Multitask Learning in Image Recognition, Segmentation, and Classification

1 Introduction

Background. Multi-task Learning (MTL) is a kind of machine learning strategy that exploits commonalities across different tasks and hence improve network performance. The way human learn from experience inspires the ideology of MTL and another sub-field called transfer learning where learned model parameters are transferred to another learning task. The most primitive mechanism of MTL is first proposed by Caruana in 1998 where she suggested that it is suitable to share hidden representations of data in neural networks.

Motivation. Compared to conventional single-task learning (STL), MTL saves more computational power. An MTL framework with shared parameters can bring higher efficiency than multiple STL models together. Because of the parameter sharing feature of MTL, information learnt from related tasks can improve a network's learning ability thus reducing overfitting and improving generalisation performance, as well as fast learning speed and robust to noisy data. Often in industry, implementing an MTL framework allow us to combine insufficiently sized datasets with related learning objectives by exploiting shared information.

Related work. Generally, MTL has two commonly used approaches: Simultaneous Learning and Sequential Learning.

1. In sequential learning, auxiliary tasks are learnt as a priori, then transfer the already trained parameters to the target task network. This storing and sharing procedure is first introduced by Stevo Bozinovski in 1976 and has been perfected by researchers ever since. Now it is one of the most well-practised algorithm applied to various fields like text classification [Andrew Ng in 2004], spam classification [Bickel Steffen] and computer vision etc.
2. Simultaneous learning involves a single network that consists of some task-specific and shared layers, and have multiple output branches each correspond to a task. These tasks are trained simultaneously by combining the losses. Sharing operations can be classified into 2 major categories called hard and soft parameter sharing. Some examples of sharing procedures are: Deep relationship network [4], cross-stitch [9], and F-Adaptive Feature Sharing [5] (for automatic grouping tasks) and Joint Many-Task Model [6] for NLP tasks.

In this paper, MTL models are formulated in the simultaneous way. Since we believe by sharing feature maps mid-training, all three tasks can benefit from information extracted by each task specific networks.

Open-ended Question. In MRP, two auxiliary task networks and target task network are trained simultaneously with cross-stitch algorithm. Considering that there are several well-trained models in the industry, we wonder if the well-trained auxiliary task models will have a positive assists to target task model. In OEQ, we pre-train two auxiliary networks, and then transfer the feature maps of them to the target network in order to assist the training of the target network. In addition to possibly improving the performance of the target task network, our idea of the OEQ may could have strong generalization.

2 Methods and Implementation details

2.1 MRP Methodology:

The main MLT network is built upon an encoder-decoder architecture(fig 1) i.e. a Seg-Net to learn the main segmentation task and a combination of encoder and fully-connected layer for the 2 auxiliary tasks. We use residual nets as building blocks for the en/decoders to retain information and prevent gradient issues.

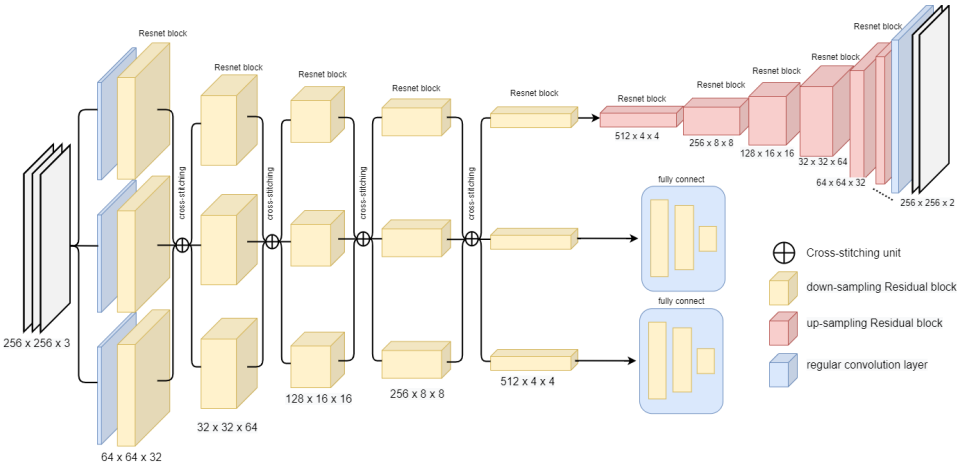

Fig. 1. Network structure of Cross-stitch MTL

Parameter Sharing with Cross-Stitch Units: In a parallel MTL setting, branches of networks share parameters. Instead of a conventional parameter combination like concatenation or point-wise addition/multiplication, we employ an operation called cross-stitch, which we perform after training each layer of residual blocks at encoding stages, with the nature of linearly combining feature

maps produced by each branch, representing by matrix operation: $X' = AX$.

$$\begin{bmatrix} x_1^{'} \\ x_2^{'} \\ x_3^{'} \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{1}$$

where $\boldsymbol{X}$ is the output feature map of previous layer, $\boldsymbol{X'}$ refers to the post cross-stitched feature maps going into the next layer, and the matrix A is the cross-stitch unit. Rather than having the weights α_{ij} fixed, we set them as learnable parameters. Therefore the network automatically determines how much information is shared between each branch.

Network Loss and Dynamic Weight Averaging: In parallel MTL, the loss is calculated as a linear combination of task-specific losses. For our implementation, a pixel-wise cross-entropy loss is employed to measure inaccuracy of image segmentation, normal cross-entropy for misclassification, and a metric called 'Distance-IoU'[8] to measure bounding box inaccuracy, which is defined as:

$$DIoU = 1 - \frac{intersection}{union} + \frac{\|c - c_{pred}\|}{d^2} \tag{2}$$

where c is the centre of the box and d is the diagonal of the smallest enclosing box covering the union.

The three losses are on the same scale which makes balancing them much easier. However, tasks have different learning difficulty. Therefore by the time the segmentation task achieve acceptable accuracy, the other tasks will certainly be overfitted. To tackle this we employ an automatic weight balancing procedure called Dynamic Weight Average[7], which would propose a weighting scheme by considering the rate of change of loss for each task. Here, weight w_k for the loss of task k is defined as:

$$w_k^{(t)} := \frac{Kexp(r_k^{(t-1)}/T)}{\sum_i exp(r_i^{(t-1)}/T)} \quad where \;\; r_k^{(t-1)} = \frac{\mathcal{L}_k^{(t-1)}}{\mathcal{L}_k^{(t-2)}} \tag{3}$$

where K is the number of task, T represent a quantity called temperature which controls the softness of the weight distribution, and $\mathcal{L}_k^{(t)}$ is the loss of task k at epoch t averaged over all iterations in this epoch. The total loss are then calculated as $\boldsymbol{w}^{(t)} \cdot \boldsymbol{\ell}^{(t)}$.

OEQ Methodology: We expand the front layers of decoder to blend the information from pre-trained auxiliary task models, and finally fuse as one single feature map for final segmentation. The method of sharing parameters between auxiliary task model to target model is sequential learning, which will transfer a $512 * 4 * 4$ feature map to the FunnelNet's decoder. In the decoder, we will use the strategy of cross stitch to extract more information from three encoders. Comparing to single self-trained encoder, it is also a kind of implicit data augmentation for the decoder, which is that a single picture creates different feature maps but a same mask label. In this way, we wish the decoder could have a strong ability of generalization.

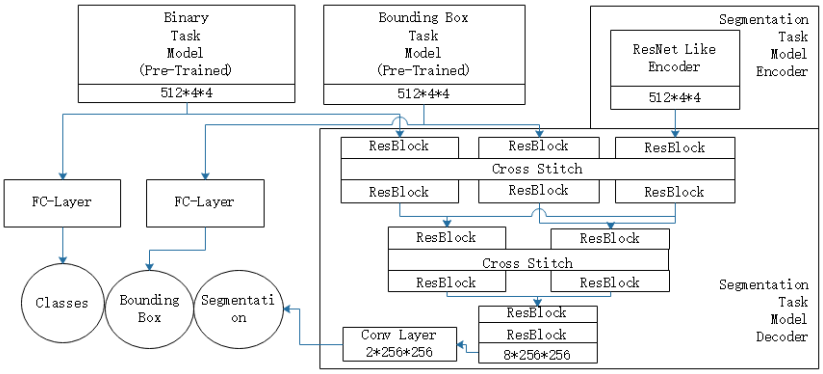

Fig. 2. OEQ Network Structure

3 Experiments and comparisons among baseline, ablation study, MTL and OEQ network

Network is experimented on the Oxford-IIIT Pet Dataset, it contains 3684 images of 37 different breeds of cats or dogs with 3 labels corresponding to different tasks: Binary classification, Object Detection, and Segmentation.

To avoid loading large amount of data into disk space in one instance, we utilize a chunk-wise data reading procedure. The dataset is recreated using HDF5's chunked storage layout before loading into data loaders. After that, the data loader could repeatedly read random data chunks into memory at a relatively fast speed. In our experiment, time consumed for iterating through the whole dataset in a batch size of 39 is around 4 seconds.

Utilising procedures stated in methodology, we are able to implement a MTL network that simultaneously learn all three tasks. To measure the network performance we employ three metrics for each learning tasks: Accuracy, Intersection over Union (IoU), mean Intersection over Union (mIoU) for classification, detection and segmentation respectively.

By observing during the experiment, task training loss starts to converge at around epoch 80. Therefore, 150 is chosen as the number of training epoch and the loss is very stable at 150_{th} epoch. During experiment, we compare ablated versions with our original network and measure their performances using metrics described above. Our purpose is to gain more intuitions on the impact of different auxiliary tasks on performance of the target task. The network structure used in the ablation study is similar to the one used in MTL, simply remove 1 branch of network and instead use a 2 by 2 cross stitch unit. Furthermore, a baseline network designed solely for segmentation is also implemented. The training of OEQ model departed in 2 stage, we use 120 epoches for training two auxiliary models and 150 epoches for training segmentation model.

4 Results and analysis

Table 1. Model performances after 150 epochs

Model	Number of parameters	Segmentation mIoU (Higher Better) Train	Val.	Test	Object Detection IoU (Higher Better) Train	Val.	Test	Binary Classification Accuracy (Higher Better) Train	Val.	Test
Baseline	18,882,133	0.970	0.804	0.813	- - -			- - -		
Binary Ablation	24,029,967	**0.971**	**0.811**	**0.816**	- - -			**0.989**	0.864	0.891
B.Box Ablation	24,030,129	0.961	0.804	0.809	0.804	0.651	0.662	- - -		
MTL	29,178,891	0.966	0.799	0.812	**0.808**	**0.682**	**0.759**	0.972	**0.915**	**0.916**
OEQ	46,271,473	0.971	0.788	0.791	0.750	0.688	0.693	0.957	0.883	0.874

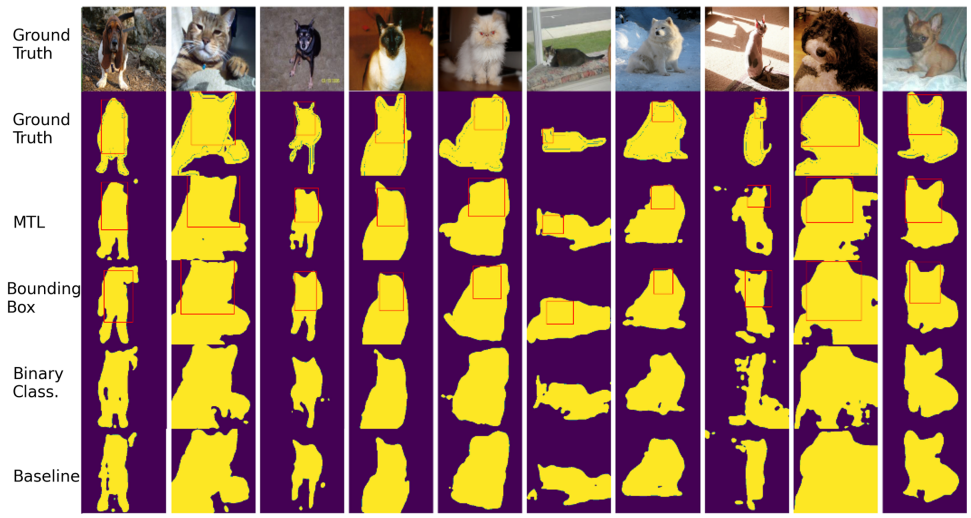

Fig. 3. Network structure of Cross-stitch MTL

Comparing the test results in Table 1, there is no significant difference between the segmentation mIoUs of different models. The ablation model with binary auxiliary task showed slightly better performances than the others. On the other hand, bounding box Ablation model is the worst of the bunch by a tiny margin. However, the MTL model performs relatively better on auxiliary tasks than both of the ablation models.

Observing the predicted segmentation masks shown in figure 3, all variant of our model produce very similar results. However we would argue that the silhouette determined by the MTL model resembles more of a pet, where as the patterns generated by other models are a bit random. MTL's predicted

bounding box is closest to the ground truth. Furthermore We can gain a glimpse of how the network determines the segmentation mask for an image, it finds sudden colouration changes in the image. As we can see from the sixth image, the segmentation enclosed part of the floor as well as the pet, and in the third last image, shadow casted is considered as silhouette of the cat too.

As for the result of the OEQ model, under the circumstance that the model has more parameters, performance did not show a high level as we imagine before, on the contrary, it was finally overfitting. The reason may be that we actually do not to need too much parameters to solve the segmentation in this experiment.

5 Discussion

Experimentation results showed that both auxiliary task has no significant positive effect on the target task. One possible explanation is that facial bounding box does not help segmenting whole body silhouettes. Classifying pets provides limited assistance to our target task as well. In some cases, pets take up most of the space in the images and we can barely tell the difference between the silhouettes of cats and dogs. In the future research, we could use more related auxiliary tasks like determining the postures of the pets or their sizes in the image. Also, generating the segmentation is determined by finding the sudden changes of colouration, thus if the auxiliary task learns the specific breeds of pets, the network might store the colour information of that specific breeds and help MTL to segment out pets from images. Even the performance of OEQ model in this experience seem not good, but we do not know its performance on complex task, which is worth exploring in-depth.

Our networks rely on the cross-stitching unit for exchanging information between network branches, by nature there is a randomness to the assigned values in cross-stitch matrix, which cannot make sure that majority of the information fed to subsequent encoder comes from earlier encoder belongs to the same task branch. We suspect by doing so we have disrupted the information flow in the encoder layers. Therefore, in future research, more procedure should be investigated that guarantees both effective sharing and continuity of information flow.

6 Conclusion

In this paper, several different multi-task learning models were implemented to explore how different auxiliary tasks and model structures can help with the image segmentation task. Based on the traditional MTL structure, we applied techniques including cross-stitching. We also implemented an innovative model structure and obtained a well trained result. The experiment showed that the bounding box task had a non-positive impact on the segmentation task in MTL models.

References

1. Reginthala, Mahesh Iwahori, Yuji Bhuyan, Manas Yoshitsugu, Hayashi Achariyaviriya, Witsarut Kijsirikul, Boonserm. (2020). Interdependent Multi-task Learning for Simultaneous Segmentation and Detection. 167-174. 10.5220/0008949501670174.
2. Vafaeikia, P., Namdar, K., Khalvati, F. (2020). A Brief Review of Deep Multi-task Learning and Auxiliary Task Learning. arXiv preprint arXiv:2007.01126.
3. Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, Silvio Savarese: "Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression", 2019; [http://arxiv.org/abs/1902.09630 arXiv:1902.09630].
4. M. Long and J. Wang, Learning Multiple Tasks with Deep Relationship Networks, abs/1506.02117, 2015 [https://dblp.org/rec/journals/corr/Long015a.bib]
5. Yongxi Lu and Abhishek Kumar and Shuangfei Zhai and Yu Cheng and Tara Javidi, "Fully-adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification", 2016, [http://arxiv.org/abs/1611.05377]
6. Kazuma Hashimoto and Caiming Xiong and Yoshimasa Tsuruoka and Richard Socher, "A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks", 2016, [http://arxiv.org/abs/1611.01587]
7. Liu, Shikun and Johns, Edward and Davison, Andrew J, "End-to-End Multi-task Learning with Attention", 2019, [https://arxiv.org/abs/1803.10704]
8. Zhaohui Zheng and Ping Wang and Wei Liu and Jinze Li and Rongguang Ye and Dongwei Ren, "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression", 2019, [http://arxiv.org/abs/1911.08287]
9. Ishan Misra and Abhinav Shrivastava and Abhinav Gupta and Martial Hebert, "Cross-stitch Networks for Multi-task Learning", 2016, [https://arxiv.org/abs/1604.03539]