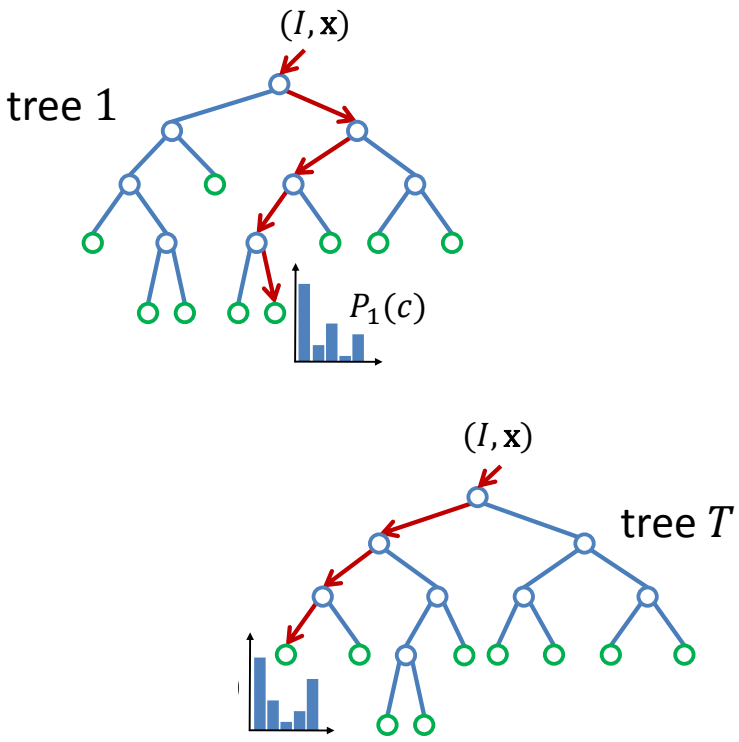
Random Forests vs. Deep Learning

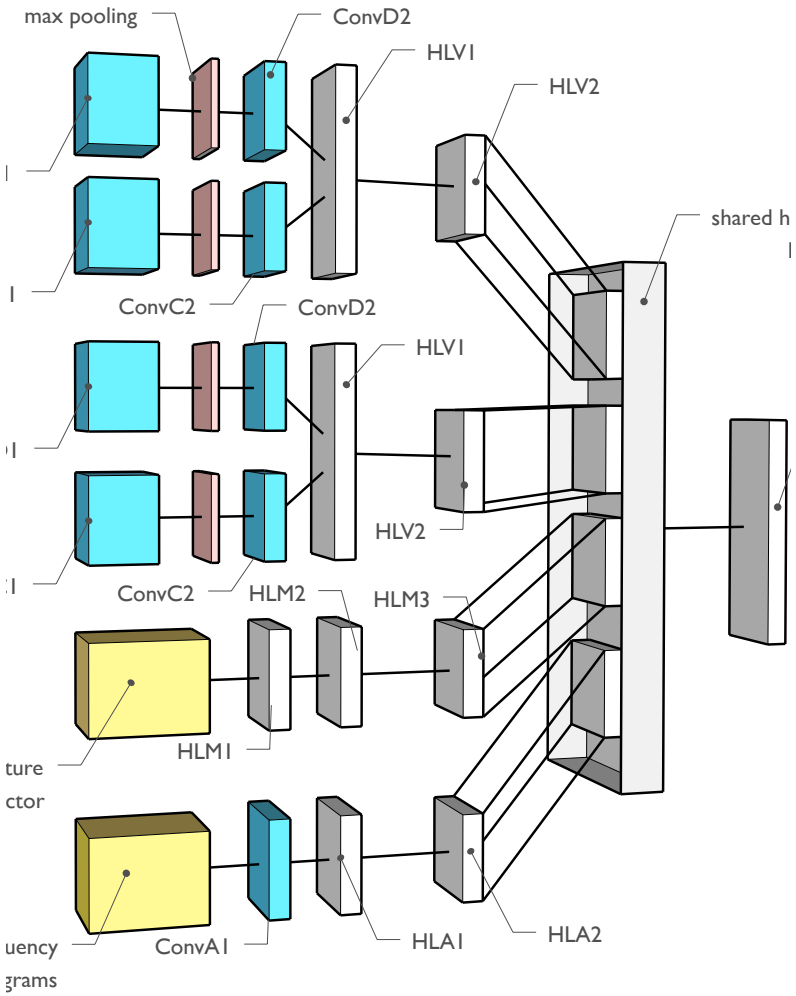
Christian Wolf
Université de Lyon, INSA-Lyon
LIRIS UMR CNRS 5205

November 26th, 2015

RF vs. DL

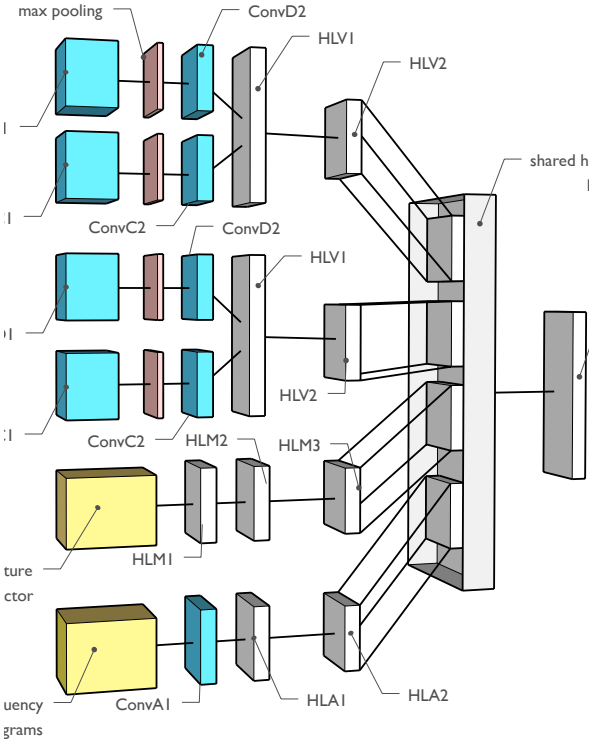
Goal: prediction (classification, regression)





Deep networks

- Many layers, many parameters …. and all of them are used for testing, for each single sample!
- Feature learning integred into classification
- End-to-end training, using gradient of the loss function

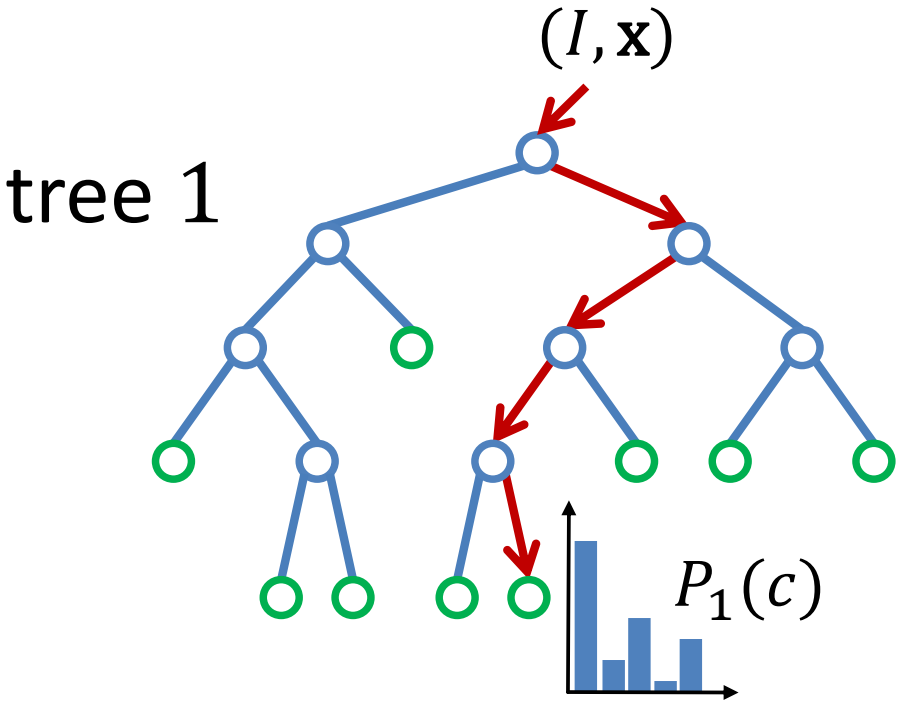


Layer	Filter size / n.o. units	N.o. parameters	Pooling
Paths V1, V2			
Input D1,D2	72×72×5	-	2×2×1
ConvD1	25×5×5×3	1900	2×2×3
ConvD2	25×5×5	650	1×1
Input C1,C2	72×72×5	-	2×2×1
ConvC1	25×5×5×3	1900	2×2×3
ConvC2	25×5×5	650	1×1
HLV1	900	3 240 900	-
HLV2	450	405 450	-
Path M			
Input M	183	-	
HLM1	700	128 800	-
HLM2	700	490 700	-
HLM3	350	245 350	-
Path A			
Input A	40×9	-	1×1
ConvA1	25×5×5	650	1×1
HLA1	700	3 150 000	-
HLA2	350	245 350	-
Shared layers			
HLS1	1600	3 681 600	-
HLS2	84	134 484	-
Output layer	21	1785	-

12.4M per scale = 37.2M parameters total!

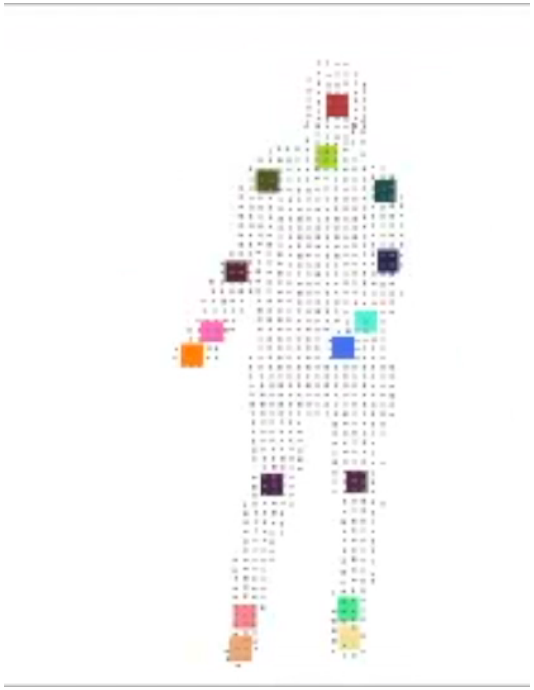
Random Forests

- Many levels, many parameters …. but only $\log_2(N)$ of them are used for testing!
- Training is done layer-wise, no end-to-end. No gradient on the objective function
- No/limited feature learning



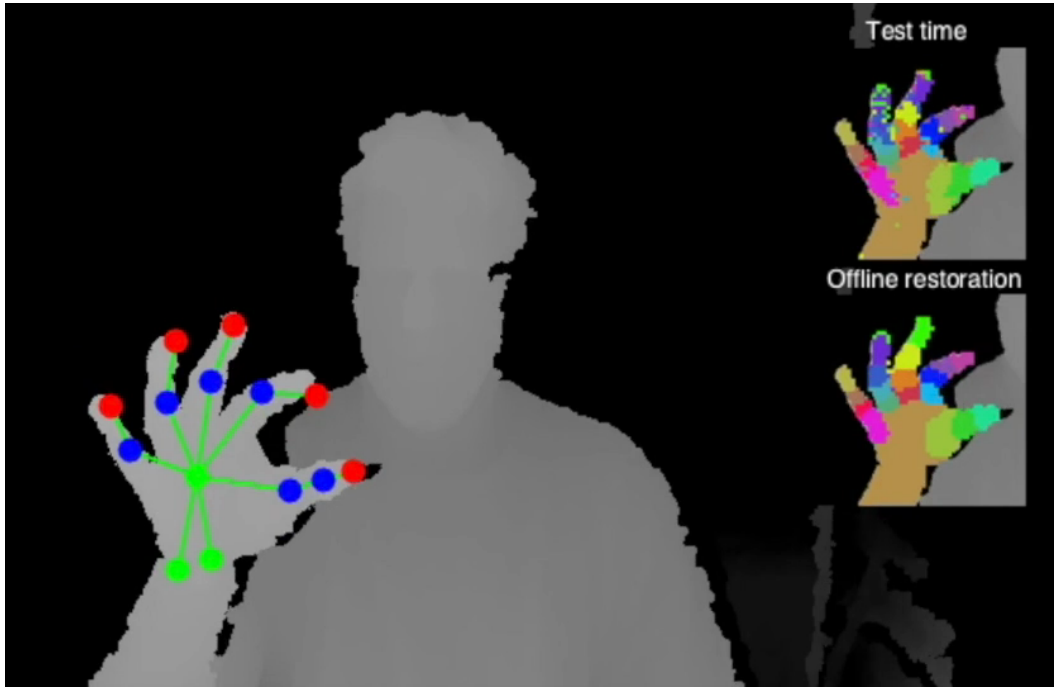
RF vs DL : applications (1)

Full body pose, Random Forest
3 Trees, depth 20
>10M parameters



[Shotton et al., CVPR 2011]
(Microsoft Research)

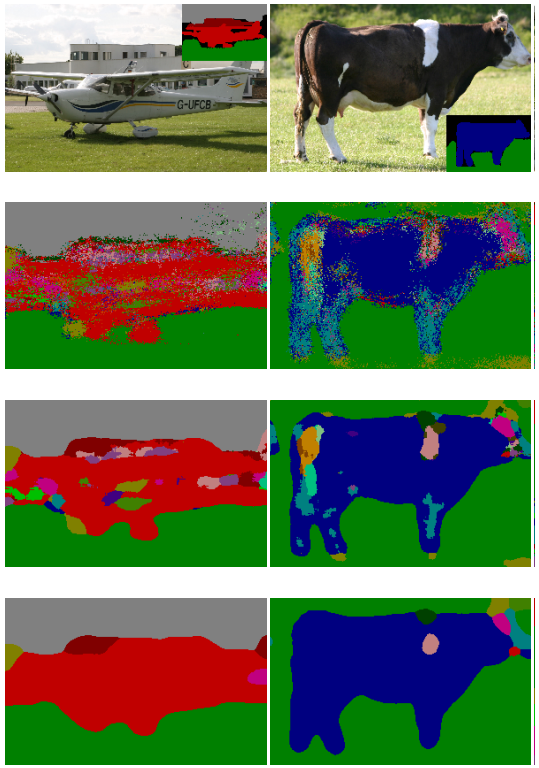
Hand pose with a deep network
Semi/weakly supervised training
8 layers, ~5M parameters



[Neverova, Wolf, Nebout, Taylor,
under review, arXiv 2015]

RF vs DL : applications (2)

Scene parsing with structured random forests

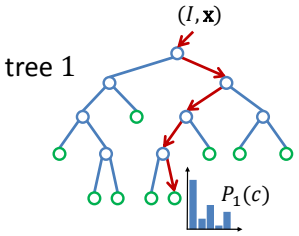


[Kontschieder et al., CVPR 2014]
(Microsoft Research)

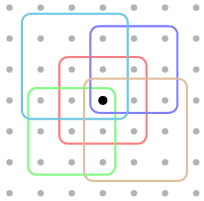
Scene parsing with deep networks (5 layers, ~2M parameters)

[Fourure, Emonet, Fromont, Muselet, Tremeau, Wolf, under review]

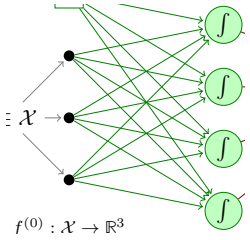
Types of random forests



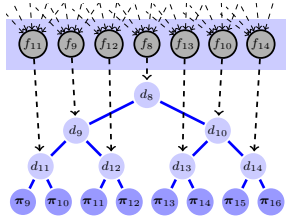
Classical random forests



Structured random forests



Neural random forests



Deep convolutional random forests

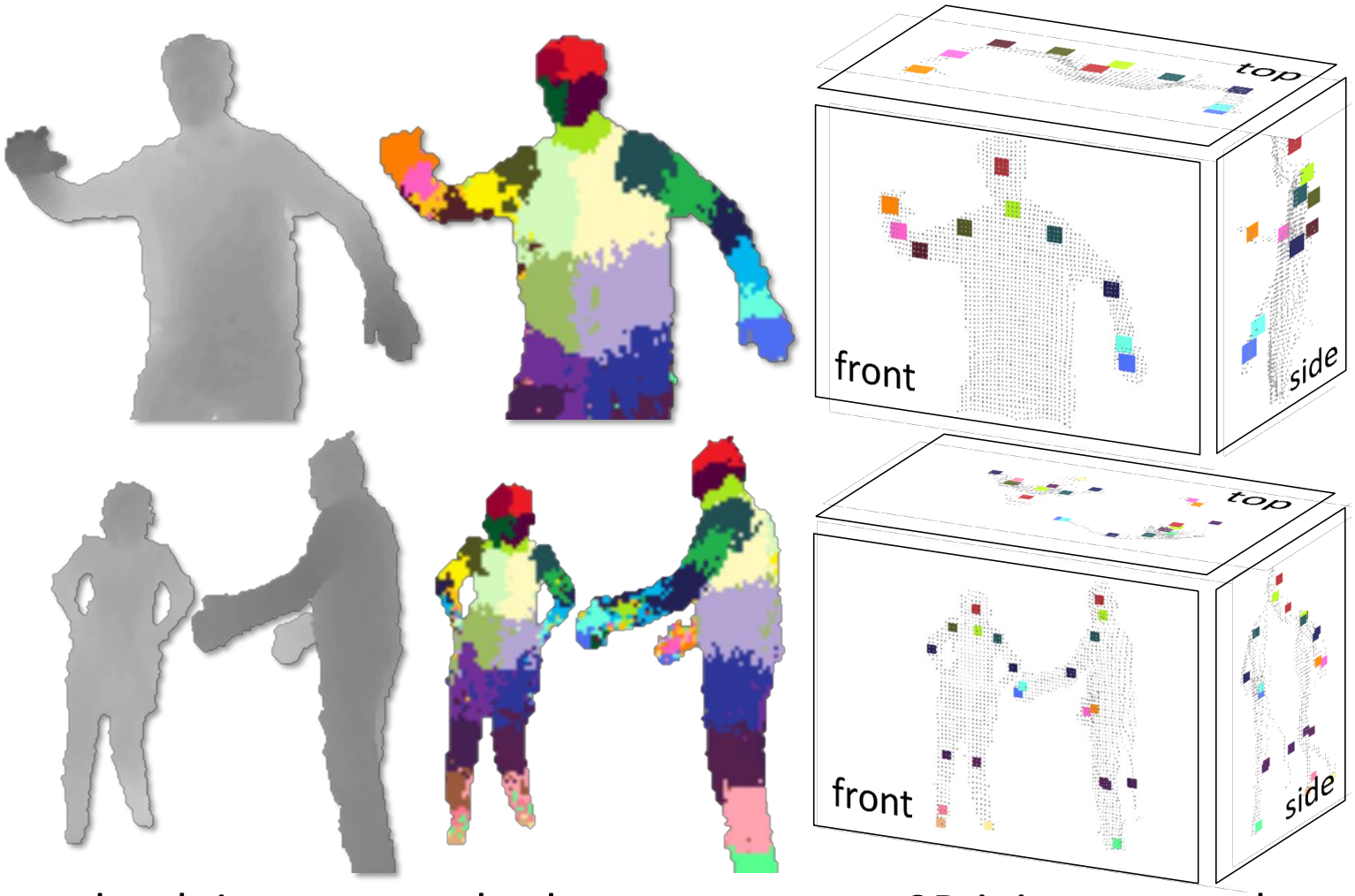
Example for classical RF

Real-Time Human Pose Recognition in Parts from Single Depth Images

Jamie Shotton Andrew Fitzgibbon Mat Cook Toby Sharp Mark Finocchio
Richard Moore Alex Kipman Andrew Blake
Microsoft Research Cambridge & Xbox Incubation

[Shotton et al., CVPR 2011]
(Best Paper!)

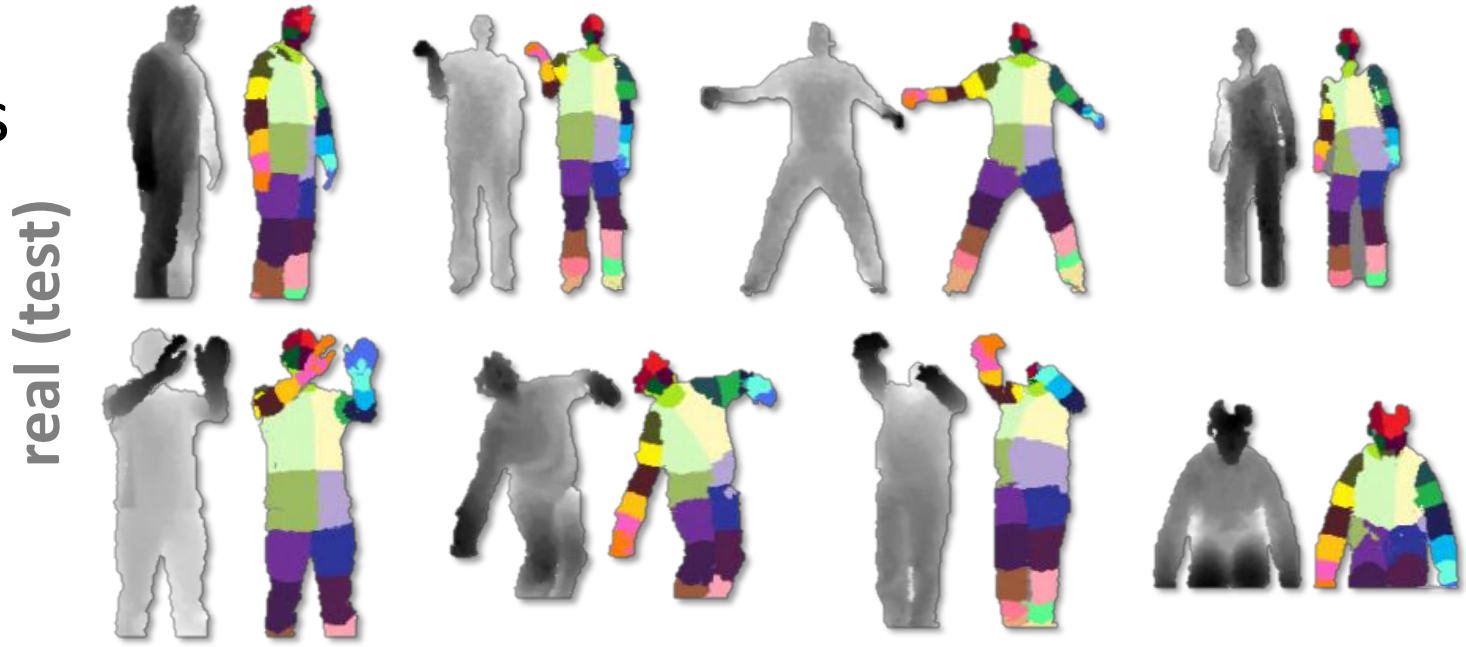
Depth images -> 3D joint locations



depth image ➡ body parts ➡ 3D joint proposals

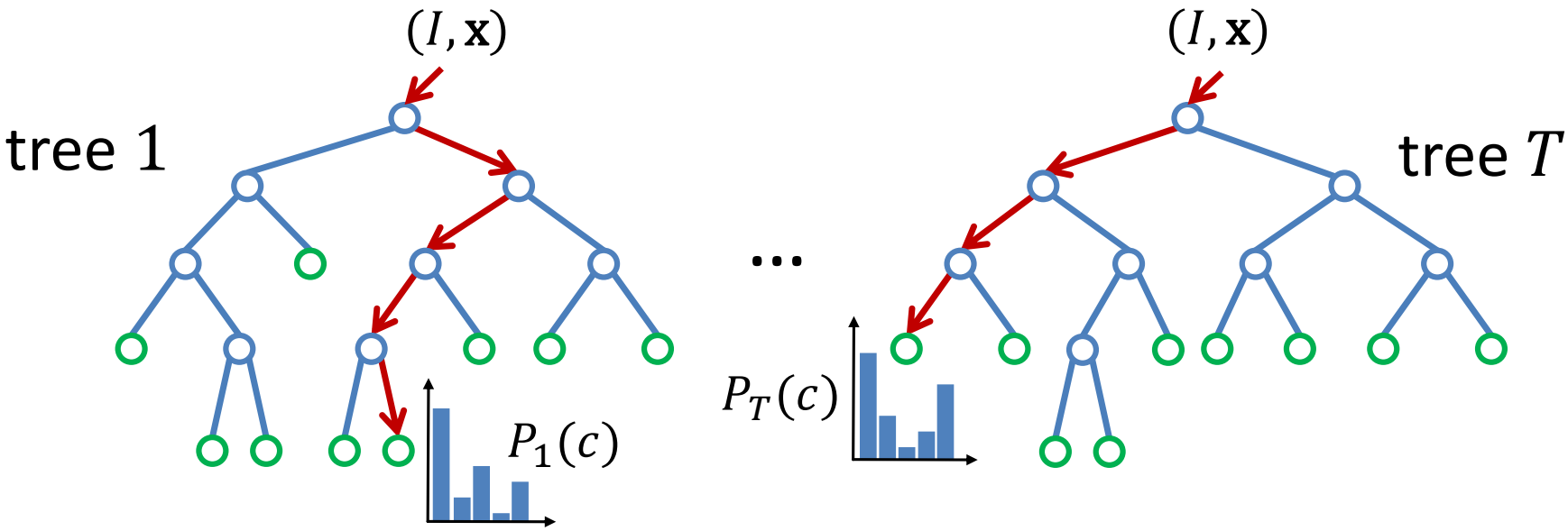
[Shotton et al., CVPR 2011]

31 body parts
(Labels)



[Shotton et al., CVPR 2011]

Classification with random forests



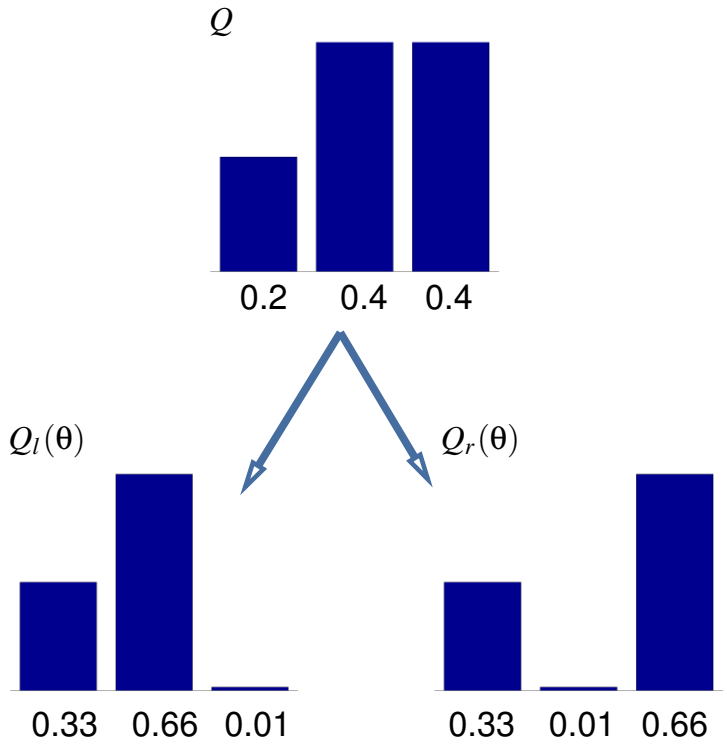
Each split node thresholds one of the features
Each leaf node contains a class distribution $P_t(c|I, \mathbf{x})$
Class distributions are averaged over the trees:

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

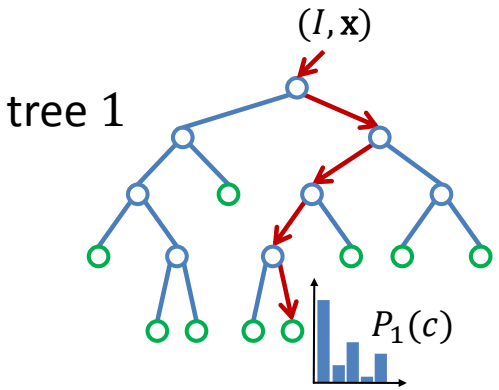
[Shotton et al., CVPR 2011]

Learning & Entropy

A good split function minimizes entropy in the label distributions.



Random forests: learning algorithm



Training:

- 3 trees
- depth 20
- 1.000.000 images
- 2000 candidate features
- 50 thresholds per feature

1 day : 1000 cores cluster

1. Randomly propose a set of splitting candidates $\phi = (\theta, \tau)$ (feature parameters θ and thresholds τ).
2. Partition the set of examples $Q = \{(I, \mathbf{x})\}$ into left and right subsets by each ϕ:

$$Q_{\mathrm{l}}(\phi) = \{ (I, \mathbf{x}) \mid f_\theta(I, \mathbf{x}) < \tau \} \quad (3)$$

$$Q_{\mathrm{r}}(\phi) = Q \setminus Q_{\mathrm{l}}(\phi) \quad (4)$$

3. Compute the ϕ giving the largest gain in information:

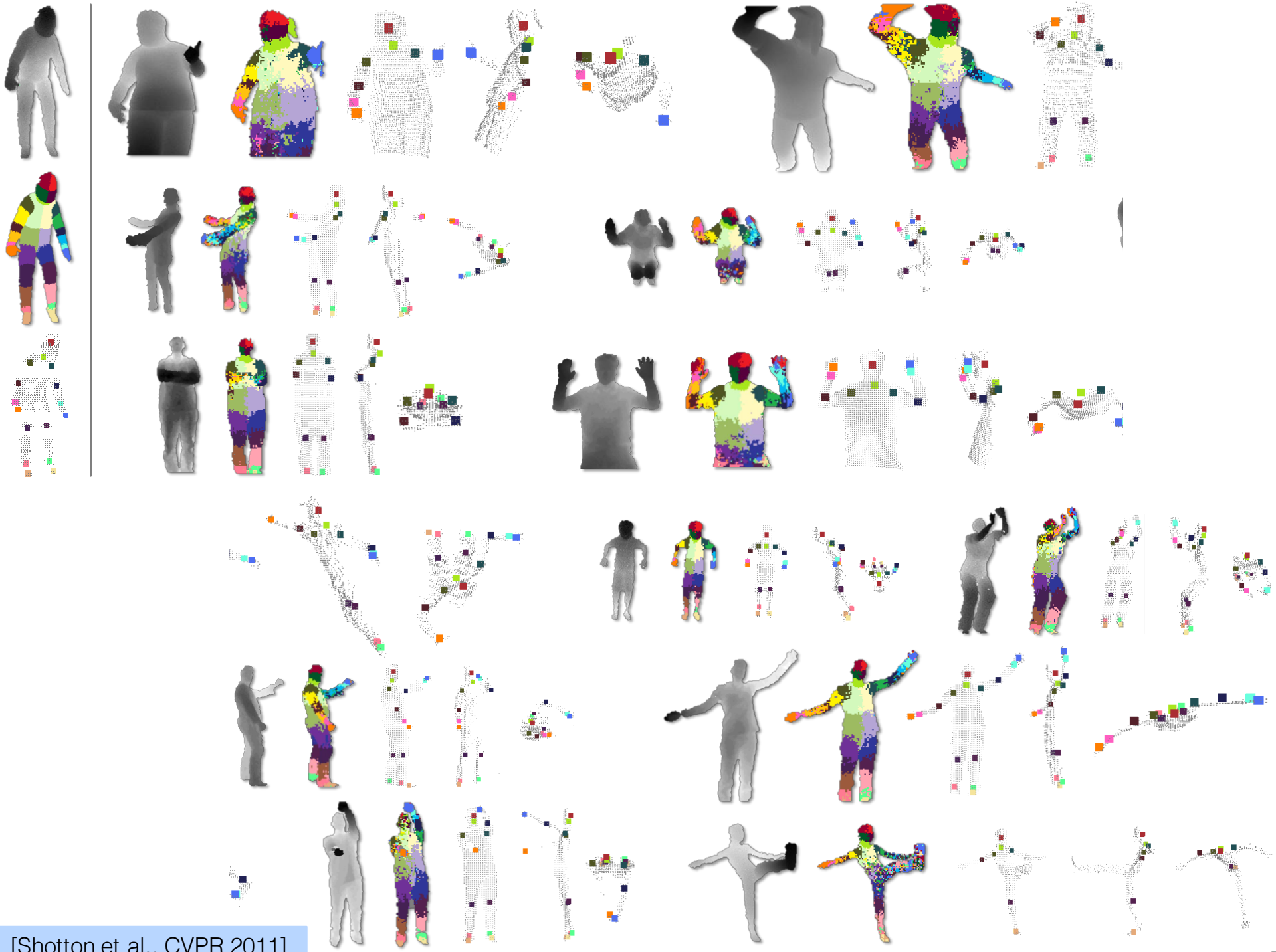
$$\phi^\star = \operatorname*{argmax}_\phi G(\phi) \quad (5)$$

$$G(\phi) = H(Q) - \sum_{s \in \{\mathrm{l},\mathrm{r}\}} \frac{|Q_s(\phi)|}{|Q|} H(Q_s(\phi)) \quad (6)$$

where Shannon entropy $H(Q)$ is computed on the normalized histogram of body part labels $l_I(\mathbf{x})$ for all $(I, \mathbf{x}) \in Q$.

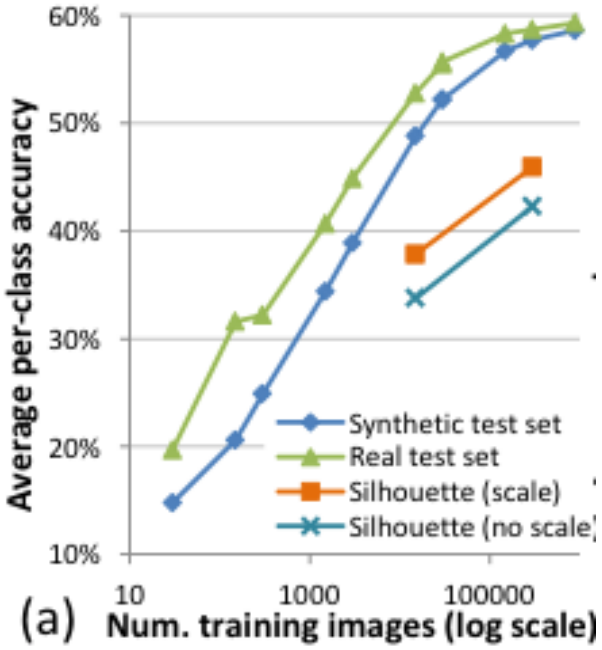
4. If the largest gain $G(\phi^\star)$ is sufficient, and the depth in the tree is below a maximum, then recurse for left and right subsets $Q_{\mathrm{l}}(\phi^\star)$ and $Q_{\mathrm{r}}(\phi^\star)$.

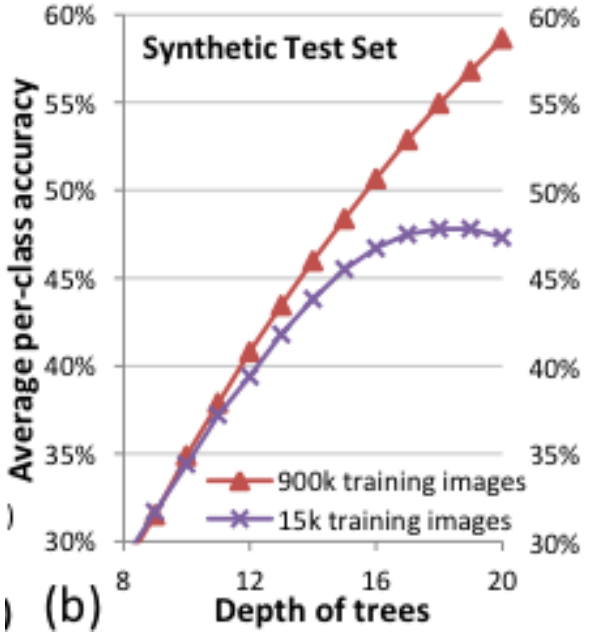
[Shotton et al., CVPR 2011]



[Shotton et al., CVPR 2011]

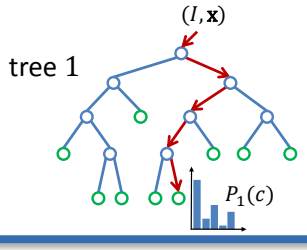
Dependencies of results on hyper-parameters



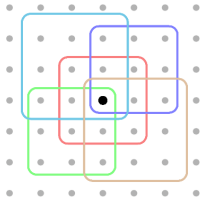


[Shotton et al., CVPR 2011]

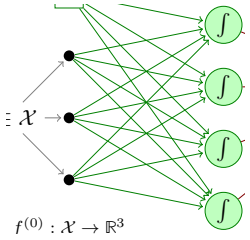
Types of random forests



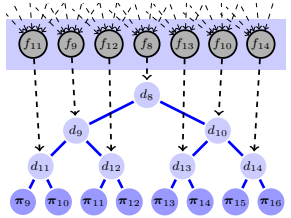
Classical random forests



Structured random forests



Neural random forests

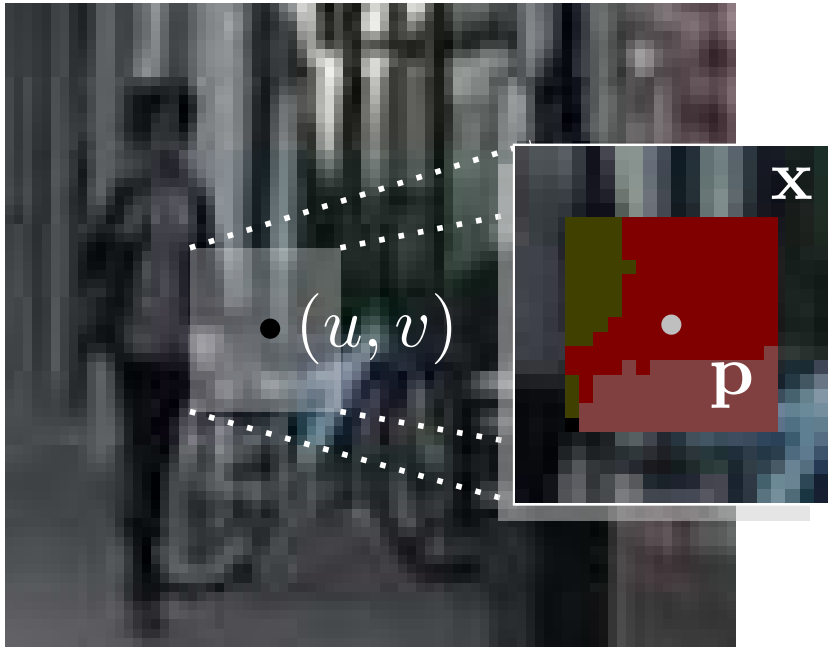


Deep convolutional random forests

Structured random forests

In the classical version, decision (=leaf) nodes contain predictions for a single pixel (a label or a posterior distribution)

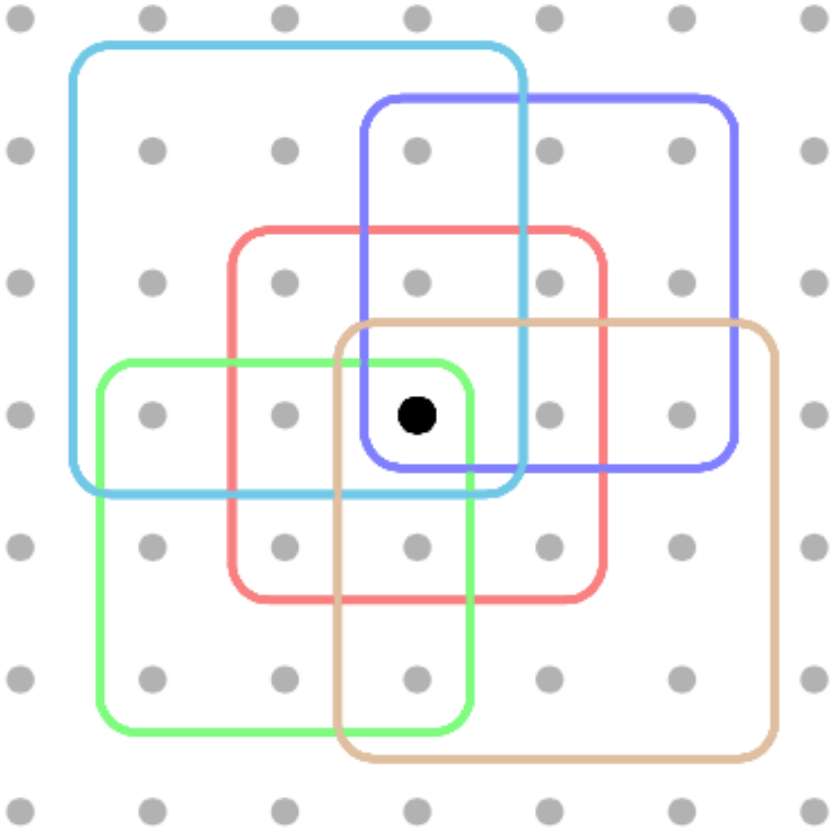
In the structured version, a decision node is assigned a rectangular patch of predictions.



[Kontschieder et al., ICCV 2011]

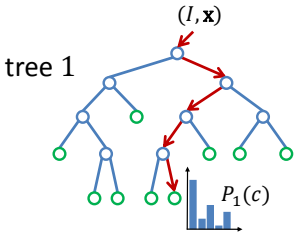
Structured version : integration

Integration over multiple pixels by vote:

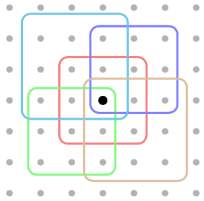


[Kontschieder et al., ICCV 2011]

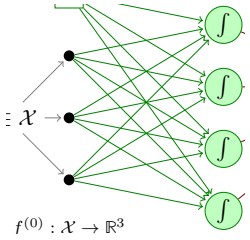
Types of random forests



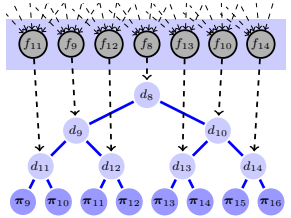
Classical random forests



Structured random forests



Neural random forests



Deep convolutional random forests

Neural Decision Forests for Semantic Image Labelling

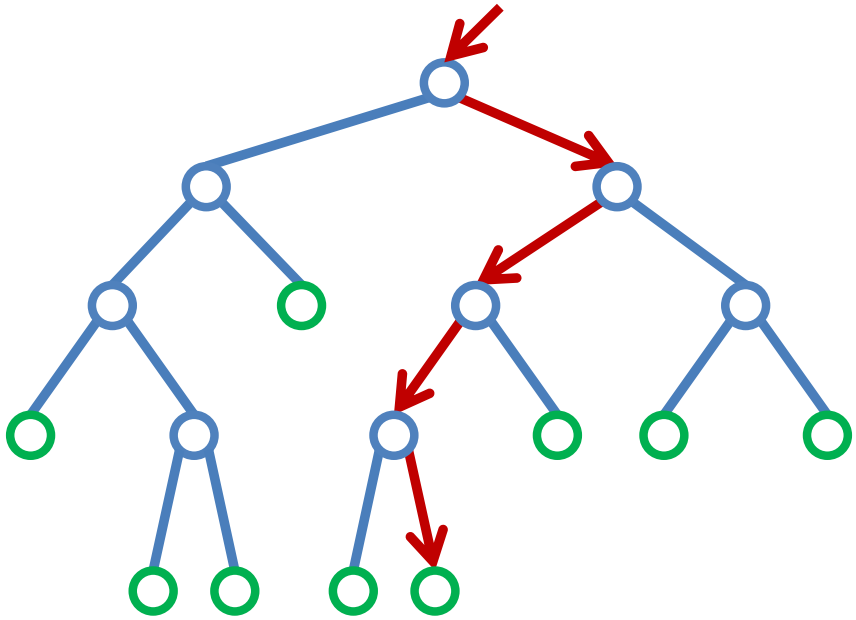
Samuel Rota Bulò
Fondazione Bruno Kessler
Trento, Italy
rotabulo@fbk.eu

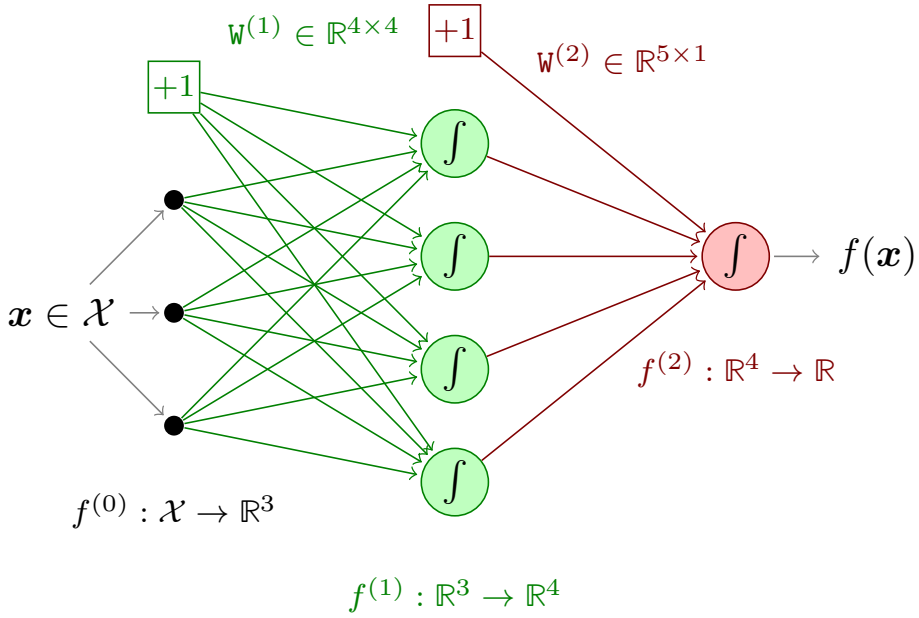
Peter Kontschieder
Microsoft Research
Cambridge, UK
pekontsc@microsoft.com

[Rota Bulo and Kontschieder, CVPR 2014]

Neural split functions

Classical random forest with neural split functions

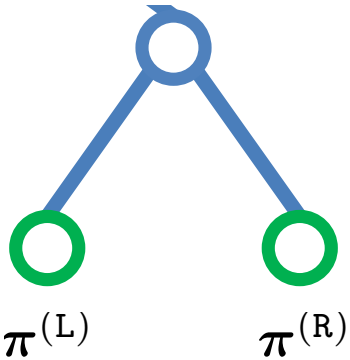




[Rota Bulo and Kontschieder, CVPR 2014]

Learning the neural split function (1)

Probabilistic loss function:



$$Q(\Theta) = \max_{\boldsymbol{\pi}} \mathbb{P}[\boldsymbol{y}|\mathbf{X}, \boldsymbol{\pi}, \Theta] \,,$$

Labels — Node input samples — Latent distributions of labels routed to left and right child nodes — Network parameters

$$\boldsymbol{\pi} = (\boldsymbol{\pi}^{(\mathrm{L})}, \boldsymbol{\pi}^{(\mathrm{R})})$$

Samples are independent

$$\boldsymbol{\pi}, \Theta] = \prod_{s=1}^{\mathrm{n}} \mathbb{P}[y_s|\boldsymbol{x}_s, \boldsymbol{\pi}, \Theta] \,,$$

$$\mathbb{P}[y_s|\boldsymbol{x}_s, \boldsymbol{\pi}, \Theta] = \sum_{d \in \{\mathrm{L},\mathrm{R}\}} \mathbb{P}[y_s|\psi_s = d, \boldsymbol{\pi}] \mathbb{P}[\psi_s = d|\boldsymbol{x}_s, \Theta]$$

$$= \sum_{d \in \{\mathrm{L},\mathrm{R}\}} \pi_{y_s}^{(d)} f_d(\boldsymbol{x}_s|\Theta) \,.$$

[Rota Bulo and Kontschieder, CVPR 2014]

Learning the neural split function (2)

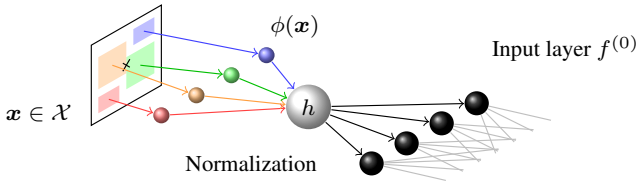
Learning procedure alternates between two steps :

$$Q(\Theta) = \max_{\boldsymbol{\pi}} \mathbb{P}[\boldsymbol{y}|\mathbf{X}, \boldsymbol{\pi}, \Theta] \,,$$

1. Update child distributions
2. Update network parameters (backprop)

[Rota Bulo and Kontschieder, CVPR 2014]

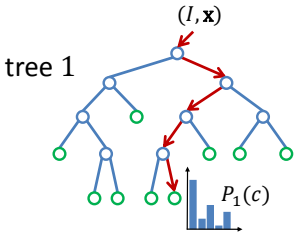
Results on semantic labelling



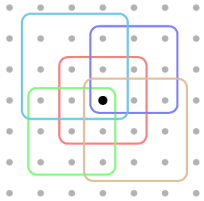
Method	ETRIMS8 Global	ETRIMS8 Class-Avg	ETRIMS8 Jaccard	CAMVID Global	CAMVID Class-Avg	CAMVID Jaccard
RF Baseline	64.5 ±1.6	59.6 ±1.7	40.3 ±1.1	64.0	41.6	27.2
NDF_{P}	69.8 ±1.8	64.3 ±2.2	45.0 ±1.9	67.4	46.5	30.8
NDF_{MLP}	68.9 ±2.0	62.4 ±2.3	44.2 ±2.1	67.1	44.4	30.1
NDF_{MLPC}	69.7 ±1.7	62.5 ±2.1	44.7 ±1.9	67.4	44.2	30.2
$\text{NDF}_{\text{MLPC}-\ell 1}$	71.7 ±2.0 (**+7.2**)	65.3 ±2.3 (**+5.7**)	46.9 ±2.0 (**+6.6**)	69.0 (**+5.0**)	46.8 (**+5.2**)	31.7 (**+4.5**)
RF Baseline	72.2 ±1.9	68.0 ±0.8	47.5 ±1.0	68.5	50.3	32.4
$\text{NDF}_{\text{MLPC}-\ell 1}$	80.8 ±0.7 (**+8.6**)	74.6 ±0.7 (**+6.6**)	56.9 ±1.2 (**+9.4**)	82.1 (**+13.6**)	56.1 (**+5.8**)	43.3 (**+10.9**)
Best RF in [13]	76.1	72.3	-	-	-	-
Best in [14]	75.1	72.4	-	-	-	-
Best RF in [19]	-	-	-	-	-	38.3
Best RF in [20]	-	-	-	72.5	51.4	36.4
Best in [8]	-	-	-	69.1	53.0	-
Best in [35]	-	-	-	73.7	36.3	29.6

[Rota Bulo and Kontschieder, CVPR 2014]

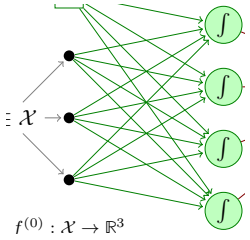
Types of random forests



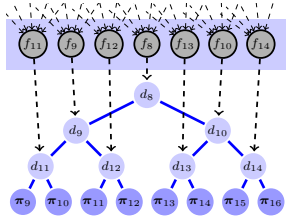
Classical random forests



Structured random forests



Neural random forests



Deep convolutional random forests

One model to rule them all …

Deep Neural Decision Forests

Peter Kontschieder[1] Madalina Fiterau[*,2] Antonio Criminisi[1] Samuel Rota Bulò[1,3]

Microsoft Research[1] Cambridge, UK
Carnegie Mellon University[2] Pittsburgh, PA
Fondazione Bruno Kessler[3] Trento, Italy

[Kontschieder et al., ICCV 2015]

Goals

Combine neural networks and random forests

Advantage of NN : representation learning

Advantage of RR : divide and conquer

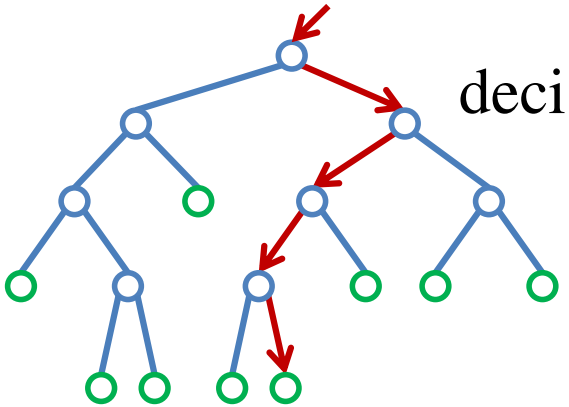
Différentiable loss function, allowing gradient backprop

« Backpropagation trees »

[Kontschieder et al., ICCV 2015]

Notation

sample $\boldsymbol{x} \in \mathcal{X}$



decision node $n \in \mathcal{N}$

prediction node $\ell \in \mathcal{L}$

decision function $d_n(\cdot;\Theta) : \mathcal{X} \rightarrow [0,1]$

Each prediction node $\ell \in \mathcal{L}$ holds a probability distribution $\boldsymbol{\pi}_\ell$ over $\mathcal{Y}$

[Kontschieder et al., ICCV 2015]

Stochastic decision functions

Decisions in split nodes are Bernouilli random variables!

$$\text{decision function } d_n(\cdot;\Theta) : \mathcal{X} \rightarrow [0,1]$$

$$\text{Bernoulli random variable with mean } d_n(\boldsymbol{x};\Theta)$$

Final prediction for sample x:

$$\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}] = \sum_{\ell\in\mathcal{L}} \pi_{\ell y}\mu_\ell(\boldsymbol{x}|\Theta)$$

where $\boldsymbol{\pi} = (\boldsymbol{\pi}_\ell)_{\ell\in\mathcal{L}}$ and $\pi_{\ell y}$ denotes the probability of a sample reaching leaf ℓ to take on class y, while $\mu_\ell(\boldsymbol{x}|\Theta)$ is regarded as the *routing function* providing the probability that sample $\boldsymbol{x}$ will reach leaf ℓ. Clearly, $\sum_\ell \mu_\ell(\boldsymbol{x}|\Theta) = 1$ for all $\boldsymbol{x} \in \mathcal{X}$.

[Kontschieder et al., ICCV 2015]

Stochastic decision functions

The routing function can be calculated with a single pass through the tree.

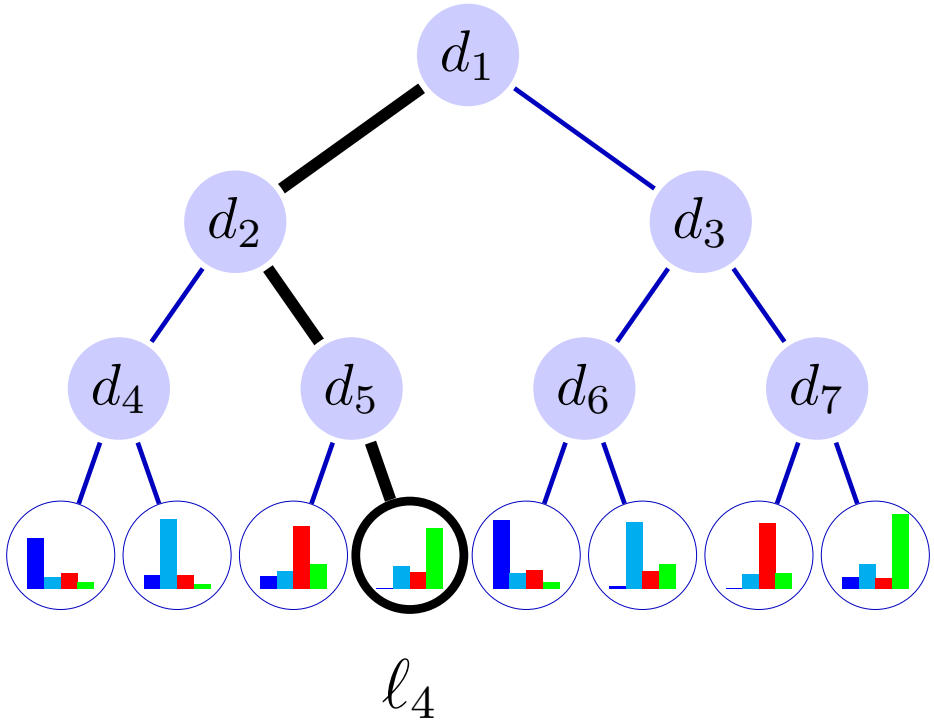
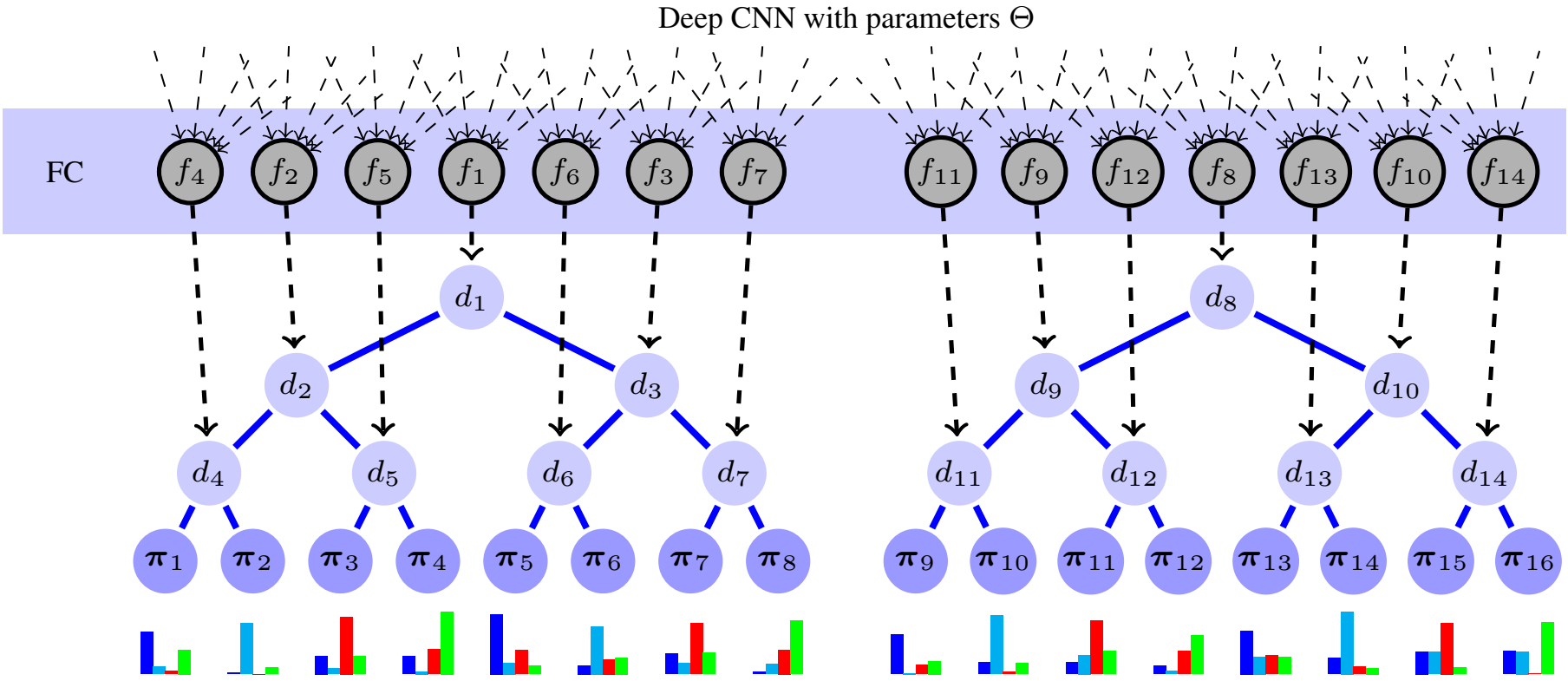


Figure 1. Each node $n \in \mathcal{N}$ of the tree performs routing decisions via function $d_n(\cdot)$ (we omit the parametrization Θ). The black path shows an exemplary routing of a sample $\boldsymbol{x}$ along a tree to reach leaf ℓ_4, which has probability $\mu_{\ell_4} = d_1(\boldsymbol{x})\bar{d}_2(\boldsymbol{x})\bar{d}_5(\boldsymbol{x})$.

[Kontschieder et al., ICCV 2015]

Illustration

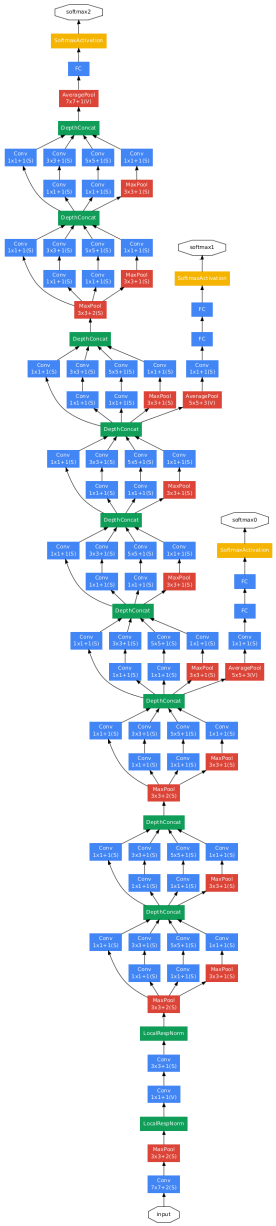


$$d_n(\boldsymbol{x};\Theta) = \sigma(f_n(\boldsymbol{x};\Theta))\,, \tag{3}$$

where $\sigma(x) = (1+e^{-x})^{-1}$ is the sigmoid function, and $f_n(\cdot;\Theta) : \mathcal{X} \to \mathbb{R}$ is a real-valued function depending

[Kontschieder et al., ICCV 2015]

The deep network used: GoogleNet



[Kontschieder et al., ICCV 2015]

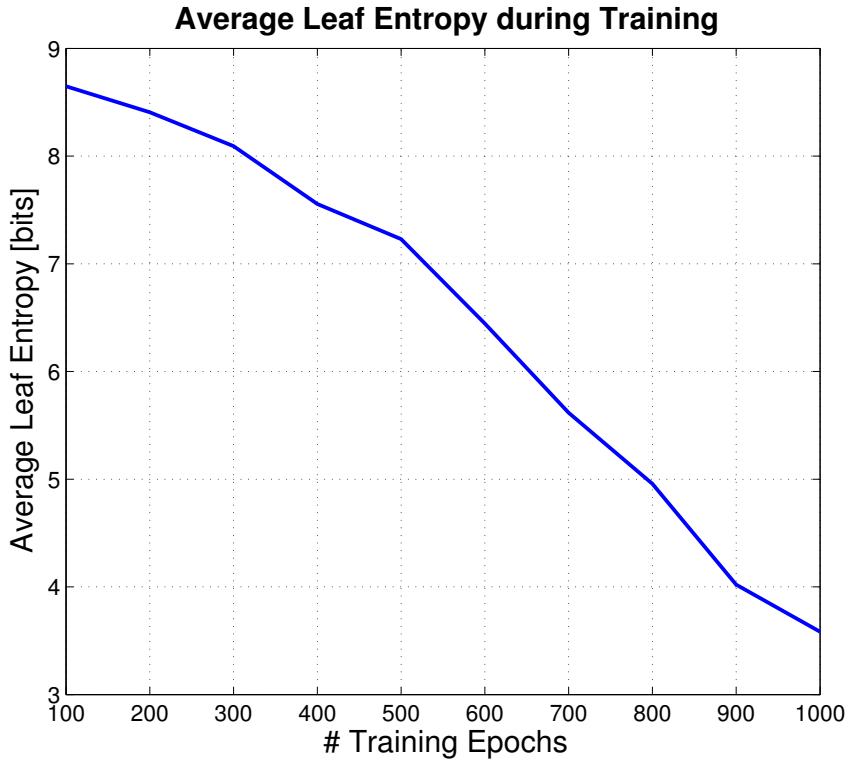
Résultats sur ImageNet

	GoogLeNet [36]						GoogLeNet⋆	dNDF.NET		
# Models	1			7			1	1		7
# Crops	1	10	144	1	10	144	1	1	10	1
Top5-Errors	10.07%	9.15%	7.89%	8.09%	7.62%	6.67%	10.02%	7.84%	7.08%	**6.38%**

Table 2. Top5-Errors obtained on ImageNet validation data, comparing our dNDF.NET to GoogLeNet(⋆).

[Kontschieder et al., ICCV 2015]

Leaf entropy during Training



[Kontschieder et al., ICCV 2015]

Results on ImageNet

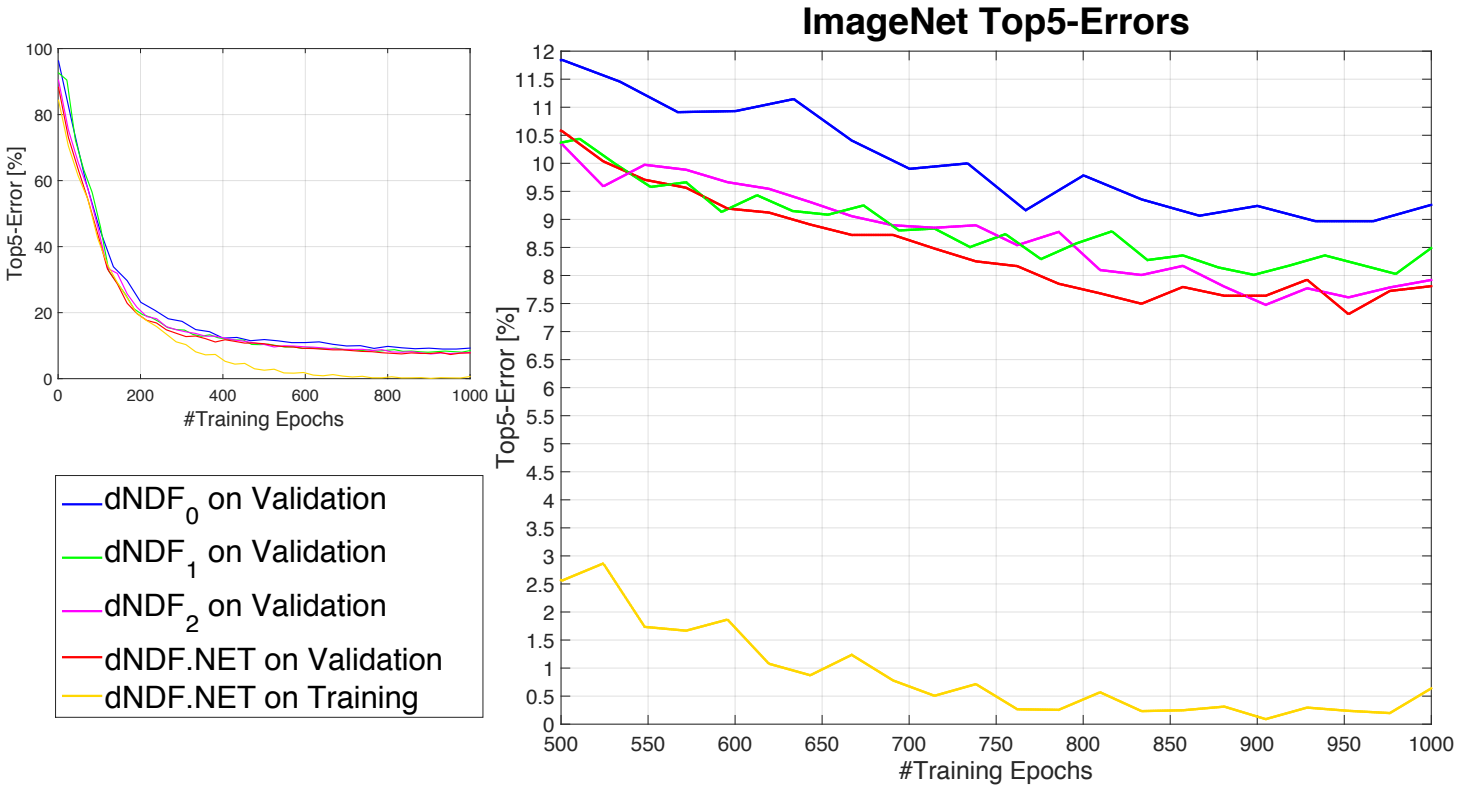


Figure 5. Top5-Error plots for individual dNDF_x used in dNDF.NET as well as their joint ensemble errors. Left: Plot over all 1000 training epochs. Right: Zoomed version of left plot, showing Top5-Errors from 0–12% between training epochs 500-1000.

[Kontschieder et al., ICCV 2015]

Conclusion

- Deep networks are still the number one model in terms of prediction performance
- Random forests still have excellent computational complexity during testing
- Combining both families is not an easy compromise
- Possibility of having classical « crisp » trees under a convolutional space displacement layer : speed!!