Exploiting Test Time Evidence to Improve Predictions of Deep Neural Networks

Dinesh Khandelwal[†] **Suyash Agrawal**[‡] **Parag Singla**[†] **Chetan Arora**[†]
[†]{dineshk,parags,chetan}@cse.iitd.ac.in [‡]suyash1212@gmail.com
Indian Institute of Technology Delhi

Abstract

Many prediction tasks, especially in computer vision, are often inherently ambiguous. For example, the output of semantic segmentation may depend on the scale one is looking at. Similarly, image saliency or video summarization is often user or context dependent. Arguably, in such scenarios, exploiting instance specific evidence, such as scale or user context, can help resolve the underlying ambiguity leading to the improved predictions. While existing literature has considered incorporating such evidence in classical models such as probabilistic graphical models (PGMs), there is limited (or no) prior work looking at this problem in the context of deep neural network (DNN) models. In this paper, we present a generic multi-task learning (MTL) based framework which handles the evidence as the output of one or more secondary tasks, while modeling the original problem as the primary task of interest. Our training phase is identical to the one used by standard MTL architectures. During prediction, we back-propagate the loss on secondary task(s) such that network weights are re-adjusted to match the evidence. An early stopping or two norm based regularizer ensures weights do not deviate significantly from the ones learned originally. Implementation in a scenario of predicting semantic segmentation given the image level tags clearly demonstrates the effectiveness of our proposed approach.

1 Introduction

Over the last decade, Deep Neural Networks (DNNs) have become a leading technique for solving variety of problems in Artificial Intelligence. Often, such networks are designed and trained for a specific task at hand. When multiple correlated tasks are given, Multi-Task Learning (MTL) [3] framework can be used to allow a DNN to jointly learn shared features from multiple tasks simultaneously. Usually, in an MTL framework, one is interested in the output of all the tasks at hand. However, researchers have also looked at the scenarios, when only a subset of tasks (called 'principal' task(s)) are of interest and the other tasks (called 'auxiliary' task(s)) are merely to help learn the shared generic representation [9, 2, 14]. In such cases, auxiliary tasks are generally derived from the easily available side information about the data. For example, one can use an MTL framework for semantic segmentation of an image as a principal task, with an auxiliary task to predict types of object present in the image.

One significant limitation of the MTL frameworks suggested so far is that they make use of auxiliary information only during the training process. This is despite the fact that, many times such information is also available at the test time e.g., tags on a Facebook image. Arguably, exploiting this information at test time can significantly boost up prediction accuracy by resolving the underlying ambiguity and/or correcting modelling errors. We will henceforth refer to the instance specific auxiliary information as *evidence*.

Presented at the NeurIPS 2019 workshop on Learning with Rich Experience: Integration of Learning Paradigms.

A natural question to ask is, whether there is a way to incorporate similar instance specific additional clues in modern deep neural network models. Similar works in classical machine learning literature such as PGMs [7] have used conditional inference to achieve this objective. A closely related research area that incorporates additional features from auxiliary information is multi-modal learning [11]. Practically designing a network when the auxiliary information comes from a highly sparse source can be quite difficult. E.g., improving segmentation using image level tags seems non-trivial due to the varied kinds of features in the two cases. There has been some recent work on doing this [15], albeit with marginal improvements. FiLM [13] proposes a general architecture for conditioning on available information, but do not explicitly deal with auxiliary information as evidence. Another limitation of multi-modal based approaches is that they require jointly annotated data for training.

Some recent works [12, 17, 10] have proposed constraining the output of DNN at training time, to help regularize the output and reduce the amount of training data required. Lee et al. [8] have proposed to enforce test time constraints on a DNN output. However, while their idea is to enforce 'prior hard (or soft) constraints' arising out of natural rule based processing, our framework is inspired from using any easily available and arbitrary type of auxiliary information.

We model our task of interest, e.g., semantic segmentation, as the primary task in the MTL framework. The evidence is modelled as the output of one or more secondary tasks, e.g., image tags. While our training process is identical to the one used by standard MTL architectures, our testing phase is quite different. Instead of simply doing a forward propagation during prediction, we back-propagate the loss on the output of the secondary tasks (evidence) and re-adjust the weights learned to match the evidence. In order to avoid over-fitting the observed evidence, we employ a regularizer in the form of two norm penalty or early stopping, so that the weights do not deviate significantly from their originally learned values. Notably, our approach does not require jointly annotated data for training.

The specific contributions of this work are as follows: (1) We propose a novel adaptation of the MTL framework which can incorporate evidence at test time to improve predictions of a DNN. (2) We suggest two approaches to re-adjust the weights of the deep network so as to match the network output to evidence. (3) We demonstrate the efficacy of proposed framework by improving state-of-the-art results on semantic segmentation problem.

The work is under-progress. This is a preliminary version of our work for quick dissemination.

2 Proposed Framework for Back-propagating Evidence

In this section, we present our approach for boosting up the accuracy of a given task of interest by incorporating evidence. Our solution employs a generic MTL [16] based architecture which consists of a main (primary) task of interest, and another auxiliary task whose desired output (label) represents the evidence in the network. The key contribution of our framework is its ability to back-propagate the loss on the auxiliary task during prediction time, such that weights are re-adjusted to match the output of the auxiliary task with given evidence. In this process, as we will see, the shared weights (in MTL) also get re-adjusted producing a better output on the primary task. This is what we refer to as back-propagating evidence through the network (at prediction time).

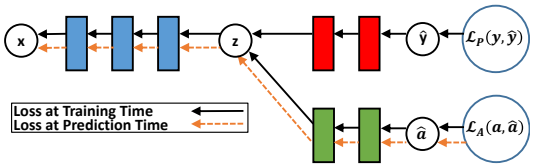

Figure 1: Proposed Architecture for incorporating evidence during both train and prediction times.

Notation We will use P to denote the primary task of interest. Similarly, let A denote the auxiliary task in the network. $x^{(i)}$ is input feature vector, $y^{(i)}$ is desired output (label) of the primary task, and $a^{(i)}$ denotes the desired output (label) of the auxiliary task for the i^{th} training example, . Correspondingly, let $\hat{y}^{(i)}$ and $\hat{a}^{(i)}$ denote the output produced by the network for the primary task and auxiliary tasks, respectively.

Model Figure 1 shows the MTL based architecture [16] for this set-up. There is a common set of layers shared between the two tasks, followed by the task specific layers. z represents the common hidden feature representation fed to the two task specific parts of the architecture. For ease of notation,

we will refer to the shared set of layers as *trunk*. The network has three sets of weights. The weights in the trunk are denoted by W_{zx}. Weights in the two task specific branches are denoted by W_{yz} and W_{az}, respectively. The total loss $\mathcal{L}_{Train}(\cdot)$ is a function of these weight parameters and can be defined as:

$$\mathcal{L}_{Train}(\cdot) = \sum_{i=1}^{m} \Big(\mathcal{L}_P(y^{(i)}, \hat{y}^{(i)}) \Big) + \lambda \sum_{i=1}^{m} \Big(\mathcal{L}_A(a^{(i)}, \hat{a}^{(i)}) \Big)$$

Here, $\mathcal{L}_P(\cdot)$ and $\mathcal{L}_A(\cdot)$ denote the loss for the primary and auxiliary tasks, respectively. λ is the importance weight for the auxiliary task. The sum is taken over the m examples in the training set. $\mathcal{L}_P$ is a function of the shared set of weights W_{zx}, and the task specific weights W_{yz}. Similarly, and $\mathcal{L}_A$ is a function of the shared weights W_{zx} and task specific weights W_{az}, respectively.

Training Training process is identical to the one used by standard MTL, where the goal of training is to find the weights which minimize the total loss over the training data. Note that the weights in the task specific branches, i.e., W_{yz} and W_{az}, can only affect losses defined over the respective tasks. On the other hand, weights W_{zx} in the trunk affect the losses defined over both the primary as well as the auxiliary tasks. The weights are learned using standard the back-propagation algorithm. Next, we describe our approach of back-propagating the loss over the evidence at test time.

Prediction time During test time, we are given additional information about the output of the auxiliary task. Let us denote this by e (evidence) to distinguish it from the auxiliary outputs during training time. Then, for the inference, instead of directly proceeding with the forward propagation, we instead first decide to adjust the weights of the network such that the network is forced to match the evidence e on the auxiliary task. Since the two tasks are correlated, we expect that this process will adjust the weights of the network in a manner such that resolving the ambiguity over the auxiliary output will also result in an improved prediction over the primary task of interest.

This feat can be achieved by defining a loss in terms of $\mathcal{L}_A(\hat{a}, e)$ and then back-propagating its gradient through the network. Note that this loss only depends on the set of weights W_{az} in the auxiliary branch, and the weights W_{zx} in the trunk. In particular, the weights W_{yx} remain untouched during this process. Finally, we would also like to make sure that our weights do not deviate too much from the originally learned weights. This is to avoid over-fitting over evidence. This can be achieved by adding a two-norm based regularizer which discourages weights which are far from the originally learned weights. The corresponding weight update equations can be derived using the following gradients:

1. $\nabla_{W_{az}} \mathcal{L}_{Test}(\cdot) = \nabla_{W_{az}} \mathcal{L}_A(\hat{a}, e) + \alpha ||W_{az} - W^*_{az}||^2$
2. $\nabla_{W_{zx}} \mathcal{L}_{Test}(\cdot) = \nabla_z \mathcal{L}_A(\hat{a}, e) \nabla_{W_{zx}}(z) + \alpha ||W_{zx} - W^*_{zx}||^2$

Here, W^*_{az} and W^*_{zx} denote the weights learned during training and α is the regularization parameter. These equations are similar to the ones which would be used during training, with the differences that (1) The loss is now computed with respect to the single test example (2) Effect of the term dependent on primary loss has been zeroed out (3) A regularizer term has been added. In our experiments, we also experimented with early stopping instead of adding the norm based regularizer. Figure 1 explains the weight update during test time pictorially. Once the new weights are obtained, they can be used in the forward propagation to obtain the desired value $\hat{y}$ on the primary task.

3 Experiments on Image Segmentation

The task of semantic segmentation involves assigning a label to each pixel in the image from a fixed set of object categories. In many semantic segmentation applications, image level tags are often easily available. We explore the use of such tags as auxiliary information at test time for improving the prediction accuracy.

Our Implementation Our implementation builds on DeepLabv3+ [4] which in one of the state of the art segmentation models. For ease of notation, we will refer the DeepLabv3+ model as 'DeepLab'. To use our framework, we have extended the DeepLab architecture[1] to simultaneously solve the

[1] `https://github.com/tensorflow/models/tree/master/research/deeplab`.

image tag prediction task in an MTL setting. Starting with the original DeepLab architecture (top part in the figure), we branch off after the encoder module to solve the classification task. The resultant feature map is passed through an average pooling layer, a fully connected layer, and then finally a sigmoid activation function to get the probability for each of the image tags (except background).

For training, we make use of cross-entropy based loss for the primary, and binary cross entropy loss for the secondary task. We use pre-trained Deeplab model [1] to initialize the weights in the MTL based architecture (for the segmentation branch). The weights in the classification branch are randomly initialized. This is followed by a joint training of the MTL architecture. During prediction time, for each image, we back-propagate the binary cross entropy loss based on observed evidence over the auxiliary task (for test image) resulting in weights re-adjusted to fit the evidence. These weights are used to make the final prediction (per-image).

Method	**mIoU**	
	Without MS	With MS
DL	82.45	83.58
DL-MT	82.59	83.54
DL-MT-Pr	84.23	84.84
DL-BP-L2	85.97	86.88
DL-BP-ES	86.01	86.84
DL-BP-L2-Pr	86.10	87.00
DL-BP-ES-Pr	**86.22**	**87.03**

Table 1: Comparison of results for semantic segmentation on PASCAL VOC.

Methodology and Dataset We compare the performance of following seven models in our experiments: (a) DL (b) DL-MT (c) DL-MT-Pr (d) DL-BP-ES (f) DL-BP-ES-Pr (g) DL-BP-L2 (h) DL-BP-L2-Pr. The first model (DL) uses vanilla DeepLab based architecture. The second (DL-MT) is trained using an MTL framework as described above. These are our baseline models without use of any auxiliary information. The suffix 'Pr' in the model name refers to post-processing step of pruning the output classes at each pixel using the given image tags. The probability of each label which is not present in the tag set is forced to be zero at each pixel. This is a very simple model using the auxiliary information. Models with prefix DL-BP refers our proposed approach where we back-propagate the auxiliary information at prediction time. These can be combined with the Pr based model. We experiment with two variations based on the choice of regularizer during prediction: DL-BP-ES uses early stopping and DL-BP-L2 uses an L2-norm based penalty.

Similar to original DeepLab paper [4] we also experiment with inputting the images at multiple scales. We refer to this as multi-scaling (MS) based approach. For our evaluation, we make use of augmented PASCAL VOC 2012 segmentation benchmark [6, 5]. It consists of 20 foreground object classes and one background class. The dataset contains 10582 training and 1449 validation images. We use mean intersection over union (mIOU) as our evaluation metric which is a standard for segmentation tasks.

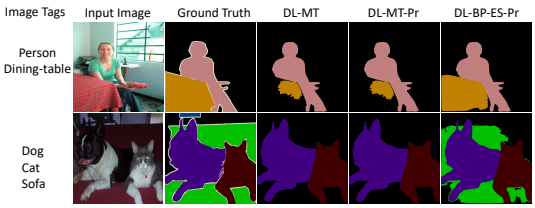

Figure 2: Comparison of visual results on semantic segmentation problem using image level tags as the test time auxiliary information.

Results Table 1 presents the results comparing the performance of various models. We see some improvement in prediction accuracy due to use of pruning the output over the baseline models. However, using our backpropagation based approach results in further significant improvement over the baselines. Additional pruning on top of our models results in marginal improvements. The gain is as much as 3.45 mIoU points compared to vanilla DeepLab and more than 2.19 points compared to the MTL based architecture with pruning (with MS). Figure 2 shows the visual comparison of results for a set of hand picked examples. The first row shows improvement in segmentation of 'dining table', second row shows the detection of a new object 'sofa' by our technique using test time tags.

4 Conclusion

We have presented a novel approach to incorporate evidence into deep networks at prediction time. Our key idea is to model the evidence as auxiliary information in an MTL architecture and then modify the weights at prediction time such that output of auxiliary task(s) matches the evidence. The approach is generic and can be used to improve the accuracy of existing diverse neural network architectures for variety of tasks, using easily available auxiliary information at test time.

Acknowledgements

Parag Singla is supported by the DARPA Explainable Artificial Intelligence (XAI) Program # N66001-17-2-4032, IBM SUR award, and Visvesvaraya Young Faculty Fellowships by Govt. of India. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of the funding agencies.

References

[1] Pre-trained Weights for DeepLabv3+. `https://github.com/tensorflow/models/blob/master/research/deeplab/g3doc/model_zoo.md`.

[2] J. Bingel and A. Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proc. of EACL*, pages 164–169, 2017.

[3] R. Caruana. Learning many related tasks at the same time with backpropagation. In *Proc. of NIPS*, pages 657–664, 1995.

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. of ECCV*, pages 801–818, 2018.

[5] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[6] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *Proc. of ICCV*, pages 991–998, 2011.

[7] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[8] J. Y. Lee, S. V. Mehta, M. Wick, J.-B. Tristan, and J. Carbonell. Gradient-based inference for networks with output constraints. In *Proc. of AAAI*, pages 4147–4154, 2019.

[9] C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *Proc. of ICLR*, 2015.

[10] P. Marquez Neila, M. Salzmann, and P. Fua. Imposing hard constraints on deep networks: Promises and limitations. In *CVPR Workshop on Negative Results in Computer Vision*, 2017.

[11] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proc. of ICML*, pages 689–696, 2011.

[12] D. Pathak, P. Krahenbuhl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proc. of CVPR*, pages 1796–1804, 2015.

[13] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proc. of AAAI*, 2018.

[14] B. Romera-Paredes, A. Argyriou, N. Berthouze, and M. Pontil. Exploiting unrelated tasks in multi-task learning. In *Proc. of AISTATS*, pages 951–959, 2012.

[15] A. Rosenfeld, M. Biparva, and J. K. Tsotsos. Priming neural networks. In *Proc. of CVPR Workshops*, pages 2011–2020, 2018.

[16] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[17] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proc. of ICML*, pages 5498–5507, 2018.