Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics

Alex Kendall
University of Cambridge
agk34@cam.ac.uk

Yarin Gal
University of Oxford
yarin@cs.ox.ac.uk

Roberto Cipolla
University of Cambridge
rc10001@cam.ac.uk

Having same variance

Abstract

Numerous deep learning applications benefit from multi-task learning with multiple regression and classification objectives. In this paper we make the observation that the performance of such systems is strongly dependent on the relative weighting between each task's loss. Tuning these weights by hand is a difficult and expensive process, making multi-task learning prohibitive in practice. We propose a principled approach to multi-task deep learning which weighs multiple loss functions by considering the homoscedastic uncertainty of each task. This allows us to simultaneously learn various quantities with different units or scales in both classification and regression settings. We demonstrate our model learning per-pixel depth regression, semantic and instance segmentation from a monocular input image. Perhaps surprisingly, we show our model can learn multi-task weightings and outperform separate models trained individually on each task.

1. Introduction

Multi-task learning aims to improve learning efficiency and prediction accuracy by learning multiple objectives from a shared representation [7]. Multi-task learning is prevalent in many applications of machine learning – from computer vision [27] to natural language processing [11] to speech recognition [23].

We explore multi-task learning within the setting of visual scene understanding in computer vision. Scene understanding algorithms must understand both the geometry and semantics of the scene at the same time. This forms an interesting multi-task learning problem because scene understanding involves joint learning of various regression and classification tasks with different units and scales. Multi-task learning of visual scene understanding is of crucial importance in systems where long computation run-time is prohibitive, such as the ones used in robotics. Combining all tasks into a single model reduces computation and allows these systems to run in real-time.

Prior approaches to simultaneously learning multiple tasks use a naïve weighted sum of losses, where the loss weights are uniform, or manually tuned [38, 27, 15]. However, we show that performance is highly dependent on an appropriate choice of weighting between each task's loss. Searching for an optimal weighting is prohibitively expensive and difficult to resolve with manual tuning. We observe that the optimal weighting of each task is dependent on the measurement scale (e.g. meters, centimetres or millimetres) and ultimately the magnitude of the task's noise.

In this work we propose a principled way of combining multiple loss functions to simultaneously learn multiple objectives using homoscedastic uncertainty. We interpret homoscedastic uncertainty as task-dependent weighting and show how to derive a principled multi-task loss function which can learn to balance various regression and classification losses. Our method can learn to balance these weightings optimally, resulting in superior performance, compared with learning each task individually.

Specifically, we demonstrate our method in learning scene geometry and semantics with three tasks. Firstly, we learn to classify objects at a pixel level, also known as semantic segmentation [32, 3, 42, 8, 45]. Secondly, our model performs instance segmentation, which is the harder task of segmenting separate masks for each individual object in an image (for example, a separate, precise mask for each individual car on the road) [37, 18, 14, 4]. This is a more difficult task than semantic segmentation, as it requires not only an estimate of each pixel's class, but also which object that pixel belongs to. It is also more complicated than object detection, which often predicts object bounding boxes alone [17]. Finally, our model predicts pixel-wise metric depth. Depth by recognition has been demonstrated using dense prediction networks with supervised [15] and unsupervised [16] deep learning. However it is very hard to estimate depth in a way which generalises well. We show that we can improve our estimation of geometry *and* depth by using semantic labels and multi-task deep learning.

In existing literature, separate deep learning models

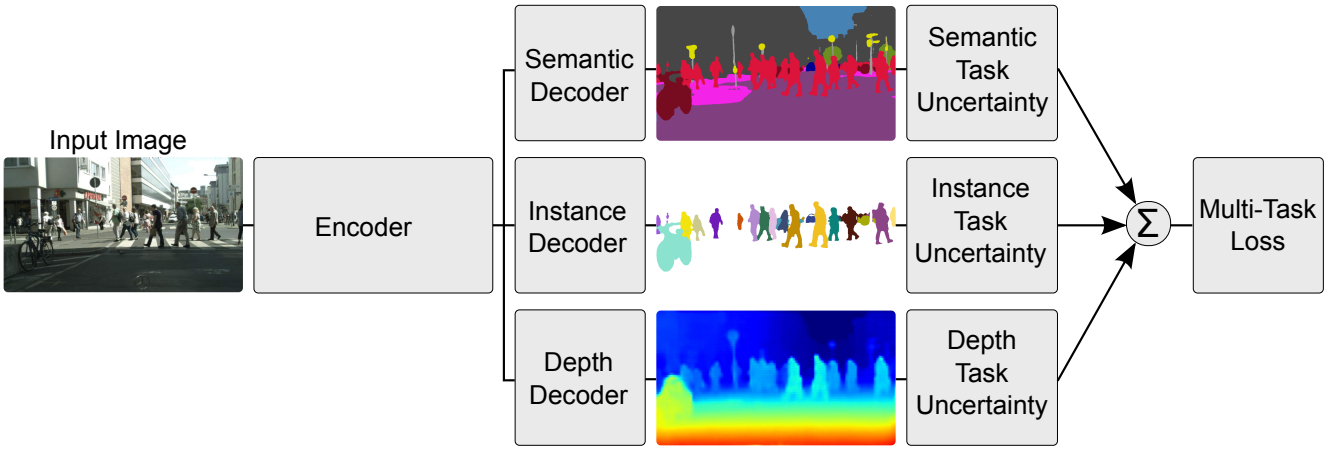


Figure 1: **Multi-task deep learning.** We derive a principled way of combining multiple regression and classification loss functions for multi-task learning. Our architecture takes a single monocular RGB image as input and produces a pixel-wise classification, an instance semantic segmentation and an estimate of per pixel depth. Multi-task learning can improve accuracy over separately trained models because cues from one task, such as depth, are used to regularize and improve the generalization of another domain, such as segmentation.

would be used to learn depth regression, semantic segmentation and instance segmentation to create a complete scene understanding system. Given a single monocular input image, our system is the first to produce a semantic segmentation, a dense estimate of metric depth and an instance level segmentation jointly (Figure 1). While other vision models have demonstrated multi-task learning, we show how to learn to combine semantics and geometry. Combining these tasks into a single model ensures that the model agrees between the separate task outputs while reducing computation. Finally, we show that using a shared representation with multi-task learning improves performance on various metrics, making the models more effective.

In summary, the key contributions of this paper are:

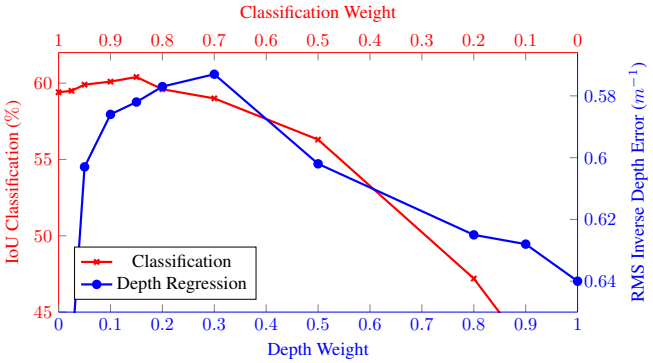
1. a novel and principled multi-task loss to simultaneously learn various classification and regression losses of varying quantities and units using homoscedastic task uncertainty,
2. a unified architecture for semantic segmentation, instance segmentation and depth regression,
3. demonstrating the importance of loss weighting in multi-task deep learning and how to obtain superior performance compared to equivalent separately trained models.

2. Related Work

Multi-task learning aims to improve learning efficiency and prediction accuracy for each task, when compared to training a separate model for each task [40, 5]. It can be considered an approach to inductive knowledge transfer which improves generalisation by sharing the domain information between complimentary tasks. It does this by using a shared representation to learn multiple tasks – what is learned from one task can help learn other tasks [7].

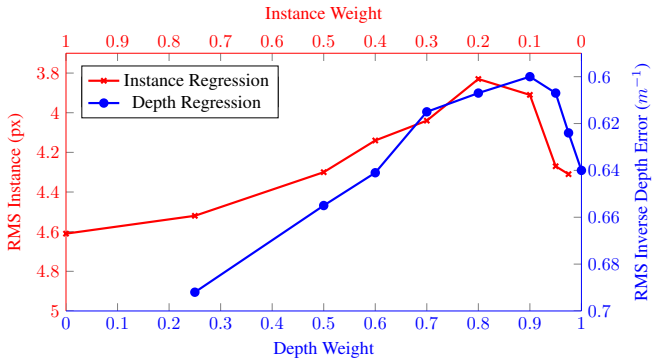
Fine-tuning [1, 36] is a basic example of multi-task learning, where we can leverage different learning tasks by considering them as a pre-training step. Other models alternate learning between each training task, for example in natural language processing [11]. Multi-task learning can also be used in a data streaming setting [40], or to prevent forgetting previously learned tasks in reinforcement learning [26]. It can also be used to learn unsupervised features from various data sources with an auto-encoder [35].

In computer vision there are many examples of methods for multi-task learning. Many focus on semantic tasks, such as classification and semantic segmentation [30] or classification and detection [38]. MultiNet [39] proposes an architecture for detection, classification and semantic segmentation. CrossStitch networks [34] explore methods to combine multi-task neural activations. Uhrig et al. [41] learn semantic and instance segmentations under a classification setting. Multi-task deep learning has also been used for geometry and regression tasks. [15] show how to learn semantic segmentation, depth and surface normals. PoseNet [25] is a model which learns camera position and orientation. UberNet [27] learns a number of different regression and classification tasks under a single architecture. In this work we are the first to propose a method for jointly learning depth regression, semantic and instance segmentation. Like the model of [15], our model learns both semantic and geometry representations, which is important for scene understanding. However, our model learns the much harder task of instance segmentation which requires knowledge of both semantics and geometry. This is because our model must determine the class and spatial relationship for each pixel in each object for instance segmentation.



Task Weights Class	Task Weights Depth	Class IoU [%]	Depth Err. [px]
1.0	0.0	59.4	-
0.975	0.025	59.5	0.664
0.95	0.05	59.9	0.603
0.9	0.1	60.1	0.586
0.85	0.15	60.4	0.582
0.8	0.2	59.6	0.577
0.7	0.3	59.0	0.573
0.5	0.5	56.3	0.602
0.2	0.8	47.2	0.625
0.1	0.9	42.7	0.628
0.0	1.0	-	0.640
Learned weights with task uncertainty (this work, Section 3.2)		**62.7**	**0.533**

(a) Comparing loss weightings when learning **semantic classification and depth regression**



Task Weights Instance	Task Weights Depth	Instance Err. [px]	Depth Err. [px]
1.0	0.0	4.61	
0.75	0.25	4.52	0.692
0.5	0.5	4.30	0.655
0.4	0.6	4.14	0.641
0.3	0.7	4.04	0.615
0.2	0.8	3.83	0.607
0.1	0.9	3.91	0.600
0.05	0.95	4.27	0.607
0.025	0.975	4.31	0.624
0.0	1.0		0.640
Learned weights with task uncertainty (this work, Section 3.2)		**3.54**	**0.539**

(b) Comparing loss weightings when learning **instance regression and depth regression**

Figure 2: **Learning multiple tasks improves the model's representation and individual task performance**. These figures and tables illustrate the advantages of multi-task learning for (a) semantic classification and depth regression and (b) instance and depth regression. Performance of the model in individual tasks is seen at both edges of the plot where $w = 0$ and $w = 1$. For some balance of weightings between each task, we observe improved performance for both tasks. All models were trained with a learning rate of 0.01 with the respective weightings applied to the losses using the loss function in (1). Results are shown using the Tiny CityScapes validation dataset using a down-sampled resolution of 128×256.

More importantly, all previous methods which learn multiple tasks simultaneously use a naïve weighted sum of losses, where the loss weights are uniform, or crudely and manually tuned. In this work we propose a principled way of combining multiple loss functions to simultaneously learn multiple objectives using homoscedastic task uncertainty. We illustrate the importance of appropriately weighting each task in deep learning to achieve good performance and show that our method can learn to balance these weightings optimally.

3. Multi Task Learning with Homoscedastic Uncertainty

Multi-task learning concerns the problem of optimising a model with respect to multiple objectives. It is prevalent in many deep learning problems. The naive approach to combining multi objective losses would be to simply perform a weighted linear sum of the losses for each individual task:

$$L_{total} = \sum_i w_i L_i. \tag{1}$$

This is the dominant approach used by prior work [39, 38, 30, 41], for example for dense prediction tasks [27], for scene understanding tasks [15] and for rotation (in quaternions) and translation (in meters) for camera pose [25]. However, there are a number of issues with this method. Namely, model performance is extremely sensitive to weight selection, w_i, as illustrated in Figure 2. These weight hyper-parameters are expensive to tune, often taking many days for each trial. Therefore, it is desirable to find a more convenient approach which is able to learn the optimal weights.

More concretely, let us consider a network which learns to predict pixel-wise depth and semantic class from an input image. In Figure 2 the two boundaries of each plot show models trained on individual tasks, with the curves showing

performance for varying weights w_i for each task. We observe that at some optimal weighting, the joint network performs better than separate networks trained on each task individually (performance of the model in individual tasks is seen at both edges of the plot: $w = 0$ and $w = 1$). At nearby values to the optimal weight the network performs worse on one of the tasks. However, searching for these optimal weightings is expensive and increasingly difficult with large models with numerous tasks. Figure 2 also shows a similar result for two regression tasks; instance segmentation and depth regression. We next show how to learn optimal task weightings using ideas from probabilistic modelling.

3.1. Homoscedastic uncertainty as task-dependent uncertainty

In Bayesian modelling, there are two main types of uncertainty one can model [24].

- *Epistemic uncertainty* is uncertainty in the model, which captures what our model does not know due to lack of training data. It can be explained away with increased training data.
- *Aleatoric uncertainty* captures our uncertainty with respect to information which our data cannot explain. Aleatoric uncertainty can be explained away with the ability to observe all explanatory variables with increasing precision.

Aleatoric uncertainty can again be divided into two sub-categories.

- *Data-dependent* or *Heteroscedastic* uncertainty is aleatoric uncertainty which depends on the input data and is predicted as a model output.
- *Task-dependent* or *Homoscedastic* uncertainty is aleatoric uncertainty which is not dependent on the input data. It is not a model output, rather it is a quantity which stays constant for all input data and varies between different tasks. It can therefore be described as task-dependent uncertainty.

In a multi-task setting, we show that the task uncertainty captures the relative confidence between tasks, reflecting the uncertainty inherent to the regression or classification task. It will also depend on the task's representation or unit of measure. We propose that we can use homoscedastic uncertainty as a basis for weighting losses in a multi-task learning problem.

3.2. Multi-task likelihoods

In this section we derive a multi-task loss function based on maximising the Gaussian likelihood with homoscedastic uncertainty. Let $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$ be the output of a neural network with weights $\mathbf{W}$ on input $\mathbf{x}$. We define the following probabilistic model. For regression tasks we define our likelihood as a Gaussian with mean given by the model output:

$$p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \mathcal{N}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma^2) \tag{2}$$

with an observation noise scalar σ. For classification we often squash the model output through a softmax function, and sample from the resulting probability vector:

$$p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \text{Softmax}(\mathbf{f}^{\mathbf{W}}(\mathbf{x})). \tag{3}$$

In the case of multiple model outputs, we often define the likelihood to factorise over the outputs, given some sufficient statistics. We define $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$ as our sufficient statistics, and obtain the following multi-task likelihood:

$$p(\mathbf{y}_1, ..., \mathbf{y}_K|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = p(\mathbf{y}_1|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))...p(\mathbf{y}_K|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \tag{4}$$

with model outputs $\mathbf{y}_1, ..., \mathbf{y}_K$ (such as semantic segmentation, depth regression, etc).

In *maximum likelihood* inference, we maximise the log likelihood of the model. In regression, for example, the log likelihood can be written as

$$\log p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \propto -\frac{1}{2\sigma^2}||\mathbf{y} - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 - \log \sigma \tag{5}$$

for a Gaussian likelihood (or similarly for a Laplace likelihood) with σ the model's observation noise parameter – capturing how much noise we have in the outputs. We then maximise the log likelihood with respect to the model parameters $\mathbf{W}$ and observation noise parameter σ.

Let us now assume that our model output is composed of two vectors $\mathbf{y}_1$ and $\mathbf{y}_2$, each following a Gaussian distribution:

$$\begin{aligned} p(\mathbf{y}_1, \mathbf{y}_2|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) &= p(\mathbf{y}_1|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \cdot p(\mathbf{y}_2|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \\ &= \mathcal{N}(\mathbf{y}_1; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_1^2) \cdot \mathcal{N}(\mathbf{y}_2; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_2^2). \end{aligned} \tag{6}$$

This leads to the *minimisation* objective, $\mathcal{L}(\mathbf{W}, \sigma_1, \sigma_2)$, (our loss) for our multi-output model:

$$\begin{aligned} &= -\log p(\mathbf{y}_1, \mathbf{y}_2|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \\ &\propto \frac{1}{2\sigma_1^2}||\mathbf{y}_1 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \frac{1}{2\sigma_2^2}||\mathbf{y}_2 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \log \sigma_1\sigma_2 \\ &= \frac{1}{2\sigma_1^2}\mathcal{L}_1(\mathbf{W}) + \frac{1}{2\sigma_2^2}\mathcal{L}_2(\mathbf{W}) + \log \sigma_1\sigma_2 \end{aligned} \tag{7}$$

Where we wrote $\mathcal{L}_1(\mathbf{W}) = ||\mathbf{y}_1 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2$ for the loss of the first output variable, and similarly for $\mathcal{L}_2(\mathbf{W})$.

We interpret minimising this last objective with respect to σ_1 and σ_2 as learning the relative weight of the losses

$\mathcal{L}_1(\mathbf{W})$ and $\mathcal{L}_2(\mathbf{W})$ adaptively, based on the data. As σ_1 – the noise parameter for the variable $\mathbf{y}_1$ – increases, we have that the weight of $\mathcal{L}_1(\mathbf{W})$ decreases. On the other hand, as the noise decreases, we have that the weight of the respective objective increases. The noise is discouraged from increasing too much (effectively ignoring the data) by the last term in the objective, which acts as a regulariser for the noise terms.

This construction can be trivially extended to multiple regression outputs. However, the extension to classification likelihoods is more interesting. We adapt the classification likelihood to squash a *scaled* version of the model output through a softmax function:

$$p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma) = \text{Softmax}(\frac{1}{\sigma^2}\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \tag{8}$$

with a positive scalar σ. This can be interpreted as a Boltzmann distribution (also called Gibbs distribution) where the input is scaled by σ^2 (often referred to as *temperature*). This scalar is either fixed or can be learnt, where the parameter's magnitude determines how 'uniform' (flat) the discrete distribution is. This relates to its uncertainty, as measured in entropy. The log likelihood for this output can then be written as

$$\begin{aligned}\log p(\mathbf{y} = c|\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma) &= \frac{1}{\sigma^2} f_c^{\mathbf{W}}(\mathbf{x}) \\ &\quad - \log \sum_{c'} \exp\left(\frac{1}{\sigma^2} f_{c'}^{\mathbf{W}}(\mathbf{x})\right)\end{aligned} \tag{9}$$

with $f_c^{\mathbf{W}}(\mathbf{x})$ the c'th element of the vector $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$.

Next, assume that a model's multiple outputs are composed of a continuous output $\mathbf{y}_1$ and a discrete output $\mathbf{y}_2$, modelled with a Gaussian likelihood and a softmax likelihood, respectively. Like before, the joint loss, $\mathcal{L}(\mathbf{W}, \sigma_1, \sigma_2)$, is given as:

$$\begin{aligned}
&= -\log p(\mathbf{y}_1, \mathbf{y}_2 = c|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \\
&= -\log \mathcal{N}(\mathbf{y}_1; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_1^2) \cdot \text{Softmax}(\mathbf{y}_2 = c; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_2) \\
&= \frac{1}{2\sigma_1^2}||\mathbf{y}_1 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \log \sigma_1 - \log p(\mathbf{y}_2 = c|\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_2) \\
&= \frac{1}{2\sigma_1^2}\mathcal{L}_1(\mathbf{W}) + \frac{1}{\sigma_2^2}\mathcal{L}_2(\mathbf{W}) + \log \sigma_1 \\
&\qquad + \log \frac{\sum_{c'} \exp\left(\frac{1}{\sigma_2^2} f_{c'}^{\mathbf{W}}(\mathbf{x})\right)}{\left(\sum_{c'} \exp\left(f_{c'}^{\mathbf{W}}(\mathbf{x})\right)\right)^{\frac{1}{\sigma_2^2}}} \\
&\approx \frac{1}{2\sigma_1^2}\mathcal{L}_1(\mathbf{W}) + \frac{1}{\sigma_2^2}\mathcal{L}_2(\mathbf{W}) + \log \sigma_1 + \log \sigma_2,
\end{aligned} \tag{10}$$

where again we write $\mathcal{L}_1(\mathbf{W}) = ||\mathbf{y}_1 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2$ for the Euclidean loss of $\mathbf{y}_1$, write $\mathcal{L}_2(\mathbf{W}) = -\log \text{Softmax}(\mathbf{y}_2, \mathbf{f}^{\mathbf{W}}(\mathbf{x}))$ for the cross entropy loss of $\mathbf{y}_2$ (with $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$ not scaled), and optimise with respect to $\mathbf{W}$ as well as σ_1, σ_2. In the last transition we introduced the explicit simplifying assumption $\frac{1}{\sigma_2} \sum_{c'} \exp\left(\frac{1}{\sigma_2^2} f_{c'}^{\mathbf{W}}(\mathbf{x})\right) \approx \left(\sum_{c'} \exp\left(f_{c'}^{\mathbf{W}}(\mathbf{x})\right)\right)^{\frac{1}{\sigma_2^2}}$ which becomes an equality when $\sigma_2 \to 1$. This has the advantage of simplifying the optimisation objective, as well as empirically improving results.

This last objective can be seen as learning the relative weights of the losses for each output. Large scale values σ_2 will decrease the contribution of $\mathcal{L}_2(\mathbf{W})$, whereas small scale σ_2 will increase its contribution. The scale is regulated by the last term in the equation. The objective is penalised when setting σ_2 too large.

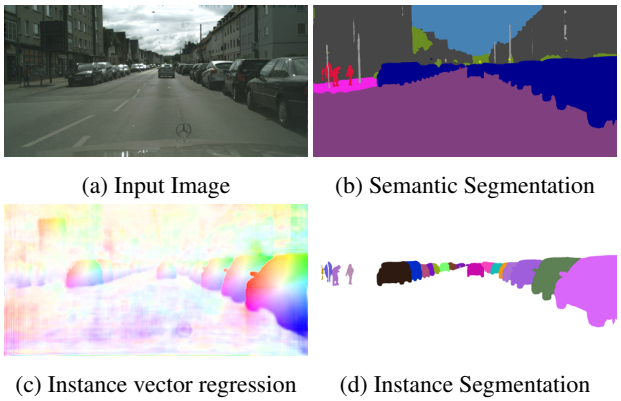
This construction can be trivially extended to arbitrary combinations of discrete and continuous loss functions, allowing us to learn the relative weights of each loss in a principled and well-founded way. This loss is smoothly differentiable, and is well formed such that the task weights will not converge to zero. In contrast, directly learning the weights using a simple linear sum of losses (1) would result in weights which quickly converge to zero. In the following sections we introduce our experimental model and present empirical results.

In practice, we train the network to predict the log variance, $s := \log \sigma^2$. This is because it is more numerically stable than regressing the variance, σ^2, as the loss avoids any division by zero. The exponential mapping also allows us to regress unconstrained scalar values, where $\exp(-s)$ is resolved to the positive domain giving valid values for variance.

4. Scene Understanding Model

To understand semantics and geometry we first propose an architecture which can learn regression and classification outputs, at a pixel level. Our architecture is a deep convolutional encoder decoder network [3]. Our model consists of a number of convolutional encoders which produce a shared representation, followed by a corresponding number of task-specific convolutional decoders. A high level summary is shown in Figure 1.

The purpose of the encoder is to learn a deep mapping to produce rich, contextual features, using domain knowledge from a number of related tasks. Our encoder is based on DeepLabV3 [10], which is a state of the art semantic segmentation framework. We use ResNet101 [20] as the base feature encoder, followed by an Atrous Spatial Pyramid Pooling (ASPP) module [10] to increase contextual awareness. We apply dilated convolutions in this encoder, such that the resulting feature map is sub-sampled by a factor of



(a) Input Image (b) Semantic Segmentation

(c) Instance vector regression (d) Instance Segmentation

Figure 3: **Instance centroid regression method.** For each pixel, we regress a vector pointing to the instance's centroid. The loss is only computed over pixels which are from instances. We visualise (c) by representing colour as the orientation of the instance vector, and intensity as the magnitude of the vector.

8 compared to the input image dimensions.

We then split the network into separate decoders (with separate weights) for each task. The purpose of the decoder is to learn a mapping from the shared features to an output. Each decoder consists of a 3×3 convolutional layer with output feature size 256, followed by a 1×1 layer regressing the task's output. Further architectural details are described in Appendix A.

Semantic Segmentation. We use the cross-entropy loss to learn pixel-wise class probabilities, averaging the loss over the pixels with semantic labels in each mini-batch.

Instance Segmentation. An intuitive method for defining which instance a pixel belongs to is an association to the instance's centroid. We use a regression approach for instance segmentation [29]. This approach is inspired by [28] which identifies instances using Hough votes from object parts. In this work we extend this idea by using votes from individual pixels using deep learning. We learn an instance vector, $\hat{x}_n$, for each pixel coordinate, c_n, which points to the centroid of the pixel's instance, i_n, such that $i_n = \hat{x}_n + c_n$. We train this regression with an L_1 loss using ground truth labels x_n, averaged over all labelled pixels, N_I, in a mini-batch: $\mathcal{L}_{Instance} = \frac{1}{|N_I|} \sum_{N_I} \|x_n - \hat{x}_n\|_1$.

Figure 3 details the representation we use for instance segmentation. Figure 3(a) shows the input image and a mask of the pixels which are of an instance class (at test time inferred from the predicted semantic segmentation). Figure 3(b) and Figure 3(c) show the ground truth and predicted instance vectors for both x and y coordinates. We then cluster these votes using OPTICS [2], resulting in the predicted instance segmentation output in Figure 3(d).

One of the most difficult cases for instance segmentation algorithms to handle is when the instance mask is split due to occlusion. Figure 4 shows that our method can handle these situations, by allowing pixels to vote for their instance centroid with geometry. Methods which rely on watershed approaches [4], or instance edge identification approaches fail in these scenarios.

(a) Input Image (b) Instance Segmentation

Figure 4: This example shows two cars which are occluded by trees and lampposts, making the instance segmentation challenging. Our instance segmentation method can handle occlusions effectively. We can correctly handle segmentation masks which are split by occlusion, yet part of the same instance, by incorporating semantics and geometry.

To obtain segmentations for each instance, we now need to estimate the instance centres, $\hat{i}_n$. We propose to consider the estimated instance vectors, $\hat{x}_n$, as votes in a Hough parameter space and use a clustering algorithm to identify these instance centres. OPTICS [2], is an efficient density based clustering algorithm. It is able to identify an unknown number of multi-scale clusters with varying density from a given set of samples. We chose OPICS for two reasons. Crucially, it does not assume knowledge of the number of clusters like algorithms such as k-means [33]. Secondly, it does not assume a canonical instance size or density like discretised binning approaches [12]. Using OPTICS, we cluster the points $c_n + \hat{x}_n$ into a number of estimated instances, $\hat{i}$. We can then assign each pixel, p_n to the instance closest to its estimated instance vector, $c_n + \hat{x}_n$.

Depth Regression. We train with supervised labels using pixel-wise metric inverse depth using a L_1 loss function: $\mathcal{L}_{Depth} = \frac{1}{|N_D|} \sum_{N_D} \left\| d_n - \hat{d}_n \right\|_1$. Our architecture estimates inverse depth, $\hat{d}_n$, because it can represent points at infinite distance (such as sky). We can obtain inverse depth labels, d_n, from a RGBD sensor or stereo imagery. Pixels which do not have an inverse depth label are ignored in the loss.

5. Experiments

We demonstrate the efficacy of our method on CityScapes [13], a large dataset for road scene understanding. It comprises of stereo imagery, from automotive grade stereo cameras with a $22cm$ baseline, labelled with instance and semantic segmentations from 20 classes. Depth images are also provided, labelled using SGM [22], which we treat as pseudo ground truth. Additionally, we assign zero inverse depth to pixels labelled as sky. The dataset was col-

Loss	Task Weights Seg.	Inst.	Depth	Segmentation IoU [%]	Instance Mean Error [px]	Inverse Depth Mean Error [px]
Segmentation only	1	0	0	59.4%	-	-
Instance only	0	1	0	-	4.61	-
Depth only	0	0	1	-	-	0.640
Unweighted sum of losses	0.333	0.333	0.333	50.1%	3.79	0.592
Approx. optimal weights	0.89	0.01	0.1	62.8%	3.61	0.549
2 task uncertainty weighting	✓	✓		61.0%	**3.42**	-
2 task uncertainty weighting	✓		✓	62.7%	-	0.533
2 task uncertainty weighting		✓	✓	-	3.54	0.539
3 task uncertainty weighting	✓	✓	✓	**63.4%**	3.50	**0.522**

Table 1: Quantitative improvement when learning semantic segmentation, instance segmentation and depth with our multi-task loss. Experiments were conducted on the Tiny CityScapes dataset (sub-sampled to a resolution of 128×256). Results are shown from the validation set. We observe an improvement in performance when training with our multi-task loss, over both single-task models and weighted losses. Additionally, we observe an improvement when training on all three tasks ($3 \times \checkmark$) using our multi-task loss, compared with all pairs of tasks alone (denoted by $2 \times \checkmark$). This shows that our loss function can automatically learn a better performing weighting between the tasks than the baselines.

lected from a number of cities in fine weather and consists of 2,975 training and 500 validation images at 2048×1024 resolution. 1,525 images are withheld for testing on an online evaluation server.

Further training details, and optimisation hyperparameters, are provided in Appendix A.

5.1. Model Analysis

In Table 1 we compare individual models to multi-task learning models using a naïve weighted loss or the task uncertainty weighting we propose in this paper. To reduce the computational burden, we train each model at a reduced resolution of 128×256 pixels, over $50,000$ iterations. When we downsample the data by a factor of four, we also need to scale the disparity labels accordingly. Table 1 clearly illustrates the benefit of multi-task learning, which obtains significantly better performing results than individual task models. For example, using our method we improve classification results from 59.4% to 63.4%.

We also compare to a number of naïve multi-task losses. We compare weighting each task equally and using approximately optimal weights. Using a uniform weighting results in poor performance, in some cases not even improving on the results from the single task model. Obtaining approximately optimal weights is difficult with increasing number of tasks as it requires an expensive grid search over parameters. However, even these weights perform worse compared with our proposed method. Figure 2 shows that using task uncertainty weights can even perform better compared to optimal weights found through fine-grained grid search. We believe that this is due to two reasons. First, grid search is restricted in accuracy by the resolution of the search. Second, optimising the task weights using a homoscedastic noise term allows for the weights to be dynamic during training. In general, we observe that the uncertainty term decreases during training which improves the optimisation process.

In Appendix B we find that our task-uncertainty loss is robust to the initialisation chosen for the parameters. These quickly converge to a similar optima in a few hundred training iterations. We also find the resulting task weightings varies throughout the course of training. For our final model (in Table 2), at the end of training, the losses are weighted with the ratio $43 : 1 : 0.16$ for semantic segmentation, depth regression and instance segmentation, respectively.

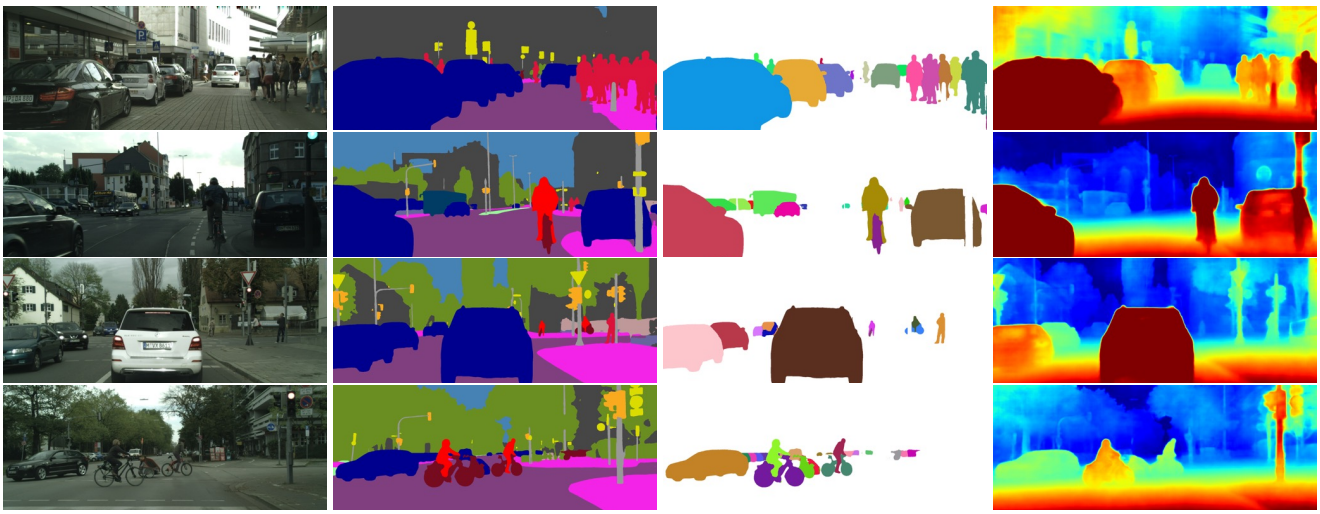
Finally, we benchmark our model using the full-size CityScapes dataset. In Table 2 we compare to a number of other state of the art methods in all three tasks. Our method is the first model which completes all three tasks with a single model. We compare favourably with other approaches, outperforming many which use comparable training data and inference tools. Figure 5 shows some qualitative examples of our model.

6. Conclusions

We have shown that correctly weighting loss terms is of paramount importance for multi-task learning problems. We demonstrated that homoscedastic (task) uncertainty is an effective way to weight losses. We derived a principled loss function which can learn a relative weighting automatically from the data and is robust to the weight initialization. We showed that this can improve performance for scene understanding tasks with a unified architecture for seman-

Method	Semantic Segmentation				Instance Segmentation				Monocular Disparity Estimation	
	IoU class	iIoU class	IoU cat	iIoU cat	AP	AP 50%	AP 100m	AP 50m	Mean Error [px]	RMS Error [px]
Semantic segmentation, instance segmentation and depth regression methods (this work)										
Multi-Task Learning	78.5	57.4	89.9	77.7	21.6	39.0	35.0	37.0	2.92	5.88
Semantic segmentation and instance segmentation methods										
Uhrig et al. [41]	64.3	41.6	85.9	73.9	8.9	21.1	15.3	16.7	-	-
Instance segmentation only methods										
Mask R-CNN [19]	-	-	-	-	26.2	49.9	37.6	40.1	-	-
Deep Watershed [4]	-	-	-	-	19.4	35.3	31.4	36.8	-	-
R-CNN + MCG [13]	-	-	-	-	4.6	12.9	7.7	10.3	-	-
Semantic segmentation only methods										
DeepLab V3 [10]	81.3	60.9	91.6	81.7	-	-	-	-	-	-
PSPNet [44]	81.2	59.6	91.2	79.2	-	-	-	-	-	-
Adelaide [31]	71.6	51.7	87.3	74.1	-	-	-	-	-	-

Table 2: **CityScapes Benchmark [13].** We show results from the test dataset using the full resolution of 1024×2048 pixels. For the full leaderboard, please see www.cityscapes-dataset.com/benchmarks. The disparity (inverse depth) metrics were computed against the CityScapes depth maps, which are sparse and computed using SGM stereo [21]. Note, these comparisons are not entirely fair, as many methods use ensembles of different training datasets. Our method is the first to address all three tasks with a single model.



(a) Input image (b) Segmentation output (c) Instance output (d) Depth output

Figure 5: **Qualitative results for multi-task learning of geometry and semantics for road scene understanding**. Results are shown on test images from the CityScapes dataset using our multi-task approach with a single network trained on all tasks. We observe that multi-task learning improves the smoothness and accuracy for depth perception because it learns a representation that uses cues from other tasks, such as segmentation (and vice versa).

tic segmentation, instance segmentation and per-pixel depth regression. We demonstrated modelling task-dependent homoscedastic uncertainty improves the model's representation and each task's performance when compared to separate models trained on each task individually.

There are many interesting questions left unanswered. Firstly, our results show that there is usually not a single optimal weighting for all tasks. Therefore, what is the optimal weighting? Is multitask learning is an ill-posed optimisation problem without a single higher-level goal?

A second interesting question is where the optimal location is for splitting the shared encoder network into separate decoders for each task? And, what network depth is best for the shared multi-task representation?

Finally, why do the semantics and depth tasks outperform the semantics and instance tasks results in Table 1? Clearly the three tasks explored in this paper are complimentary and useful for learning a rich representation about the scene. It would be beneficial to be able to quantify the relationship between tasks and how useful they would be for multitask representation learning.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015. 2

[2] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod Record*, volume 28, pages 49–60. ACM, 1999. 6

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 1, 5

[4] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. *arXiv preprint arXiv:1611.08303*, 2016. 1, 6, 8

[5] J. Baxter et al. A model of inductive bias learning. *J. Artif. Intell. Res.(JAIR)*, 12(149-198):3, 2000. 2

[6] S. R. Bulò, L. Porzi, and P. Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. *arXiv preprint arXiv:1712.02616*, 2017.

[7] R. Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998. 1, 2

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 1

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[10] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 5, 8, 11

[11] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008. 1, 2

[12] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002. 6

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 6, 8

[14] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 1

[15] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015. 1, 2, 3

[16] R. Garg and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. *Computer Vision–ECCV 2016*, pages 740–756, 2016. 1

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 580–587, 2014. 1

[18] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 447–456. IEEE, 2014. 1

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017. 8

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 5, 11

[21] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 807–814. IEEE, 2005. 8

[22] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2008. 6

[23] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7304–7308. IEEE, 2013. 1

[24] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017. 4

[25] A. Kendall, M. Grimes, and R. Cipolla. Convolutional networks for real-time 6-dof camera relocalization. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 2, 3

[26] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835, 2017. 2

[27] I. Kokkinos. Ubernet: Training auniversal'convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *arXiv preprint arXiv:1609.02132*, 2016. 1, 2, 3

[28] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision (IJCV)*, 77(1-3):259–289, 2008. 6

[29] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan. Proposal-free network for instance-level object segmentation. *arXiv preprint arXiv:1509.02636*, 2015. 6

[30] Y. Liao, S. Kodagoda, Y. Wang, L. Shi, and Y. Liu. Understand scene categories by objects: A semantic regularized scene classifier using convolutional neural networks. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2318–2325. IEEE, 2016. 2, 3

[31] G. Lin, C. Shen, I. Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv preprint arXiv:1504.01013*, 2015. 8

[32] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2015. 1

[33] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967. 6

[34] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016. 2

[35] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011. 2

[36] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1717–1724. IEEE, 2014. 2

[37] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015. 1

[38] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *International Conference on Learning Representations (ICLR)*, 2014. 1, 2, 3

[39] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun. Multinet: Real-time joint semantic reasoning for autonomous driving. *arXiv preprint arXiv:1612.07695*, 2016. 2, 3

[40] S. Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pages 640–646. MORGAN KAUFMANN PUBLISHERS, 1996. 2

[41] J. Uhrig, M. Cordts, U. Franke, and T. Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. *arXiv preprint arXiv:1604.05096*, 2016. 2, 3, 8

[42] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 1

[43] S. Zagoruyko and N. Komodakis. Wide residual networks. In E. R. H. Richard C. Wilson and W. A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.

[44] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *arXiv preprint arXiv:1612.01105*, 2016. 8

[45] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, 2015. 1

A. Model Architecture Details

We base our model on the recently introduced DeepLabV3 [10] segmentation architecture. We use ResNet101 [20] as our base feature encoder, with dilated convolutions, resulting in a feature map which is downsampled by a factor of 8 compared with the original input image. We then append dilated (atrous) convolutional ASPP module [10]. This module is designed to improve the contextual reasoning of the network. We use an ASPP module comprised of four parallel convolutional layers, with 256 output channels and dilation rates (1, 12, 24, 36), with kernel sizes (1^2, 3^2, 3^2, 3^2). Additionally, we also apply global average pooling to the encoded features, and convolve them to 256 dimensions with a 1×1 kernel. We apply batch normalisation to each of these layers and concatenate the resulting 1280 features together. This produces the shared representation between each task.

We then split the network, to decode this representation to a given task output. For each task, we construct a decoder consisting of two layers. First, we apply a 1×1 convolution, outputting 256 features, followed by batch normalisation and a non-linear activation. Finally, we convolve this output to the required dimensions for a given task. For classification, this will be equal to the number of semantic classes, otherwise the output will be 1 or 2 channels for depth or instance segmentation respectively. Finally, we apply bilinear upsampling to scale the output to the same resolution as the input.

The majority of the model's parameters and depth is in the feature encoding, with very little flexibility in each task decoder. This illustrates the attraction of multitask learning; most of the compute can be shared between each task to learn a better shared representation.

A.1. Optimisation

For all experiments, we use an initial learning rate of 2.5×10^{-3} and polynomial learning rate decay $(1 - \frac{iter}{max\ iter})^{0.9}$. We train using stochastic gradient descent, with Nesterov updates and momentum 0.9 and weight decay 10^4. We conduct all experiments in this paper using PyTorch.

For the experiments on the Tiny CityScapes validation dataset (using a down-sampled resolution of 128×256) we train over $50,000$ iterations, using 256×256 crops with batch size of 8 on a single NVIDIA 1080Ti GPU. We apply random horizontal flipping to the data.

For the full-scale CityScapes benchmark experiment, we train over $100,000$ iterations with a batch size of 16. We apply random horizontal flipping (with probability 0.5) and random scaling (selected from 0.7 - 2.0) to the data during training, before making a 512×512 crop. The training data is sampled uniformly, and is randomly shuffled for each epoch. Training takes five days on a single computer with four NVIDIA 1080Ti GPUs.

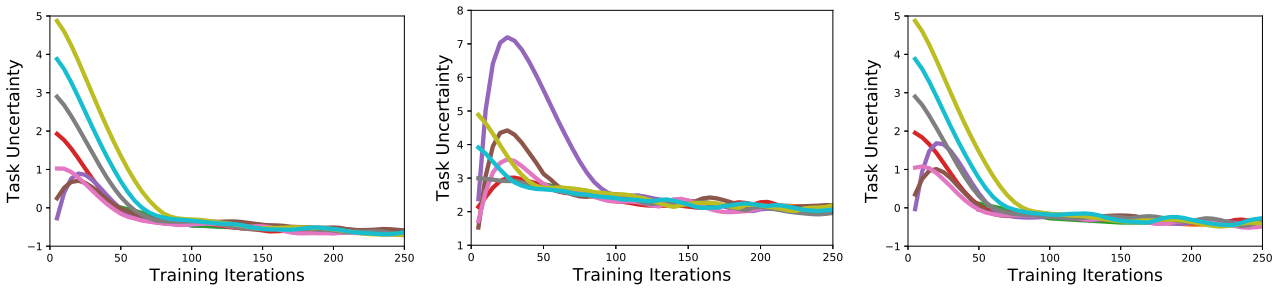
B. Further Analysis

This task uncertainty loss is also robust to the value we use to initialise the task uncertainty values. One of the attractive properties of our approach to weighting multi-task losses is that it is robust to the initialisation choice for the homoscedastic noise parameters. Figure 6 shows that for an array of initial choices of $\log \sigma^2$ from -2.0 to 5.0 the homoscedastic noise and task loss is able to converge to the same minima. Additionally, the homoscedastic noise terms converges after only 100 iterations, while the network requires $30,000+$ iterations to train. Therefore our model is robust to the choice of initial value for the weighting terms.

Figure 7 shows losses and uncertainty estimates for each task during training of the final model on the full-size CityScapes dataset. At a point 500 iterations into training, the model estimates task variance of 0.60, 62.5 and 13.5 for semantic segmentation, instance segmentation and depth regression, respectively. Becuase the losses are weighted by the inverse of the uncertainty estimates, this results in a task weighting ratio of approximately $23 : 0.22 : 1$ between semantics, instance and depth, respectively. At the conclusion of training, the three tasks have uncertainty estimates of 0.075, 3.25 and 20.4, which results in effective weighting between the tasks of $43: 0.16 : 1$. This shows how the task uncertainty estimates evolve over time, and the approximate final weightings the network learns. We observe they are far from uniform, as is often assumed in previous literature.

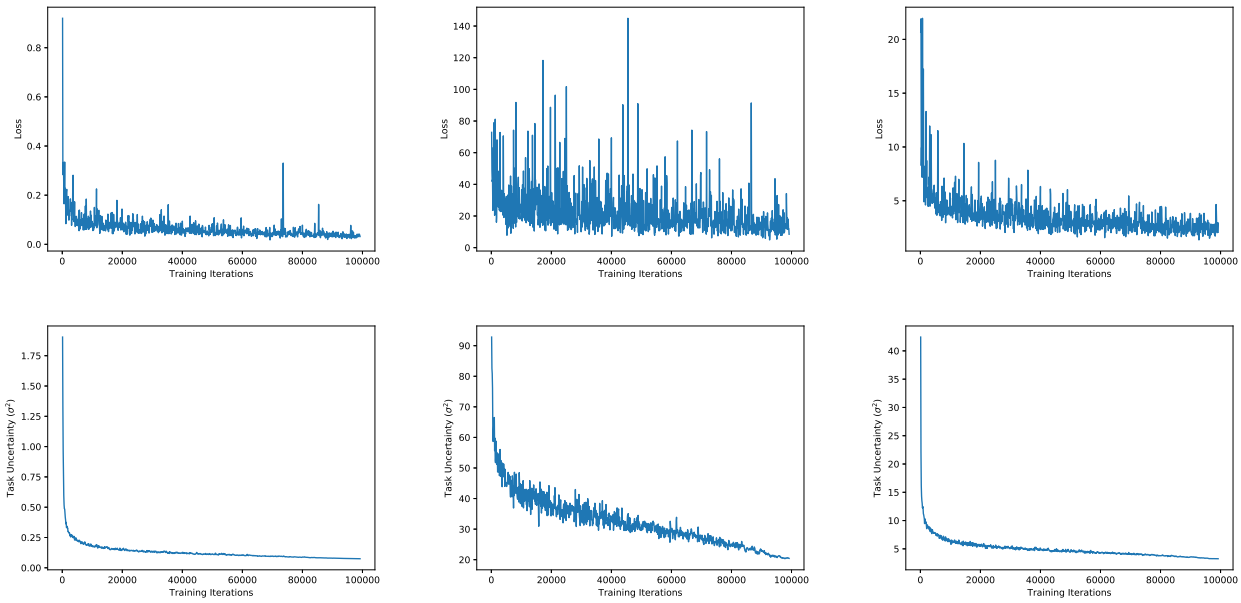
Interestingly, we observe that this loss allows the network to dynamically tune the weighting. Typically, the homoscedastic noise terms decrease in magnitude as training progresses. This makes sense, as during training the model becomes more effective at a task. Therefore the error, and uncertainty, will decrease. This has a side-effect of increasing the effective learning rate – because the overall uncertainty decreases, the weight for each task's loss increases. In our experiments we compensate for this by annealing the learning rate with a power law.

Finally, a comment on the model's failure modes. The model exhibits similar failure modes to state-of-the-art single-task models. For example, failure with objects out of the training distribution, occlusion or visually challenging situations. However, we also observe our multi-task model tends to fail with similar effect in all three modalities. Ie. an erroneous pixel's prediction in one task will often be highly correlated with error in another modality. Some examples can be seen in Figure 8.



(a) Semantic segmentation task (b) Instance segmentation task (c) Depth regression task

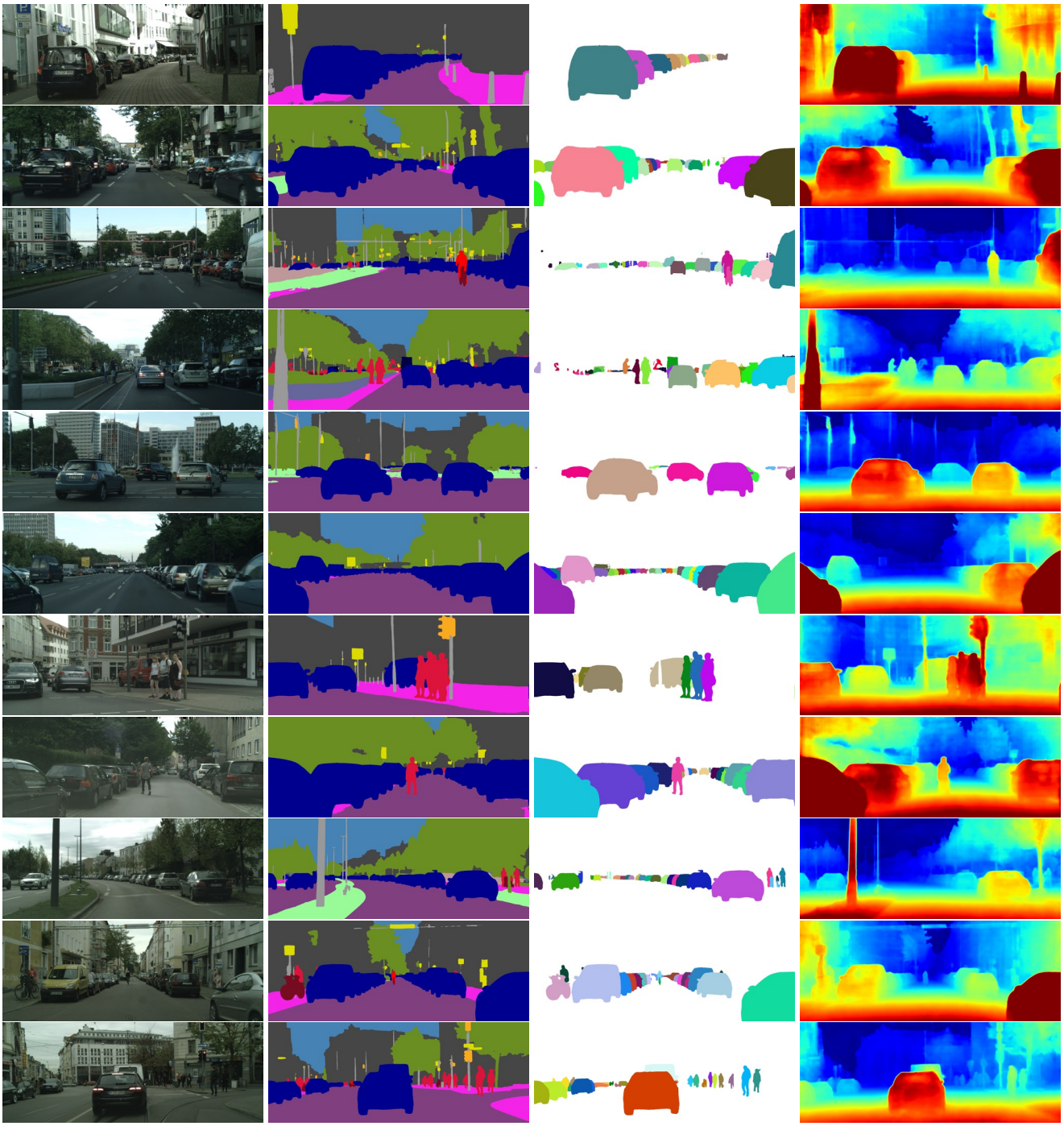
Figure 6: **Training plots showing convergence of homoscedastic noise and task loss** for an array of initialisation choices for the homoscedastic uncertainty terms for all three tasks. Each plot shows the the homoscedastic noise value optimises to the same solution from a variety of initialisations. Despite the network taking $10,000+$ iterations for the training loss to converge, the task uncertainty converges very rapidly after only 100 iterations.



(a) Semantic segmentation task (b) Instance segmentation task (c) Depth regression task

Figure 7: **Learning task uncertainty.** These training plots show the losses and task uncertainty estimates for each task during training. Results are shown for the final model, trained on the fullsize CityScapes dataset.

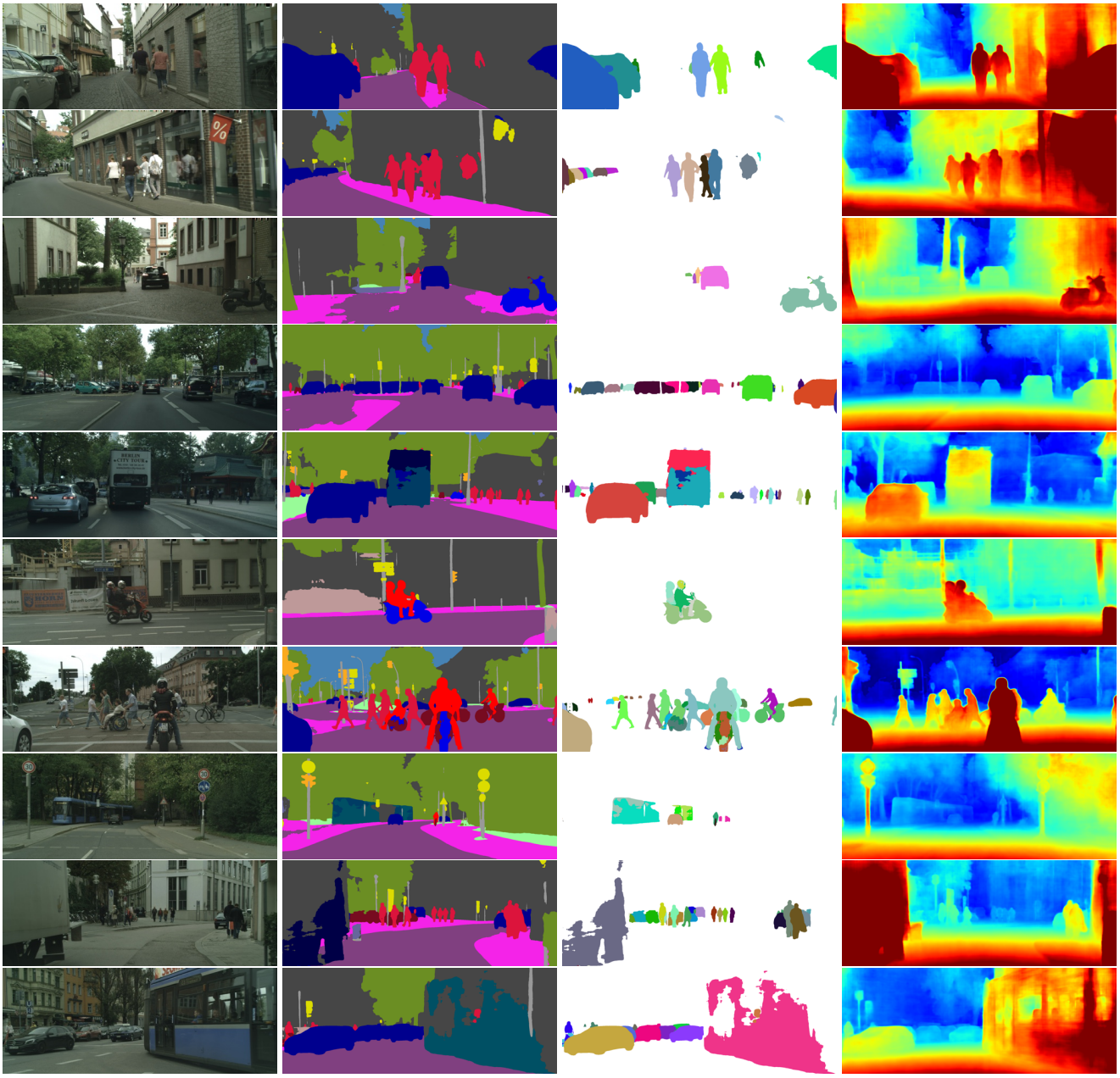
C. Further Qualitative Results



(a) Input image (b) Semantic segmentation (c) Instance segmentation (d) Depth regression

Figure 8: **More qualitative results on test images from the CityScapes dataset.**

D. Failure Examples



(a) Input image (b) Semantic segmentation (c) Instance segmentation (d) Depth regression

Figure 9: **Example where our model fails on the CityScapes test data.** The first two rows show examples of challenging visual effects such as reflection, which confuse the model. Rows three and four show the model incorrectly distinguishing between road and footpath. This is a common mistake, which we believe is due to a lack of contextual reasoning. Rows five, six and seven demonstrate incorrect classification of a rare class (bus, fence and motorbike, respectively). Finally, the last two rows show failure due to occlusion and where the object is too big for the model's receptive field. Additionally, we observe that failures are highly correlated between the modes, which makes sense as each output is conditioned on the same feature vector. For example, in the second row, the incorrect labelling of the reflection as a person causes the depth estimation to predict human geometry.