GeoX Young Academy: Machine Learning in Remote Sensing Best practice and recent developments - Part 3: ML and Images

Ronny Hänsch & Andreas Ley
Dept. Computer Vision and Remote Sensing
TU Berlin - Germany

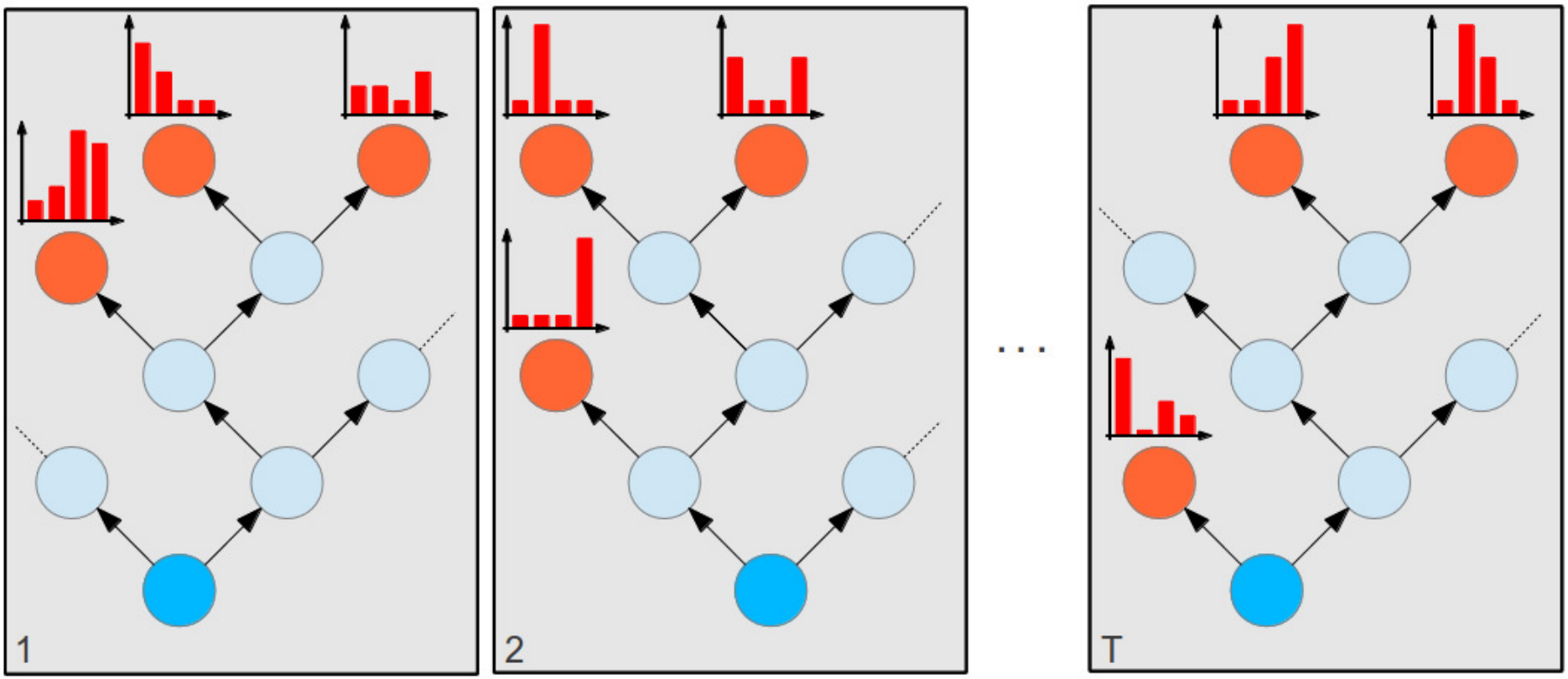
Overview

1. Random Forests
   - Fusion
   - Node Tests
   - Interpretation
   - Application Tips
2. Graphical Models
   - Motivation
   - MRF
   - Application
   - Example
3. ConvNets
   - MLP to ConvNet
   - Convolution
   - Architectures
   - Auto Encoder
   - Frameworks

Overview

1. Random Forests
   - Fusion
   - Node Tests
   - Interpretation
   - Application Tips
2. Graphical Models
   - Motivation
   - MRF
   - Application
   - Example
3. ConvNets
   - MLP to ConvNet
   - Convolution
   - Architectures
   - Auto Encoder
   - Frameworks

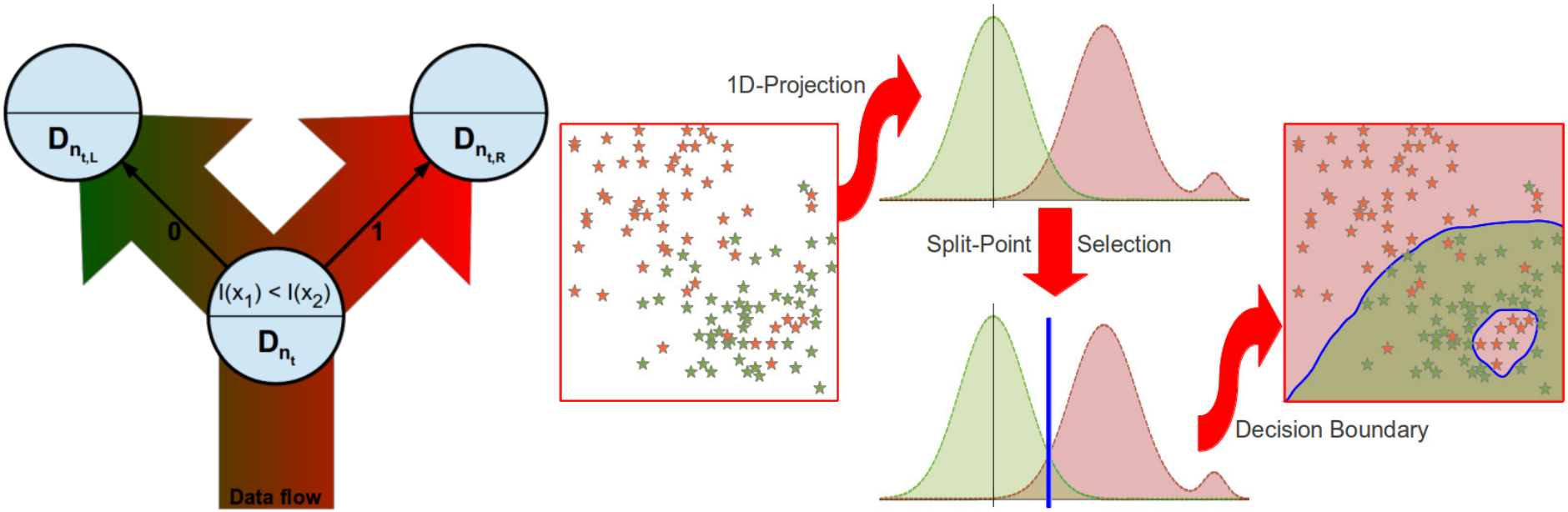
Recap



Set of decision trees

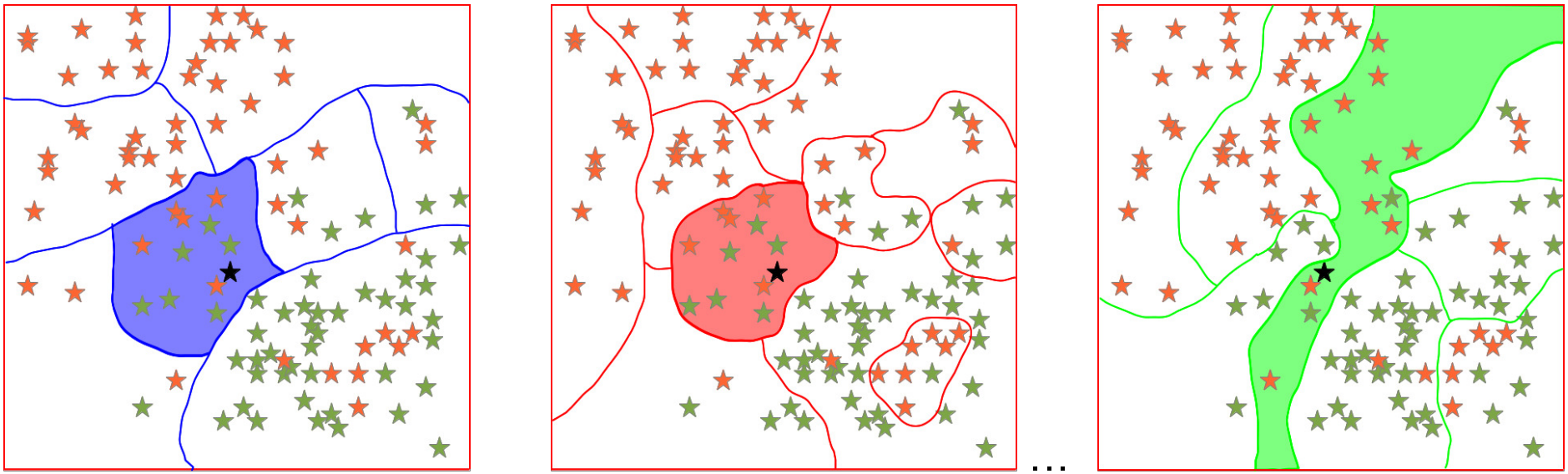
- Each tree t generated from training data $\mathrm{D}_t \subseteq \mathrm{D} \subset \mathbb{D}$
- Creation of one tree independent of all other trees
- Based on random processes to produce diverse set of trees

Data Propagation



- Data enters each tree in root node
- Each non-terminal / internal node performs a (simple) binary test
- Data propagation is based on test outcomes

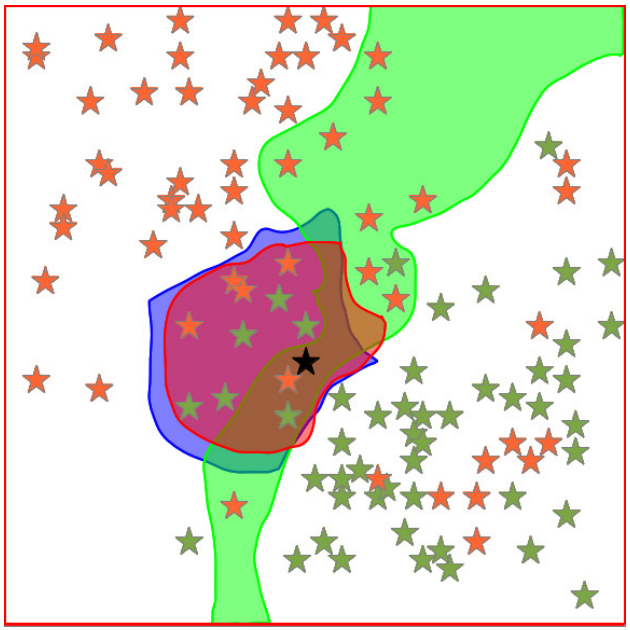
Tree Fusion



$t = 1$ $t = 2$ ... $t = T$

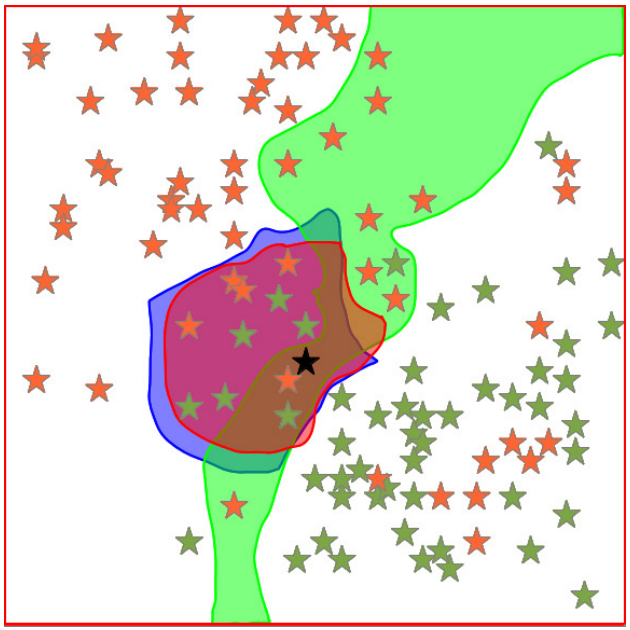
- Query sample is propagated through all trees
- Reaches exactly one leaf in each tree
- Information about target variable assigned to these leafs need to be combined

Tree Fusion



How to combine information assigned to individual leafs?

Tree Fusion

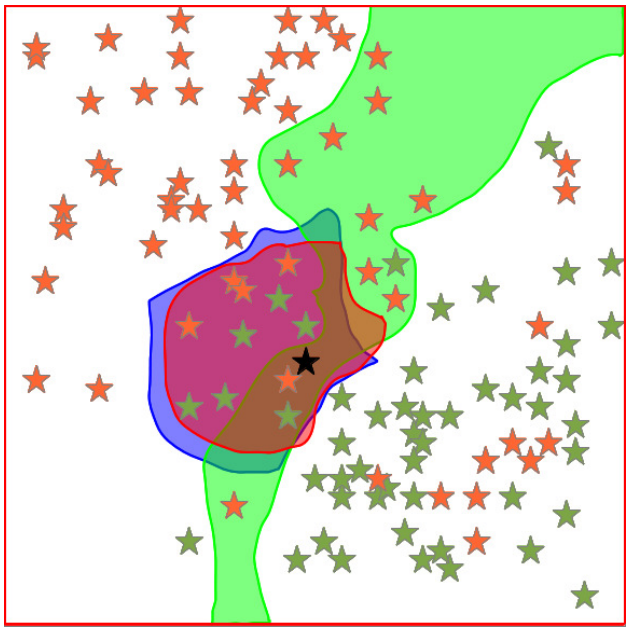


Random Forests

$$P(c|x) = \frac{1}{T} \sum_{t=1}^{T} \delta(M(n_t), c)$$

$M(n)$ is dominant class
of samples in leaf node n

Tree Fusion

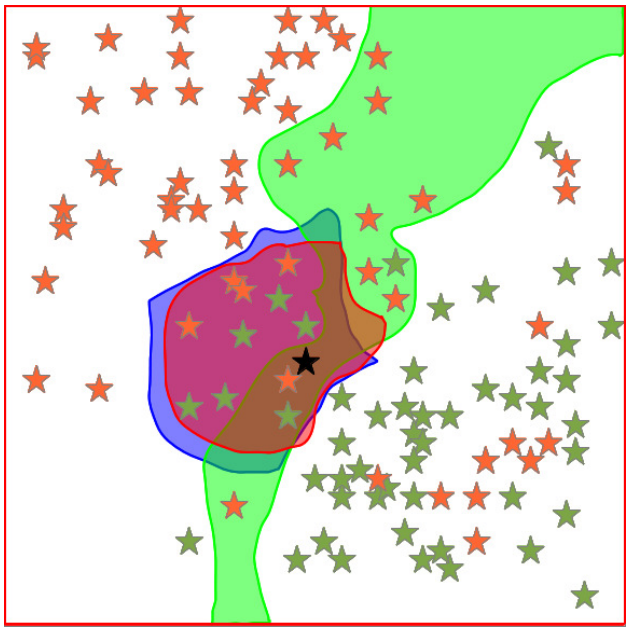


Randomized Trees

$$P(c|x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|x)$$

$P_t(c|x)$ is class posterior in leaf node n_t

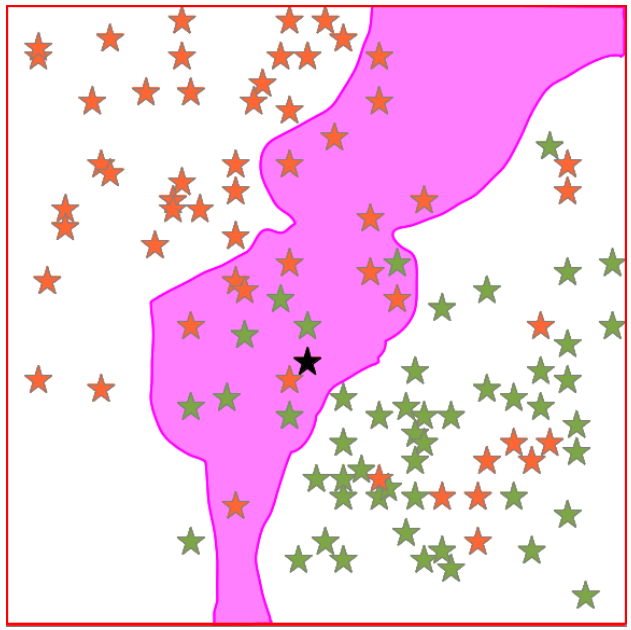
Tree Fusion



Randomized Trees

$$P(c|x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|x)$$

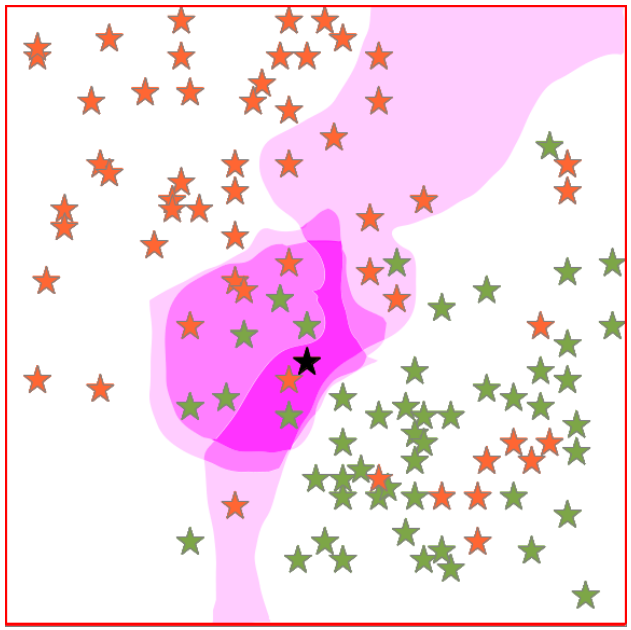
$P_t(c|x)$ is class posterior in leaf node n_t



Fuse before estimation:

$$D_{L_x} = \bigcup_{t=1}^{T} D_{n_t}$$

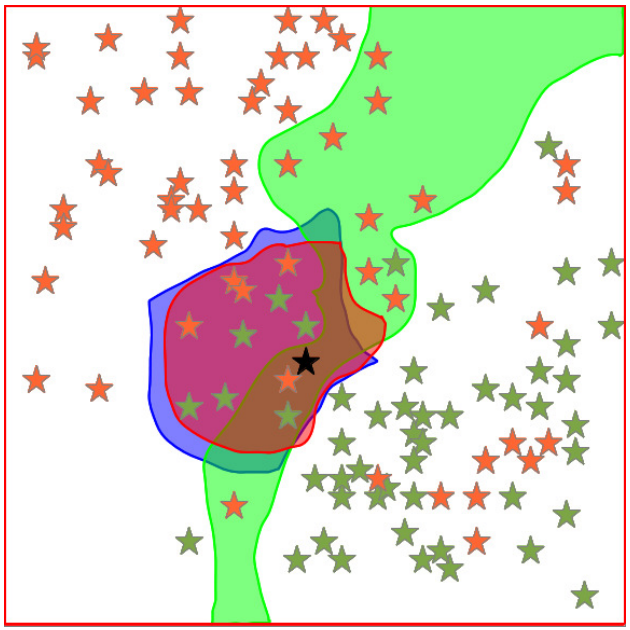
(Multi-set)



$$P(c|x) = \frac{P(L_x|c)P(c)}{P(L_x)}$$

(Details in [Hänsch, 2014])

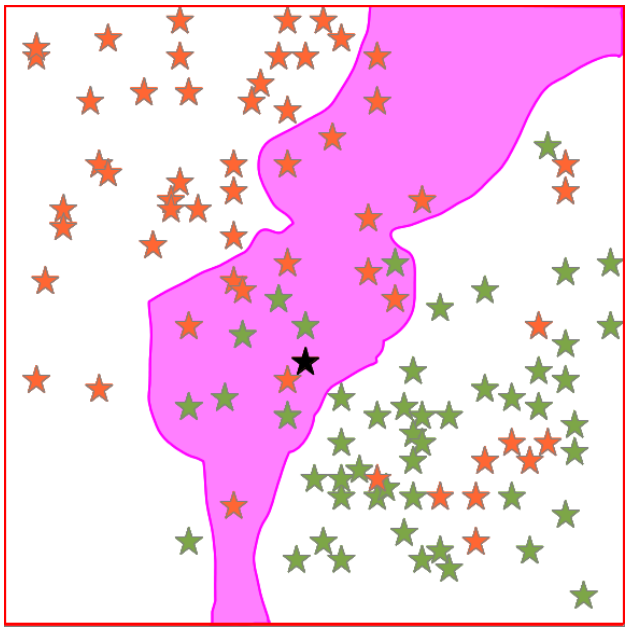
Tree Fusion



Randomized Trees

$$P(c|x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|x)$$

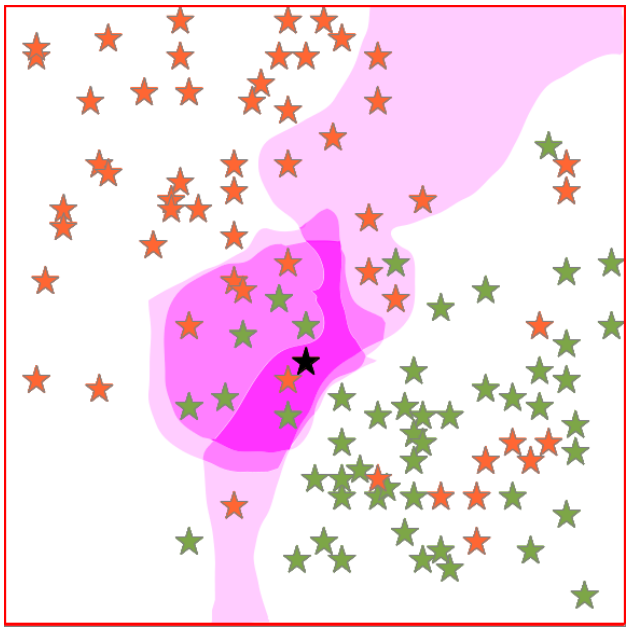
$P_t(c|x)$ is class posterior in leaf node n_t



Fuse before estimation:

$$D_{L_x} = \bigcup_{t=1}^{T} D_{n_t}$$

(Multi-set)



Weighted fusion.

$$P(c|x) = \sum_{t=1}^{T} w_t P_t(c|x)$$

(Details in [Hänsch, 2014])

Node tests

Concatenation of several functions with different tasks

$t_\tau : \mathbb{D} \rightarrow \{0, 1\}$ $\quad \tau \in \mathrm{T} \equiv$ Parameter set

$t_\tau = \xi \circ \psi \circ \phi$

$\phi : \mathbb{D} \rightarrow \mathbb{R}^n \equiv$ Implicit feature extraction

e.g. $x \in \mathbb{R}^n : \ \phi : \mathbb{R}^n \rightarrow \mathbb{R}^2, x \mapsto (x_i, x_j)^T$

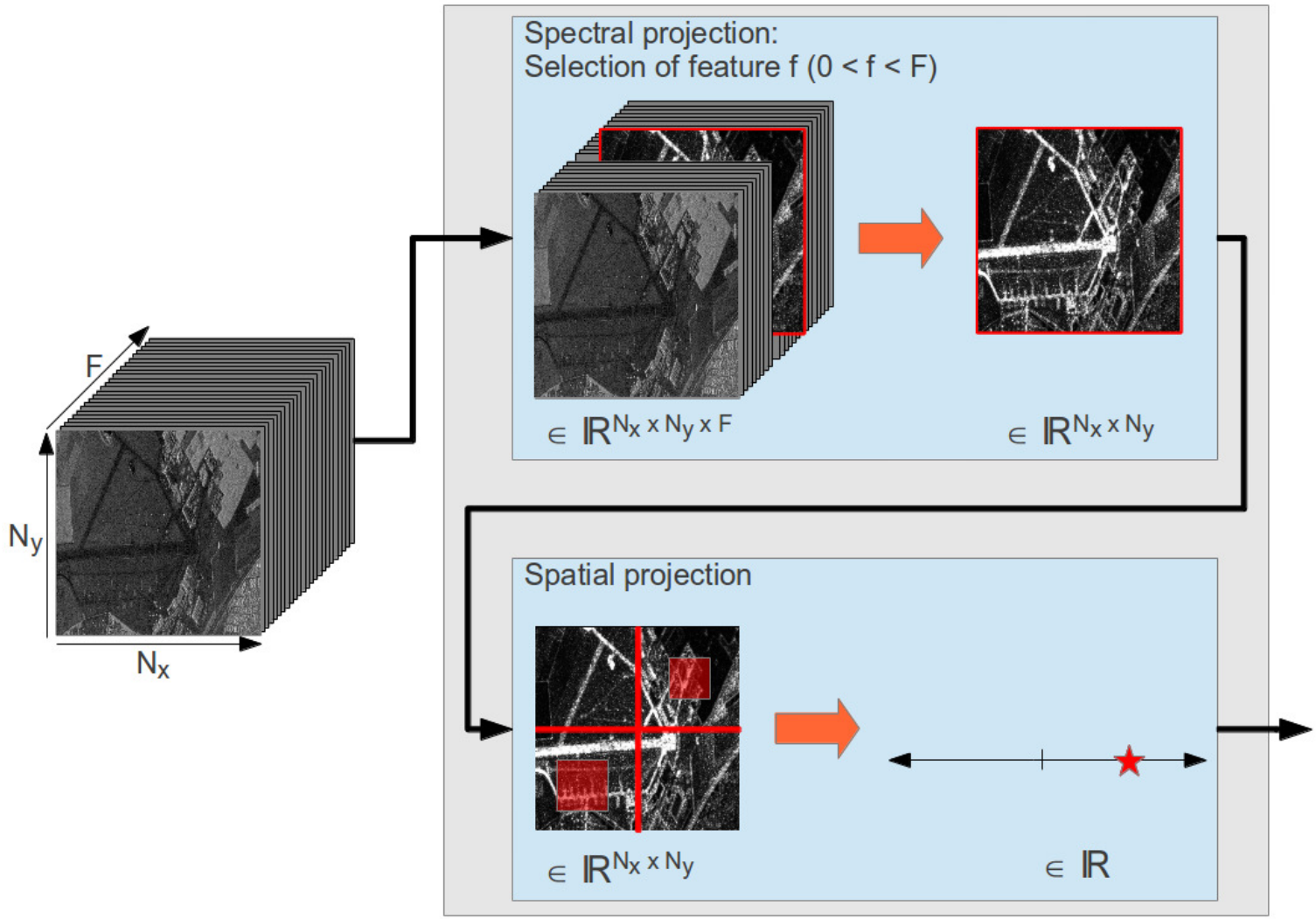
$\psi : \mathbb{R}^n \rightarrow \mathbb{R} \equiv$ Feature fusion

e.g. $\phi(x) \in \mathbb{R}^2 : \ \psi : \mathbb{R}^2 \rightarrow \mathbb{R}, \phi(x) \mapsto [\psi_i, \psi_j] \cdot \phi(x)$

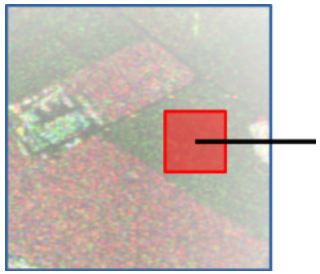
$\xi : \mathbb{R} \rightarrow \{0, 1\} \equiv$ Child node assignment

e.g. thresholding

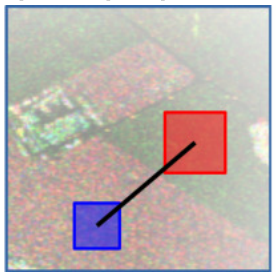
Implicit Feature Extraction



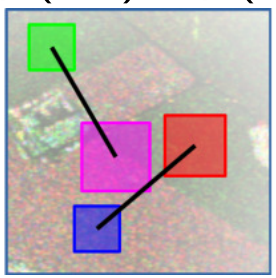
Implicit Feature Extraction



$d(op(R), ref)$



$d(op(R_1), op(R_2))$

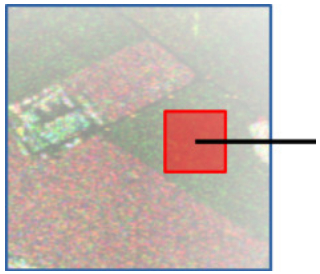


$d(op(R_1), op(R_2)) - d(op(R_3), op(R_4))$

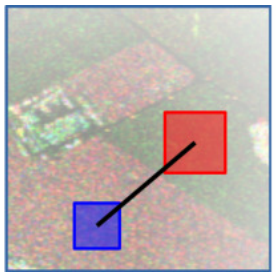
$$t_\tau(\mathbf{x}) = \xi(\psi(\phi(\mathbf{x})))$$

- $\phi : \mathbb{D} \rightarrow \mathbb{R}^n$ can be adopted for different data types
- Region position and size are randomly sampled.
- Different operators *op* (e.g. min, max, selection, averaging, ...)
- Different distance measures *d* (e.g. Euclidean, Frobenius, Bartlet, Wishart, ...)
- Allows RF to work directly on any data type.
- Increases number of possible node tests and thus decreases tree correlation.

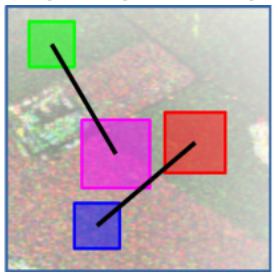
Implicit Feature Extraction



$d(op(R), ref)$



$d(op(R_1), op(R_2))$



$d(op(R_1), op(R_2)) - d(op(R_3), op(R_4))$

$$t_\tau(\mathbf{x}) = \xi(\psi(\phi(\mathbf{x})))$$

Example: Polarimetric SAR data

$\rightarrow$ Patches containing Hermitian matrices

$\rightarrow \mathbf{x} \in \mathbb{C}^{n \times n \times 3 \times 3}$

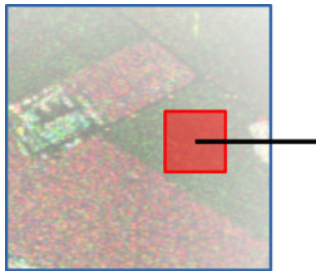
Possible operators

- Select matrix at the center of $\mathbf{x}$
- Average over whole patch
- Select matrix with minimal/maximal span
- ...

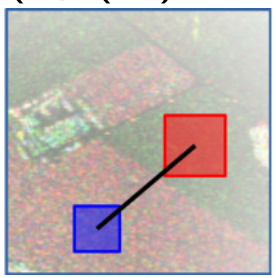
Possible distance measures

- Frobenius norm
- Wishart
- Bartlet
- ...

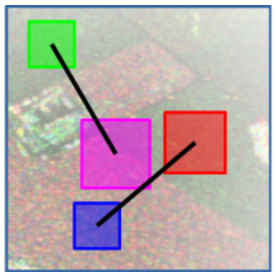
Implicit Feature Extraction



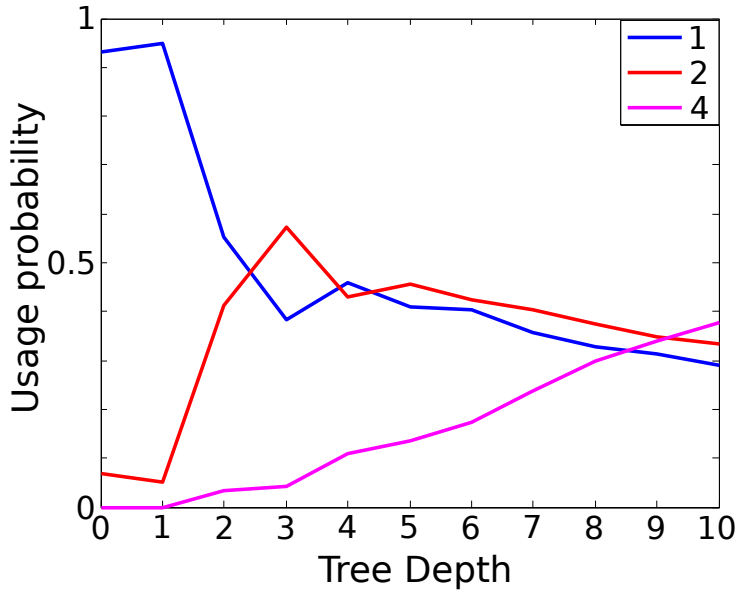
$d(op(R), ref)$



$d(op(R_1), op(R_2))$

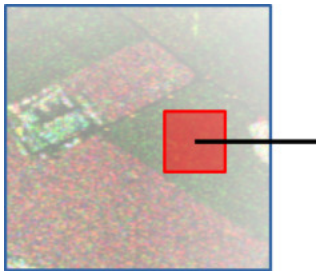


$d(op(R_1), op(R_2)) - d(op(R_3), op(R_4))$

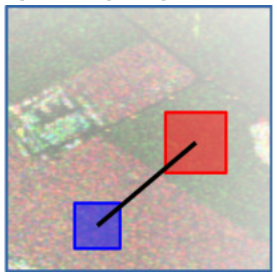


- Works directly on PolSAR data, i.e. no explicit feature extraction
- Automatically selects scale, operators, distance measures, ...
- Reaches competitive results compared to methods based on sophisticated feature extraction

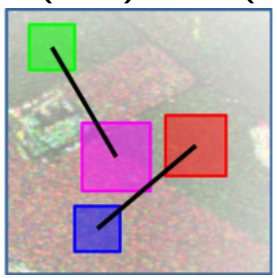
Implicit Feature Extraction



$d(op(R), ref)$

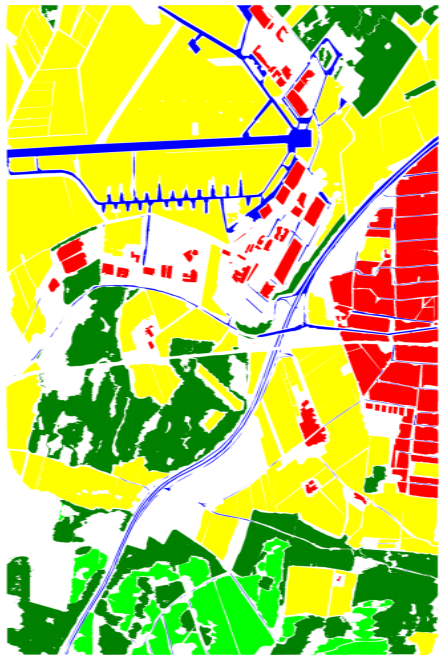


$d(op(R_1), op(R_2))$

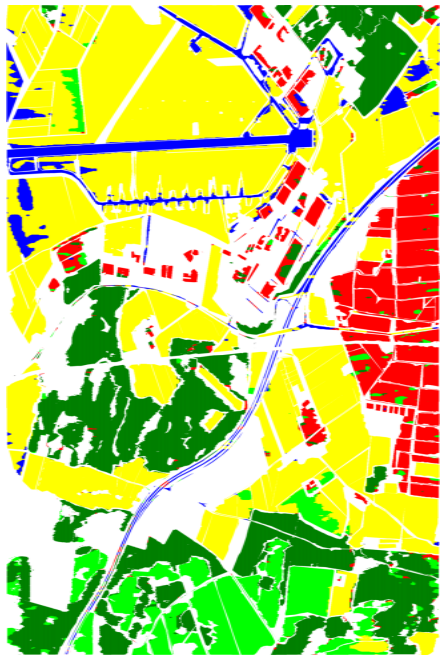


$d(op(R_1), op(R_2)) - d(op(R_3), op(R_4))$

E-SAR, Oberpfaffenhofen



Reference Data
Five classes

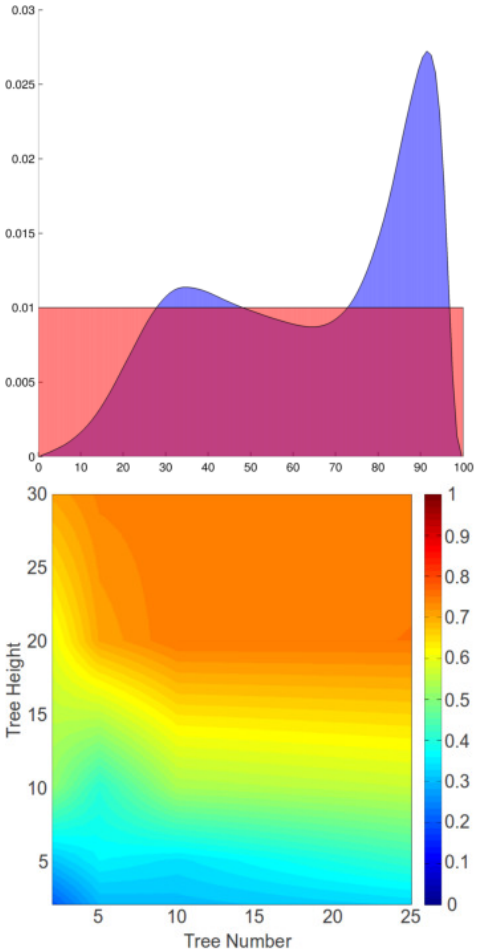


Result
$\sim$ 87% acc.

Random Forests - Split point selection

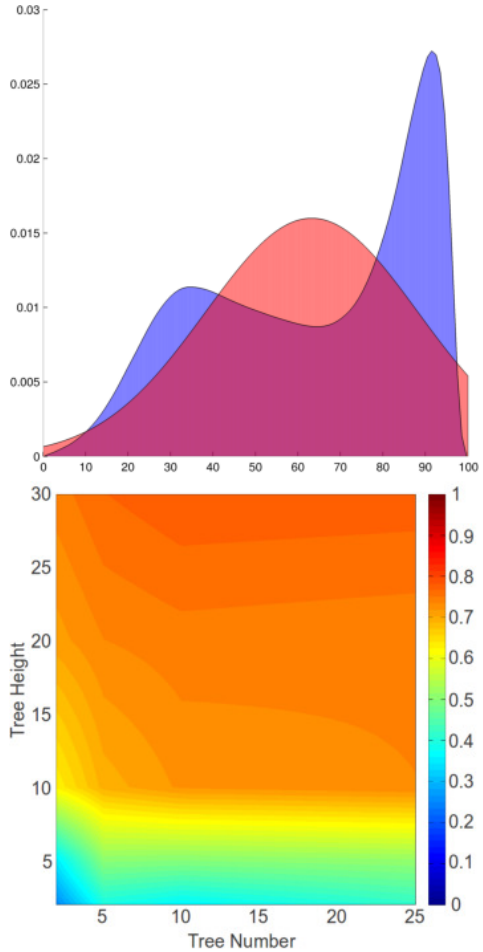
Uniform sampled

$\theta \sim U(\min(\hat{D}), \max(\hat{D}))$



Gaussian sampled

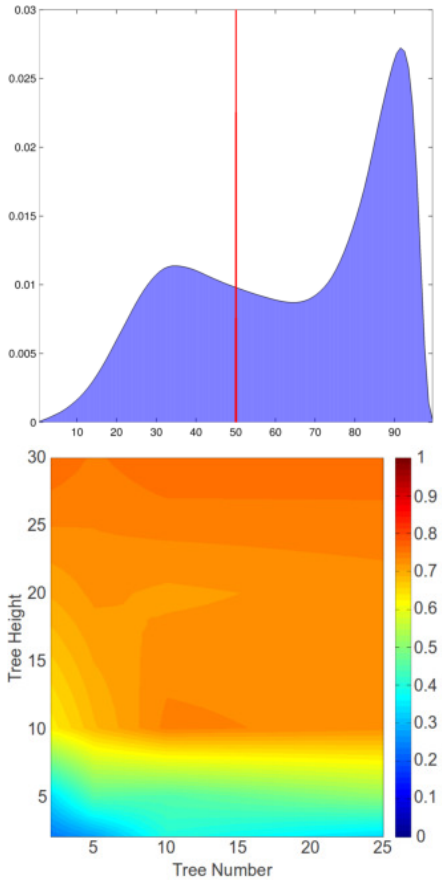
$\theta \sim N(\mu(\hat{D}), \sigma(\hat{D}))$



Random Forests - Split point selection

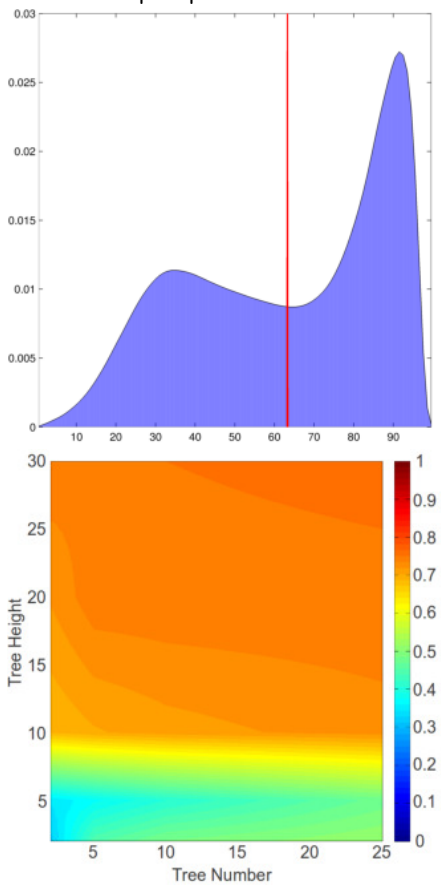
Interval center

$$\theta = \frac{min(\hat{D}) + max(\hat{D})}{2}$$



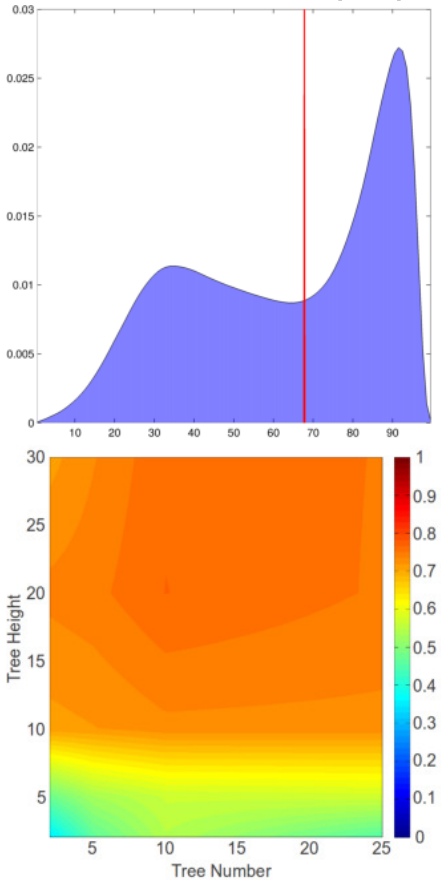
Mean value

$$\theta = \frac{1}{|\hat{D}|} \sum_{x \in \hat{D}} \hat{x}_i$$



Median value

$$\theta = median(\hat{D})$$

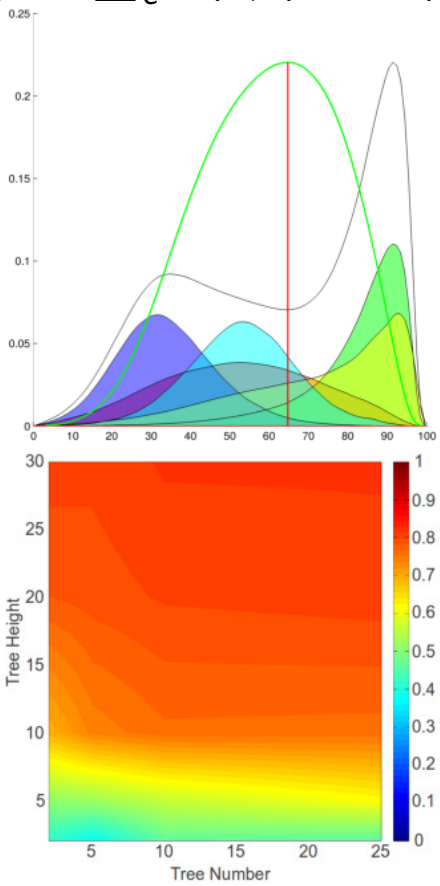


Random Forests - Split point selection

Max. drop of impurity $\theta = \arg\min_{\hat{\theta}} [I(n) - P_L I(n_L) - P_R I(n_R)]$

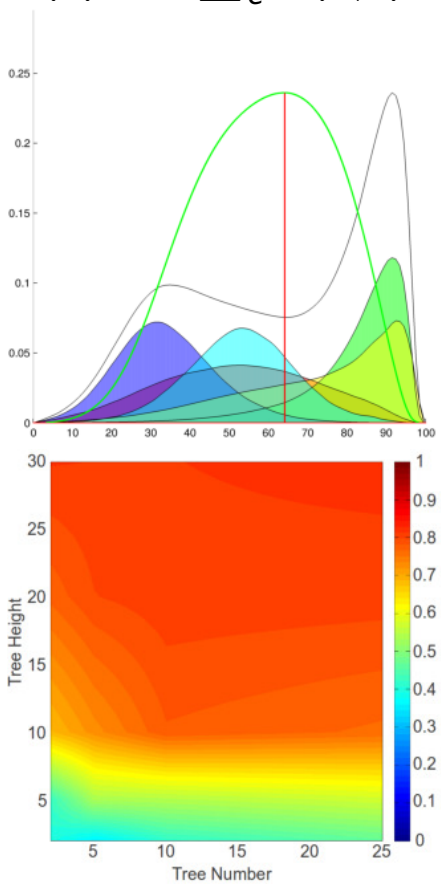
Entropy

$I(n) = -\sum_c P(c|n) \cdot \log P(c|n)$



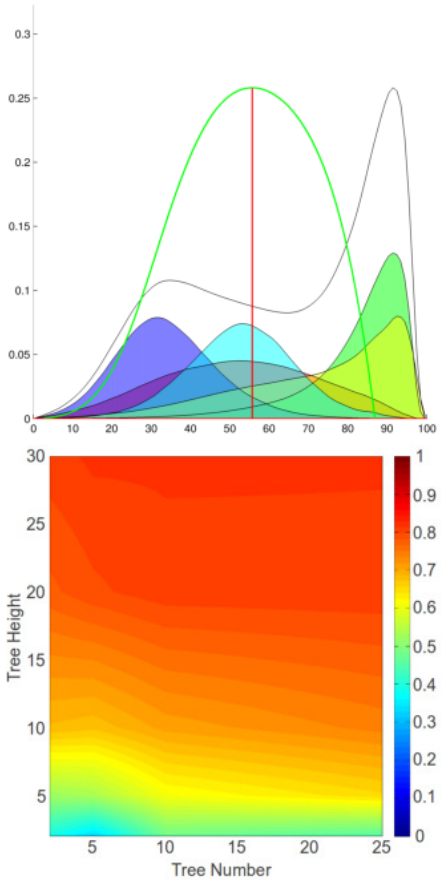
Gini

$I(n) = 1 - \sum_c P(c|n)^2$

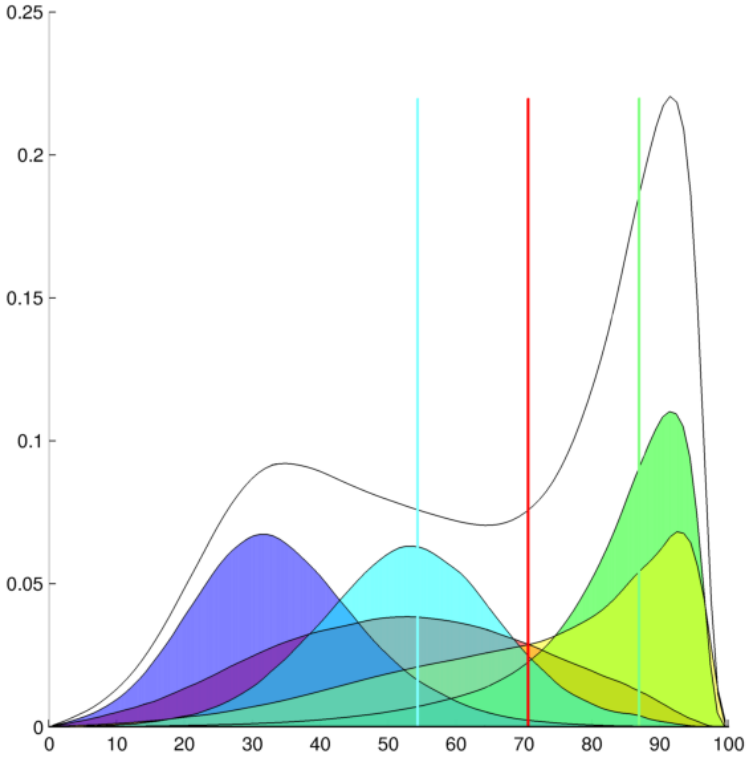


Misclassification

$I(n) = 1 - \max_c P(c|n)$



Random Forests - Split point selection

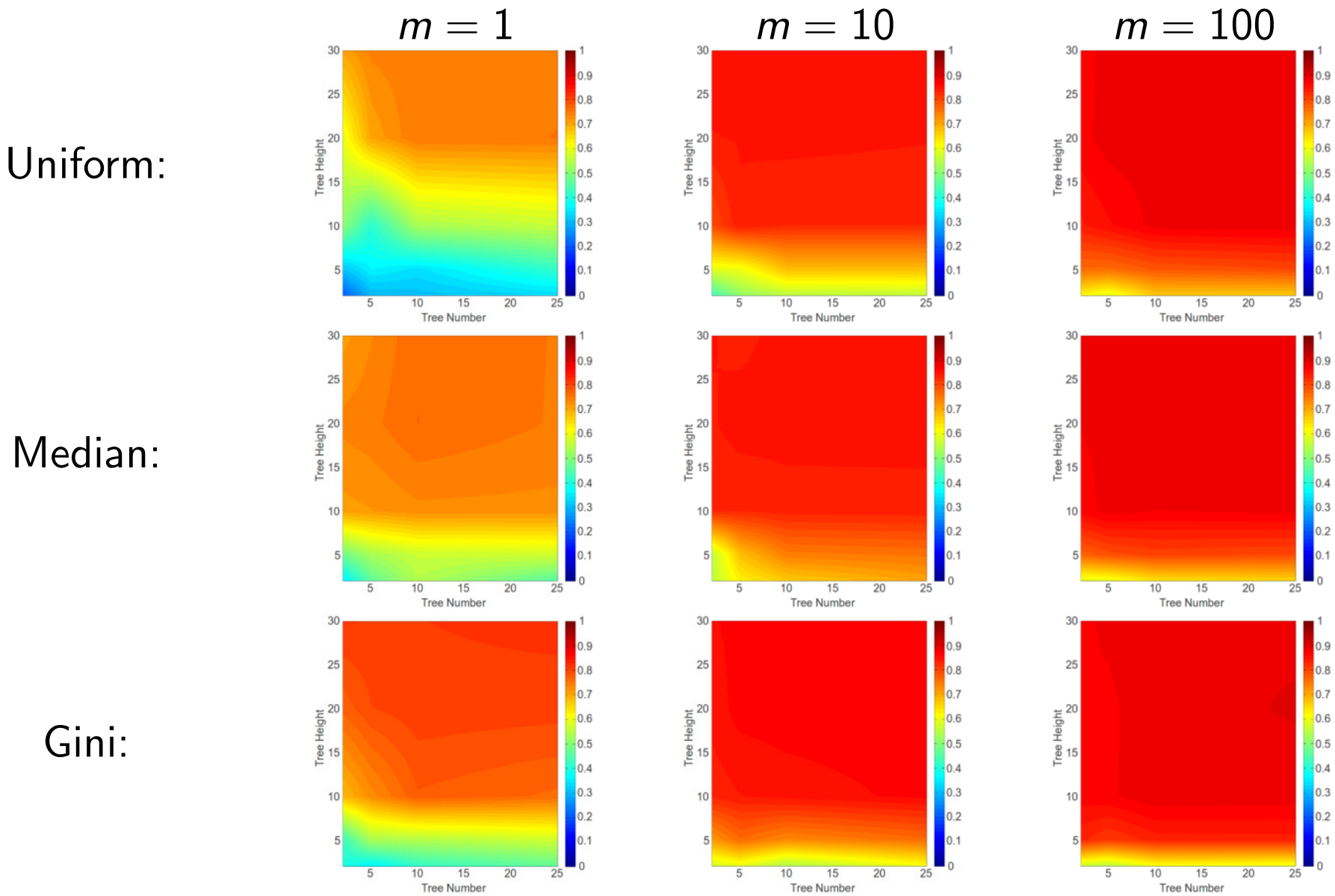


- Other possibilities available
  $\rightarrow$ Intervals, structured label spaces, **inter-class split**
- Need for computational efficiency since selection is performed thousand to million times during training
- Avoid exhaustive search

Random Forests - Node optimization

- Generate m split candidates
  $\rightarrow$ "Traditionally": $m = \sqrt{d}$, where d is data dimension
  $\rightarrow$ "Modern" approaches: $m \approx 10^5$
  $\rightarrow$ Usually even $m = 2$ leads to performance increase
  $\rightarrow$ Trade-off between high performance and high correlation
- Select best split, reject all others
- Measure optimality of a split
  - Classification: "Purity" of child nodes (e.g. Gini, entropy, etc.)
  - Regression: e.g. variance
  - In general: How much better is the estimation of the child nodes (as a weighted average) than parent nodes?

Random Forests - Node optimization



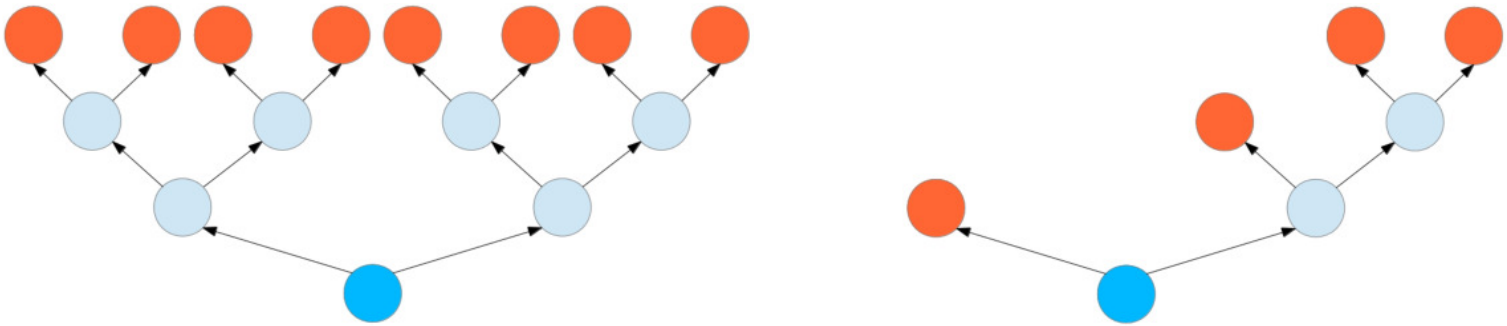
Random Forests - Node optimization

- Different energy functions allow simultaneous optimization of different targets
- Common example: Classification (Object class) **and** regression (Object position)
- e.g. **Hough Forests**[Gall et al., 2011]
  - Training Data: $D = \{P_i = (I_i, c_i, d_i)\}$
  - Randomly decide for one of two energy functions:
    Entropy of posterior: $U_1(A) = -|A| \cdot \sum_c P(c|A) log(P(c|A))$
    Variance of offset vectors: $U_2(A) = \sum_c \sum_{d \in D_c^A} ||d - \bar{d}_c^A||$
  - Select best test t according to
    $\arg\min_t(U_*(\{P_i|t=0\}) + U_*(\{P_i|t=1\}))$
  - Use offset vectors and class posterior to perform Hough voting during prediction

Random Forests - Structured prediction

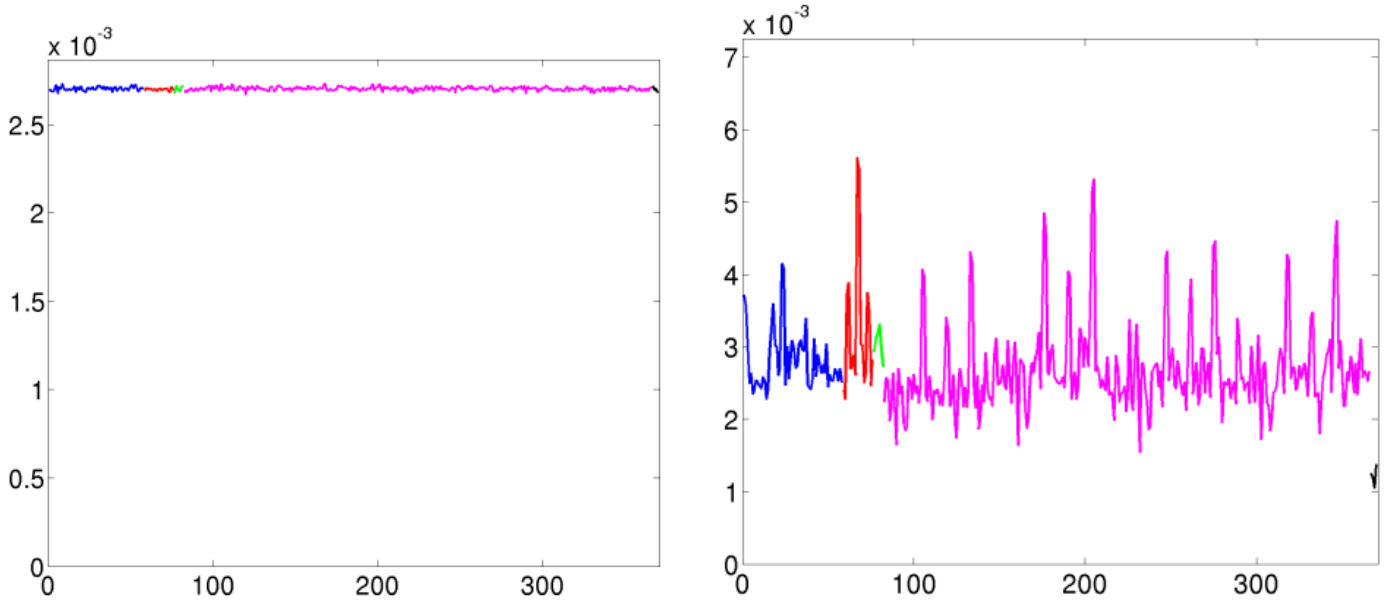
- Image data is structured
- Exploited already during structured projection within node tests
- Target variable can be structured as well
  $\rightarrow$ Offset vectors, label patch
- Enriched spatial estimate for image labeling
- Disadvantage: Increased memory footprint

Random Forests - Interpretation



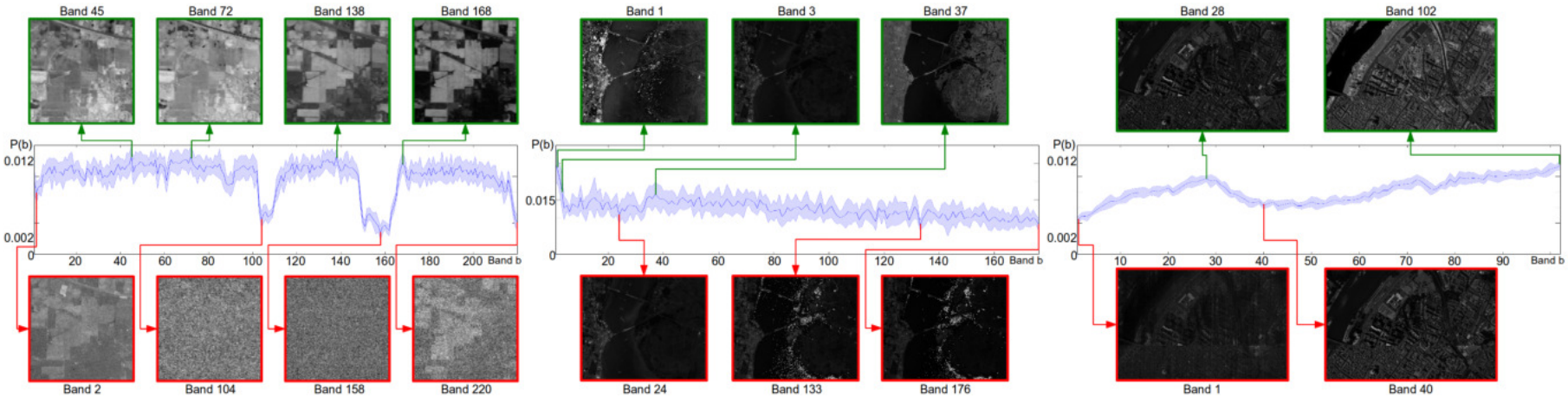
- Is maximum tree height reached?
- How balanced is a tree? $\frac{\#\text{nodes}}{2^{H+1}-1}$
- How large is largest leaf? $1 - \frac{\max_{n_t} |D_{n_t}|}{|D|}$
- How pure is largest leaf? $I(n^*)$ with $n^* = \arg\max_{n_t} |D_{n_t}|$
- Out-of-bag estimate for generalization error
- Out-of-bag estimate for correlation

Random Forests - Feature relevance



- RF for PolSAR image labeling, roughly 360 image features as input:
  (PolSAR-blue, SAR-red, color-green, grayscale-magenta, binary-black)
- Each feature has same probability to be used (as seen on the left)
- Each feature is actually selected by the RF with very unequal frequency (as seen on the right)
- Features that have been used often are more important / descriptive for the task at hand

Random Forests - Feature relevance



[R. Hänsch, 2015a]

- RF for hyperspectral image labeling, $>$ 200 spectral bands as input
- Each band has same probability to be used
- Each band is actually selected by the RF with very unequal frequency
- Bands that have been used often are more important / descriptive for the task at hand

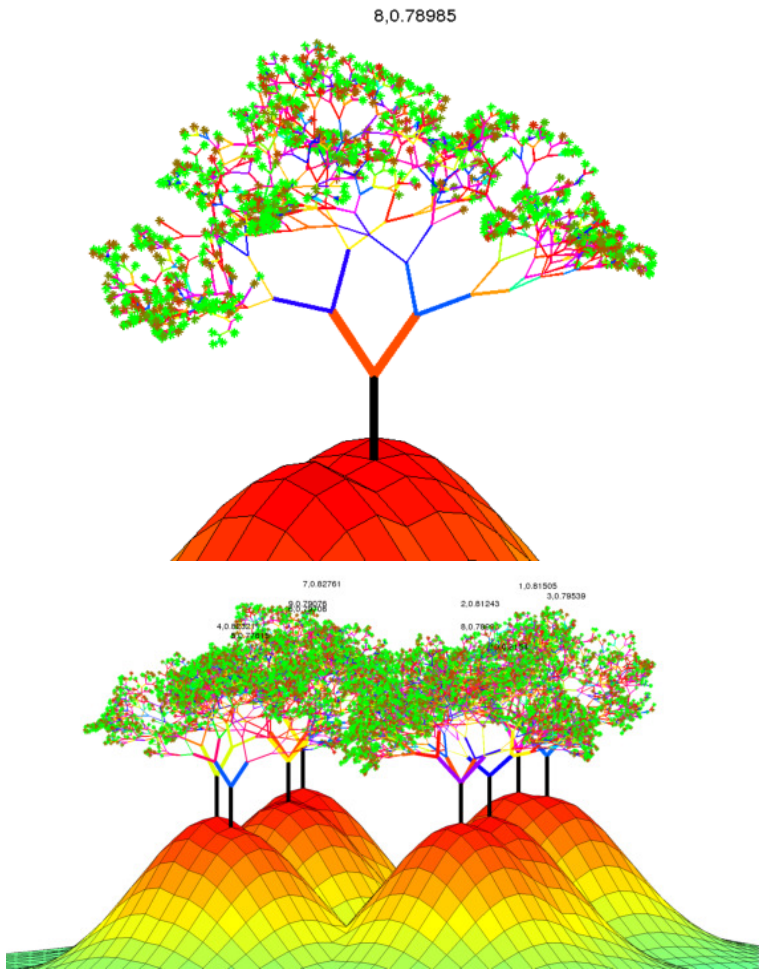
Random Forests - Visualization

Important Characteristics of a Random Forests

- Forest Level
  - Strength of the whole forest
    → e.g. classification accuracy
  - Correlation between trees
    → e.g. correlation of classification maps
- Tree Level
  - Strength of the individual tree
    → e.g. based on out-of-bag error
  - Structural layout of individual trees
    → e.g. balanced vs. degenerated chain
- Node Level
  - Node features
    → e.g. size, split dimension, drop of impurity, leaf impurity, etc.

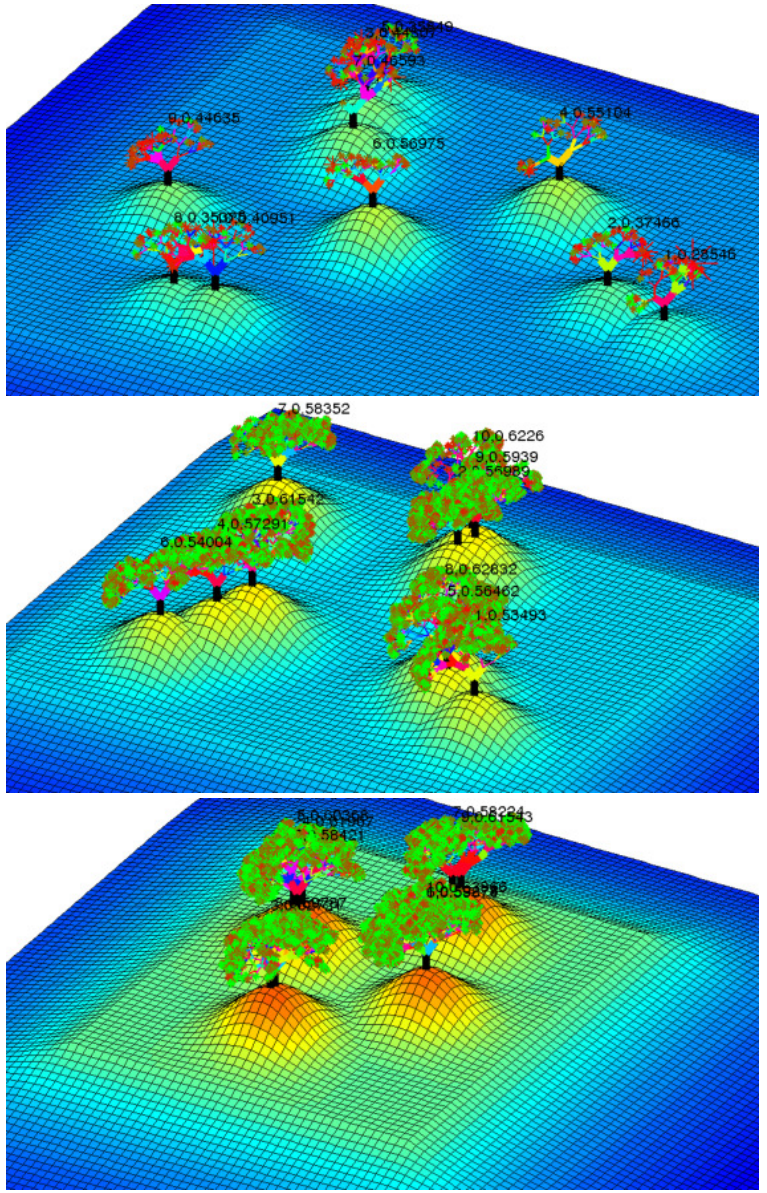
Random Forests - Visualization

- Branch
  - Color:
    (e.g.) split dimension
  - Thickness:
    Number of samples
  - Length:
    $l_{h+1}^{L/R} = l_h \cdot \kappa_2 \cdot ((f_{\text{max}} - f_{\text{min}}) + f_{\text{min}})$
  - Orientation:
    $(\alpha, \beta)_{h+1}^{L/R} = (\alpha, \beta)_h \pm (30°, \kappa_1 \cdot 45°)$
- Leaf
  - Color: Leaf impurity
  - Size: Leaf size
- 2D position
  Based on pairwise correlation



Random Forests - Visualization

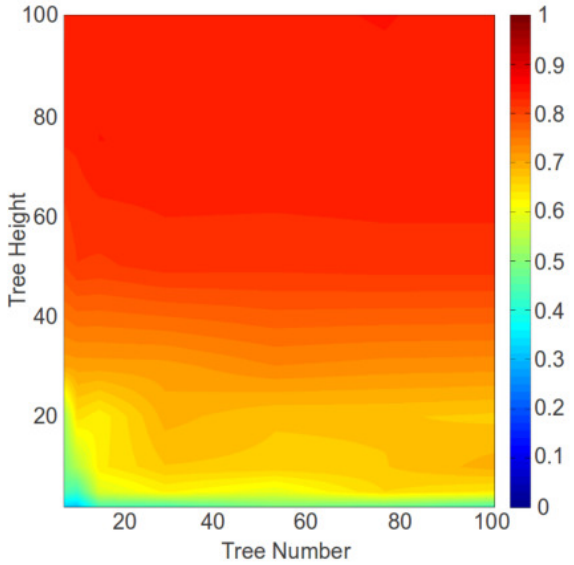
- Increasing tree height
- Trees getting higher
- Leafs getting purer
- Trees getting stronger
- Trees correlate stronger
- Forest gets stronger
- [R. Hänsch, 2015b]



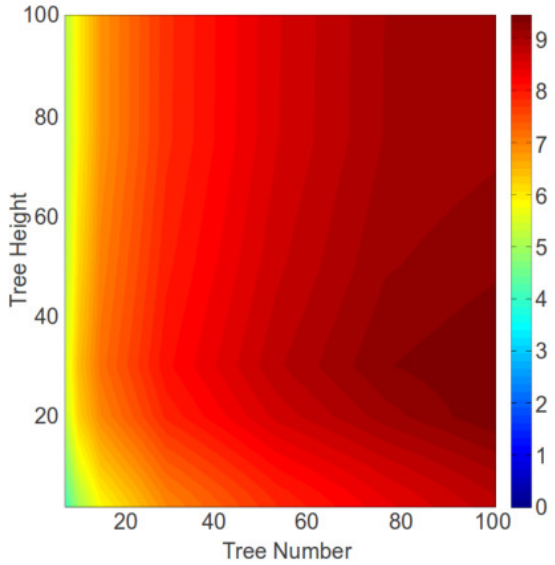
Random Forests - Practical Considerations

- Most implementations refer to standard RF (no spatial projections)
- Projection-based RF common in CV community (due to images!)
- GPU implementations available
- Many spatial projections can be approximated by integral images
- Not all data samples in a node have to be used to define / select split point
- Tree creation (i.e. development of tree topology) can be performed independently from tree training (i.e. assign estimates to leaf nodes)
  $\rightarrow$ Only tree creation needs multiple data samples in memory.
  $\rightarrow$ Tree training can be done one sample after another
  $\rightarrow$ Online training for RF

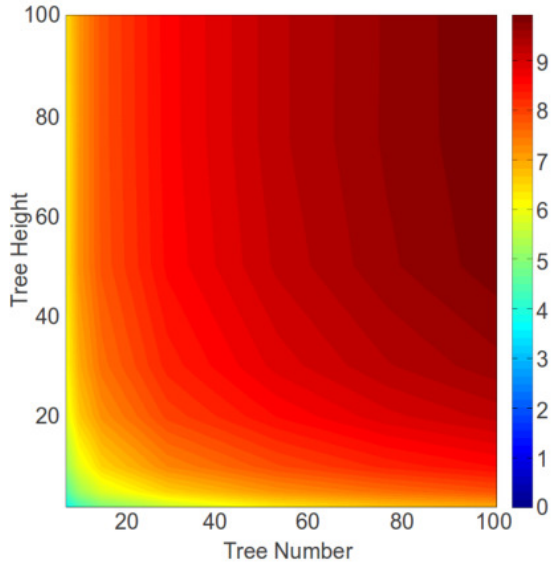
Random Forests - Best Practice



Accuracy



log-Training time

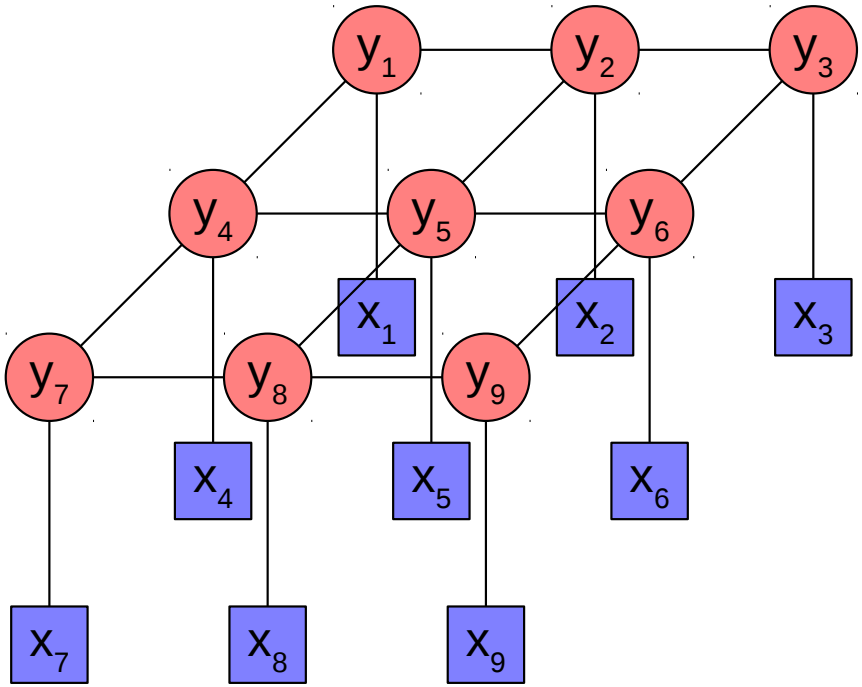


log-Prediction time

- Accuracy usually grows faster with tree height than tree number
- But: Tree height limited by amount of training data
- Use projections / features that are as diverse as possible
- Use simple split point definitions in combination with node optimization (i.e. selection)
- Check tree properties / visualize

Overview

1. Random Forests
   - Fusion
   - Node Tests
   - Interpretation
   - Application Tips
2. Graphical Models
   - Motivation
   - MRF
   - Application
   - Example
3. ConvNets
   - MLP to ConvNet
   - Convolution
   - Architectures
   - Auto Encoder
   - Frameworks



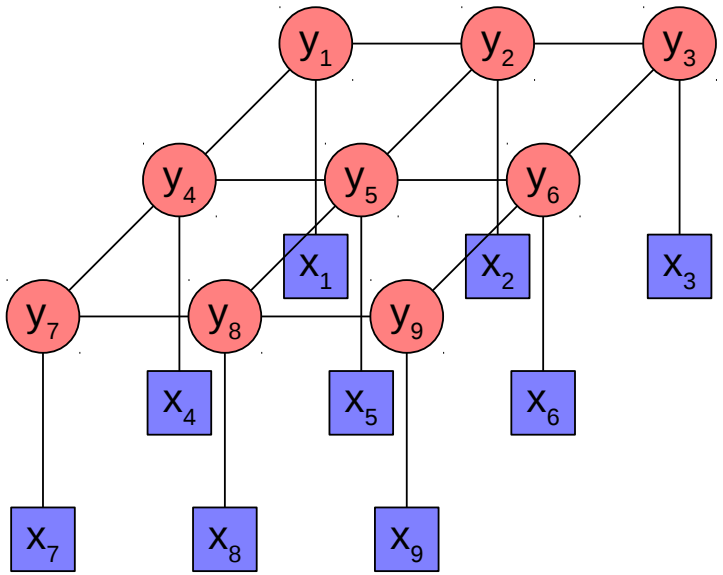
MRF

The same probabilistic model (e.g. for image labeling) as

mathematical equation

$$
\begin{aligned}
p(\mathbf{x},\mathbf{y}) \quad = \quad & p(y_3,x_3)p(y_5,y_6)p(y_8,y_9) \\
& p(y_8,x_8)p(y_9,x_9)p(y_3,y_6) \\
& p(y_6,y_9)p(y_1,x_1)p(y_2,y_5) \\
& p(y_4,y_7)p(y_4,y_5)p(y_6,x_6) \\
& p(y_2,x_2)p(y_5,y_8)p(y_7,x_7) \\
& p(y_2,y_3)p(y_1,y_4)p(y_5,x_5) \\
& p(y_1,y_2)p(y_4,x_4)p(y_7,y_8)
\end{aligned}
$$

graphical model



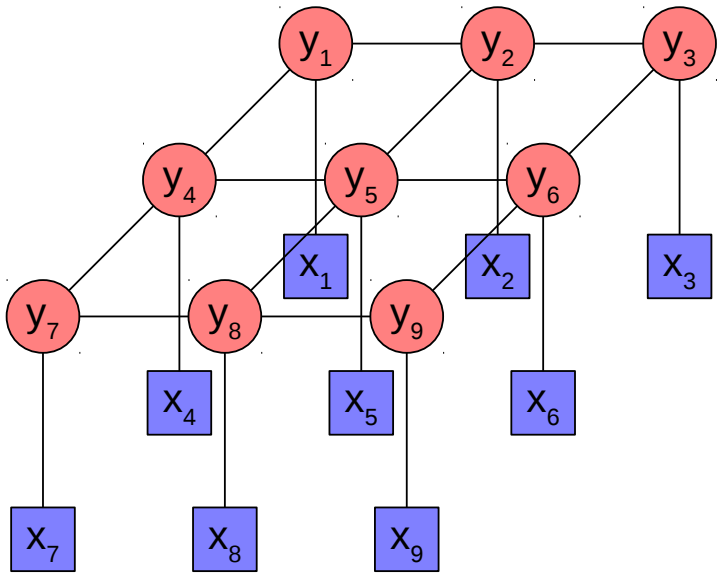
MRF

The same probabilistic model (e.g. for image labelling) as

mathematical equation

$$
\begin{aligned}
p(\mathbf{x}, \mathbf{y}) \quad = \quad & p(y_3, x_3)p(y_5, y_6)p(y_8, y_9) \\
& p(y_8, x_8)p(y_9, x_9)p(y_3, y_6) \\
& p(y_6, y_9)p(y_1, x_1)p(y_2, y_5) \\
& p(y_4, y_7)p(y_4, y_5)p(y_6, x_6) \\
& p(y_2, x_2)p(y_5, y_8)p(y_7, x_7) \\
& p(y_2, y_3)p(y_1, y_4)p(y_5, x_5) \\
& p(y_1, y_2)p(y_4, x_4)p(y_7, y_8)
\end{aligned}
$$

graphical model

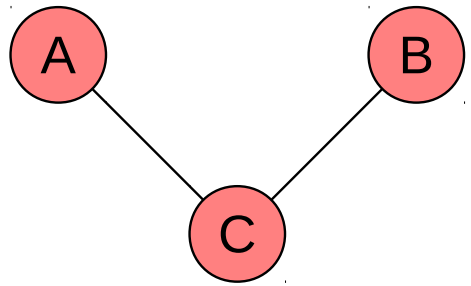


- Dependencies visually recognizable
- Different types of nodes
- Scales easily to larger models (if there are no structural changes)

MRF - Conditional Independence

Events A and B are conditional independent given event C: $A \perp B|C$

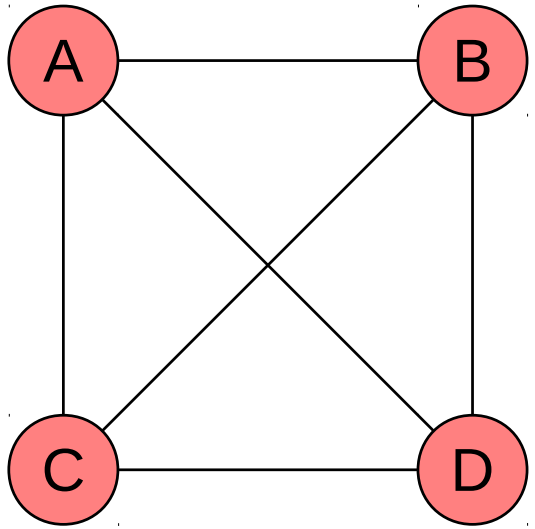
$$
\begin{aligned}
P(A, B|C) &= P(A|C) \cdot P(B|C) \\
P(A|B, C) &= P(A|C) \\
P(B|A, C) &= P(B|C)
\end{aligned}
$$



- My blood type $\perp$ Blood type of grandparents | Blood type of parents
- Weather tomorrow $\perp$ Weather yesterday | Weather today
- Value of a pixel $\perp$ Value of pixels far away | Value of adjacent pixels

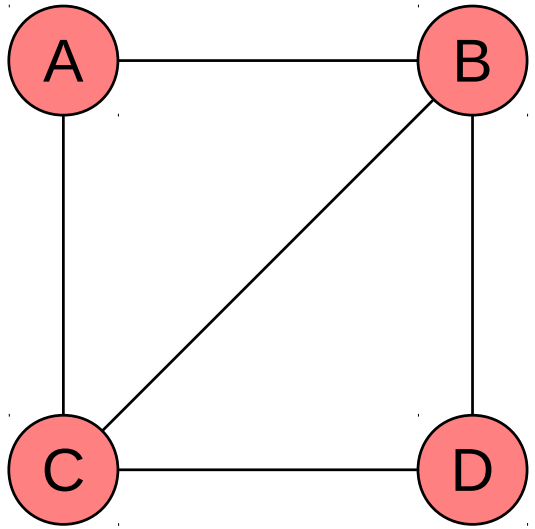
MRF - Conditional Independence

$$
\begin{aligned}
P(A,B,C,D) &= P(A|B,C,D) \\
&\cdot P(B|C,D) \\
&\cdot P(C|D) \\
&\cdot P(D)
\end{aligned}
$$



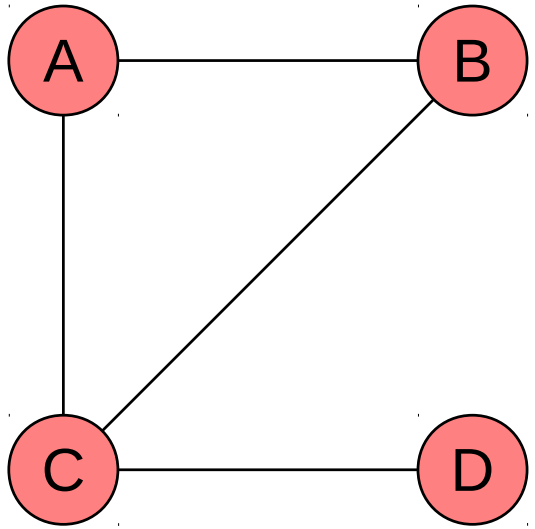
MRF - Conditional Independence

$$
\begin{aligned}
A \perp D \quad &| \quad C, B \\
P(A,B,C,D) \quad &= \quad P(A|B,C) \\
&\quad \cdot P(B|C,D) \\
&\quad \cdot P(C|D) \\
&\quad \cdot P(D)
\end{aligned}
$$



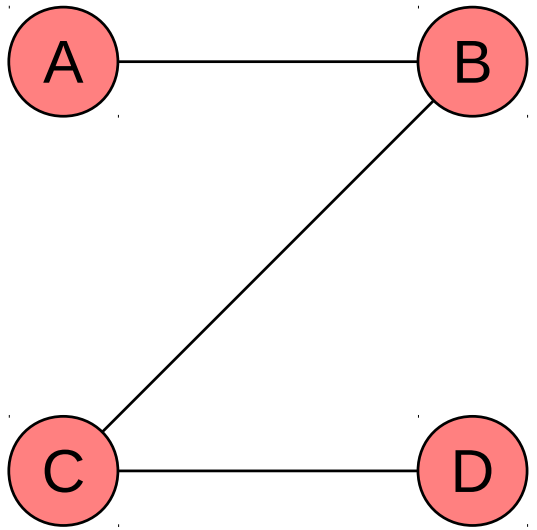
MRF - Conditional Independence

$$
\begin{aligned}
A \perp D \quad &| \quad C, B \\
B \perp D \quad &| \quad C \\
P(A, B, C, D) \quad &= \quad P(A|B, C) \\
&\quad \cdot P(B|C) \\
&\quad \cdot P(C|D) \\
&\quad \cdot P(D)
\end{aligned}
$$



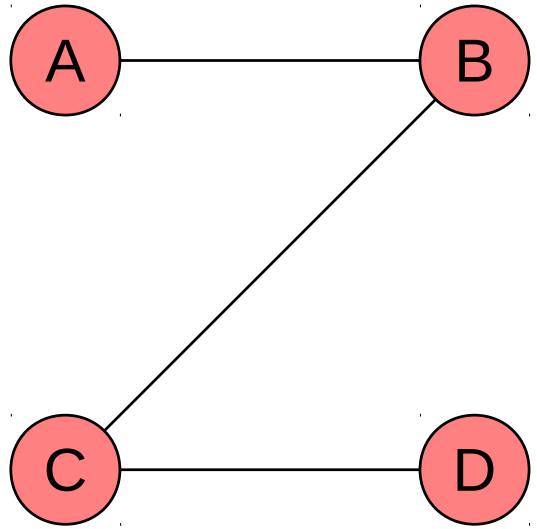
MRF - Conditional Independence

$$
\begin{aligned}
A \perp D \quad &| \quad C, B \\
B \perp D \quad &| \quad C \\
A \perp C \quad &| \quad B \\
P(A,B,C,D) \quad &= \quad P(A|B) \\
&\quad \cdot P(B|C) \\
&\quad \cdot P(C|D) \\
&\quad \cdot P(D)
\end{aligned}
$$



MRF - Conditional Independence

$$
\begin{aligned}
A \perp D \quad &| \quad C, B \\
B \perp D \quad &| \quad C \\
A \perp C \quad &| \quad B \\
P(A,B,C,D) \quad &= \quad P(A|B) \\
&\quad \cdot P(B|C) \\
&\quad \cdot P(C|D) \\
&\quad \cdot P(D)
\end{aligned}
$$

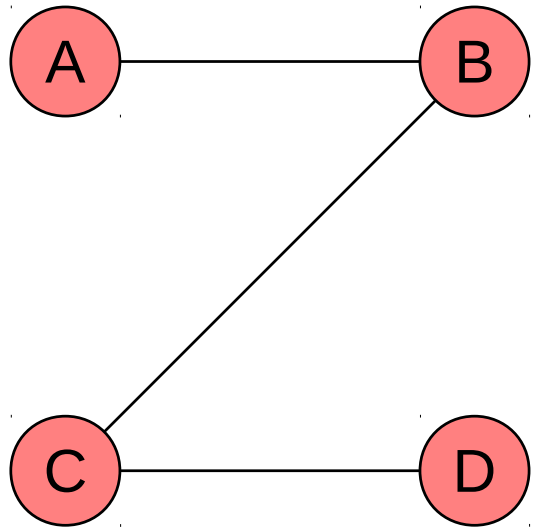


Marginalization: $P(A) = \sum_B \sum_C \sum_D P(A,B,C,D)$

$\rightarrow |A| \cdot |B| \cdot |C| \cdot |D|$ operations

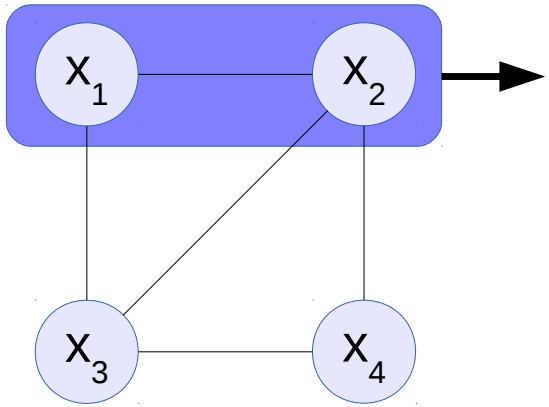
MRF - Conditional Independence

$$
\begin{aligned}
A \perp D \quad &| \quad C, B \\
B \perp D \quad &| \quad C \\
A \perp C \quad &| \quad B \\
P(A,B,C,D) \quad &= \quad P(A|B) \\
&\quad \cdot P(B|C) \\
&\quad \cdot P(C|D) \\
&\quad \cdot P(D)
\end{aligned}
$$



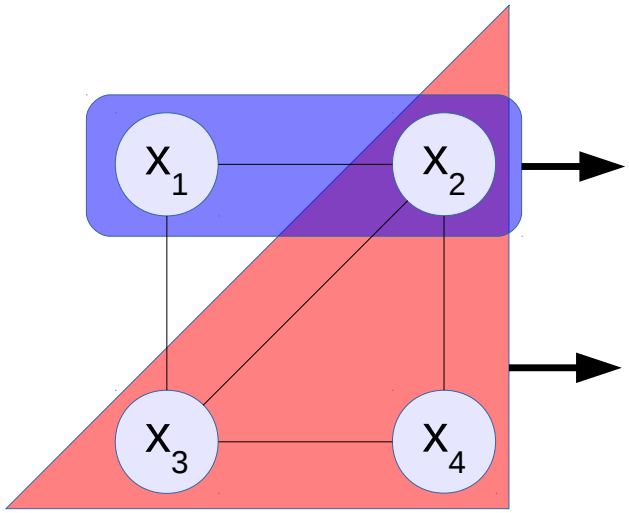
$$
\begin{aligned}
\text{Marginalization: } P(A) &= \sum_B \sum_C \sum_D P(A|B) \cdot P(B|C) \cdot P(C|D) \cdot P(D) \\
&= \sum_B P(A|B) \cdot \sum_C P(B|C) \cdot \sum_D P(C|D) \cdot P(D) \\
&\rightarrow |A| \cdot (|B| + |C| + |D|) \text{ operations}
\end{aligned}
$$

MRF - Notation



Clique: Complete subgraph
→ all pairs of nodes connected

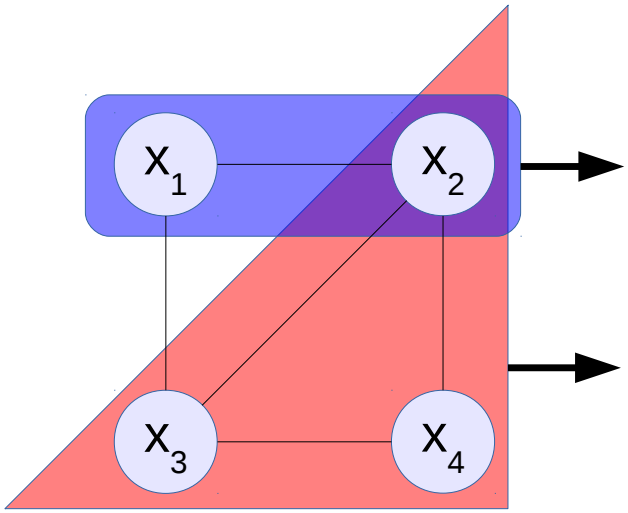
MRF - Notation



Clique: Complete subgraph
→ all pairs of nodes connected

Maximal Clique:
→ No subset of any other clique

MRF - Notation



Clique: Complete subgraph
→ all pairs of nodes connected

Maximal Clique:
→ No subset of any other clique

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in C} \Psi_c(\mathbf{x}_c)$$

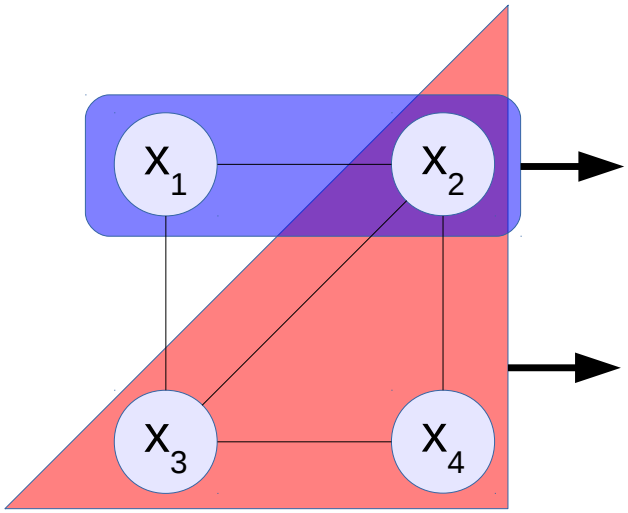
C : Set of maximal cliques

$\mathbf{x}_c$: Nodes in clique c

$\Psi(\cdot)$: Non-negative real-valued function, e.g. $\exp(\psi(\mathbf{x}_c))$

Z : Normalization such that $\int p(\mathbf{x}) d\mathbf{x} = 1$

MRF - Notation

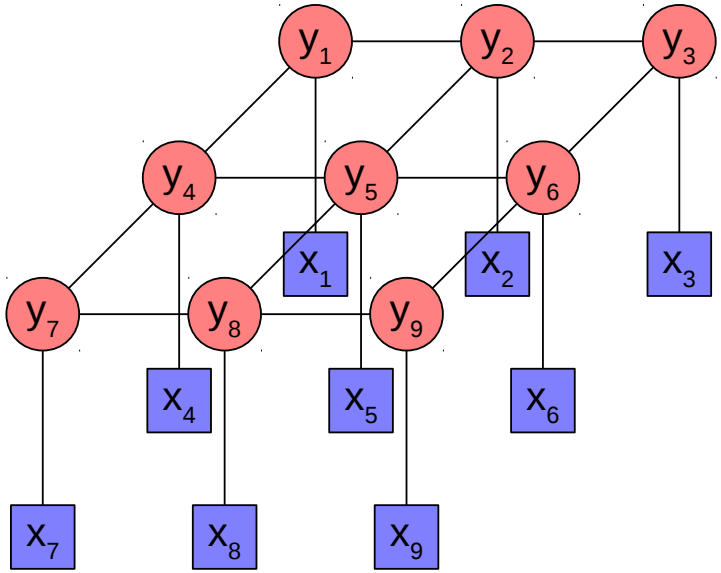


Clique: Complete subgraph
→ all pairs of nodes connected

Maximal Clique:
→ No subset of any other clique

$$
\begin{aligned}
p(\mathbf{x}) &= \frac{1}{Z} \prod_{c \in C} \Psi_c(\mathbf{x}_c) \\
&= \frac{1}{Z} \prod_{c \in C} \exp(\psi_c(\mathbf{x}_c)) \\
&= \frac{1}{Z} \exp\left(\sum_{c \in C} \psi_c(\mathbf{x}_c) \right) = \frac{1}{Z} \exp(-E(\mathbf{x}))
\end{aligned}
$$

MRF - Notation

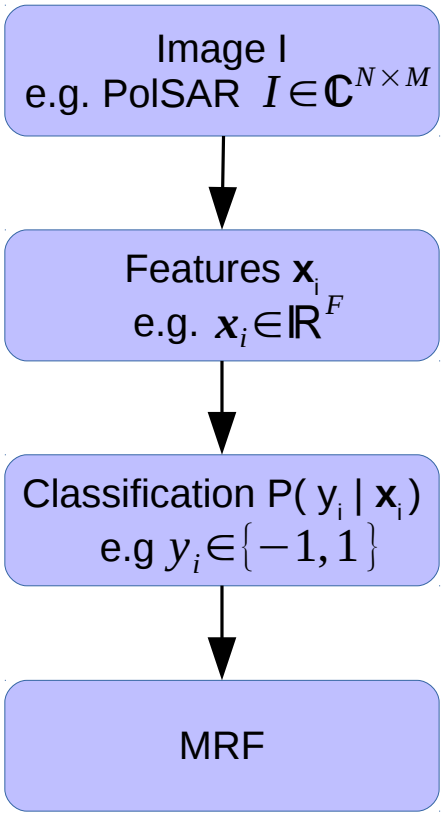


$$p(\mathbf{x},\mathbf{y}) = \frac{1}{Z}\prod_{c\in C}\Psi_c(\mathbf{x}_c,\mathbf{y}_c) = \frac{1}{Z}\exp(-E(\mathbf{x},\mathbf{y}))$$

$$E(\mathbf{x},\mathbf{y}) = \sum_{y_i}\underbrace{\psi^U(y_i,x_i)}_{\text{Unary energy}} + \sum_{y_i,y_j}\underbrace{\psi^B(y_i,y_j)}_{\text{Binary energy}} + \sum_{c\in C}\underbrace{\psi^H(\mathbf{y}_c,\mathbf{x}_c)}_{\text{Higher-order energy}}$$

MRF - Simple Example

Binary classification task
MRF used for simple label smoothing

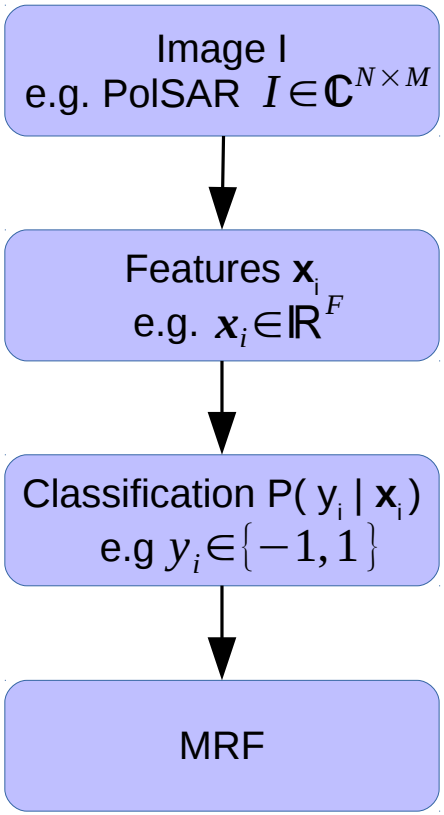


$$\psi^U(y_i, x_i) = -\log P(y_i|x_i)$$
$$\psi^B(y_i, y_j) = -y_i \cdot y_j$$

MRF - Simple Example

Binary classification task

MRF with contrast-dependent smoothness prior



$$\psi^U(y_i, x_i) = -\log P(y_i|x_i)$$

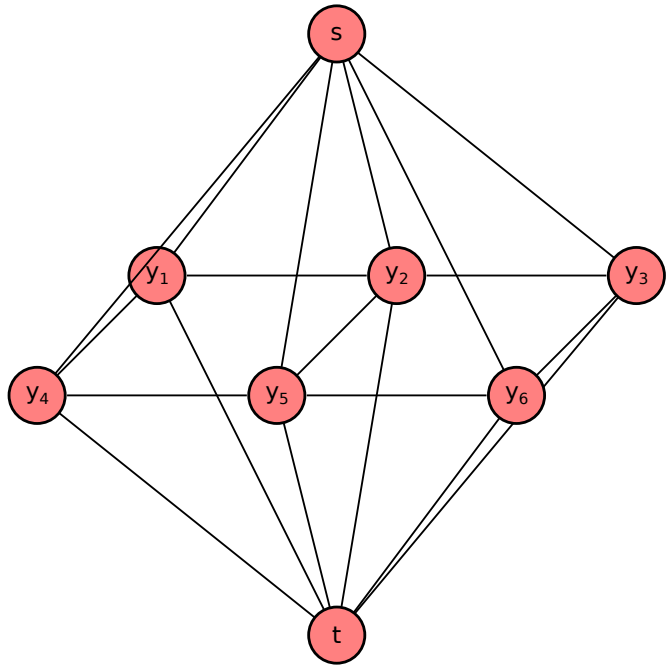
$$\psi^B(y_i, y_j) = \begin{cases} \lambda_0 + \lambda_1 e^{\left(-\frac{||x_i - x_j||^2}{2\beta}\right)} & \text{if } y_i \neq y_j \\ 0 & \text{otherwise.} \end{cases}$$

Graph Cut

- Given all $\mathbf{x}$, what are the most probable $\mathbf{y}$?
- Which $\mathbf{y}$ minimizes $E(\mathbf{x}, \mathbf{y}) = \sum_{y_i} \psi^U(y_i, x_i) + \sum_{y_i, y_j} \psi^B(y_i, y_j)$?

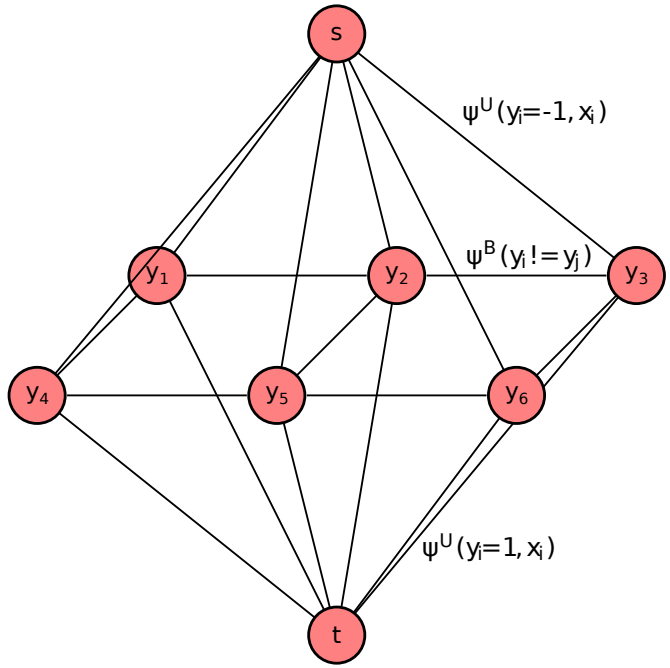
Graph Cut

- Given all **x**, what are the most probable **y**?
- Which **y** minimizes $E(\mathbf{x}, \mathbf{y}) = \sum_{y_i} \psi^U(y_i, x_i) + \sum_{y_i, y_j} \psi^B(y_i, y_j)$?
- Build a new graph with source and sink
- Binary potentials as weights between pixels
- Unary potentials als weights towards source/sink



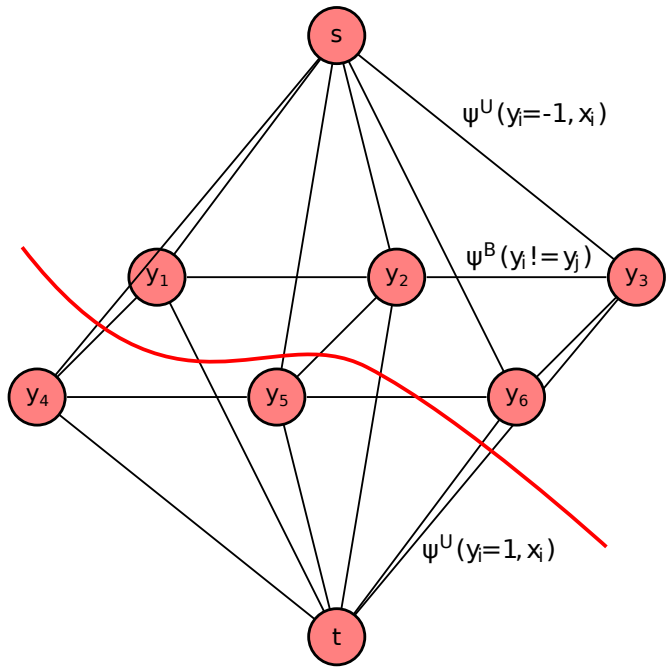
Graph Cut

- Given all **x**, what are the most probable **y**?
- Which **y** minimizes $E(\mathbf{x}, \mathbf{y}) = \sum_{y_i} \psi^U(y_i, x_i) + \sum_{y_i, y_j} \psi^B(y_i, y_j)$?
- Build a new graph with source and sink
- Binary potentials as weights between pixels
- Unary potentials as weights towards source/sink



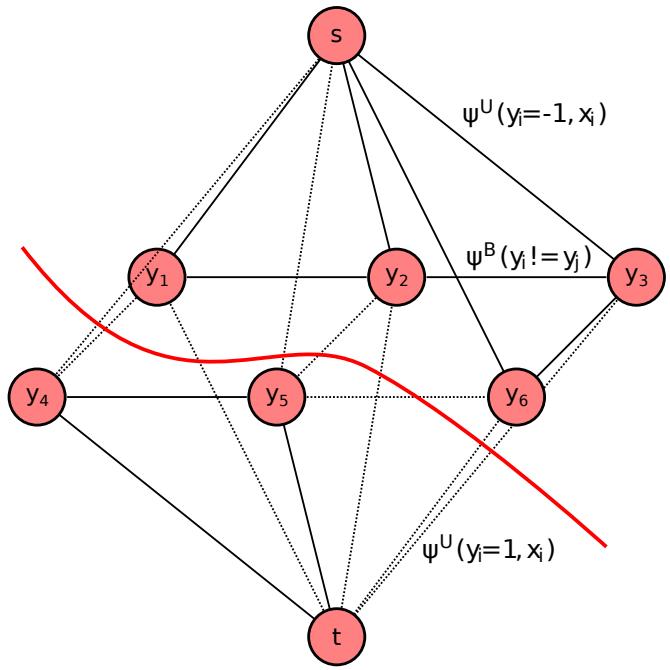
Graph Cut

- Min-Cut:
  - Cut that separates source and sink
  - Sum of cut edges (their weight) minimal
- This gives labeling of lowest MRF energy!
- Computing Min-Cut standard problem with lots of implementations



Graph Cut

- Min-Cut:
  - Cut that separates source and sink
  - Sum of cut edges (their weight) minimal
- This gives labeling of lowest MRF energy!
- Computing Min-Cut standard problem with lots of implementations



Non-Binary Classification

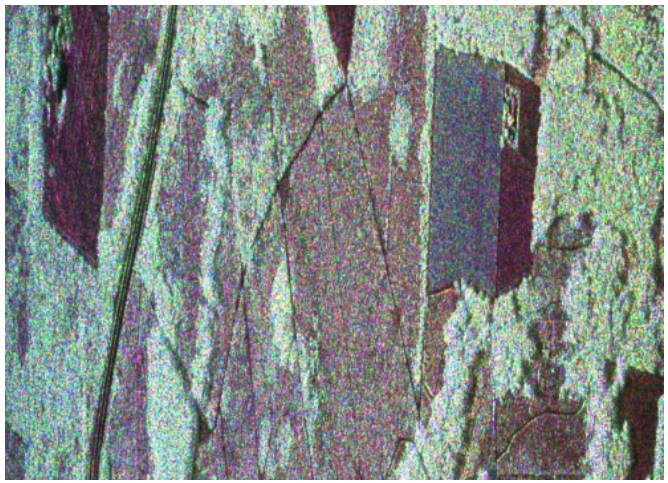
- What about non-binary, multi class problems?

Non-Binary Classification

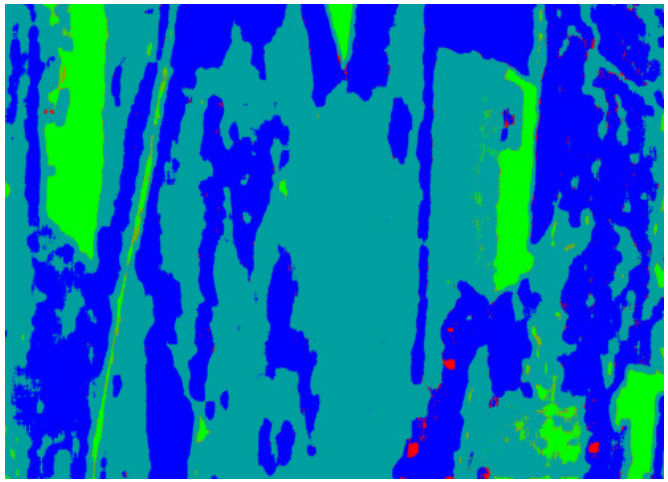
- What about non-binary, multi class problems?
- Approximate solution: Alpha-Expansion [Boykov et al., 2001]
- Loop over all labels
- For each label, allow region expansion based on graph cut
- Make multiple passes
- Graph slightly different

MRF - Alpha Expansion

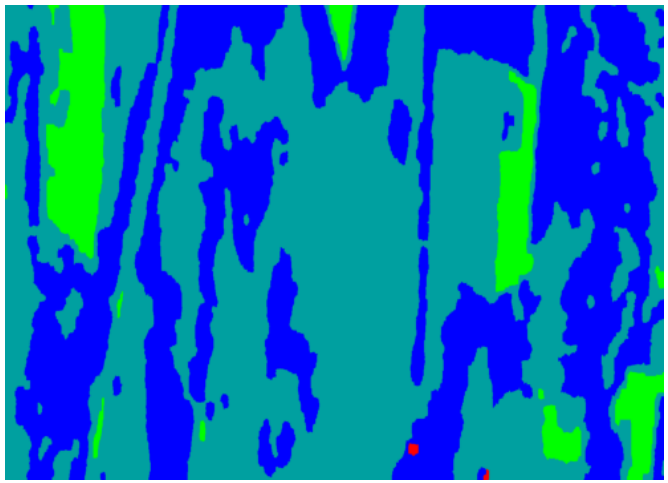
- Extract standard PolSAR features (span, entropy, anisotropy, etc.)
- First prediction by kNN
- Label smoothing by MRF + Alpha expansion



Original data



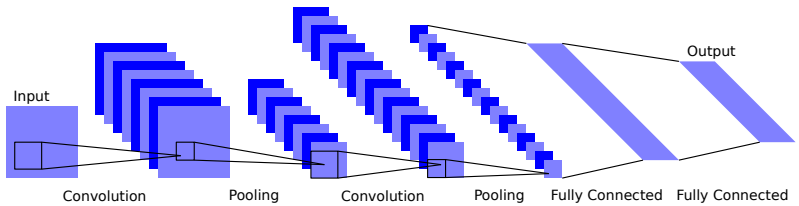
Initial estimate



After label smoothing

Overview

1. Random Forests
   - Fusion
   - Node Tests
   - Interpretation
   - Application Tips
2. Graphical Models
   - Motivation
   - MRF
   - Application
   - Example
3. ConvNets
   - MLP to ConvNet
   - Convolution
   - Architectures
   - Auto Encoder
   - Frameworks



Recap MLP

MLPs

- Provide a mapping from $\mathcal{X} \rightarrow \mathcal{Y}$, i.e. from a features space (usually $\mathcal{X} \equiv \mathbb{R}^n$) to a label space $\mathcal{Y}$
- Are based on concatenation of "simple" functions that depend on parameters (i.e. weights)
- Are optimized by gradient descent (and its modern extensions)

Recap MLP

MLPs

- Provide a mapping from $\mathcal{X} \rightarrow \mathcal{Y}$, i.e. from a features space (usually $\mathcal{X} \equiv \mathbb{R}^n$) to a label space $\mathcal{Y}$
- Are based on concatenation of "simple" functions that depend on parameters (i.e. weights)
- Are optimized by gradient descent (and its modern extensions)
- Work great, BUT:

ConvNets and Deep Learning

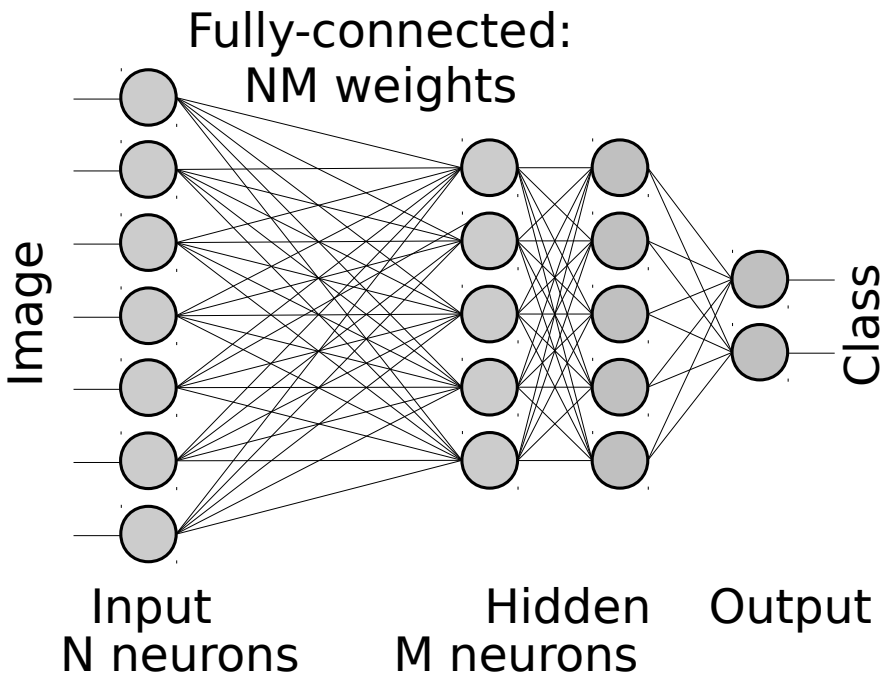
- Used by
  - Facebook: Automatic tagging
  - Google: Photo search
  - Amazon: Recommendations
  - Pinterest: Home feed personalization
  - Instagram: Search
- Buzzwords
  - Deep Learning, Deep Networks
  - Convolutional Neural Networks (CNNs), Convolutional Networks (ConvNets)
  - Note: There are more Deep Networks / Deep Learning approaches than ConvNets

ConvNets and Deep Learning

"CNNs are inspired by biological principles in the visual cortex."

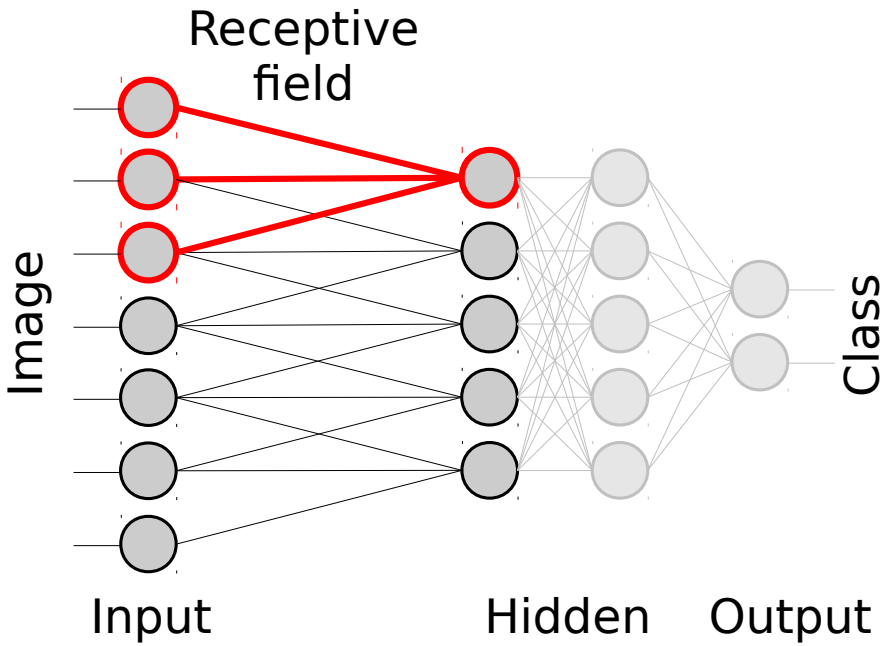
- Small regions of cells sensitive to specific regions within the visual field.
- 1962, Hubel and Wiesel
  - Neuronal cells fire only in the presence of certain structures e.g. edges of a specific orientation
  - Organized in columns
- Good selling point, BUT:
  - Extracting image features is neither new, nor the main point of ConvNets
  - Training works very differently

From FullyConnected (MLP) to Convolution (ConvNet)



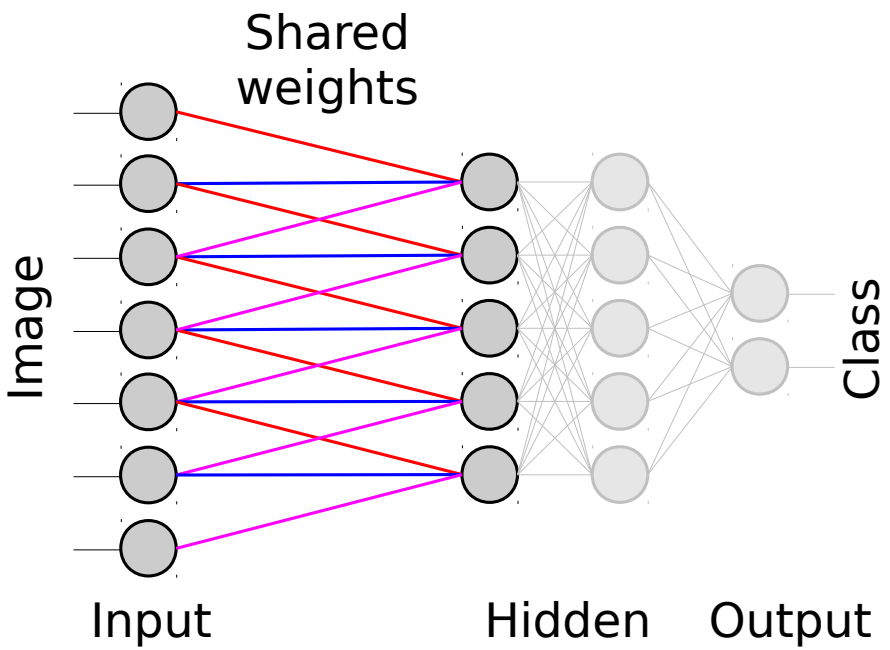
- Multiple layers of units
- All-to-all connection between two adjacent layers
- No lateral connections
- A tremendous amount of parameters in case of images
  $\rightarrow$ Untrainable

From FullyConnected (MLP) to Convolution (ConvNet)



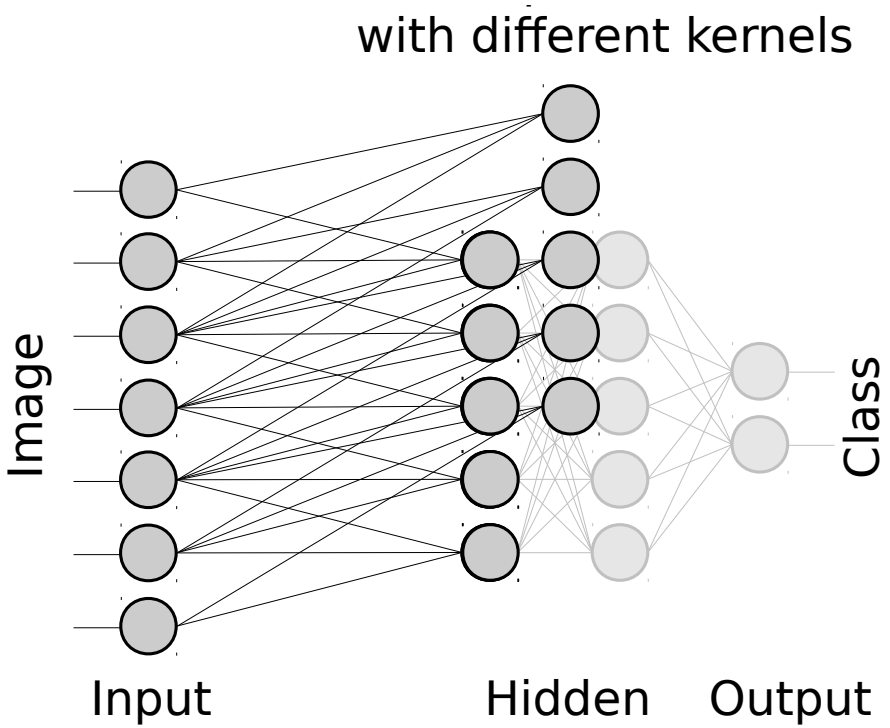
- Set most weights to zero and thus delete most connections and decrease parameters.

From FullyConnected (MLP) to Convolution (ConvNet)



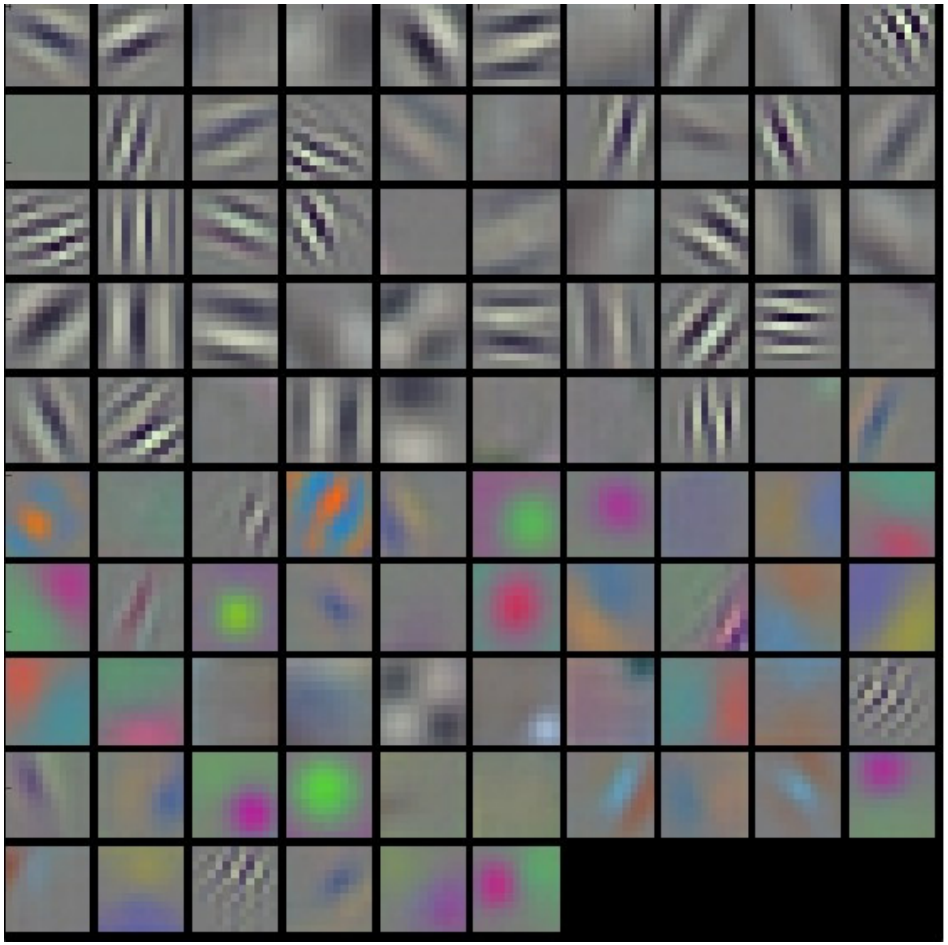
- Set most weights to zero and thus delete most connections and decrease parameters.
- Use same values for weights of different neurons within a layer.
- The multiplication of the input with identical weights for different neurons corresponds to a convolution.
- The kernel of this convolution is automatically learned.

From FullyConnected (MLP) to Convolution (ConvNet)



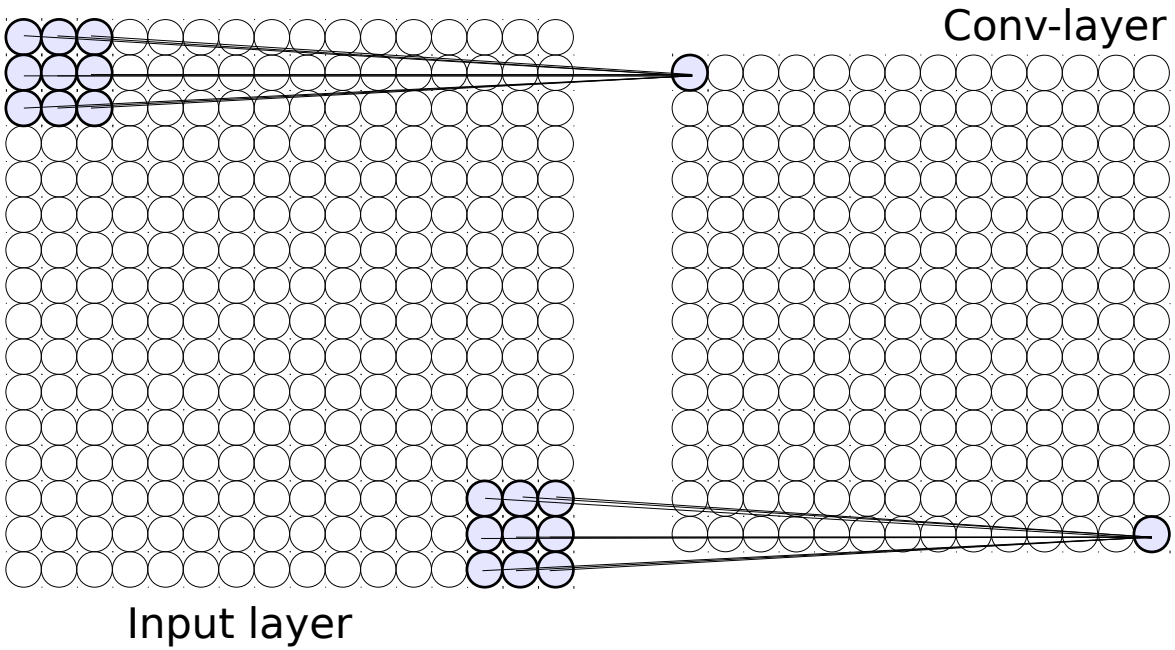
- Set most weights to zero and thus delete most connections and decrease parameters.
- Use same values for weights of different neurons within a layer.
- The multiplication of the input with identical weights for different neurons corresponds to a convolution.
- The kernel of this convolution is automatically learned.
- Use multiple convolutional layers to enable different kernels to be learned.

Example of First Level Filters

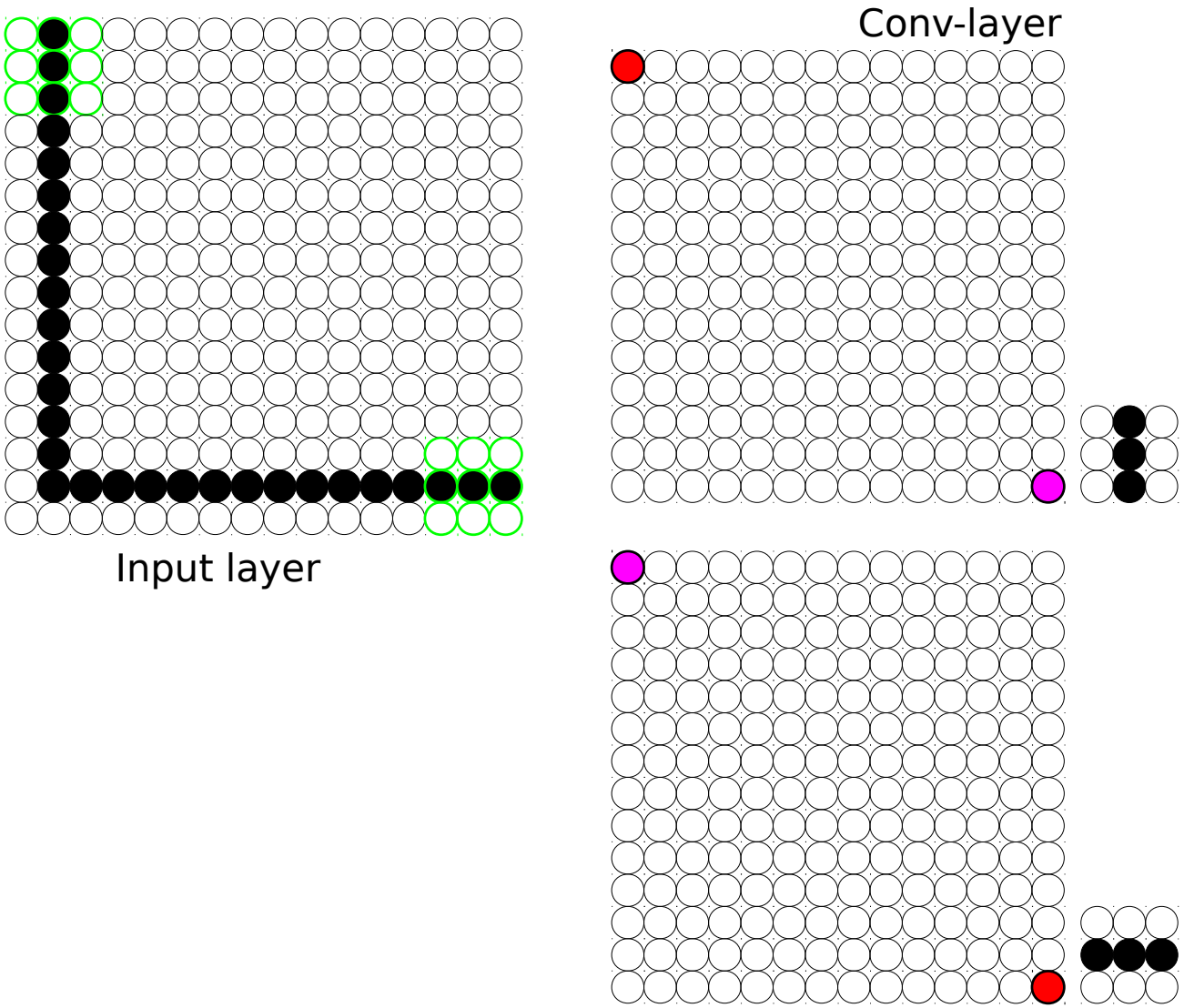


- Learned kernels of first convolutional layer of a ConvNet (AlexNet).
- Correspond mostly to edges and corners of different orientations.
- Note: Grouping is caused by network architecture (two independent streams were used to handle the large amount of data).

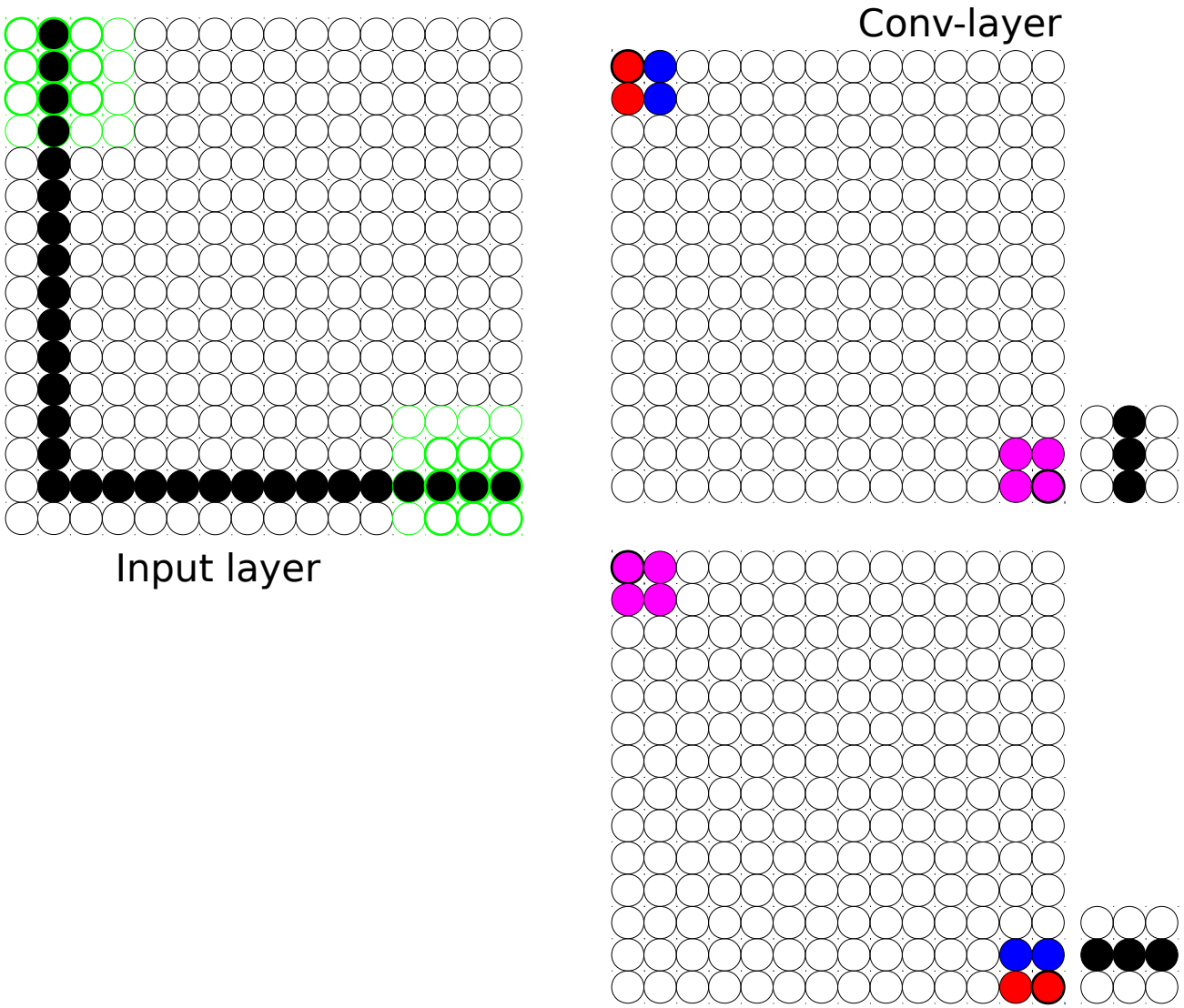
Pooling



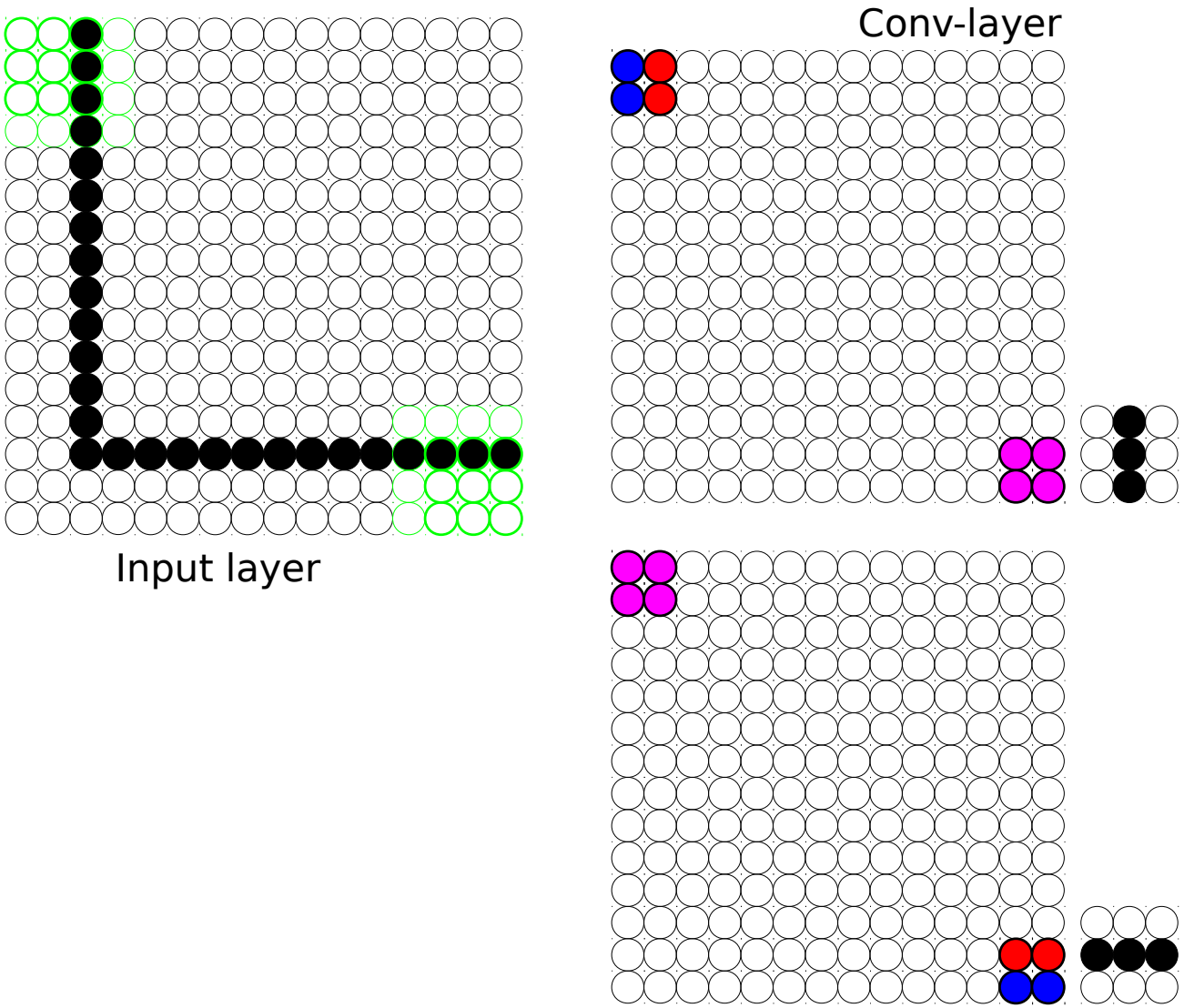
Pooling



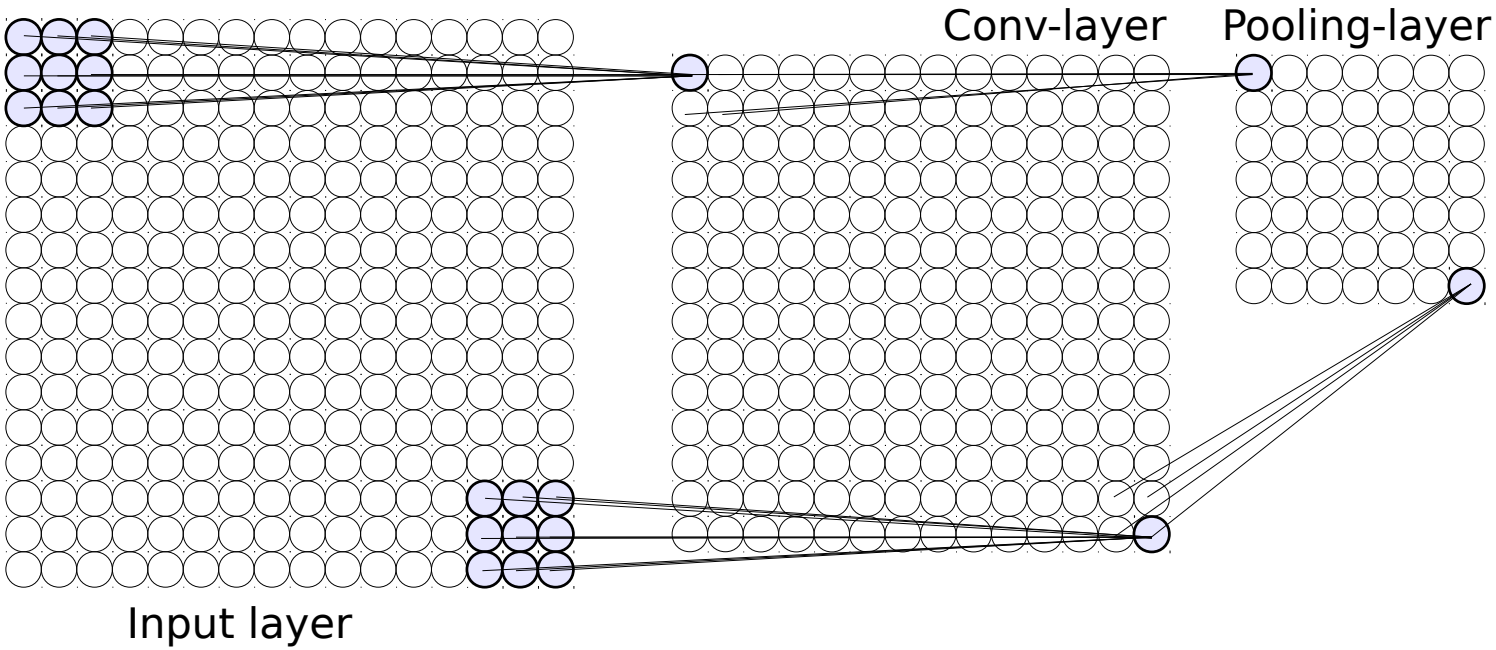
Pooling



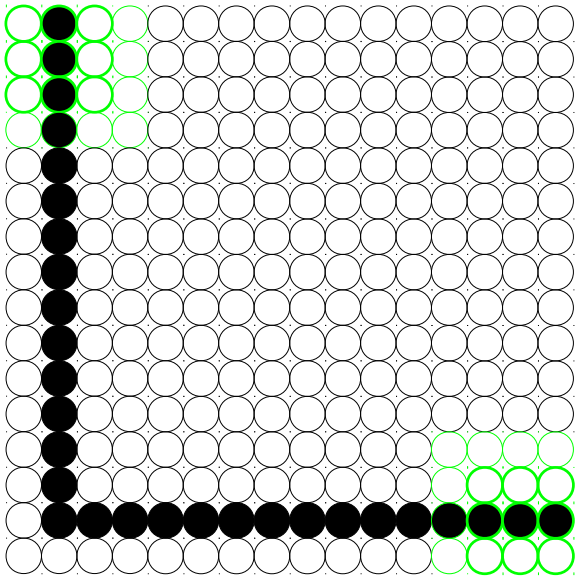
Pooling



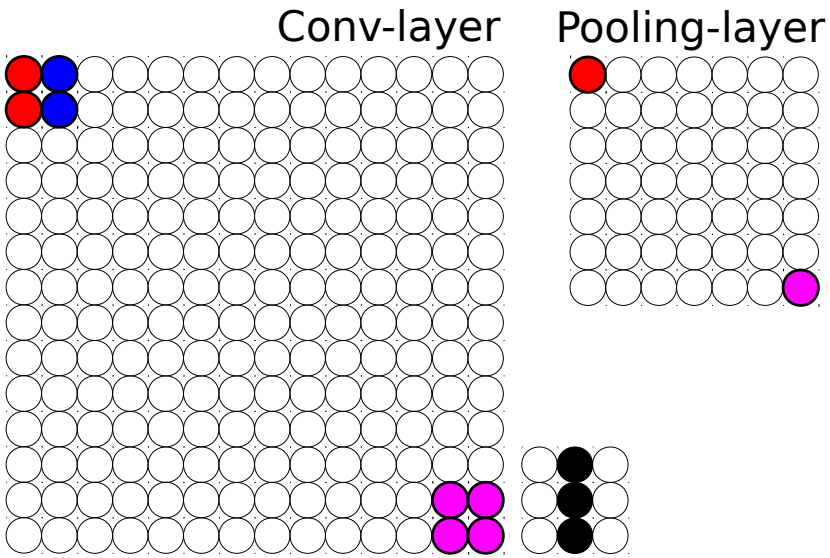
Pooling

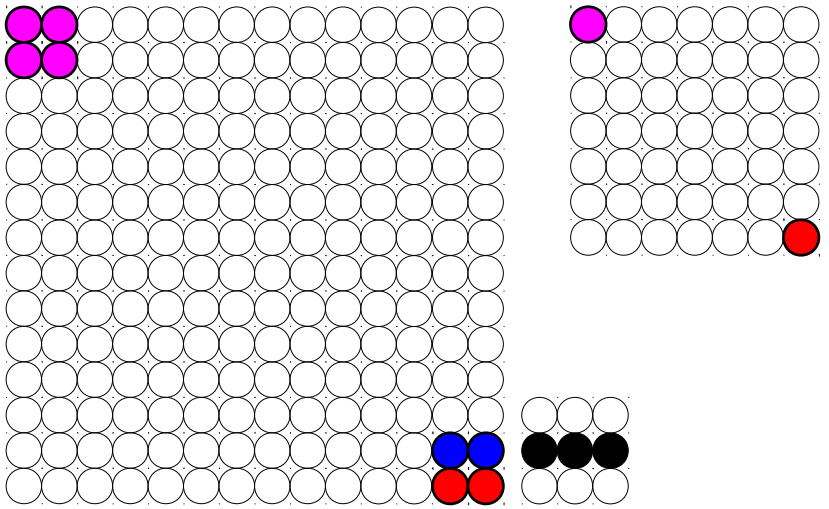


Pooling

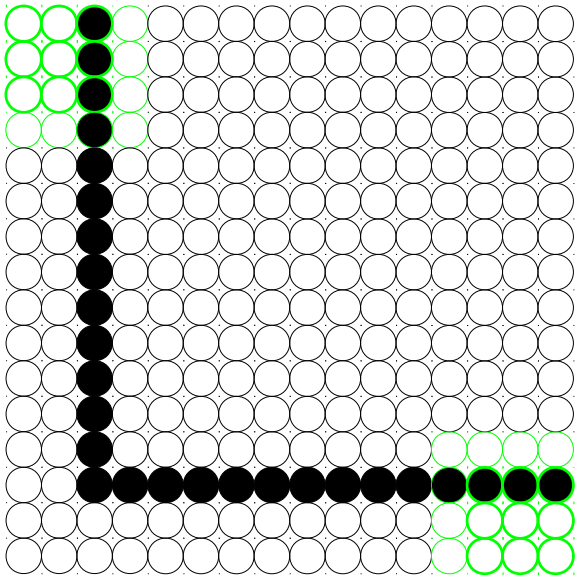


Input layer

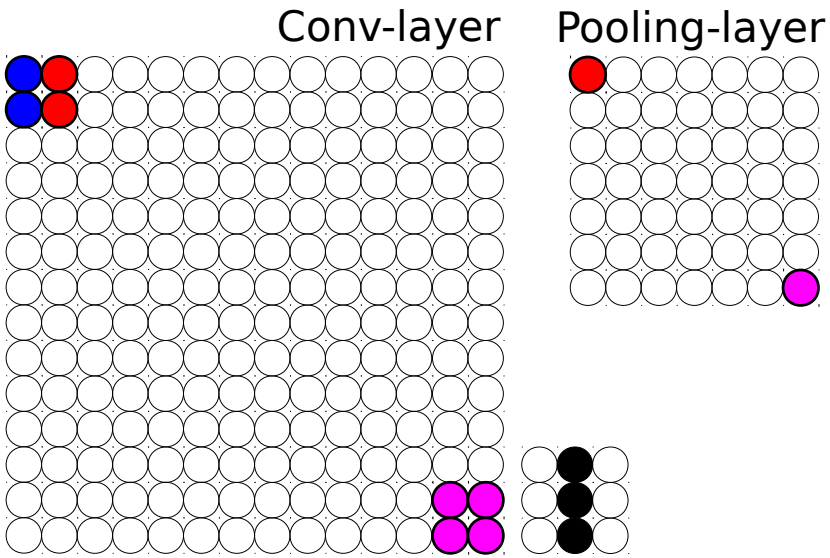


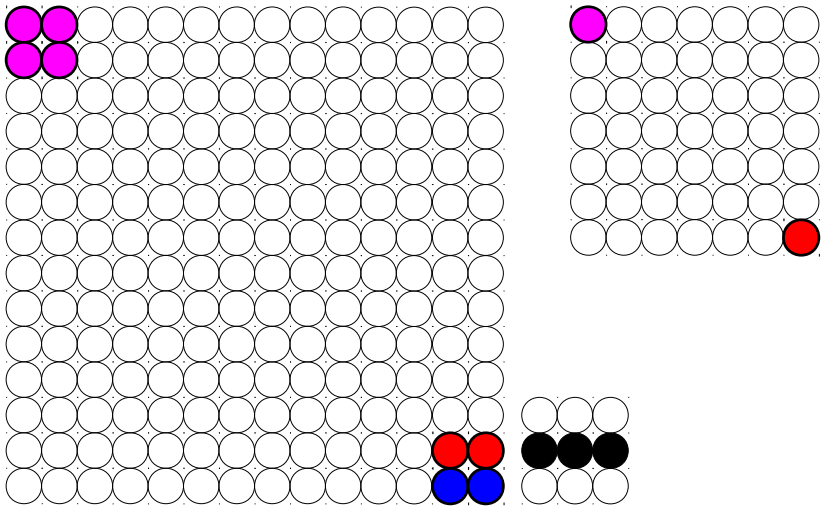


Pooling

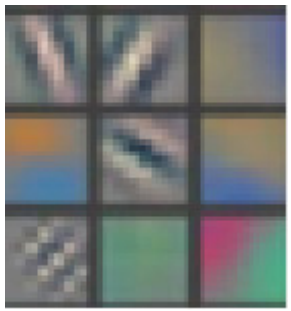


Input layer





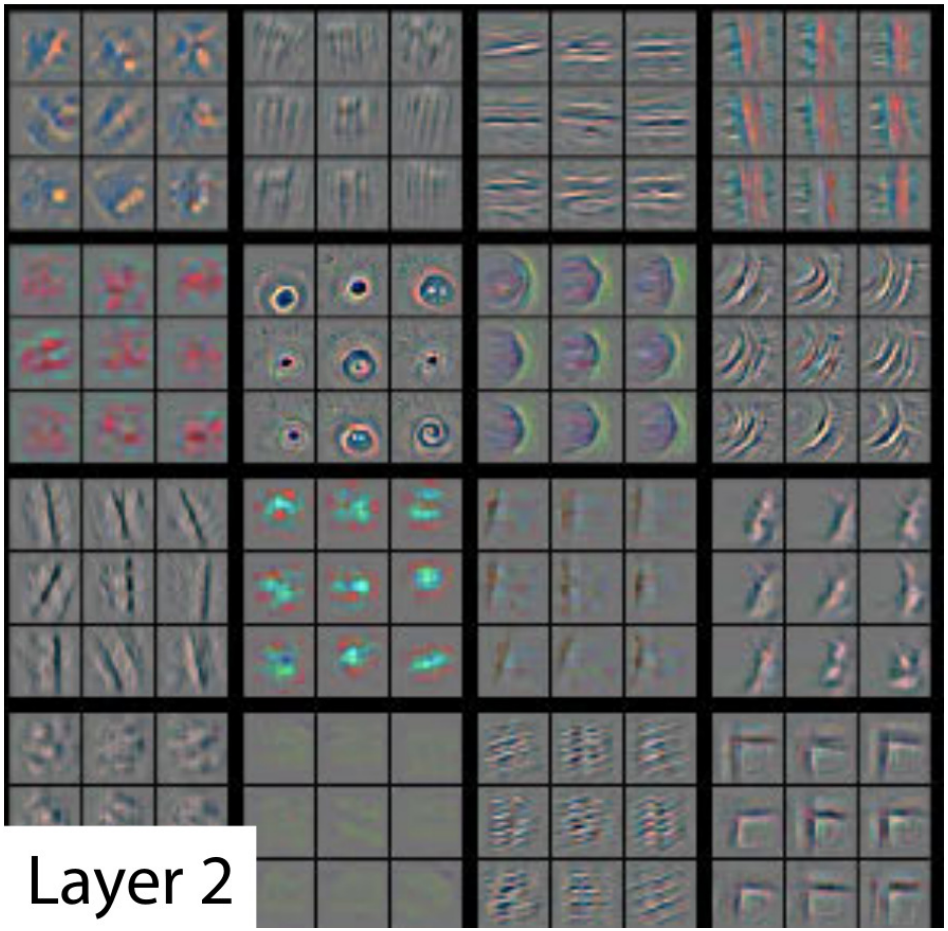
Example of Higher Level Filters



Layer 1

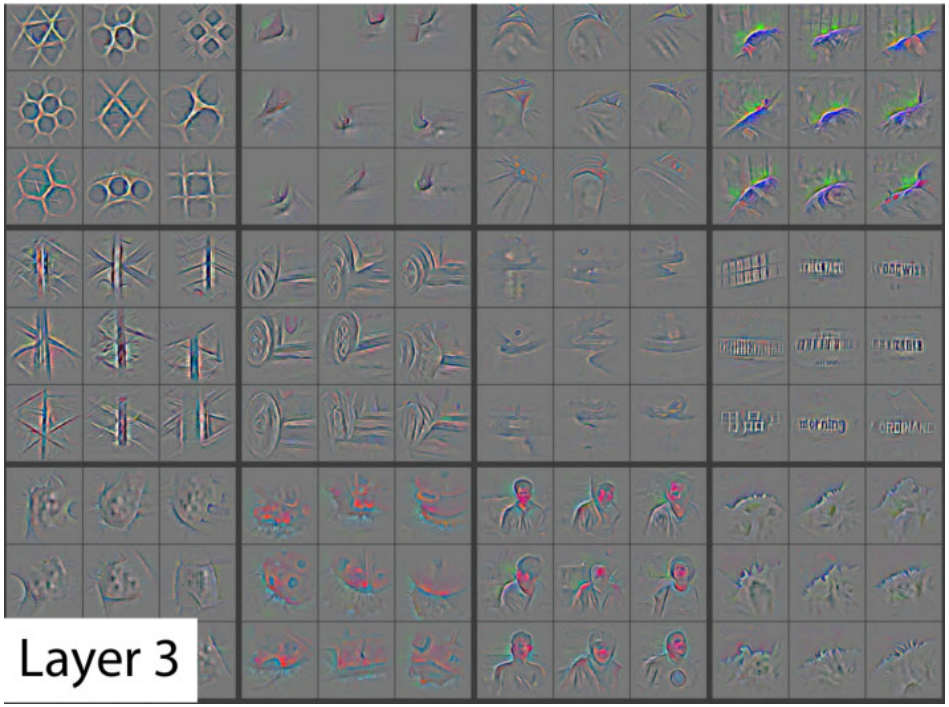
- Top nine activations in feature maps
- Projected to pixel space using a deconvolutional network
- Reconstructed patterns that cause high activations
- Note: Images taken from [Zeiler and Fergus, 2013].

Example of Higher Level Filters



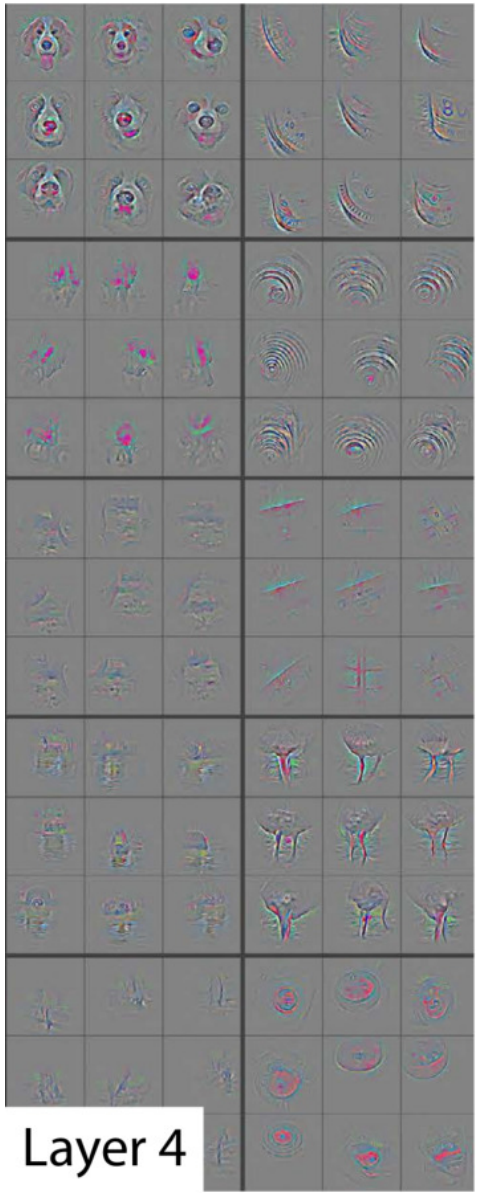
- Top nine activations in feature maps
- Projected to pixel space using a deconvolutional network
- Reconstructed patterns that cause high activations
- Note: Images taken from [Zeiler and Fergus, 2013].

Example of Higher Level Filters



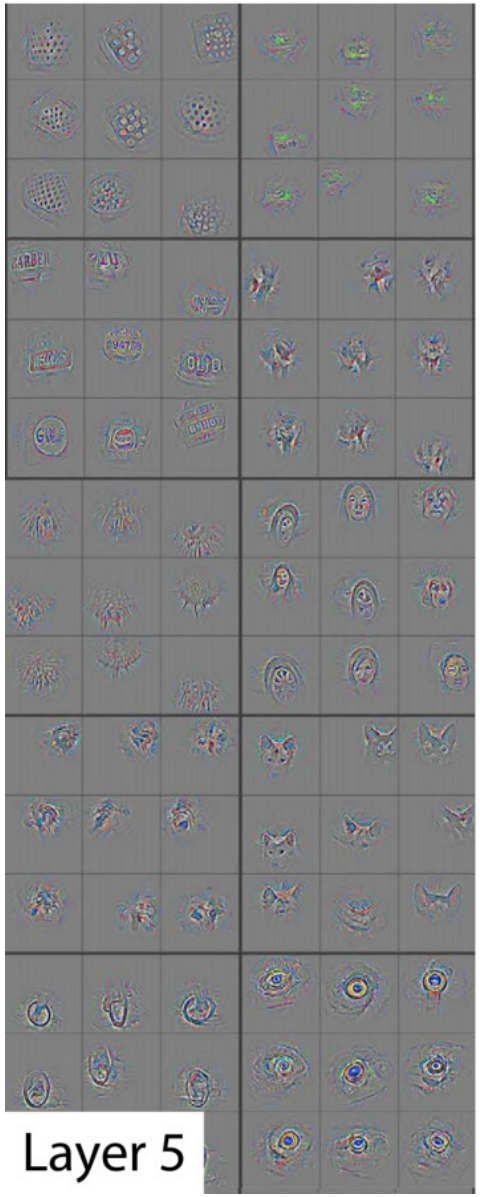
- Top nine activations in feature maps
- Projected to pixel space using a deconvolutional network
- Reconstructed patterns that cause high activations
- Note: Images taken from [Zeiler and Fergus, 2013].

Example of Higher Level Filters



- Top nine activations in feature maps
- Projected to pixel space using a deconvolutional network
- Reconstructed patterns that cause high activations
- Note: Images taken from [Zeiler and Fergus, 2013].

Example of Higher Level Filters



- Top nine activations in feature maps
- Projected to pixel space using a deconvolutional network
- Reconstructed patterns that cause high activations
- Note: Images taken from [Zeiler and Fergus, 2013].

Architectures

LeNet (1998)

- One of the first successful applications of ConvNets
- Digital digit / character recognition

Architectures

LeNet (1998)

- One of the first successful applications of ConvNets
- Digital digit / character recognition

AlexNet (2012)

- Started the hype
- Similar to LeNet, but deeper and bigger
- Stacked conv-layers
- Image classification (ImageNet Large-Scale Visual Recognition Challenge)
- Trained on 15 million annotated images from over 22,000 categories
- Trained on two GTX 580 GPUs for five to six days

Architectures

LeNet (1998)

- One of the first successful applications of ConvNets
- Digital digit / character recognition

AlexNet (2012)

- Started the hype
- Similar to LeNet, but deeper and bigger
- Stacked conv-layers
- Image classification (ImageNet Large-Scale Visual Recognition Challenge)
- Trained on 15 million annotated images from over 22,000 categories
- Trained on two GTX 580 GPUs for five to six days

ZF Net (2013)

- Similar to AlexNet
- Trained on 1.3 million annotated images
- Trained on a GTX 580 GPU for twelve days

Architectures

VGG Net (2014)

- Simple and deep: Only 3x3 filters and 2x2 pooling
- Stacked conv-layers to increase effective receptive field size
- Used Caffe toolbox
- Trained on 4 Nvidia Titan Black GPUs for two to three weeks

Architectures

VGG Net (2014)

- Simple and deep: Only 3x3 filters and 2x2 pooling
- Stacked conv-layers to increase effective receptive field size
- Used Caffe toolbox
- Trained on 4 Nvidia Titan Black GPUs for two to three weeks

GoogLeNet (2015)

- 22 layers
- Proposed inception module: Running multiple filter operations in parallel
- 12x fewer parameters than AlexNet
- Trained on multiple high-end GPUs for a week

Architectures

VGG Net (2014)

- Simple and deep: Only 3x3 filters and 2x2 pooling
- Stacked conv-layers to increase effective receptive field size
- Used Caffe toolbox
- Trained on 4 Nvidia Titan Black GPUs for two to three weeks

GoogLeNet (2015)

- 22 layers
- Proposed inception module: Running multiple filter operations in parallel
- 12x fewer parameters than AlexNet
- Trained on multiple high-end GPUs for a week

Microsoft ResNet (2015)

- 152 layers
- Trained on an 8 GPUs for two to three weeks
- 3.6% error on ImageNet LSVRC (AlexNet: 15.4%)

Common Architectures and Tricks

- Designing good architecture somewhat tricky
- Some designs, or parts of designs, exist that work well
- Usually a good idea to look at papers of common architectures
  - Most of the time, at least some intuition or motivation for choice of layers

Network in a Network

- Alternate between actual conv layers, and conv layers of size $1x1$
- Use the (per pixel) FC layers to compress (reduce channels)
  - Next (actual) convolution faster
  - Deeper network, but less parameters

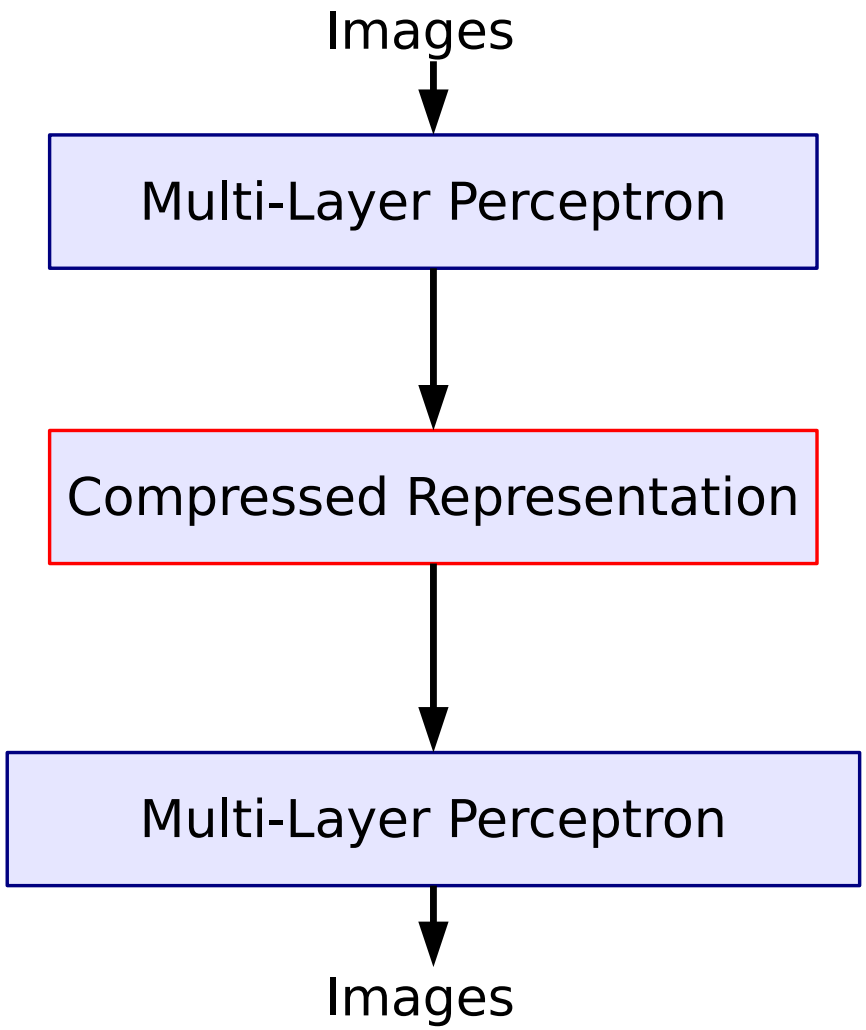
ResNets and Shortcuts

- For deep network, good gradient propagation super important
- ResNet:
  - Concatenation of blocks
  - Each block performs usual operations (conv, relu) ...
  - Result gets added to block's input
  - Neutral/Zero case: Signal passes through unaltered
- Shortcuts:
  - Idea quite similar to ResNets
  - Signal can bypass some layers
  - Signal/Gradient can jump and progress better in deep networks

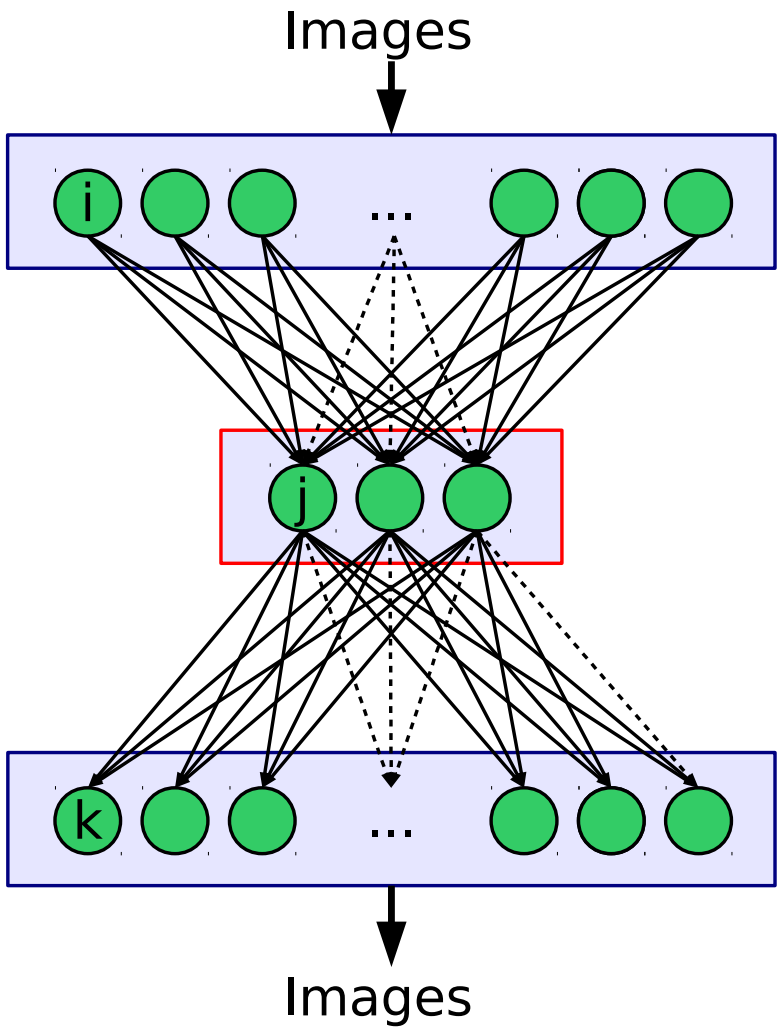
Inception Module

- Used by Google
- Multiple versions
- Compilation of multiple ideas
- Network in a Network
- Use of small filters (only $3x3$)
- Using two $3x3$ filters same receptive field size as one $5x5$ filter
  - But less parameters

(Convolutional) Auto Encoder

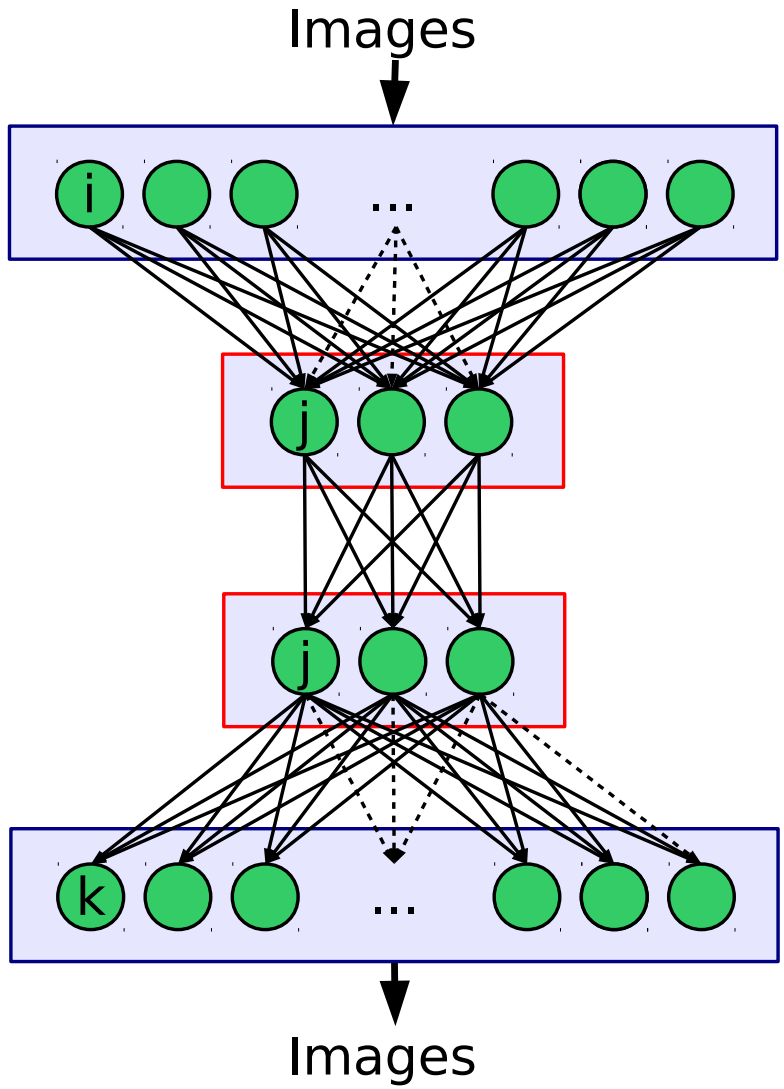


(Convolutional) Auto Encoder



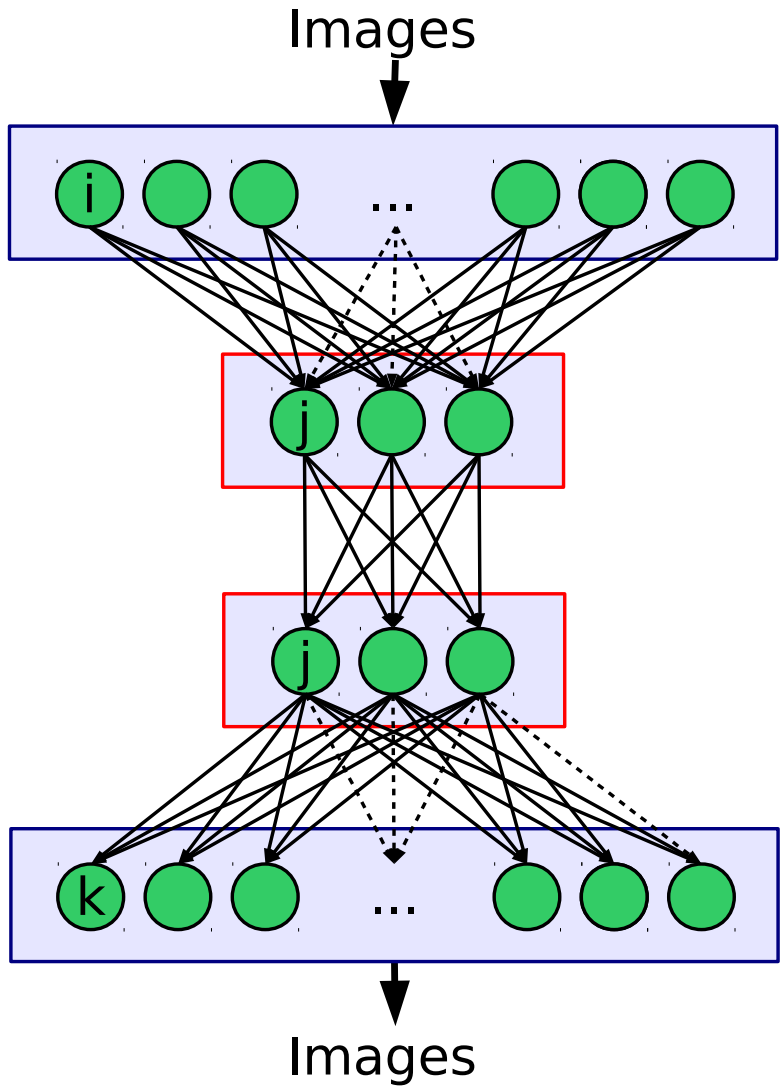
(Convolutional) Auto Encoder

- Stacked Autoencoder



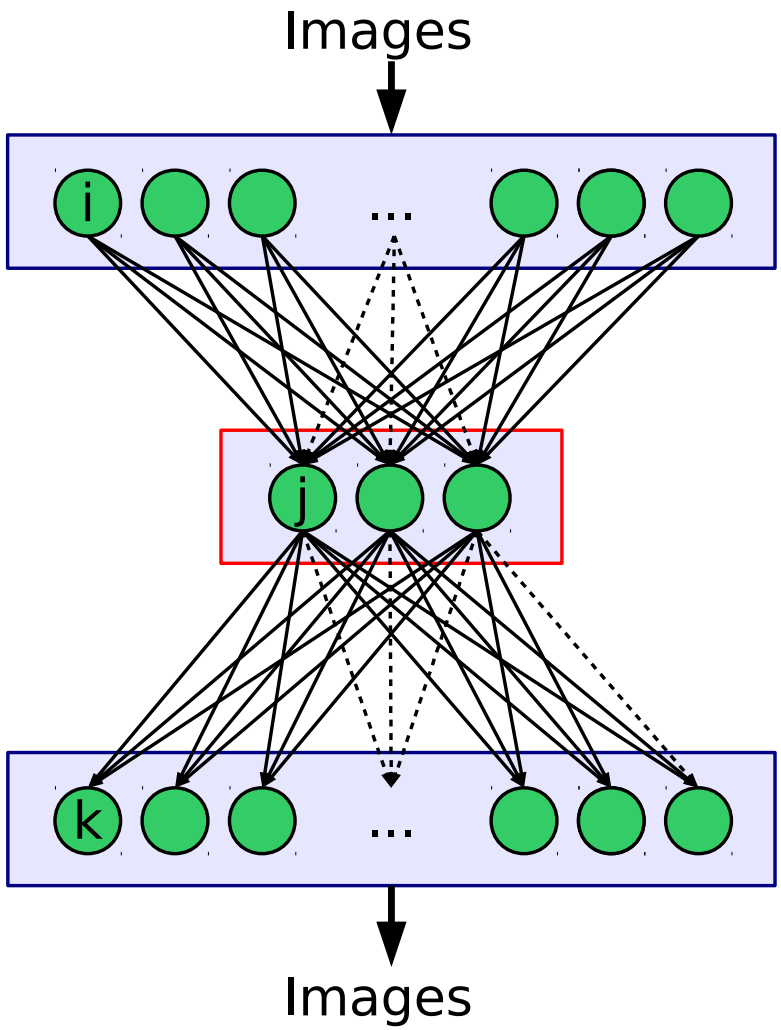
(Convolutional) Auto Encoder

- Stacked Autoencoder
- Problem: Vanishing gradients



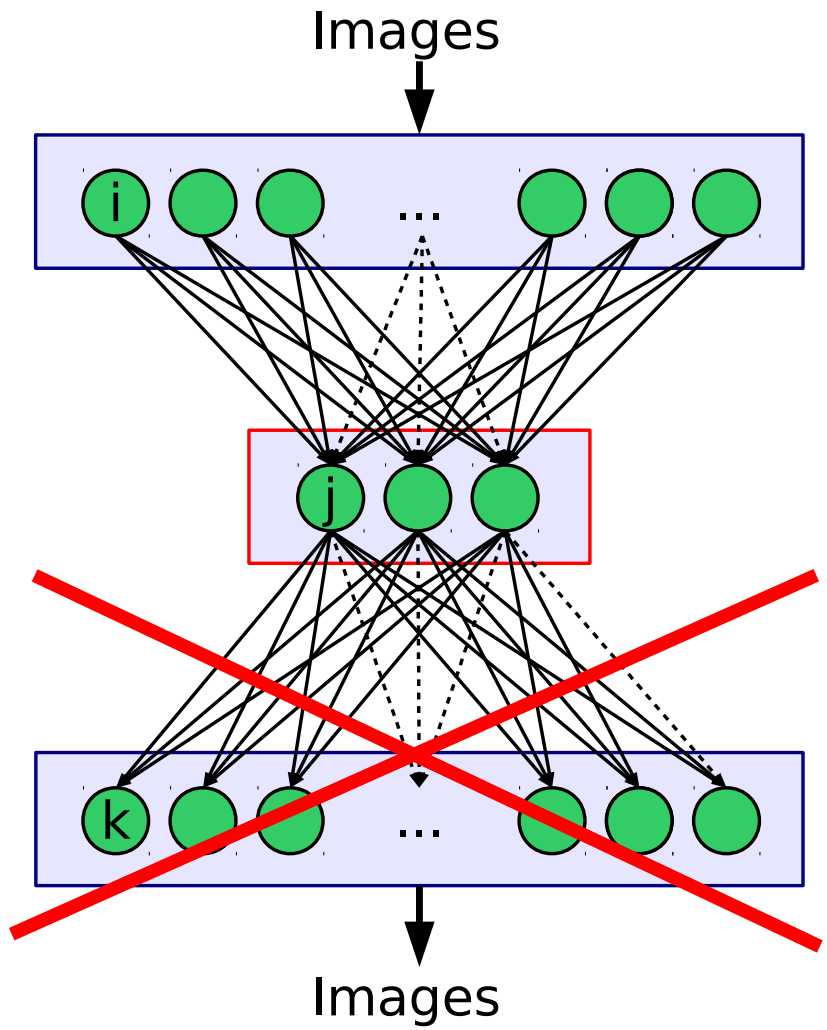
(Convolutional) Auto Encoder

- Stacked Autoencoder
- Problem: Vanishing gradients
- Solution: Pre-training



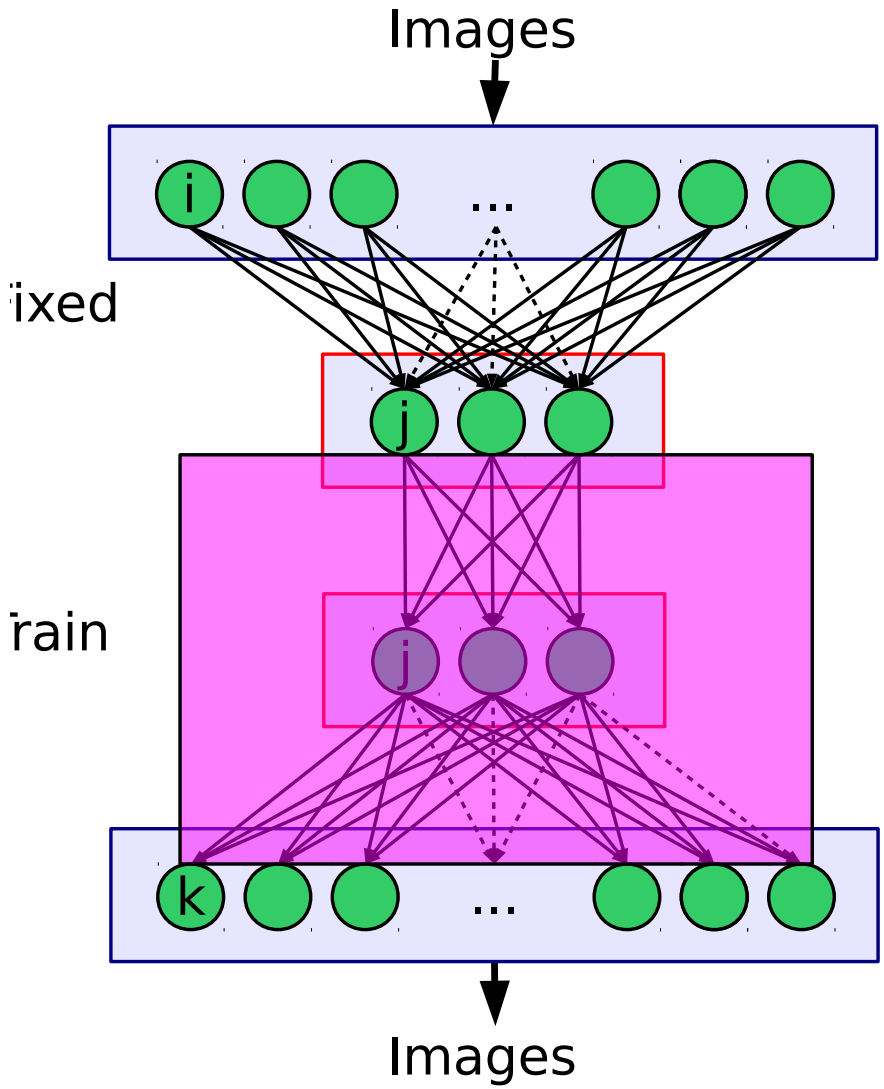
(Convolutional) Auto Encoder

- Stacked Autoencoder
- Problem: Vanishing gradients
- Solution: Pre-training



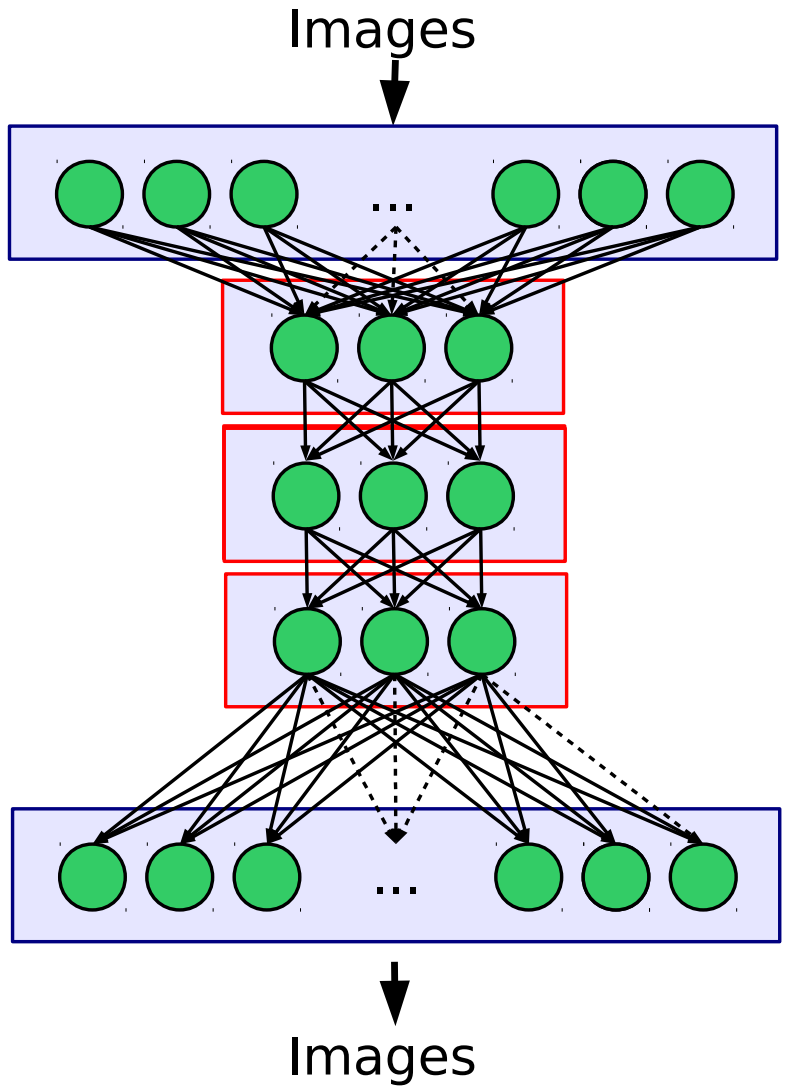
(Convolutional) Auto Encoder

- Stacked Autoencoder
- Problem: Vanishing gradients
- Solution: Pre-training



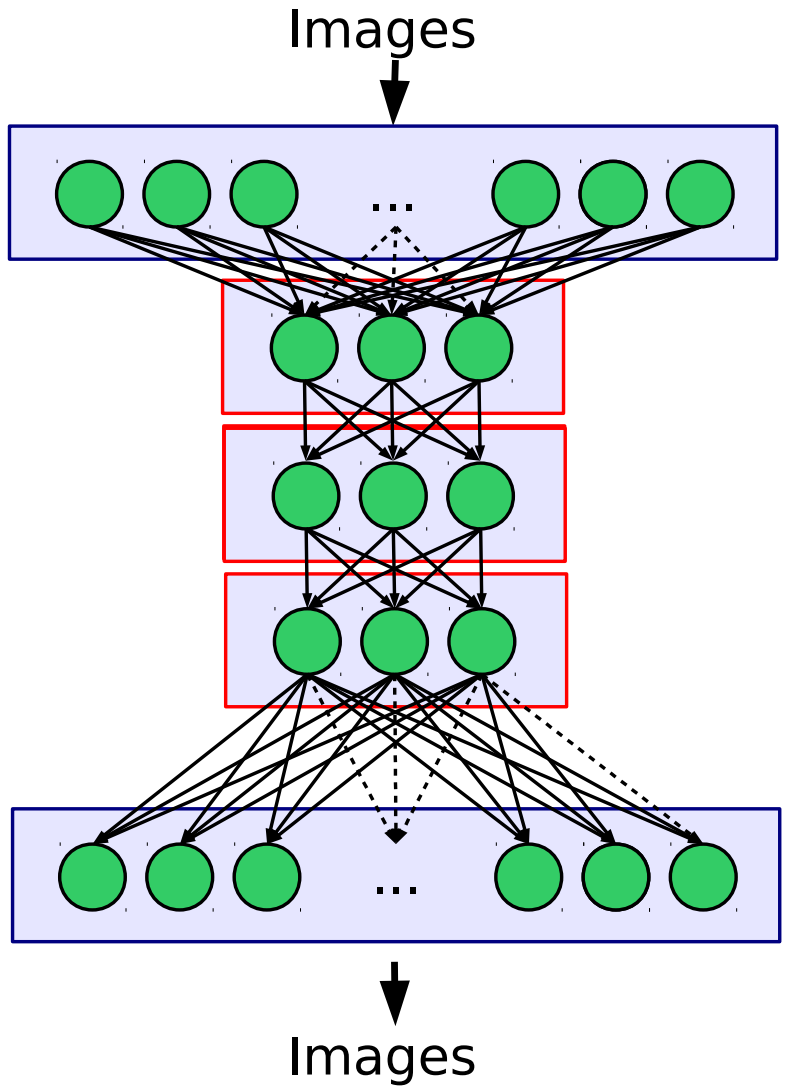
(Convolutional) Auto Encoder

- Stacked Autoencoder
- Problem: Vanishing gradients
- Solution: Pre-training



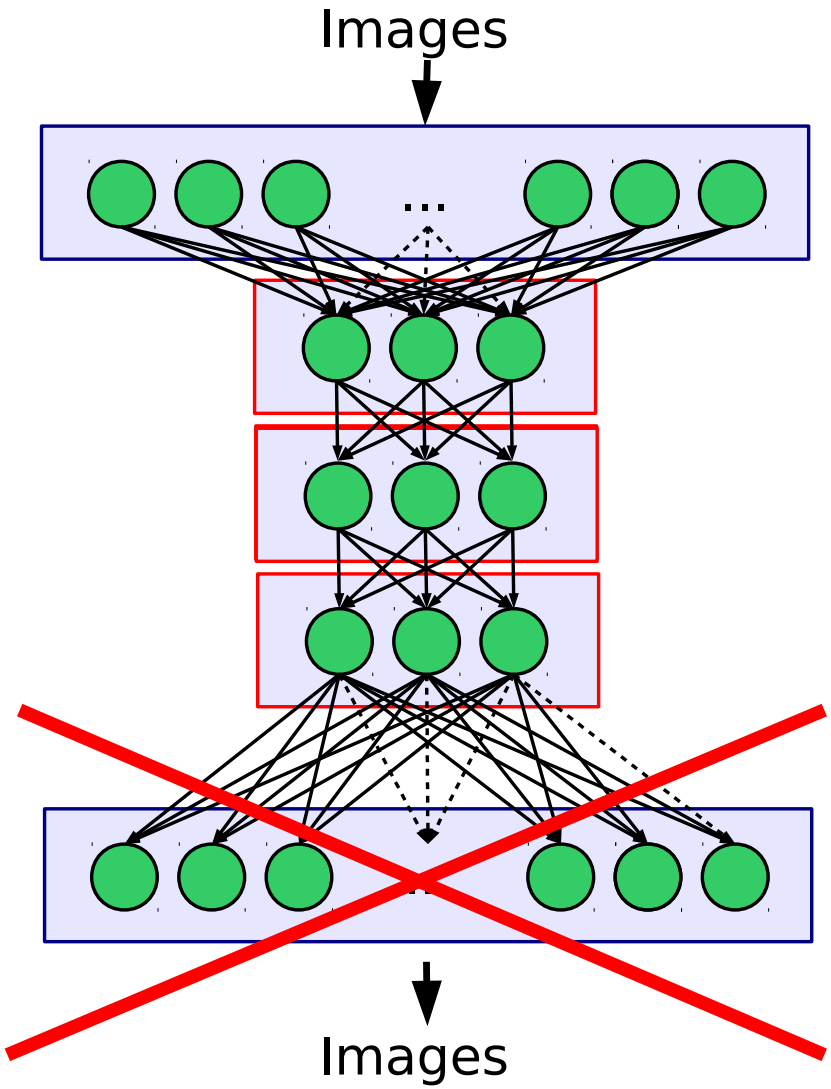
(Convolutional) Auto Encoder

- Stacked Autoencoder
- Problem: Vanishing gradients
- Solution: Pre-training
- Application: Deep Learning



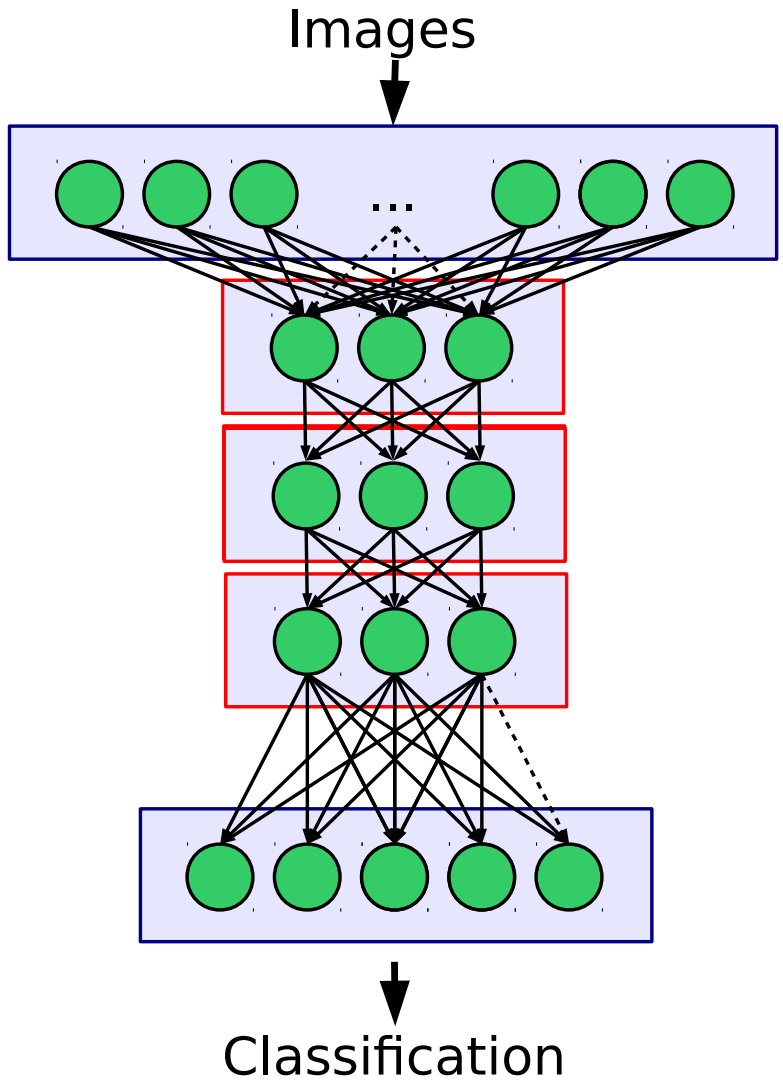
(Convolutional) Auto Encoder

- Stacked Autoencoder
- Problem: Vanishing gradients
- Solution: Pre-training
- Application: Deep Learning



(Convolutional) Auto Encoder

- Stacked Autoencoder
- Problem: Vanishing gradients
- Solution: Pre-training
  $\rightarrow$ Learn "reasonable" features from unlabeled data
- Application: Deep Learning
  $\rightarrow$ Supervised learning (via Backpropagation) only as refinement



Frameworks

- Implementing fast, multi-channel convolutions just as hard as implementing fast matrix multiplications
- *Use existing tools!*
  - Caffe
  - Tensorflow
  - Torch
- For larger datasets you want to use a (good) GPU!

Caffe

Caffe

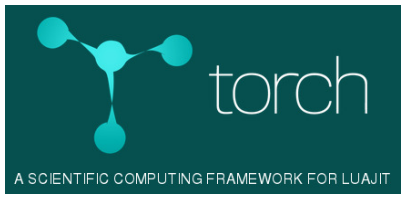
Deep learning framework by BAIR

- Started by Yangqing Jia at UC Berkeley
- Maintained by Berkeley AI Research and many contributers
- Backend in C++, frontends for Python and Matlab
- `http://caffe.berkeleyvision.org/`
- `https://github.com/BVLC/caffe`
- Version 2 now available

Tensorflow

- Developed by Google Brain team
- Python frontend
- `https://www.tensorflow.org/`
- `https://github.com/tensorflow`

Torch



- Lua frontend
- `http://torch.ch/`
- `https://github.com/torch/torch7`

DL Example

Exercise: DL

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239.

Gall, J., Yao, A., Razavi, N., Gool, L. V., and Lempitsky, V. (2011). Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202.

Hänsch, R. (2014). *Generic object categorization in PolSAR images - and beyond*. PhD thesis, TU Berlin.

R. Hänsch, O. H. (2015a). Feature-independent classification of hyperspectral images by projection-based random forests. In *WHISPERS 2015*.

R. Hänsch, O. H. (2015b). Performance assessment and interpretation of random forests by three-dimensional visualizations. In *IVAPP 2015*.

Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

I