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

CVPR 2018

Authors: Alex Kendall, Yarin Gal, and Roberto Cipolla

Presented by: David Kolschowksy and Adit Bhagat

Overview

1. Background & Motivation
2. Introduction of Homoscedastic Uncertainty
3. Derivation of Learnable Weights
4. Experiment Overview
5. Results
6. Conclusion & Future Work

Background: Multi-Task Learning

- Multi-task learning has proven to be an effective tool in numerous deep learning applications
  - ➢ Learning multiple tasks at once can increase the performance of each task through the usage of a shared representation

- Classically, the multi-task learning loss is determined by computing a weighted linear sum of the task specific losses:

$$L_{\text{total}} = \sum_i w_i L_i$$

Motivation: Challenges with Traditional Multi-Task Formulation

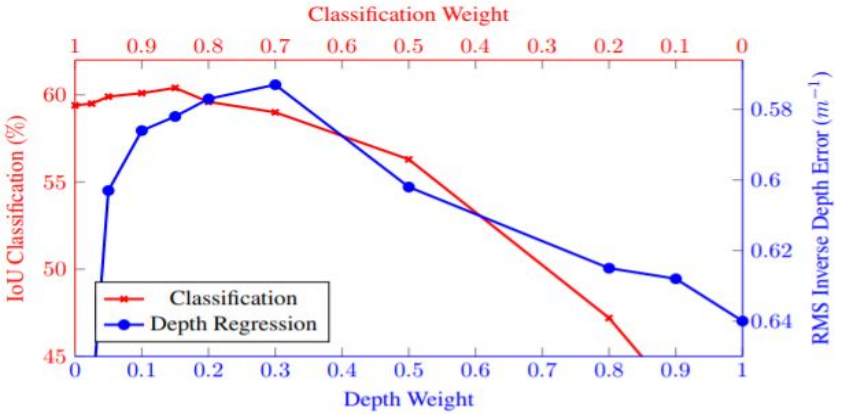
This formulation of the problem has several challenges:

1. Tuning task weights by hand is a difficult and expensive process

2. Model performance is extremely sensitive to weight selection, w_i

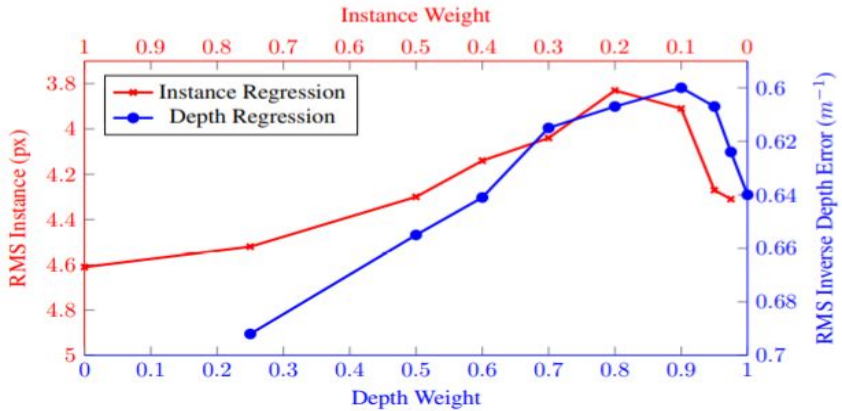
3. The optimal weighting of each task is dependent on the measurement scale (e.g. meters, centimeters, or millimeters), and ultimately the magnitude of the task's noise

Motivation: Multi-Task Learning Effectiveness and Weight Sensitivity



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**

Uncertainty in Bayesian Modelling

There are two types of uncertainty that can be modeled, **Epistemic** and **Aleatoric**.

- **Epistemic uncertainty** is uncertainty in the model itself and captures what the model does not know due to a lack of training data.
  - ➢ This type of uncertainty can be reduced by collecting more data.

- **Aleatoric uncertainty** is uncertainty with respect to information that the data cannot explain.
  - ➢ This type of uncertainty can be reduced by observing more explanatory variables to a higher precision.

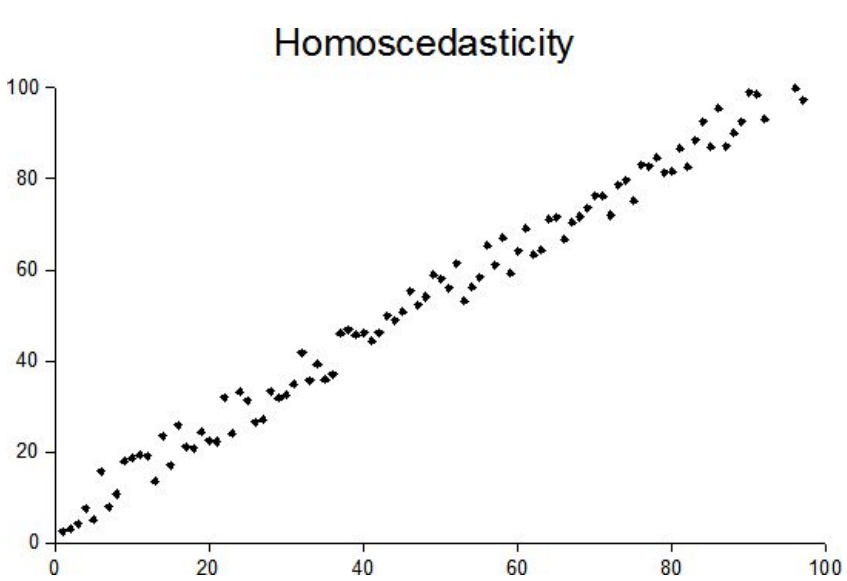
Further Expansion of Aleatoric Uncertainty

Aleatoric uncertainty can be further expanded into both **Homoscedastic** and **Heteroscedastic** uncertainty:

- **Heteroscedastic uncertainty** is uncertainty which depends on the input data itself and is predicted as model output. For this reason, it is also known as *data-dependent* uncertainty.

- **Homoscedastic uncertainty** is uncertainty which is independent of the input data. It is not a model output and is a constant quantity that varies between tasks. For this reason, it can also be called *task-dependent* uncertainty.

Homoscedastic vs. Heteroscedastic Uncertainty



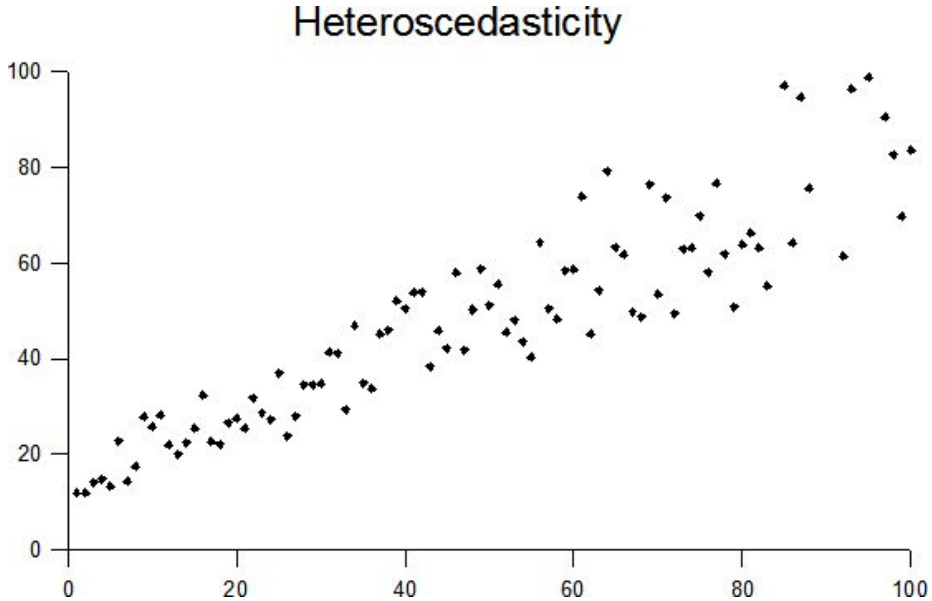


Image:Sreebot/Wikimedia Commons

Deriving Learnable Weights: Modeling the Multi-Task Problem

To see how we can go from Bayesian modelling to learnable weights let us consider first consider the Bayesian likelihood representation of a multitask model as:

$$p(\mathbf{y}_1, \ldots, \mathbf{y}_k | \mathbf{f}^{\mathbf{W}}(\mathbf{x})))$$

Given a sufficient statistic we can assume conditional independence of the tasks and factorize the above as:

$$p(\mathbf{y}_1 | \mathbf{f}^{\mathbf{W}}(\mathbf{x}))) \cdots p(\mathbf{y}_k | \mathbf{f}^{\mathbf{W}}(\mathbf{x})))$$

In this case we consider $\mathbf{f}^{\mathbf{W}}(\mathbf{x})$ to be the output of the network and a sufficient statistic.

Deriving Learnable Weights: Tasks as Distributions

Let us consider a two task model where task one is a regression task and task two is a classification task.

We can define the regression likelihood as:

$$p(\mathbf{y_1}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \mathcal{N}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_1^2)$$

Typically the classification likelihood would be written as:

$$p(\mathbf{y}_2|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \text{Softmax}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$

In order to accommodate homoscedastic uncertainty we can rescale the above as:

$$p(\mathbf{y_2}|\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_2) = \text{Softmax}(\frac{1}{\sigma_2^2}\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$

which can be viewed as a Boltzmann distribution where the input is scaled by σ_2^2

Deriving Learnable Weights: From Likelihoods to Classical Losses

We can now construct a new joint loss function which will be the minimization of the log-likelihood:

$$\mathcal{L}(\mathbf{W}, \sigma_1, \sigma_2) = -\log(\mathbf{y}_1, \mathbf{y}_2 = c|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$

Using our sufficient statistic to factorize the loss yields:

$$= -\log \mathcal{N}(\mathbf{y}_1; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_1^2) \cdot \text{Softmax}(\mathbf{y}_2 = c; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_2)$$

Plugging in the formulas for gaussian distribution and the scaled softmax yields:

$$= \frac{1}{2\sigma_1^2}||\mathbf{y}_1 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \log \sigma_1 - \frac{1}{\sigma_2^2}\mathbf{f}_c^{\mathbf{W}}(\mathbf{x}) + \log \sum_{c'} \exp\left(\frac{1}{\sigma_2^2}\mathbf{f}_{c'}^{\mathbf{W}}(\mathbf{x})\right)$$

Which finally yields:

$$\approx \frac{1}{2\sigma_1^2}\mathcal{L}_1(\mathbf{W}) + \frac{1}{\sigma_2^2}\mathcal{L}_2(\mathbf{W}) + \log \sigma_1 + \log \sigma_2$$

where $\mathcal{L}_1(\mathbf{W})$ is the classical regression loss and $\mathcal{L}_2(\mathbf{W})$ is the non-scaled cross-entropy loss.

Experiment Overview

- Goal: **Apply multi-task learning to visual scene understanding in computer vision, utilizing homoscedastic uncertainty to weigh losses**

- Used the CityScapes dataset for road scene understanding

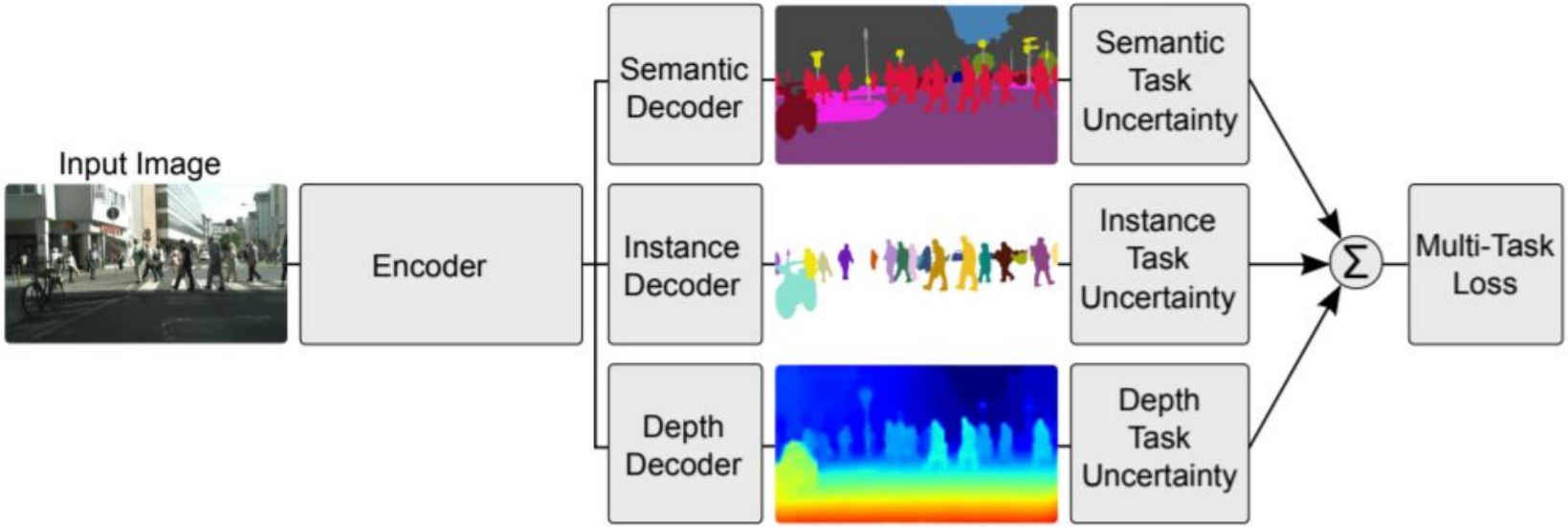
- Attempt to take advantage of a shared representation to jointly predict the following 3 tasks:
  - *Semantic Segmentation*
  - *Instance Segmentation*
  - *Depth Regression*

CityScapes Dataset

- 2975 training and 500 validation stereo images, with additional online evaluation server consisting of 1525 images
- Resolution of 2048 x 1024
- Automotive grade stereo cameras with a 22cm baseline
- Images taken from several different cities in fine weather conditions
- Tiny CityScapes dataset created by downsampling the original dataset to 128 x 256

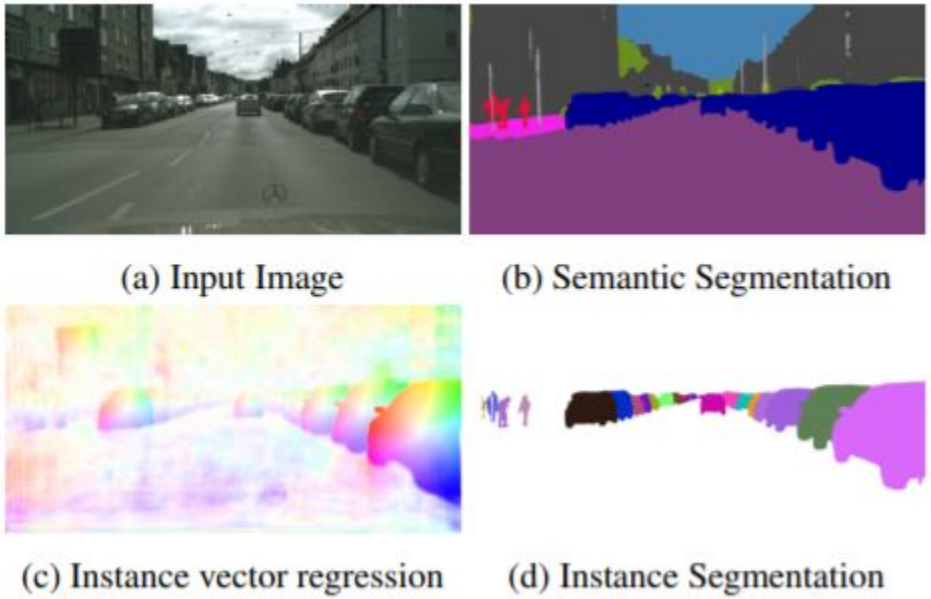
Model Setup

- Encoder-decoder network utilizing homoscedastic uncertainties
- Based on DeepLabV3, uses ResNet101 as feature encoder with Atrous Spatial Pyramid Pooling (ASPP)
- Individual decoders for each task



Semantic Segmentation

- Goal: **Classify objects at a pixel level**
- Classify pixels into one of 20 classes
- Model utilizes cross-entropy loss, with averaging in each mini-batch



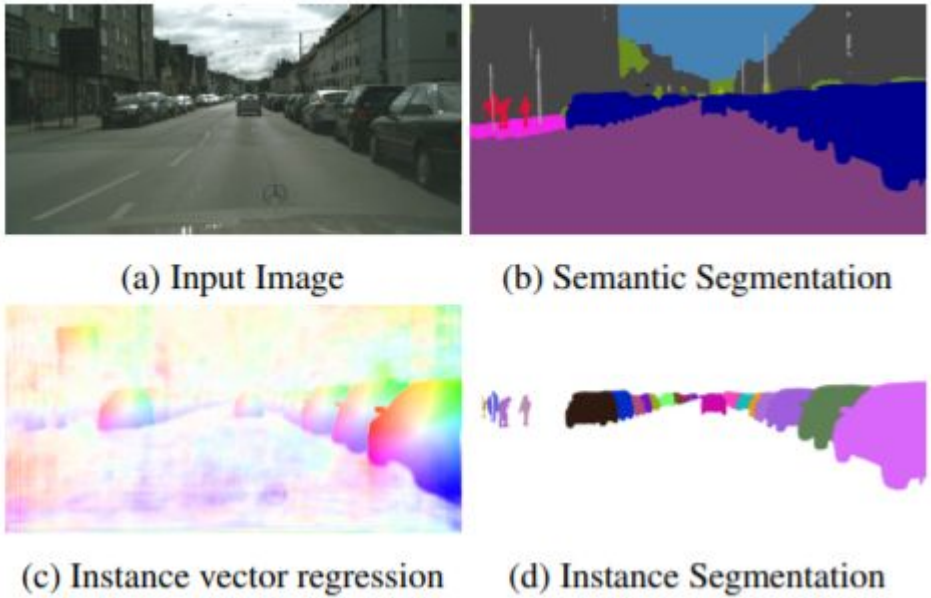
Instance Segmentation

- Goal: **Treat each instance of an object separately, and segment independent masks for each of them**

- Regression approach which learns vectors pointing to the centroid of each pixel instance

- Loss function used:

$$\mathcal{L}_{Instance} = \frac{1}{|N_1|} \sum_{N_1} \|x_n - \hat{x}_n\|_1$$



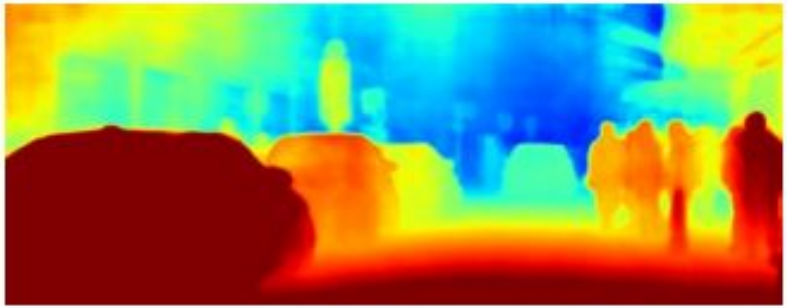
(a) Input Image (b) Semantic Segmentation

(c) Instance vector regression (d) Instance Segmentation

Depth Regression

- Goal: **Predict pixel-wise metric depth**

- Utilizes inverse depth in order to better represent points at infinite distance (e.g. the sky)



- Loss function used:

$$\mathcal{L}_{Depth} = \frac{1}{|N_D|} \sum_{N_D} \left\| d_n - \hat{d}_n \right\|_1$$

Tiny CityScapes Results

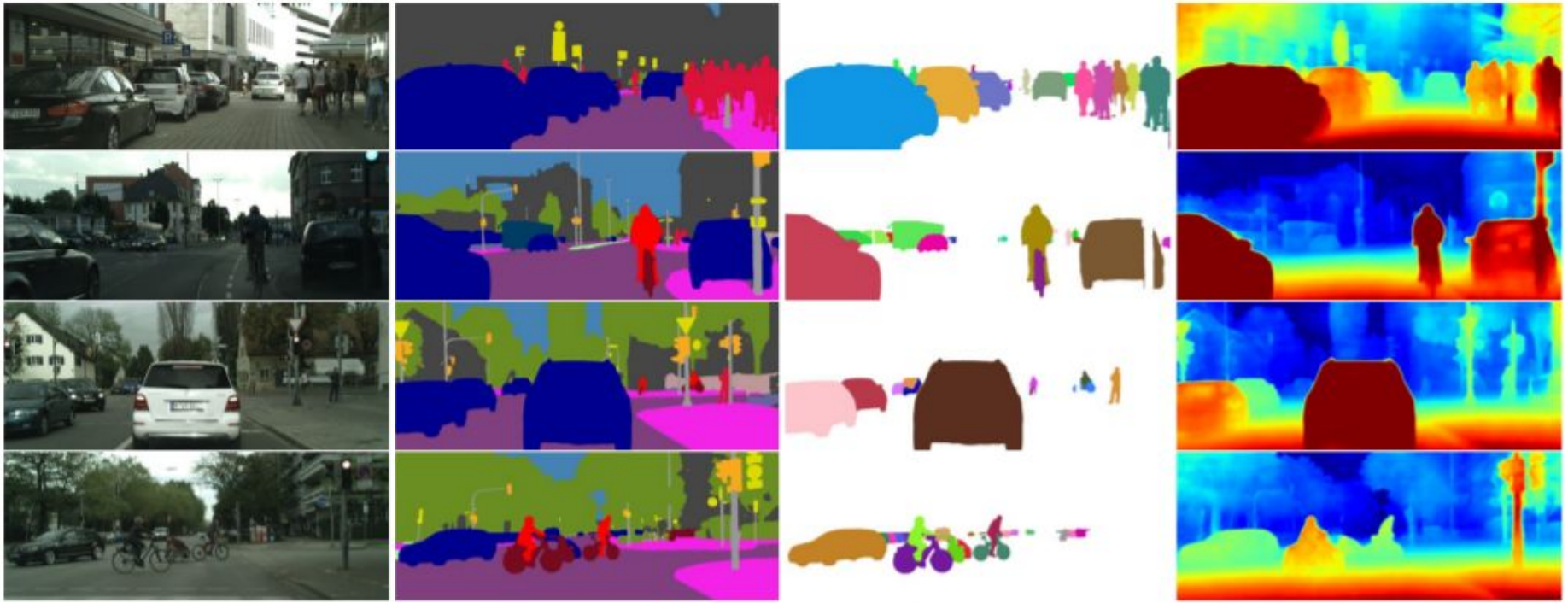
- Tested various MTL networks with different parameters on the Tiny CityScapes dataset
- Best results were achieved with 3-task uncertainty weighting, although 2-task uncertainty performed better on instance segmentation

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**

Quantitative CityScapes Results

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	-	-	-	-	-	-

Qualitative CityScapes Results



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

A Unified Representation

- Results clearly indicate that each individual task is able to benefit from the usage of a shared representation of the input image
- Model is able to gain a better understanding of scene geometry and semantics, which allows it to handle occlusion effectively
- Depth regression outputs are also very accurate and smooth, presumably due to the model utilizing cues from the segmentation tasks

(a) Input Image (b) Instance Segmentation

Conclusions

- The relative weighting of losses is an essential factor in determining the success of a multi-task model

- Homoscedastic uncertainty weighting has proven to be an extremely effective and efficient way to model multi-task learning problems

- Some comparisons in the final CityScapes results are not entirely fair due to the fact that many models use ensembles of different datasets for training, but the overall performance is still quite impressive

Future Work

There are many interesting questions that arise from the results of this work, such as:

- How exactly to determine the optimal loss weighting, when there is usually no single optimal weighting for a set of tasks.
- How does the joint learning of a specific set of tasks affect the model's performance on each task (compared to another set of tasks)?
- Would different network architectures be more beneficial in taking advantage of the shared representation (e.g. where to split from encoder to decoder portion)?
- Is there a way to quantify the relationship between multiple tasks and their joint performance in multi-task learning?

Additionally, it would be worth exploring alternatives to performing a weighted linear combination of losses.

- One example of this is **Multi-Task Learning as Multi-Objective Optimization**, which was presented earlier in class

Questions?

Thanks!