Hierarchical Losses for Semantic Segmentation

BMVC 2019 Submission # 645

Abstract

In this paper we exploit knowledge of class hierarchies to aid the training of semantic segmentation convolutional neural networks. We do not modify the architecture of the network itself, but rather propose to compute a loss that is a summation of classification losses at different levels of class abstraction. This allows the network to differentiate serious errors (the wrong superclass) from minor errors (correct superclass but incorrect finescale class) and to learn visual features that are shared between classes that belong to the same superclass. The method is straightforward to implement (we provide a PyTorch implementation that can be used with any existing semantic segmentation network) and we show that it yields performance improvements (faster convergence, better mean IOU) relative to training with a flat class hierarchy and exactly the same CNN architecture. We provide results for the Helen face segmentation dataset and Mapillary Vistas road scene segmentation dataset.

1 Introduction

The visual world is full of structure, from relationships between objects to scenes and objects composed of hierarchical parts. For example, at the most abstract level, a road scene could be segmented into three parts: ground plane, objects on the ground plane and the sky. The next finer level of abstraction might differentiate the ground plane into road and pavement, then the road into lanes, white lines and so on. Human perception exploits this structure in order to reason abstractly without having to cope with the deluge of information when all features and parts are considered simultaneously. Moreover, it is quite easy for a human to describe this structure in a consistent way and to reflect it in annotations or labels. It is therefore surprising that the vast majority of work on learning-based object recognition, object detection, semantic segmentation and many other tasks completely ignores this structure. Classification tasks are usually solved with a flat class hierarchy, but in this paper we seek to make use of this structure. Another motivation is that there is often inhomogeneity between datasets in terms of labelling. For example, the LFW parts label database [5] segments face images into background, hair and skin, while the segment annotations [15] for the Helen dataset [6] define 11 segments. Utilising both datasets to train a single network, while retaining the richness of the labels in the latter one, is not straightforward. Depending on the application, we may also wish to be able to vary the fineness of labels provided by the same network.

In this paper, we tackle the problem of semantic segmentation and introduce the idea of hierarchical classification loss functions. The idea is very simple. Any existing semantic

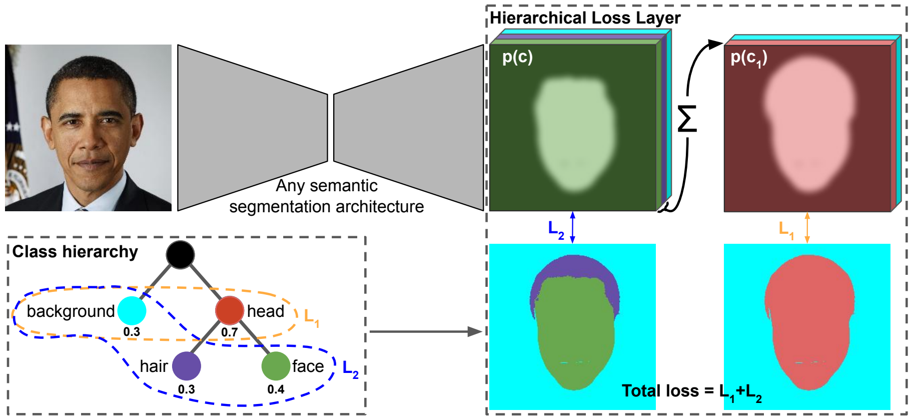

Figure 1: Overview of our idea. Given the output of any semantic segmentation architecture and a class hierarchy, we compute losses for each level of abstraction within the hierarchy, inferring probabilities of superclasses from their children.

segmentation architecture that outputs one logit per class per pixel (and which would conventionally be trained using a single classification loss such as cross entropy) can compute a sum of losses at each level of abstraction within the class hierarchy. The benefit is to differentiate serious errors from less serious. In the toy example shown in Figure 1, a mistake between *face* and *hair* would be penalised less severely than a mistake between *background* and *face* since both *face* and hair belong to the superclass *head* and so L1 would not penalise the error. This encourages the network to learn visual features that are shared between classes belonging to the same superclass, i.e. the knowledge conveyed by the class hierarchy allows the network to exploit regularity in appearance. Since coarser classification into fewer abstract classes is presumably simpler than finescale classification, it also means that the learning process can naturally proceed in a coarse to fine manner, learning the more abstract classes earlier. The approach is very simple to implement, requiring only a few lines of code to compute the hierarchical losses once the tree has been constructed.

2 Related work

Semantic segmentation The state-of-the-art in semantic segmentation has advanced rapidly in the last 5 years thanks to end-to-end learning with fully convolutional networks [7]. The field is large, so we provide only a brief summary here. A landmark paper was SegNet [1] which utilised an encoder-decoder architecture with skip connections with a novel unpooling operation for upsampling. Wu *et al.* [17] recently explored extensions of this architecture. Güçlü *et al.* [4] obtain state-of-the-art face semantic segmentation by augmenting a CNN with conditional random fields (CRFs) and an adversarial loss. Ning *et al.* [12] achieve very fast performance using hierarchical dilation and feature refinement. They exploit bypass and intermediate supervision in the upsample steps to help refine coarse features. Rota Bulò *et al.* [14] achieve state-of-the-art road scene semantic segmentation performance. They combine a DeepLabv3 head with a wideResNExt body and propose a special form of activated batch normalisation which saves memory. Zhao *et al.* [19] proposed Pyramid Scene Parsing Network (PSPNet) which utilises a new architecture module to capture contextual information. Chen *et al.* [2] extend DeepLabv3 by adding a decoder module based on atrous separable convolution. None of these methods exploit scene structure provided by class hierarchies.

Hierarchical methods Existing hierarchy-based methods have focused on *hierarchical architectures*, i.e. methods that specifically adapt the architecture of the network to the specific hierarchy for a particular task. This is typified by Branch-CNN [20] and Hierarchical Deep CNN (HD-CNN) [18] in which a network architecture is constructed to reflect the classification hierarchy. Luo *et al.* [8] use face and component detections to constrain face segmentation. Deng *et al.* [3] encode class relationships in a Hierarchy and Exclusion (HEX) graph. This enables them to reason probabilistically about label relations using a CRF. While very powerful, this also makes inference on their model more expensive and defining a HEX graph requires richer information than the straightforward tree-based hierarchies that we use. Srivastava and Salakhutdinov [16] similarly take a probabilistic approach and attempt to learn a hierarchy as part of the training process. This does not necessarily reflect the semantic class structure but is rather chosen to optimise subsequent classification performance. Again, inference is more complex and expensive. Meletis *et al.* [9] use a restricted set of classifiers in a hierarchical fashion on the output of a standard deep learning architecture to harness differing levels of semantic description.

3 Hierarchical loss

Our method is based on computing a sum of classification losses over each level of abstraction within a classification hierarchy. In order to use the approach, one simply needs a class hierarchy defined by a tree and a segmentation architecture that outputs a classification for each of the classes corresponding to leaf nodes in the tree. In this section we describe our representation and how the losses at each level in the hierarchy are computed.

Tree-based representation We represent our class hierarchy using a tree (V,E), with $V = \{v_1,\dots,v_n\}$ the set of vertices and $E \subset V \times V$ the set of ordered edges such that $(v_i,v_j) \in E$ encodes that v_i is a parent of v_j. We assume that the first m numbered nodes correspond to leaves in the tree, i.e. $\nexists v_j \in V, (v_i,v_j) \in E \Rightarrow 1 \leq i \leq m$. These nodes correspond to the finest scale classes. If $(v_i,v_j) \in E$ then v_i is a more general, more abstract, superclass of v_j. The label for a pixel, c, should be at the finest level of classification, i.e. $c \in \{1,\dots,m\}$.

We define $\text{depth}(v_i)$ to mean the number of edges between vertex v_i and the root node. Hence, the depth of the tree is given by $D_{\max} = \max_i \text{depth}(v_i)$. We define $\text{ancestor}(v_i,v_j)$ to be true if v_i is an ancestor of v_j, i.e. that v_i is a superclass of v_j and false otherwise.

Inferring coarse classes from fine We assume that the segmentation CNN outputs one logit, x_i, per pixel per leaf node, i.e. the output of the network is of size $H \times W \times m$. Hence, the probability, p_i, associated with node i can be computed by applying the softmax function, $\sigma : \mathbb{R} \mapsto [0,1]$, to x_i. The probability associated with non-leaf nodes is defined recursively by summing the probabilities of its children until leaf nodes are reached:

$$p_i = \begin{cases} \sigma(x_i) & \text{if } 1 \leq i \leq m \\ \sum\limits_{(v_i,v_j)\in E} p_j & \text{otherwise} \end{cases} . \quad (1)$$

```
background
foreground
   hair
   face
      mouth
         upper_lip
         inner_mouth
         lower_lip
      skin
         face_skin
         nose
      eyes
         left_eye
         right_eye
      eyebrows
         left_eyebrow
         right_eyebrow
```

```
level_loss_list

D=4:
   [ [background],[hair],[upper_lip],[inner_mouth],[lower_lip],[face_skin],[nose],
     [left_eye],[right_eye],[left_eyebrow],[right_eyebrow] ]

D=3:
   [ [background],[hair],[upper_lip],[inner_mouth],[lower_lip],[face_skin,nose],
     [left_eye,right_eye],[left_eyebrow,right_eyebrow] ]

D=2:
   [ [background],[hair],[upper_lip,inner_mouth,lower_lip,face_skin,nose,
     left_eye,right_eye,left_eyebrow,right_eyebrow] ]

D=1:
   [ [background],[hair,upper_lip,inner_mouth,lower_lip,face_skin,nose,
     left_eye,right_eye,left_eyebrow,right_eyebrow] ]
```

Figure 2: Class hierarchy implementation: the classification hierarchy is provided as a text file (left), we precompute lists of leaf nodes (right) for each depth corresponding to the summations required for computing internal node probabilities.

Depth dependent losses The appropriate label varies depending on the level of abstraction, i.e. the depth considered in the tree. We define the correct class at a depth $d \in \{0,\dots,D_{\max}\}$ to be:

$$c_d = \begin{cases} c & \text{if } \operatorname{depth}(v_c) \le d \\ i, \operatorname{depth}(v_i) = d \wedge \operatorname{ancestor}(v_i, v_c) & \text{otherwise} \end{cases} . \tag{2}$$

The classification loss at depth d is computed using the negative log loss as $L_d = -\log(p_{c_d})$. Note that $L_0 = 0$. The total hierarchical loss is then a summation over all depths in the tree: $L = \sum_{d=0}^{D_{\max}} L_d$.

4 Implementation

In this section we describe implementation details for our hierarchical loss function. Our implementation is written using standard PyTorch layers and we make it publicly available in a form that is easy to integrate with any existing semantic segmentation architecture (URL removed for review). We also explain how to ensure numerical stability in the computation of these losses.

Tree structure The hierarchical class structure is provided as a text file using indentation to signify parent/child relationships (see Figure 2 left for an example). When we load this, we internally store the tree as a structure of linked python objects. Each leaf-node in the hierarchy represents a class in our dataset. In addition, each leaf node encapsulates an integer representing the channel it will use in the output of the neural network. The numbering between class labels in the training data and these node numbers must be consistent. To implement our hierarchical loss efficiently, we precompute lists representing the set of leaf node classes that belong to each superclass (see Figure 2 right for an example). Evaluating the loss for a given depth is achieved using the PyTorch code snippet in Listing 1. The output from the CNN is deep copied for each depth in the tree to compute the separate losses. Each level loss list represents the branches with which we need to sum probabilities. We use the level loss list for a particular tree level to extract the softmaxed CNN output channels to be summed over (lines 2 to 10 of Listing 1). Every summed slice of a branch is inserted into each channel associated with that branch for ease of implementation (lines 12 and 13 of Listing 1). Finally to compute the loss for each level, we take the log of the summed probability tensor, which is passed to the negative log likelihood loss layer (nll_loss in PyTorch) along with labels.

```
summed_probabilities = sm_cnn_out.clone()
for branch in level_loss_list:
    initial = True
    for chan in branch:
        if initial:
            initial = False
            sli = sm_cnn_out[:,chan,:,:].unsqueeze(1)
        else:
            sli = torch.cat((sli,sm_cnn_out[:,chan,:,:].unsqueeze(1)),1)
    summed_branch_slice = sli.sum(1, keepdim=True)

    for chan in branch:
        summed_probabilities[:,chan:(chan+1),:,:] = summed_branch_slice
```

Listing 1: PyTorch code for computing the inferred probabilities on a single level of a hierarchy. The hierarchy of classes at each level is represented by a `level_loss_list`. Note `sm_cnn_out chan` and `sli` are short for softmaxed CNN output-tensor class-channel and slice respectively.

Numerical stability Evaluating cross entropy (log) loss of a probability computed using the softmax function is numerically unstable and can easily lead to overflow or underflow. In most implementations, this is circumvented using the "log-sum-exp trick" [10]:

$$L = -\log p_i = -\log\left(\frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)}\right) = -x_i + b + \log\sum_{j=1}^{n}\exp(x_j - b), \tag{3}$$

where $b = \max_{i\in\{1,\dots,n\}} x_i$ is chosen such that the maximum exponential has value one and thus avoids overflow, while also ensuring that at least one summand will avoid underflow and hence avoid taking a logarithm of zero.

Our hierarchical classification losses involve computing log losses on internal nodes in the class hierarchy tree. The probabilities in these nodes are in turn formed by summing probabilities computed by applying softmax to CNN outputs. This leads to evaluation of losses of the form $-\log(\sum_{i\in\mathcal{C}} p_i)$ where $\mathcal{C}$ is the set of leaf nodes contributing to the super-class. This can be made numerically stable by double application of the log-sum-exp trick:

$$L = -\log\sum_{i\in\mathcal{C}} p_i = -\log\frac{\sum_{i\in\mathcal{C}}\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)} = b + \log\sum_{j=1}^{n}\exp(x_j - b) - b_{\mathcal{C}} - \log\sum_{i\in\mathcal{C}}\exp(x_i - b_{\mathcal{C}}). \tag{4}$$

b is defined as before while $b_{\mathcal{C}} = \max_{i\in\mathcal{C}} x_i$. The use of two different shifts for the two logarithm terms is required to avoid underflow when $b_{\mathcal{C}} \ll b$.

5 Experiments

We seek to investigate the relative performance gain in using the hierarchical loss versus training simply on a flat hierarchy. To this end, in our experiments we train two networks

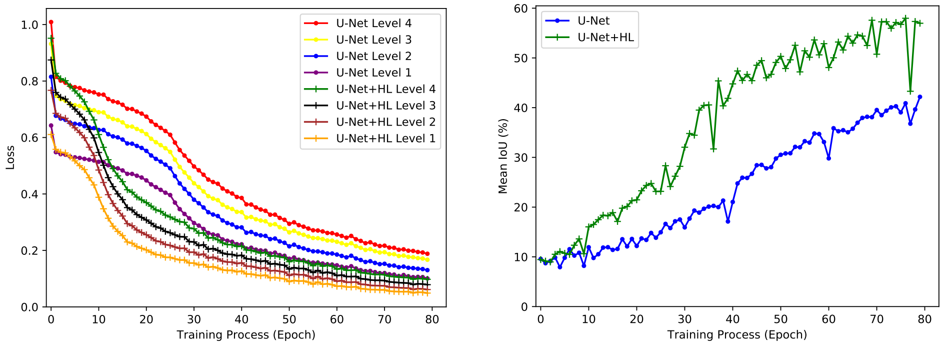

Figure 3: Helen leaning curves: Level losses (left) and IOU (right) as models are trained with vanilla (U-Net) and hierarchical (U-Net+HL) losses.

	50 epochs		200 epochs	
Class	U-Net	U-Net+HL	U-Net	U-Net+HL
Background	90.17	91.66	92.04	92.63
Face skin	76.70	84.02	86.53	87.04
Left eyebrow	0.0	17.84	63.12	62.68
Right eyebrow	0.0	21.25	63.65	64.25
Left eye	3.18	28.68	63.98	67.81
Right eye	3.29	29.18	64.92	72.72
Nose	63.75	63.79	84.05	82.55
Upper lip	4.8	41.59	52.88	56.48
Inner mouth	0.91	50.17	62.17	67.94
Lower lip	17.54	46.13	65.64	67.92
Hair	57.89	64.58	65.41	66.11
Mean	28.93	48.99	69.49	71.65

Table 1: IOU (%) on the Helen dataset.

	1 epoch		80 epochs	
Class	U-Net	U-Net+HL	U-Net	U-Net+HL
Car	12.14	25.75	80.35	80.91
Terrain	12.99	27.53	54.21	56.07
Lane Marking	0.0	15.3	49.14	51.69
Building	28.9	36.43	77.31	79.67
Road	49.4	59.39	82.31	82.69
Trash Can	0.0	0.0	5.38	18.63
Manhole	0.0	0.0	2.25	16.96
Catch Basin	0.0	0.0	1.58	13.59
Snow	0.0	0.0	56.97	71.46
Person	0.0	0.0	39.23	48.3
Water	0.0	0.0	29.87	16.1
Mean	3.67	4.79	24.74	26.51

Table 2: IOU (%) on the Vistas dataset.

for each task. One is a “vanilla” U-net [13], the other is exactly the same U-net architecture but trained with hierarchical losses (referred to as U-net+HL). Note that we do not seek nor achieve state-of-the-art performance. A more complex architecture, problem-specific tuning and so on would lead to improved performance but our goal here is to assess relative performance gain using a simple baseline architecture. We use basic Stochastic Gradient Decent with a learning rate of 0.01 and a batch size of 5. During training images/annotations were randomly square-cropped using the shortest dimension and re-sized to 256^2. The only further augmentation used was random flipping (p = 0.5).

Datasets For experimenting with hierarchical losses on segmentation we chose two very different datasets: the Helen [6] facial dataset and Mapillary’s Vistas [11] road scene dataset. The Helen dataset [1] covers a wide variety of facial types (age, ethnicity, colour/grayscale, expression, facial pose), originally built for facial feature localisation [6]. For our experiments we used an augmented Helen [15] dataset [2] processed for class labelling, and suitable for semantic segmentation. Helen contains 2000, 230 and 100 images/annotations for training, validation and testing respectively, for 11 classes (10 facial and background, see Table 1). For our facial experimentation we chose to use the four level hierarchy shown in Figure 2

[1]http://www.ifp.illinois.edu/~vuongle2/helen/
[2]http://pages.cs.wisc.edu/~lizhang/projects/face-parsing/

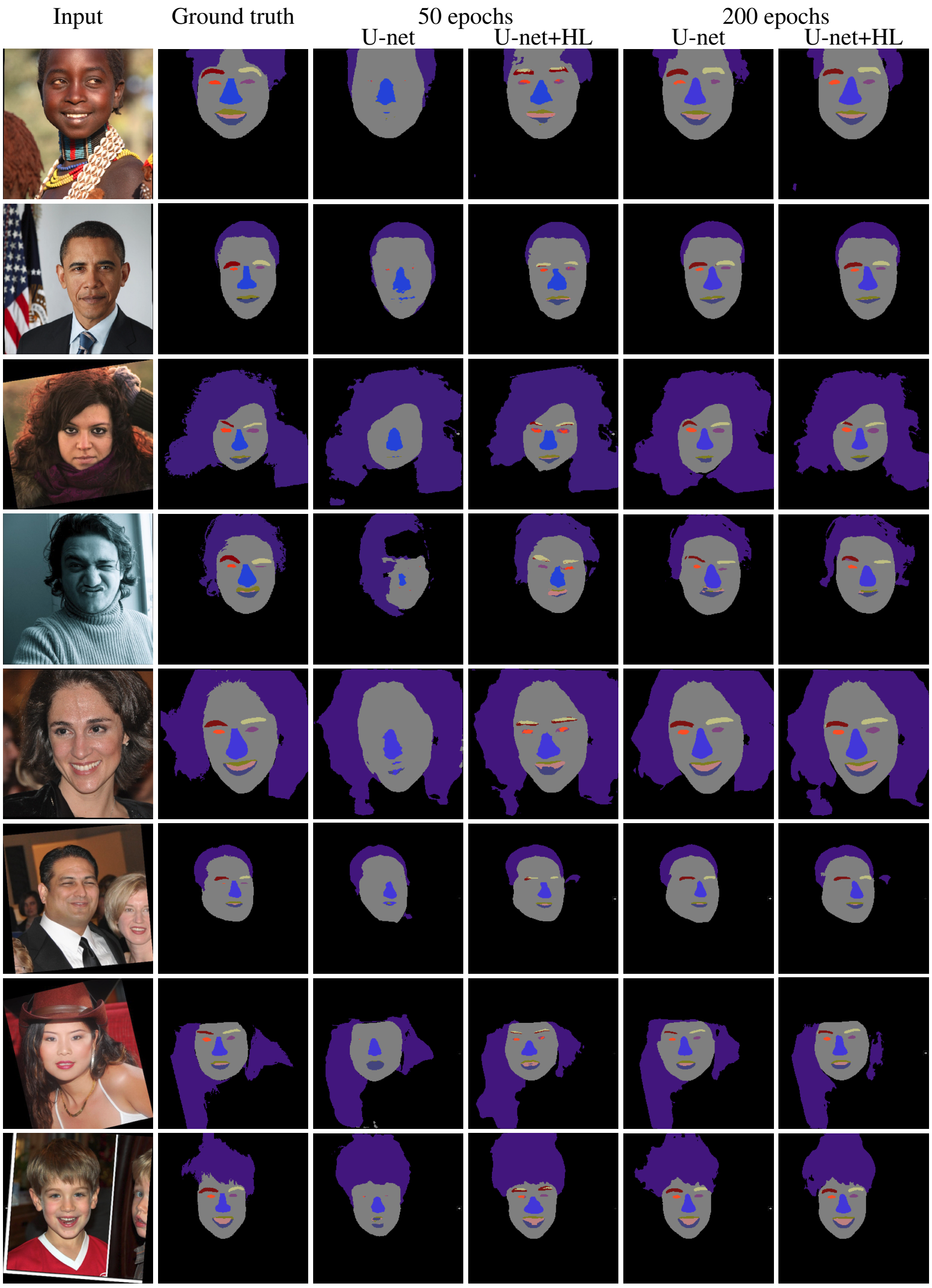

Figure 4: Prediction comparisons on Helen dataset. From left to right: raw input image, ground truth annotation, vanilla trained U-Net prediction at 50 epochs, hierarchically trained U-Net prediction at 50 epochs, vanilla trained U-Net prediction at 200 epochs, hierarchically trained U-Net prediction at 200 epochs.

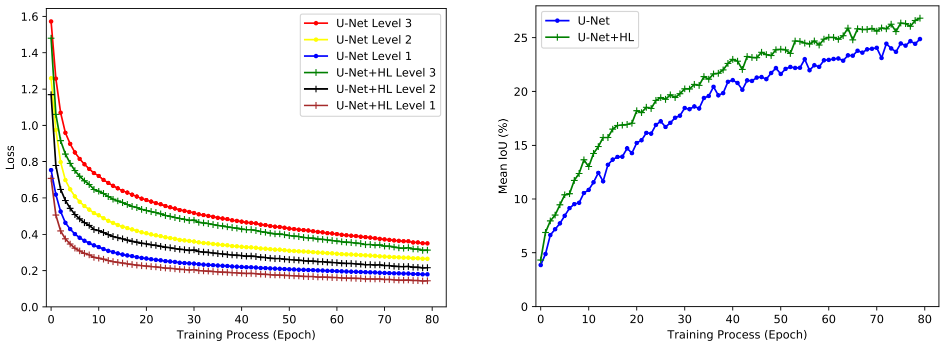

Figure 5: Vistas leaning curves: Level losses (left) and IOU (right) as models are trained with vanilla (U-Net) and hierarchical (U-Net+HL) losses.

(left) as it seemed to capture the 10 facial classes most intuitively. The hierarchy contains the 11 leaf nodes, and 6 internal nodes. It should be noted that the ground truth annotations are occasionally inaccurate, particularly for hair which makes it challenging to learn. The road scene Vistas dataset is composed of 25000 images/annotations (18000 training, 2000 validation, 5000 testing), with 66 classes (see Table 2 for a selection). Vistas is constructed with variation and balance in mind (country, resolution, capture quality and perspective). We chose to use Vistas because of its size, quality and variation, but also as it already had a clear hierarchy given by Mapillary in their dataset publication [11]. The road scene hierarchy we used is three levels deep, contains 66 leaf nodes, and 11 internal nodes.

Results Figure 3 (left) shows losses for each depth of the class hierarchy for the Helen dataset. Note that the deeper loss is always larger than a shallower one. This shows that our hierarchically train method is significantly benefiting from the hierarchical structure in the class labels, particularly in the early phase of training, learning much faster than the vanilla model. Figure 3 (right) illustrates the mean IOU during training. Performance gain is most significant around epoch 50 and can be observed in the qualitative results from Figure 4. The IOU scores in Table 1 also indicate the hierarchical method is outperforming the vanilla method quantitatively at epoch 200. We achieve an overall mean IOU of 72% with our U-Net+HL which is not far from the state-of-the-art of 79% [4] on this dataset.

Next, we show results for the Vistas road scene segmentation data. Figures 5 and 6 show the training curves and qualitative results respectively. The IOU performance gain is less significant than on Helen, but we show the hierarchically trained model outperforming the vanilla model in both level losses and mean IOU. Table 2 outlines a selection of per class and mean IOU scores on both methods for Vistas at epochs 1 and 80. The qualitative results in Figure 6 depict predictions for both methods at epoch 1 to outline where there was an obvious difference in performance. Most significantly, after 1 epoch of training the hierarchically trained model is able to classify correctly a significant proportion of white lines (lane-marking) whereas the vanilla trained model cannot at all.

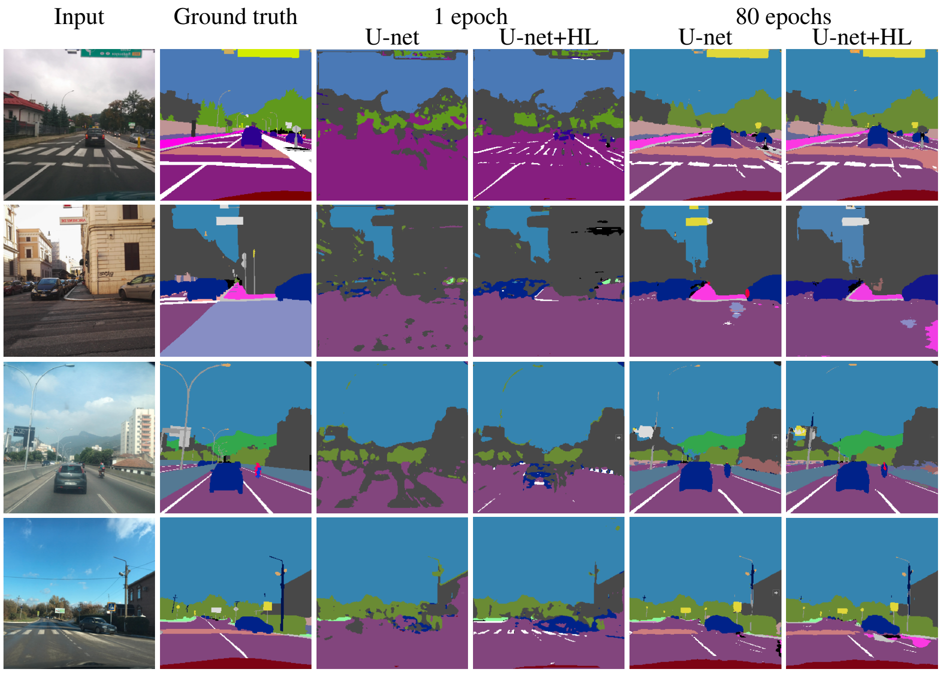

Figure 6: Prediction comparisons on Vistas. From left to right: raw input image, ground truth annotation, vanilla trained U-Net prediction at 50 epochs, hierarchically trained U-Net prediction at 50 epochs, vanilla trained U-Net prediction at 200 epochs, hierarchically trained U-Net prediction at 200 epochs.

6 Conclusions

Our results illustrate the great potential and merit of using losses that encourage semantically similar classes within a hierarchy to be classified close together, where the model parameters are guided towards a solution not only better quantitatively, but faster in training than using a standard loss implementation. There is a possible link to metric learning here where, rather than positive and negative class labels, we are provided with classes that can be more or less similar within a hierarchy. A particular advantage of this work is its generality and self-contained nature allows the possibility of plugging this hierarchical loss on the end of any deep learning architecture. Taking advantage of the hierarchical cues readily apparent to us can help train a deep neural network faster and with greater accuracy. Moreover any hierarchical structure can be provided to help train your model. We also contribute a numerically stable formulation for computing log and softmax of a network output separately, a necessity for summing probabilities according to a hierarchical structure.

We hope to experiment further with other hierarchies and segmentation datasets. Additional future work possibilities could include learning the hierarchy itself which best solves your task. In addition we would like to use our ideas to construct a level of abstraction segmenter for tree based labels on hierarchically trained models. Ability to extract segmentations at multiple levels in a hierarchy describing your data is quite useful, intuitive and is not something really done by semantic segmentation solvers in the current domain.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.

[3] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *European conference on computer vision*, pages 48–64. Springer, 2014.

[4] Umut Güçlü, Yağmur Güçlütürk, Meysam Madadi, Sergio Escalera, Xavier Baró, Jordi González, Rob van Lier, and Marcel AJ van Gerven. End-to-end semantic face segmentation with conditional random fields as convolutional, recurrent and adversarial networks. *arXiv preprint arXiv:1703.03305*, 2017.

[5] Andrew Kae, Kihyuk Sohn, Honglak Lee, and Erik Learned-Miller. Augmenting CRFs with Boltzmann machine shape priors for image labeling. 2013.

[6] Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas S Huang. Interactive facial feature localization. In *European conference on computer vision*, pages 679–692. Springer, 2012.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[8] Ping Luo, Xiaogang Wang, and Xiaoou Tang. Hierarchical face parsing via deep learning. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2480–2487. IEEE, 2012.

[9] Panagiotis Meletis and Gijs Dubbelman. Training of convolutional networks on multiple heterogeneous datasets for street scene semantic segmentation. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[10] Kevin P Murphy. Naive bayes classifiers. Technical Report 18, University of British Columbia, 2006.

[11] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4990–4999, 2017.

[12] Qingqun Ning, Jianke Zhu, and Chun Chen. Very fast semantic image segmentation using hierarchical dilation and feature refining. *Cognitive Computation*, 10(1):62–72, 2018.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[14] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5639–5647, 2018.

[15] Brandon M Smith, Li Zhang, Jonathan Brandt, Zhe Lin, and Jianchao Yang. Exemplar-based face parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3484–3491, 2013.

[16] Nitish Srivastava and Ruslan R Salakhutdinov. Discriminative transfer learning with tree-based priors. In *Advances in Neural Information Processing Systems*, pages 2094–2102, 2013.

[17] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019.

[18] Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 2740–2748, 2015.

[19] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[20] Xinqi Zhu and Michael Bain. B-cnn: Branch convolutional neural network for hierarchical classification. *arXiv preprint arXiv:1709.09890*, 2017.