AuxiLight: Fast, Lightweight Auxiliary Loss Balancing Algorithm with Application in 6D Pose Estimation

Tianzong Zhang *tianzong.zhang@epfl.ch*
CVLAB, EPFL

Yinlin Hu *huyinlin@gmail.com*
CVLAB, EPFL; Magic Leap

Kaicheng Yu *kyu@westlake.edu.cn*
CVLAB, EPFL; Westlake University

Mathieu Salzmann *mathieu.salzmann@epfl.ch*
CVLAB, EPFL

Reviewed on OpenReview: *https://openreview.net/forum?id=iQGmKWqr4r*

Abstract

Jointly learning multiple tasks has proven to be beneficial. Auxiliary task learning builds on this by jointly training a set of predefined auxiliary tasks to improve the performance of a desired main task. Unfortunately, auxiliary task learning is often not practical because 1) jointly training multiple tasks requires an exponential hyperparameter search space of task loss weights; 2) the auxiliary tasks often lead to expensive annotation costs to obtain ground truth. In this work, we propose *AuxiLight*, a generic auxiliary-task loss-balancing algorithm, and use 6D pose estimation as a concrete real-world case study to demonstrate its practicality. AuxiLight addresses the first issue via an algorithm that adaptively balances the auxiliary task losses based on an analysis of the training dynamics. Experiments on standard multi-task datasets show that our method consistently improves over single-task models and achieves a favorable efficiency-performance tradeoff compared to state-of-the-art auxiliary task learning methods, being computationally efficient compared to more complex auxiliary learning methods. To demonstrate the practicality of auxiliary learning on real-world tasks, we further apply our method to 6D object pose estimation. We highlight that, for this task, multiple ground-truth auxiliary annotations can in fact be generated for free. This lets us showcase a concrete use of auxiliary learning for real-world problems that does induce annotation costs.

1 Introduction

In the field of deep learning, jointly learning multiple tasks has long been observed to improve the performance of the model compared to one trained with a single task. Such practice, known as multi-task learning, has been used in all kinds of deep learning applications, such as computer vision, natural language processing, and drug discovery (Standley et al., 2020; Zhang & Yang, 2021; Eftekhar et al., 2021). Indeed, when different tasks are jointly learned together, the knowledge of one task can be often learned by another task, and the generalization ability of the deep model is expected to be improved. In light of this, the notion of *auxiliary-task learning* has been recently proposed (Zhang et al., 2014; Navon et al., 2020; Liu et al., 2022) as a special case of multi-task learning. The auxiliary tasks aim to improve the performance of a desired single task, referred to as the "main task", by jointly training all tasks together. Unlike multi-task learning where the performances of all tasks are evaluated, in auxiliary-task learning only the performance of the main task is considered; the auxiliary tasks are solely used to improve the performance of the main task.

Unfortunately, auxiliary-task learning is often not practical in improving the performance of real-world tasks. In most cases, ground-truth annotations for auxiliary tasks often introduce expensive annotation costs, and the annotation cost of the auxiliary tasks might outweigh the performance gain they bring about. In addition, even in the case where auxiliary annotation is available, the contribution of different auxiliary tasks to the main task is often unknown. For example, when the main task is surface normal estimation, one could imagine that the task of depth estimation contributes more than object detection (Standley et al., 2020). As a result, different auxiliary task losses should have different weights when jointly trained with the main task. However, doing so requires an exponential hyperparameter search space of task loss weights, which makes tuning the network impossible when multiple auxiliary tasks are present. While many works have addressed this issue (Navon et al., 2020; Standley et al., 2020; Du et al., 2018), via multiple algorithms to balance auxiliary task losses, we will see in Sec. 4 that they either perform poorly, or are too computationally demanding to implement on any real-world tasks.

In this work we introduce *AuxiLight*, a generic, fast, lightweight auxiliary-task loss-balancing algorithm, and showcase its practicality with a real-world annotation cost-free use case on 6D pose estimation. Specifically, we address the aforementioned barriers in auxiliary-task learning as follows:

Weight balancing algorithm. To develop an algorithm that balances different auxiliary tasks when they are jointly trained with the main task, we make the assumption that the optimal weights of the auxiliary task losses should dynamically evolve with time to improve generalization during the training stage. As a result, we jointly optimize the weights of the auxiliary tasks throughout training, which results in a dynamical auxiliary task loss balancing algorithm that we refer to as *AuxiLight*. Such optimization on the auxiliary task weights is carried out by theoretically analyzing the training dynamics of the auxiliary tasks using the so-called technique of gradient flow approximation. Our results on several standard multi-task datasets show that AuxiLight improves over the single-task model and is computationally more efficient than more complex state-of-the-art auxiliary task balancing algorithms such as Auto-λ (Liu et al., 2022), in terms of speed and memory. In terms of accuracy, AuxiLight achieves competitive performance with the state-of-the-art algorithms and performs better in two of the three evaluated tasks.

Real-world auxiliary learning use case. To showcase a real-world use case of auxiliary task learning without hefty annotation costs, we apply our AuxiLight algorithm to the task of 6D object pose estimation, one of the key tasks in computer vision with many real-world applications, including virtual reality, autonomous driving, and robotics. Formally, given a 3D object model (e.g., a 3D mesh) and an RGB image captured by a calibrated camera, this task aims to predict the 3D orientation and 3D position of the object relative to the camera. We highlight that, while the annotations required for the additional tasks in multi-task learning or auxiliary task learning for scene understanding are expensive to obtain, in 6D object pose estimation, they come virtually for free, thanks to the availability of the 3D object model. Specifically, we investigate the use of four different auxiliary tasks: Depth estimation, surface normal prediction, principal curvature estimation, and image reconstruction. The first three exploit the 3D object model to obtain ground-truth annotations, and the latter relies solely on the input image. To the best of our knowledge, this constitutes the first annotation cost-free application that takes advantage of auxiliary tasks. We showcase the benefits and generality of AuxiLight in 6D object pose estimation using two state-of-the-art 6D pose estimation algorithms, WDR-Net (Hu et al., 2021) and ZebraPose (Su et al., 2022). Our approach outperforms the single-task models and achieves competitive performance with state-of-the-art auxiliary task learning algorithms, at a much lower computational cost. Note that the runtime during inference remains the same as with a single task because the auxiliary tasks are only needed during training. In other words, our approach improves the test-time accuracy of a 6D pose estimation network without requiring additional manual annotations and without affecting inference speed.

Our contributions can thus be summarized as follows:

- We introduce *AuxiLight*, a generic algorithm that adaptively balances the auxiliary task losses based on a theoretical analysis of the training dynamics, and that is computationally efficient compared to state-of-the-art task-weighting algorithms.
- While auxiliary task learning is often hard to deploy in real-world situations because of the cost of annotating the auxiliary ground truths, we showcase its use on the task of 6D pose estimation,

evidencing that automatically-generated auxiliary ground-truth annotations can boost performance on such an application. To our best knowledge, this is the first practical use case of auxiliary learning for a real-world application.

2 Related Work

2.1 Multi-task Learning

Multi-task learning aims at jointly learning multiple tasks, achieving provable information gain compared to learning single tasks separately when the tasks are related (Ben-David & Schuller, 2003). In the context of deep learning, multi-task learning is often carried out by constructing an encoder-decoder network where the encoder is shared across all tasks (Kokkinos, 2017). By jointly training different tasks, the network aims to learn a robust and universal representation, hence improving model generalization (Zhang & Yang, 2021). Many works have focused on the methodology of multi-task learning, and we refer the reader to (Standley et al., 2020) for a detailed discussion.

Auxiliary task learning. In context, our goal is not to achieve the best possible results in all tasks, but to improve only the main task. In the general multi-task learning literature, this is referred to as *auxiliary learning* (Navon et al., 2020; Du et al., 2018; Liu et al., 2022), which aims to exploit auxiliary tasks as implicit supervision to improve the performance of one single task. However, auxiliary learning does not come for free; training for the auxiliary tasks requires the corresponding ground truth, which, in general, involves an expensive annotation cost. For example, Zamir et al. (2018); Standley et al. (2020); Eftekhar et al. (2021) had to create new datasets with all tasks annotated to evaluate their proposed multi-task learning algorithm; in the context of facial landmark registration, Zhang et al. (2014) relied on manual annotations for auxiliary tasks such as gender recognition and predicting the presence of glasses. By contrast, in our constructed use case of 6D pose estimation, we leverage the fact that this specific task relies on the availability of a 3D object model, and propose auxiliary tasks whose ground truths can be automatically generated.

Adaptive task balancing. In the multi-task learning scenario, the task-specific losses are combined into an overall loss to supervise the network. The performance of the model is known to be sensitive to the weighting of each task-specific loss (Liu et al., 2022; Navon et al., 2020). Therefore, it is essential to carefully balance the loss of each task. The most naive way to do so is to perform a grid search on the task weights. However, the search space grows exponentially with the number of tasks, hence limiting the applicability of this strategy. Recent research has thus focused on automatically balancing the task weights (Liu et al., 2022; Navon et al., 2020; Du et al., 2018; Jiang et al., 2024; Mahapatra & Rajan, 2020; Lin et al., 2019; 2024). In particular, Du et al. (2018) heuristically use the cosine similarity of the task-specific loss gradients w.r.t. the shared encoder as task weights; Jiang et al. (2024) improve auxiliary task learning by mitigating a so-called negative transfer of auxiliary tasks; Auto-λ (Liu et al., 2022) explicitly formalizes task loss weighting as a bi-level optimization problem, the solution of which can be obtained by calculating the second-order gradients. As a result, although achieving state-of-the-art results, Auto-λ yields heavy computations and GPU memory usage, making the algorithm inapplicable to real-world scenarios, including 6D pose estimation. In this paper, we propose a fast, lightweight task loss balancing scheme that can be used for large models such as 6D pose estimation ones. Our method is derived from a theoretical analysis of the network dynamics and bears much less computational costs than Auto-λ with comparable results.

2.2 6D Object Pose Estimation

In our constructed annotation cost-free auxiliary-task learning use case, we focus on 6D pose estimation from a single RGB image. Below, we discuss the methods that address the same task, either by direct prediction, or by exploiting a PnP-related step.

Direct methods. Early attempts (Kendall et al., 2015; Kehl et al., 2017; Xiang et al., 2017) aimed at directly regressing the 6D pose by training an end-to-end encoder-decoder CNN network. However, it is commonly believed that CNNs are ill-suited to directly map the input image into rotation and translation (Rad & Lepetit, 2017; Hu et al., 2019), and as a result, most state-of-the-art methods rather follow a geometry-

inspired approach, predicting 3D-to-2D correspondences that can be exploited in PnP algorithm (Lepetit et al., 2009) to obtain the pose.

PnP-based sparse methods. A common strategy to obtain 3D-to-2D correspondences consists of predicting the image locations of a set of sparse object keypoints (Rad & Lepetit, 2017; Hu et al., 2019; Peng et al., 2019a). For example, BB8 (Rad & Lepetit, 2017) defines the keypoints as the eight 3D object bounding box corners. Hu et al. (2019) extend this idea by predicting the 2D keypoint locations for every pixel in the object mask. PVNet (Peng et al., 2019a) predicts the direction from the object pixels to the 2D keypoints to establish the correspondences. However, as these methods only exploit a few keypoints, they tend to neglect the full geometry of the object model.

PnP-based dense methods. As a consequence, the recent state-of-the-art methods predict dense 3D-to-2D correspondences instead of the sparse ones (Di et al., 2021; Li et al., 2019; Su et al., 2022). This is achieved by predicting the corresponding 3D coordinate on the model surface for each object pixel in the input image. The state-of-the-art ZebraPose (Su et al., 2022) encodes the object mesh vertices into binary codes, turning 3D-to-2D correspondence prediction into a classification task instead of a 3D coordinate regression task.

Following this trend, we argue that exploiting the object model is important to improve the 6D pose estimation algorithms. In our work, however, we further exploit the object model by rendering several geometrical features in addition to the 3D-to-2D correspondence. Leveraging the idea of auxiliary learning discussed in Section 2.1, we use these model-induced features to implicitly supervise the network in an auxiliary task learning fashion.

3 AuxiLight: Balancing Auxiliary Tasks

Let us now introduce *AuxiLight*, our generic auxiliary task loss balancing algorithm. In Sec. 5, we will see a concrete use case of AuxiLight algorithm.

Preliminaries Consider a general framework of auxiliary learning, where we have one main task $\mathcal{T}_0$ and a bunch of auxiliary tasks $\mathcal{T}_1, ..., \mathcal{T}_n$. The neural network is a multi-task hard parameter sharing network with a shared encoder $\boldsymbol{\theta}_{\text{enc}}$ and different heads $\boldsymbol{\theta}_0; \boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_n$, respectively for the main task and each auxiliary task. This is formalized as

$$\hat{y}_i = f_i(x; \boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_i), \quad i = 0, 1, ..., n,$$

where x is the ground truth input and $\hat{y}_i$ is the predicted output for task $\mathcal{T}_i$. Additionally, we denote y as the ground truth target. With the prediction and the ground target, each task $\mathcal{T}_i$ is associated with a loss, which is a function of the encoder and the corresponding head:

$$L_i(y_i, \hat{y}_i) = L_i(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_i).$$

3.1 Warm-up: Single-task learning

To better introduce the mechanism of auxiliary learning, we first take a look at the standard single-task learning (STL), where only the main task presents with no auxiliary task. Given training set $\{(x^{(k)}, y^{(k)})\}_{k=1,\cdots,K}$ sampled from an unknown (real) data distribution D, STL aims to minimize the expected loss

$$\bar{L}_0 = \mathbb{E}_{(x,y)\sim D} L_0(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_0). \tag{1}$$

As this expected loss is unknown, the population loss on the training set is minimized instead. Practically, people run SGD algorithm on a mini-batch loss, which follows the following formula:

$$\begin{aligned} \boldsymbol{\theta}_{\text{enc}} &\leftarrow \boldsymbol{\theta}_{\text{enc}} - \alpha \cdot \frac{\partial L_0(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_0)}{\partial \boldsymbol{\theta}_{\text{enc}}}, \\ \boldsymbol{\theta}_0 &\leftarrow \boldsymbol{\theta}_0 - \alpha \cdot \frac{\partial L_0(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_0)}{\partial \boldsymbol{\theta}_0}. \end{aligned} \tag{2}$$

While SGD is hard to analyze theoretically, many recent papers have been investigating on gradient flow instead (Mei et al., 2019). Gradient flow is the continuous-time analogue to the discrete-time gradient step in Eq. 2:

$$\begin{aligned} \frac{d\boldsymbol{\theta}_{\text{enc}}(t)}{dt} &= -\alpha \cdot \frac{\partial L_0(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_0)}{\partial \boldsymbol{\theta}_{\text{enc}}}, \\ \frac{d\boldsymbol{\theta}_0(t)}{dt} &= -\alpha \cdot \frac{\partial L_0(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_0)}{\partial \boldsymbol{\theta}_0}. \end{aligned} \tag{3}$$

It has been shown that, with small learning rate α, the gradient flow is a good approximation to the gradient step (Nesterov, 2003; Santambrogio, 2017; Scheithauer). With the same technique, our proposed auxiliary learning algorithm will be based on the gradient flow of a combined loss of the main task and the auxiliary tasks.

3.2 AuxiLight: Auxiliary learning

Same as STL, the main goal of auxiliary task is to minimize the expected loss of the main task (Eq. 1), but instead, auxiliary learning minimizes a surrogate multi-task loss

$$L = L_0(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_0) + \sum_{i \geq 1} \lambda_i L_i(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_i),$$

where λ_i is the weight of the auxiliary task $\mathcal{T}_i$. Our proposed AuxiLight algorithm relies on the assumption that λ_i should evolve over time to improve generalization, i.e., $\lambda_i = \lambda_i(t)$ is a function of time. As a result, we solve for the best λ by minimizing the expected main loss $\bar{L}_0$ as defined in Eq. 1. This is done by iteratively updating λ_i, with a gradient flow step as follows

$$\frac{d\lambda_i}{dt} = -\beta \cdot \frac{\partial \bar{L}_0}{\partial \lambda_i},$$

where β is the learning rate for updating λ. Similarly to Eq. 3 in STL, the training dynamics of the auxiliary-task network parameter is [1]

$$\begin{aligned} \frac{d\boldsymbol{\theta}_{\text{enc}}}{dt} &= -\alpha \cdot \frac{\partial L}{\partial \boldsymbol{\theta}_{\text{enc}}} = -\alpha \sum_{i \geq 0} \lambda_i \cdot \frac{\partial L_i(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_i)}{\partial \boldsymbol{\theta}_{\text{enc}}}, \\ \frac{d\boldsymbol{\theta}_i}{dt} &= -\alpha \cdot \frac{\partial L}{\partial \boldsymbol{\theta}_i} = -\alpha \lambda_i \cdot \frac{\partial L_i(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_i)}{\partial \boldsymbol{\theta}_i}. \end{aligned}$$

Therefore, by using chain rule, the dynamics of the task weighting can be calculated as follows: For $i \geq 1$,

$$\begin{aligned} \frac{d\lambda_i(t)}{dt} &= -\beta \cdot \frac{\partial \bar{L}_0}{\partial \lambda_i(t)} = -\beta \cdot \frac{\partial \bar{L}_0(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_0)}{\partial \lambda_i(t)} \\ &= -\beta \left(\left(\frac{\partial \boldsymbol{\theta}_{\text{enc}}}{\partial \lambda_i(t)} \right)^\top \frac{\partial \bar{L}_0}{\partial \boldsymbol{\theta}_{\text{enc}}} + \left(\frac{\partial \boldsymbol{\theta}_0}{\partial \lambda_i(t)} \right)^\top \frac{\partial \bar{L}_0}{\partial \boldsymbol{\theta}_0} \right). \end{aligned}$$

It is worth noting that

$$\begin{aligned} \frac{\partial}{\partial \lambda_i(t)} \boldsymbol{\theta}_{\text{enc}} &= \frac{\partial}{\partial \lambda_i(t)} \left(\boldsymbol{\theta}_{\text{enc}}^{(t=0)} + \int_{\tau=0}^{t} \frac{d\boldsymbol{\theta}_{\text{enc}}}{dt} d\tau \right) \\ &= -\alpha \frac{\partial}{\partial \lambda_i(t)} \int_{\tau=0}^{t} \sum_{i \geq 0} \lambda_i(\tau) \cdot \frac{\partial L_i(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_i)}{\partial \boldsymbol{\theta}_{\text{enc}}} d\tau \\ &= -\alpha \frac{\partial L_i(\boldsymbol{\theta}_{\text{enc}}, \boldsymbol{\theta}_i)}{\partial \boldsymbol{\theta}_{\text{enc}}} = -\alpha \frac{\partial L_i}{\partial \boldsymbol{\theta}_{\text{enc}}}, \end{aligned}$$

[1] For simplicity, we define the constant $\lambda_0 \equiv 1$.

where the derivative calculation in the last step takes into consideration the fact that the $\lambda(\tau)$s are independent of the $\lambda(t)$s for all $\tau < t$. Following a similar calculation, we have $\partial \boldsymbol{\theta}_0 / \partial \lambda_i = \mathbf{0}$. Therefore, the weights of the auxiliary tasks follow the dynamics

$$\begin{aligned} \frac{d\lambda_i(t)}{dt} &= -\beta \left(\left(\frac{\partial \boldsymbol{\theta}_{\text{enc}}}{\partial \lambda_i} \right)^\top \frac{\partial \bar{L}_0}{\partial \boldsymbol{\theta}_{\text{enc}}} + \left(\frac{\partial \boldsymbol{\theta}_0}{\partial \lambda_i} \right)^\top \frac{\partial \bar{L}_0}{\partial \boldsymbol{\theta}_0} \right) \\ &= \alpha\beta \cdot \left(\frac{\partial L_i}{\partial \boldsymbol{\theta}_{\text{enc}}} \right)^\top \frac{\partial \bar{L}_0}{\partial \boldsymbol{\theta}_{\text{enc}}}. \end{aligned} \tag{4}$$

In practice, the expected loss $\bar{L}_0$ is impossible to calculate. Following a similar argument to that of Liu et al. (2022), we use the current training batch instead to evaluate to gradient of the expected loss. By discretizing the gradient flow update of λ_i, we obtain the update

$$\lambda_i \leftarrow \lambda_i + \alpha\beta \cdot \left(\frac{\partial L_0}{\partial \boldsymbol{\theta}_{\text{enc}}} \right)^\top \frac{\partial L_i}{\partial \boldsymbol{\theta}_{\text{enc}}}. \tag{5}$$

The overall algorithm of our proposed AuxiLight is summarized in Algorithm 1. Interestingly, the update rule comes with an intuitive explanation. For each step, the task loss weight is increased by the inner product of the shared encoder gradient of the main task and the auxiliary task, which measures the similarity or relatedness between the auxiliary task and the main task. The auxiliary weight stops increasing at the later training stage when the two gradients become orthogonal, which means that all knowledge from the auxiliary task has been learned.

Algorithm 1 AuxiLight: Balancing Auxiliary Tasks

Input: Main task $\mathcal{T}_0$, auxiliary tasks $\{\mathcal{T}_i\}_{i=1}^n$
Input: Corresponding loss functions L_i, $i \geq 0$
Output: Trained network of main task $\boldsymbol{\theta}_{\text{enc}}^*, \boldsymbol{\theta}_0^*$

```
initialize learning rates α, β
initialize model parameters θ = {θ_enc, {θ_i}_{i=0}^n}
initialize auxiliary task weights λ = [λ_1, ··· , λ_n]
while NETWORK NOT CONVERGED do
    L ← L_0 + Σ_{i=1}^n λ_i · L_i
    θ ← θ − α · ∇_θ L
    for i = 1, ··· , n do
        λ_i ← λ_i + αβ · ∇_{θ_enc} L_0^⊤ ∇_{θ_enc} L_i
    end for
end while
return θ_enc, θ_0
```

In Sec. 4, we will see that AuxiLight is able to improve the performance of the single main task model. Compared to existing auxiliary loss balancing algorithms (Du et al., 2018; Liu et al., 2022), our algorithm requires the least runtime and GPU memory while achieving a comparable accuracy.

4 Experiments

4.1 AuxiLight vs Other Task Balancing Schemes

We evaluate our AuxiLight algorithm on the NYUv2 dataset (Silberman et al., 2012) and the CityScapes dataset (Cordts et al., 2016), which are standard datasets widely used to evaluate multi-task algorithms. Specifically, for the NYUv2 dataset, we train for three tasks: semantic segmentation, depth estimation and surface normal prediction; while on the CityScapes dataset, we train for semantic segmentation, disparity estimation and part segmentation. Following existing auxiliary weight balancing algorithms (Du et al., 2018; Liu et al., 2022), we carry out three experiments where one task is taken as the main task and the other

Algorithm	GPU Memory	# FW Pass	# BW Pass
No Auxiliary	$2M_{\text{enc}} + 2M_{\text{dec}}$	1	1
Equal weight	$2M_{\text{enc}} + 2(\lvert\mathcal{S}\rvert + 1) \cdot M_{\text{dec}}$	1	1
GradSim	$3M_{\text{enc}} + 2(\lvert\mathcal{S}\rvert + 1) \cdot M_{\text{dec}}$	1	1
Auto-λ	$5M_{\text{enc}} + 5(\lvert\mathcal{S}\rvert + 1) \cdot M_{\text{dec}}$	2	3
ForkMerge	$3M_{\text{enc}} + 5(\lvert\mathcal{S}\rvert + 1) \cdot M_{\text{dec}}$	≥ 2	≥ 2
AuxiLight	$3M_{\text{enc}} + 2(\lvert\mathcal{S}\rvert + 1) \cdot M_{\text{dec}}$	1	1

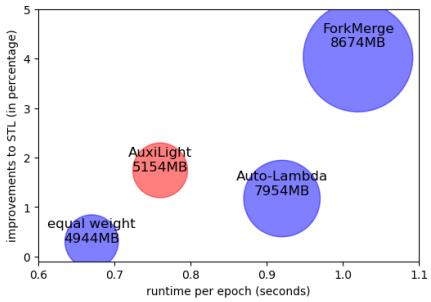

Table 1: **Memory & Runtime comparison between AuxiLight and other auxiliary task loss balancing algorithms.** In the left table, we perform a theoretical analysis of the memory consumption and run time, where M_{enc} denotes the memory for storing the shared encoder weights, M_{dec} the memory for storing the decoder for each auxiliary task or the main task (assuming the same size), and $\mathcal{S}$ is the set of auxiliary tasks. # FW Pass and # BW Pass refer to the number of forward and backward passes needed with pyTorch implementation. For ForkMerge, the number of FW and BW passes is not constant and depends on a chosen hyperparameter in the implementation. In the right figure, we compare the existing auxiliary task algorithms on NYUv2 dataset by plotting the actual memory and runtime usage, as well as the accuracy. Specifically, the x axis is the runtime per epoch, the y axis is the accuracy improvement, and the disk size shows the GPU memory needed for the auxiliary learning algorithm.

	(a) NYUv2 dataset: Performance of main task ...				(b) CityScapes dataset: Performance of main task ...			
Algorithm	Sem. Seg. mIOU (↑)	Depth aErr. (↓)	Normal mDist.(↓)	Δ_{aux} (↑)	Sem. Seg. mIOU (↑)	Pt. Seg. mIOU (↑)	Disp. mDist.(↓)	Δ_{aux} (↑)
No Auxiliary	51.42	42.74	22.82	-	56.20	52.74	0.84	-
Equal weight	52.13	39.03	24.14	+0.3%	55.72	52.62	0.83	+0.0%
GradSim	52.25	39.31	23.86	+0.6%	55.76	52.19	0.80	+1.0%
Auto-λ	52.40	39.25	23.38	+1.2%	57.89	53.56	**0.77**	**+4.3%**
ForkMerge	**53.67**	**38.91**	**22.18**	**+4.0%**	**58.64**	52.93	0.80	+3.1%
AuxiLight	52.73	39.32	23.98	+1.7%	58.50	**53.58**	0.80	+3.5%

Table 2: **Performance comparison between AuxiLight and other auxiliary task loss balancing algorithms.** We show the performance of the algorithms on each corresponding main task of (a) NYU-v2 and (b) CityScapes. In each experiment, the two tasks other than the main task are used as auxiliary tasks.

two are the auxiliary tasks. We use the same multi-task learning framework with hard parameter sharing discussed in Section 5.2 as network architecture. In addition to NYUv2 and CityScapes, we also evaluate our methods on the Office-31 dataset Hoffman et al. (2012) and on the DomainNet dataset Peng et al. (2019b), where instead of evaluating multiple tasks, multiple domains are evaluated. For every experiment, one domain is selected as main task while the other domains are used for auxiliary training.

Furthermore, we compare our proposed AuxiLight with a naive baseline consisting of using equal weights, as well as with GradSim (Du et al., 2018), Auto-λ (Liu et al., 2022) and Forkmerge Jiang et al. (2024). A summary of the comparison on NYUv2 is shown in the right figure of Table 1. We highlight that our proposed AuxiLight is a faster and more lightweight alternative to the state-of-the-art algorithms with comparable performance, which is crucial for real-world tasks (see Sec 5.5).

We note that, unlike ForkMerge (Jiang et al., 2024), which explicitly detects and mitigates negative transfer from harmful auxiliary tasks, AuxiLight relies solely on a gradient-alignment signal and does not explicitly guard against negative transfer; a harmful auxiliary task can therefore still receive a non-negligible weight. We discuss and illustrate this limitation in Appendix A.3.

	(a) DomainNet dataset: Performance of main domain ...							(b) Office-31 Dataset: Performance of main domain ...			
Algorithm	C (↑)	I (↑)	P (↑)	Q (↑)	R (↑)	S (↑)	Δ_{aux} (↑)	A (↑)	D (↑)	W (↑)	Δ_{aux} (↑)
No Auxiliary	77.6	41.4	71.8	**73.0**	84.6	70.2	-	86.4	98.0	95.7	-
Equal weight	78.0	38.1	67.2	50.8	77.1	63.0	-9.6%	87.2	98.4	99.4	+1.7%
GradSim	76.4	38.5	67.8	52.4	77.9	67.6	-9.0%	86.9	98.2	**98.9**	+1.3%
Auto-λ	78.3	37.8	70.2	56.3	79.7	67.1	-7.2%	88.4	**98.4**	**98.9**	+1.7%
ForkMerge	**79.9**	**42.7**	**73.5**	**73.0**	**85.2**	**72.0**	**+2.0%**	**88.9**	98.2	**98.9**	**+2.1%**
AuxiLight	78.9	42.5	72.9	55.3	84.7	68.2	-3.5%	88.7	98.2	**98.9**	**+2.1%**

Table 3: **Additional performance comparisons between AuxiLight and other auxiliary task loss balancing algorithms.** In addition to the experiments shown in Table 2, we also compare our results for the (a) DomainNet and (b) Office-31 dataset. Each dataset contains multiple domains, and for every experiment, one domain is selected as main task while the other domains are used for auxiliary training.

Memory & Runtime. In Table 1, we show a memory and run-time comparison of all auxiliary task loss balancing algorithms. M_{enc} denotes the memory for storing the shared encoder weights, M_{dec} the memory for storing the decoder for each auxiliary task or the main task (without loss of generality, we assume that all decoders have the same size), and $\mathcal{S}$ is the set of auxiliary tasks. Note that in most neural network implementations (*e.g.*, pyTorch), storing the gradients requires an equal amount of memory as storing the parameters. To our surprise, Auto-λ and Forkmerge require significantly more memory and computations than our proposed AuxiLight and other task loss balancing schemes. This is due to the fact that Auto-λ involves calculating a second-order hessian of the weights, whose approximation requires at least twice as much memory and number of forward / backward passes. In fact, for our specific application of 6D pose estimation, both algorithms are not able to run on a single GPU, which justifiably shows the drawback of the state-of-the-art auxiliary learning algorithms (see Sec 3).

Concretely, we compare the memory of our proposed AuxiLight and other auxiliary weight-balancing algorithms on the NYUv2 dataset: Equal weight uses 4944 MB memory, Auto-λ uses 7948 MB memory, and our proposed AuxiLight uses 5154 MB memory. As per runtime, Equal weight takes 0.67s per step, Auto-λ takes 0.76s per step, our method takes 0.92s per step, while ForkMerge takes 1.02s per step. A summary of the comparison can be found in the left figure of Table 1.

Accuracy. In Table 2 and Table 3, we report the performance for the main task when the other tasks are jointly trained as auxiliary task. The evaluation metrics are identical with those of Liu et al. (2022) and the details are elaborated in the appendix. Additionally, we calculate the average improvement across all the tasks compared to the performance of training single tasks, denoted as Δ_{aux}. As shown by the results, our method ranks second in most of the experiments; however, such suboptimality is compensated by the fact that our method requires less computational resources than the state-of-the-art algorithms. More importantly, AuxiLight consistently outperforms GradSim and equal weight algorithms, sometimes by a large margin, making our method the best option when the computational resources are limited.

5 Annotation Cost-free Auxiliary Learning: A Case Study on 6D Pose Estimation

As discussed in Sec. 1, one of the barriers to practising auxiliary learning lies in the expensive annotation costs of the auxiliary tasks. In this section, however, we discuss a use case of auxiliary learning where annotations for auxiliary tasks can be obtained for free, with the main task being object 6D pose estimation. Our proposed auxiliary 6D pose estimation framework builds on any existing PnP-based 6D pose estimation algorithm (Hu et al., 2019; Su et al., 2022; Hu et al., 2021; Li et al., 2019; Di et al., 2021), and is empirically shown to be able to improve the performance of these existing methods with little annotation cost.

Below, we first briefly present our multi-task model, then introduce the four auxiliary tasks we will study in our experiments, and finally show how auxiliary learning can be used to improve existing 6D pose estimation methods.

5.1 Main-task Model

The task of object 6D pose estimation is formally defined as follows. Given an object 3D model, represented by its n surface vertices $\{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^n$, object 6D pose estimation aims at predicting the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ and the translation vector $\mathbf{t} \in \mathbb{R}^3$ (i.e. 6D pose) of the object from an input RGB image $\mathbf{RGB} \in \mathbb{R}^{3\times H\times W}$. Additionally, the camera has a known intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$, and a 3D point $\mathbf{p} = [x, y, z]^\top$ on the 3D model surface and its projected pixel $\mathbf{u} = [u, v]^\top$ in the image space should follow the perspective projection equation:

$$c[\mathbf{u}] \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \left(\mathbf{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{t} \right), \tag{6}$$

where $c[\mathbf{u}]$ is the depth at pixel $\mathbf{u}$.

As discussed in Sec. 2.2, instead of directly predicting the 6D pose $(\mathbf{R}, \mathbf{t})$, most existing methods predict the 3D-to-2D correspondence $\{\mathbf{p} \leftrightarrow \mathbf{u}\}$, followed by a P$n$P algorithm that solves 6D pose from the 3D-to-2D correspondence. Practically, such correspondence is predicted by regressing $\mathbf{u}$ for a fixed set of $\mathbf{p}$ for sparse methods, or by regressing $\mathbf{p}$ for every $\mathbf{u}$ visible in the image for dense methods. In any case, predicting 3D-to-2D correspondence is carried out by an encoder-decoder network and all existing methods can by summarized and formalized as follows:

$$\mathbf{pred}_{\text{3D}-\text{2D}}, \mathbf{pred}_{\text{mask}} = f_{\text{dec}}(\mathbf{repr}; \boldsymbol{\theta}_{\text{dec}}),$$

where

$$\mathbf{repr} = f_{\text{enc}}(\mathbf{RGB}; \boldsymbol{\theta}_{\text{enc}})$$

is the latent representation extracted by the encoder f_{enc} with parameters θ_{enc}, and processed by the decoder f_{dec} with parameters θ_{dec} to obtain the 3D-to-2D correspondences. Normally an object mask $\mathbf{pred}_{\text{mask}}$ is predicted jointly for loss evaluation.

5.2 Multi-task Architecture

We first describe our multi-task model for auxiliary-task object 6D pose estimation. Our model is based on any existing 6D pose estimation algorithms that follow encoder-decoder architecture, as depicted in Sec 5.1. To jointly learn auxiliary tasks and improve prediction accuracy, we adopt a hard parameter sharing multi-task learning formalism (Kokkinos, 2017), where the latent representation **repr** is shared across all auxiliary tasks, and, for each task, a task-specific decoder is used to map the shared representation to the prediction. In other words, for the i-th task, we can write

$$\mathbf{pred}_{\text{task}_i} = f_{\text{dec}}(\mathbf{repr}; \boldsymbol{\theta}_{\text{task}-\text{dec}_i}) \ ,$$

where $\boldsymbol{\theta}_{\text{task}-\text{dec}_i}$ denotes the parameters of the decoder for the i-th task.

5.3 Proposed Auxiliary Tasks

Let us now define the auxiliary tasks that we will study in our experiments. As discussed in Sec. 1, the auxiliary tasks, in general, should (i) introduce minimum annotation costs with no manual labeling; (ii) contain shared knowledge with the main task that can be used to improve generalization. We highlight that, fortunately, for 6D pose estimation, many tasks meet the aforementioned requirements. Specifically, these tasks include predicting depth, surface normal, and principal curvature.

The details of the proposed auxiliary tasks are explained in the following paragraphs. Due to the fact that both ground-truth object 3D model and 6D pose are available, the scene can be perfectly reconstructed by using the perspective projection function Eq. 6. As a result, the ground truths of the auxiliary tasks can be generated effortlessly.

Depth. Depth plays an important role in 6D pose estimation, as it is directly related to the translation component of the 6D pose. This has been exploited by RGB-D 6D pose estimation methods, which take measured depth (He et al., 2020; Wang et al., 2019; Li et al., 2018) as input. Here, however, we do not take depth as input, but leverage the fact that it shares information with the 6D pose to exploit depth prediction as an auxiliary task. Formally, we can thus define the ground truth for our *depth task* as

$$\mathrm{gt}_{\mathrm{depth}}[\mathbf{u}] = c[\mathbf{u}]. \tag{7}$$

While this information is valuable when predicting 6D pose from the entire input image, it is ill-suited in the case where the object is first detected and cropped prior to pose estimation (Li et al., 2019; Su et al., 2022; Di et al., 2021), as absolute depth is hard to infer from a crop. To handle this scenario, for each pixel in the object mask, we calculate the distance in depth to the pixel with the smallest depth in the mask. This lets us define the ground truth for our *differential depth task* as

$$\mathrm{gt}_{\Delta\mathrm{depth}}[\mathbf{u}] = c[\mathbf{u}] - \min_{\mathbf{M}[\mathbf{u}']=1} c[\mathbf{u}'], \tag{8}$$

where $\mathbf{M}$ is the 0-1 object mask, with $\mathbf{M}[\mathbf{u}'] = 1$ if and only if pixel $\mathbf{u}'$ belongs to the object. Differential depth is invariant under cropping or zoom-in transformations.

Surface normal. Surface normal characterizes the first-order derivative of the object 3D shape, and is crucial for scene understanding. Multiple works have identified surface normal as a beneficial auxiliary task (Standley et al., 2020; Liu et al., 2022). Given the object 3D model, we can offline calculate the unit surface normal $\mathbf{n}$, ($\|\mathbf{n}\|_2 = 1$) in the object space for every point $\mathbf{p}$ on the object surface. The surface normal is then rotated to the camera space according to the object pose to make the task predictable from a standalone image. This ground-truth information for the *surface normal task* can thus be calculated as

$$\mathrm{gt}_{\mathrm{normal}}[\mathbf{u}] = \mathbf{R} \cdot \mathbf{n}[\mathbf{p}]. \tag{9}$$

Principal curvature. Principal curvature captures the second-order properties of the object 3D shape. It is an excellent candidate for an auxiliary task because curvature is an intrinsic 3D shape property which is invariant under rotation and translation. When the 3D object model is provided as a triangulated mesh, the principal curvatures $\mathbf{k}_1, \mathbf{k}_2$ can be calculated using a discrete curvature operator (Cohen-Steiner & Morvan, 2003; Meyer et al., 2003). We denote the ground truth for the *principal curvature task* as

$$\mathrm{gt}_{\mathrm{curvature}}[\mathbf{u}] = \big[\mathbf{k}_1[\mathbf{p}], \mathbf{k}_2[\mathbf{p}]\big]. \tag{10}$$

Image reconstruction. Finally, we also include image reconstruction as an additional auxiliary task. This is also referred to as auto-encoding (Hinton & Salakhutdinov, 2006), with the encoder-decoder network being trained to produce the input image as output. While reconstruction does not capture the properties of the 3D object model, it can help to find a low-dimensional, robust-to-noise representation for the input image, potentially improving model generalization. The *reconstruction task* is thus formalized as

$$\mathrm{gt}_{\mathrm{reconst.}}[\mathbf{u}] = \mathbf{RGB}[\mathbf{u}], \tag{11}$$

where $\mathbf{RGB}[\mathbf{u}]$ is the RGB vector of the input image at pixel $\mathbf{u}$.

5.4 Auxiliary Task Training

To exploit the object 3D geometry and improve generalization from the auxiliary tasks, we train the multi-task model under the supervision of the auxiliary tasks defined in Eq. 7 - Eq. 11. Formally, we define

the set of auxiliary tasks $\mathcal{S}$ as $\mathcal{S} = \{\text{depth}, \text{normal}, \text{curvature}, \text{reconst.}\}$ when the network input is the full image, and $\mathcal{S} = \{\Delta\text{depth}, \text{normal}, \text{curvature}, \text{reconst.}\}$ when the image is first cropped using a detector. Following Standley et al. (2020), we adopt the ℓ_1 loss to supervise each task. Moreover, due to the fact that all tasks are only defined on the pixels of the visible part of the target object, the corresponding losses are confined to the object foreground as well. As a result, the loss for each $\text{task}_i \in \mathcal{S}$ can be formalized as

$$\mathcal{L}_{\text{task}_i} = \|\mathbf{M} \odot \mathbf{gt}_{\text{task}_i} - \mathbf{M} \odot \mathbf{pred}_{\text{task}_i}\|_1, \tag{12}$$

where $\mathbf{M} = \mathbf{gt}_{\text{mask}}$ is the 0-1 mask of the object, and $\mathbf{gt}$, $\mathbf{pred}$ are the ground truth and prediction of the corresponding task defined in Sec. 5.3.

With all the auxiliary tasks and their losses defined, now we can follow the AuxiLight algorithm in Sec. 3 to train the network. The overall training loss is defined as a linear combination of main loss and task-specific auxiliary losses, i.e.,

$$\mathcal{L} = \mathcal{L}_{\text{main}} + \sum_{i=1}^{|\mathcal{S}|} \lambda_{\text{task}_i} \cdot \mathcal{L}_{\text{task}_i}, \tag{13}$$

where λ_{task_i} is the weight of the i-th auxiliary task $\text{task}_i \in \mathcal{S}$, and $\mathcal{L}_{\text{main}}$ is the original loss of the 6D pose estimation method, i.e., typically a loss combining the 3D-to-2D correspondence error and the segmentation error for predicting the object mask. The AuxiLight algorithm is then followed to train the auxiliary tasks and the main task jointly. We comment that, during test time, all auxiliary tasks are discarded, and the inference speed thus remains the same as the base model trained with the single main task.

5.5 Experiment on Auxiliary-task 6D Pose Estimation

We now introduce our experiments on auxiliary-task 6D pose estimation discussed in Sec. 5. As discussed in Sec 1, such experiments showcase a real-world use-case of the auxiliary task algorithm, where no additional labeling is required manually.

Base models. Our proposed multi-task 6D pose estimation framework can be applied to any existing PnP-based method that regresses the 6D pose from 3D-to-2D correspondences. To showcase this, following the taxonomy in Section 2.2, we choose one state-of-the-art 6D pose estimation method from each category, i.e., one PnP-based sparse method and one PnP-based dense method. Specifically, we use WDR-Net (Hu et al., 2021) and ZebraPose (Su et al., 2022) as base models in our experiments. WDR-Net establishes sparse 3D-to-2D correspondences by predicting 8 projected 2D keypoint coordinates using an FPN-based network. By contrast, ZebraPose predicts binary codes encoding 3D coordinates on the object surface to obtain dense 3D-to-2D correspondences, using a decoder adapted from Deeplabv3 (Chen et al., 2017). In both methods, a semantic object mask is jointly predicted to confine the 3D-to-2D correspondences within the object foreground.

Multi-task framework. To exploit our multi-task framework to improve existing 6D pose estimation algorithms, we first render the ground truths of the auxiliary tasks defined in Section 5.3. Given the object 3D mesh and the ground-truth 6D pose, the scene is rendered using OSMesa, an off-screen renderer, and the ground truths of task $\{\text{depth}, \Delta\text{depth}, \text{normal}, \text{curvature}\}$ are consequently calculated according to Eq. 7 - Eq. 10. Note that generating the auxiliary ground truths is performed offline, automatically, and requires no expensive annotation costs.

The multi-task architecture follows Sec. 5.2. We use the same encoder as the base model and, for each auxiliary task, a task-specific decoder with the same decoder architecture as the base model, except for the last convolutional layer which is modified to match the number of channels of the target tasks.

In our experiments, we will compare three task balancing algorithms to optimize the multi-task model: 1) equal weights, 2) GradSim and 3) our proposed AuxiLight (Sec. 3). We excluded both Auto-λ and Forkmerge from this set of experiments because they induce too large memory and computation overhead for the task of 6D pose estimation, as discussed in Section 4.1.

	(a) WDR-Net				(b) ZebraPose			
Class	BM	+EQ	+GS	+AL	BM	+EQ	+GS	+AL
ape	33.8	35.0	39.4	35.2	58.2	59.4	59.6	59.2
can	58.0	60.5	59.6	60.2	96.0	96.3	96.4	96.3
cat	48.7	50.3	47.6	50.7	61.0	60.0	60.7	60.8
drill.	63.9	65.5	65.2	65.0	95.0	94.5	93.9	93.7
duck	38.1	39.2	39.6	40.0	58.8	62.2	63.7	62.9
egg.	71.0	71.3	72.0	71.8	70.7	71.9	72.5	72.7
glue	67.5	68.2	67.3	67.0	89.4	90.1	90.2	90.2
hol.	47.0	48.5	46.9	50.5	81.0	81.0	81.7	80.1
Avg.	53.5	<u>54.8</u>	54.7	**55.1**	76.2	76.9	**77.3**	<u>77.1</u>

Table 4: **Results of our proposed multi-task object 6D pose framework.** We report the ADD-0.1d accuracy for each object class and for each task weighting scheme. BM is the original base model, either (a) WDR-Net or (b) ZebraPose, without auxiliary tasks. Three auxiliary task weighting schemes are compared here: EQ=equal Weights, GS=GradSim, AL=AuxiLight. As discussed in Sec. 5.5, the Auto-λ and ForkMerge algorithms are excluded from comparison because of their impracticability on large, real-world models.

Datasets. We perform our experiments on the 8-object Occ-LINEMOD dataset (Brachmann et al., 2014), which is the dataset used by both ZebraPose and WDR-Net, i.e., the two base models selected in this paper.

Evaluation Metric. The metric is the same as our selected base models. Specifically, we use the ADD(-S) metric to evaluate 6D pose estimation performance, which measures the point-to-point distance in 3D space between the 3D object vertices transformed by the predicted pose and the ground-truth pose. We report the ADD-0.1d accuracy, i.e. the percentage of samples for which the ADD(-S) error is smaller than 10% of the object diameter.

Results on WDR-Net. In Table 4 (a), we report the results of our proposed multi-task 6D pose model based on WDR-Net (Hu et al., 2021). These results evidence the benefits of the auxiliary tasks; all three task weighting schemes improve the performance of the base model. Nevertheless, our AuxiLight yields the largest boost, with an increased ADD-0.1d accuracy of 1.6 percent.

Results on ZebraPose. In addition to WDR-Net, we also test our multi-task method with ZebraPose as source model. The results are shown in Table 4 (b). Due to computational restrictions, we were not able to include all auxiliary tasks in our experiments, and only surface normal and reconstruction are included as auxiliary tasks. These tasks correspond to the individual tasks bringing the largest accuracy boost, as will be elaborated in Appendix. While here, GradSim has a slight advantage over AuxiLight, both task weighting schemes outperform the single-model baseline. Given the margin over GradSim on other cases, we still consider that AuxiLight is in general advantageous over GradSim.

6 Conclusion

In this work, we have introduced AuxiLight, a fast, lightweight auxiliary task weighting scheme that achieves comparable performance as the state-of-the-art algorithm, but at a much lower computational and memory burden. In addition, we have constructed an annotation cost-free use case of auxiliary learning based on 6D pose estimation methods, with the advantage that the tasks are freely-available via the 3D object models. Such annotation cost-free use case is the first of its kind to our best knowledge. We will release the code and the auxiliary ground-truth annotations publicly.

Limitations. AuxiLight relies purely on a gradient-alignment signal to weight auxiliary tasks and does not explicitly detect or mitigate negative transfer from harmful auxiliary tasks, unlike methods such as ForkMerge (Jiang et al., 2024); we illustrate this limitation in Appendix A.3. As a case study, the auxiliary tasks used for 6D pose estimation do not automatically generalize to other tasks. Nevertheless, the general idea behind cost-free annotation is the fact that access to a 3D model provides means to generate the ground

truth for different tasks. In principle, this idea could be leveraged in other contexts, such as 3D reconstruction and 6D object tracking from images, or potentially 3D object part segmentation from point clouds, where auxiliary tasks could include curvature and signed distance prediction. For 2D tasks where 3D models are not provdied, creating zero-annotation auxiliary tasks will be difficult due to the lack of ability to reconstruct the scene. However, self-supervised learning and similar tasks, including image inpainting or super-resolution, will nonetheless be possible in these cases. Demonstrating this, however, goes beyond the scope of this work, but provides interesting directions for future research.

Acknowledgement

This work is funded by the Swiss National Science Foundation and the Swiss Innovation Agency (Innosuisse) via the BRIDGE Discovery grant No. 40B2-0_194729.

References

Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *Learning theory and kernel machines*, pp. 567–580. Springer, 2003.

Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pp. 536–551. Springer, 2014.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pp. 312–321, 2003.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Yan Di, Fabian Manhardt, Gu Wang, Xiangyang Ji, Nassir Navab, and Federico Tombari. So-pose: Exploiting self-occlusion for direct 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12396–12405, 2021.

Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10786–10796, 2021.

Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11632–11641, 2020.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Judy Hoffman, Brian Kulis, Trevor Darrell, and Kate Saenko. Discovering latent domains for multisource domain adaptation. In *European Conference on Computer Vision*, pp. 702–715. Springer, 2012.

Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3385–3394, 2019.

Yinlin Hu, Sebastien Speierer, Wenzel Jakob, Pascal Fua, and Mathieu Salzmann. Wide-depth-range 6d object pose estimation in space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15870–15879, 2021.

Junguang Jiang, Baixu Chen, Junwei Pan, Ximei Wang, Dapeng Liu, Jie Jiang, and Mingsheng Long. Forkmerge: Mitigating negative transfer in auxiliary-task learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE international conference on computer vision*, pp. 1521–1529, 2017.

Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 2938–2946, 2015.

Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6129–6138, 2017.

Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166, 2009.

Chi Li, Jin Bai, and Gregory D Hager. A unified framework for multi-view multi-class object pose estimation. In *Proceedings of the european conference on computer vision (eccv)*, pp. 254–269, 2018.

Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7678–7687, 2019.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32, 2019.

Xi Lin, Xiaoyuan Zhang, Zhiyuan Yang, Fei Liu, Zhenkun Wang, and Qingfu Zhang. Smooth tchebycheff scalarization for multi-objective optimization. *arXiv preprint arXiv:2402.19078*, 2024.

Shikun Liu, Stephen James, Andrew J Davison, and Edward Johns. Auto-lambda: Disentangling dynamic task relationships. *arXiv preprint arXiv:2202.03091*, 2022.

Debabrata Mahapatra and Vaibhav Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In *International Conference on Machine Learning*, pp. 6597–6607. PMLR, 2020.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory*, pp. 2388–2464. PMLR, 2019.

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pp. 35–57. Springer, 2003.

Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetaya. Auxiliary learning by implicit differentiation. *arXiv preprint arXiv:2007.02693*, 2020.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4561–4570, 2019a.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019b.

Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE international conference on computer vision*, pp. 3828–3836, 2017.

Filippo Santambrogio. {Euclidean, metric, and Wasserstein} gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7:87–154, 2017.

Guntram Scheithauer. Jorge nocedal and stephen j. wright: Numerical optimization, springer series in operations research, 1999, isbn 0-387-98793-2 in the preface the authors state... our goal in this book is to give a compre-hensive description of the most powerful, state-of-the-art, techniques for solving continuous optimization problems. by presenting the motivating ideas for each.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. *ECCV (5)*, 7576:746–760, 2012.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pp. 9120–9132. PMLR, 2020.

Yongzhi Su, Mahdi Saleh, Torben Fetzer, Jason Rambach, Nassir Navab, Benjamin Busam, Didier Stricker, and Federico Tombari. Zebrapose: Coarse to fine surface encoding for 6dof object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6738–6748, 2022.

Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3343–3352, 2019.

Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3712–3722, 2018.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pp. 94–108. Springer, 2014.

A Appendix

A.1 Illustration of Auxiliary-Task Learning

To help the reader better understand the scheme of the proposed auxiliary-task 6D pose estimation discussed in Sec. 5, we show the corresponding framework in Fig. 1. Our proposed auxiliary-task learning framework (colored in red) can be employed to improve the training of existing 6D pose estimation methods (colored in grey). Solid arrows indicate the forward data flow, while dashed arrows show the backward loss flow. It is self-evident that 1) The ground truths of auxiliary tasks can be generated automatically with little annotation cost, thanks to the availability of 3D model in the specific task of 6D pose estimation; 2) Our auxiliary-task framework can be seen as an add-on that is not involved during inference, and thus yields a more effective model at no additional cost in test-time speed.

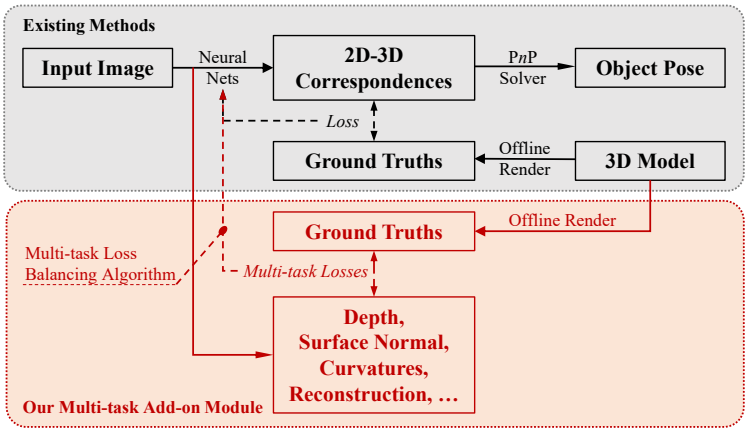

Figure 1: High-level summary of our proposed auxiliary-task learning framework (colored in red). Our framework can be employed to improve the training of existing 6D pose estimation methods (colored in grey). Solid arrows indicate the forward data flow, while dashed arrows show the backward loss flow. (Better viewed in color.)

WDRNet	+depth	+normal	+curvature	+reconst.
53.5	54.1	54.5	54.0	54.4
ZebraPose	+depth	+normal	+curvature	+reconst.
76.2	76.9	77.1	76.1	77.4

Table 5: Results of our multi-task framework when one single auxiliary task is used.

A.2 Implementation Details of Sec. 4.1

In this section we elaborate on implementation details of the AuxiLight experiments that are carried out on the NYUv2 dataset (Silberman et al., 2012) and the CityScapes dataset (Cordts et al., 2016), which are omitted in Sec. 4.1. Our experimental setup, as well as code implementation largely follows (Jiang et al., 2024), which is the state-of-the-art auxiliary loss balancing algorithm to date. We will release our code upon acceptance.

Implementation details. For the NYUv2 dataset, we train semantic segmentation, depth estimation and surface normal prediction; and for the CityScapes dataset, we train semantic segmentation, disparity estimation and part segmentation. The tasks of segmentation, depth and normal are respectively evaluated by mean intersection over union (IoU, the larger the better), absolute error (Err., the smaller the better), and mean angle distance error (Err., the smaller the better). For the task of disparity estimation on the CityScapes dataset, we calculate the depth error as well, as essentially disparity is the inverse of depth. For both datasets, we train three models where one task is the main task and the other two tasks are auxiliary tasks, and we evaluate the performance of the main task with its corresponding metric. Each model is trained and evaluated twice to reduce randomness. For Office-31 and DomainNet datasets, multiple domaines are evaluated and the accuracy is reported. 50 percent of images are used as validation set to keep up with the set up used in (Jiang et al., 2024).

A.3 Discussion on the Auxiliary Weighting

In the AuxiLight implementation, the auxiliary weights $\lambda_i(t)$ are updated based on a cumulative gradient-alignment signal and are not explicitly constrained, which in principle could lead to negative values or unbounded growth if the alignment has a non-zero mean. Conceptually, $\lambda_i(t)$ is intended to act as an online measure of task alignment for auxiliary task weighting, rather than a parameter that must converge to a

fixed or strictly positive value. However in practice, we found the training to be stable with appropriately small step sizes, and $\lambda_i(t)$ remained well-behaved across experiments.

To showcase the weight dynamics of auxiliary task learning, we run an experiment where both useful and harmful auxiliary tasks are present. Specifically, we train on the NYUv2 dataset with semantic segmentation as the main task, with three auxiliary tasks: depth prediction, surface normal prediction and in addition, a random noise prediction. The random noise prediction is intended as an irrelevant or harmful task for auxiliary task learning. We plot the weight of the auxiliary task against the training epochs. We start with an initial weighting of $\lambda = 0.1$ for all auxiliary tasks, and run our AuxiLight algorithm. As we can see, the depth auxiliary task achieves the highest weighting, which implies a strong relevance of depth estimation is identified by the algorithm with respect to the main task, semantic segmentation. On the other hand, the random noise prediction yields a minimum auxiliary task weighting compared to the other tasks. However, the weight does not decrease to zero as expected for random noise prediction. Unlike ForkMerge Jiang et al. (2024) that explicitly solves the issue of negative transfer and has a better performance, our method is not capable of solving such problem of negative transfer. This is a limitation of our paper, but comes with a benefit of the method being faster and more lightweight.

A.4 Additional Analysis of Sec. 5.5

In this section, we provide additional analyses for the problem of auxiliary-task 6D pose estimation discussed in Sec. 5.5.

Contribution of the individual auxiliary tasks. In addition to the previous experiments where multiple auxiliary tasks were used, we conduct experiments using one single auxiliary task at a time, to compare the contribution of the different auxiliary tasks. In this case, a fixed auxiliary weight is used. The results are shown in Table 5, and we show that, except for curvature with ZebraPose, all auxiliary tasks help improve the base 6D pose estimation model. Moreover, in both cases, surface normal and reconstruction rank as the top two auxiliary tasks.

Runtime analysis. Table 6 shows the running time of our proposed auxiliary-task 6D pose framework. Compared to the base model, our auxiliary-task framework entails two additional computations: 1) Generating auxiliary task ground truth annotations, and 2) training additional auxiliary branches. While rendering ground truths costs time, it is performed only once in an offline fashion. We highlight that 1) our proposed AuxiLight is indeed a fast, lightweight auxiliary loss balancing algorithm compared to other equivalents; and 2) our method does not introduce additional computation during inference; the additional auxiliary branches can be discarded.

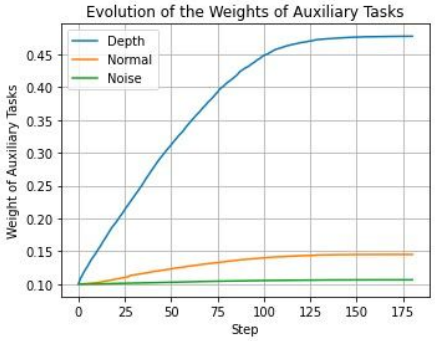

Figure 2: **The auxiliary-task weighting of AuxiLight** on NYUv2 dataset against epoch, with an additional noise prediction present as an auxiliary task.

Task			Average time per frame
Rendering Auxiliary GT of...	depth		240 ms
	normal		50 ms
	curvature		140 ms
	overall		430 ms
Training	ZebraPose	Baseline	100 ms
		+ GradSim	180 ms
		+ Auto-λ	N/A
		+ ForkMerge	N/A
		+ AuxiLight	180 ms
	WDRNet	Baseline	110 ms
		+ GradSim	140 ms
		+ Auto-λ	N/A
		+ ForkMerge	N/A
		+ AuxiLight	140 ms
Testing	ZebraPose	Baseline	220 ms
		+ GradSim	220 ms
		+ Auto-λ	N/A
		+ ForkMerge	N/A
		+ AuxiLight	220 ms
	WDRNet	Baseline	70 ms
		+ GradSim	70 ms
		+ Auto-λ	N/A
		+ ForkMerge	N/A
		+ AuxiLight	70 ms

Table 6: **Runtime Analysis of Auxiliary-task 6D Pose Estimation.** The times are rounded up to multiples of 10ms. Results with Auto-λ are simply marked with N/A because this algorithm requires too much GPU memory and running time that it fails with our experiment set-up (as discussed in Sec. 4.1).