CS231N Final Project: Hand Segmentation and Depth Estimation

Yuxuan Wu
Stanford University
450 Jane Stanford Way, Stanford, CA
wyx@stanford.edu

Marina Qian
Stanford University
450 Jane Stanford Way, Stanford, CA
qiany11@stanford.edu

Abstract

Recent advancements in augmented reality (AR), virtual reality (VR), and mixed reality (MR) have enabled applications like remote surgery, where surgeons can practice complex procedures in virtual environments. This project aims to address the hand tracking problem essential to such applications by focusing on two main tasks: 2D segmentation and depth estimation from monocular videos. We propose an encoder-decoder architecture with a shared encoder between both tasks and separate decoders for each task. This approach aims to efficiently solve both the 2D segmentation task and the depth estimation task, enhancing the VR experience for surgical training. We employ a fully convolutional network as baseline and explore a deconvolution network for improved performance.

1. Introduction

With recent advancements in augmented reality (AR), virtual reality (VR), and mixed reality (MR), new opportunities are emerging for various applications. One exciting possibility is remote surgery using MR technology. Imagine surgeons practicing complex procedures in a virtual setting, allowing them to improve their skills without any risk to patients.

However, realizing such an application is not easy. For an immersive experience, surgeons need to interact with the virtual environment in real-time with precision. This requires the VR system to track the positions of the surgeons' hands accurately, ensuring the virtual environment does not interfere with their ability to see and place their hands correctly.

Tracking the position of a surgeon's hands is a challenging task. To tackle this, we begin by assuming that the VR system's visual perspective is fixed within the virtual environment. This means the global coordinate system remains static.

Next, we consider a VR system that uses only a single monocular camera. The goal of this system is to find the 3D coordinates of the surgeon's hands within the fixed coordinate system. To do this, the system first locates the hands on a 2D plane parallel to the camera lens using a 2D segmentation task. Then, it reconstructs the depth information using RGB videos, determining the radial distance from the hands to the camera. This approach breaks down the complicated task of hand tracking into two well-investigated subtasks: 2D segmentation and depth estimation.

The first subtask, 2D segmentation, has been extensively studied in the field of computer vision. Although this project uses video inputs, we can still approach this problem as a dense image classification problem and perform semantic segmentation on each video frame. A fully convolutional network (FCN) serves as a robust baseline for this purpose. An FCN employs a convolutional neural network (CNN) to process image features, which are then interpolated into pixel labels [1]. However, traditional FCNs often rely on basic non-learnable decoders, which leads to a significant loss of detailed information during the upsampling process and results in limited output resolutions [2]. To address this issue, a modified architecture incorporating a deep deconvolution network has been proposed to generate high-resolution segmentation results [2]. The model we proposed for this project is thus built on this work.

Similarly, much prior work has been conducted on depth estimation from images or videos, with both supervised and unsupervised methods proposed to tackle this challenge. Like the 2D segmentation task, many of these solutions are also based on CNNs, just incorporating some additional features such as optical flow to enhance performance. This similarity indicates that the depth estimation problem might share common aspects with the 2D segmentation problem, suggesting that we could potentially reuse parts of the segmentation architecture for depth estimation.

In summary, our project focuses on addressing two main tasks: 2D segmentation and depth estimation based on RGB images. We aim to provide an efficient solution that tackles both problems using a single unified model. Utilizing an encoder-decoder architecture, we employ a shared encoder for both tasks, with distinct decoders attached for each specific task. Our baseline model is an FCN network based on [1], while we propose an additional model featuring a deconvolution network for enhanced performance.

2. Related Work

The first of the two subtasks in this project is 2D segmentation on video inputs. As mentioned in the introduction, this can be approached as a standard semantic segmentation problem on images. For this purpose, FCNs are a strong baseline. FCNs first obtain coarse labels using CNNs [2]. Then, upsampling is implemented with simple methods such as bilinear interpolation for pixel-wise labeling [1].

However, FCN-based methods have intrinsic limitations due to the absence of a complex upsampling strategy. The bilinear interpolation layer used for pixel-wise labeling is not learnable, meaning that upsampling is learned indirectly through the convolutional network. This approach makes it difficult to accurately reconstruct highly nonlinear structures, such as object boundaries[2].

To tackle this problem, a modified architecture incorporating a deep deconvolution network has been suggested to generate high-resolution segmentation results [2]. The deconvolution network includes unpooling layers that correspond to the pooling layers in the convolutional network, as well as deconvolution layers, which function as transposed convolution operations. The unpooling layers perform sparse upsampling to reconstruct the structure of an input object, while the deconvolution layers transform these sparse activations into a denser form to reconstruct the object's shape [2].

For the depth estimation subtask, on the other hand, we had to do a lot more investigation to clearly define our objective. We began by reviewing several papers [3, 4, 5, 6, 7] that analyzed different model architectures, training strategies, and evaluation metrics for depth estimation from images or videos. However, not all studies were relevant to our project. For instance, due to the challenges in collecting training data with depth labels, many works focused on depth estimation from monocular videos using unsupervised learning methods [8]. However, since the EgoGesture dataset we are using contains depth information, we decided that it is better for us to leverage this valuable data using a supervised learning approach.

Moreover, we found that a lot of the proposed solutions for depth estimation are based on convolutional networks, just like for the 2D segmentation task, with some additional features such as temporal information and optical flow to take into account [3, 6, 4, 5]. This similarity between the approaches for both tasks inspired us to develop a unified framework to address both problems simultaneously.

To evaluate the feasibility of this approach, we delved into the concept of joint multi-task learning. This technique projects features extracted from different tasks onto each other to enhance the final outputs [3]. Our goal is to combine the depth estimation task with semantic segmentation. Given that these tasks share similar contextual information, a common strategy is to use a single neural network with a shared encoder to capture scene structure features, while employing separate decoders for semantic segmentation and depth regression [9, 10, 11]. Since the deconvolution network proposed by [2] also employs an encoder-decoder architecture, we can readily adapt this architecture for multi-task learning using this strategy.

3. Methods

3.1. Fully Convolutional Network

Our baseline architecture is an FCN inspired by the design proposed by Simonyan et al. [1]. In their work, Simonyan et al. adapted classic image classification models, such as AlexNet [12], VGG nets [13], and GoogLeNet [14], for the task of semantic segmentation. By treating semantic segmentation as a dense classification problem, they employed transfer learning on these pre-trained models for the new task. Specifically, they replaced the final layers of the original models with a 1×1 convolution layer to obtain scores for dense classification.

Following their approach, we designed our FCN block based on the VGG-16 network. We replaced the fourth max-pooling layer and all subsequent layers with a 1×1 convolution layer with a single output channel, as illustrated in Figure 1. To generate outputs that match the original image size, the scores are then upsampled using a simple nearest neighbor method for the final predictions.

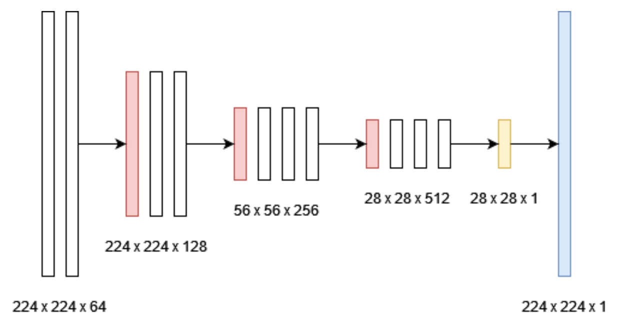

Figure 1. Diagram of an FCN block. Red represents 2×2 max-pooling layers. Yellow represents a 1×1 convolution layer. Blue represents an upsampling layer. All other unmarked layers are convolutional layers with a kernel size of 3 and padding of 1, followed by batch normalization and ReLU activation. Text labels denote the output size of the convolutional layers.

Our baseline model features two parallel FCN blocks, as illustrated in Figure 2. Each block is dedicated to either mask or depth prediction. We treated the 2D segmentation task as a dense binary classification problem, applying a sigmoid activation function to the classification scores. This allowed us to utilize Binary Cross-Entropy Loss (BCELoss) for training. For depth estimation, treated as a regression problem, we employed Mean Squared Error (MSE) loss to measure the discrepancy between the predicted and ground-truth depth maps.

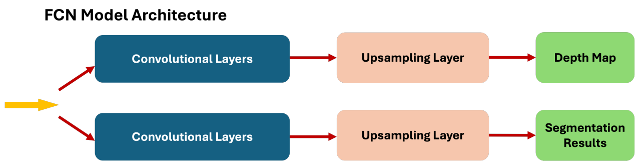

Figure 2. Block Diagram showing the architecture of the baseline model, adapted from the FCN architecture.

3.2. Deconvolution Network

The model we propose builds on the design from Long et al. [2]. The original model is an encoder-decoder network designed for pixel-level segmentation tasks. The encoder, based on the VGG-16 architecture, transforms each image into a 4096-dimensional feature vector, as shown in Figure 3. The decoder, which mirrors the encoder, uses deconvolution and unpooling layers to progressively reconstruct coarse-to-fine object structures, as shown in Figure 4.

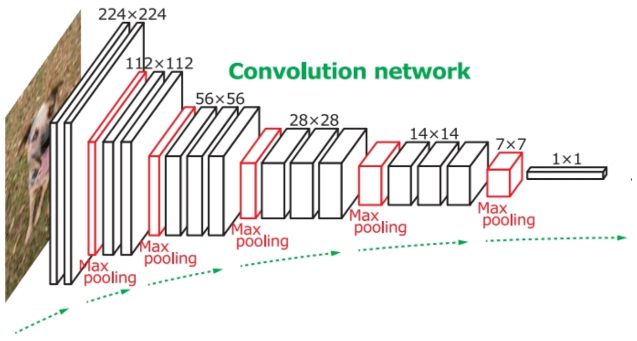

Figure 3. Architecture of the encoder network. Adapted from [2].

We modified this model by adding a second decoder network, which enables our model to perform concurrent 2D segmentation and depth estimation. Given that the VGG-16 network in the encoder is very good at capturing intricate object details, we expect that the same set of object features would be sufficient for the depth estimation task. Consequently, with the encoder-decoder structure, we can conveniently use a single shared encoder to extract features for both segmentation and depth estimation tasks.

Moreover, the learnable decoder's large receptive field, achieved through a sequence of down-sampling and up-sampling operations, accommodates objects of various scales. Since the same property is also desired for depth estimation, we expect that another decoder with an identical architecture will work well for the depth estimation task. A block diagram of our proposed model is shown in Figure 5.

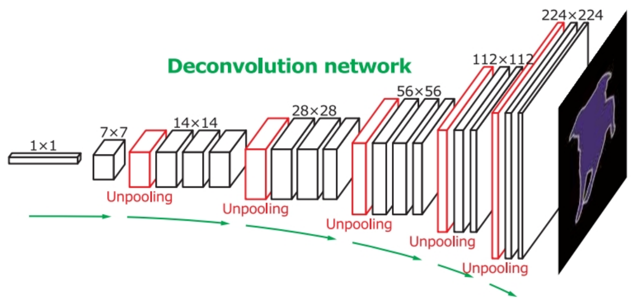

Figure 4. Architecture of the decoder network. Adapted from [2].

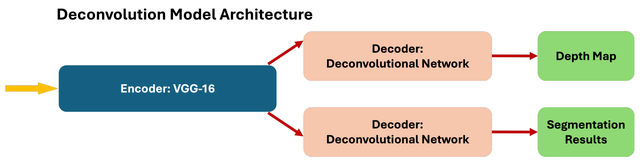
Deconvolution Model Architecture
Encoder: VGG-16
Decoder: Deconvolutional Network
Depth Map
Decoder: Deconvolutional Network
Segmentation Results

Figure 5. Block Diagram showing the architecture of the deconvolution model.

3.3. Evaluation Metrics

To assess the model's performance, we introduced two metrics: the depth score and the mask score. The depth score measures the accuracy of depth estimation, allowing for a maximum error margin of 5%. On the other hand, for the mask estimation task, we used the F1 score instead of accuracy. This is because the hands are relatively small compared to the video frame size, so a model can achieve seemingly "good" accuracies without providing genuinely valuable predictions. Intuitively, consider a model that always produces an all-zero depth map and an all-negative segmentation map. If the hands occupy only 10% of the frame, such a model can still achieve 90% accuracy on both tasks. An example prediction embodying this issue is shown in Figure 6. Therefore, to alleviate the undesired effects from the significant class imbalance between hand pixels ($\approx 10\%$) and background pixels, we opted to utilize the F1 score to evaluate the model's segmentation performance.

The F1 score, a harmonic mean of precision and recall, can be computed as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

where TP represents True Positives and FP represents False

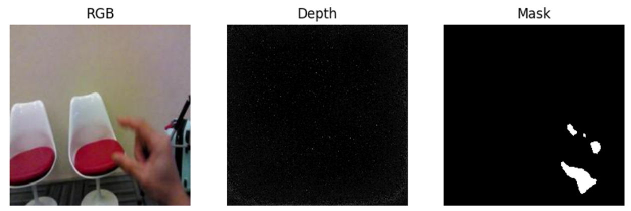

Figure 6. Without considering the F1 score, most pixels in the model's outputs tend to be close to zero.

Positives; and

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where FN represents False Negatives.

The formula for calculating the F1 score is:

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

From this formula, we can see that the F1 score achieves its highest value when both precision and recall are equally high. This is especially useful in situations with class imbalance, where one class occurs much more frequently than the other.

4. Dataset

We used a subset of the EgoGesture dataset [15] for this project. This dataset contains egocentric gesture data featuring 83 gestures from 50 subjects across 6 different scenes. It includes RGB-D videos collected with an Intel RealSense SR300 camera, providing both RGB and depth videos of the gestures. Additionally, the dataset offers a version with the videos separated into individual JPEG image frames available for download. Examples of RGB frames and their corresponding depth frames are shown in Figure 7.

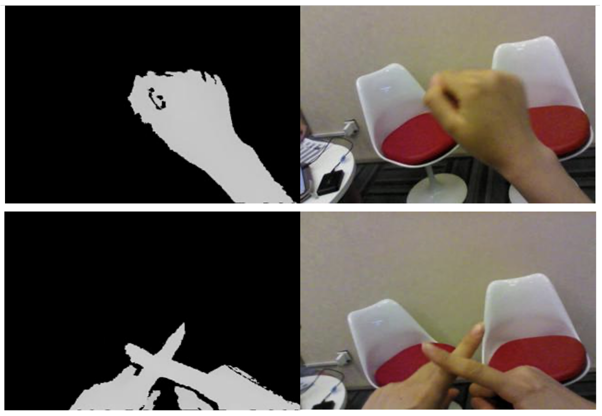

Figure 7. Example frames of RGB video and corresponding depth map images.

One issue with the EgoGesture dataset is that many videos contain numerous ”idle” frames where the subjects' hands are completely out of frame. Therefore, we decided to remove all frames where the depth map is entirely zero, reducing the dataset size to enable more efficient model training.

Another challenge is that the EgoGesture dataset does not provide masks for supervised training in the segmentation task, and the image size of 320×240 does not match the input size of the baseline model, which is 224×224. Therefore, we needed to resize all images and generate the masks ourselves. Upon examining the depth images, we determined that they were close enough to serve as masks for the subjects' hands. Thus, we decided to generate hand masks from the depth images using classic signal processing techniques.

After resizing the depth images, we smoothed the edges to ensure that the masks covered the entire area of the hands. This required passing the depth images through a low-pass filter. For this, we chose a simple median pool with a kernel size of 7. We then applied a hard threshold of 0.5 to identify all the pixels corresponding to the hand masks. Examples of the results from these transformations are shown in Figure 8.

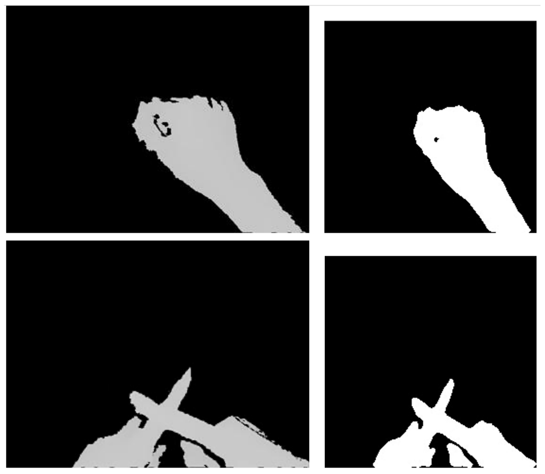

Figure 8. Examples of depth image and mask generated from a resized version of the original image.

Due to time and memory limitations, we focused on processing data from four subjects: subjects 1, 2, 3, and 10. We divided the derived dataset into three subsets. The training set comprised up to 512 frames from each video for subjects 1 and 10. Similarly, the validation set included up to 512 frames from each video for subject 2. The test set contained up to 512 frames from each video for subject 3. The total number of videos and frames in each dataset are shown in Table 1.

	Number of videos	Number of frames
Training set	96	39,484
Validation set	48	19,402
Test set	32	11,597

Table 1. Summary of the datasets.

	FCN	Deconvolution
Learning Rate	5.84×10^{-4}	8.05×10^{-4}
Weight Decay	1.67×10^{-8}	1.45×10^{-9}
α	8.47	23.0

Table 2. Hyperparameters.

5. Experiments

In our experiment, we trained and evaluated three models: the FCN-based model, the deconvolution model, and an LSTM network modified from the deconvolution model using Pytorch [16]. The primary focus of the project was on the first two models, while the LSTM network was an exploratory effort to incorporate temporal information in addition to spatial features.

5.1. Training

To perform end-to-end training of the entire network, we combined the depth loss and the mask loss by introducing a weight parameter, α. This parameter, along with the learning rate and weight decay, are the hyperparameters that need to be specified for the model.

Before training, we initialized all corresponding layers with pretrained weights from the VGG-16 network. This transfer learning strategy is beneficial, especially considering our limited data and computational resources. Additionally, we initialized other layers using the Kaiming initialization method [17]. To train the model, we employed the Nesterov Stochastic Gradient Descent optimizer with a momentum of 0.95. Due to the limitation of CUDA memory, we adjusted the batch sizes accordingly. For the FCN-based model, we utilized a batch size of 128, while for the deconvolution network, we set the batch size to 64.

For each model, we conducted a random parameter search for 20 parameter sets. These models were trained on a subset of the training set and evaluated on a subset of the validation set. Each parameter set underwent training for 5 epochs, and the model achieving the best average depth and mask scores on the validation set was selected. Finally, we trained the model for 50 epochs using the identified hyperparameters shown in Table 2 and saved the model that achieved the highest average scores on the validation set. Figure 9 illustrates the convergence of loss functions during training of both models.

5.2. FCN & Deconvolution Network

The performance of both models on all three datasets are shown in Table 3 and Table 4. Although both models exhibited slight overfitting, the results on the test set remained reasonable, likely due to the utilization of weight decay and batch normalization. While the two models showed simi-

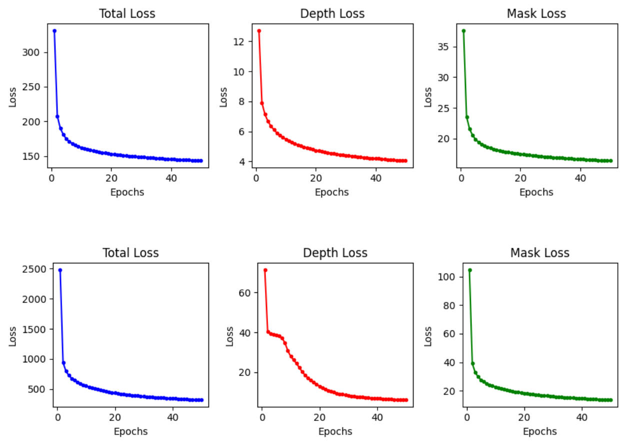

Figure 9. Top: Learning curves of the FCN-based model. Bottom: Learning curves of the deconvolution network.

lar performance on the mask prediction task, our proposed model outperformed the FCN on the depth estimation task. This outcome is intuitive, as the regression problem is inherently more complex than the binary classification problem and thus requires greater representational power and a more sophisticated model. The local information extracted by the FCN may be sufficient for the segmentation task, but depth estimation demands information beyond the local context. Consequently, the large receptive fields provided by the deconvolution model capture information beyond the 8-by-8 window of the FCN block, resulting in better performance for the depth estimation task.

	Depth score	Mask score
Training set	0.7329	0.9317
Validation set	0.6680	0.8667
Test set	0.6686	0.9087

Table 3. Performance of the FCN-based model.

Some prediction results from both models are illustrated in Figure 10 and Figure 11. Each model produced reasonable outcomes. As anticipated, the deconvolution network generated more intricate and detailed spatial features, while the FCN-based model delivered structured, coarse-grained results due to its interpolation approach.

	Depth score	**Mask score**
Training set	0.8509	0.9785
Validation set	0.7999	0.8537
Test set	0.8153	0.9093

Table 4. Performance of our proposed model.

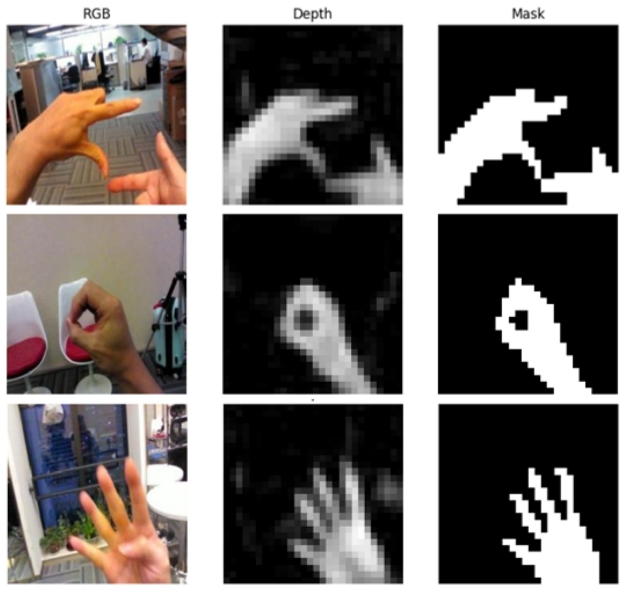

Figure 10. Prediction outputs of the FCN-based model.

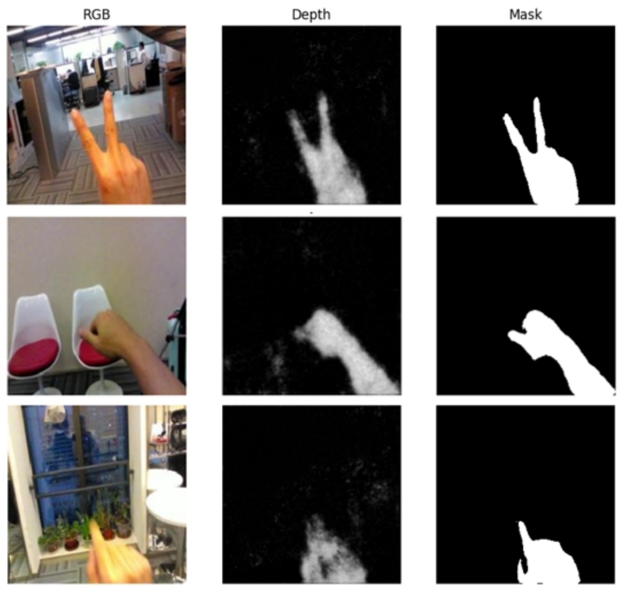

Figure 11. Prediction outputs of the deconvolution network.

5.3. LSTM Network

We also investigated the use of recurrent networks in this project. The rationale was that convolutional network-based architectures like FCN and the deconvolution network are designed to process single images, thus neglecting the temporal relationships between video frames. We hypothesized that incorporating a time-aware network could potentially enhance performance by capturing these temporal dynamics.

Specifically, we modified our deconvolution network by appending two LSTM layers with a hidden size of 4096 between the encoder and the decoder. Although LSTMs are computationally expensive and slower to train due to their sequential nature, they excel at capturing long-term dependencies in sequential data.

However, during our experiments, we observed that training the LSTM network was considerably more challenging compared to the other two CNN-based networks. It exhibited higher sensitivity to hyperparameters, subjects, and scenes, requiring significantly more time for tuning and training. As presented in Table 5, after 160 epochs of training, the best-performing model attained an average score of 0.5335 on the training set and 0.6199 on the test set. This discrepancy might be attributed to the varying performance of the model across different subjects.

	Depth score	**Mask score**
Training	0.4856	0.5814
Validation	0.4745	0.4672
Test	0.4670	0.7728

Table 5. Performance of the LSTM network.

Due to time constraints, we were unable to conduct further exploration with this recurrent network. One potential solution to the optimization issue could be to decrease the size of the hidden layer to facilitate training. Another idea would be to decrease the complexity of the encoder network, possibly by removing a few layers from the VGG-16 network. By simplifying the encoder, we may be able to balance the computational resources allocated to spatial features with the temporal dynamics captured by the LSTM.

6. Conclusion

This project tackled the problem of determining the 3D coordinates of the user's hands. Assuming a fixed global coordinate system, we based our approach on data sourced from a single monocular camera. By dividing the hand-tracking task into 2 subtasks: 2D segmentation and depth estimation, we leveraged well-studied techniques in computer vision. We created a unified model for these tasks, using an encoder-decoder architecture with a shared encoder and separate decoders for each task. We employed fully convolutional networks (FCNs) as a robust baseline for 2D segmentation and developed a deconvolution network for enhanced performance. Finally, we briefly explored the use of an LSTM network to incorporate temporal information, but encountered significant challenges in the training process.

In the end, our deconvolution model achieved an impressive average score of 0.8623 on the test set. As anticipated, the baseline FCN model's performance was limited by its overly simple upsampling structure. However, we were somewhat surprised by the poor performance of the LSTM model, which was likely due to optimization difficulties.

In general, we found depth estimation to be a significantly more challenging problem compared to 2D segmentation across all methods. Another issue we encountered is that the performance of all three models is subject-dependent. This is intuitive, as different subjects may have hands of varying sizes. With a monocular RGB video, it is difficult to determine if hands appear small because they are farther from the camera (indicating greater depth) or simply because they are physically smaller. To address this limitation and enhance the model's generalizability, implementing a calibration mechanism is essential. Therefore, future work should focus on improving depth estimation results and incorporating subject-specific calibration.

7. Contributions

M.Q. processed the data and prepared the dataset. Y.W. implemented and trained the models. Both authors contributed to the design of the models, conducted the experiments, interpreted the results, and finalized the manuscript.

References

[1] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[2] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

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

[4] Armin Masoumian, Hatem A Rashwan, Julián Cristiano, M Salman Asif, and Domenec Puig. Monocular depth estimation using deep learning: A review. *Sensors*, 22(14):5353, 2022.

[5] Ruan Xiaogang, Yan Wenjing, Huang Jing, Guo Peiyuan, and Guo Wei. Monocular depth estimation based on deep learning: A survey. In *2020 Chinese Automation Congress (CAC)*, pages 2436–2440. IEEE, 2020.

[6] Chaoqiang Zhao, Qiyu Sun, Chongzhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, 63(9):1612–1627, 2020.

[7] Vasileios Arampatzakis, George Pavlidis, Nikolaos Mitianoudis, and Nikos Papamarkos. Monocular depth estimation: A thorough review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[8] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. *CoRR*, abs/1704.07813, 2017.

[9] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *Proceedings of the European conference on computer vision (ECCV)*, pages 53–69, 2018.

[10] Liangfu Chen, Zeng Yang, Jianjun Ma, and Zheng Luo. Driving scene perception network: Real-time joint detection, depth estimation and semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1283–1291. IEEE, 2018.

[11] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Yan Yan, Nicu Sebe, and Jian Yang. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4106–4115, 2019.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[15] Yifan Zhang, Congqi Cao, Jian Cheng, and Hanqing Lu. Egogesture: A new dataset and benchmark for egocentric hand gesture recognition. *IEEE Transactions on Multimedia*, 20(5):1038–1050, 2018.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. pages 8024–8035, 2019.

[17] 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.