Mapping Stacked Decision Forests to Deep and Sparse Convolutional Neural Networks for Semantic Segmentation

David L Richmond¹, Dagmar Kainmueller¹, Michael Y Yang², Eugene W Myers¹, and Carsten Rother²

¹MPI-CBG, ²TU Dresden

Presented by Daniela Massiceti
Reading Group - January 2016
Random Forests vs Neural Networks

• RFs:
  – Easy to train; multi-class; training requires little data
  – Non-differentiable; each tree trained greedily

• NNs:
  – End-to-end learning of all parameters; feature learning
  – Training requires a lot of data & time

• Goal: bridge gap between RFs & NNs
Sethi and Welbl’s mapping

\[ x = (x_1, x_2, x_3) \]

\[ p(c|x) = y^\text{leaf}(x) \]

\[ n \in N^{\text{split}} \]
\[ l \in N^{\text{leaf}} \]
\[ x \in X_n \]
\[ x_f(n) < \theta(n) \]

\[ H1 : a(H_1(n)) = a(\omega_{01} x_n - \theta(n)) \]
\[ H2 : a(H_2(l)) = a(\omega_{12} a(H_1(n)) + b_{H_2}(l)) \]

\[ a(H_2(\text{leaf}(x))) = 1, \text{ all others} = 0 \]
Sethi and Welbl’s mapping

\( p(c|x) = \frac{p_c^{leaf(x)}}{\sum_{c=1}^C p_c^{leaf(x)}} \)

\( p(c|x) = \frac{\exp(p_c^{leaf(x)})}{\sum_{c=1}^C \exp(p_c^{leaf(x)})} \)
Sethi and Welbl’s mapping

- Replicate T times for T trees in the RF
- Fully connect output layer
Relationship between RFs and CNNs

- RF with contextual features = special case of CNN with sparse convolutional kernels & no max pooling
- Difference:
  - RFs: pixel's feature vector is precomputed
  - CNNs: features learnt
Mapping forward: RF Stack to Deep CNN

\[
x = (x_1, x_2, x_3)
\]

\[
p(c|x) = y_c^{\text{leaf}(x)}
\]

INPUT

\[
\begin{align*}
H_1 & \quad 0 & 1 & 4 & b \\
H_2 & \quad 2 & 3 & 5 & 6 \\
H_3 & \quad x_1 & x_2 & x_3 & p_1 & p_2 & b \\
H_4 & \quad 7 & 8 & 11 & b \\
H_5 & \quad 9 & 10 & 12 & 13 \\
\end{align*}
\]

\[
\begin{align*}
& \{ w_{f(n)}, H_1(n) \} \\
& \{ w_{H_1(n)}, H_2(l) \} \\
& \{ w_{H_2(l)}, H_3(c) \} \\
& \{ w_{H_3(c)}, H_4(n) \} \\
& \{ w_{H_4(n)}, H_5(l) \} \\
& \{ w_{H_5(l)}, c \}
\end{align*}
\]

OUTPUT

\[
\begin{align*}
& \{ w_{f(n)}, H_1(n) \} \\
& \{ w_{H_1(n)}, H_2(l) \} \\
& \{ w_{H_2(l)}, H_3(c) \} \\
& \{ w_{H_3(c)}, H_4(n) \} \\
& \{ w_{H_4(n)}, H_5(l) \} \\
& \{ w_{H_5(l)}, c \}
\end{align*}
\]
Mapping back: Deep CNN to RF Stack

- **Map back #1**
  - Keep weights and bias between $H_1$ and $H_2$ fixed, and re-train by backprop
  - Trivial mapping back to original RF with updated thresholds and leaf distributions

- **Map back #2**
  - Distributed activation on $H_2 = \text{many leafs contributing to prediction}$
  - New leaf distributions:

\[
 z_x(c) := \sum_{l \in N^{\text{leafs}}} a_x(H_2(l)) \cdot \omega_{H_2(l),c} \\
 \hat{y}_c^l = \frac{1}{|\{x \in X^\text{train}: \text{leaf}(x) = l\}|} \sum_{x \in X^\text{train}: \text{leaf}(x) = l} z_x(c)
\]
Results

- Kinect Body Part Classification

Depth input  Ground truth  Stacked RF  Retrained CNN

82%  91%
Results

- Zebrafish Somite Classification

<table>
<thead>
<tr>
<th>Method</th>
<th>RF</th>
<th>FCN</th>
<th>CNN</th>
<th>MB1</th>
<th>MB2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dice score</td>
<td>0.60</td>
<td>0.18</td>
<td><strong>0.66</strong></td>
<td>0.59</td>
<td>0.63</td>
</tr>
</tbody>
</table>
Conclusions & comments

- Mapping from RF to CNN and back again
  - Works with minimal training data
  - Distributed activation on $H_2$ – soft DTs
  - Hand-selected features – representation learning
  - Weights between $H_1$ and $H_2$ fixed – Decision Jungles

- Methodology-focussed rather than results-focussed

- Map-backed RF – improved performance, fast evaluation at test time?