FocusOnDepth: A Monocular Depth Estimation for *in-the-wild* Auto-focus application

Younes Belkada Antoine Cadiou
École Normale Supérieure Paris-Saclay
firstname.lastname@ens-paris-saclay.fr

Abstract

*Running very accurate and large models on a small embedded device for various tasks in challenging. Some recent work has shown a significant improvement in the prediction quality using Vision Transformers for Dense Prediction tasks (*e.g.*, Image segmentation, Depth estimation). However, a model is trained from scratch for each task. Through our work, we get in condition for an auto-focus application on humans using Monocular images, and we demonstrate that a single ViT-based encoder is sufficient for Monocular Depth estimation and segmentation. Our code is publicly available at* https://github.com/antocad/FocusOnDepth *for an interactive application and future work.*

1. Introduction

Depth estimation task could be achieved with 2 inputs/images for the same scene but taken from 2 different places. This method corresponds to a binocular Depth-estimation.
Recent works have shown that in the real world, humans rely on the image obtained by their left and right eyes in order to estimate depths of surrounding objects. Thus, depth estimation is a classic task in computer vision, which is of great significance for many applications such as augmented reality, target tracking and autonomous driving.
First of all, we will explore the existing Deep Learning models for monocular depth estimation, more precisely the methods based on Convolutional Neural Networks (CNN) and on Vision Transformers (ViT).
Secondly, we will implement a recent Vision Transformers based architecture for this task and we will bring some modifications to the architecture to fulfill our initial problem (*c.f*. Auto-focus). We will seek to improve it by adding a segmentation head in order to perform multi-task learning using a customly built dataset.
Finally, we will implement our model for *in-the-wild* images (*i.e*. with no control on the environment, the distance and size of objects of interests, and their physical properties (rotation, dynamics, etc.)) for Auto-focus application on humans and will give qualitative comparison across other methods.

Figure 1. Example of an output after having applied a post-processing on the depth estimation and the semantic segmentation

2. Problem Definition

More formally, the dataset D contains for each image x_i, a pair of depth map y_i^d and semantic segmentation mask y_i^s containing only the class *human*. The objective function L_{total} that has to be minimized is the sum of the depth estimation objective L_{ssi} with the semantic segmentation objective L_{ce}.

$$L_{total} = L_{ssi} + L_{ce} \quad (1)$$

For the depth estimation objective, we refer to [12][6] and use a scale-and-shift invariant trimmed loss to achieve smooth depth estimation. For the segmentation objective, we use a classic CrossEntropy loss.

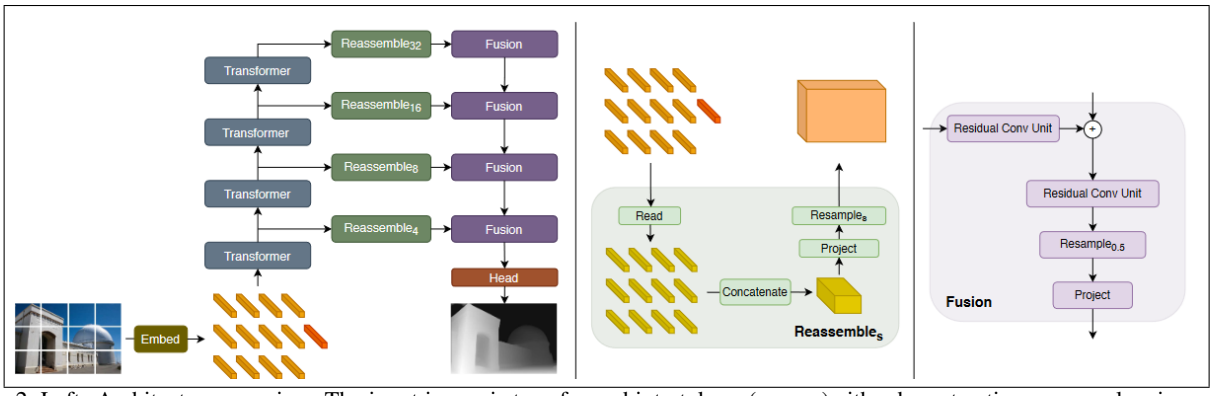

Figure 2. Left: Architecture overview. The input image is transformed into tokens (orange) either by extracting non-overlapping patches followed by a linear projection of their flattened representation (DPT-Base and DPT-Large). The image embedding is augmented with a positional embedding and a patch-independent readout token (red) is added. The tokens are passed through multiple transformer stages. We reassemble tokens from different stages into an image-like representation at multiple resolutions (green). Fusion modules (purple) progressively fuse and upsample the representations to generate a fine-grained prediction. Center: Overview of the Reassembles operation. Tokens are assembled into feature maps with 1 s the spatial resolution of the input image. Right: Fusion blocks combine features using residual convolutional units [7] and upsample the feature maps.

3. Related Work

In this section, we briefly present the relevant methods for Dense Prediction tasks in the current literature, by reviewing the classic methods, as well and recent promising approaches.

3.1. CNN based models

Classic approaches for Dense Prediction tasks mostly includes techniques based on Fully Convolutional Networks (FCN)[9] [4] [10]. These architectures are trained in an auto-encoder fashion with a pixel-to-pixel correspondence. Furthermore, the Unet architecture[14] has also proven to be a good dense predictor.

3.2. Vision Transformers based models

Most recent approaches consist of using Vision Transformers (ViT) to achieve this task. ViTs have proven to be better than CNNs in most of the computer vision tasks such as classification, solving inverse problems (*i.e.* denoising, deblurring, ...). We focused on a paper entitled 'Dense Prediction Transformer - DPT'[12] which is very interestiing because the architecture is combining both ViT and CNN approaches. In fact, as we can see in the Figure 3, the architecture is similar to a U-Net, in the sense that there is a kind of encoder composed of severals transformers, and a decoder (Fusions blocks), essentially composed of convolutional layers.

4. Methodology

In this section we present in detail the pipeline of training our method, from the dataset creation to the detailed architecture of the method.

4.1. Dataset creation

Following [13] we build our paired dataset (containing for each image both the segmentation map and the depth map) using a combination of 3 datasets. In order to maximize the chances to obtain interesting frames containing humans, we have decided to take several scenes from:

- Inria Pose 3d dataset[1]
- PoseTrack [2]

and most scenes from NYUV2 [11] dataset. Since the ground truth semantic segmentation map on the class *human* is not provided for the mentioned datatsets, we run an off-the-shelf *DPT-large* model[12] trained on ADE20K[15] for semantic segmentation. We save our ground truth masks considering only the class *human*.

For the depth estimations, NYUV2 dataset[11] already contains ground truth depth estimation maps, whereas for Inria Pose 3d[1] and PoseTrack[2] we also run the off-the-shelf *DPT-large* model[12] and consider the predicted depth maps as the ground truth depth map.

Table 1. Table detailing the number of selected frames per dataset

Dataset	**Selected frames**
Inria Pose 3d dataset[1]	1350
PoseTrack[2]	1170
NYUV2[11]	1449

4.2. Model architecture

Our architecture is inspired from [12], but instead of having two different models for each task, we epxlore the possibility of having a single ViT and two different head to achieve the aforementioned tasks.

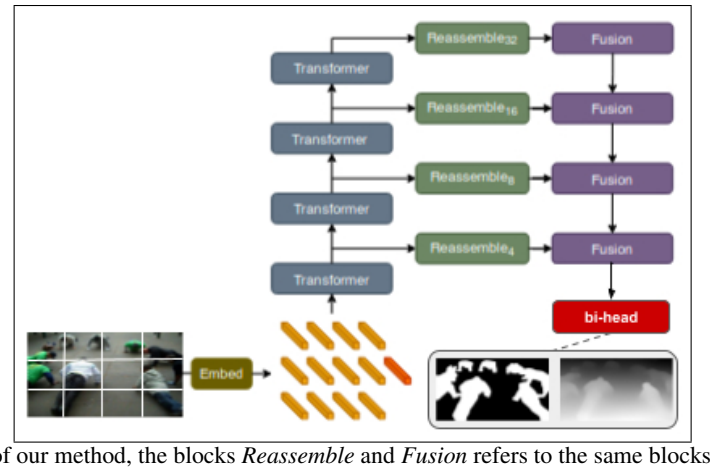

Figure 3. Detailed architecture of our method, the blocks *Reassemble* and *Fusion* refers to the same blocks of the original DPT[12] paper.

Our method, will divide the input images into 16x16 patches and pass the flattened representation of each patch into a Transformer-based ViT encoder. The encoder is pre-trained on ImageNet[3]. One can also use different ViT encoders (*e.g.*, *ViT-base* or *ViT-large*) for different results trade-offs.

We extract the representations of each patch at several levels, these representations are combined in the *Reassemble* block that will aim to transform these representations in a higher space that can be used in the *Fusion* block.

The *Reassemble* block is a composition of 3 different sub-blocks.

- *Read* block: The input of lenght $nb_{patches} + 1 \times D$ needs to be first mapped into a representation of size $nb_{patches} + 1 \times D$. This process is called the *Read* process. Therefore, one can read the input by ignoring the the representation of the CLS token (in red in the figure 3), by having an MLP that learns this mapping or by adding the representation of the CLS token on each representation.

- *Concatenate* block: The next step consists of combining these representations using the *Concatenate* block. This step simply consists in concatenating each representation following the order of the patches (since each representation corresponds to a representation of a patch). This yields into an image-like representation of this feature map.

- *Resample* block: This block consists of applying a 1×1 convolution to project the input image-like representation into a space $\hat{D}$ of dimension 256. This convolutional block is followed by another convolution where the inner parameter of it changes with respect to the layers of the ViT encoder (you can refer to our code or to the main paper [12] for more details)

The representations are transformers in order to be passed to the *Fusion* block at each level. This block takes as input the image-like representations from the reassemble block, as well as the previous representation from the *Fusion* block. After summing these two representations, we apply two successive convolutional units and upsample the predicted representations.

Finally, the representation from the last *Fusion* layer is used as input for our bi-head module. Each head consists of a small deconvolutional block with an unsampling module. You can refer to our provided code for the details of our bi-head module.

As mentioned before, one can train our model using different backbones *ViT-base* and *ViT-large*. Some parameters of the architecture will vary according the backbone type we select:

- ***ViT-base*:** Embedding dimension D is equal to 768, and we select the residual embeddings respectively from the layers $2, 5, 8, 11$.

- ***ViT-large*:** Embedding dimension D is equal to 1024, and we select the residual embeddings respectively from the layers $5, 11, 17, 23$.

4.3. Training

We have noticed that the hyper-parameter tuning step is a very crucial point. Depending on the initialization and/or hyper-parameters the model tends to converge differently. In our training script, we support the use of 2 different encoders that have different size, an encoder called *ViT-base* that contains less layers than a larger model called *ViT-large*. Depending on the choice of the encoder, we have to choose the batch_size accordingly. For our experiments, we use a batch size of 8 when training the *ViT-base* backbone model and we use a batch size of 4 when training the *ViT-large* backbone model.

The encoders weights are initialized from an ImageNet[3] trained checkpoint. Therefore, in our

model the decoder blocks (*i.e*. the *Fusion* and *Reassemble* blocks as well as the bi-head module) are trained from scratch. To take the advantages of the features learned from ImageNet[3], we train our model using 2 different optimizers using different learning rates. More precisely, we update our weights using an optimizer $O_{scratch}$ for the parameters trained from scratch and another optimizer $O_{backbone}$ for the weights of the backbone. We use an Adam[5] optimizer with a learning rate of 1.10^{-5} for $O_{backbone}$ and an Adam[5] optimizer with a learning rate of 3.10^{-4} for $O_{scratch}$. We also use a learning rate scheduler that decreases the learning rate if the validation loss is on a plateau. These techniques allow us to have a smoother convergence and update of weights.

Following [12], we train our models during 60 epochs. Since the depth estimation and segmentation are flip and crop invariant, we apply a random horizontal flipping (50%) as well as a random cropping (30%) and a random rotation (20%, with a maximum angle value of $10°$) on images in order to make the model robust against random perturbations. A summary of our training and validation plots can be monitored here.

Concerning the data used for the training, we have splitted our *merged dataset* 4.1 after having shuffled it (since the data from *Inria-dataset*[1] and from *Posetrack-dataset*[2] are sequences of movies, we have lost the time information, and we are subject to overfitting because the images from a same scene are very similar. One possibility we thought to for solving this was to put entire distinct sequences in the training and in the testing phase).
Finally, we have kept 50% of the dataset for the training set, 20% for the evaluation set and 30% for the testing set.

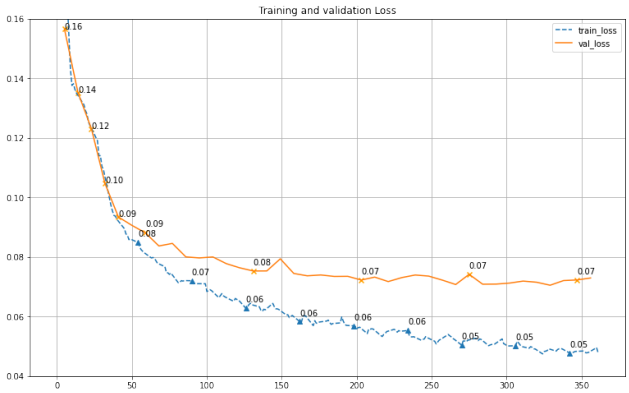

Figure 4. Training and Validation loss of FocusOnDepth with ViT-Base as encoder

5. Evaluation & Observation

5.1. Convergence of the model

Throughout the training procedure we have noticed a very interesting phenomenon. At the beginning of the convergence the model learned to associate the predicted depth map together with the semantic segmentation of the class *human*.

Figure 5. Example of the Image and its associates depth and segmentation prediction after epoch 1

As it can be observed from the comparison figure 5.1, the model understands first that it needs to perform the same task for both heads, then converges from the next epoch to what we want to achieve.

5.2. Quantitative Evaluation

5.2.1 Comparison using different backbones

We study the impact of using different backbones by running the training procedure on 2 different models. We compare their performance on our custom loss function 1 on the validation set. 6

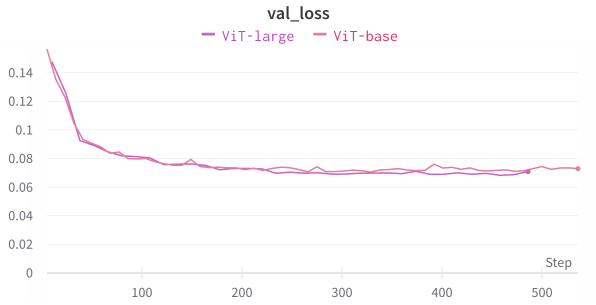

Figure 6. Validation loss of FocusOnDepth with *ViT-large*, *ViT-base*

We achieve comparable performances while using different backbones. Thus, in our dataset there is not a high gap between using a larger backbone for the dense prediction tasks.

5.3. Qualitative Evaluation

We have qualitatively evaluated our whole pipeline by training a *ViT-base* model. The results can be seen following this link: https://www.youtube.com/watch?v=dW02Ng6gy4w.

5.4. At inference time

At inference time, after extracting the semantic segmentation map and the predicted depth map, there are several possibilities to achieve human auto-focus. Currently, after selecting a pixel in the input image (that has to be a human pixel), we combine the information of the depth map to estimate the depth of the selected region with the information of the localization of humans using the segmentation mask. Once combined, the humans that have the same depth value as the human selected by the user will not be blurred. An example result can be observed in the figure 1. Other techniques can be applied but we did not investigated that much into this direction since the core contribution of this work was to show that a single ViT encoder in sufficient to perform different Dense Prediction Tasks.

6. Conclusions & Future works

In this project, we have succeeded to implement the recent [12] architecture from scratch with a strong code-base, by creating a simple and exciting application which is the auto-focus on humans. As we opened the black box of the model, we went deep into the details of how a Vision Transformer has to be implemented.

In terms of personal satisfaction, we are very proud of our work, given that in the official repository of DPT[12], there is no training script that is provided although there is a large interest from the community to release the training script. We plan to advertise our repository which already contains a strong training script to the community in the near future.

This project has also accomplished his research duty. We were asking to ourselves if a single ViT-based encoder is powerful enough to perform two similar but yet different Dense Prediction Tasks. We have demonstrated through our work that this is feasible, and further research needs to be investigated to confirm (or not) this first answer, for *e.g.* by training on more classes using famous datasets such as COCO[8], or ADE20k[15]

We have left the possibility to researchers and practitioners to engineer their own way of applying autofocus using the depth map and the semantic segmentation mask. This engineering step is a bit hard, especially when several people occludes between each other, therefore is impossible to apply autofocus on one person. This can be improved in the future, by focusing on an end-to-end prediction for this task or to perform instance segmentation instead of semantic segmentation.

We have noticed that after post-processing the images to obtain the final results, some parts of the images contain several artifacts on the borders between the blurred regions and clear people. One can try to find a way to smooth the rendering results using classic Computer Vision algorithms or some end-to-end approaches.

As mentioned before, another exciting direction would be to adapt this method for instance segmentation. If the instance segmentation map is provided, we would be able to apply a tracking on a single or multiple people of the video. The post-processing step could be better by estimating the density distribution of the depth of a single person, which will potentially lead to a smoother rendering of the final result.

References

[1] K. Alahari, G. Seguin, J. Sivic, and I. Laptev. Pose estimation and segmentation of people in 3d movies. In *Proc. IEEE International Conference on Computer Vision*, 2013. 2, 4

[2] M. Andriluka, U. Iqbal, E. Ensafutdinov, L. Pishchulin, A. Milan, J. Gall, and Schiele B. PoseTrack: A benchmark for human pose estimation and tracking. In *CVPR*, 2018. 2, 4

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3, 4

[4] Shir Gur and Lior Wolf. Single Image Depth Estimation Trained via Depth from Defocus Cues. *arXiv:2001.05036 [cs]*, Jan. 2020. arXiv: 2001.05036. 2

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 4

[6] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos, 2018. 1

[7] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. *CoRR*, abs/1611.06612, 2016. 2

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 5

[9] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[10] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, May 2021. 2

[11] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2

[12] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision Transformers for Dense Prediction. *arXiv:2103.13413 [cs]*, Mar. 2021. arXiv: 2103.13413. 1, 2, 3, 4, 5

[13] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 2

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 2

[15] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 2, 5