

Zdolność sieci neuronowych do adaptacji w uczeniu maszynowym

Paulina Tomaszewska

Seminarium z Metod Inteligencji Obliczeniowej - 28 kwietnia 2021

Outline

- 1 Introduction
- 2 Architectures
 - Shared trunk
 - Cross-talk
- 3 Self-paced learning
 - Uncertainty based
 - GradNorm
- 4 Summary

Multi-task learning

Motivation:

- compact models inspired by human ability of multi-tasking
- better representations (more universal)
- increased performance
- decreased inference time (autonomous cars)

Main problem: negative transfer

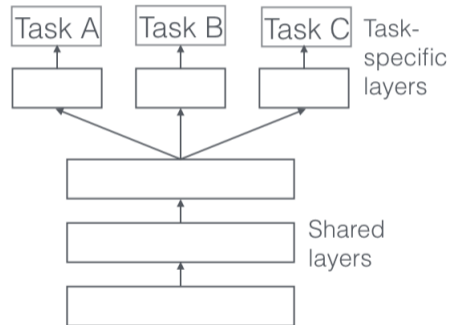


Figure 1: High level architecture in multi-task learning [7]

Initial application

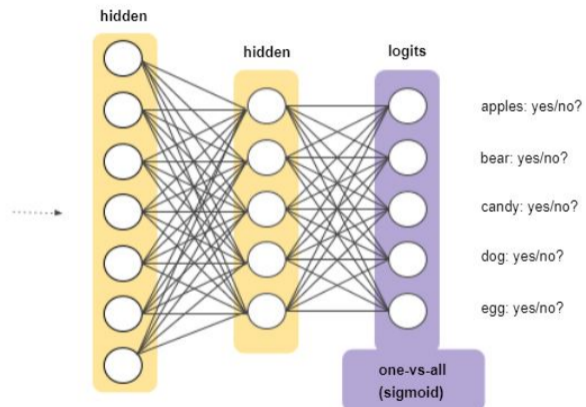


Figure 2: Multi-label architecture [1]

Shared trunk

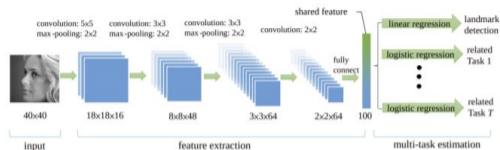


Figure 1: Architecture for TCDCN (Zhang et al., 2014). The base feature extractor is made of a series of convolutional layers which are shared between all tasks, and the extracted features are used as input to task-specific output heads.

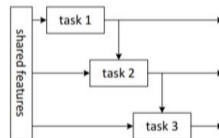


Figure 2: Illustration of Multi-task Network Cascades (Dai et al., 2016). The output of the first task is used as an input for the second task, the second task's output is used as an input for the third task, and so on.

Figure 3: Shared trunk architectures [3]

Cross-stitch Networks for Multi-task Learning [6] (CVPR 2016)¹

single shared extractor \rightarrow separate network for each task with information flow

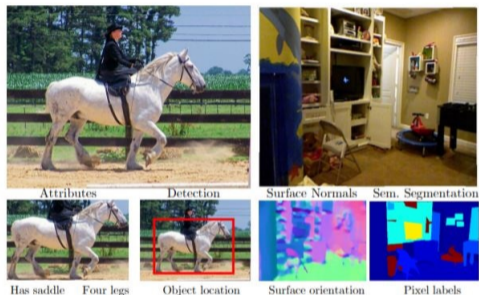


Figure 4: Problem definition [6]

¹474 citations

Cross-stitch Networks for Multi-task Learning [6] - motivation

"Given a pair of tasks, how should one pick a network architecture?"

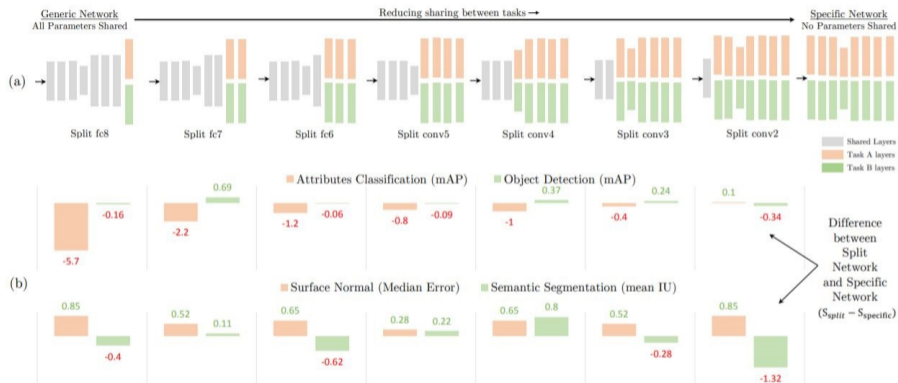


Figure 5: Analysis of performance at different split levels [6]

Cross-stitch unit

"Enumerating all possible architectures for each set of tasks is impractical. Should we have a completely shared representation between tasks? Or should we have a combination of shared and task-specific representations?"

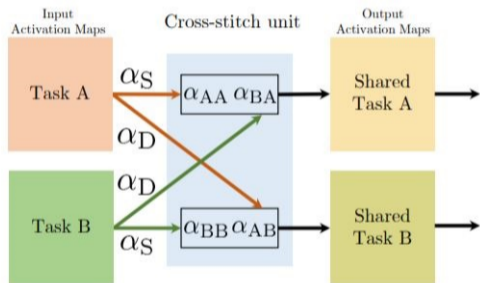


Figure 6: Cross-stitch unit [6]

$$\begin{bmatrix} \tilde{x}_A^{ij} \\ \tilde{x}_B^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} x_A^{ij} \\ x_B^{ij} \end{bmatrix}$$

Figure 7: Cross-stitch unit - formula [6]

Cross-stitch network

$$\begin{bmatrix} \frac{\partial L}{\partial x_A^{ij}} \\ \frac{\partial L}{\partial x_B^{ij}} \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{BA} \\ \alpha_{AB} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial \tilde{x}_A^{ij}} \\ \frac{\partial L}{\partial \tilde{x}_B^{ij}} \end{bmatrix}$$

$$\frac{\partial L}{\partial \alpha_{AB}} = \frac{\partial L}{\partial \tilde{x}_B^{ij}} x_A^{ij}, \quad \frac{\partial L}{\partial \alpha_{AA}} = \frac{\partial L}{\partial \tilde{x}_A^{ij}} x_A^{ij}$$

Figure 8: Cross-stitch unit - backpropagation [6]

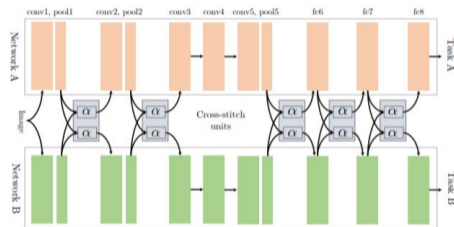


Figure 9: Cross-stitch network [6]

Cross-stitch network - initialization

- $\alpha_{S,D} \in [0, 1]$, $\alpha_S + \alpha_D = 1$ for stable learning (ensures that input and outputs are of the same order of magnitude)
- better initialize weights by training one-task networks than using transfer learning (Imagenet)

Performance on data-starved categories

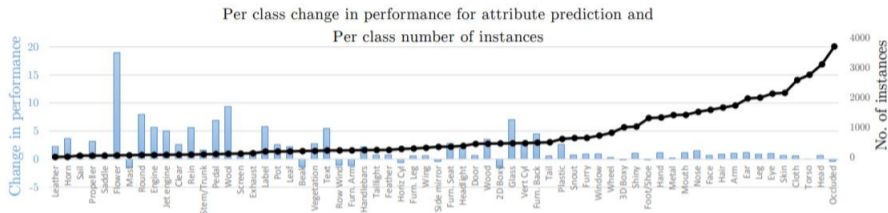


Figure 10: Performance gain in attribute detection task [6]

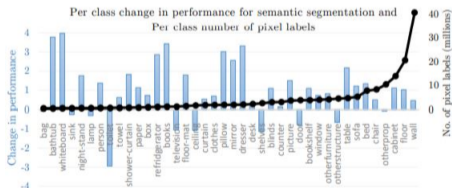


Figure 11: Performance gain in semantic segmentation task [6]

NDDR-CNN: Layerwise Feature Fusing in Multi-Task CNNs by Neural Discriminative Dimensionality Reduction

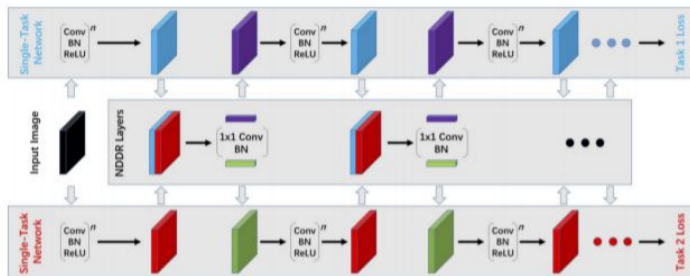


Figure 12: NDDR-CNN architecture [4]

Motivation

"What are the key components to determine whether or not MTL is better than single-task learning (STL)? In response, we identify three components: model capacity, task covariance, and optimization scheme." [8]

Baseline loss in MTL

$$L = \sum_{k=1}^K \alpha_k * L_k$$

K - number of tasks in multi-task learning

L_k - loss in single task k

α_k - coefficient corresponding to task k

Is it good idea to set coefficients uniformly?

Problem: losses from different tasks have different scales \rightarrow gradient from small loss tasks will be small making the training slow

Possible solution: use grid search to find optimal coefficients (but they are static)

Could it be done better?

Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics (CVPR 2018) [5]

Two types of uncertainties in NN:

- epistemic (could be mitigated in case of more data)
- aleatoric:
 - data-dependent, heteroscedastic
 - **task-dependent, homoscedastic**

$$L_k := L_k / \sigma^2 + \log(\sigma)$$

$$L = \sum_{k=1}^K L_k$$

σ - homoscedastic uncertainty

$\log(\sigma)$ - regularization

The bigger the uncertainty, the smaller the contribution to overall loss.

GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks (ICLM 2018) [2]

- W : The subset of the full network weights $W \subset \mathcal{W}$ where we actually apply GradNorm. W is generally chosen as the last shared layer of weights to save on compute costs¹.
- $G_W^{(i)}(t) = \|\nabla_W w_i(t)L_i(t)\|_2$: the L_2 norm of the gradient of the weighted single-task loss $w_i(t)L_i(t)$ with respect to the chosen weights W .
- $\bar{G}_W(t) = E_{\text{task}}[G_W^{(i)}(t)]$: the average gradient norm across all tasks at training time t .

We also define various training rates for each task i :

- $\tilde{L}_i(t) = L_i(t)/L_i(0)$: the loss ratio for task i at time t . $\tilde{L}_i(t)$ is a measure of the *inverse* training rate of task i (i.e. lower values of $\tilde{L}_i(t)$ correspond to a faster training rate for task i)².
- $r_i(t) = \tilde{L}_i(t)/E_{\text{task}}[\tilde{L}_i(t)]$: the relative *inverse* training rate of task i .

Figure 13: Preliminaries [2]

our gradients. Concretely, the higher the value of $r_i(t)$, the higher the gradient magnitudes should be for task i in order to encourage the task to train more quickly. Therefore, our desired gradient norm for each task i is simply:

$$G_W^{(i)}(t) \mapsto \bar{G}_W(t) \times [r_i(t)]^\alpha, \quad (1)$$

Figure 14: Target gradient norm [2]

$$L_{\text{grad}}(t; w_i(t)) = \sum_i \left| G_W^{(i)}(t) - \bar{G}_W(t) \times [r_i(t)]^\alpha \right|_1 \quad (2)$$

Figure 15: Loss related to GradNorm (second term is treated as constant during differentiation [2])

GradNorm

Algorithm 1 Training with GradNorm

Initialize $w_i(0) = 1 \forall i$

Initialize network weights \mathcal{W}

Pick value for $\alpha > 0$ and pick the weights W (usually the final layer of weights which are shared between tasks)

for $t = 0$ **to** max_train_steps **do**

Input batch x_i to compute $L_i(t) \forall i$ and

$L(t) = \sum_i w_i(t) L_i(t)$ [standard forward pass]

 Compute $G_W^{(i)}(t)$ and $r_i(t) \forall i$

 Compute $\bar{G}_W(t)$ by averaging the $G_W^{(i)}(t)$

 Compute $L_{grad} = \sum_i |G_W^{(i)}(t) - \bar{G}_W(t) \times [r_i(t)]^\alpha|_1$

 Compute GradNorm gradients $\nabla_{w_i} L_{grad}$, keeping targets $\bar{G}_W(t) \times [r_i(t)]^\alpha$ constant

 Compute standard gradients $\nabla_{\mathcal{W}} L(t)$

 Update $w_i(t) \mapsto w_i(t+1)$ using $\nabla_{w_i} L_{grad}$

 Update $\mathcal{W}(t) \mapsto \mathcal{W}(t+1)$ using $\nabla_{\mathcal{W}} L(t)$ [standard backward pass]

 Renormalize $w_i(t+1)$ so that $\sum_i w_i(t+1) = T$

end for

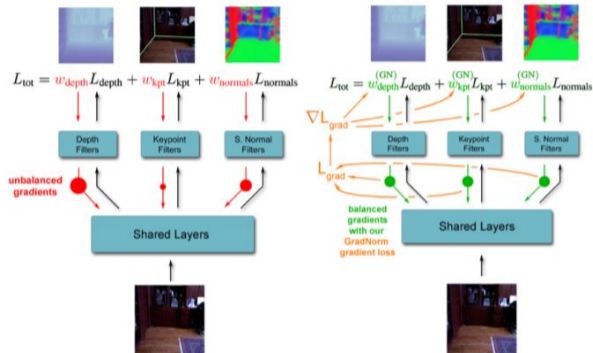







Figure 16: Scheme [2]

Summary




Multitask learning- research in 3 domains:

- task similarity
- architecture
- optimization

References I

-  <https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/one-vs-allnote={Accessed: 2021-04-25}>.
-  Zhao Chen et al. *GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks*. 2018. arXiv: 1711.02257 [cs.CV].
-  Michael Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020. arXiv: 2009.09796 [cs.LG].
-  Yuan Gao et al. *NDDR-CNN: Layerwise Feature Fusing in Multi-Task CNNs by Neural Discriminative Dimensionality Reduction*. 2019. arXiv: 1801.08297 [cs.CV].
-  Alex Kendall, Yarin Gal, and Roberto Cipolla. *Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics*. 2018. arXiv: 1705.07115 [cs.CV].

References II

-  Ishan Misra et al. *Cross-stitch Networks for Multi-task Learning*. 2016. arXiv: 1604.03539 [cs.CV].
-  Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*. 2017. arXiv: 1706.05098 [cs.LG].
-  Sen Wu, Hongyang R. Zhang, and Christopher Ré. “Understanding and Improving Information Transfer in Multi-Task Learning”. In: *International Conference on Learning Representations*. 2020. URL: <https://openreview.net/forum?id=SylzhkBtDB>.