An Enhanced Deep Neural Network Modeling Framework for Multi-Task Learning

Weiwei Zhou*, Chengkun Ling*, Jiada Lu*, Xiaoyun Gong, Lina Cao, Weifeng Wang
China Telecom Cloud
{zhouweiwei, lingchengk, lujiada, gongxiaoyun, caoln, wangweifeng}@chinatelecom.cn

Abstract

In this paper, we present an enhanced version of a multi-task network architecture that accomplishes semantic segmentation, image classification, and object detection simultaneously within a single model. Our proposed model leverages the All-in-One training strategy, which allows for efficient joint learning of multiple tasks. To construct this architecture, we employ ConvNeXt-large as the backbone network, UPerNet as the segmentation head, CBAM as the classification head, and YOLOv5 head as the object detection head. Through extensive experimentation, we have demonstrated the effectiveness of our approach, leading to our model achieving 1st place in the multi-task track of the CVPR 1st foundation model challenge.

1. Introduction

Recent technological advancements have facilitated the emergence of large-scale pre-trained models that undergo training on extensive datasets. These models possess the capacity to acquire more generalized knowledge by effectively processing immense volumes of data. As a result, fine-tuning these models for novel tasks holds immense potential for achieving favorable outcomes. Moreover, pre-trained models can leverage their existing knowledge to augment their performance on tasks that are closely related but distinct, thereby substantially diminishing the requisite amount of task-specific training data.

Despite the notable advantages associated with the utilization of pre-trained models, the repetitive nature of fine-tuning these models for each new task entails significant costs. Consequently, substantial research efforts have been directed towards the development of techniques that aim to minimize the extent of required fine-tuning while concurrently achieving exceptional performance. One such technique is the Unified Feature Optimization (UFO) technology, which was introduced by Baidu [8]. This innovative technology exhibits the ability to handle multiple tasks through training on data encompassing diverse tasks. Specifically, the VIMER-UFO All-in-One multi-task training scheme not only enhances the performance of individual tasks by leveraging cross-task information but also obviates the need for downstream task-specific fine-tuning processes.

To improve the generalization ability of models through multi-task joint training, the CVPR 1st foundation model multi-task challenge was organized. Participants were required to design a unified model capable of classification, detection, and segmentation for traffic scenarios. In this paper, we present a multi-task model based on the baseline framework given by the organizer [5]. The baseline provides a multi-task model with ViT-base [1] as the backbone and corresponding heads for each task. For the task of object detection, DINO [10] is utilized. In order to perform semantic segmentation, SETR [11] is used. For the image classification task, a straightforward approach is adopted, where a fully connected layer is employed to predict the image's corresponding category.

In this paper, we enhance the baseline by proposing a novel multi-task model. We incorporated ConvNeXt [3] as the backbone and redesigned the head for each task. Specifically, we utilized UPerNet [9] for segmentation, CBAM [7] and fully connected layers for classification, and YOLOv5 [2] for detection. This modification resulted in an approximate 11% improvement in the overall score. During the inference phase, we employed the Test Time Adaptation strategy (TTA), which encompassed the application of multi-scale and horizontal flip techniques. As a result, our model surpassed the baseline and achieved the top-ranking position in the competition.

2. Method

2.1. Problem Definition

The objective of this track is to enhance the model's generalization ability through multi-task joint training and resolve conflicts between different tasks. In the context of

*These authors contributed equally to this work.

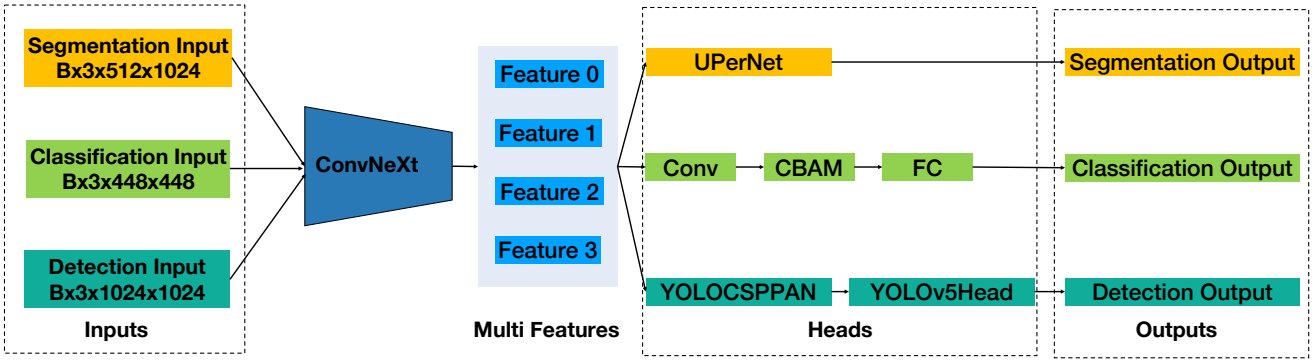

Figure 1. The architecture of our proposed model. We utilize the ConvNext model as the shared backbone. Each task has its own independent input shape, and the backbone generates four distinct levels of features that can be utilized by the tasks. In terms of segmentation, we employ a UPerNet head. As for classification, we utilize convolution layers along with the CBAM attention module. In the case of detection, we leverage YOLOCSPPAN and YOLOv5 Head to produce object boxes.

Task	Train	Val	Test	Avg size
segmentation	7000	674	1000	[1280, 720]
classification	8144	786	8041	[572, 307]
detection	6103	610	3067	[2048, 2048]

Table 1. The distribution of the number of datasets.

traffic scenarios, this track specifically selects three representative tasks: classification, detection, and segmentation for All-in-One joint training. This allows the single model to possess the ability to perform all three tasks while achieving performance that surpasses that of specific single-task models.

In the leaderboard A, the distribution of the training, validation, and test datasets is presented in Table 1. It is worth noting that the tasks and data pertaining to leaderboard B remain undisclosed. To bolster the model's robustness, it is advisable to avoid excessive dependence on strategies tailored exclusively to the dataset. Rather, the emphasis should be placed on the development of improved network architectures and training strategies.

2.2. Modeling

For this particular challenge, the selection of an appropriate backbone stands as a critical factor. The initial choice for the backbone in the baseline model was ViT-base, comprising a mere 86 million parameters. However, considering the current task at hand, it is plausible that the model may be slightly undersized. Hence, taking into consideration both memory utilization and experimental findings, we employed ConvNeXt-L [3] as the backbone for our multi-task model, which possesses a larger parameter count, amounting to 198 million.

The architecture of our proposed model is depicted in Figure 1. Each task is assigned its own distinct input shape. To extract features from various levels, we employ ConvNeXt as the shared backbone, which effectively generates four distinct features.

For the segmentation task, we employ the UPerNet head [9], which accepts four features as input and integrates an auxiliary loss to enhance the training process. The loss function employed in this context is an unweighted cross-entropy loss.

For the classification task, our approach incorporates a structured architecture consisting of a convolutional layer, a Convolutional Block Attention Module (CBAM) [7], and a fully connected layer. This architecture takes the output of the final layer of ConvNeXt as its input. To optimize the model during training, an unweighted cross-entropy loss function is employed.

For the detection task, we leverage the well-established object detection algorithm YOLOv5 [2]. To enhance the fusion of features from multiple layers, we employ YOLOCSPPAN. Subsequently, prediction boxes are generated using YOLOv5 Head. To train the detection task, we utilize YOLOv5 Loss.

2.3. Training strategy

We employ the training strategy outlined in Algorithm 1. In dataloader, each batch is a composite batch that contains data for segmentation, classification, and detection. These batches, referred to as "task-batch" have varying batch sizes. During each training step, we initially forward the backbone to extract multiple features. Subsequently, these features are passed through task-specific heads to generate predictions. The task-specific loss function is then employed to compute the loss, followed by the backward propagation step. Once all tasks have been processed, the model parameters are updated.

Algorithm 1 Train strategy of multi-task model

N is total epochs, M is steps of each epoch.
for epoch $= 1$ *to* N **do**
  for step $= 1$ *to* M **do**
    for task-batch *in* composite-batch **do**
      features = backbone.forward(task-batch)
      task-pred = task-head.forward(features)
      loss = task-loss-func(task-pred, task-target)
      loss.backward()
    end for
    optimizer.step().
  end for
end for

3. Experiments

3.1. Implementation Details

The proposed multi-task model is implemented with the deep learning framework PaddlePaddle [4]. The model training process was conducted on a machine equipped with six A100 GPUs, each possessing 80GB of memory.

Our network was trained with hyperparameters listed as follows. For segmentation tasks, classification tasks, and detection tasks, we allocated batch sizes of 3, 6, and 2, respectively. The training process employed the AdamW optimizer with an initial learning rate of 0.0001 and a weight decay of 0.0001. Additionally, we incorporated a Warmup learning rate decay for the initial 200 steps, followed by Cosine decay over 100 epochs. These hyperparameters were carefully selected to optimize performance across a diverse range of tasks.

3.2. Data augmentation

To enhance the quality and diversify the input images utilized in the backbone, we implemented task-specific data augmentation techniques for each respective task.

In order to improve the images utilized for the segmentation task, we employed the subsequent data augmentation techniques:

- Randomly scale the image by a factor of 0.5 to 2.0.
- Randomly rotate the image at an angle between -15° and 15°, and fill in with a value of 127.5 and a label of 255 (background).
- Randomly crop a region of size [1024, 512], and if the image size is smaller than [1024, 512], fill it with a value of 127.5 and a label of 255 (background);
- Horizontally flip the image with a probability of 0.5.
- Randomly transform brightness, contrast, and exposure.
- Add blur with a probability of 0.3.

Team	Score	Seg	Cls	Dec
huster	**0.8749**	**0.71214**	**0.96045**	0.95209
toothpick	0.8717	0.70453	0.95601	0.95452
Ours	0.8617	0.67104	0.95784	**0.95615**
Baseline	0.7461	0.5513	0.91683	0.77012

Table 2. The test scores of the leaderboard A. The bold fonts indicate the best results.

Team	Score	Cls	Dec
Ours	**0.771**	0.962	**0.580**
huster	0.760	**0.97.3**	0.54.7
toothpick	0.745	0.962	0.527

Table 3. The test scores of the leaderboard B. The bold fonts indicate the best results.

To address classification tasks, we implemented the following data augmentation methods:

- Random erase image with a probability of 0.5.
- Horizontally flip the image with a probability of 0.5.
- Colorspace saturation with a probability of 0.5.

To address detection tasks, we employed the following data augmentation methods:

- Perform random crops on the original image, ensuring that the size after cropping is within the range of [0.3, 0.6] of the original image.
- Resize to [1024, 1024].
- Horizontally flip the image with a probability of 0.5.

3.3. Test Time Augmentation (TTA)

In this challenge, we employed the TTA (Test Time Augmentation) technique during the inference.

For the segmentation task, in order to effectively adapt to the multi-scale training, we employed the multi-scale and horizontal flip TTA strategy. Additionally, we set the default input size to match the original image size.

For the classification task, we also utilized the TTA strategy of multi-scale and horizontal flip, while keeping the input size as [448,448].

For the detection task, To ensure better scale matching during training, we also incorporated the TTA strategy of multi-scale and horizontal flip during testing inference. Additionally, we employed the Weighted Boxes Fusion (WBF) [6] method as the post-processing module for merging the bounding boxes. The initial input size was set to [2048,2048].

3.4. Experimental Results

The overall results can be observed from Table 2 and Table 3, where the top 3 teams' scores significantly surpass

Method	Scales	mIoU on Val
Original	[1.0]	0.6565
MS	[0.9, 1.0, 1.2]	0.6765
MS	[1.0, 1.2, 1.6]	0.6786
MS + Flip_H	[1.0, 1.2, 1.6]	**0.6804**

Table 4. TTA results for the segmentation task on the validation dataset. The bold fonts indicate the best results. (MS represents multi-scale, and Flip_H represents flip horizontally.)

Method	Scales	Acc@1 on Val
Original	[1.0]	0.9578
MS	[0.6, 0.8, 1.0]	0.9565
MS	[1.0, 1.2, 1.4]	0.9608
MS + Flip_H	[1.0, 1.2, 1.4]	**0.9656**

Table 5. TTA results for the classification task on the validation dataset. The bold fonts indicate the best results.(MS represents multi-scale, and Flip_H represents flip horizontally.)

Method	Sizes	mAP50 on Val
Original	[2048]	0.949
MS	[2048, 1344, 1504]	0.952
MS	[2048, 1568, 2240]	0.956
MS+ Flip_H+NMS	[2048, 1568, 2240]	0.960
MS + Flip_H+WBF	[2048, 1568, 2240]	**0.966**

Table 6. TTA results for the detection task on the validation dataset. The bold fonts indicate the best results. (MS represents multi-scale, and Flip_H represents flip horizontally.)

the baseline. Our approach ranks 3rd on leaderboard A and achieves a remarkable leap on leaderboard B, securing the 1st position. It is worth noting that our detection scores consistently rank first, far surpassing other teams on leaderboard B.

In addition, during the testing phase, we discovered that utilizing the TTA (Test Time Augmentation) method significantly improves the inference performance, as shown in Tables 4, 5, and 6. Employing multi-scale inference and horizontal flipping resulted in an approximate score improvement of 1 percent. In the detection task, employing the WBF post-processing method further yielded a 1 percent improvement.

4. Conclusion

In this paper, we propose a multi-task model architecture capable of simultaneously performing segmentation, classification, and detection tasks. We utilize ConvNeXt-L as the backbone, UPerNet as the segmentation head, CBAM+FC as the classification head, and YOLOv5 as the detection head. During the testing phase, we employ the TTA (Test Time Augmentation) method with multi-scale and horizontal flipping. Ultimately, our team achieved first place in the CVPR 1st foundation model challenge.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[2] Glenn Jocher. YOLOv5 by Ultralytics, May 2020. 1, 2

[3] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 1, 2

[4] Yanjun Ma, Dianhai Yu, Tian Wu, and Haifeng Wang. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing*, 1(1):105–115, 2019. 3

[5] Yifeng Shi, Feng Lv, Xinliang Wang, Chunlong Xia, Shaojie Li, Shujie Yang, Teng Xi, and Gang Zhang. Open-transmind: A new baseline and benchmark for 1st foundation model challenge of intelligent transportation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6327–6334, 2023. 1

[6] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117, 2021. 3

[7] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 1, 2

[8] Teng Xi, Yifan Sun, Deli Yu, Bi Li, Nan Peng, Gang Zhang, Xinyu Zhang, Zhigang Wang, Jinwen Chen, Jian Wang, et al. Ufo: Unified feature optimization. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*, pages 472–488. Springer, 2022. 1

[9] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018. 1, 2

[10] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022. 1

[11] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021. 1