Multi-Loss Weighting with Coefficient of Variations

Rick Groenendijk[1] Sezer Karaoglu[3,1] Theo Gevers[1,3] Thomas Mensink[2,1]
[1] University of Amsterdam [2] Google Research [3] 3DUniversum
{r.w.groenendijk, th.gevers}@uva.nl s.karaoglu@3duniversum.com mensink@google.com

Abstract

Many interesting tasks in machine learning and computer vision are learned by optimising an objective function defined as a weighted linear combination of multiple losses. The final performance is sensitive to choosing the correct (relative) weights for these losses. Finding a good set of weights is often done by adopting them into the set of hyper-parameters, which are set using an extensive grid search. This is computationally expensive. In this paper, we propose a weighting scheme based on the coefficient of variations and set the weights based on properties observed while training the model[1]*. The proposed method incorporates a measure of uncertainty to balance the losses, and as a result the loss weights evolve during training without requiring another (learning based) optimisation. In contrast to many loss weighting methods in literature, we focus on single-task multi-loss problems, such as monocular depth estimation and semantic segmentation, and show that multi-task approaches for loss weighting do not work on those single-tasks. The validity of the approach is shown empirically for depth estimation and semantic segmentation on multiple datasets.*

1. Introduction

In a wide variety of computer vision tasks, models are taught predictive capabilities through optimising some objective function. While for some tasks the objective function consists of a single loss, *e.g.* such as cross-entropy loss for classification, for many tasks the objective is a combination of loss functions, *e.g.* a L1-loss and a Structural Similarity loss for monocular depth estimation [14]. In general, the final objective function $\mathcal{L}_{total}$[2] is a linear combination of a set of loss functions $\mathcal{L}_i$:

$$\mathcal{L}_{total} = \sum_i \alpha_i \, \mathcal{L}_i + R(\boldsymbol{\alpha}), \tag{1}$$

where $\boldsymbol{\alpha}$ denotes a set of weights and $R(\cdot)$ denotes some regularisation on these weights.

The model performance is sensitive to choosing the correct weight values $\boldsymbol{\alpha}$. In literature, the weighing of different losses is usually based on equal weighting, with the inherent assumption that each loss should contribute equally to the problem at hand, or by a hyper-parameter grid-search over the loss weights $\boldsymbol{\alpha}$. This manual tuning of loss weights is (still) prevalent, for example in segmentation [32, 16, 40], depth estimation [13, 9], pose estimation [36], style transfer [10, 18], and adversarial learning [20]. This is feasible only when a small number of loss functions are combined, when more loss functions are combined. To illustrate this consider [38] who combine 40 losses, or [25] who combine 78 losses. In these cases, performing a grid-search would be too expensive computationally, and a method to set $\boldsymbol{\alpha}$ along with model parameters θ is desirable.

Static loss weights, *i.e.* weights which are fixed through training, either by equal weighting or grid-search, might not result in optimal learning behaviour and final performance. For example, for some learning tasks it has been shown that it is beneficial to learn easy tasks first before the more difficult tasks are introduced [8, 26]. Ideally, the weights $\boldsymbol{\alpha}$ are adapted over time to guide the learning process.

Weighting schemes for combining multiple losses has been studied extensively in the context of multi-task learning, where multiple tasks, each with a single loss, are combined. This is appealing since conceptually task-specific information could be leveraged in related tasks to encode a shared representation [2, 34]. Research in this context is tremendously broad: from architectural modifications to facilitate joint learning [28, 29, 27], to multi-task learning from domains that do not share intrinsic commonalities [22] and from multi-task to reinforcement learning [1, 33], to the use of style generation methods to learn the joint data distribution [31]. The main focus of this paper however is on adaptive weighting schemes for multiple losses.

In multi-task settings, the weights could be determined by, prioritising the most difficult task [15], learning the easiest tasks first [26], estimating the aleatoric uncertainty [23, 24], normalising the gradients [3] or finding the Pareto optimal so-

[1]Source code available at: https://github.com/rickgroen/cov-weighting

[2]For clarity and consistency, the term **objective function** always denotes the final learning objective for a model / task, while the term **loss** is used to denote a single element in such a (composite) objective function.

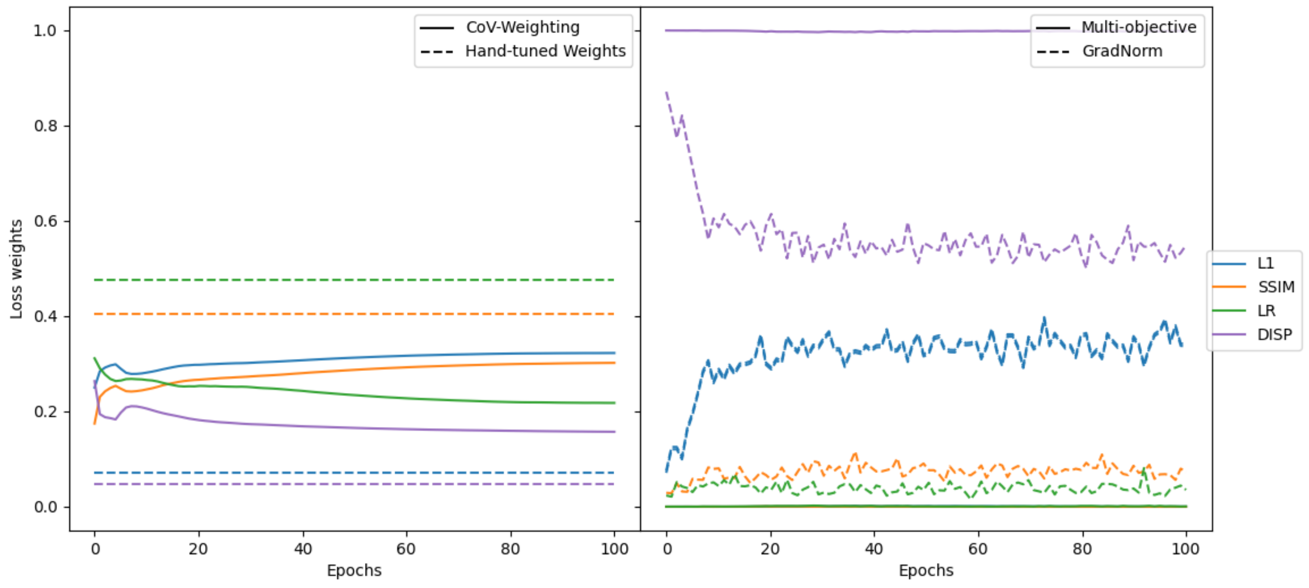


Figure 1. **(left)** Loss weights throughout training for both our method (lines) and the static optimal weights (dotted); **(right)** Averaged loss weights throughout training of Multi-objective optimization [35] (lines) and loss weights for GradNorm [3] (dotted).

lution to share model capacity across the different tasks [35]. In contrast to these approaches this paper focus on the single-tasks, such as monocular depth estimation and semantic segmentation in which multiple losses are combined and we show that multi-task approaches do not work for these tasks.

In this paper we propose CoV-Weighting, a single-task multi-loss weighting scheme that explicitly makes use of the statistics inherent to the losses to estimate their relative weighing. The method is founded on the *coefficient of variation* (CoV), which is the ratio of the standard deviation σ to the mean μ and shows the extent of variability of the observed losses in relation to the mean of the observed losses. In order to compare the CoV values of the different losses, these must have a common meaningful zero point. Therefore the CoV is estimated based on the observed loss ratio between the current observation and the mean value of that loss, instead of the direct observations of the loss. This transform different loss functions into a common scale and allow for comparisons. In Figure 1 the development of the loss weights of CoV-Weighting are shown throughout training.

To validate the proposed method, extensive experimental evaluation across two tasks (depth prediction and semantic segmentation) and three datasets (KITTI, CityScapes, and PASCAL Context) is performed. The experiments show that CoV-Weighting surpasses all multi-task loss weighting schemes and performs on par with hand-tuned weights in a single pass, with a learning rate optimised for the hand-tuned weighing. After a hyper-parameter search over the learning rate, CoV-Weighting outperforms any of all current models.

2. Method

In this paper, we introduce a weighting scheme for single-task learning problems where the objective function is defined as a linear combination of losses. Each loss quantifies the cost with respect to a desired output or auxiliary objective, for example the pixel-wise L1 loss to measure the reconstruction error, a cross-entropy loss for (pixel-wise) classification, or a semantic embedding loss to capture the classes present in an image. The goal is to find the (optimal) set of weights $\boldsymbol{\alpha}$ for the combination of losses, *c.f*. Eq. 1. In order to find these weights, the following hypothesis is used:

Hypothesis 1 *A loss term has been satisfied when its variance has decreased towards zero.*

In other words, this hypothesis says that a loss with a constant value should not be optimised any further. Variance alone, however, is not sufficient, given that it can be expected that a loss which has a larger (mean) magnitude, also has a higher absolute variance. Even if the loss is relatively less variant. Therefore, we propose to use the *coefficient of variation* $c_{\mathcal{L}}$ of loss $\mathcal{L}$, which shows the variability of the observed loss in relation to the (observed) mean:

$$c_{\mathcal{L}} = \frac{\sigma_{\mathcal{L}}}{\mu_{\mathcal{L}}}, \tag{2}$$

where $\mu_{\mathcal{L}}$ and $\sigma_{\mathcal{L}}$ denote the mean and the standard deviation of loss $\mathcal{L}$. The coefficient of variation is also known as the *relative standard deviation* and has the advantage that the value is independent of the scale/magnitude in which the observations are measured. This is relevant, given that we known that different losses act on different scales. Coefficient of variation decouples the loss magnitude from loss weighting, so a loss with small magnitude may still be relatively impactful when it is complex and variant; a bigger loss that hardly values across training examples is assigned less weight.

2.1. Loss Ratios for Coefficient of Variation

A requirement for computing a correct coefficient of variation is that the observations are derived from a ratio-scale, that is with a unique and non-arbitrary zero value. Then, it allows to fairly compare the uncertainty between different series of measurements, even with different magnitudes.

Not all loss functions are measured on such a ratio-scale. Therefore, we propose to use the *loss-ratio* ℓ as measurement, instead of the loss value itself, which we define as:

$$\ell_t = \frac{\mathcal{L}_t}{\mu_{\mathcal{L}_{t-1}}}, \tag{3}$$

where $\mathcal{L}_t$ is the observed loss value at time step t, and $\mu_{\mathcal{L}_{t-1}}$ is the mean of the observed losses up to time $t-1$. The loss-ratio has also been used in multi-task loss weighting setting. In [3] the ratio of the current observation with the first loss measurement is used ($\frac{\mathcal{L}_t}{\mathcal{L}_0}$), which we found to yield noisy and initialisation dependent ratios. In [27] the ratio of the current observation and the previous observation is used ($\frac{\mathcal{L}_t}{\mathcal{L}_{t-1}}$), which we found to yield unstable ratios. Our ratio is more robust and experimentally we explore the effect of using a decay mean instead of the mean over all observations.

The loss ratio ℓ has the same meaningful zero point across different loss functions: when $\mathcal{L}_t$ is zero $\ell_t = 0$. Moreover it is a relative comparison of two measurements of the loss statistic. The weight α_{it} is based on the coefficient of variation of the loss-ratio c_{ℓ_i} for loss $\mathcal{L}_i$ at time step t:

$$\alpha_{it} = \tfrac{1}{z_t}\, c_{\ell_{it}} = \tfrac{1}{z_t}\, \frac{\sigma_{\ell_{it}}}{\mu_{\ell_{it}}}, \tag{4}$$

where z_t is a normalising constant independent of i: $z_t = \sum_i c_{\ell_{it}}$. This ensures that $\sum_i \alpha_{it} = 1$, which is important to decouple the loss weighting from the learning rate.

There are two forces that simultaneously determine the value of the loss weights:

- The loss weights increase when the loss ratio ℓ_{it} decreases, that is when the loss $\mathcal{L}_{it}$ is below the mean loss $\mu_{\mathcal{L}_i}$. This encourages losses that are learning quickly, and dampens the effect on high outliers on the magnitude of the loss.

- The loss weights increase when the standard deviation over the history of loss ratios $\sigma_{\ell_{it}}$ increases. This ensures that more learning occurs when a loss-ratio is more variant. That is, when a particular objective has historically been more challenging, this term makes the cost function more powerful.

Method	Definition α_i	Main Property
Uncertainty [24]	$\frac{1}{\sigma_i^2} + \frac{\log \sigma_i}{\mathcal{L}_i}$	jointly learned
Multi-objective [35]	$\sum_i \alpha_i \nabla_{\theta_s} \mathcal{L}_i = 0$	gradient based
GradNorm [3]	$\frac{\mathcal{L}_i(t)/\mathcal{L}_i(0)}{g_i(t)}$	separate loss
CoV-Weighting	$\frac{\sigma_{\ell_i}}{\mu_{\ell_i}}$	observed

Table 1. Overview of the different weighting schemes considered in this paper, with property and definition of α_i. See text for details.

2.2. Robust Estimation

Using CoV-Weighting the loss weightings are inferred directly from the history of the observed loss values. To estimate the loss-ratio and the coefficient of variation robustly, we use Welford's algorithm [37], an online estimate, to track the mean of $\mathcal{L}$, and the mean and standard deviation of ℓ using the following update rules:

$$\mu_{\mathcal{L}_t} = \left(1 - \tfrac{1}{t}\right) \mu_{\mathcal{L}_{t-1}} + \tfrac{1}{t}\mathcal{L}_t, \tag{5}$$

$$\mu_{\ell_t} = \left(1 - \tfrac{1}{t}\right) \mu_{\ell_{t-1}} + \tfrac{1}{t}\ell_t, \quad \text{and} \tag{6}$$

$$\boldsymbol{M}_{\ell_t} = \left(1 - \tfrac{1}{t}\right) \boldsymbol{M}_{\ell_{t-1}} + \tfrac{1}{t} \left(\ell_t - \mu_{\ell_{t-1}}\right) \left(\ell_t - \mu_{\ell_t}\right), \tag{7}$$

the standard deviation is then given by $\sigma_{\ell_t} = \sqrt{\boldsymbol{M}_{\ell_t}}$. Assuming converging losses and ample training iterations, the online mean and standard deviation converge to the true mean and standard deviation of the observed losses over the data.

A potential downside of this approach is that variance over time of a loss is smoothed out. Therefore we also experiment with decaying online estimates, then t is a fixed factor, *e.g.* 20 or 100, to weight the aggregated previous observations and the current observation.

3. Relation to Other Methods

In this section CoV-Weighting is compared in-depth to three multi-task loss weighting methods: Gradient normalisation (GradNorm) [3], Multi-objective optimisation [35], and Uncertainty Weighting [24]. Table 1 summarises the main differences between the multi-task loss weighting methods and CoV-Weighting .

Uncertainty Weighting [23, 24]. Uncertainty Weighting models the homoscedastic aleatoric uncertainty in multi-task settings. While homoscedastic noise is not dependent on the input data, it might be task-dependent. The observed task-loss $\mathcal{L}_i$ is seen as an observation from a Gaussian distribution $\mathcal{N}(\mathcal{L}_i; \sigma_i)$. The final objective minimises the log-likelihood of these Gaussian distributions:

$$\mathcal{L}_{total} = \sum_i \frac{\mathcal{L}_i}{2\sigma_i^2} + \log \sigma_i, \tag{8}$$

where the variance σ_i^2 is learned jointly with the model parameters. It can be shown, however, that when the optimal variance could be used, Uncertainty Weighting results in a parameter less log-loss: $\sum_i \log(\mathcal{L}_i)$. This means the smallest loss has the most impact on the gradient.

Uncertainty Weighting uses the homoscedastic (data independent) noise, in the single-task multi-loss setting this might be unsuitable since the observational noise over the same output cannot be used to weight different losses. Moreover, in Uncertainty Weighting the loss weights increase when the observational noise in the outputs decreases. This naturally happens when the model is presented with more training data. In contrast CoV-Weighting uses the variance in the observed losses throughout training and when the noise in a loss ratio increases the loss weight also increases.

As opposed to other methods, Uncertainty Weighting, in the original formulation, does not satisfy $\sum_i \alpha_i = 1$, and hence there is a coupling between the loss weights (which can be arbitrarily) and the global learning rate. In [24] the global learning rate is annealed by a power law to counteract the exponential increase of loss weights. Unfortunately this complicates direct comparison against other loss weighting methods because it is exceedingly difficult to fairly control for global learning rate.

GradNorm [3]. In GradNorm [3] gradient normalisation is suggested to balance the gradient norms for each task at each training iteration step at a chosen layer $\mathcal{W}$. This layer is often the last shared layer between the different tasks. The authors argue that gradients are ideally balanced at this shared layer. In the single-task multi-loss setting, all layers are shared and hence GradNorm is computed at the output layer.

To balance multiple loss terms, the weights $\boldsymbol{\alpha}$ are learned along the model parameters, with a separate loss function. It can be derived, however, that the optimal weight values $\boldsymbol{\alpha}$ (before normalisation) equal to:

$$\alpha_i \propto \frac{\mathcal{L}_i(t)/\mathcal{L}_i(0)}{g_i(t)}, \tag{9}$$

where $g_i(t)$ denotes the norm of the gradient of the parameters with respect to the loss $\mathcal{L}_i$ at the shared layer, hence the name GradNorm. In other words, the loss ratio, between the loss at time t and the first loss at time 0, is divided by the current norm of the gradient. This is counter intuitive for single-task multi-loss learning, given that their loss ratio is an *inverse* measure: the smaller the value, the better the loss is training. So better performing losses are slowed down during training compared to difficult losses (where $\mathcal{L}(t) \approx \mathcal{L}(0)$), when they have an equal gradient norm.

Compared to the proposed loss ratio, in GradNorm the loss ratio is based on the initial loss value. When the network has just been initialised, this can be a poor estimate to measure the velocity with which the network learns for a specific loss. This is solved in CoV-Weighting by using the loss ratio between the current loss and the mean of the observed losses.

Multi-Objective Optimisation [35]. The authors of [35] state that while multi-task learning can, in general, be beneficial for all tasks, some individual tasks could compete for model capacity. That is, the shared encodings may not be equally informative for all tasks. Hence, the authors are interested in finding Pareto optimal solutions for tasks using the Frank Wolve algorithm [21]. A potential issue in single-task learning for multi-objective optimisation is that single-task learning is inherently a single-objective optimisation. That is, the optimal solution cannot be assumed to be a Pareto optimum between the different loss functions. It is expected that dealing with auxiliary losses will be especially challenging for this method.

4. Experiments

In this section, the proposed method is evaluated on two distinct scene understanding tasks: Depth estimation on KITTI [12, 11] and CityScapes [4], and semantic segmentation on the PASCAL Context dataset [30]. The purpose of these experiments is two-fold: First, to compare the proposed dynamic weights to a set of static weights (equal weighting or hand tuned). Second, to test the proposed method against three dynamic multi-task loss weighing approaches.

To fairly compare the different methods, our code includes the proposed weighing scheme and implementations of the baseline methods. The dynamic baselines that are used are Uncertainty weighing [24], GradNorm [3], and Multi-objective optimisation [35]. GradNorm requires an additional hyper-parameter as a form of temperature scaling on the loss weights (before normalisation). Preliminary experiments show that the performance is sensitive to this value; using grid-search it is set to 1.5 for all experiments. It should be noted that careful tuning of this parameter is required before performance is satisfactory, unlike with the other baselines that are used.

4.1. Depth Estimation

In these experiments, we learn to estimate depth from left/right image pairs by means of photo-metric reconstruction, using an estimated disparity map; depth can be inferred by warping the disparity map using the camera intrinsics. We follow the network architecture and objective functions of [13]. The objective function combines the L1 loss, Structural Similarity loss (SSIM), left-right consistency loss (LR), and disparity gradient loss (DISP) to train a single network. Following [13], the hand-tuned weights are set to $\{\alpha_{L1} = 0.15, \alpha_{SSIM} = 0.85, \alpha_{LR} = 1.0, \alpha_{DISP} = 0.1\}$,

	ARD	RMSE log	$\delta < 1.25$
	lower		higher
full history	<u>0.1988</u>	0.3118	0.6988
$t = 20$	0.2159	0.3479	0.7063
$t = 100$	0.2095	0.3498	0.7050
$t = 1000$	0.2037	<u>0.3009</u>	<u>0.7084</u>

Table 2. The effect of setting *t* to a factor that controls how much the updates of the statistics are effected by previous statistics. A **full history** is shown in e.g. Equation 5, in which *t* is the iteration count. Alternatively, *t* could be set to a static number to allow for a fixed decay. There is no clear benefit of using decay, hence the full history of statistics is used in further experiments.

which are normalised such that $\sum_i \alpha_i = 1.0$. Besides four different loss functions, disparity maps are regressed at four scales. At each scale all loss functions are evaluated for both left and right disparity maps. The full objective combines losses from all four scales *s*:

$$\mathcal{L}_{total} = \sum_{s=0}^{3} \sum_{d \in \{d_{left}, d_{right}\}} \mathcal{L}_{(s,d)} \tag{10}$$

$$\begin{aligned}\mathcal{L}_{(s,d)} = {} & \alpha_{L1}\mathcal{L}_{L1} + \alpha_S\mathcal{L}_S + \alpha_{LR}\mathcal{L}_{LR} \\ & + \alpha_{DISP}\frac{1}{2^s}\mathcal{L}_{DISP},\end{aligned} \tag{11}$$

where each of the αs are the loss weights that should be automatically weighted using one of the loss weighting schemes. A total of 32 losses is used to train the networks. See supplementary material for full details.

Dataset & Implementation Details Depth estimation is evaluated on the KITTI dataset [12, 11] using the Eigen split [6] and on the main split of the CityScapes dataset [4]. For all methods, images are down-sampled to a resolution of 256x512 and fed to an encoder-decoder network using batch normalisation [19]. The encoder is based on a ResNet50 [17] network; the decoder alternates bilinear interpolation up-sampling and convolutional layers [13]. The models are trained for 100 epochs on CityScapes and for 30 epochs on KITTI, using an Adam optimiser with a learning rate of *1e-4* (selected based on related works). For quantitative evaluation, a set of common metrics is used [6, 13]: Absolute Relative Distance (ARD), Squared Relative Distance (SRD), Root Mean Squared Error (RMSE), log Root Mean Squared Error (log RMSE), and multiple accuracies δ within a threshold *t* (δ_t, with $t \in \{1.25, 1.25^2, 1.25^3\}$). It is common for models that are trained on CityScapes to be evaluated on KITTI, since the quality of the ground truth disparities for CityScapes is relatively low. In this paper, this commonly-used evaluation strategy is also used. It means models that are trained on KITTI or on CityScapes are all evaluated on the improved Eigen test split [6] of the KITTI dataset.

Loss Statistic Estimation As was mentioned above, a potential downside of using a full history of losses to compute the statistics might result in over-smoothing the estimates. See for example the update steps in Equation 5. To verify whether this is actually the case, different parameters for *t* are tested. Results are depicted in Table 2. It is concluded that there is not a significant and apparent effect of using any particular factor *t*. Other initial experimentation confirms this conclusion. Consequently, a full history of loss statistics is kept throughout training.

CityScapes In this set of experiments, CoV-Weighting is compared against the models trained using a single-loss (*e.g.* SSIM or L1) and against the models trained on equal & hand-tuned weighted combination of losses. For training the models the CityScapes dataset is used, while evaluation is performed on the improved Eigen test split [6]. The results are shown in Table 3.

The performances of the single-loss models show that using auxiliary losses only will not work well for the original task. For example, the LR loss is tasked with rewarding symmetry in left and right disparity predictions; training with only the LR loss results in predicting purely zero-valued disparity maps, a perfect symmetry, albeit not valuable for depth estimation. These auxiliary losses by themselves do not correctly estimate depth, however they could guide or regularise the learning process.

The performance of using only the SSIM loss is close to the hand-tuned multi-loss counterpart. This is in line with the conclusions reported in [14]. However, the hand-tuned variant always outperforms the single SSIM loss model and the equal weighted variant.

The performance of CoV-Weighting is close to the hand-tuned weights for most of the metrics, and even outperforms these for RMSE log and $\delta < 1.25^3$. Moreover CoV-Weighting outperforms equal weighting on all but one metric (RMSE). Combined this shows that CoV-Weighting can adaptively set good weights for the different loss components.

KITTI In this set of experiments, CoV-Weighting is also compared against dynamic baselines developped for multi-task learning: Uncertainty weighing [24], GradNorm [3], and Multi-objective optimisation [35]. For this experiment the models are trained on the KITTI data using the improved Eigen train and test split. The depth estimation results are shown in Table 4.

We observe that the single-loss model trained on SSIM is a strong baseline, this is even more distinct than in the CityScapes experiment in Table 3. Moreover, the hand-tuned weights do not outperform SSIM and equal weighting on all metrics. Most likely the set of hand tuned weights from [13] are tuned for a (slightly) different setting and do

		ARD	SRD	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
		lower is better				higher is better		
Single-Loss	DISP	2.0301	103.0575	28.9682	2.4904	0.0024	0.0055	0.0102
	LR	0.7424	11.1500	17.3990	1.8043	0.0110	0.0382	0.0978
	L1	0.2502	3.2033	7.3217	**0.3290**	0.6838	0.8650	**0.9382**
	SSIM	*0.2002*	**1.6460**	**6.4964**	0.3682	**0.7066**	*0.8678*	0.9308
Multi-Loss	Equal	0.2106	1.7460	*6.6303*	0.3838	0.6887	0.8580	0.9231
	Hand-tuned	**0.1955**	**1.6028**	**6.3803**	*0.3507*	**0.7182**	**0.8734**	*0.9336*
	CoV-Weighting	**0.1988**	*1.6673*	6.6989	**0.3118**	*0.6988*	**0.8723**	**0.9426**

Table 3. Performance of depth estimation models trained on CityScapes. As is common for CityScapes models, evaluation is performed on the improved Eigen test set [6] in **depth** (meters). The **first**, **second**, and *third* best scoring methods are highlighted for each metric.

		ARD	SRD	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
		lower is better				higher is better		
Single-Loss	DISP	0.9040	13.5185	18.3165	2.5593	0.0000	0.0000	0.0000
	LR	6.3400	442.7141	65.5198	1.9283	0.0097	0.0265	0.0507
	L1	0.1132	0.8551	4.4699	0.1708	0.8700	0.9664	0.9897
	SSIM	**0.0901**	**0.5202**	**3.8581**	**0.1435**	**0.8991**	*0.9755*	*0.9935*
Multi-Loss	Static Weights							
	Equal	*0.0923*	*0.5477*	3.9856	*0.1464*	*0.8971*	**0.9763**	**0.9936**
	Hand-tuned	0.0944	0.5609	**3.8678**	0.1469	0.8950	0.9731	0.9926
	Dynamic Weights							
	Uncertainty [24]	1.0712	18.5936	11.7052	0.7417	0.3226	0.4760	0.5989
	GradNorm [3]	0.1152	1.1530	4.7621	0.1759	0.8669	0.9642	0.9887
	Multi-objective [35]	6.3400	442.7141	65.5198	1.9283	0.0097	0.0265	0.0507
	CoV-Weighting	**0.0912**	**0.5108**	*3.8717*	**0.1425**	**0.9023**	**0.9775**	**0.9942**

Table 4. Performance of depth estimation models trained on KITTI. Evaluated on the improved Eigen test set [6] in **depth** (meters). The **first**, **second**, and *third* best scoring methods are highlighted. CoV-Weighting outperforms all other methods on 5 out of 7 metrics.

not generalise to this setting. This shows the importance of finding (optimal) weights for the task and dataset at hand.

From the dynamic weight methods, two of the baselines (Uncertainty weighting and Multi-Objective optimisation) have difficulties training for this task and have far from good evaluation results. The third baseline, GradNorm, is able to train a decent model, but the performance stays behind compared to the model trained on SSIM, L1 or the model using equal weighted combination. Finally, CoV-Weighting shows best performance on 5 from the 7 metrics.

In Figure 1, the loss weights throughout training are shown for hand-tuned weights, GradNorm, Multi-objective optimisation, and CoV-Weighting for training on the CityScapes dataset. It is clear that Multi-objective optimisation underestimates the importance of SSIM and/or L1. GradNorm assigns the most loss weight to the disparity gradient loss, probably because it is the loss with the lowest magnitude. After the disparity loss, GradNorm assigns most weight to the L1 loss, *c.f.* Figure 1, this is reflected by the performance, in Table 4, the performance of GradNorm is just below the performance of using the L1 loss alone. CoV-Weighting attach relatively low importance to SSIM and high importance to the L1 loss compared to the static weights. The weights assigned to SSIM and L1 develop similarly. This is not surprising, since both losses measure reconstructed image quality, whereas the other two losses measure disparity quality. Also it gradually assigns more weight to SSIM and L1, and less to LR.

Win rates In this experiment we evaluate the win rate [39], the win rate indicates for how many images in the test-set a method is beneficial compared to a baseline method. This is implemented as a majority voting scheme over the 7 metrics evaluated on a single image. We show the win rates in Table 5 of the dynamic methods compared to the static multi-loss baselines. Since the win rate is the percentage of images in the test set for which a method outperforms another (baseline) method, it gives another insight in the

	Equal	Hand-tuned	Uncertainty [24]	GradNorm [3]	Multi-objective [35]	*CoV-Weighting*
Equal	-	0.4663	1.0000	0.8359	1.0000	0.4463
Hand-tuned	0.5322	-	1.0000	0.8574	1.0000	0.4356
CoV-Weighting	0.5537	0.5583	1.0000	0.8681	1.0000	-

Table 5. Win rates for methods compared to either an equal weighting or a hand-tuned baseline. Evaluated on the Eigen [6] split of KITTI. Models in rows are tested against the models in the columns. For example, *CoV-Weighting* outperforms equal weighting in 55% of the cases.

usefulness of the proposed method. From the results we observe that the three multi-task baselines are outperformed by the equal weighted and hand-tuned weighted variants. CoV-Weighting is favoured by over 55% of the images in the test set over both equal weighted and hand-tuned weighted models. This shows that our proposed method is able to improve over the performance of hand-tuned models without manually having to tune the loss weights.

4.2. Semantic Segmentation

The method is further evaluated on a semantic segmentation task to verify how well it generalises to other tasks. To this end, all loss weighing methods are implemented in conjunction with a Context Encoding Network (EncNet) [40]. The authors of [40] propose a Context Encoding module, that jointly learns to predict the presence of semantic classes in an image as well as the actual pixel-level class predictions. This module leverages global contextual information to aid pixel-level prediction. Additionally, it is possible to add another head to the penultimate layer of the encoder network to further aid prediction. In total the objective function consists of a standard Cross-Entropy loss (CE), a Semantic Encoding loss (SE), and an auxiliary Cross-Entropy loss (AUX) from a separate Fully Convolutional (FCN) head. The hand-tuned parameters α as given in [40] are $\{\alpha_{CE} = 1.0, \alpha_{SE} = 0.2, \alpha_{AUX} = 0.2\}$ Again, these weights are normalised to ensure $\sum_i \alpha_i = 1.0$

Implementation Details The EncNet is adapted and augmented with all loss weighing methods. The methods are tested on the PASCAL Context dataset [30, 7] which uses 4998 training images and 5105 validation images. For each image, there are annotations for up to 59 semantic classes. The encoder network is a ResNet50 [17] network using batch normalisation [19]; for the decoders, there is one Encoding Context Module that is attached to the final layer of the encoder, and one FCN head attached to the penultimate layer of the decoder. For optimisation, an SGD optimiser is used with a learning rate of *1e-4*, unless otherwise stated. The network is pre-trained on ImageNet [5] and then trained for 40 epochs on PASCAL Context. For quantitative evaluation, Pixel Accuracy (pACC) and Mean Intersection over Union (mIoU) are used, as in [40]. Background pixels are ignored during evaluation.

	pACC	**mIoU**
Single-Loss		
CE	0.769	0.440
AUX	0.012	0.004
SE	0.010	0.001
Static Weights		
Equal	0.759	0.419
Hand-tuned	0.768	0.437
Dynamic Weights		
Uncertainty [24]	**0.781**	**0.448**
GradNorm [3]	0.750	0.404
Multi-objective [35]	0.012	0.003
Proposed		
CoV-Weighting ($\gamma = 10^{-2}$)	**0.779**	**0.455**
CoV-Weighting ($\gamma = 10^{-3}$)	*0.770*	*0.441*
CoV-Weighting ($\gamma = 10^{-4}$)	0.768	0.436

Table 6. Performance of semantic segmentation models trained on PASCAL Context. Evaluated on 5104 images in the validation set. The **first**, **second**, and *third* best scoring methods are highlighted for each evaluation metric. *CoV-Weighting* is top performing when suitable learning-rate is chosen.

Results In this experiment different models are compared trained on a single loss, on a static weighted combination of losses, and on the dynamic weighted combination. The results on PASCAL Context are depicted in Table 6. Similarly as in the depth prediction task, also for semantic segmentation there is a single loss which provides a strong baseline. For this experiment, using only the CE loss yields performance on par with or slightly better than the hand-tuned weights. This could be due to the more shallow encoder network used (compared to [40]) or because the FCN head could be replaced with another Context Encoding module to improve performance as suggested in [40]. For this task, multi-objective weighting yields far from satisfactory results, probably due to attempting to assign high weight to an auxiliary loss with high gradient magnitude that has negative transfer with the main loss. GradNorm on the other hand, performs slightly worse than equal weighting. Similar to the case of multi-objective training, a possible explanation could be that a loss that serves purely as an auxiliary loss receives a too high loss weighting because its gradient has a high magnitude. Uncertainty weighting and CoV-Weighting

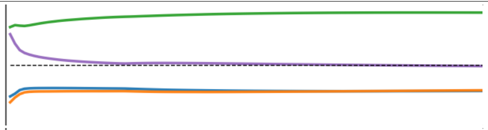
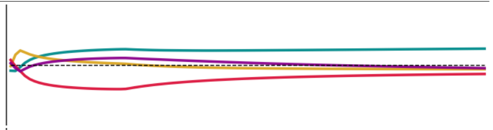
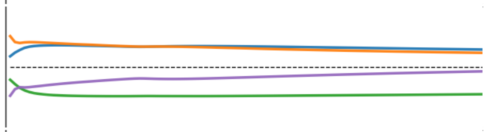
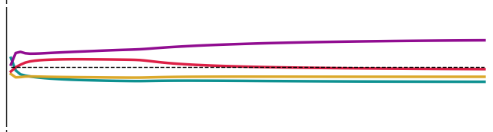
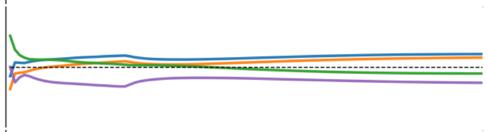
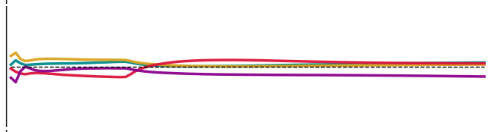
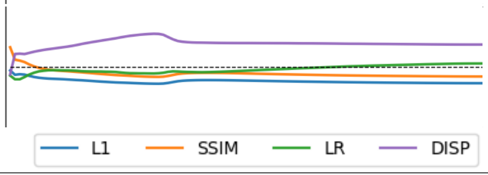
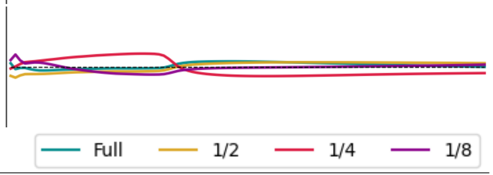
Variant	RMSE ↓	$\delta < 1.25$ ↑	Loss Weights	Scale Weights
$\frac{\sigma_L}{\mu_L}$	**6.6483**	0.7050		
$\frac{\mu_L}{\sigma_L}$	6.6977	0.7043		
$\frac{\sigma_l}{\mu_l}$ *	**6.6556**	**0.7063**		
$\frac{\mu_l}{\sigma_l}$	6.7670	**0.7072**		

Table 7. A comparison of four variants of CoV-Weighting on the CityScapes dataset. The top two rows indicate methods that have been trained using statistics over the **losses**; the bottom two rows indicate methods using statistics over the **loss ratios**. The * indicates CoV-Weightingas used throughout the paper. The **first** and **second** best scoring methods are highlighted for each metric. For each method the mean loss weights over the four different losses and mean loss weights over the different scales are shown; the weights are shown in the range [0, 0.5]; the dotted horizontal lines indicate a weight of 0.25. Best seen in color.

out perform both equal weighting and hand-tuned weights. For uncertainty weighting, it seems that this is in part due to the unnormalised weights used, *i.e.* $\sum_i \alpha_i$ can have any value, which in turn tunes the global learning rate to a higher value. Similarly, CoV-Weighting also benefits from a higher learning rate for this task ($\gamma = 10^{-2}$). Choosing a suitable learning rate is necessary regardless of the method that is chosen; our method does not require additional learning rate tuning compared to any of the baseline methods.

4.3. CoV-Weighting Variations

In this final set of experiments, several variants of CoV-Weighting[3] are explored. So far, the loss weights have been derived from the mean and variance in the observed loss ratios $\frac{\sigma_l}{\mu_l}$. In this experiment the mean and variances from the observed losses are used: $\frac{\sigma_L}{\mu_L}$, and the inverse-weighting is used: $\frac{\mu_l}{\sigma_l}$ & $\frac{\mu_L}{\sigma_L}$. The rationale for the inverse-weighting is to test the hypothesis that low-variance losses should be assigned a low weight.

For this experiment the depth estimation task on the CityScapes dataset is used. The performance is measured in RMSE and $\delta < 1.25$. The results are in Table 7. For qualitative analysis also the weights over time for the different losses (*middle*) and the different scales (*right*) are plotted.

From the results we first observe that all variants obtain decent performance, albeit assigning different weights to each of the losses and each of the scales. From this we may conclude that for the multi-loss single task scenario multiple paths lead to good performance.

Based on the previous experiments the expectation is that models with relatively high weights to L1 and/or SSIM will perform the best. CoV-Weighting and the $\frac{\mu_L}{\sigma_L}$-variant do so. However, per scale these methods assign different weights, since final predictions are required at full scale, the coarser scales act as auxiliary losses. The variant $\frac{\mu_L}{\sigma_L}$ wrongly assigns high weights to losses at coarse scale, *c.f.* Table 7.

[3]Thanks to the anonymous reviewer for this suggestion!

5. Conclusion

In this paper, CoV-Weighting has been introduced to automate tuning of loss weights specifically on single-task problems. Related methods from multi-task learning [35, 24, 3], are shown to not always be suited in a single-task setting, given that auxiliary losses cannot be weighed too heavily because they by themselves do not solve the task. These losses are often comparatively small and less complex. Consequently, the losses show less variance throughout training. CoV-Weighting explicitly makes use of these statistics, inspired by the coefficient of variation and assigns higher weights to losses that show higher relative variance. Experimentally CoV-Weighting either outperforms or performs on par with hand-tuned defined weights and outperform these when the optimal learning rate is used.

Future work could address CoV-Weighting in a multi-task setting. The use of the coefficient of variation allows to compare different one another. Since this holds also for losses from different tasks, it would be interesting to see how CoV-Weighting performs in multi-task learning.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

[2] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[3] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 2018.

[4] 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 *CVPR*, 2016.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[6] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html, 2010.

[8] Ysbrand Galama and Thomas Mensink. Itergans: Iterative gans to learn and control 3d object transformation. *Computer Vision and Image Understanding*, 2019.

[9] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.

[10] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423, 2016.

[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[13] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[14] Rick Groenendijk, Sezer Karaoglu, Theo Gevers, and Thomas Mensink. On the benefit of adversarial training for monocular depth estimation. *Computer Vision and Image Understanding*, 190:102848, 2020.

[15] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *ECCV*, 2018.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *CVPR*, 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[21] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML*, 2013.

[22] Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

[23] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, 2017.

[24] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

[25] Jae-Han Lee and Chang-Su Kim. Multi-loss rebalancing algorithm for monocular depth estimation. In *ECCV*, 2020.

[26] Changsheng Li, Junchi Yan, Fan Wei, Weishan Dong, Qingshan Liu, and Hongyuan Zha. Self-paced multi-task learning. In *AAAI*, 2017.

[27] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *CVPR*, 2019.

[28] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. 2015.

[29] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*.

[30] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[31] Arghya Pal and Vineeth N Balasubramanian. Adversarial data programming: Using gans to relax the bottleneck of curated labeled data. In *CVPR*, 2018.

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[33] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

[34] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[35] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *NeurIPS*, 2018.

[36] Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. Fast and robust hand tracking using detection-guided optimization. In *CVPR*, 2015.

[37] B.P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[38] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *ECCV*, 2018.

[39] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.

[40] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018.