Fast Edge Detection Using Structured Forests
by Piotr Dollar, C. Lawrence Zitnick (PAMI 2015)

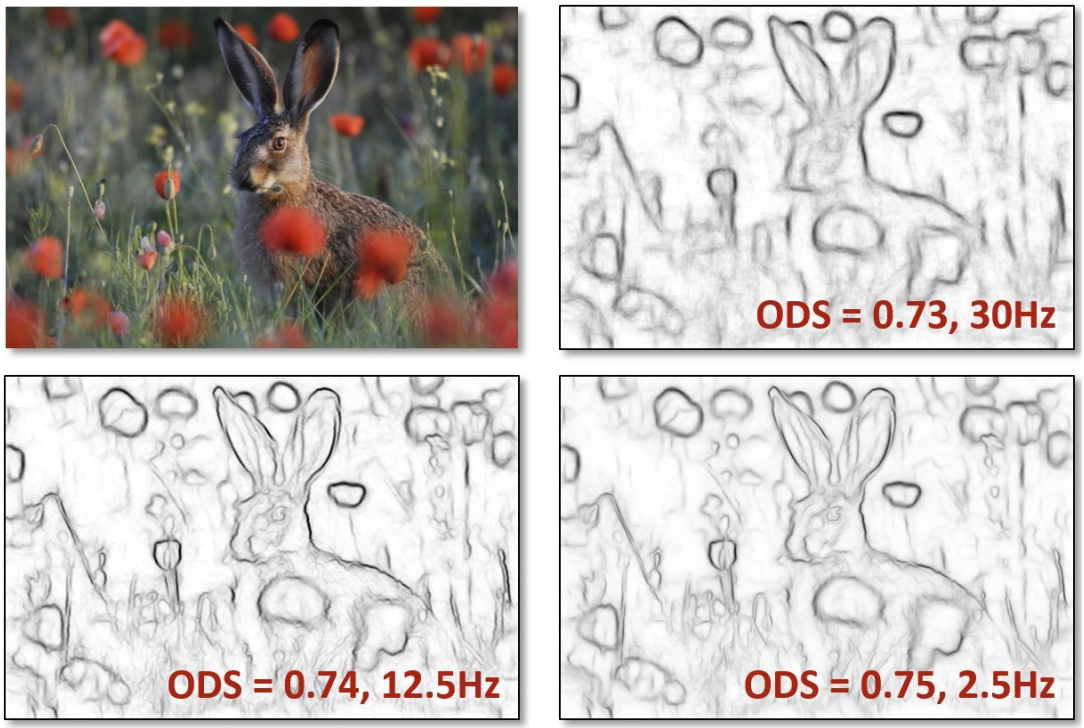
Vojtěch Cvrček

Reading Group Presentation

June 6, 2018

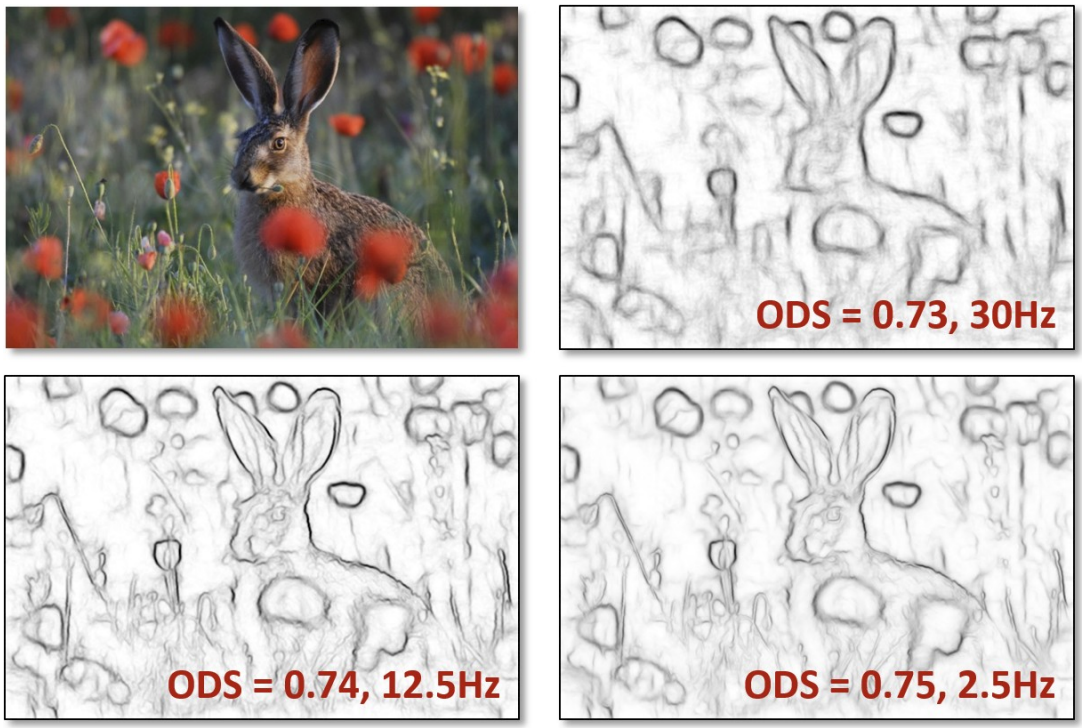
Problem Definition

- given a set of training images and corresponding segmentation masks

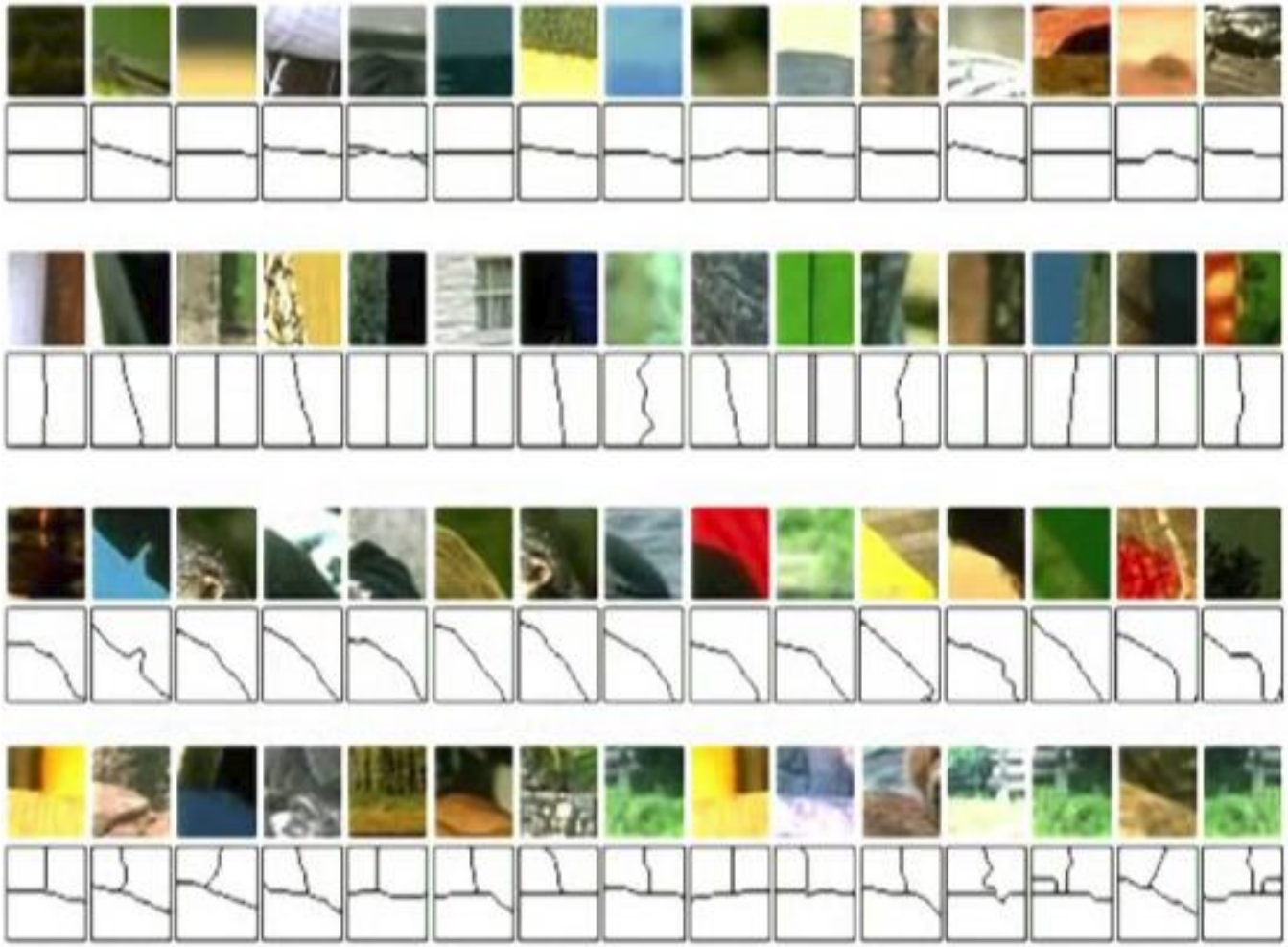


Problem Definition

- given a set of training images and corresponding segmentation masks
- predict an edge map for a novel input image



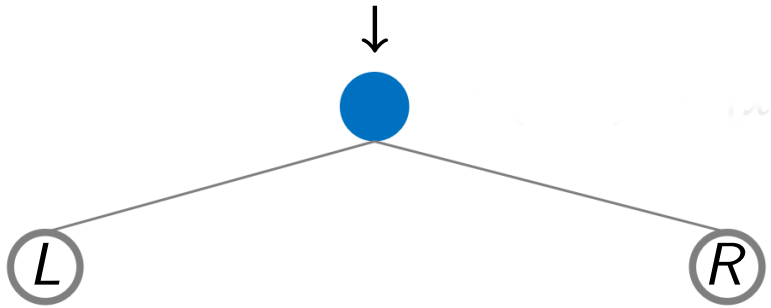
Edge structure



Random Forest - Single Tree

$f_t : \mathcal{X} \rightarrow \mathcal{Y}$

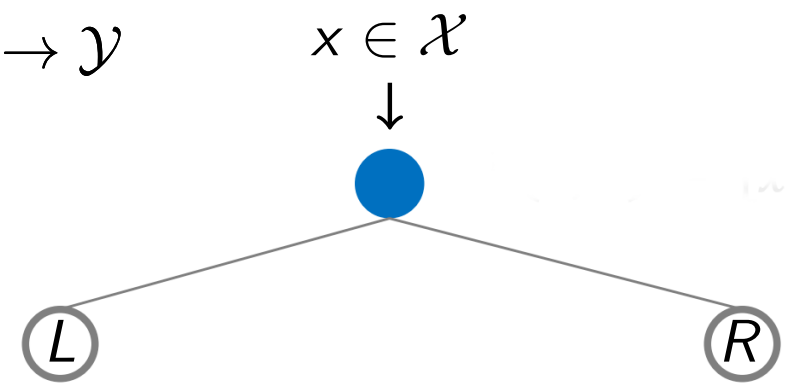
$x \in \mathcal{X}$



$h_{\theta_j} : \mathcal{X} \rightarrow \{L, R\}$

Random Forest - Single Tree

$f_t : \mathcal{X} \rightarrow \mathcal{Y}$



$$h^1_{\theta^1_j}(x) = x(k) < \tau$$

$$\theta^1_j = (k, \tau)$$

$$h^2_{\theta^2_j}(x) = x(k_1) - x(k_2) < \tau$$

$$\theta^2_j = (k_1, k_2, \tau)$$

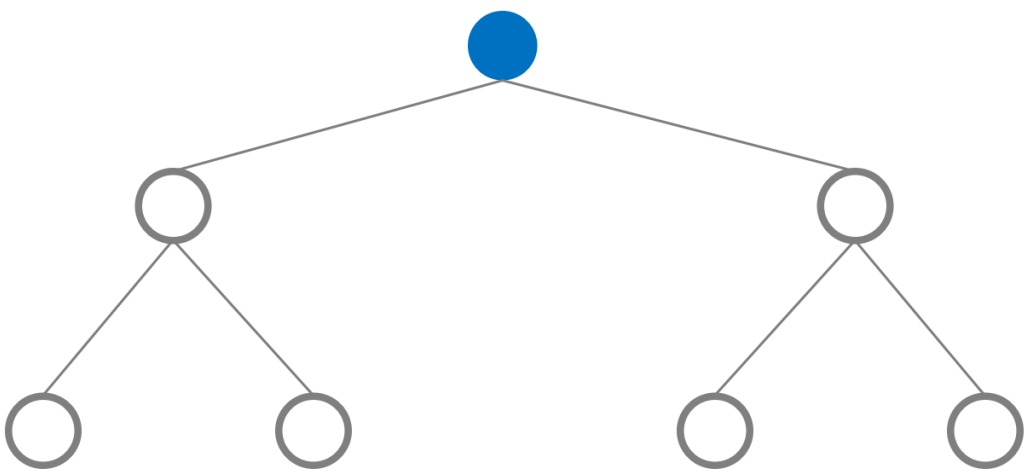
$$h_{\theta_j}(x) = \delta h^1_{\theta^1_j}(x) + (1 - \delta) h^2_{\theta^2_j}(x)$$

$$\delta \in \{0, 1\}$$

Random Forest - Single Tree

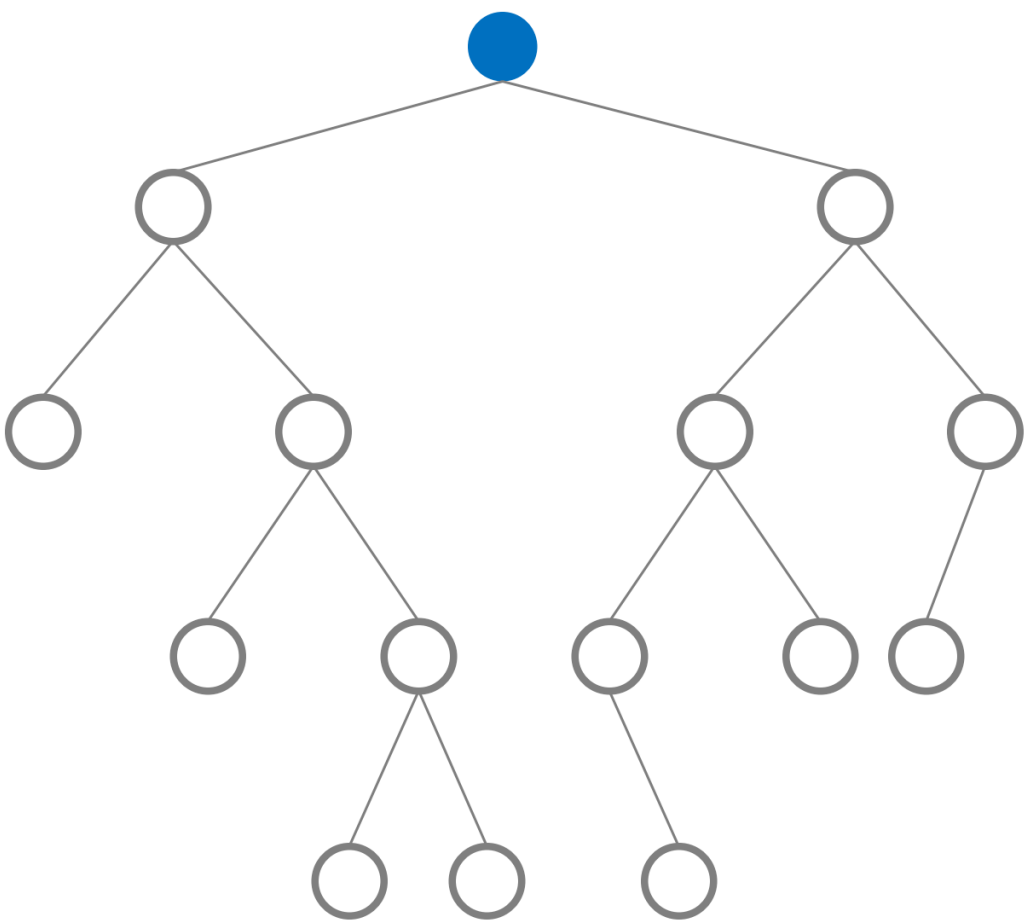
$f_t : \mathcal{X} \rightarrow \mathcal{Y}$

$x \in \mathcal{X}$



Random Forest - Single Tree

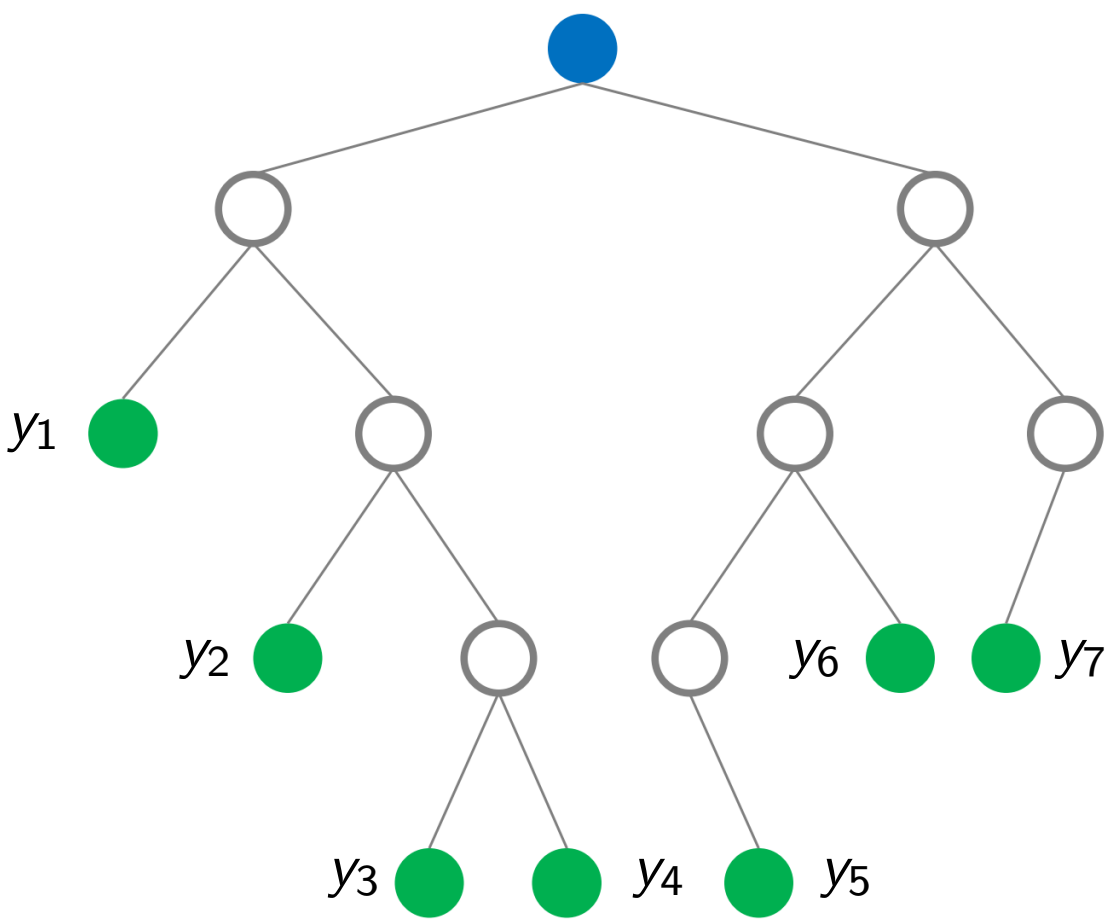
$f_t : \mathcal{X} \rightarrow \mathcal{Y}$ $x \in \mathcal{X}$



Random Forest - Single Tree

$f_t : \mathcal{X} \rightarrow \mathcal{Y}$

$x \in \mathcal{X}$

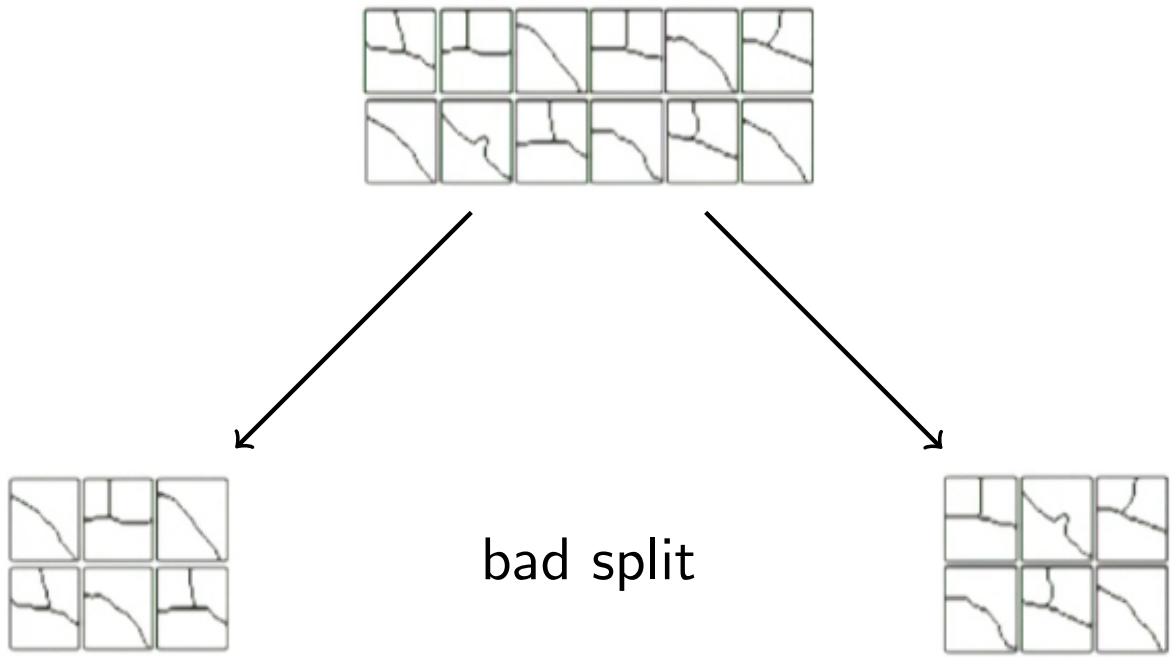


Training

- each tree is trained independently in a recursive manner

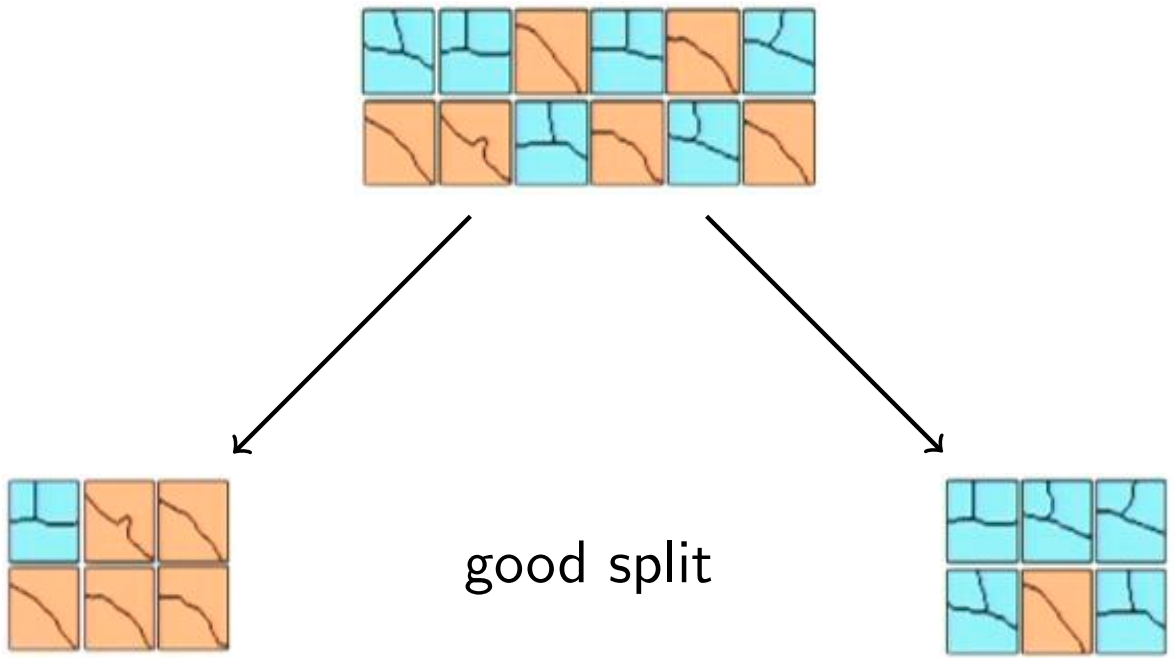
Training

- each tree is trained independently in a recursive manner
- for a given node j and training set $\mathcal{S}_j \subset \mathcal{X} \times \mathcal{Y}$, randomly sample parameters θ_j from parameters space



Training

- each tree is trained independently in a recursive manner
- for a given node j and training set $\mathcal{S}_j \subset \mathcal{X} \times \mathcal{Y}$, randomly sample parameters θ_j from parameters space
- Select θ_j resulting in a 'good' split of the data



Training - Information Gain Criterion

- *information gain criterion:*

$$I_j = I(\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R), \tag{1}$$

where $\mathcal{S}_j^L = \{(x, y) \in \mathcal{S}_j \mid h(x, \theta) = 0\}$, $\mathcal{S}_j^R = \mathcal{S}_j \setminus \mathcal{S}_j^L$ are splits

Training - Information Gain Criterion

- *information gain criterion:*

$$I_j = I(\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R), \tag{1}$$

  where $\mathcal{S}_j^L = \{(x, y) \in \mathcal{S}_j \mid h(x, \theta) = 0\}$, $\mathcal{S}_j^R = \mathcal{S}_j \setminus \mathcal{S}_j^L$ are splits
- $\theta_j = \text{argmax}_\theta \, I_j(\mathcal{S}_j, \theta)$

Training - Information Gain Criterion

- *information gain criterion:*

$$I_j = I(\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R), \tag{1}$$

  where $\mathcal{S}_j^L = \{(x, y) \in \mathcal{S}_j \mid h(x, \theta) = 0\}$, $\mathcal{S}_j^R = \mathcal{S}_j \setminus \mathcal{S}_j^L$ are splits
- $\theta_j = \text{argmax}_\theta \, I_j(\mathcal{S}_j, \theta)$
- for multiclass classification ($\mathcal{Y} \subset \mathbb{Z}$) the standard definition of information gain is:

$$I_j = H(\mathcal{S}_j) - \sum_{k \in \{L,R\}} \frac{|\mathcal{S}_j^k|}{|\mathcal{S}_j|} H(\mathcal{S}_j^k) \tag{2}$$

Training - Information Gain Criterion

- *information gain criterion:*

$$I_j = I(\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R), \tag{1}$$

  where $\mathcal{S}_j^L = \{(x, y) \in \mathcal{S}_j \mid h(x, \theta) = 0\}$, $\mathcal{S}_j^R = \mathcal{S}_j \setminus \mathcal{S}_j^L$ are splits
- $\theta_j = \text{argmax}_\theta \, I_j(\mathcal{S}_j, \theta)$
- for multiclass classification ($\mathcal{Y} \subset \mathbb{Z}$) the standard definition of information gain is:

$$I_j = H(\mathcal{S}_j) - \sum_{k \in \{L,R\}} \frac{|\mathcal{S}_j^k|}{|\mathcal{S}_j|} H(\mathcal{S}_j^k) \tag{2}$$

- where $H(\mathcal{S})$ is either the Shannon entropy ($H(\mathcal{S}) = -\sum_y p_y log(p_y)$) or alternatively the Gini impurity ($H(\mathcal{S}) = -\sum_y p_y(1 - p_y)$)

Training

- training stops when a maximum depth is reached or if information gain or training set size fall below fixed threshold

Training

- training stops when a maximum depth is reached or if information gain or training set size fall below fixed threshold
- single output $y \in \mathcal{Y}$ is assigned to a leaf node based on a problem specific ensemble model

Testing

- combining results from multiple trees depends on a problem specific ensemble

Testing

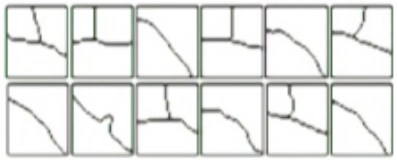
- combining results from multiple trees depends on a problem specific ensemble
- classification $\rightarrow$ majority voting

Testing

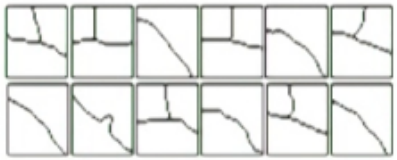
- combining results from multiple trees depends on a problem specific ensemble
- classification $\rightarrow$ majority voting
- regression $\rightarrow$ averaging

Structured Forests

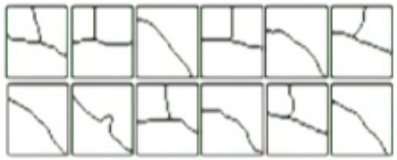
- structured output space $\mathcal{Y}$, e.g.:



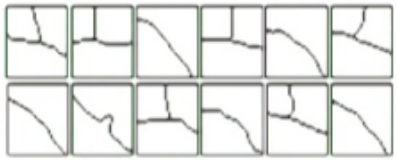
Structured Forests

- structured output space $\mathcal{Y}$, e.g.: 
- use of structured outputs in random forests presents following challenges:

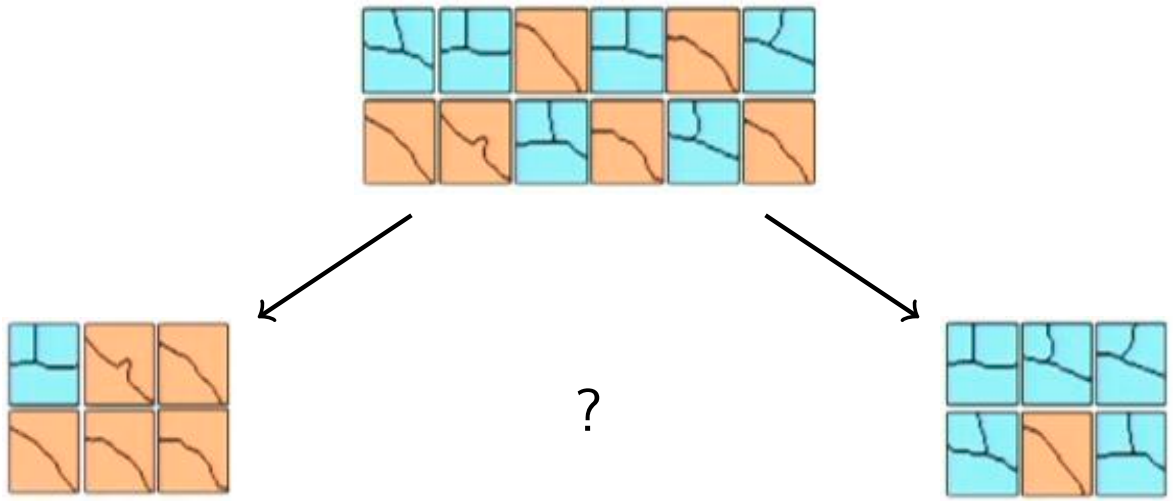
Structured Forests

- structured output space $\mathcal{Y}$, e.g.: 
- use of structured outputs in random forests presents following challenges:
  - computing splits for structured output spaces of high dimensions/complexity is time consuming

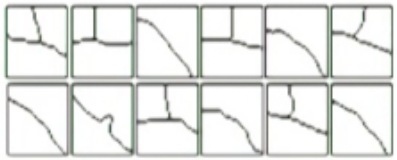
Structured Forests



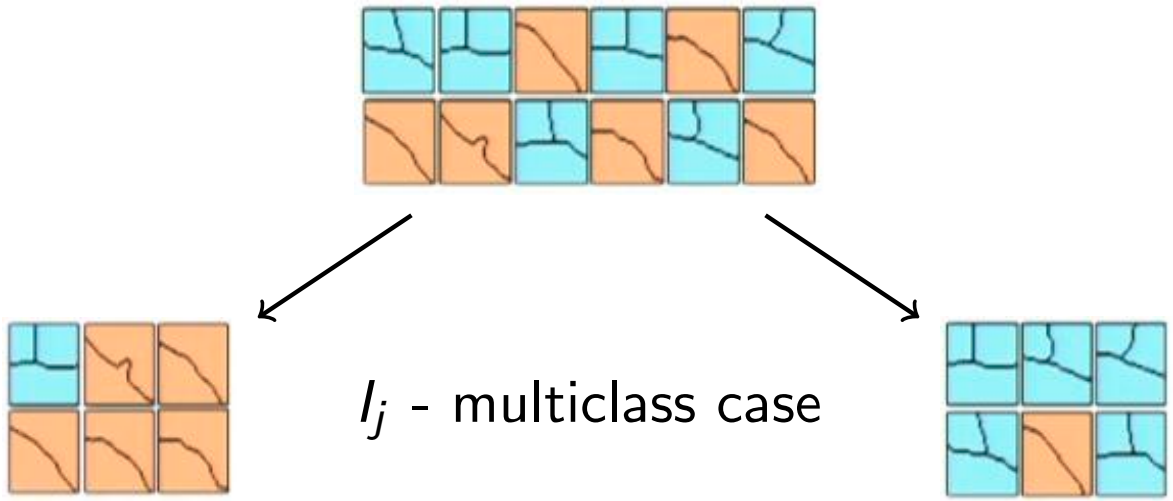
- structured output space $\mathcal{Y}$, e.g.:
- use of structured outputs in random forests presents following challenges:
  - computing splits for structured output spaces of high dimensions/complexity is time consuming
  - it is unclear how to define information gain



Structured Forests



- structured output space $\mathcal{Y}$, e.g.:
- use of structured outputs in random forests presents following challenges:
  - computing splits for structured output spaces of high dimensions/complexity is time consuming
  - it is unclear how to define information gain
- solution: mapping (somehow) structured output space $\mathcal{Y}$ into multiclass space $\mathcal{C} = \{1, ..., k\}$



Structured Edges Clustering

- How to efficiently compute splits?

Structured Edges Clustering

- How to efficiently compute splits?
- Can the task be transformed into a multiclass problem?

$$\mathcal{Y} \xrightarrow{?} \mathcal{C} = \{1, ..., k\}$$

Structured Edges Clustering

- How to efficiently compute splits?
- Can the task be transformed into a multiclass problem?

$$\mathcal{Y} \xrightarrow{?} \mathcal{C} = \{1, ..., k\}$$

- start with an intermediate mapping:

$$\Pi : \mathcal{Y} \to \mathcal{Z} \tag{3}$$

Structured Edges Clustering

- How to efficiently compute splits?
- Can the task be transformed into a multiclass problem?

$$\mathcal{Y} \xrightarrow{?} \mathcal{C} = \{1, ..., k\}$$

- start with an intermediate mapping:

$$\Pi : \mathcal{Y} \to \mathcal{Z} \tag{3}$$

- $z = \Pi(y)$ is a **long** binary vector, which encodes whether every pair of pixels in the y belongs to the same or different segment

Structured Edges Clustering

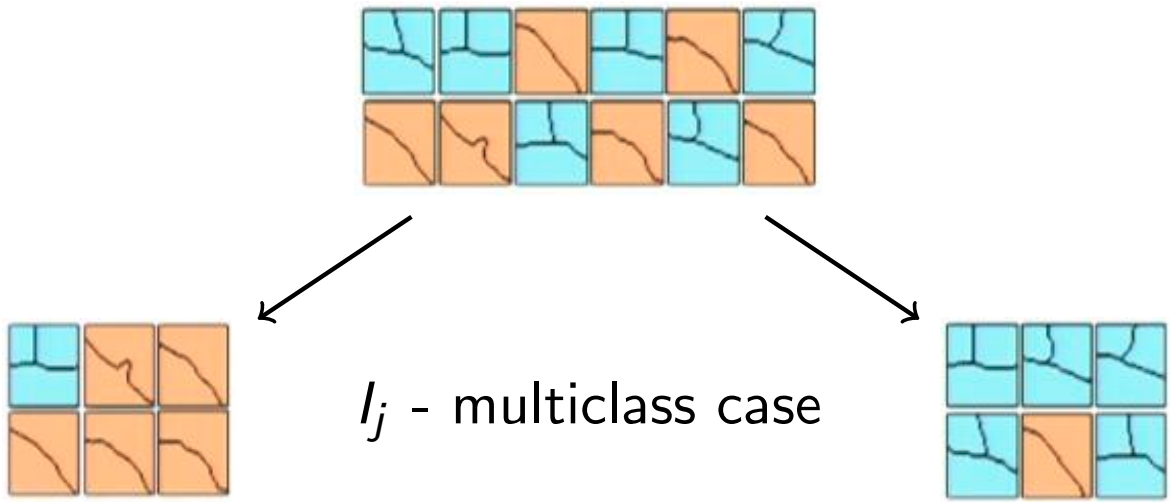
- dimension of vectors $z \in \mathcal{Z}$ is reduced by PCA to $m = 5$, and clustering (k-means) splits $\mathcal{Z}$ into $k = 2$ clusters

Structured Edges Clustering

- dimension of vectors $z \in \mathcal{Z}$ is reduced by PCA to $m = 5$, and clustering (k-means) splits $\mathcal{Z}$ into $k = 2$ clusters
- $\mathcal{Y} \xrightarrow{\text{pairs}} \mathcal{Z} \xrightarrow{\text{PCA, k-means}} \mathcal{C}$

Structured Edges Clustering

- dimension of vectors $z \in \mathcal{Z}$ is reduced by PCA to $m = 5$, and clustering (k-means) splits $\mathcal{Z}$ into $k = 2$ clusters
- $\mathcal{Y} \xrightarrow{\text{pairs}} \mathcal{Z} \xrightarrow{\text{PCA, k-means}} \mathcal{C}$



Structured Edges Clustering

- since the elements of $\mathcal{Y}$ are of size 16 x 16, the dimension of $\mathcal{Z}$ is $\binom{256}{2}$

Structured Edges Clustering

- since the elements of $\mathcal{Y}$ are of size 16 x 16, the dimension of $\mathcal{Z}$ is $\binom{256}{2}$
- too expensive $\rightarrow$ randomly sample $m = 256$ dimensions of $\mathcal{Z}$

Structured Edges Clustering

- since the elements of $\mathcal{Y}$ are of size 16 x 16, the dimension of $\mathcal{Z}$ is $\binom{256}{2}$
- too expensive $\rightarrow$ randomly sample $m = 256$ dimensions of $\mathcal{Z}$
- $\mathcal{Y} \xrightarrow{\text{sampled pairs}} \mathcal{Z} \xrightarrow{\text{PCA, k-means}} \mathcal{C}$

Ensemble model

- during training, we need to assign a single prediction to a leaf node

Ensemble model

- during training, we need to assign a single prediction to a leaf node
- during testing, we need to combine multiple predictions into one

Ensemble model

- during training, we need to assign a single prediction to a leaf node
- during testing, we need to combine multiple predictions into one
- to select a single output from a set $y_1, ..., y_k \in \mathcal{Y}$:

Ensemble model

- during training, we need to assign a single prediction to a leaf node
- during testing, we need to combine multiple predictions into one
- to select a single output from a set $y_1, ..., y_k \in \mathcal{Y}$:
  - compute $z_i = \Pi y_i$

Ensemble model

- during training, we need to assign a single prediction to a leaf node
- during testing, we need to combine multiple predictions into one
- to select a single output from a set $y_1, ..., y_k \in \mathcal{Y}$:
  - compute $z_i = \Pi y_i$
  - select y_{k^*} such that $k^* = \text{argmin}_k \sum_{i,j} (z_{k,j} - z_{i,j})^2$ (medoid)

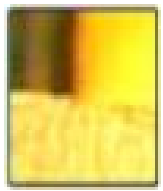
Ensemble model

- during training, we need to assign a single prediction to a leaf node
- during testing, we need to combine multiple predictions into one
- to select a single output from a set $y_1, ..., y_k \in \mathcal{Y}$:
  - compute $z_i = \Pi y_i$
  - select y_{k^*} such that $k^* = \text{argmin}_k \sum_{i,j}(z_{k,j} - z_{i,j})^2$ (medoid)
- a domain specific ensemble model (for edge map): $y'_{k^*} = E[y'_i]$

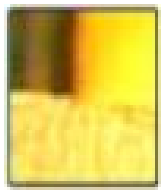
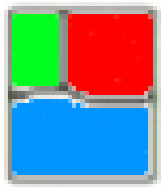
Structured Forest Training - Overview

- **input:**

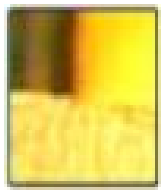
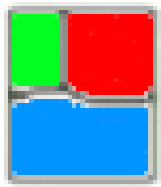
Structured Forest Training - Overview

- **input:**
  - image patch 32×32, sampled into 7228 features 

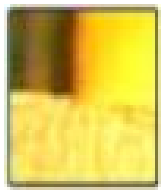
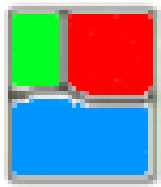
Structured Forest Training - Overview

- **input:**
  - image patch 32×32, sampled into 7228 features 
  - corresponding segmentation mask 16×16 

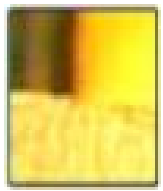
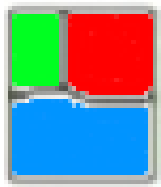
Structured Forest Training - Overview

- **input:**
  - image patch 32×32, sampled into 7228 features 
  - corresponding segmentation mask 16×16 
- randomly selected features per split

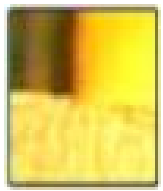
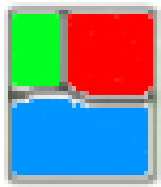
Structured Forest Training - Overview

- **input:**
  - image patch 32×32, sampled into 7228 features 
  - corresponding segmentation mask 16×16 
- randomly selected features per split
- segmentation masks $\rightarrow$ clusters $\rightarrow$ split information gain

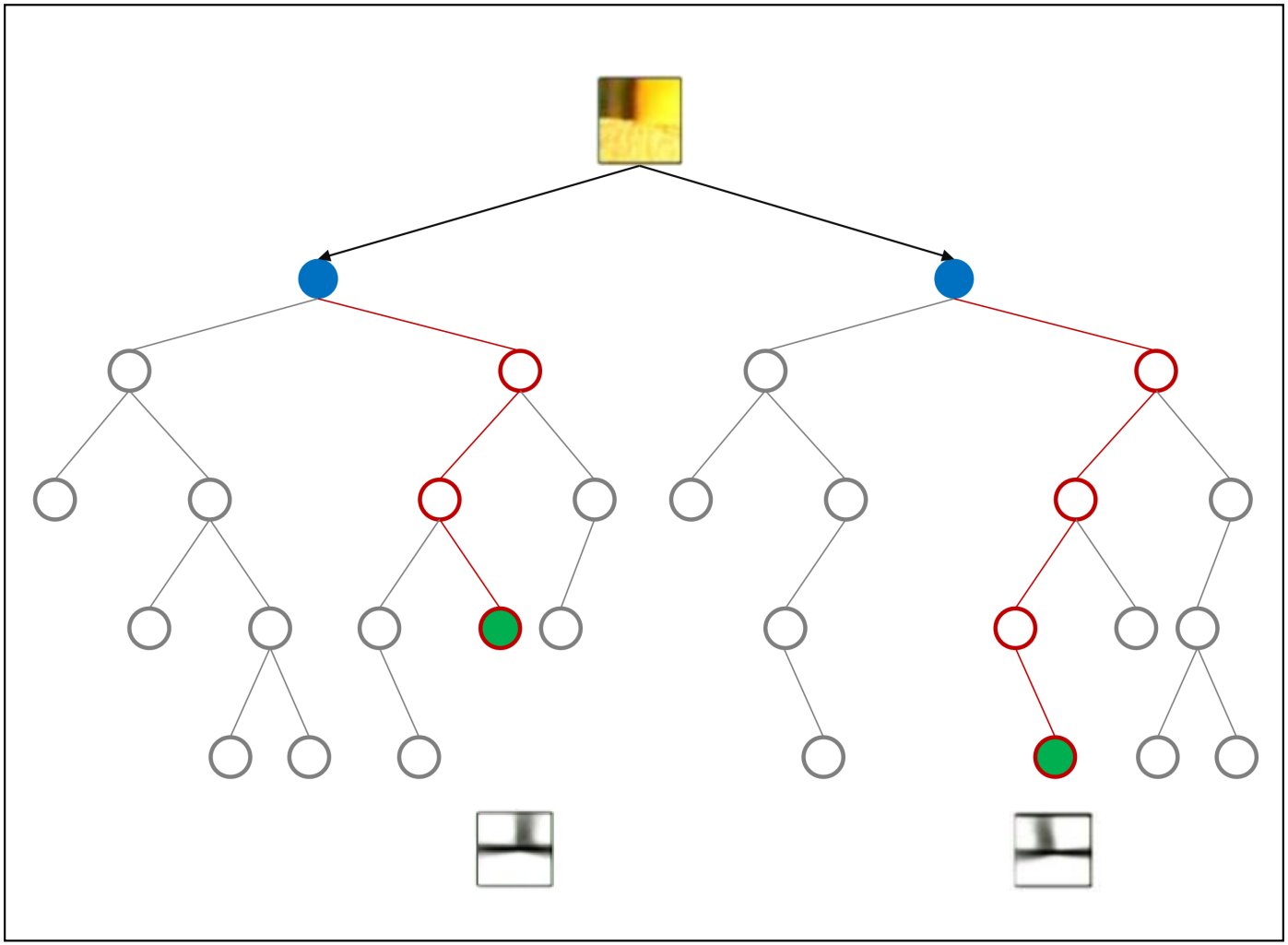
Structured Forest Training - Overview

- **input:**
  - image patch 32×32, sampled into 7228 features 
  - corresponding segmentation mask 16×16 
- randomly selected features per split
- segmentation masks $\rightarrow$ clusters $\rightarrow$ split information gain
- medoid $\rightarrow$ segmentation mask

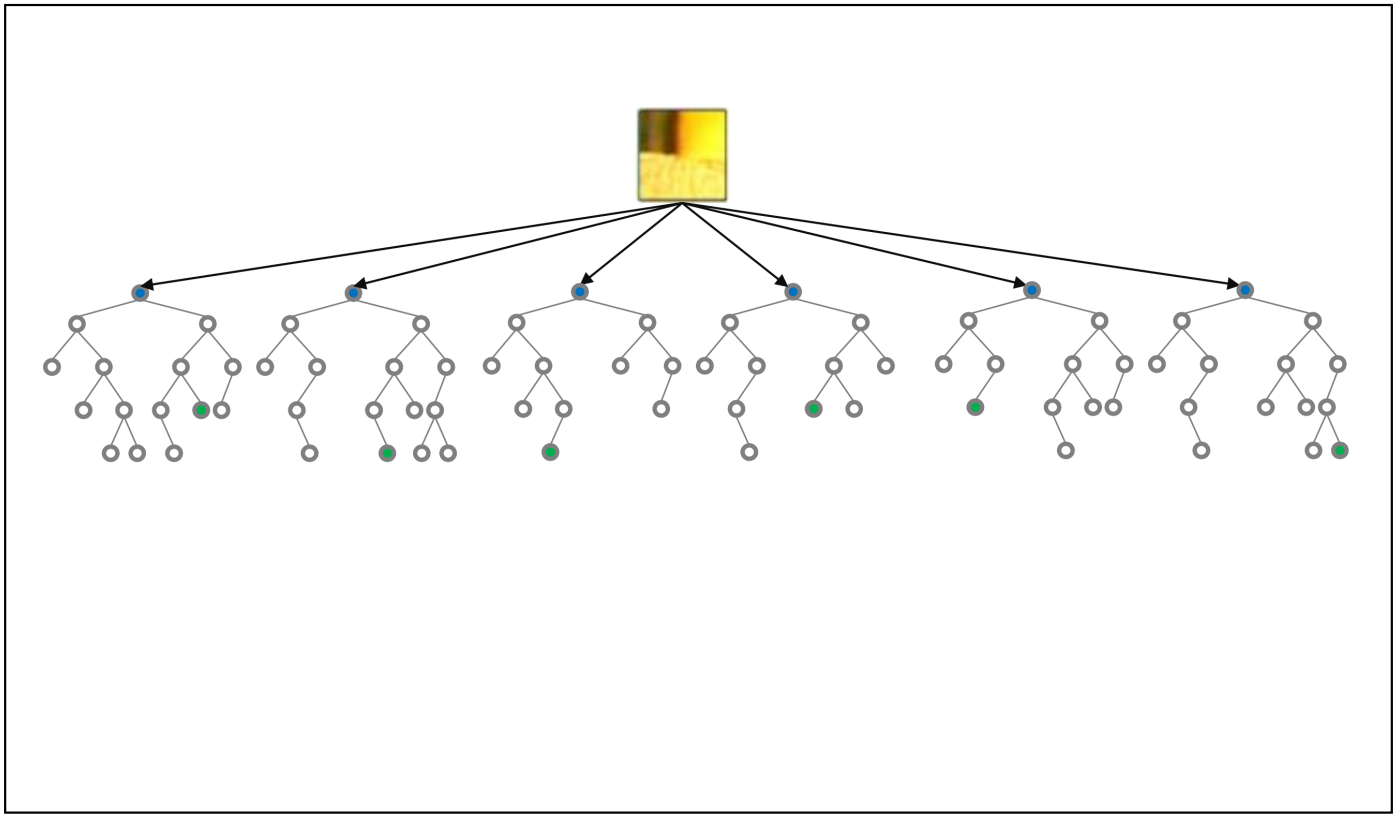
Structured Forest Training - Overview

- **input:**
  - image patch 32×32, sampled into 7228 features 
  - corresponding segmentation mask 16×16 
- randomly selected features per split
- segmentation masks $\rightarrow$ clusters $\rightarrow$ split information gain
- medoid $\rightarrow$ segmentation mask
- averaging $\rightarrow$ soft edge map

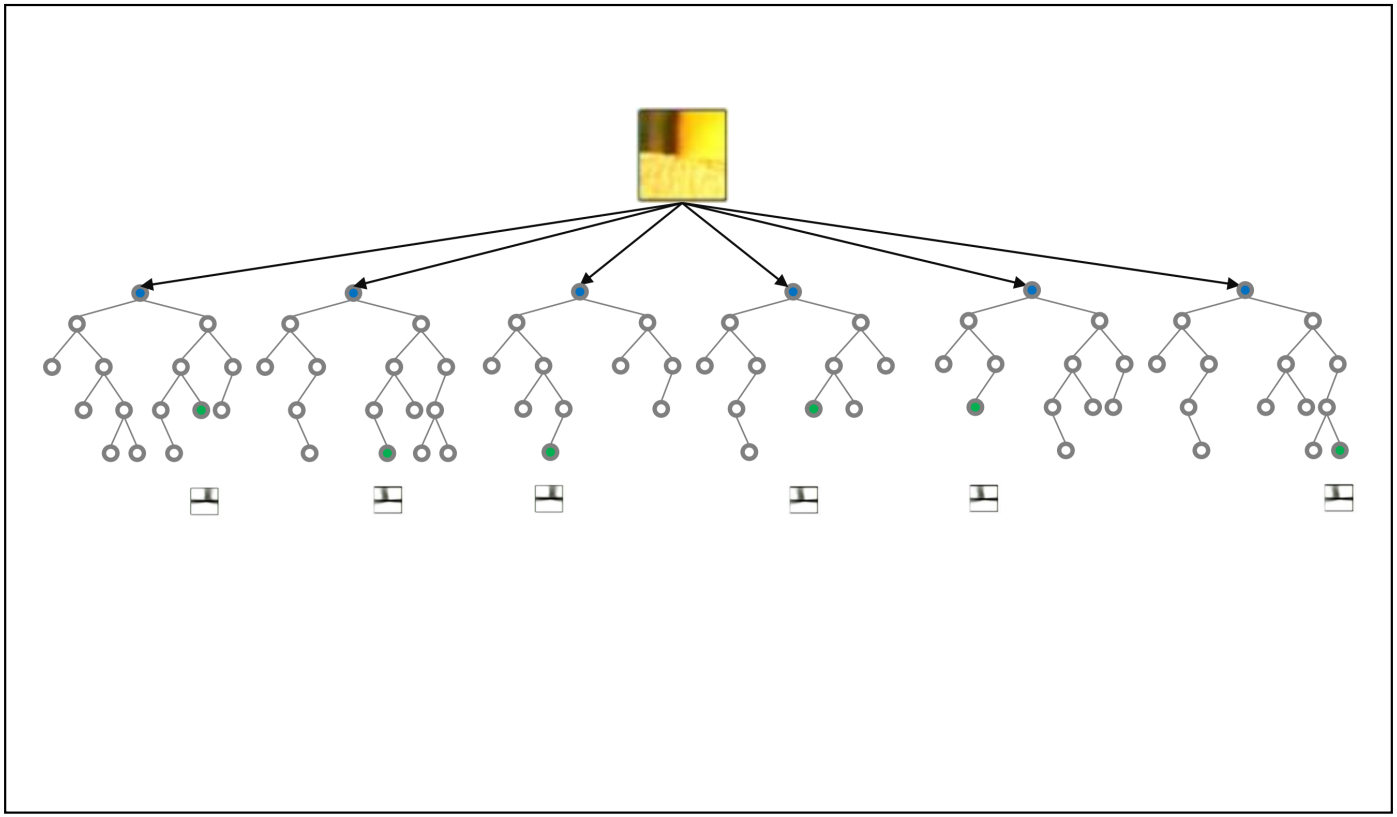
Structured Forest Testing



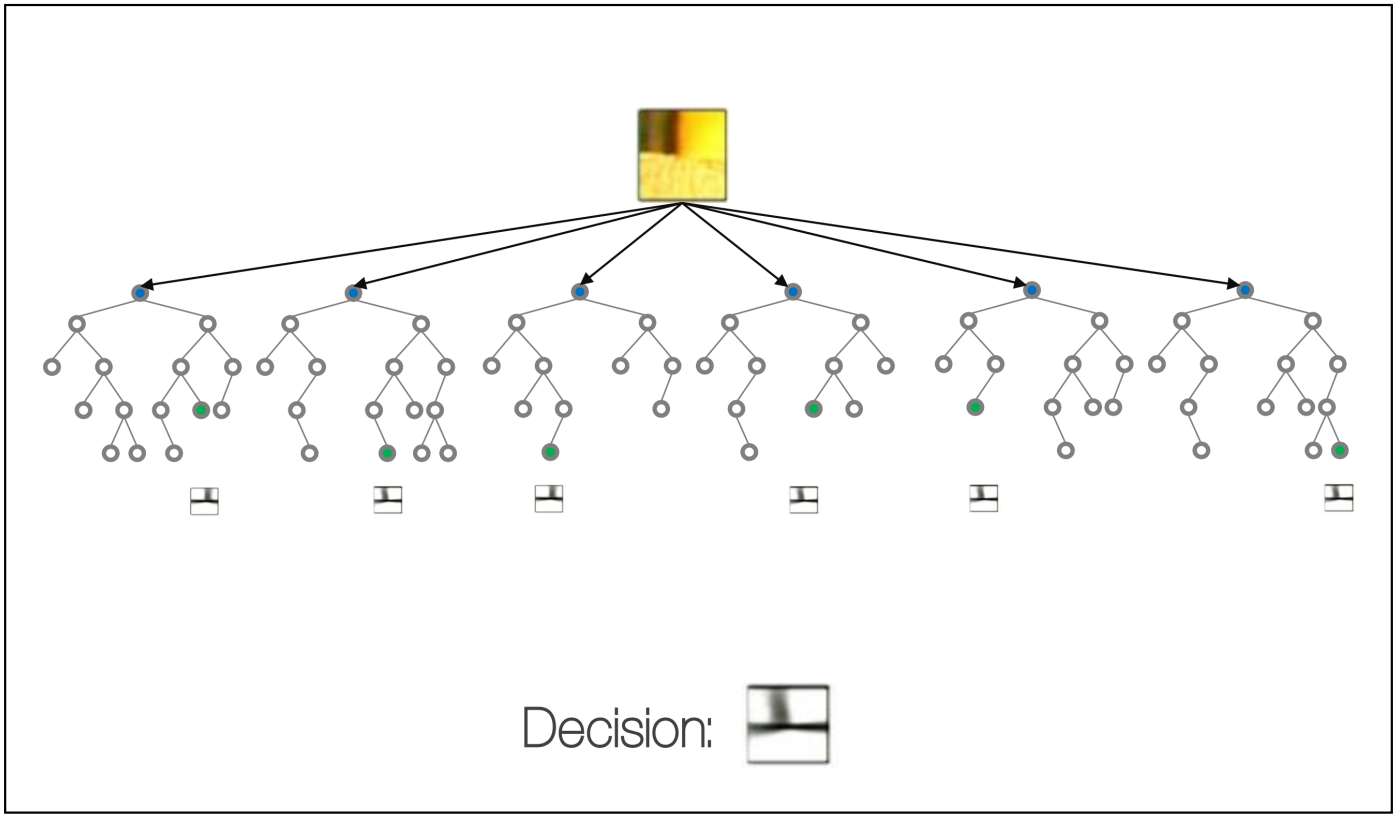
Structured Forest Testing



Structured Forest Testing



Structured Forest Testing



Efficiency

- Why is the method so fast?

Efficiency

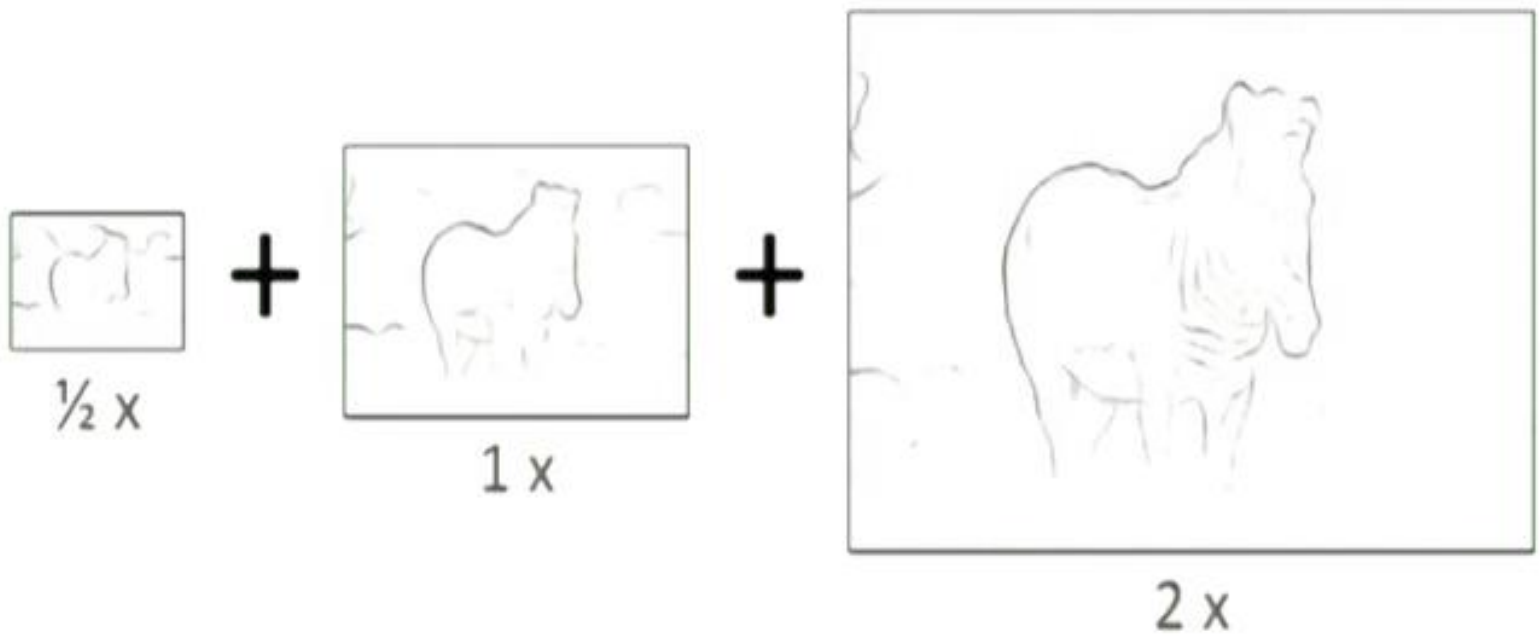
- Why is the method so fast?
- a single decision tree $\rightarrow$ lots of pixel information

Efficiency

- Why is the method so fast?
- a single decision tree $\rightarrow$ lots of pixel information
- lots of pixel information $\rightarrow$ a small random forest $\rightarrow$ fast evaluation

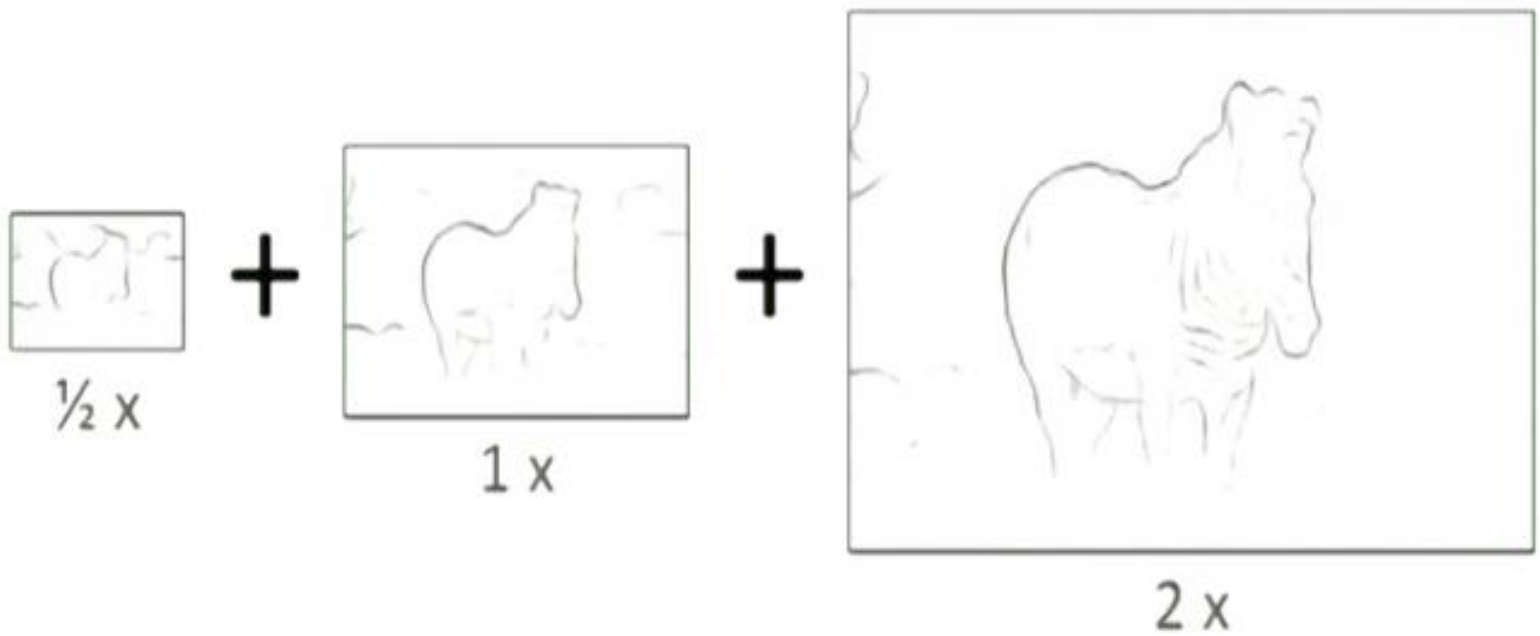
Multiscale Detection

- multiscale version takes original, double, and half resolution of an input image



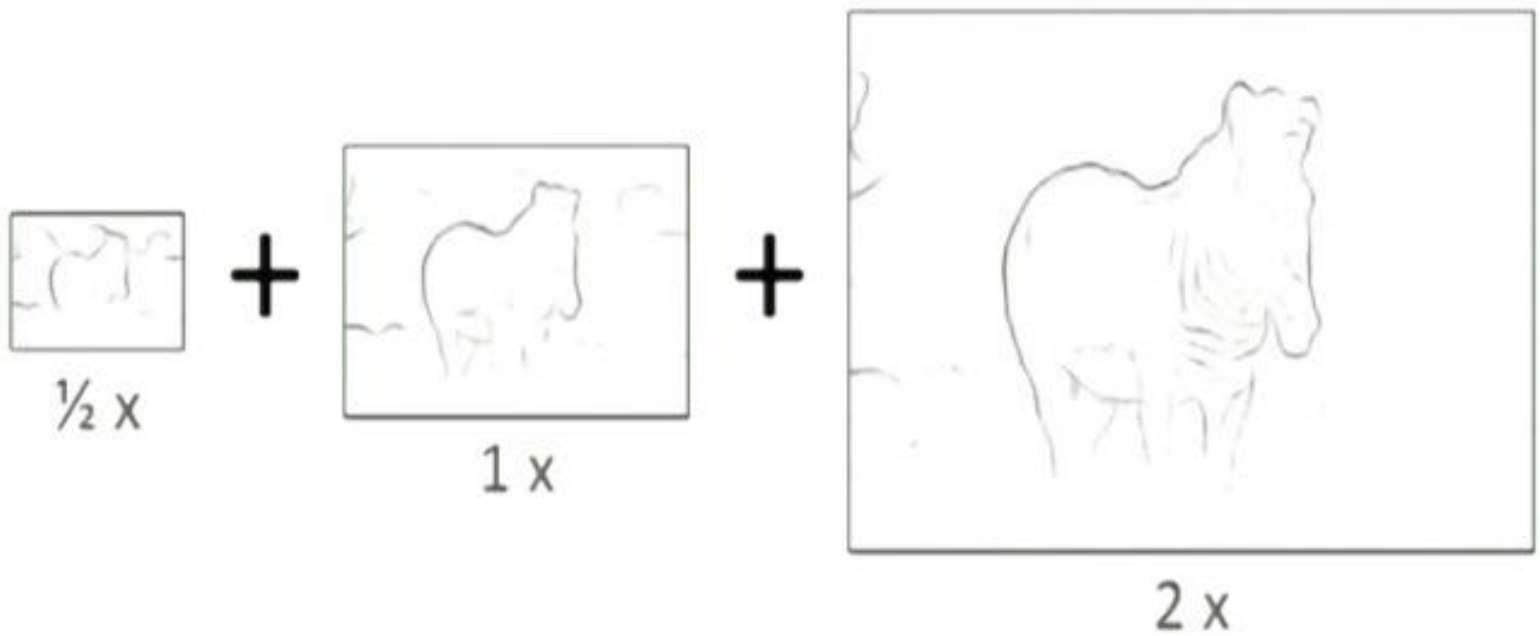
Multiscale Detection

- multiscale version takes original, double, and half resolution of an input image
- resulting three edge maps are averaged



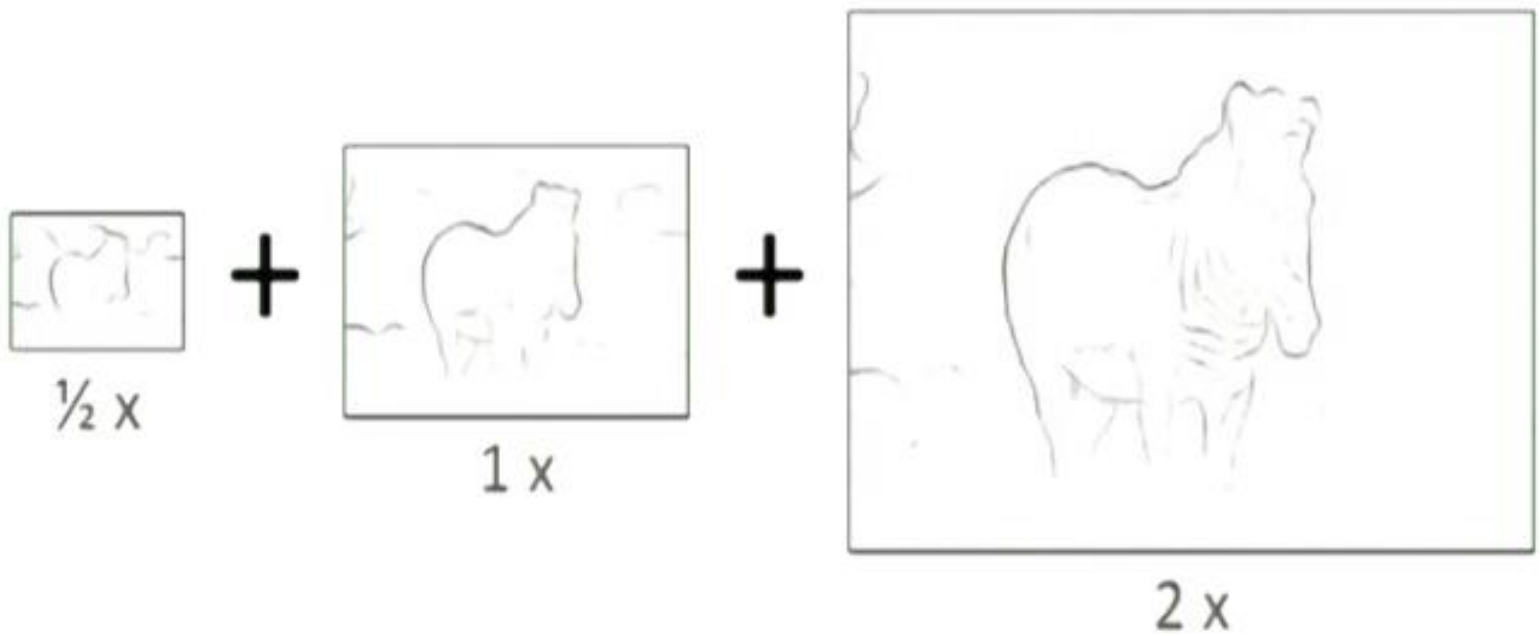
Multiscale Detection

- multiscale version takes original, double, and half resolution of an input image
- resulting three edge maps are averaged
- slower ($\times 5$)



Multiscale Detection

- multiscale version takes original, double, and half resolution of an input image
- resulting three edge maps are averaged
- slower ($\times 5$)
- improved edge quality



Edge Sharpening

- individual predictions are noisy and do not perfectly align to each other or the underlying image data

Edge Sharpening

- individual predictions are noisy and do not perfectly align to each other or the underlying image data
- sharpening takes individual prediction y and produces a new mask that better aligns it to the image patch x:

Edge Sharpening

- individual predictions are noisy and do not perfectly align to each other or the underlying image data
- sharpening takes individual prediction y and produces a new mask that better aligns it to the image patch x:
  - compute mean segment s color, $\mu_s = E[x(j) \mid y(j) = s]$

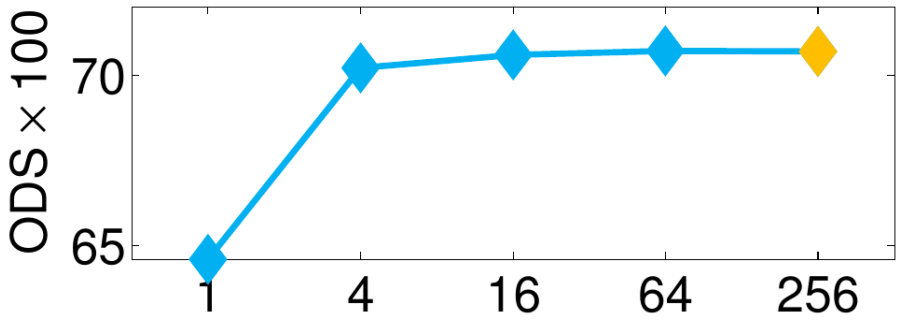
Edge Sharpening

- individual predictions are noisy and do not perfectly align to each other or the underlying image data
- sharpening takes individual prediction y and produces a new mask that better aligns it to the image patch x:
  - compute mean segment s color, $\mu_s = E[x(j) \mid y(j) = s]$
  - change pixel j assignment if the pixel color $x(j)$ is closer to different segment ($s^* = \text{argmin}_s \|\mu_s - x(j)\|$) and such segment labeling is in 4-connected vicinity

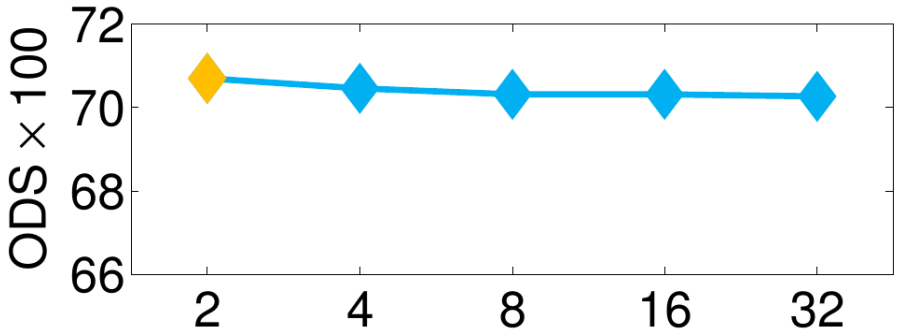
Edge Sharpening

- individual predictions are noisy and do not perfectly align to each other or the underlying image data
- sharpening takes individual prediction y and produces a new mask that better aligns it to the image patch x:
  - compute mean segment s color, $\mu_s = E[x(j) \mid y(j) = s]$
  - change pixel j assignment if the pixel color $x(j)$ is closer to different segment ($s^* = \text{argmin}_s \|\mu_s - x(j)\|$) and such segment labeling is in 4-connected vicinity
- sharpening can be repeated multiple times, Dollar & Zitnick claims that in practice two steps suffice

Parameter Sweeps

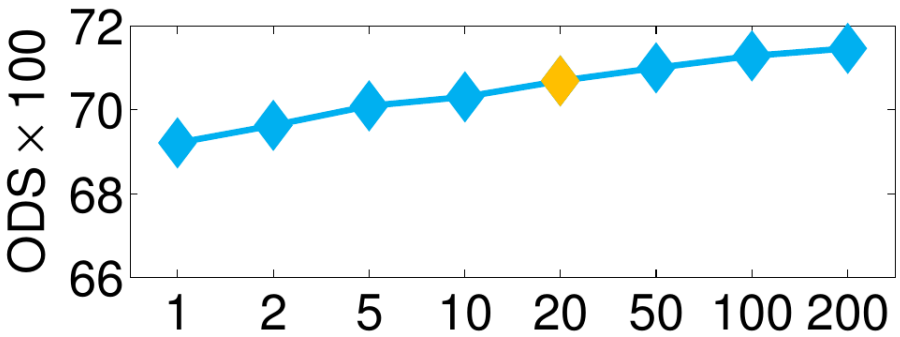


(a) m (size of $\mathcal{Z}$)

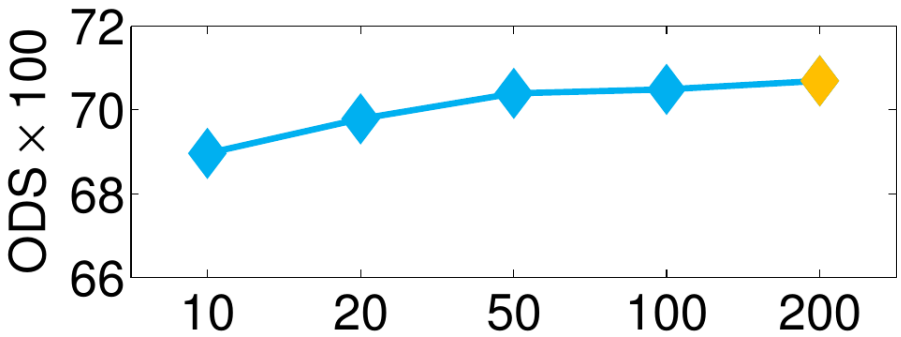


(b) k (size of $\mathcal{C}$)

Parameter Sweeps

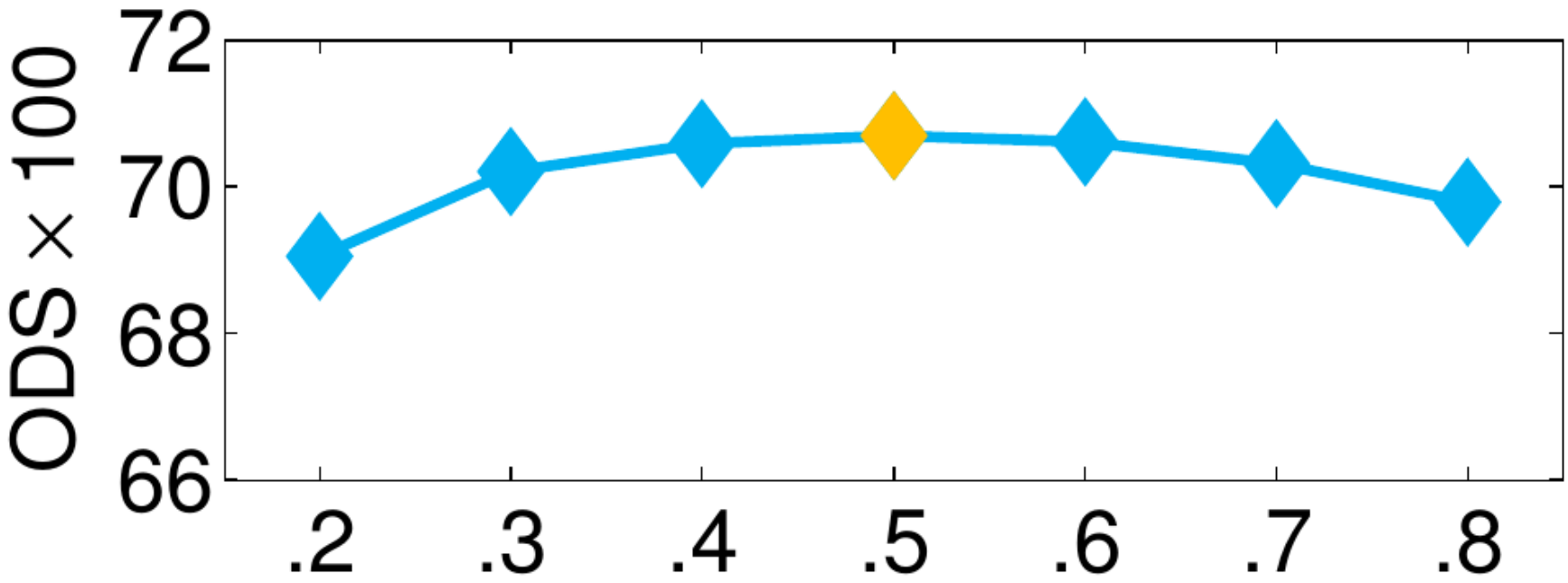


(c) # train patches $\times 10^4$



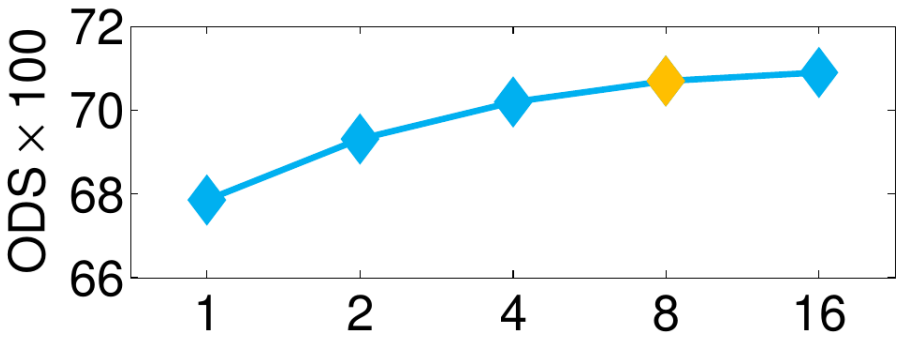
(d) # train images

Parameter Sweeps

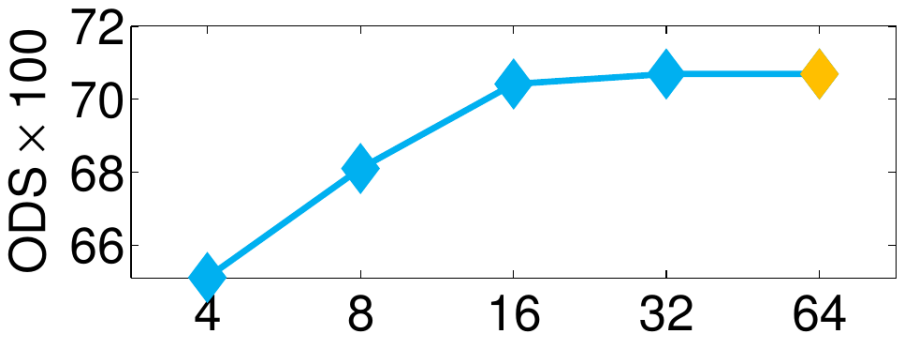


(e) fraction 'positives'

Parameter Sweeps



(g) # decision trees



(h) max tree depth

Results - SE

	ODS	OIS	AP	R50	FPS
Human	.80	.80	-	-	-
SE	.73	.75	.77	.90	**30**
SE+SH	.74	.76	.79	**.93**	12.5
SE+MS	.74	.76	.78	.90	6
SE+MS+SH	**.75**	.77	**.80**	**.93**	2.5

Results - HED

Table 4. Results on BSDS500. ∗BSDS300 results,†GPU time

	ODS	OIS	AP	FPS
Human	.80	.80	-	-
Canny	.600	.640	.580	15
Felz-Hutt [9]	.610	.640	.560	10
BEL [5]	.660∗	-	-	1/10
gPb-owt-ucm [1]	.726	.757	.696	1/240
Sketch Tokens [24]	.727	.746	.780	1
SCG [31]	.739	.758	.773	1/280
SE-Var [6]	.746	.767	.803	2.5
OEF [13]	.749	.772	.817	-
DeepNets [21]	.738	.759	.758	1/5†
N4-Fields [10]	.753	.769	.784	1/6†
DeepEdge [2]	.753	.772	.807	$1/10^3$†
CSCNN [19]	.756	.775	.798	-
DeepContour [34]	.756	.773	.797	1/30†
HED (ours)	**.782**	**.804**	**.833**	2.5†, 1/12

Conclusion

- realtime

Conclusion

- realtime
- structured learning

Conclusion

- realtime
- structured learning
- suitable preprocessing step for methods requiring speed