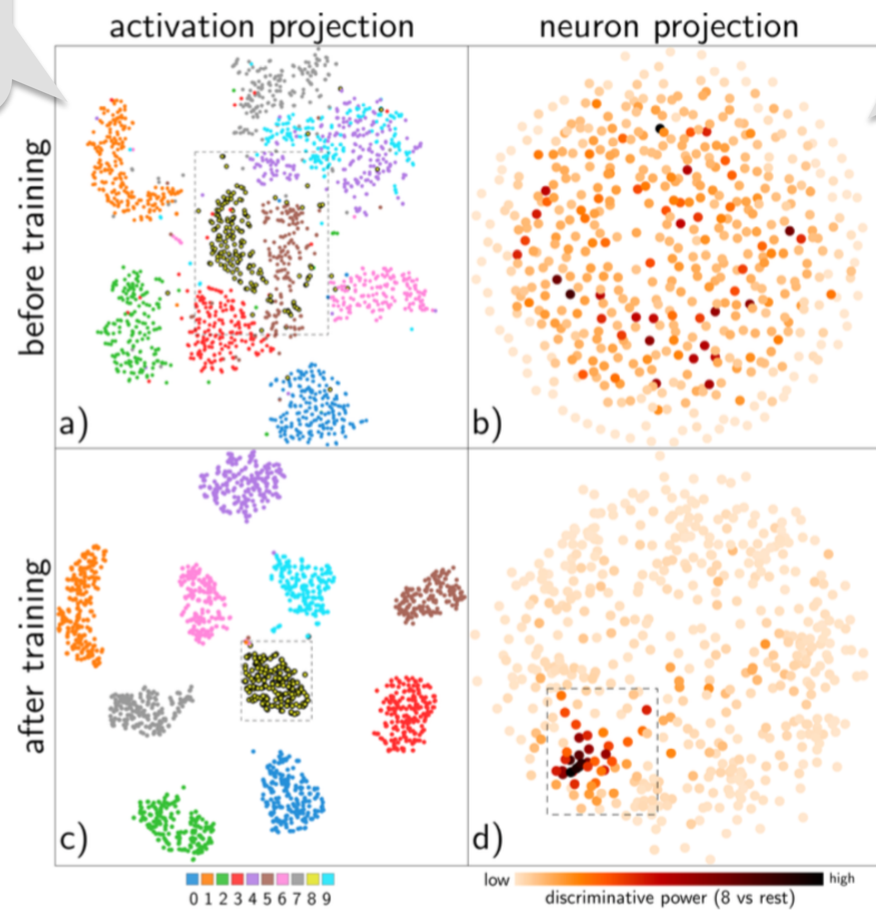
T3: Explore neuron specializations

Context

- choosing an ANN architecture is (often) a kind of black magic
- help this by explaining roles of neurons (in a layer)
- use **two projections** (one for activations, one for neurons)

Activation projection shows similarities of observations (via activations in a layer)

Neuron projection shows similarities of neuron activations (in a layer) for all observations



Training improves *separation* of observations into groups

Training increases neuron *specialization* for the different classes

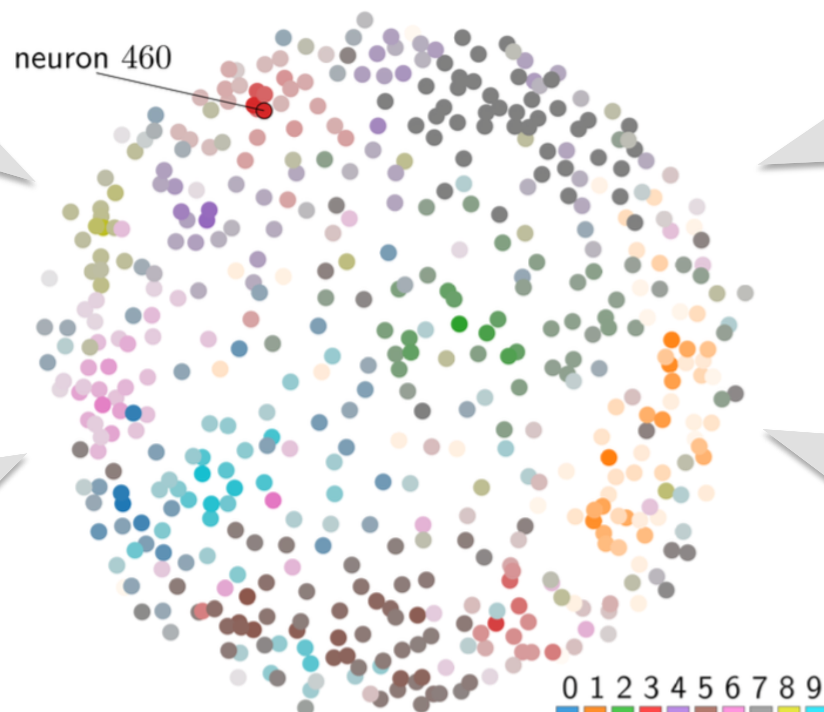
T3: Explore neuron specializations

Discriminative neuron map

- summarizes role and power of all neurons in a layer for a task
  - position: similarity of correlations of neuron activations
  - color: most important class the neuron is responsible for
  - saturation: how important the neuron is for that class *vs* other classes

Balanced #neurons/ color, so a balanced training set was used (which is good)

Uniform spread of neurons over drawing, so small differences in activation correlation



Hues are relatively well clustered, so similar neurons work collaboratively

Few saturated colors, so many neurons have unclear roles in the classifier (may be bad)

SVHN dataset, CNN classifier, last hidden layer after training

Conclusions

Classifier design

- the main (and toughest) challenge in machine learning

- we open the black-box of ‘design magic’ by visual analytics
  - extend multivariate projections to be useful and usable in practice
  - use these for classifier prediction, understanding, and improvement
    - interactive feature scoring/selection
    - predict classification accuracy from projection separability
    - prune feature space to reduce computation cost
    - explain the training and working of deep neural networks

Lots of applications are now possible!

Thank you for the interest!

a.c.telea@rug.nl