LOGICSEG: Parsing Visual Semantics with Neural Logic Learning and Reasoning

Liulei Li\textsuperscript{1, 2}, Wenguan Wang\textsuperscript{1}\textsuperscript{*}, Yang Yi\textsuperscript{1}
\textsuperscript{1} ReLER, CCAI, Zhejiang University \textsuperscript{2} ReLER, AAII, University of Technology Sydney

https://github.com/lingorX/LogicSeg/

In this document, we first provide the pseudo code of LOGICSEG in §A. We next show the detailed label hierarchy for each dataset in §B. In addition, we offer more qualitative results in §C. Finally, we discuss the limitations and border impact of our algorithm in §D and §E, respectively. To ensure reproducibility and foster future research, our full implementation will be released after acceptance.

A. Pseudo Code

To facilitate a comprehensive understanding of LOGICSEG, we provide pseudo code for the logic-induced inference (§3) of LOGICSEG in Algorithm S1 and Algorithm S2, respectively. It can be seen that all the message creation processes are implemented in matrix operation which can enjoy the acceleration of the parallel architecture of GPUs. The \textit{for-loop} is merely adopted in Algorithm S1 when normalizing or summarizing the prediction in a level-wise manner, with $O(n)$ time complexity.

B. Label Hierarchy

For Mapillary Vistas 2.0\cite{1} and Cityscapes\cite{2} datasets, we adopt the officially provided label hierarchies following\cite{3}. For Pascal-Part-108\cite{4}, we use the hierarchy defined in \cite{5, 6}. For ADE20K\cite{7}, we organize a three-level label hierarchy by considering the semantic relations between labels according to the WordNet\cite{8}. The detailed label hierarchies for each datasets are provided in Fig.S1-S4.

C. More Qualitative Comparison Result

We provide more visual results that compare LOGICSEG to Mask2Former\cite{9} and to DeeplabV3+\cite{10} in Fig.S5-S6 and Fig.S7-S8, respectively. It can be observed that LOGICSEG performs robust in hard cases and can consistently deliver more satisfying results compared with the baseline algorithms.

D. Limitation

Currently our algorithm is specifically designed for tree-shape label hierarchy. It is interesting to extend our algorithm to handle more complicated and real-world semantic structures, for example, parent classes sharing some child classes. We leave this as a part of our future work.

E. Border Impact

This paper contributes to research on intelligent scene understanding, and thus is expected to eventually benefit automatic driving, education, health care, and economic development of the human society. Moreover, although our algorithm is able to parse the hierarchical relations between semantic concepts and yields improved performance over current top-leading competitors, the relevant security measures still need to be erected and caution should always be exercised.
**Algorithm S1** Pseudo-code for logic-induced inference of LOGICSEG in a PyTorch-like style (Part I).

```python
***
T: index matrix indicates the hierarchy, for example:
\[
\begin{array}{cccc}
  & a & b & c \\
\hline
 a & 0 & 1 & 0 & 0 \\
 b & 0 & 0 & 0 & 1 \\
 c & 0 & 0 & 0 & 0
\end{array}
\]
\]
\]
P: matrix indicates the peer relation, for example:
\[
\begin{array}{cccc}
  & a & b & c \\
\hline
 a & 0 & 0 & 0 & 0 \\
 b & 0 & 0 & 0 & 0 \\
 c & 0 & 0 & 0 & 0
\end{array}
\]
\]
V: array stores the class number in each hierarchical level, \( V[1] = |V|^+1 \)
R: round of message passing
L: number of hierarchical level
s_k: grounded predicates (|V| \times HW)

```python
def message_passing(s_k):
    s_k += c_score(s_k) + d_score(s_k) + e_score(s_k)
    # hierarchical level-wise normalization
    n = 0
    for l in range(L, 0, -1):
        s_k[n:n+V[l-1]] = s_k[n:n+V[l-1]].softmax(dim=0)
        n += V[l]
    return s_k

def inference(s_k):
    # R times of message passing
    for t in range(R):
        # (N_p x |V| x l) x (1 x |V| x HW)
        s_f = T.unsqueeze(-1) * s_k.unsqueeze(0)
        n = V[L-1]
        t_s = s_f[:V[L-1]]
        # top-scoring path (Eq. 18)----------#
        for l in range(L-2, -1, -1):
            t_m = t_s.unsqueeze(1)
            # (|V|^l x |V| x HW) --> (|V|^l x HW)
            t_s = s_f[:V[n:V[l+1]] + t_m]
            # (|V|^l x |V| x HW) --&gt; (|V|^l x HW)
            t_m = t_s * (T[n:V[l+1]].unsqueeze(-1)).sum(0)
            n = V[l]
            # (|V|^l x HW)
            t_s = t_s[:n:n+V[l-1]]
        # (|V|^0 x HW) --&gt; (HW)
        pred = t_s.argmax(dim=0)
    return pred
***
```
Figure S1: Hierarchical label structure of Mapillary Vistas 2.0[1].
Figure S2: Hierarchical label structure of Cityscapes[2].
Figure S3: Hierarchical label structure of Pascal-Part-108[4].
Figure S4: Hierarchical label structure of ADE20K[7].
Figure S5: Visual comparison results on Mapillary Vistas 2.0[1] val. Top: Mask2Fomer[9] vs. Bottom: LOGICSEG
Figure S7: Visual comparison results on ADE20K\textsuperscript{7} val. \textit{Top}: DeepLabV3+[10] vs. \textit{Bottom}: LOGICSEG
Figure S8: Visual comparison results on Pascal-Part-108 test. Top: DeepLabV3+[10] vs. Bottom: LOGICSEG
References


