Mitigating Task Interference in Multi-Task Learning via Explicit Task Routing 
with Non-Learnable Primitives 
(Supplementary Materials)

Chuntao Ding\textsuperscript{1*} Zhichao Lu\textsuperscript{2†} Shangguang Wang\textsuperscript{3} Ran Cheng\textsuperscript{4} Vishnu N. Boddeti\textsuperscript{5}
\textsuperscript{1} Beijing Jiaotong University \textsuperscript{2} Sun Yat-sen University \textsuperscript{3} Beijing University of Posts and Telecommunications \textsuperscript{4} Southern University of Science and Technology \textsuperscript{5} Michigan State University
chuntaoding@163.com \{luzhichaocn, ranchengcn\}@gmail.com sgwang@bupt.edu.cn vishnu@msu.edu

This appendix includes the following:
1. Extended description of Related Work in Section A.
2. Additional results of configurations on NLPs-based feature extraction in Section B.
3. Extended description of datasets and baseline methods in Section C.

A. An Extended Description of Related Work
Existing methods related to MTL architectures can be classified into encoder or decoder-focused ones. Encoder-focused approaches primarily lay emphasis on architectures that can encode multi-purpose feature representations through supervision from multiple tasks. Such encoding is typically achieved, for example, via feature fusion \cite{5,12,14}, branching \cite{7,10,11,19}, self-supervision \cite{3}, shared and task-specific modules \cite{8,13}, filter grouping \cite{1}, filter modulation \cite{6, 27}, task routing \cite{12, 16, 18}, or neural architecture search \cite{4}. Decoder-focused approaches start from the feature representations learned at the encoding stage, and further refine them at the decoding stage by distilling information across tasks in a one-off \cite{22}, sequential \cite{24}, recursive \cite{25}, or even multi-scale \cite{20} manner. Due to inherent layer sharing, the approaches above typically suffer from task interference and negative transfer \cite{21}.

In the context of MTL, our explicit task routing layer is conceptually related to \cite{8, 13}, but notably different in motivation and design. First, both of these two approaches operate on the features obtained from a shared backbone to extract task-specific features. In contrast, our shared and task-specific branches operate in parallel and extract features from the common features extracted by the non-learnable layer. Second, these existing approaches utilize attention mechanisms to distill task-specific features from the shared features, while we use lightweight $1 \times 1$ convolution for the same purpose. Third, our explicit task routing layer is tailored to exploit the non-learnable layer for MTL optimally. Finally, unlike baselines, our multi-branch design affords simple and explicit control over the ratio of shared and task-specific parameters.

Additionally, our work is also closely related to reparameterized convolutions for multi-task learning (RCM) \cite{6}, which first introduced the concept of using non-learnable convolutional filters for MTL. However, there are three notable differences. First, the non-learnable layer of RCM only includes standard convolution, while we consider other non-learnable primitives such as pooling, identity, and additive noise \cite{23} operations. Second, RCM uses pre-trained network weights to initialize non-learnable convolutional filters, while in our case they are sampled from a random distribution. Relying on pre-trained weights limits RCM’s ability to reduce the model size and its generalizability to architectures without readily available pre-trained weights. Finally, there is no collaboration between tasks in RCM as it only comprises task-specific modulators, while we utilize a shared branch to help tasks use each other’s training signals. Having both shared and task-specific branches allows tasks to amortize parameters that are commonly useful across multiple tasks, thereby minimizing redundancy in the task-specific branches, unlike RCM. Moreover, our method also offers fine-grained control over the ratio of parameters that are shared or task-specific.

B. Results of NLPs-based feature extraction
In this section, we first provide the full version of Table 1 from the main paper in Table S1, showing the effect of different configurations of NLPs on CelebA multi-attribute classification. Then, we present the effect of different hyperparameter settings of NLPs on NYU-v2 dense prediction MTL problem in Figure S1.

\textsuperscript{*}Work done as a visiting scholar at Michigan State University.

\textsuperscript{†}Corresponding author
C. Description of Datasets and Baselines

In this section, we first provide the additional details of CelebA and Cityscapes datasets in Table 1 and 2, respectively. Then, we provide a brief overview of the baseline methods that we compared against in this work, as follows:

- STL: single task learning with one network for each task.
- Hard sharing: standard multi-task learning, i.e., a fully shared network with uniform task weighting.
- GradNorm [26]: a MTL method with a fully shared network and learnable tasks weighting.
- MGDA-UB [17]: a multi-objective alternative to MTL with a fully shared network.
- Task Routing [18]: a parameter partitioning method with randomly initialized binary masks.
- Max. Roaming [15]: another parameter partitioning method with dynamic masks.
- Cross-stitch [14]: a soft-sharing method with feature fusion.
- MTAN [8]: another soft-sharing method with attention.

![Figure 1](image-url)

Figure 1. Effect of different hyperparameters of individual NLP on NYU-v2 dense prediction MTL problem. For each sub-figure (a) - (e), we show the semantic segmentation mIoU (↑), depth estimation absolute error (↓), and surface normal estimation mean error (↓).


