# DEEP LEARNING

TECHNICAL REPORT ON THE IMPLEMENTATION OF THE MULTI-TASK LEARNING METHOD MTAN LIU ET AL. (2019)

**Mohamed Salah Jebali**

**Matricola: 7078487**

**School of Engineering - Master of Engineering Degree in Artificial Intelligence**

**Università degli Studi di Firenze**

April 2023

## Contents

## Abstract

This short technical report aims to give a description of the experiments concerning the implementation of the MTAN multi-task learning method described in *End-to-End Multi-Task Learning with Attention* Liu et al. (2019).
The aim of this report is to give an overview of the experimental environment and setup, the definition of the experiment, a description of the architectural choices and to present the main results obtained.

***Keywords*** Deep Learning · Multi-Task learning · image segmentation · image depth estimation · attention · encoder-decoder · computer vision · SegNet

## 1 Introduction

**Introduction to the experiment** In this work I implemented the method explained in Liu et al. (2019) with a reduced version of **SegNet** (Badrinarayanan et al. (2017)), in order to respect the constraints of the computational resources available to me.
The aim of my work was to conduct the multi-task learning experiment on 2 tasks: **semantic segmentation** and **depth estimation**.
The experiment was conducted on the **CityScapes** dataset Cordts et al. (2016). The dataset contains 19 classes for pixel-wise semantic segmentation, but the experiment was performed for **7-class** version of the dataset because in the reference paper, the experiments were conducted in this mode and I wanted the results obtained to be comparable with theirs.

Most of the architectural choices remained faithful to those of the reference paper, in order to have at least the same setup and to ensure that the results obtained were comparable, while a few modifications were made to meet practical requirements concerning the computational resources available to me.
More details on the architectural choices will follow in the report.

**Implementation environment** The implementation was done in Python language in version 3.9.16 and with the main use of the Pytorch framework (Paszke et al. (2019)) in version 1.13.1.
The code was written with readability in mind and is provided with comments to make it as self-explanatory as possible.
Due to the limited computational resources I possessed, the experiments were carried out in a virtual environment provided by Google Colab (Bisong and Bisong (2019)) (paid version), using their standard NVIDIA T4 Tensor Core GPUs which have the main following technical characteristics:

- 16GB memory
- 320 Tensor Cores
- 2,560 CUDA cores

and they are optimized for deep learning, support TensorFlow, PyTorch, and MXNet. More details about it can be found here.

## 2 Methodology

### 2.1 Architecture Design

MTAN consists of two components:

- A single shared network
- K task-specific attention networks

According to the reference paper, the authors utilize the SegNet architecture as their encoder-decoder network for image-to-image dense pixel-level prediction, although the single shared network can take on any feed-forward network.
The authors specify that the task-specific network includes a series of attention modules, which are responsible for applying a soft attention mask to a specific layer of the shared network to learn task-specific features

To summarize, the shared network learns a condensed global feature pool across all tasks, whereas the attention masks can be viewed as feature selectors that operate on the shared global pool.

**SegNet** SegNet (Badrinarayanan et al. (2017)) is a convolutional neural network architecture designed for semantic segmentation of images. The network consists of an encoder network and a corresponding decoder network.
The **encoder** network is composed of a series of convolutional and max-pooling layers, which downsample the input image and extract features from it. The feature maps produced by each layer are then fed into the decoder network.
The **decoder** network is designed to upsample the encoded feature maps and produce a segmentation map that has the same resolution as the input image. This is achieved using a series of upsampling and convolutional layers, which gradually increase the spatial resolution of the feature maps while maintaining their semantic content.
To facilitate the upsampling process, SegNet uses **pooling indices** that are produced during the max-pooling operation in the encoder network. These indices are used to perform unpooling in the decoder network, which allows the network to recover the spatial information lost during the downsampling process.

However, due to limited computational resources, I implemented a reduced version of SegNet, called **SegNet-Basic**, that is a variants suggested by the authors of SegNet.
In my version, SegNet-Basic includes **five encoder layers and five decoder layers**, which still provide reasonable segmentation performance while requiring less computation than the full SegNet architecture.

**Task Specific Attention Module** The task specific attention module consists of a set of attention masks, which are learned using a soft attention mechanism. Each attention mask is applied to a particular layer of the shared network to obtain task-specific feature maps.
The task-specific attention module applies the following steps:

1. The output is first passed through a 1x1 convolutional block with batch normalization and ReLU activation.
2. This is followed by another 1x1 convolutional block with batch normalization and sigmoid activation, which generates a soft attention mask.
3. The attention mask is then multiplied with the output of the previous convolutional block element-wise, resulting in a task-specific feature representation.
4. Then, the result is passed through a 3x3 convolutional block with batch normalization and ReLU activation.
5. Finally, the feature representation is concatenated with the features from other task-specific attention modules and passed through the next set of shared network layers.

### 2.2 Training

In this subsection, I describe the implementation choices for the experiments. Whenever possible, I used the same values as in the reference paper to ensure maximum comparability.

- **Loss Weighting Methods:** I used two weighting methods, *equal weighting* and *Dynamic Weighting Average (DWA)*, as proposed by the authors of the paper. I chose to use only these two methods due to computational resource constraints. For DWA, I set the hyperparameter temperature T equal to 2 as suggested by the authors.
- **Epochs:** 100 instead of 200, due to computational constraints.
- **Optimizer:** I used the ADAM (Kingma and Ba (2014)) optimizer with a learning rate of $10^{-4}$ as in the reference paper, and added *weight decay* equal to $10^{-4}$ to prevent overfitting due to the reduced number of epochs compared to the original paper.
- **Learning Rate Scheduler:** StepLR scheduler was chosen with a step size of 25 and a gamma value of 0.5. This means that the learning rate of the optimizer will be reduced by a factor of 0.5 every 25 epochs. The rationale behind this choice is that it allows the model to converge faster during the initial phase of training, where the gradients are larger, and then gradually reduce the learning rate to fine-tune the model parameters. Again, due the less number of training epochs.
- **Batch Size:** 8 as in the reference paper.

**Weights Initialization** Regarding weight initialization, I used **He** (He et al. (2015)) initialization for the convolutional blocks with ReLU activation functions to take advantage of the non-linear properties of ReLU. This initialization method sets the initial weights for each layer by sampling from a Gaussian distribution with mean zero and variance equal to $\frac{2}{n}$, where n is the number of input neurons in the layer.
This choice of variance makes the signal propagate equally well across all layers, leading to a more stable training process and better overall performance.
Therefore, using the He initialization method for the convolutional blocks with ReLU activation functions is a well-justified choice for improving the training and convergence of the network.

While, for BatchNorm I used constant initialization to ensure that the mean and variance remain relatively constant throughout training and to prevent vanishing or exploding gradients.

### 2.3 Loss Functions

The general loss formula for multi-task learning with K tasks, input **X**, and task-specific labels $Y_i$ (with i from 1 to K) can be written as:

$$L_{total} = \sum_{i=1}^{K} \lambda_i L_i(X, Y_i) \tag{1}$$

where $L_i$ is the task-specific loss function for task i, and $\lambda_i$ is a weighting coefficient that balances the importance of each task in the overall loss.

**Semantic Segmentation** For this task, as suggested in the reference paper, I apply a pixel-wise cross-entropy loss for each predicted class label from a depth-softmax classifier:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}) + (1 - y_{ij})\log(1 - p_{ij}) \tag{2}$$

Where $N$ is the number of pixels, $M$ is the number of classes, $y_{ij}$ is the ground truth label for pixel $i, j$ (a binary indicator for whether the pixel belongs to class $j$), and $p_{ij}$ is the predicted probability for pixel $i, j$ belonging to class $j$.

**Depth Estimation** And, for this task, as suggested by the reference paper, I apply a $L_1$ norm comparing the predicted and ground-truth depth:

$$L_{L1} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y_i}| \tag{3}$$

where $N$ is the total number of pixels, $y_i$ is the ground truth depth value and $\hat{y_i}$ is the predicted depth value.

**Dynamic Weight Average** As mentioned before, in multi-task learning, the global loss is a weighted sum of the single task loss.
**Dynamic Weight Average (DWA)** is technique proposed by the authors to calculate those weights. It learns to average task weighting over time by considering the rate of change of loss for each task and since it only requires the numerical task loss, it is simple to implement.
The $\lambda_k$ weight for task k is then defined as:

$$\lambda_k(t) = \frac{K exp(w_k(t-1)/T)}{\sum_i exp(w_i(t-1)/T)} \tag{4}$$

where

$$w_k(t-1) = \frac{L_k(t-1)}{L_k(t-2)} \tag{5}$$

where $w_k(\dot{})$ calculates the relative descending rate in the range $(0, +\inf)$, $t$ is the iteration index and $T$ represents a temperature which controls the softness of task weighting.
With $T$ large enough we obtain $\lambda_i \approx 1$, hence **equal weights**.

## 2.4 Dataset

The Cityscapes (Cordts et al. (2016)) dataset is a popular benchmark for urban scene understanding, containing high-quality pixel-level annotations for a wide range of tasks including semantic segmentation, instance segmentation, and depth estimation. The dataset consists of high-resolution images of urban street scenes captured from a vehicle-mounted camera, covering a variety of weather conditions, illumination, and camera viewpoints.

For my experiments, I used a reduced version of the dataset consisting of 2975 training images and 500 testing images, each resized to 128 x 256 resolution, as the paper suggested.
Since in the reference paper the experiments were conducted in the 7-class version of the dataset, I did the same to compare the results:

- void
- flat
- construction
- object
- nature
- sky
- human
- vehicle

## 3 Results

In this section, I will present the results of experiments, the aim of which was to compare the results obtained from my implementation with the original one in the paper.
I will first give a quick description of the metrics used for the tasks, which are the same as those used in the paper.
Then I will present the quantitative results and finally the qualitative results.

### 3.1 Metrics

**Semantic Segmentation** For the segmentation task, I used two metrics: **mean intersection-over-union (mIoU)** and **pixel accuracy**.
The mIoU measures the overlap between the predicted segmentation mask and the ground truth mask. It is defined as the ratio between the intersection and the union of the two masks, averaged over all classes.
It can be reffered as *Jaccard Index* and its general formula is:

$$IoU = \frac{Target \bigcap Prediction}{Target \bigcup Prediction} \tag{6}$$

While, the formula for mIoU can be written as:

$$mIoU = \frac{1}{C} \sum_{i=1}^{C} \frac{TP_i}{TP_i + FP_i + FN_i}, \tag{7}$$

where C is the number of classes, $TP_i$ is the number of true positive pixels for class i, $FP_i$ is the number of false positive pixels for class i, and $FN_i$ is the number of false negative pixels for class i.

Pixel accuracy, on the other hand, measures the percentage of pixels that are correctly classified. It is calculated by dividing the number of correctly classified pixels by the total number of pixels in the image.

**Depth Estimation** For the depth estimation task, I used two metrics: **absolute error (Abs Err)** and **relative error (Rel Err)**.
Absolute error is the absolute difference between the predicted depth and the ground truth depth, while relative error is the absolute difference divided by the ground truth depth.
The formulas for absolute error and relative error are:

$$AbsErr = \frac{1}{N} \sum_{i=1}^{N} |d_i - \hat{d}_i|, \tag{8}$$

$$RelErr = \frac{1}{N} \sum_{i=1}^{N} \frac{|d_i - \hat{d}_i|}{d_i}, \tag{9}$$

where N is the number of pixels in the image, $d_i$ is the ground truth depth at pixel i, and $\hat{d}_i$ is the predicted depth at pixel i.

### 3.2 Quantitative Results

| | | Segmentation | Depth | | |
|---|---|---|---|---|---|
| Model | Weighting | mIoU | Pix Acc | Abs Err | Rel Err |
| SegNet MTAN (mine) | DWA, $T = 2$ | **68.09** | 91.14 | 0.0161 | 38.39 |
| SegNet MTAN (mine) | Equal Weights | 67.76 | **91.15** | 0.0160 | 38.01 |
| SegNet MTAN (paper) | DWA, $T = 2$ | 53.29 | 91.09 | 0.0144 | 34.14 |
| SegNet MTAN (paper) | Equal Weights | 53.04 | 91.11 | **0.0144** | **33.63** |

Table 1: 7-class semantic segmentation and depth estimation results on CityScapes validation dataset. The results confront my implementation with the paper's one. For segmentation task, the higher the better, while for the depth task, the lower the better. In bold the best results.

**Discussion** The experimental results show that the implemented MTAN model achieved promising results on the Cityscapes dataset for both the segmentation and depth estimation tasks.
Interestingly, my implementation outperformed the MTAN baseline in the segmentation task, despite training for 100 epochs instead of 200.
This could be attributed to the use of weight decay regularization, He weight initialization, and a learning rate scheduler.
On the other hand, the original MTAN original model performed slightly better on the depth estimation task, but the difference was not significant.
These results suggest that the proposed MTAN model is effective in jointly learning multiple tasks, and that incorporating appropriate regularization and initialization techniques can improve the performance of the model.
Overall, the experiments demonstrate the potential of multi-task learning approaches for addressing different vision tasks simultaneously, while achieving competitive performance compared to single-task models.

### 3.3 Qualitative Results

In this section, I present the qualitative results of the experiments with my implementation of MTAN model for multi-task learning on the Cityscapes dataset. To evaluate the performance of the model, I visually compare the predicted segmentation and depth maps with the corresponding ground truth maps on a set of validation images.

First, I show some examples of semantic segmentation results obtained with the MTAN model. The predicted segmentation maps are compared to the ground truth segmentation maps for each image in the set. The results show that the model is able to accurately capture the semantic information in the images,

Next, I present the results of the depth estimation task, comparing the predicted depth maps with the corresponding ground truth maps. The depth estimation task is a challenging problem in computer vision, and the MTAN model performs well on this task, achieving comparable results to quantitative ones.

Finally, I provide some examples of the superimposition of the segmentation and depth masks on the original images. The superimposed masks clearly show the areas of the images that have been correctly segmented and the depth information estimated by the model.

Overall, the qualitative results of the experiments support the quantitative results presented earlier, showing that my implementation of the MTAN model is effective in jointly learning the tasks of semantic segmentation and depth estimation.

**Segmentation Task** The following displays the validation results on 7-class semantic labelling.

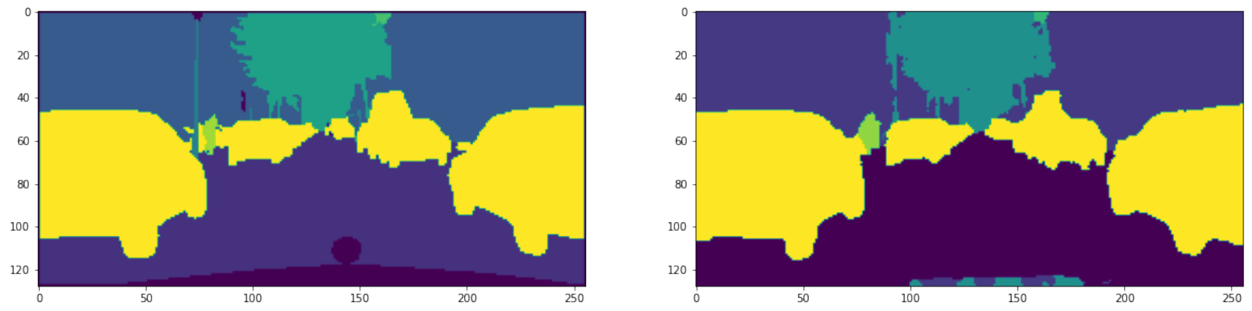

Figure 1:

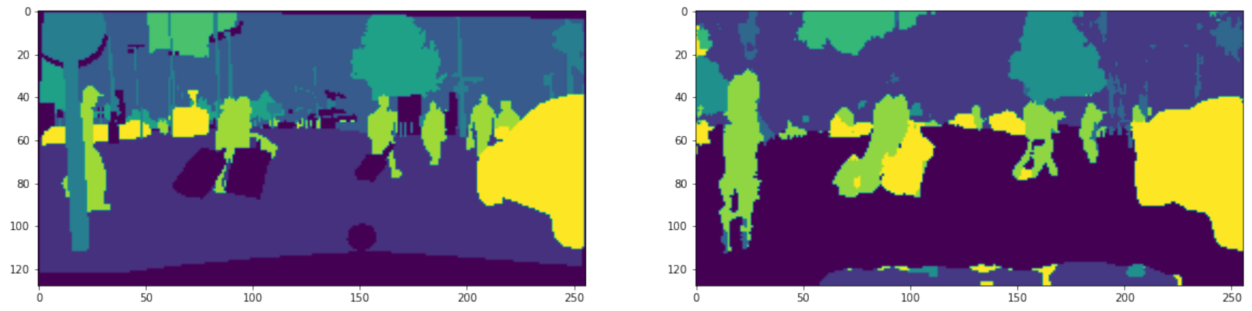

Figure 2:

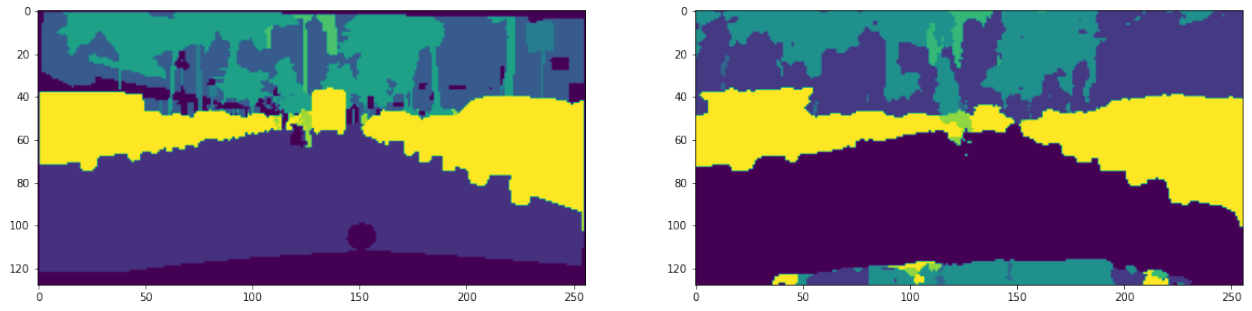

Figure 3:

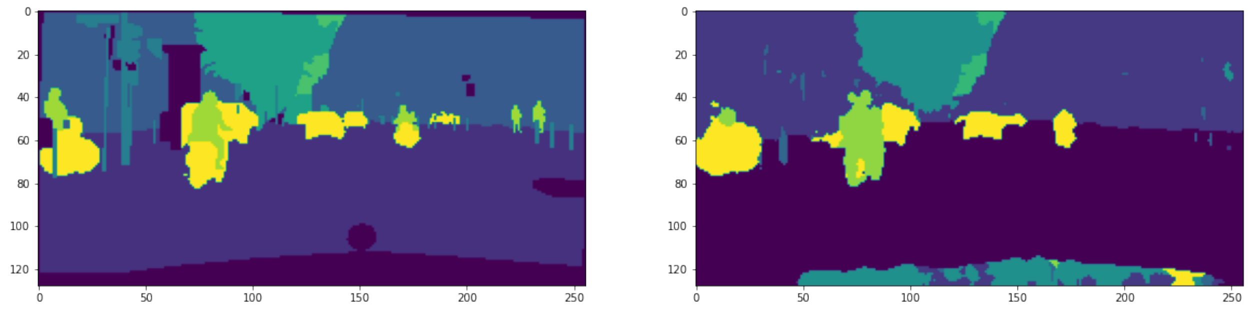

Figure 4:

Figure 5: On the left there is the *ground truth* and on the right there is the prediction.

The following shows the overlap between the prediction and the true images.

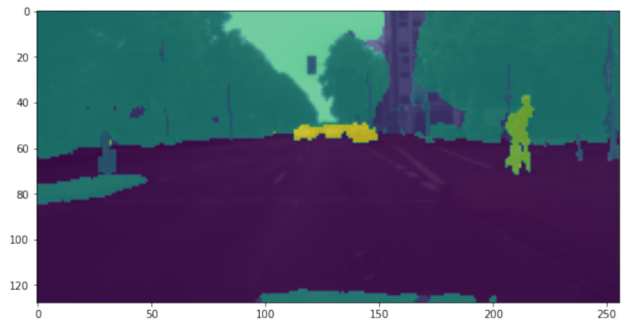

Figure 6:

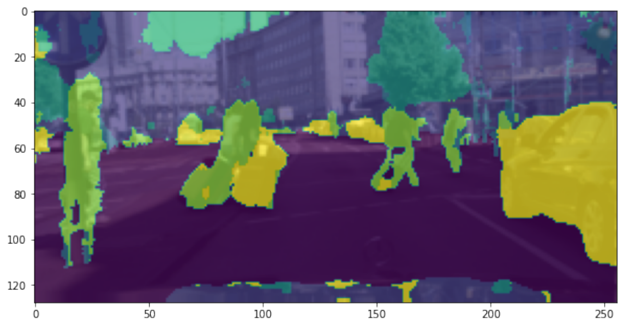

Figure 7:

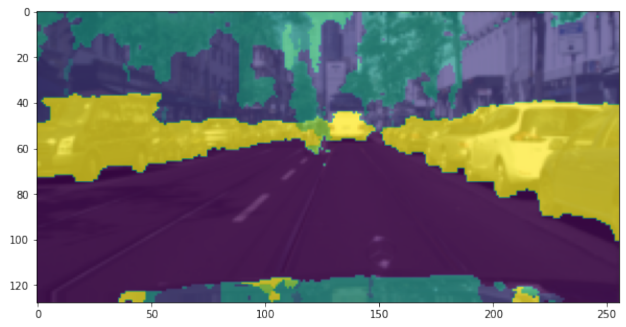

Figure 8:

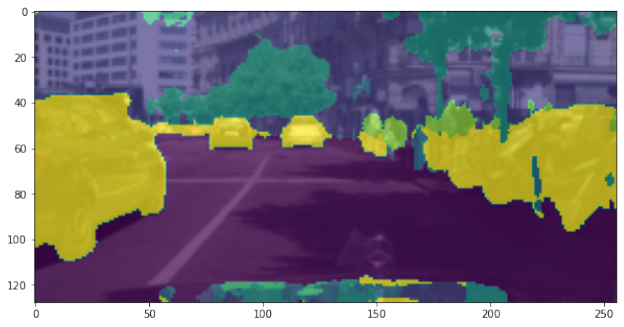

Figure 9:

**Depth Task** The following displays the validation results on 7-class semantic labelling.

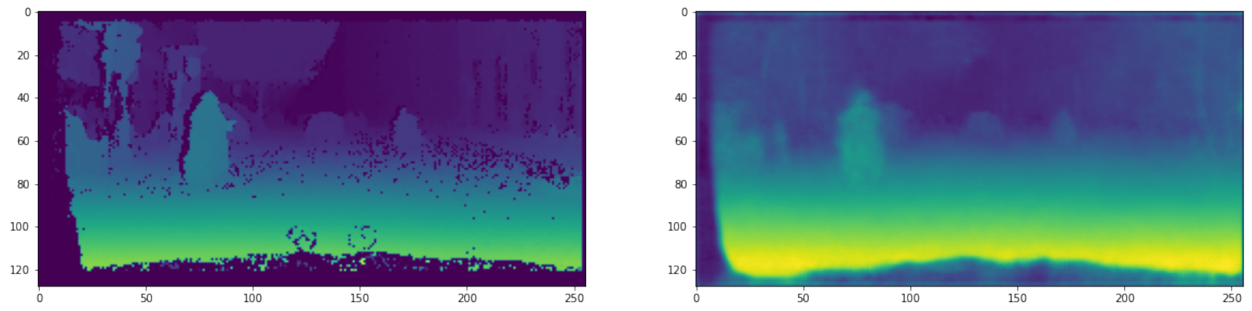

Figure 10:

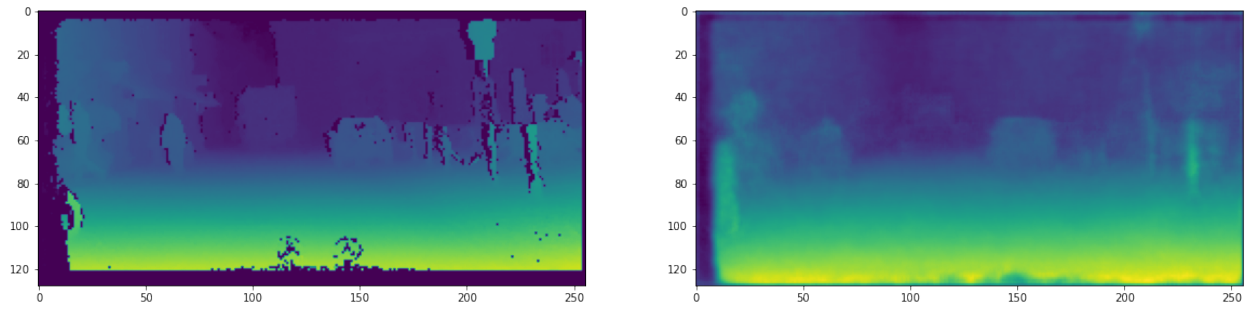

Figure 11:

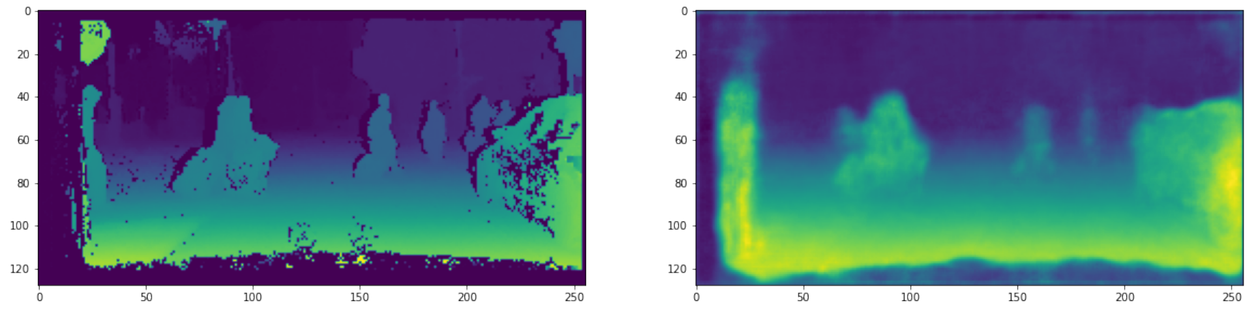

Figure 12:

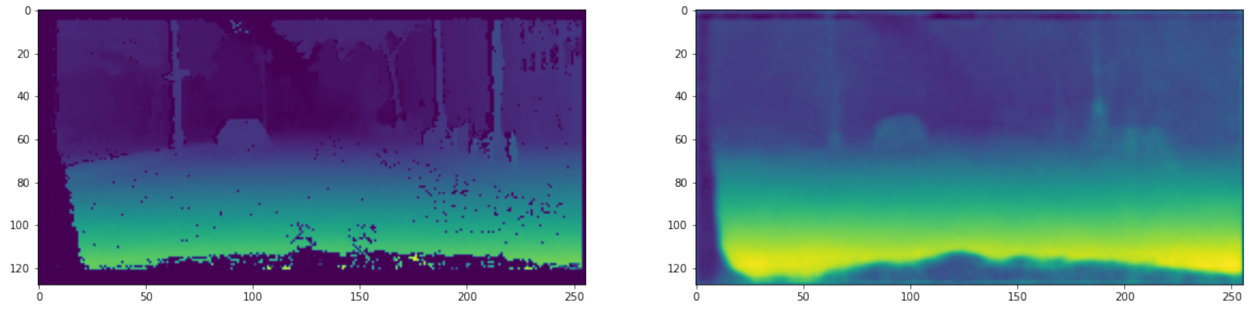

Figure 13:

Figure 14: On the left there is the *ground truth* and on the right there is the prediction.

The following shows the overlap between the prediction and the true images.

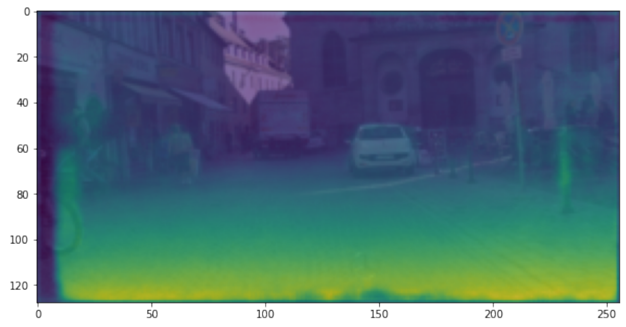

Figure 15:

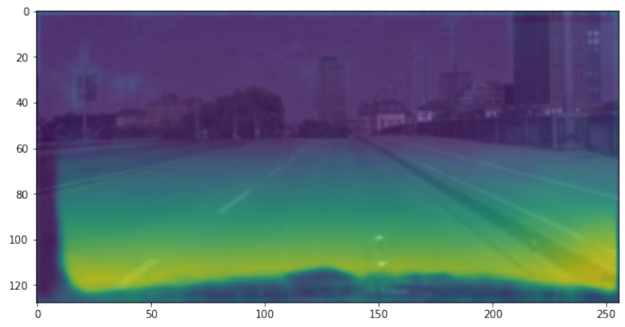

Figure 16:

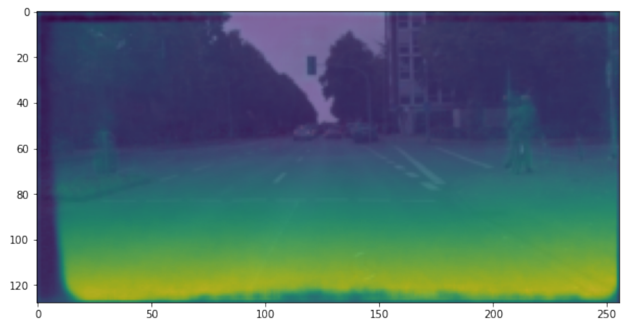

Figure 17:

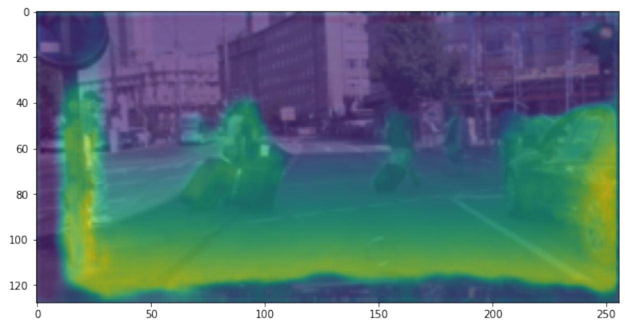

Figure 18:

## 4 Conclusion

This work presented the implementation of the MTAN model, as proposed in the paper "End-to-End Multi-Task Learning with Attention," for the tasks of segmentation and depth estimation. Despite using a smaller version of the SegNet architecture and training for fewer epochs, I achieved comparable results to the original paper in terms of mIoU and pixel accuracy for the segmentation task and Abs Err and Rel Err for the depth estimation task.

The experimental results showed that the implemented MTAN model achieved promising results for both tasks, and even outperformed the MTAN baseline in the segmentation task, despite the shorter training duration.
These findings suggest that incorporating appropriate regularization and initialization techniques can improve the performance of the model.
Furthermore, the experiments demonstrate the potential of multi-task learning approaches for addressing different vision tasks simultaneously.

In conclusion, this work demonstrates the effectiveness of the MTAN model for multi-task learning, and provides valuable insights into the benefits and challenges of implementing such models in real-world scenarios.
The findings highlight the importance of appropriate regularization and initialization techniques in achieving optimal performance, even with limited resources

## References

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39 (12):2481–2495, 2017.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Ekaba Bisong and Ekaba Bisong. Google colaboratory. *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, pages 59–64, 2019.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.