MTL

Jianqi Huang

2023-08-19

MTL in Deep Learning

- Different from single task learning
- Training multiple tasks simultaneously to exploit task relationships. (Rusu et al. 2016)

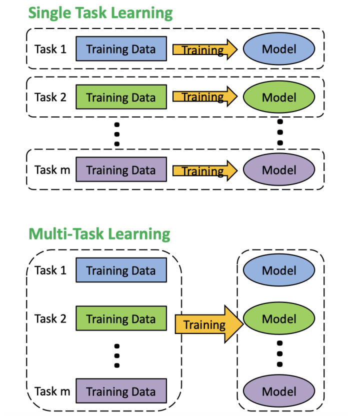


Figure 1: figure form vandenhende(2020)

Exploit Task Relationships

- **The key challenge** in multi-task learning: Exploiting (statistical) relationships between the tasks so as to improve individual and/or overall predictive accuracy (in comparison to training individual models)!

End-to-End Multi-Task Learning with Attention (S. Liu, Johns, and Davison 2019)

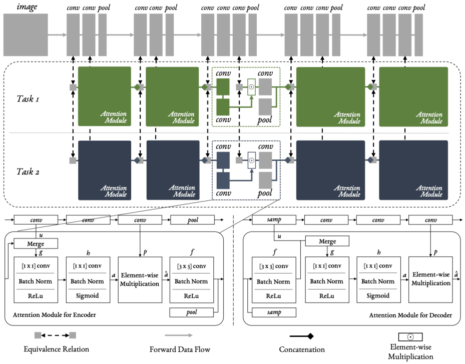
- The attention module is designed to allow the task-specific network to learn task-related features, by applying a soft attention mask to the features in the shared network.

The Architecture Design

- MTAN consists of two components: a single shared net- work, and K task-specific attention networks.

The Architecture Design(Cont.)

- each task-specific network consists of a set of attention modules
- The attention applies a soft attention mask to a particular layer of the shared network, to learn task-specific features.
- The dependence of attention on features learned jointly to maximize the generalization of the shared features across multiple tasks.



Task Specific Attention Module

The task-specific features $\hat{a}_i^{(j)}$ computed by element-wise multiplication of the attention masks with the shared features:

$$\hat{a}_i^{(j)} = a_i^{(j)} \otimes p^{(j)}$$

The Model Objective

$$\mathcal{L}_{tot}(X, Y_{1:K}) = \sum_{i=1}^{K} \lambda_i \mathcal{L}_i(X, Y_i)$$

Y_i as one task with total three tasks for evaluation.

For semantic segmentation:

$$\mathcal{L}_1(X, Y_1) = -\frac{1}{pq} \sum_{p,q} Y_1(p, q) \log \hat{Y}_1(p, q)$$

For depth estimation:

$$\mathcal{L}_2(X, Y_1) = \frac{1}{pq} \sum_{p,q} |Y_2(p, q) - \log \hat{Y}_2(p, q)|$$

For surface normals

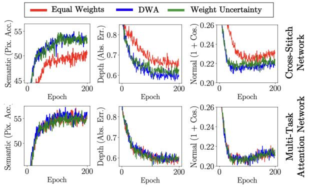
$$\mathcal{L}_3(X, Y_3) = -\frac{1}{pq} \sum_{p,q} Y_3(p, q) \log \hat{Y}_3(p, q)$$

Evaluation

Dynamic Weight Average (DWA)

$$\lambda_k(t) := \frac{K \exp(w_k(t-1)/T)}{\sum_i \exp(w_i(t-1)/T)}, w_k(t-1) = \frac{\mathcal{L}_k(t-1)}{\mathcal{L}_k(t-2)}$$

Here $w_k(\cdot,)$ calculates the relative descending rate. T represents a temperature which controls the softness of task weighting. T enough large, then $\lambda_i \approx 1$. softmax op multiplied by K, ensures that $\sum_i \lambda_i = K$



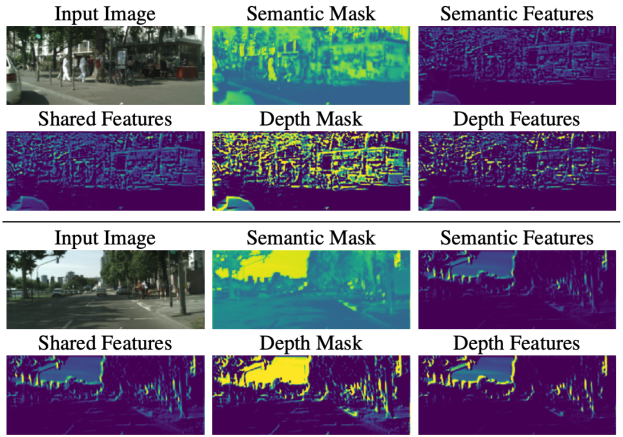


Figure 5: Visualisation of the first layer of 7-class semantic and depth attention features of our proposed network. The colours for each image are rescaled to fit the data.

Attentive single-tasking of multiple tasks (Maninis, Radosavovic, and Kokkinos 2019)

The single-tasking multiple tasks modifies its behavior through task-dependent feature adaptation, or task attention. It gives the network the ability to accentuate the features that are adapted to tasks.

Task-specific feature modulation

two tasks A and B that share a common feature tensor $F(x, y, c)$. $c = 1, 2, \cdots, C$ are the tensor channels.Assume subset $\mathcal{S}_A$ is better suited for task A.

$$F_A(x, y, c) = m_A \cdot F(x, y, c) \tag{1}$$

where $m_A[c] \in \{0, 1\}$

Residual Adapters

shunning features that do not contribute to the task.

$$L_A(x) = x + L(x) + RA_A(x), \tag{2}$$

where $L(x)$ denotes the default behavior of a residual layer, RA_A is the task-specific residual adapter of task A. and $L_A(x)$ is the modified layer.

Adversarial Task Disentanglement

- a shared representation has better memory/computation complexity, every task can profit by having its own ‘space’.
- gradients used to train the shared parameters are statistically indistinguishable across tasks.

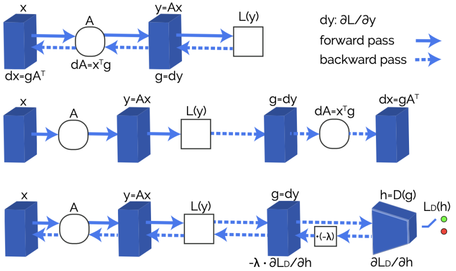


Figure 4. Double backprop [10] exposes the gradients computed during backprop (row 1) by unfolding the computation graph of gradient computation (row 2). Exposing the gradients allows us to train them in an adversarial setting by using a discriminator, forcing them to be statistically indistinguishable across tasks (row 3). The shared network features x then receive gradients that have the same distribution irrespective of the task, ensuring that no task abuses the shared network, e.g. due to higher loss magnitude. The

Adversarial Task Disentanglement(Cont.)

The optimization problem:

$$\min_{wD} \max_{wN} L(D(g_t(w_N), w_D), t),$$

where $g_t(w_N)$ is the gradient of task t computed with w_N, $D(\cdot, w_D)$ is the discriminator's output for input. And $L(\cdot, t)$ the cross-entropy loss for label t that indicates the source task of the gradient.

Summary

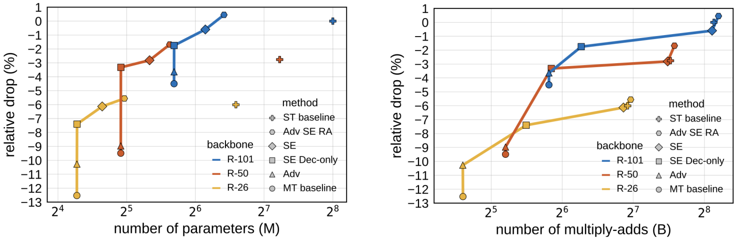
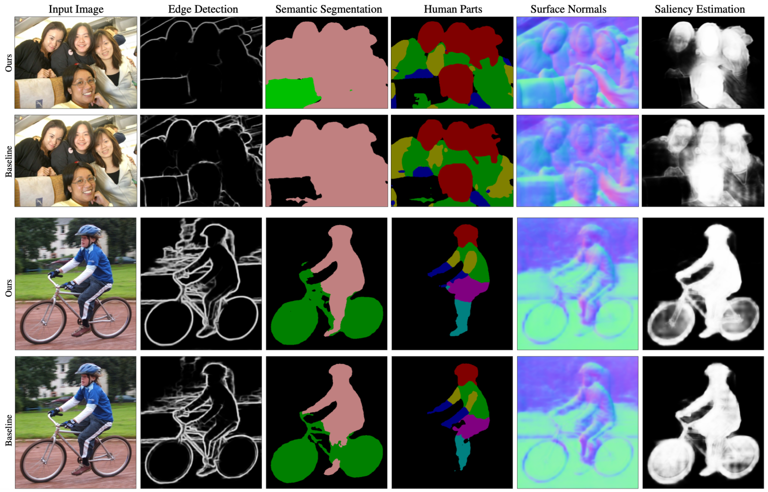


Figure 5. **Performance vs. Resources:** Average relative drop ($\Delta_m\%$) as a function of the number of parameters (left), and multiply-adds (right), for various points of operation of our method. We compare 3 different backbone architectures, indicated with different colors. We compare against single-tasking baseline (ST baseline), and multi-tasking baseline (MT baseline). Performance is measured relative to the best single-tasking model (R-101 backbone). An increase in performance comes for free with adversarial training (Adv). Modulation per task (SE) results in large improvements in performance, thanks to the disentangled graph representations, albeit with an increase in computational cost if used throughout the network, instead of only on the decoder (SE Dec-only vs. SE). We observe a drastic drop in number of parameters needed for our model in order to reach the performance of the baseline (SE, Adv). By using both modulation and adversarial training (Adv SE RA), we are able to reach single-task performance, with far fewer parameters.



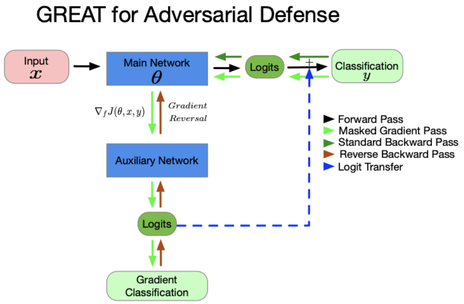
Input Image
Edge Detection
Semantic Segmentation
Human Parts
Surface Normals
Saliency Estimation
Ours
Baseline
Ours
Baseline

Gradient Adversarial Training of Neural Networks (Sinha et al. 2018)

Adversarial defense

The adversarial gradient signal ϱ flowing forward in the main network can be shown to be,

$$\varrho^{l+1} = -w^{l+1}\varrho^{l} \odot \sigma'(z^{l}) \tag{3}$$



GREACE: adapts the cross-entropy loss function

- add weight to the negative classes whose gradient tensors are similar to those of the primary class.
- The weight is evaluated using soft-max distribution from the auxiliary network. In math,

$$\nabla_a \hat{C} \to \nabla_a C + \beta \cdot \sigma(\hat{a}) \mathbf{1}_{\hat{y} \neq y} \tag{4}$$

  where $a, \hat{a}$ are the output activations from the main and auxiliary network respectively. σ is the soft-max function, β is a penalty parameter.

The combined objective for adversarial defence is:

$$\min_{\theta} \hat{J}(\theta, x, y) + \alpha \max_{\hat{\theta}} J(\hat{\theta}, \nabla \bar{J}(\theta, x, y), y).$$

The $\hat{J}$ denotes the GREACE, J denote the standard cross-entropy and $\bar{J}$ indicates the masked cross-entropy, and α is a weight parameter for the auxiliary network's loss.

Different scenarios

Knowledge distillation

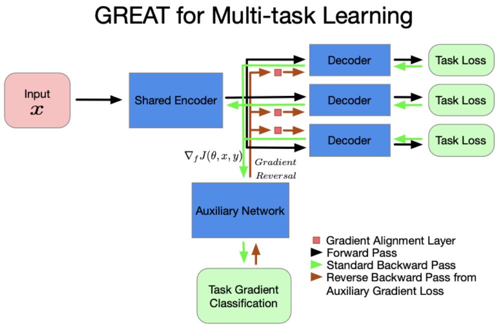
- student's output distribution $S(x)$
- teacher's output distribution $T(x)$
- A solution for student, $S(x)$ which jointly minimizes the supervised loss and $\nabla S(x) = \nabla T(x)$ exists.
- GREAT procedure: mimics GAN training procedures
- Objective to be optimized is:

$$(1-\alpha)\min_{\theta} J(\theta, x, y) + \alpha \min_{\theta} \max_{\omega} D(\theta, x, y) \tag{5}$$

$$D(\theta, x, y) = E_{t \sim \nabla T(x)} \log f(t, \omega) + E_{s \sim \nabla J(\theta, x, y)} \log(1 - f(s, \omega)) \tag{6}$$

Multi-task learning

Gradient-alignment layers(GAL) are placed after the shared encoder and before each of the task-specific decoders. And they are only during the backward pass, i.e. the GALs are dropped during forward inference.

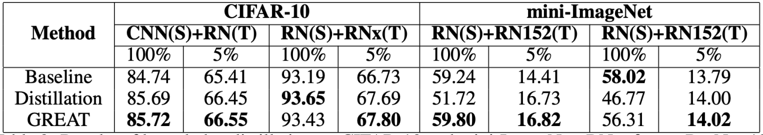

GREAT for Multi-task Learning
Input x
Shared Encoder
Decoder
Decoder
Decoder
Task Loss
Task Loss
Task Loss
$\nabla_f J(\theta, x, y)$
Gradient Reversal
Auxiliary Network
Task Gradient Classification
■ Gradient Alignment Layer
▶ Forward Pass
▶ Standard Backward Pass
▶ Reverse Backward Pass from Auxiliary Gradient Loss

Multi-task learning(Cont.)

Algorithm 3 Algorithm for multi-task learning using GREAT on GALs

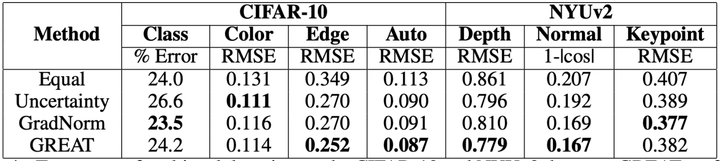
```
1:  procedure TRAIN($\theta, \acute{\theta}, \omega_i, \gamma_i$)          ▷ Requires inputs $x$, labels for tasks $y_i$
2:      $\gamma_i \leftarrow \mathbb{1}, \quad C_i^0 \leftarrow J_i(\theta, \omega_i, x, y_i)$          ▷ Initialize GAL tensors with ones and initial task losses
3:      while $j < j_{max}$ do          ▷ $j$ is current iteration
4:          $C_i \leftarrow J_i(\theta, \omega_i, x, y_i)/C_i^0 \quad \forall i$          ▷ Normalize task losses after forward pass
5:          $g_i^f \leftarrow \nabla J_i(\omega_i, x, y_i) \quad \forall i$          ▷ Evaluate task gradient tensors w.r.t. feature $f$
6:          $\omega_i(j) \rightarrow \omega_i(j+1) \quad \forall i$          ▷ Update weights in decoders using $\nabla C_i$
7:          $\theta(j) \rightarrow \theta(j+1)$          ▷ Update weights in encoder using $\sum_i g_i^f \gamma_i$
8:          $\acute{C} \leftarrow \acute{J}(\acute{\theta}, g_i^f \gamma_i, \acute{y})$          ▷ Task classification loss by forward pass
9:          $\acute{\theta}(j) \rightarrow \acute{\theta}(j+1)$          ▷ Update weights in task classifier network using $\acute{C}$
10:         $\nabla_{\gamma_i} \acute{C} \leftarrow -\nabla \acute{J}(\acute{\theta}, g_i^f \gamma_i, \acute{y})$          ▷ Evaluate reversed gradient w.r.t $\gamma_i$
11:         $\gamma_i(j) \rightarrow \gamma_i(j+1) \quad \forall i$          ▷ Update weights in GALs using $\nabla_{\gamma_i} \acute{C}$
```

Results



Method	CIFAR-10				mini-ImageNet			
	CNN(S)+RN(T)		RN(S)+RNx(T)		RN(S)+RN152(T)		RN(S)+RN152(T)	
	100%	5%	100%	5%	100%	5%	100%	5%
Baseline	84.74	65.41	93.19	66.73	59.24	14.41	**58.02**	13.79
Distillation	85.69	66.45	**93.65**	67.69	51.72	16.73	46.77	14.00
GREAT	**85.72**	**66.55**	93.43	**67.80**	**59.80**	**16.82**	56.31	**14.02**

Table 3: Results of knowledge distillation on CIFAR-10 and mini-ImageNet. RN refers to ResNet-18. The third row indicates the % of all train samples used during training. GREAT performs best in the sparse regime for all combinations and better than distillation on all but 1 scenario.



Method	CIFAR-10				NYUv2		
	Class	Color	Edge	Auto	Depth	Normal	Keypoint
	% Error	RMSE	RMSE	RMSE	RMSE	1-|cos|	RMSE
Equal	24.0	0.131	0.349	0.113	0.861	0.207	0.407
Uncertainty	26.6	**0.111**	0.270	0.090	0.796	0.192	0.389
GradNorm	**23.5**	0.116	0.270	0.091	0.810	0.169	**0.377**
GREAT	24.2	0.114	**0.252**	**0.087**	**0.779**	**0.167**	0.382

Table 4: Test errors of multi-task learning on the CIFAR-10 and NYUv2 datasets. GREAT performs best on 2 tasks each for CIFAR and NYUv2, and has comparable performance on the other tasks.

Summary

- strong defense to both targeted and non-targeted adversarial examples
- can easily distill knowledge from different teacher networks without heavy parameter tuning
- aid multi-task learning by tuning a gradient alignment layer.
- which is similar to the architecture in (Ganin and Lempitsky 2015)

Adversarial Multi-task Learning for Text Classification (P. Liu, Qiu, and Huang 2017)

- **Hazard**: the shared and private latent feature spaces from interfering with each other. And it will cause the capacity of shared space wasted by some unnecessary features.
- Proposed adversarial training to ensure that the shared feature space simply contains common and task-invariant information.

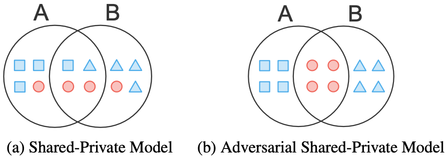
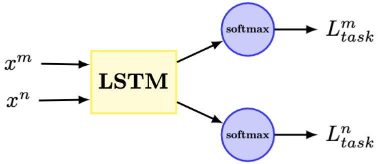


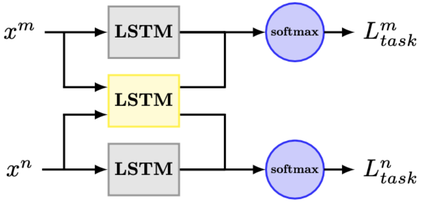
Figure 1: Two sharing schemes for task A and task B. The overlap between two black circles denotes shared space. The blue triangles and boxes represent the task-specific features while the red circles denote the features which can be shared.

Multi-task Learning for Text Classification

1. Fully-Shared Model (FS-MTL)
2. Shared-Private Model (SP-MTL)



(a) Fully Shared Model (FS-MTL)



(b) Shared-Private Model (SP-MTL)

MTL with LSTM blocks:

$$s_t^k = LSTM(x_t, s_{t-1}^k, \theta_s), \tag{7}$$

$$h_t^k = LSTM(x_t, h_{t-1}^k, \theta_k) \tag{8}$$

- feed into corresponding task-specific softmax layer for classification.

$$L_{Task} = \sum_{k=1}^{K} \alpha_k L(\hat{y}^{(k)}, y^{(k)})$$

where α is the weights for each task k and L is the cross-entropy loss.

Incorporating Adversarial Training

Just like GAN to learn a generative dist $p_G(x)$ that matches the real data dist $P_{data}(x)$ via discriminative model D.(Goodfellow, Bengio, and Courville 2016)

More specifically, G generates samples from the generator distribution $p_G(x)$. and D learns to determine whether a sample is from $p_G(x)$ or $p_{data}(x)$.

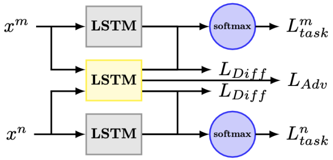


Figure 3: Adversarial shared-private model. Yellow and gray boxes represent shared and private LSTM layers respectively.

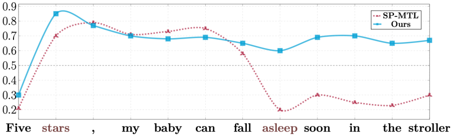
Adversarial Loss L_{adv}

- prevent task-specific fea- ture from creeping in to shared space.

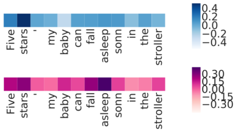
$$L_{adv} = \min_{\theta_s} \left(\lambda \max_{\theta_D} \left(\sum_{k=1}^{K} \sum_{i=1}^{N_k} d_i^k \log[D(E(x^k))] \right) \right)$$

- shared LSTM generates representation to mislead the task discriminator.
- the discriminator tries its best to make a correct classification on the type of task.

Summary



(a) Predicted Sentiment Score by Two Models



(b) Behaviours of Neuron $\mathbf{h}^s_{18}$ and $\mathbf{h}^s_{21}$

Materials and Reference I

- The teaching video of transfer learning and attention mechanism by Lee Hung-yi: https://www.youtube.com/watch?v=qD6iD4TFsdQ and https://www.youtube.com/watch?v=hYdO9CscNes
- This blog http://colah.github.io/posts/2015-08-Understanding-LSTMs/ introducing LSTM is very easily understandable.
- This blog has a comprehensive introduction to Transformer family including Attention and self-attention https://lilianweng.github.io/posts/2020-04-07-the-transformer-family/

Materials and Reference II

Ganin, Yaroslav, and Victor Lempitsky. 2015. "Unsupervised Domain Adaptation by Backpropagation." In *International Conference on Machine Learning*, 1180–89. PMLR.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Liu, Pengfei, Xipeng Qiu, and Xuanjing Huang. 2017. "Adversarial Multi-Task Learning for Text Classification." *arXiv Preprint arXiv:1704.05742*.

Liu, Shikun, Edward Johns, and Andrew J Davison. 2019. "End-to-End Multi-Task Learning with Attention." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1871–80.

Maninis, Kevis-Kokitsi, Ilija Radosavovic, and Iasonas Kokkinos. 2019. "Attentive Single-Tasking of Multiple Tasks." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1851–60.

Materials and Reference III

Rusu, Andrei A, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. "Progressive Neural Networks." *arXiv Preprint arXiv:1606.04671*.

Sinha, Ayan, Zhao Chen, Vijay Badrinarayanan, and Andrew Rabinovich. 2018. "Gradient Adversarial Training of Neural Networks." *arXiv Preprint arXiv:1806.08028*.