LOGICSEG: Parsing Visual Semantics with Neural Logic Learning and Reasoning

Liulei Li[1, 2], Wenguan Wang[1]*, Yang Yi[1]
[1] ReLER, CCAI, Zhejiang University [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 *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[1] and Cityscapes[2] datasets, we adopt the officially provided label hierarchies following[3]. For Pascal-Part-108[4], we use the hierarchy defined in [5, 6]. For ADE20K[7], we organize a three-level label hierarchy by considering the semantic relations between labels according to the WordNet[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[9] and to DeeplabV3+[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).

```
"""
T: index matrix indicates the hierarchy, for example:
          a                a b c d e f
         / \             a 0 1 1 0 0 0
        b   c    -->     b 0 0 0 1 1 0
       /\    \           c 0 0 0 0 0 1
      d e    f
P: matrix indicates the peer relation, for example:
                         a b c d e f
          a              a 0 0 0 0 0 0
         / \             b 0 0 1 0 0 0
        b   c    -->     c 0 1 0 0 0 0
       /\    \           d 0 0 0 0 1 1
      d e    f           e 0 0 0 1 0 1
                         f 0 0 0 1 1 0
V: array stores the class number in each
   hierarchical level, V[l] = |V^{l+1}|
R: round of message passing
L: number of hierarchical level
s_k: grounded predicates (|V| x HW)
"""

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]] = s_k[n:n+V[l]].softmax(dim=0)
        n += V[l]

    return s_k

def inference(s_k):
    # R times of message passing
    for t in range(R):
        s_k = message_passing(s_k)

    # (N_p x |V| x 1)*(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_s = t_s.unsqueeze(1)
        # (|V^l| x |V| x HW) + (|V^l| x 1 x HW)
        t_s = (s_f[n:n+V[l]] + t_s)
        # (|V^l| x |V| x HW) --> (|V| x HW)
        t_s = t_s*(T[n:n+V[l]].unsqeeze(-1)).sum(0)
        n += V[l]
        # (|V^{l-1}| x HW)
        t_s = t_s[n:n+V[l-1]]
    # (|V^1| x HW) --> (HW)
    pred = t_s.argmax(dim=0)

    return pred
```

Algorithm S2 Pseudo-code for logic-induced inference of LOGICSEG in a PyTorch-like style (Part II).

```
"""
N_p: class number of non-leaf nodes
s_k: grounded predicates (|V| x HW)
"""
def c_score(s_k):
    #-------------C-message (Eq. 16)-------------#
    # (N_p x |V| x 1)*(1 x |V| x HW)
    c_f = T.unsqueeze(-1) * s_k.unsqueeze(0)
    # (N_p x 1 x HW)*(N_p x |V| x HW)
    c_m = s_k[:N_p].unsqueeze(1) * c_f
    # 1-s_k[v]+s_k[v]·s_k[p_v]
    c_m = 1 - c_f + c_m

    #-----gather received C-messages (Eq. 17)----#
    # (N_p x HW)
    c_s = (c_f * c_m).sum(dim=1)
    c_s = c_s / T.sum(dim=1).unsqueeze(-1)
    # (|V| x HW)
    c = torch.zeros(|V|, HW)
    c[:N_p, :] = c_s

    return c

def d_score(s_k):
    #-------------D-message (Eq. 16)-------------#
    # (N_p x |V| x 1)*(1 x |V| x HW)
    d_f = T.unsqueeze(-1) * s_k.unsqueeze(0)
    # (N_p x HW)*(N_p x HW)
    d_m = s_k[:N_p] * d_f.max(dim=1)
    # 1-s_k[v]+s_k[v]·max({s_k[c_v^n]}_n)
    d_m = 1-s_k[:N_p] + d_m

    #-----gather received D-messages (Eq. 17)----#
    # (N_p x HW)x(N_p x HW)
    d_s = s_k[:N_p] * d_m
    # (N_p x 1 x HW)*(N_p x |V| x 1)
    d_s = d_s.unsqueeze(1) * T.unsqueeze(-1)
    # (|V| x HW)
    d_s = d_s.sum(dim=0)

    return d_s

def e_score(s_k):
    #-------------E-message (Eq. 16)-------------#
    # (|V| x |V| x 1)*(1 x |V| x HW)
    e_f = P.unsqueeze(-1) * s_k.unsqueeze(0)
    # (|V| x 1 x HW)*(|V| x |V| x HW)
    e_m = s_k.unsqueeze(1) * e_f
    # - (1-1/M sum_{m=1}^M s_k[v]·s_k[a_v^m])
    e_m = -1+e_m.sum(dim=1)/P.sum(dim=1).unsqueeze(-1)

    #-----gather received E-messages (Eq. 17)----#
    # (|V| x HW)
    e_s = e_f.sum(dim=1) / P.sum(dim=1).unsqueeze(-1)
    # E-message should be same for all nodes in the
        same hierarchical level
    e_s = e_m * e_s

    return e_s
```

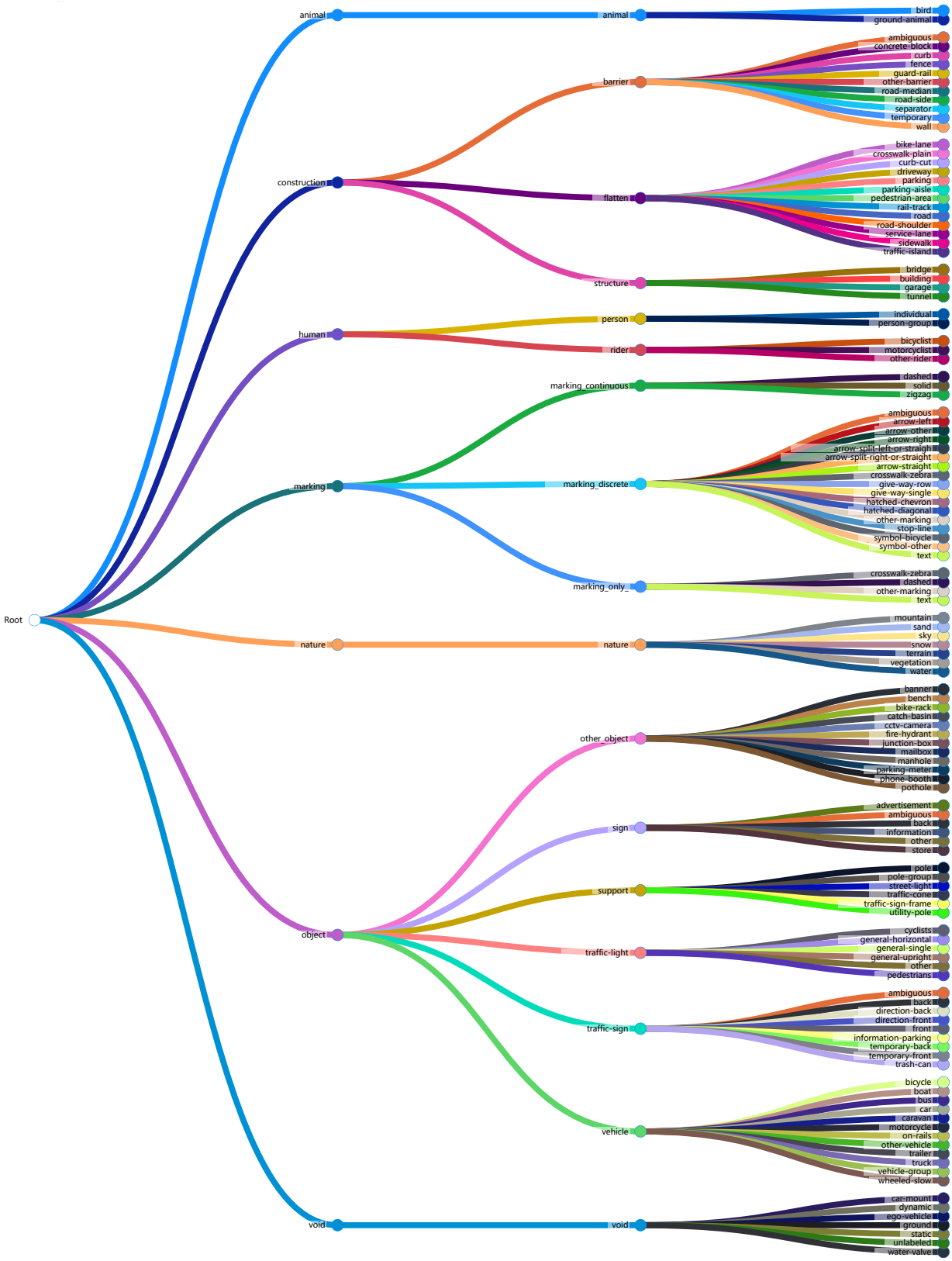

Figure S1: **Hierarchical label structure** of Mapillary Vistas 2.0[1].

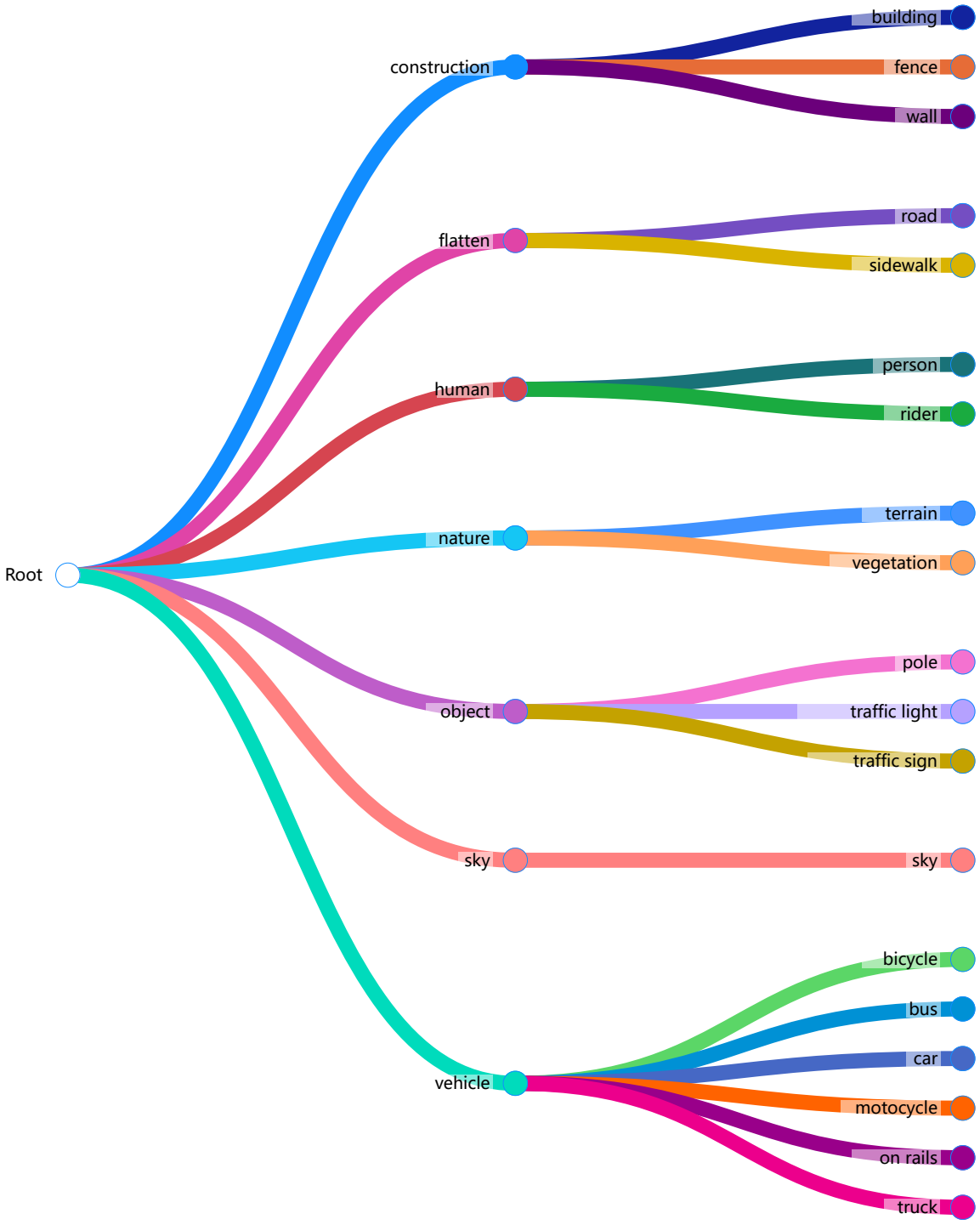

Figure S2: **Hierarchical label structure** of Cityscapes [2].

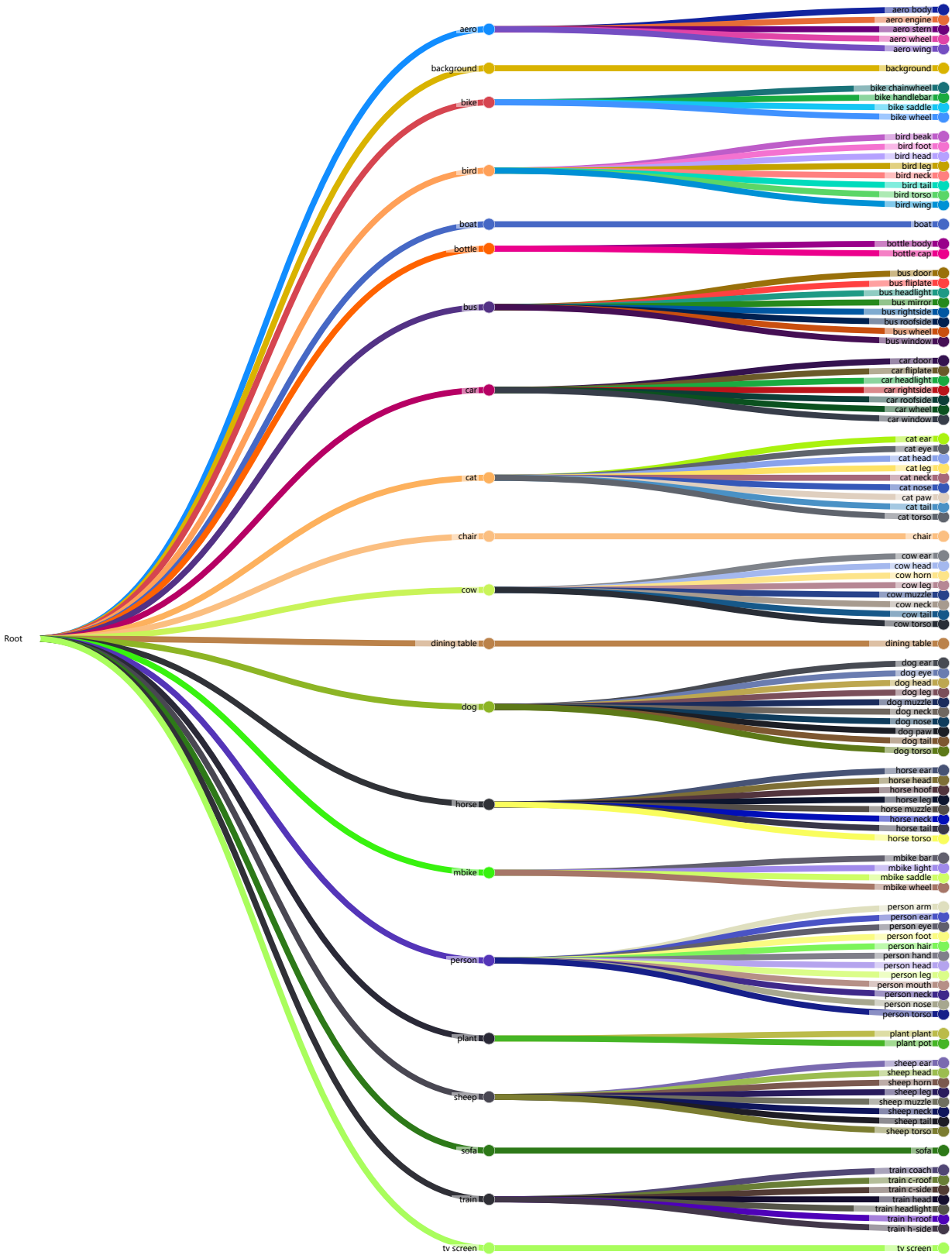

Figure S3: **Hierarchical label structure** of Pascal-Part-108 [4].

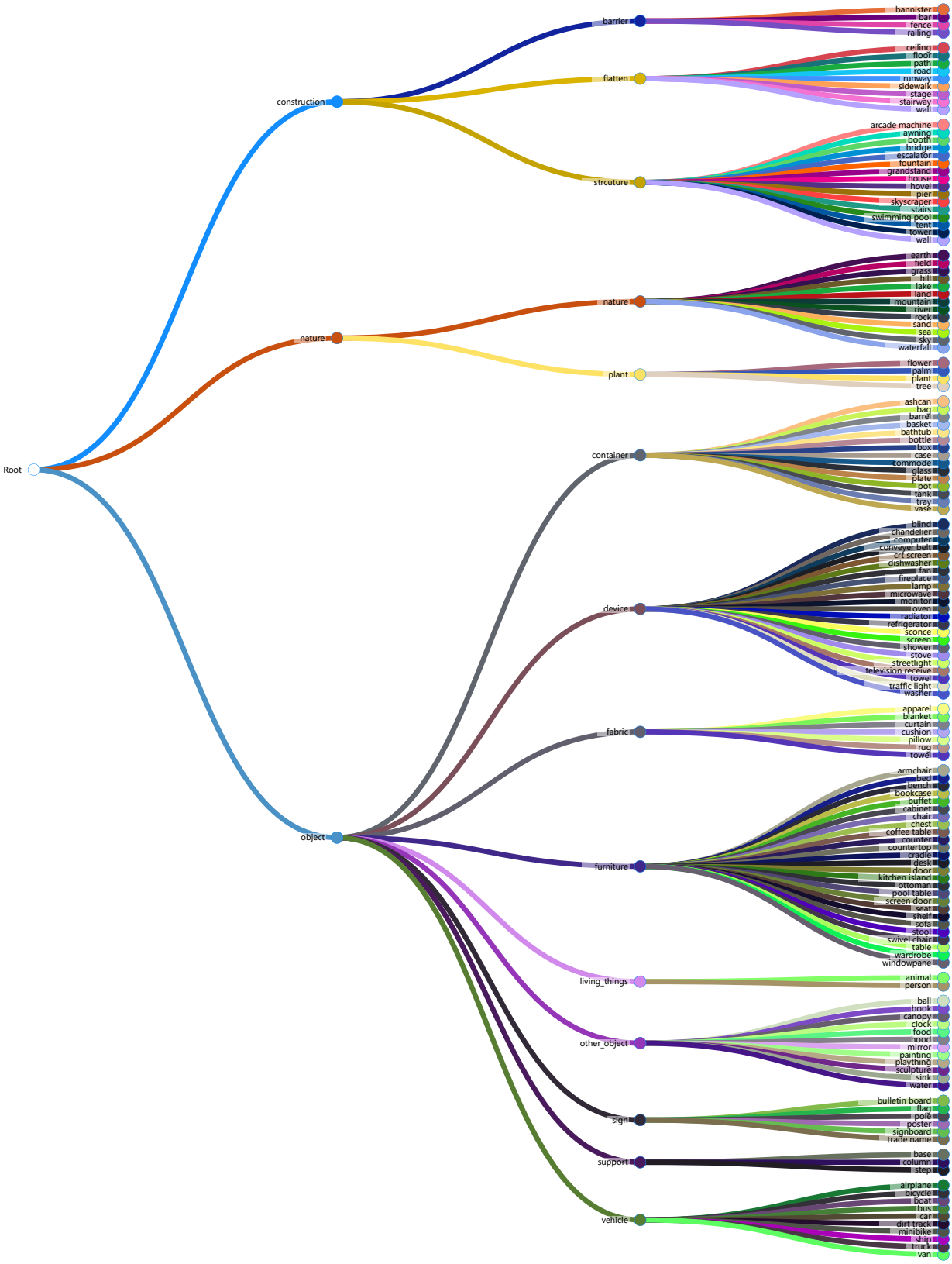

Figure S4: **Hierarchical label structure** of ADE20K[7].

Figure S5: **Visual comparison results** on Mapillary Vistas 2.0[1] `val`. *Top*: Mask2Fomer[9] *vs*. *Bottom*: LOGICSEG

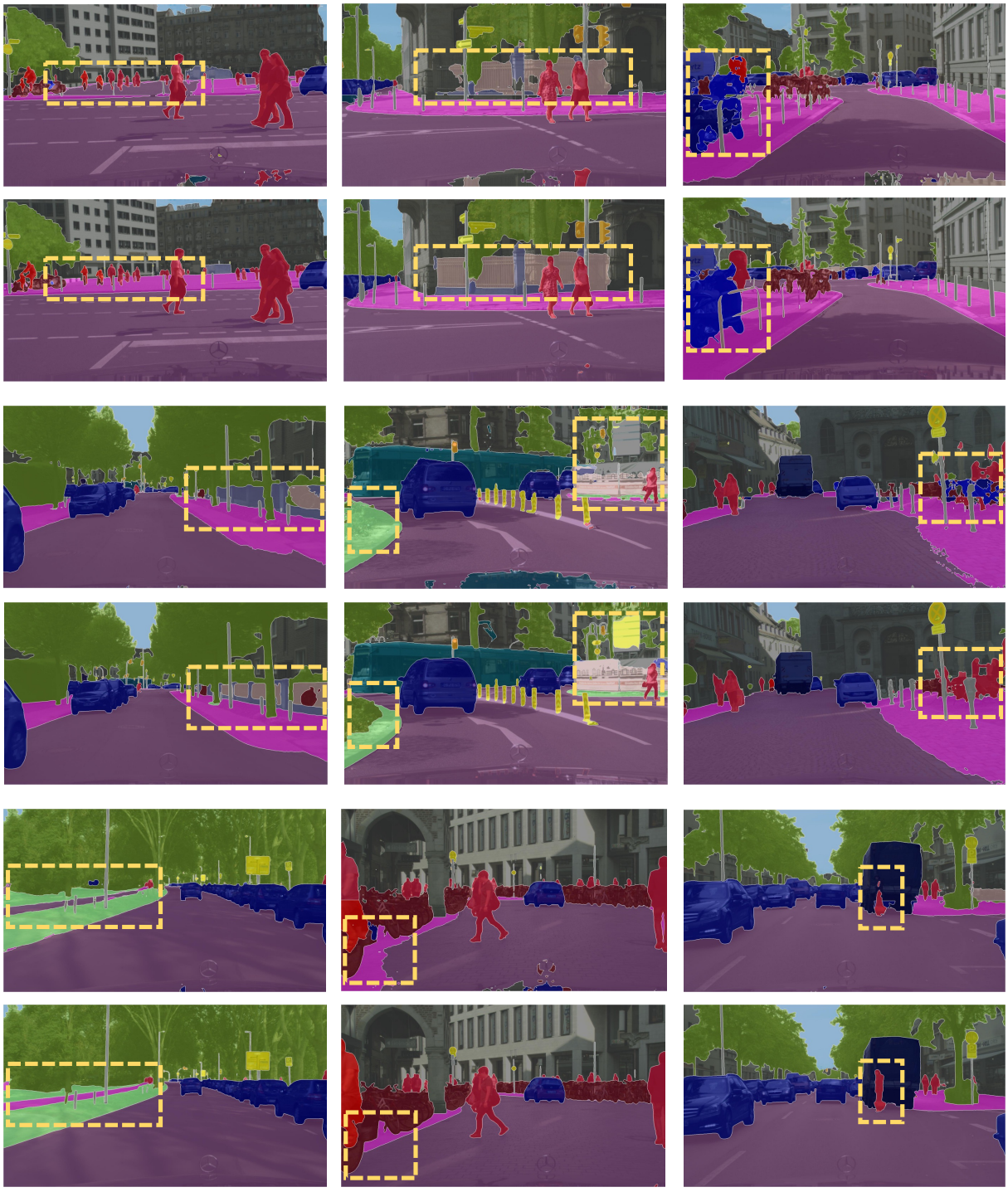

Figure S6: **Visual comparison results** on Cityscapes[2] `val`. *Top*: Mask2Fomer[9] *vs*. *Bottom*: LOGICSEG

Figure S7: **Visual comparison results** on ADE20K[7] `val`. *Top*: DeepLabV3+[10] *vs*. *Bottom*: LOGICSEG

Figure S8: **Visual comparison results** on Pascal-Part-108[4] `test`. *Top*: DeepLabV3+[10] *vs*. *Bottom*: LOGICSEG

References

[1] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. S1, S3, S7

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. S1, S4, S8

[3] Liulei Li, Tianfei Zhou, Wenguan Wang, Jianwu Li, and Yi Yang. Deep hierarchical semantic segmentation. In *CVPR*, 2022. S1

[4] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014. S1, S5, S10

[5] Umberto Michieli, Edoardo Borsato, Luca Rossi, and Pietro Zanuttigh. Gmnet: Graph matching network for large scale part semantic segmentation in the wild. In *ECCV*, 2020. S1

[6] Rishubh Singh, Pranav Gupta, Pradeep Shenoy, and Ravikiran Sarvadevabhatla. Float: Factorized learning of object attributes for improved multi-object multi-part scene parsing. In *CVPR*, 2022. S1

[7] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. S1, S6, S9

[8] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. S1

[9] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. S1, S7, S8

[10] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. S1, S9, S10