End-to-End Multi-Task Learning Driven by a Novel Task-Specific Feature Learning - An Application to Scene Understanding

Paper Id - 5178

Abstract

We tackle the problem of deep end-to-end multi-task learning (MTL) for jointly performing image segmentation and depth estimation from monocular images. It is proven that learning several related tasks together helps in attaining improved performance per task than training them autonomously. This is due to the fact that related tasks share important feature properties among themselves, which the MTL techniques can effectively explore and share among the tasks. In this paper, we are interested in segregating task-centric feature learning stage from a learnable task-generic feature space. To this end, we follow the typical U-net based encoder-decoder architecture where the densely connected deep convolutional neural network (CNN) based feature encoder is shared among the tasks while the soft attention based task-specific decoder modules produce the desired outputs. Additionally, we encourage cross-talk between the tasks by introducing cross-task skip connections at the decoder end. Besides, we perform weight learning for the task-specific loss functions in the final cost measure. We find that the proposed framework offers learning of improved task-specific features with respect to the traditional approaches which are highly dependent on several hyper-parameters tuning. We validate the proposed framework on the challenging CityScapes and NYUv2 datasets, where our method sharply outperforms the current state-of-the-art.

Introduction

The recent times have witnessed an unprecedented performance gain in visual inference tasks with the application of deep CNN models. This can be attributed to the availability of large-scale labeled training samples specific to a given task (Krizhevsky, Sutskever, and Hinton 2012). However as opposed to the traditional single task-specific CNN frameworks, the notion of learning multiple tasks together by exploring their similarities has secured much potential (Zhang and Yang 2017). This is predominantly inspired by the human learning abilities where one often applies the knowledge learned from previous tasks to assimilate a new task. From theoretical point of view, the inductive bias is provided by one of the competing tasks (auxiliary task) which causes the model to prefer hypotheses that properly explain more than one tasks within an unified framework. The idea is particularly eminent in the domain of visual computing as complementary information can be inferred by sensibly exploring the local image properties. For example, semantic image segmentation and depth estimation are very related in nature and learning both of them together is expected to help in improved overall scene understanding by jointly apprehending the scene semantics and geometry.

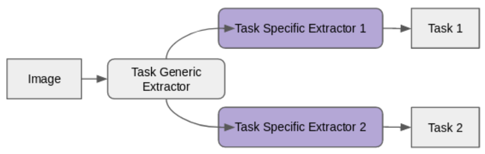

Figure 1: Encoder-Decoder based standard MTL framework with shared encoder and task-specific decoders.

We note that the feature representations in case of MTL can broadly be divided into *task-shared* and *task-specific*, respectively. In this regard, the deep CNN based MTL approaches (Doersch and Zisserman 2017), (Liu, Johns, and Davison 2019), (Misra et al. 2016) are primarily based on the notion of parameter sharing (hard or soft). Ideally, there can be a common feature extractor for all the tasks or we can have separate task-specific feature extractors. In the second case, it is further demanded that the feature extractors should resemble each other through some soft similarity constraints. However, the question of *how to share* the feature extraction network among the tasks is still considered as one of the key challenges in MTL. Nonetheless, it is encouraged that the feature extraction module should judiciously balance between learning more generalisable task-shared features in order to avoid model over-fitting and features tailored for each task to avoid any under-fitting issue. In this regard, the idea of encoder-decoder based MTL models with task-shared encoder and task-specific decoders have shown some promises (Figure 1). However, learning the task specific features in the hard way from the shared feature space may incur redundancy as not all of the shared features are equally influential for all the tasks. In this case, we foresee two possible directions: i) the notion of *task-specific attention* modeling can help in better capturing the task-specific features, ii) in addition to the shared encoder, we also feel the task-specific feature decoders should communicate among each other since many a time the auxiliary task can provide important high-level complementary cues for better learning of the main task.

Another critical issue in this respect arises from the fact that: *how to assign weights to the task-specific loss terms* in the overall cost function. Most of the previous approaches rely on manual weight tuning, however, finding the optimal weights can be prohibitively expensive and difficult to unravel through manual processing. Instead, learning the task-specific weights as part of the model training process with minimum overhead is deserved.

Inspired by both the aforementioned issues regarding efficiently training a deep MTL model, we propose an U-Net (Ronneberger, Fischer, and Brox 2015) based encoder-decoder model (MT-UNet) in this paper. However, we follow an intuitive network design strategy which guarantees better learning of both the task-generic and task-specific feature representations. In particular, the shared feature encoder module is devised as a dense convolution network which can intelligently capture very primitive to very abstract image features together to produce a more grounded shared feature space. We note that the high-level CNN features are generally devoid of fine-grained image information like edges, curves etc, thus creating confusions for dense prediction tasks like segmentation or depth perception. Hence, we have to ensure that such low-level information is efficiently propagated through the network. The densely connected CNN is of much help in this respect. On top of this feature space, separate decoder streams are deployed to learn the task-oriented feature representations. In order to ensure better learning of the task-specific features, we further propose the followings: i) separate task-specific attention modules are utilized. ii) sophisticated skip connections are used between decoders for allowing cross-task information sharing. While the shared feature extractor is devoted to learn the commonalities among the tasks in a generic sense, the proposed decoder modules cooperate among each other for learning better task-specific abstractions.

In order to mitigate the problem of assigning weights to the individual loss terms, we subsequently propose to learn the loss weights as a part of the optimization process where the weight terms are updated using the standard gradient descent based optimizer. For our case, we find that although one task overshadows the other initially, they tend to have similar weights as the training progresses. As a whole, we summarize the major contributions as follows:

- We propose an intuitive deep U-net based MTL framework which offers better task-generic to task-specific feature learning as endorsed by the experimental results.
- While we design the shared feature encoder in terms of densely connected CNN, we propose to consider attention based decoder modules for learning the task-specific features. We also introduce cross-decoder skip connections, something which further aid in improved task-specific feature learning. Additionally, adaptive weight learning for the tasks in the final objective function is also considered.
- We utilize the model for joint segmentation and depth estimation from monocular images on the CityScapes (Cordts et al. 2016) and NYUv2 (Silberman et al. 2012) datasets, where our model outperforms the literature as well as the baseline CNN models trained on single tasks.

Related Works

Multi-task learning: MTL (Caruana 1997), (Evgeniou and Pontil 2004), (Kumar and Daume III 2012) has been regarded as one of the cost-effective solutions for deciphering several tasks simultaneously. Through this process, the aim of MTL is to improve the learning capabilities for each of the task models by efficiently exploring the complementary and shared information jointly present in all the tasks. Earlier, MTL was primarily solved using traditional feature transformation based approaches such as latent support vector machine (SVM) for classification (Zhu, Chen, and Xing 2011), Bayesian Matrix Factorization (Yuan 2011), Task Clustering (Jacob, Vert, and Bach 2009), Matrix Decomposition (Chen, Zhou, and Ye 2011), to name a few.

Subsequently, the traditional ad-hoc approaches have been replaced by the deep learning techniques which are greatly benefited by adhering to their feature learning perspectives. MTL approaches developed in conjunction with the deep CNN models have successfully been implemented in problems concerning joint semantic segmentation, depth prediction, and surface normal estimation (Eigen and Fergus 2015) (Misra et al. 2016), preferably from monocular images. In this respect, majority of the CNN based frameworks are designed in the encoder-decoder fashion. Cross-stitch network (Misra et al. 2016) performs MTL by fusing the cross-tasks neural activations. (Kokkinos 2017) solves a number of classification and regression tasks within a common framework. Learning each task's weights automatically has also drawn attention given that manually setting the weights is non-trivial (Kendall, Gal, and Cipolla 2018)(Liu, Johns, and Davison 2019).

Attention Module: The attention modules in general help in highlighting visually interesting regions in images. Typically, attention modules are deployed in either of two ways: soft or hard attention. In *soft attention*, an attention mask is created which is learned throughout the training. Hadamard product between this attention mask and feature map is computed. As desired, this suppresses the irrelevant areas of image by attenuating the respective feature values. On the other hand, *hard attention* processes single patch of an image at a time.

The deep CNN models can be extended to support the attention modeling along with end-to-end feature learning. In this regard, Convolutional block attention module (CBAM) (Woo et al. 2018), local attention masks (Harley, Derpanis, and Kokkinos 2017), attention U-Net (Oktay et al. 2018), multi-task attention network (MTAN) (Liu, Johns, and Davison 2019) are some of the popular attention modeling variants. CBAM introduces an adaptive feature refinement method by multiplying the input space with the output of the channel and spatial attention masks. Similarly, the attention gate (AG) of U-Net is a grid-based hard attention approach and the extracted information from AG units are fused using skip-connections. On the other hand, MTAN utilizes the soft-attention module, which works on soft parameter sharing within an MTL environment.

How are we different?: The existing method most similar to us is MTAN (Liu, Johns, and Davison 2019) which also follows the attention based encoder-decoder setup for MTL. Overall, we choose to follow the U-net driven architecture over the VGG-16 based segnet model followed in (Liu, Johns, and Davison 2019) since U-net model offers better feature space exploration by merging the encoder and decoder layers during up-sampling. In addition, i) while (Liu, Johns, and Davison 2019) uses attention modules in all the encoder and decoder branches, we restrict our model to have attention based decoder branches only. On the other hand, we use a dense architecture at the feature encoder end for efficiently combining more trivial to more abstract image features, ii) we allow cross-task information sharing in terms of cross-decoder skip connections, and iii) we introduce a simple approach for task-specific weight learning with minimum overhead. Finally, we find that our model sharply outperforms (Liu, Johns, and Davison 2019) for both the datasets.

Proposed Methodology

Preliminaries: The objective is to develop a neural network model which is capable of simultaneously learning different but related tasks mainly by utilizing task-specific attention modules over a shared feature space. Let us consider a multi-task (with T tasks) learning dataset $\mathbf{X} = \{x_i, \{y_i^t\}_{t=1}^T\}$, where $x \in \mathcal{X}$ denotes the input space and $y^t \in \mathcal{Y}^t$ is the output corresponding to the t^{th} task. In our experiments, we fix $T = 2$ for image segmentation and depth estimation while $\mathcal{X}$ is the space of RGB images. We further note that ours is a homogeneous MTL setup since all the tasks are trained on the same training data. We also follow the hard feature sharing between the tasks given the shared feature encoder. Under this setup, we are interested to i) model a shared encoded feature space, and ii) model separate task-specific decoder modules on top of the shared feature space. In the following, we detail the workings of all the encoder and decoder modules.

Overall Model Architecture

The proposed model is built upon the U-Net architecture (Ronneberger, Fischer, and Brox 2015). By design, it consists of two major modules, the shared global feature learning network ($f^E(; , \theta^E)$) and separate task-specific decoders ($f_t^D(; , \theta_t^D)$) corresponding to T tasks and $\theta's$ define the learnable parameters. We further note that f^E and f_t^D are realized in terms at deep conv. networks. Each of the corresponding encoder and decoder blocks (for both the decoder streams separately) are furthermore connected by bridge connections which directly transfers a given encoder-block feature maps to the respective decoder end (Figure 2). Precisely, for a given input x, the encoder and t^{th} decoder outputs are obtained as:

$$\left.\begin{aligned} \hat{x} &= f^E(x, \theta^E) \\ \hat{y}_t &= f_t^D(\hat{x}, \theta_t^D) \end{aligned}\right\} \tag{1}$$

Shared feature encoder: We purposefully follow a dense CNN architecture for the feature encoder part (Figure 2). To this end, we note that the convolutional kernels corresponding to the initial conv. layers extract more low-level image features while the more deeper level conv. layers focus on abstract high-level feature learning. Since we are interested finally in pixel-level structured dense prediction tasks, we require a proper balance between more generic to more abstract features in the task-generic feature space. From a different point of view, given the large number of trainable parameters in our model, the problem of vanishing gradients may arise. However, a dense model with multiple skip connections in this respect guarantees better gradients flow during the backward propagation stage of training. Precisely, the encoder follows a dense-block network architecture where dense forward connections exist between all pairs of conv. layers. The final encoder block is subsequently connected to the shared latent feature space which is again constructed in terms of a conv. layer. As already mentioned, $\hat{x}$ represents the encoded feature representation corresponding to a given x. Learning the encoder means obtaining the optimal values for θ^E minimizing the final multi-task loss function.

Decoder end: For T tasks we construct T separate decoder modules. In our case, the two decoders take care of the segmentation and depth estimation tasks, respectively. Both the decoder streams follow symmetric structure with respect to the shared encoder f^E. First, we define the loss measures used in both the decoder ends and subsequently discuss the proposed modifications.

a) Decoder for depth estimation: Let us consider the decoder module $f_1^D(; , \theta_1^D)$ which is assigned to the depth estimation job given x. Strictly speaking, we have to ensure that output resolution of $f_1^D(x)$ should match that of x for such a structured prediction task. We follow the standard ℓ_1-norm based distance to define the depth loss as follows:

$$\mathcal{L}_1(y^1, \hat{y}^1) = \frac{1}{HW} \sum_{j=0}^{H-1} \sum_{k=0}^{W-1} || \, y^1(j,k) - \hat{y}^1(j,k) \, ||_1^1 \tag{2}$$

where y^1 and $\hat{y}^1$ define the ground-truth and predicted depth masks for a given x and (H, W) mentions the number of rows and columns of x.

b) Decoder block for semantic segmentation: On the other hand, the other decoder $f_2^D(:, \theta_2^D)$ is deployed for the semantic segmentation task. Note that this decoder is very much similar to f_1^D except, it deals with a multi-class prediction problem. Hence, cross-entropy loss function is considered in order to train f_2^D. Typically, considering that there exists pixels from C semantic categories in $\mathbf{X}$, the segmentation loss is:

$$\mathcal{L}_2(y^2, \hat{y}^2) = -\frac{1}{HW} \sum_{j=0}^{H-1} \sum_{k=0}^{W-1} y^2(j,k) \log \hat{y}^2(j,k) \tag{3}$$

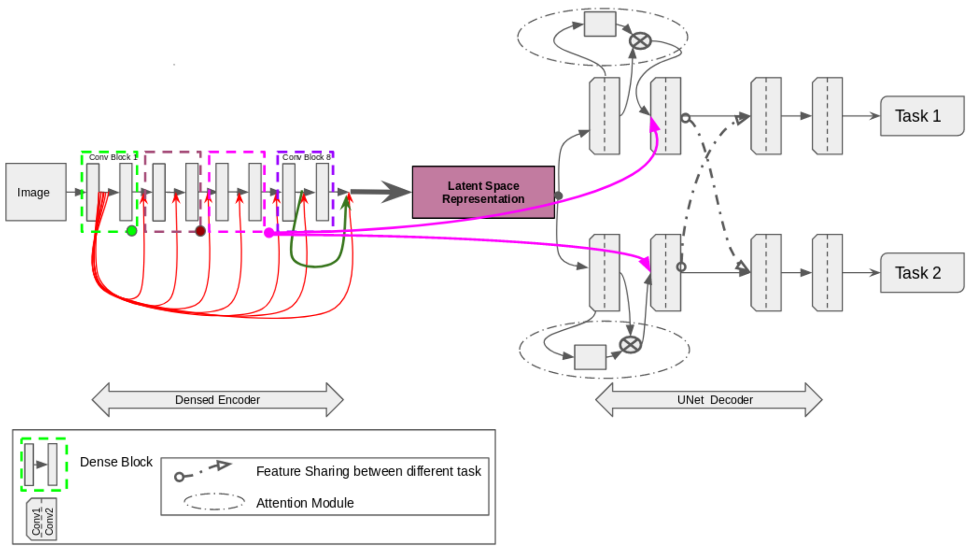

Figure 2: The proposed model with U-net backbone. The shared encoder follows the densenet architecture while there are two task-specific decoders. Attention is applied in each of the decoder blocks. Cross-talk between the decoders is also depicted. The skip connection between the respective encoder and decoder blocks; for example, the connection is shown by pink color. Red and dark green colors indicate dense connections. The pair of blocks enclosed in green, maroon, pink and violet are convolutional blocks of the encoder.

where, y^2, $\hat{y}^2$ are the ground truth labels and predicted segmentation maps for x.

Additional novel design strategies followed in the decoder end: In addition to the U-Net based architecture mentioned above, we specifically propose two extensions in the decoder end to ensure the learning of improved task-specific features. In particular, i) each of the decoder streams has a dedicated learnable attention module to better highlight the task-related feature maps, and ii) we find that the segmentation and depth estimation tasks have cooperative nature as they influence each other tremendously. For example, adjoining pixels with similar depth can be considered to form a segment and vice-versa. Hence, we allow cross-decoder skip connections between certain layers of f_1^D and f_2^D to aid in the information sharing process. In the following, both these aspects are elaborated.

- **Task specific attention**: We apply attention modules with respect to each of the convolution blocks for both the decoders separately. Any attention model could have been utilized in this respect, however, Our attention module is inspired by (Liu, Johns, and Davison 2019) since it considers the effects of all the feature maps within a given block in an incremental fashion. Ideally, for a feature map p from a given layer, the desired output after attention is $a \odot p$ where the learnable attention mask a takes value in the range $[0, 1]$. The learning of the attention scores is performed in a self-supervised fashion.
- **Cross-talk between the decoders**: We subsequently introduce cross-decoder skip connections. Practically, such a connection goes from a deeper level conv. layer of a given decoder to a middle level conv. layer of the other decoder. To this end, the notion of cross-task gradient flow is found to aid in learning better task-specific features as observed from the per task accuracy measures.

Overall training and Inference

In MTL, the total loss can be defined in various ways such as just adding all the loss terms or weighted sum of loss terms with learnable weights etc. In this paper, we analyzed both types of total loss computation. In the following, both the schemes are mentioned.

- The linear combination of loss terms with equal weights:
$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 \tag{4}$$
- The total loss as a weighted sum of the component loss measures:
$$\mathcal{L} = \frac{W_1\alpha_1}{(W_1\alpha_1 + W_2\alpha_2)}\mathcal{L}_1 + \frac{W_2\alpha_2}{(W_1\alpha_1 + W_2\alpha_2)}\mathcal{L}_2 \tag{5}$$
where we consider the normalized weights given learnable weights W_1 and W_2. Also, $\alpha_1 = \frac{\sigma_1(t-1)+\epsilon}{\sigma_1(t-2)+\epsilon}$ where $\sigma_1(t)$ is the average variance of $\mathcal{L}_1$ over a fixed number of previous epochs till the t^{th} one and ϵ is a small constant to avoid any numerical instability. Note that this approach reduces the notion of uncertainty during learning a specific task by deliberately tuning the loss variance over the iterations.

We train the model end-to-end using the stochastic gradient descent based alternate optimization given $\mathcal{L}$. Apart from θ^E and θ_t^Ds, the updates for W_1 and W_2 are carried out considering the loss measures fixed but with a different learning rate than the one used for the encoder and decoders. During inference, the images are fed to the network and the respective segmentation or depth outputs are obtained.

Experimental Evaluations

In this section, we evaluate our network for semantic segmentation and depth estimation on CityScapes and NYUv2 datasets both quantitatively and qualitatively. In the following, we first mention the dataset descriptions and experimental protocols considered. A more detailed discussion on the results with critical analysis on some of the aspects are followed henceforth.

Datasets

CityScapes: The CityScapes dataset contains high-resolution street-view images for monocular segmentation and depth estimation. We consider two different splits with 7 and 19 semantic classes for evaluating our model. The images are re-sized to $[128, 256]$ prior to feeding to the network.

NYUv2: Similar to CityScapes, we have evaluated the performance of the model on NYUv2 dataset for the joint segmentation and depth estimation tasks. This dataset consists of RGB-D in door scene images from 13 semantic categories. This dataset is unarguably more complicated than CityScapes dataset mainly due variations in camera viewpoint, scene occlusion, differences in lighting conditions etc. The images are re-sized into $[128, 256]$. We followed the same evaluation procedure as MTAN (Liu, Johns, and Davison 2019) for both the datasets.

Architecture Design and Protocols

The proposed architecture has three dominant layers: the feature encoder, which is followed by a latent feature extraction block, also known as bottleneck and two decoders specific to the tasks. The encoder in total has eight conv. layers which are divided into four encoder blocks and each of these blocks contains two conv. layers sequentially. A given conv. layer has kernels of size 3×3 and is followed by a ReLu non-linearity and a batchnorm layer. These conv. layers are interconnected in similar fashion as the layers inside dense blocks of DenseNet (Huang et al. 2017) are connected e.g, input to any conv. layer is obtained by concatenating outputs of all previous conv. layers. By performing the same, at each conv. layer, the network always has access to all the previous information. Besides, each of the dense encoder blocks is succeeded by dropout and max-pooling with a receptive field of 2×2 and stride 2. The consecutive blocks in the encoder compute feature maps of depths 64, 128, 256 and 512. The bottleneck module contains two 3×3 conv. layers, each followed by a ReLu layer and a batchnorm layer and at the end, one transposed convolution layer is used for upsampling which produces feature maps of depth 1024. Essentially, this block provides the connection between the encoder and the decoder modules. In addition, the network contains two task-specific decoder modules. Similar to the encoder module, each of the decoders consist of four decoder blocks. There are two 3×3 conv. layers in each decoder block and each of these conv. layers is appended by a ReLu and a batchnorm layer. The input to each of these blocks is computed by concatenating feature maps received from corresponding encoder block and the immediate previous block in the decoder. Additionally, we have cross-task skip-connections from the second decoder block of task 1 to the third decoder block of task 2 and vice-versa. Finally, each decoder block is equipped with an attention module for learning task-specific features. By design, each attention module contains two 3×3 conv. layers, the first one having ReLu and the second one having sigmoid activations, respectively. Besides, batchnorm is considered for each of the conv. layers within the attention module. Output of the first conv. layer of a decoder is fed to the attention module to learn an attention mask. This attention mask is then pixel-wise multiplied ($\odot$) with the output of second conv. layer of the same decoder block. This attention output is forwarded to the next decoder block along with respective encoder block's feature maps. This process continues till the last block of the decoder.

The model is trained for 200 epochs using Adam optimizer (Kingma and Ba 2014) with an initial learning rate of $1e - 4$. We further consider a batch size of 8 and 2 for CityScapes and NYUv2 datasets, respectively. On the other hand, a learning rate of $1e - 3$ is considered to update the weights of task-specific loss terms in Equation 5.

Evaluation Metrics

To evaluate our network performance, we rely on four matrices: intersection over union (IoU) and mean intersection over union (mIoU) are the measures for semantic segmentation performance while absolute error and relative error are used for depth estimation.

Discussions

Comparison to the state-of-the-art: In this study, we design a densely encoded U-Net network (MT-UNet) for MTL and two single task U-Net models (ST-UNet). Evaluation on Cityscape dataset highlights that our multi-task model (MT-UNet) achieves 6.29% and 0.83% improvement in terms of the pixel accuracy (IoU) whereas 0.07% and 0.17% improvement in absolute error measures over single task attention network (STAN) (Liu, Johns, and Davison 2019) and ST-UNet respectively, while on NYUv2 dataset MT-UNet achieved 14.16% and 2.05% improvement in accuracy over STAN and ST-UNet, respectively. The sharp enhancement in the performance of the MTL model can be attributed cooperative learning framework between the tasks which helps in better modeling both the task's outcomes.

We also compared our model to previous state-of-the-art networks such as MTAN (Liu, Johns, and Davison 2019), cross-stitch (Misra et al. 2016) and dense (Liu, Johns, and Davison 2019) network variants. In all the cases, we observe that our model sharply outperforms all the stated networks for both the datasets. On CityScapes dataset, our model achieves 5.7%, 6.5%, and 5.9% higher pixel-level accuracy than MTAN, cross-stitch and dense networks respectively and 15.3%, 18% and 16% more pixel accuracy than the above respective networks on NYUv2 dataset. This gain in performance is obtained mainly because of three reason: i) the encoder in our model has densely connected layers which ensures better gradient flow. ii) our network uses hard parameter sharing so the encoder tries to find features in a

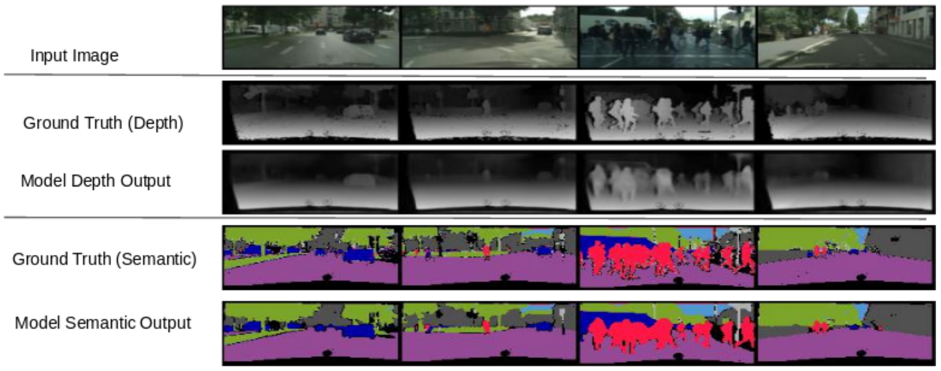

Figure 3: Qualitative results for semantic segmentation and depth estimation for CityScapes dataset (7 semantic classes).

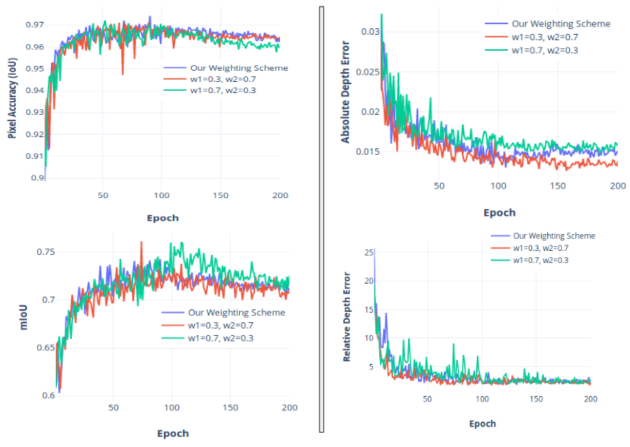

Figure 4: Network performance on different weight learning schemes for task-specific performance metrics for the CityScapes dataset. For fixed weights we used loss function $L = W_1\mathcal{L}_1 + W_2\mathcal{L}_2$ but for our weighing scheme we used Equation 5.

Method	Segmentation		Depth error	
	IoU	mIoU	Absolute	Relative
ST-UNet (Ours)	96.03%	71.76%	0.0154	3.609
STAN (Liu, Johns, and Davison 2019)	90.57%	51.90%	0.0145	27.46
Dense (Huang et al. 2017)	90.89%	51.91%	0.0138	27.21
Cross-Stitch (Misra et al. 2016)	90.33%	50.08%	0.0154	34.49
MTAN (Liu, Johns, and Davison 2019)	91.11%	53.04%	0.0144	33.63
MT-UNet with 7-classes (Semantic Segmentation)	**96.86%**	**73.19%**	**0.0137**	**2.7502**
MT-UNet with 19-classes (Semantic Segmentation)	**94.42%**	**34.53%**	**0.0141**	**2.8747**

Table 1: Multi-task validation result on CityScapes dataset both for semantic segmentation and depth estimation.

Method	Segmentation		Depth error	
	IoU	mIoU	Absolute	Relative
ST-UNet (Ours)	59.34%	23.81%	0.1152	0.2147
STAN (Liu, Johns, and Davison 2019)	47.23%	16.81%	0.1413	0.3502
Dense (Huang et al. 2017)	45.41%	15.42%	0.1464	0.3790
Cross-Stitch (Misra et al. 2016)	43.35%	13.71%	0.1590	0.4044
MTAN (Liu, Johns, and Davison 2019)	46.05%	16.05%	0.1425	0.3459
MT-UNet	**61.39%**	**26.54%**	**0.1304**	**0.1969**

Table 2: Multi-task validation result on NYUv2 dataset for 13-classes semantic segmentation and depth estimation.

space which is good for both tasks, this ensures better generalization. iii) the decoders in our model share features to each other and since these two tasks are similar to each other, this cross-talk improves their performance.

Method	Segmentation		Depth error	
	IoU	mIoU	Absolute	Relative
MT-UNet With Cross-talk	96.86%	73.19%	0.0137	2.7502
MT-UNet Without Cross-talk	96.71%	72.30%	0.0142	2.8921

Table 3: Model performance with and without cross-talk on CityScapes dataset with 7 semantic classes used for segmentation.

As described earlier, the performance is evaluated using four metrics. Table 1 and Table 2 shows the comparative results among the different methods mentioned. There is a significant decrease in absolute and relative depth errors. The absolute error and relative error relatively decrease by 4.5% and 65% respectively on NYUv2 dataset. It can be seen that the chosen tasks support each other while performing MTL which prove that we have outperformed all the baselines. To further check the task complexity and robustness of our model, we increased the number of classes to 19. In essence, it proves that as the class complexity increases mIoU decreases, as shown in Table 1.

Critical Analysis

- **Effect of cross-talk**: From Table 3, we want to show the effects of cross-talk and constant weights allotted to the losses on the model performance on CityScapes dataset. It shows how the model performance increases by considering the cross-talk between the task-specific decoders. The performance increases because sometimes the model is not able to compute the prominent features for the single tasks. Therefore, cross-talk helps the network to learn more effectively, i.e., the features calculated by the task corresponding to depth estimation supports the one corresponding to semantic segmentation and vice-versa. There is an increase in IoU and mIoU by 0.15% and 0.9% respectively, whereas absolute depth error and relative depth error reduce by 0.05% and 14% respectively.
- **Learnable weights vs constant weights**: We have also experimented our model with constant weights for the losses to check the performance between the tasks on CityScapes dataset. This setting is manually tuned, whereas our weight learning scheme is automatic. In the first case, we consider the loss $W_1\mathcal{L}_1 + W_2\mathcal{L}_2$ with weights, $W_1 = 0.3$ and $W_2 = 0.7$ where the depth estimation task is assigned lower priority in comparison to semantic segmentation. Whereas in the second case, i.e., with weights $W_1 = 0.7$ and $W_2 = 0.3$, the depth estimation task is given higher weightage. In this regard, it can easily be observed from Figure 4 that the depth error for the second case is less than the first scenario. On the other hand, it also signifies semantic segmentation performance is better in the first case. In addition, it can be inferred that the semantic segmentation contributes more information than depth estimation task, whatever be the weight configuration. As per our weighting scheme, the weights associated with the losses converge to nearly similar values (≈ 0.5 each) as the training converges. Hence, our weighting scheme's outputs lie between the two cases stated above. It proves that due to similarities between the tasks, our model seeks to provide equal weights to both the tasks. On a different note, we further compare our weight learning scheme with the dynamic average weighting (DWA) (Liu, Johns, and Davison 2019) (proposed in the MTAN paper) within the setup of our based model, where we find a performance gain of 5.4% with respect to the MTAN's segnet architecture. In, in turn, proves the efficacy of our proposed modified U-net based network design over the segnet architecture for MTL even with less network depth.
- **Visualization**: The depictions of the initial attention masks at the decoder end are shown in Figure 5 for images in the CityScapes data. Both the decoders are structurally same except they learn different attention masks for the different tasks which can easily be seen from the Figure 5. These variations in features signify the fact that both the task-specific attention masks focus on completely different aspects. Nevertheless, the attention modules help in providing important information towards producing better task outputs.

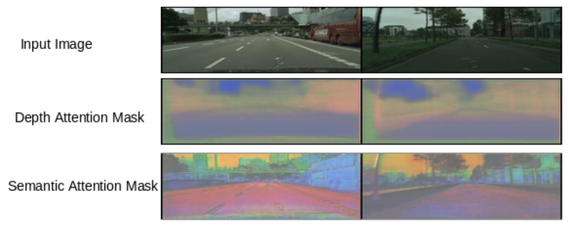

Figure 5: Semantic segmentation and depth estimation attention masks for the given input images (from top to bottom) on CityScapes dataset.

Conclusions

In this paper, we propose a novel U-Net based multi-task learning framework for joint segmentation and depth estimation from monocular images. Our model efficiently combines the task-generic and task-specific features through proper information sharing. While we follow a dense CNN model in the shared feature encoder part, separate attention-driven task-specific decoder modules perform the individual tasks. We further allow cross-talk between the tasks at the decoder end, apart from learning the weights for the task-specific loss functions. Our experimental results show sharp improvements on both the tasks on the challenging CityScapes and NYUv2 datasets in comparison to the previous state-of-the-art. Currently, we are engaged in scaling-up the model to several tasks and incorporating the notion of task clustering for better training.

References

Caruana, R. 1997. Multitask learning. *Machine learning* 28(1):41–75.

Chen, J.; Zhou, J.; and Ye, J. 2011. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 42–50. ACM.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3223.

Doersch, C., and Zisserman, A. 2017. Multi-task self-supervised visual learning. In *The IEEE International Conference on Computer Vision (ICCV)*.

Eigen, D., and Fergus, R. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, 2650–2658.

Evgeniou, T., and Pontil, M. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 109–117. ACM.

Harley, A. W.; Derpanis, K. G.; and Kokkinos, I. 2017. Segmentation-aware convolutional networks using local attention masks. In *Proceedings of the IEEE International Conference on Computer Vision*, 5038–5047.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Jacob, L.; Vert, J.-p.; and Bach, F. R. 2009. Clustered multi-task learning: A convex formulation. In *Advances in neural information processing systems*, 745–752.

Kendall, A.; Gal, Y.; and Cipolla, R. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7482–7491.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Kokkinos, I. 2017. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Kumar, A., and Daume III, H. 2012. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417.*

Liu, S.; Johns, E.; and Davison, A. J. 2019. End-to-end multi-task learning with attention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Misra, I.; Shrivastava, A.; Gupta, A.; and Hebert, M. 2016. Cross-stitch networks for multi-task learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Oktay, O.; Schlemper, J.; Folgoc, L. L.; Lee, M.; Heinrich, M.; Misawa, K.; Mori, K.; McDonagh, S.; Hammerla, N. Y.; Kainz, B.; et al. 2018. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999.*

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Silberman, N.; Hoiem, D.; Kohli, P.; and Fergus, R. 2012. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, 746–760. Springer.

Woo, S.; Park, J.; Lee, J.-Y.; and So Kweon, I. 2018. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19.

Yuan, C. 2011. Multi-task learning for bayesian matrix factorization. In *2011 IEEE 11th International Conference on Data Mining*, 924–931. IEEE.

Zhang, Y., and Yang, Q. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114.*

Zhu, J.; Chen, N.; and Xing, E. P. 2011. Infinite latent svm for classification and multi-task learning. In *Advances in neural information processing systems*, 1620–1628.