




## Article

# Training Fully Convolutional Neural Networks for Lightweight, Non-Critical Instance Segmentation Applications

Miguel Veganzones, Ana Ciscal , Eusebio de la Fuente  and Juan Carlos Fraile \* 

Instituto de las Tecnologías Avanzadas de la Producción (ITAP), Escuela de Ingenierías Industriales, Universidad de Valladolid, Paseo Prado de la Magdalena 3-5, 47011 Valladolid, Spain; efuente@uva.es (E.d.l.F.)

\* Correspondence: jcfraile@uva.es

**Abstract:** Augmented reality applications involving human interaction with virtual objects often rely on segmentation-based hand detection techniques. Semantic segmentation can then be enhanced with instance-specific information to model complex interactions between objects, but extracting such information typically increases the computational load significantly. This study proposes a training strategy that enables conventional semantic segmentation networks to preserve some instance information during inference. This is accomplished by introducing pixel weight maps into the loss calculation, increasing the importance of boundary pixels between instances. We compare two common fully convolutional network (FCN) architectures, U-Net and ResNet, and fine-tune the fittest to improve segmentation results. Although the resulting model does not reach state-of-the-art segmentation performance on the EgoHands dataset, it preserves some instance information with no computational overhead. As expected, degraded segmentations are a necessary trade-off to preserve boundaries when instances are close together. This strategy allows approximating instance segmentation in real-time using non-specialized hardware, obtaining a unique blob for an instance with an intersection over union greater than 50% in 79% of the instances in our test set. A simple FCN, typically used for semantic segmentation, has shown promising instance segmentation results by introducing per-pixel weight maps during training for light-weight applications.

**Keywords:** computer vision; convolutional neural networks; deep learning; hand segmentation; semantic segmentation



**Citation:** Veganzones, M.; Ciscal, A.; de la Fuente, E.; Fraile, J.C. Training Fully Convolutional Neural Networks for Lightweight, Non-Critical Instance Segmentation Applications. *Appl. Sci.* **2024**, *14*, 11357. <https://doi.org/10.3390/app142311357>

Academic Editor: Rui Araújo

Received: 21 October 2024

Revised: 28 November 2024

Accepted: 4 December 2024

Published: 5 December 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Real-time hand segmentation information is essential in augmented reality (AR) applications involving human interaction with virtual objects. Additional instance-specific information can further enhance segmentation data to model complex interactions, but can be computationally expensive to obtain. AR has been proved to increase the effectiveness of motor function rehabilitation systems, particularly in upper extremities [1,2]. In this study, we explore a light-weight semantic segmentation technique that preserves some instance information, aiming to power an AR-based upper-limb rehabilitation platform. This instance segmentation approximation allows minimizing the hardware requirements of the target rehabilitation platform, as this information is only occasionally required and is not critical for the proper functioning of the application. Instance segmentation is inherently a more challenging task than semantic segmentation, and is more closely related to object detection than semantic segmentation. Furthermore, in the general case, images can contain an arbitrary number of instances, requiring the use of clever segmentation strategies or complex architectures.

Existing fully supervised instance segmentation methods can be divided into three categories based on number of stages present in the inference pipeline: single-stage, two-stage, and multi-stage methods. Two-stage instance segmentation is the most widely used [3], with Mask R-CNN the most representative work in this category [4]. These pipelines

generally compose object detection with semantic segmentation, and can be divided into top-down or bottom-up methods, depending on the order in which segmentation and detection operations are applied. Examples of top-down methods, which first select regions of interest and then segment, are Sharpmask [5], PANet [6], Mask SSD [7], Mask R-CNN [4], Mask Scoring R-CNN [8], MaskLab [9], and Explicit Shape Encoding (ESE) [10]. Examples of bottom-up methods, which first segment and then detect instances, are DeepCRFs [11], Dynamically Instantiated Network (DIN) [12], Contour Information Deep Watershed Transform (DWT) [13], Pixel Center Prediction Box2pix [14], Depth Information Pixel-level Encoding and Depth Layering (PEDL) [15], Recurrent Pixel Embedding (RPE) [16], Deep Metric Learning (DML) [17]. Example of multi-stage methods include those based on cascade architectures [18–20] and Recurrent Neural Network (RNN)-based approaches such as Recurrent Instance Segmentation (RIS) [21], End-to-End Instance Segmentation (ETE) [22], Recurrent Neural Networks for Semantic Instance Segmentation (RSIS) [23]. Additionally, transformers-based methods like QueryInst [24], SOLQ [25], ISTR [26], and Mask2Former [27] exemplify advanced techniques in this domain.

The use of multi-stage detectors presents a number of disadvantages. The overall performance is constrained by the weakest element of the pipeline, with errors cascading irreversibly through the inference process. Scalability may be compromised, as many multi-stage approaches require computation for each instance detected at some point in the pipeline. Some of these approaches, particularly top-down methods, do not segment images holistically, which reduces the contextual information available during segmentation. Certain detectors, moreover, cannot be trained end-to-end, and may require multiple datasets and complex training schemes. Single-stage detectors avoid these problems but require employing novel techniques to accomplish the complex task of instance segmentation. Some single-stage approaches use parallel branches, in favor of sequential ones, that perform subtasks that are composed into the final segmentation masks, such as You Only Look At Coefficients (YOLACT) [28] and SOLO [29]. Single-stage methods can be further divided into anchor-based and anchor-free methods. Examples of anchor-based methods are InstanceFCN [30], FCIS [31], TensorMask [32], and YOLACT [28], with the latter being the most representative. Examples of anchor-free methods are EmbedMask [33], BlendMask [34], CondInst [35], Polarmask [36], CenterMask [37], SOLO [29], and SOLOv2 [38].

However, these modern solutions required high-end GPUs to achieve very high frame per second (FPS) rates. To reduce the complexity of instance segmentation, a simple approach is imposing an upper limit on the number of instances detected but would inevitably introduce biases and limitations; it would not be scalable, as different datasets would be required to train different versions with different limits. Therefore, this solution was discarded to maintain flexibility, especially for AR rehabilitation platforms that may require interactions involving therapists or additional users besides the patient. The deployed model must run within a real-time application, which imposes a minimum average inference rate to the model of approximately 40 FPS, ideally in regular consumer CPUs.

In light of these constraints, we opted for a light-weight solution. The instance-specific information required for the task is relatively minimal, as basic instance segmentation data are sufficient to model interactions between multiple objects, such as in a rehabilitation scenario. Thus, we chose simple, fully convolutional networks (FCNs) like U-Net and ResNet, integrated with a pixel-weighted loss function to enable segmentation of distinct instances into separate blobs, particularly emphasizing boundary pixels between interacting hands. Morphological operations are employed as a processing step to extract individual instances from the segmentation mask, adding minimal overhead. Our approach effectively substitutes complex instance segmentation models with simpler semantic segmentation networks, prioritizing real-time performance over high accuracy. Although the instance information produced by this method may be of lower quality, it is sufficient for the task at hand, and it bypasses the complications typically associated with instance segmentation.

In this article, we first introduce the materials and methods employed, including the EgoHands dataset, the required image preprocessing, the two proposed FCN architectures

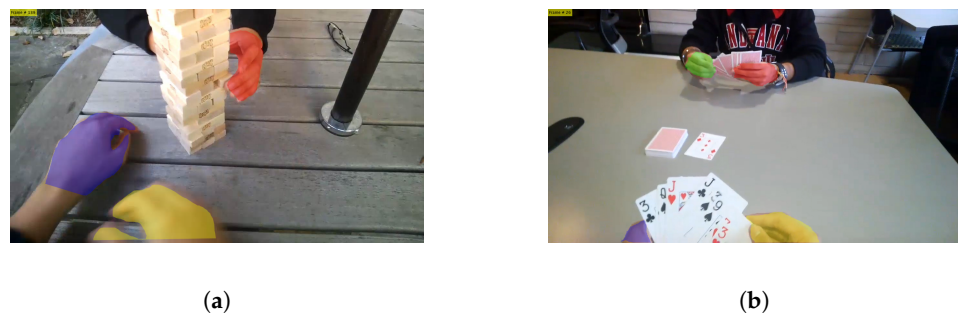
(U-Net and ResNet) along with different loss functions (binary cross entropy, focal loss, and dice loss), activation functions (ReLU and PReLU), and performance metrics used to evaluate the models. Subsequently, we present the results obtained for the two proposed architectures, and upon selecting the optimal architecture, we perform a fine-tuning of model hyperparameters, including dropout rate, loss function, and learning rate scheduler. Finally, we discuss and conclude the results obtained, comparing them with state-of-the-art segmentation results.

## 2. Materials and Methods

This section presents the dataset, and the preprocessing steps required to make it suitable for the proposed approach. It also outlines the neural network architectures to be tested, and the key hyperparameters tuned to optimize the model for the presented task.

### 2.1. Dataset

Our segmentation approach has been evaluated using the Egocentric Hands (Ego-Hands) dataset [39]. The dataset contains 4800 labeled images from 48 videos of complex interactions between two people, recorded from a first-person perspective. The labels are four precise segmentation masks, classified into four possible classes, distinguishing between the left and right hands of the observer and the other person. This representation allows for instance segmentation, semantic segmentation, and object detection, with simple mask manipulation. Both the segmentation masks and the images have a resolution of  $1280 \times 720$  pixels, which is considered high quality for the majority of relevant applications. The 48 videos feature two out of four people, in one of three environments, performing one of four different activities (Figure 1). These different combinations introduce variability in the environments and lighting conditions in which the classifier must learn. This variability suggests that there are relevant invariants in hand detection represented in the data, allowing a network trained with this dataset to learn to ignore irrelevant contextual information and properly generalize.



**Figure 1.** Images from the EgoHands dataset with their segmentation masks. (a) Jenga game in the backyard. (b) Card game at the office.

### 2.2. Image Preprocessing and Data Augmentation

The four independent masks are preprocessed before being combined into a single segmentation mask to ensure that there is at least one background pixel between each instance, which is required to encode hand separation using weight maps. Combining the mask directly leads to two problems. First, there are overlapping masks, where in extreme cases, a single pixel partially belongs to two classes in the original ground truth. Second, some masks are adjacent without actually overlapping.

The dataset masks, with values ranging from 0 to 255, blob interiors have a value of 255, background pixels are encoded with a value of 0, and the borders that separate these have intermediate values. Masks are first binarized, to encode whether a pixel belongs to a hand or not. A threshold of 128 is empirically chosen as a value that produces regular and coherent hand borders. Once the masks are binarized, dilations ( $\oplus$ ) and logical operations AND ( $\wedge$ ), OR ( $\vee$ ), and NOT ( $\neg$ ) will be used to detect and eliminate conflicts between

instance masks. Conflicting regions are obtained from dilated versions of the original masks, rather than the original masks directly; this allows removing conflicts from overlapping and also non-overlapping masks. A round  $5 \times 5$  kernel ( $k$ ) that preserves smooth borders is used to dilate these masks.

$$k = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Let  $A$ ,  $B$ ,  $C$ , and  $D$  denote the four segmentation masks associated with an image. Their dilated versions, denoted as  $A^d$ ,  $B^d$ ,  $C^d$ , and  $D^d$ , are calculated as follows:

$$A^d = A \oplus k \quad (1)$$

$$B^d = B \oplus k \quad (2)$$

$$C^d = C \oplus k \quad (3)$$

$$D^d = D \oplus k \quad (4)$$

Then, the intersection  $R$  of any pair of these dilated masks and their dilation  $R^d$  are obtained,

$$R_0 = A^d \wedge (B^d \vee C^d \vee D^d) \quad (5)$$

$$R_1 = B^d \wedge (C^d \vee D^d) \quad (6)$$

$$R_2 = C^d \wedge (D^d) \quad (7)$$

$$R = R_0 \vee R_1 \vee R_2 \quad (8)$$

$$R^d = R \oplus k \quad (9)$$

Finally, the processed mask  $M$  is the logical union of  $A$ ,  $B$ ,  $C$ , and  $D$ , representing the original data, excluding the dilated conflicting areas  $R^d$ ,

$$M = (A \vee B \vee C \vee D) \wedge \neg R^d \quad (10)$$

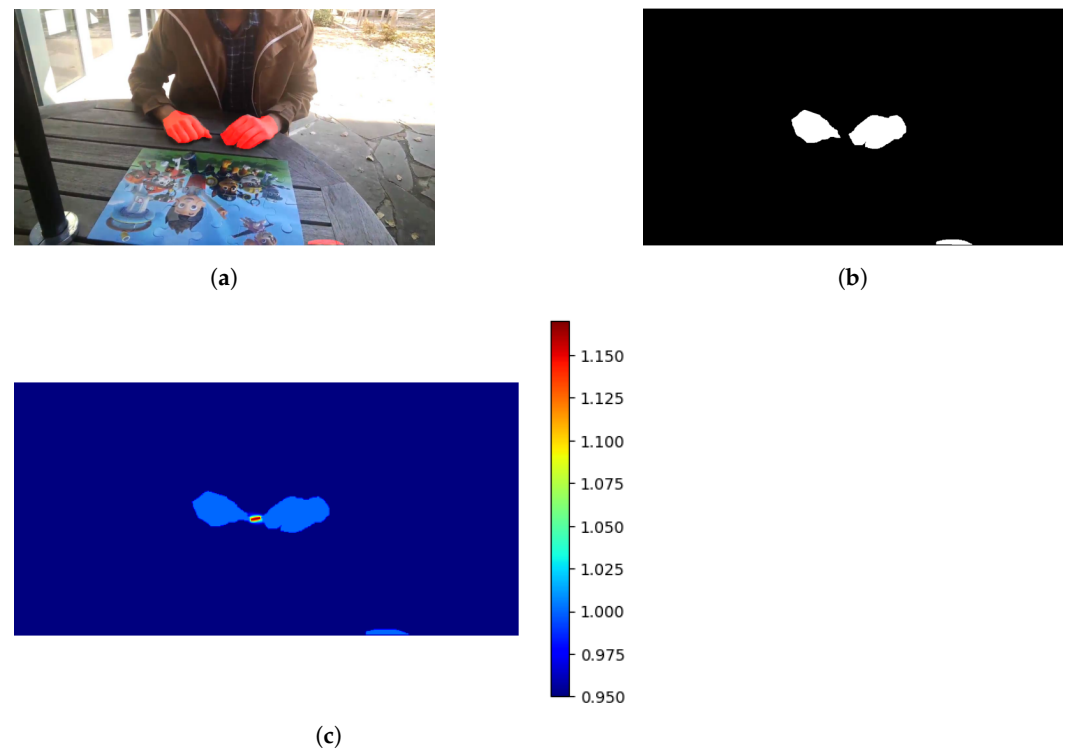
Thus, the conflicting areas become part of the background. Images are then re-scaled to a resolution of  $640 \times 368$  pixels to increase throughput of the final model. Pixel weight maps are then used on top of a per-pixel loss function to emphasize the importance of pixels separating two blobs, forcing the network to learn this separation (Figure 2). Increasing weight in certain background regions further amplifies class imbalance, hence the base weight of background pixels has been reduced to 0.95 compared to 1.0 for hand pixels. This value of 0.95 was experimentally determined to achieve a balanced rate of false positives ( $FP$ ) and false negatives ( $FN$ ) in an initial test. The weight of background pixels is, therefore,

$$w(x) = w_c(x) + w_0 \cdot \exp\left(-\frac{(d_1(x) + d_2(x))^2}{2\sigma^2}\right) \quad (11)$$

where  $w_c$  represents the base weight of background pixels,  $w_0$  is a multiplier,  $\sigma$  is a reference distance in pixels, and  $d_1$  and  $d_2$  are distances in pixels from the current pixel to the nearest two hands. The chosen values were  $w_c = 0.95$ ,  $w_0 = 12.0$ , and  $\sigma = 5.0$ . The exponentially decaying nature of the weight function ensures that the value over the base weight is only significant when both  $d_1$  and  $d_2$  are sufficiently small. These parameters lead to a maximum pixel weight of approximately 13 times the base weight.

Datasets used in computer vision typically contain a limited number of images because their acquisition process is often non-automatable. This is particularly true for segmen-

tation tasks, which require highly precise labels. Expanding a segmentation dataset is straightforward by applying simple transformations to images and their masks, thereby enhancing data utilization efficiency [40]. In this case, 2242 new images were generated from the initial set of 3200 training images. Among these, 65% are horizontally flipped, 55% have applied zoom and small rotations, 40% exhibit color variations, and 30% have per-pixel Gaussian noise added.



**Figure 2.** Example image with its segmentation mask and associated pixel weight map. (a) Original image and its segmentation mask. (b) Segmentation mask. (c) Pixel weight map.

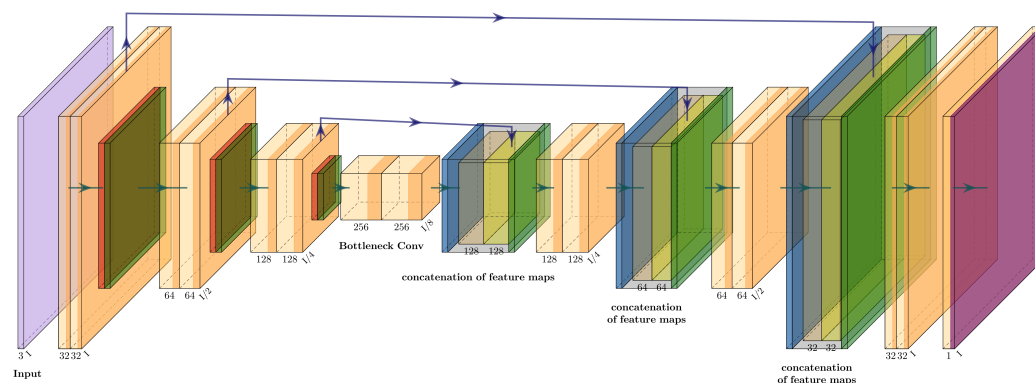
### 2.3. Network Architectures

This study compares two simple yet modern FCNs: U-Net and ResNet. The U-Net architecture is a promising candidate for this application due to the positionally precise image information that is accessible to the decoder when making predictions, which may facilitate the accurate classification of precise boundaries. The ResNet architecture can be viewed as a more modern iteration of the U-Net, where long skip connections are replaced with short ones, and concatenation is replaced by addition. This architectural change has deep implications, skip connections are no longer used to carry positionally precise information to the decoder, but to center the network around the identity mapping.

#### 2.3.1. U-Net

The first architecture tested is a standard U-Net (Figure 3), containing three convolutional blocks in both the encoder and the decoder, connected by a bottleneck and the characteristic skip connections of the U-Net architecture [41]. Each block in the encoder has two convolutional layers with ReLU activation and a max-pooling operation to reduce the resolution of the input to the next block. In the decoder, a transposed convolution with a stride greater than one is used as an upsampling method to increase the resolution of the intermediate results, which are concatenated with the output of the convolutional layers of the corresponding encoder block. Then, two convolutional layers are used as processing units that also reduce the depth channels of the intermediate results. Skip connections grant the decoder access to positionally precise, lightly processed image information, to use jointly with the heavily processed, less positionally precise, higher level information coming from the bottleneck. This makes the U-Net a very appealing architecture for this

application as positionally precise information might be required to produce reliable results. The encoder and decoder are connected by a bottleneck, where the most complex, high-level, and less positionally accurate information should be obtained. All convolutions use  $3 \times 3$  kernels, as is customary in FCNs. Other sizes were tested, but the best results were obtained with small kernels. The network has a total of 2.1 million parameters and dropout is included as regularization method.



**Figure 3.** U-Net architecture. Color used: lilac—input; light orange—convolutions; dark orange—ReLU activation; red—MaxPooling; green—dropout; dark blue—upsampling; gray—concatenation; yellow—concatenated block; purple—sigmoid activation.

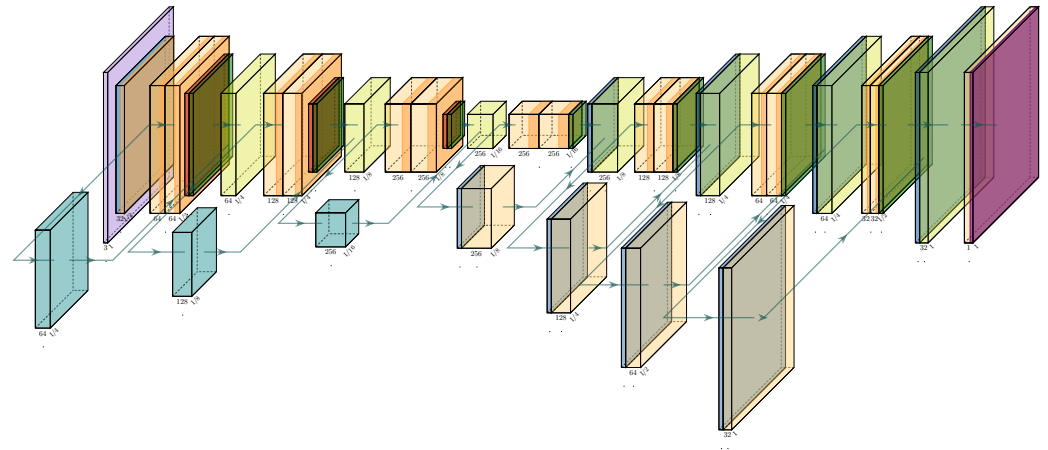
### 2.3.2. ResNet

The ResNet [42] used is also standard (Figure 4), except for the unconventional inclusion of dropout. The first layer is a convolution with a stride greater than 1, which reduces the resolution of the input image while adding depth channels and initiating processing. The encoder contains three blocks of convolutional layers, max-pooling, and dropout, similar to the previous architecture. Additionally, sum operations are added between the input and output of each residual block. The residual connections in the encoder are implemented with a convolutional layer using strided convolutions, which adjusts the size of one of the tensors to be summed. The decoder has three residual blocks symmetrical to the encoder, using transposed convolutions with a stride of 2 as the upsampling method, as seen earlier. In this case, the residual connections are made with an upsampling layer and a convolutional layer. The bottleneck is another convolutional residual block, analogous to the bottleneck found in U-Net. ReLU activations and primarily  $3 \times 3$  convolutional kernels are used, except in the residual connections where  $1 \times 1$  kernels are used. These  $1 \times 1$  kernels are conventionally used to adjust the resolution and depth of the inferred feature tensors. The network has a total of 3.1 million parameters, while requiring the same amount of memory than the presented U-Net. Batch normalization along with dropout are used as regularization methods.

### 2.4. Loss Function

Three loss functions were evaluated to optimize network performance: Binary Cross Entropy (BCE), Focal Loss (FL), and Dice Loss. The selection of the loss function enables fine tuning the balance between precision and recall, preserving boundaries while maximizing IoU. BCE in combination with pixel-weight maps can effectively produce well-defined segmentation and individual mask for each hand. However, its equal treatment of foreground and background pixels can lead to class imbalance, as background pixels dominate in number. This imbalance, though partially mitigated by the high confidence typically assigned to background predictions, could be further optimized. FL extends BCE by introducing parameters  $\alpha$  and  $\gamma$  to emphasize harder-to-classify pixels and address class imbalance more effectively [43]. By adjusting these parameters, segmentation performance is enhanced, especially in challenging regions. Dice Loss, also designed for imbalanced

data, prioritizes foreground regions by directly optimizing the similarity between predicted and ground truth masks [44]. It calculates a differentiable Dice similarity coefficient [45] to emphasize true positives ( $TP$ ) while minimizing the impact of class imbalance [46]. To counteract prediction noise near the decision threshold, predictions were polarized using a sigmoid function, yielding marginally improved results.



**Figure 4.** ResNet architecture. Colors used: lilac—input; light blue—strided convolution; light orange—convolutions; dark orange—BatchNorm and ReLU activation; red—MaxPooling; green—dropout; yellow—sum; dark blue—upsampling; purple—sigmoid activation.

### 2.5. Activation Function

The ReLU activation function is widely used due to its simplicity, efficient gradient computation, and ability to mitigate the vanishing gradient problem. However, a drawback is that neurons producing consistently negative outputs can become inactive during training. To address this limitation, the PReLU activation function introduces a parameter  $\alpha$ , which allows negative outputs to take on small negative values rather than being zero [47].  $\alpha$  adapts during training and can be unique to each weight or shared across multiple weights. In this case,  $\alpha$  is shared across each feature channel of every layer, efficiently increasing model complexity with a modest addition of parameters (around 3000), enhancing its ability to learn and represent complex patterns.

### 2.6. Performance Metrics

Intersection over Union (IoU) is a commonly used metric for evaluating segmentation algorithms. It measures the overlap between predicted and ground truth segmentation masks. Additionally, accuracy, precision and recall were also used as performance metrics. Accuracy represents the proportion of correctly predicted pixels (both foreground and background). Precision indicates the percentage of predicted foreground pixels that are correctly classified, highlighting the model's conservativeness by minimizing false positives ( $FP$ ). Recall represents the percentage of ground truth pixels correctly identified by the model, reflecting its ability to capture relevant information by minimizing false negatives ( $FN$ ). While maximizing both precision and recall is ideal, precision is prioritized in this application due to the high cost of  $FP$ , whereas  $FN$  poses minimal risk. Note that balancing these metrics is challenging, as they directly impact the demanding IoU score.

The instance segmentation performance of the model cannot be adequately measured using the AP score [48], as the network is designed for semantic segmentation. Hence, we developed a custom metric, the Centroid Correlation Coefficient (CCC), tailored to estimate how accurately individual masks are obtained for each ground truth instance, crucial for approximating instance segmentation through semantic segmentation. The CCC metric ideally achieves a value of 1.0, typically ranges between 0.95 and 0.3, but can be negative in highly fragmented segmentations. The CCC metric measures the average deviation between the centroids of inferred blobs and ground truth blobs, assuming each

blob represents an instance. It penalizes deviations from the ideal scenario, where each instance corresponds to a single blob, and accounts for mismatches in the number of blobs and cases where blobs lack a direct counterpart.

The CCC pseudocode implementation is presented in Algorithm 1. If the number of blobs is the same in both segmentations, they are paired such that the total distance within the pairs is minimized. If the number of blobs does not match, each blob in the segmentation with more blobs is associated with the nearest blob in the other segmentation. The average distance is calculated as the sum of these distances divided by the number of blobs in the segmentation with fewer blobs. Hence, this approach penalizes a blob less when it is closer to any blob in the other segmentation and penalizes based on the disparity of blob count. The average distance is normalized against the maximum possible distance (the image diagonal), yielding a result between 0 and 1 when the same number of masks matches, and greater than 0 otherwise. Finally, the square root of this measure is computed to expand the most common range, and is then subtracted from 1 to obtain a value less than 1.

---

**Algorithm 1** Centroid correlation coefficient pseudocode implementation.

---

*input* : 2DArray target\_seg(target segmentation),  
2DArray inferred\_seg(inferred segmentation)

*output*: ccc (Centroid Correlation Coefficient)  $\in (-\infty, 1]$

```

target_c ← find_centroids(target_seg)
pred_c ← find_centroids(inferred_seg)

mean_dist ← 0
if len(target_c) = len(inferred_c) then
    // Pair blobs such that total distance is minimized
    dist_matrix ← distance_matrix(target_c, inferred_c)
    (row_ind, col_ind) ← linear_sum_assignment(dist_matrix)
    mean_dist ← mean(dist_matrix[row_ind, col_ind])
else then
    // Associate each blob with its closest counterpart
    short, long ← sorted([target_c, pred_c], key=lambda x : len(x))
    total_dist ← sum([min([dist(p, q) foreach q in short]) foreach p in long])
    mean_dist ← total_dist / len(short)
end

// Normalize the mean distance with its diagonal
diag ← sqrt(height(target_seg)2 + width(target_seg)2)
normalized_dist ← mean_dist / diag

// sqrt is used to dilate the usable range near CCC values of 1
ccc ← 1 - sqrt(normalized_dist)
return ccc

```

---

### 3. Results

The choice of architecture was initially made between U-Net and ResNet, two widely recognized FCNs. Proper regularization is crucial for FCNs to prevent overfitting. Dropout has been widely used as an effective regularization technique [49], as batch normalization [50]. However, recent works suggest that the standard dropout can be ineffective or even detrimental to FCN training [51]. Due to the lack of mathematical intuition for dropout in FCNs, which can make it unpredictable and challenging to adjust during training, dropout is carefully considered in the architecture selection process. Once the optimal architecture is selected, its hyperparameters are tuned, including activation function, loss function, and learning rate schedule.



### 3.1. Architecture Selection

Both models were trained using the same data and hyperparameters to ensure a fair comparison, despite introducing some bias in the architecture selection. Augmented images were excluded from the set, as the primary goal was to identify the best architecture rather than achieve precise results at this stage. A pixel-weighted BCE loss function was employed, combined with the Adam optimizer and the He-Normal kernel initialization [42] for robust performance without additional adjustments [52]. The models used a constant learning rate of 0.001, ReLU activation, and a batch size of 2 was used due to memory constraints. All models were trained over 30 epochs, which preliminary tests showed to be adequate for validation IoU convergence without significant overfitting.

Table 1 presents the average performance metrics from three training runs for both ResNet and U-Net models, with dropout rates varying from 0.0 to 0.7. U-Net encountered convergence issues at dropout rates above 0.3, so results for these rates were excluded. These results highlight ResNet's potential to outperform U-Net, which can be attributed to its 50% higher parameter count, despite both models requiring equivalent GPU memory and ResNet being faster. This greater parameterization enables ResNet to model more complex transformations inherent in the dataset, albeit with increased susceptibility to overfitting. In terms of inference speed, U-Net achieves an unoptimized rate of 19 FPS on  $640 \times 368$  resolution images when tested on an Intel 11600KF CPU, whereas ResNet achieves 21 FPS, further underscoring its efficiency despite its larger size. Based on these findings, ResNet is selected as the preferred architecture for subsequent experiments.

The ResNet architecture was retrained from scratch with augmented samples, testing dropout rates from 0.0 to 0.4. Using augmented samples increases the importance of regularization methods like dropout, as augmented images do not contain purely new information, introducing biases into the training. The averaged results of three training runs for this training are shown in Table 2. The highest IoU was achieved at a dropout rate of 0.3, making it the selected model for further fine tuning.

**Table 1.** Performance of U-Net and ResNet architectures based on the amount of dropout.

Dropout Rate	ResNet		U-Net	
	IOU	Precision	IOU	Precision
0.0	0.783	0.901	0.763	0.898
0.1	0.784	0.903	0.758	0.893
0.2	<b>0.786</b>	<b>0.908</b>	0.721	0.878
0.3	0.785	0.905	0.733	0.887
0.4	0.780	0.899	—	—
0.5	0.761	0.894	—	—
0.6	0.749	0.889	—	—
0.7	0.722	0.878	—	—

Bold numbers indicate the best results obtained.

**Table 2.** ResNet performance results based on the amount of dropout using augmented images for training.

Dropout Rate	IOU	Precision	Recall	CCC
0.0	0.789	0.907	0.861	0.813
0.1	0.790	<b>0.915</b>	0.855	0.821
0.2	0.791	0.904	<b>0.864</b>	0.817
0.3	<b>0.795</b>	0.913	0.861	<b>0.822</b>
0.4	0.783	0.908	0.851	0.817

Bold numbers indicate the best results obtained.

### 3.2. Hyperparameter Optimization

After selecting ResNet as the optimal architecture, hyperparameter optimization was performed to enhance model performance. This process aimed to add complexity where

needed, align training with the target objective, and increase search efficiency within the model's solution space. Various loss functions and activation functions were tested to better address the task, while learning rates were adjusted across epochs to optimize search efficiency. The same sets of videos used for architecture selection were employed for the train-validation-test split (32-8-8), with additional augmented images for training. Models were trained for 30 epochs with a batch size of 2, using the Adam optimizer [52] and the He-Normal kernel initialization method [42], continuing the previously established approach.

### 3.2.1. Activation Function

The PReLU activation function with a single parameter per depth channel was tested along with the ReLU activation function. The averaged results from three training runs using ReLU and PReLU are shown in Table 3. The PReLU increased the model's unoptimized inference time from 52 to 68 ms/image, representing a 40% increase. Training time also extended from 5.5 h to 6.7 h, a 20% increase. While the accuracy gains achieved may not justify the slower inference speed in some applications, this new model has been chosen to be optimized further as results suggest that our original model is indeed parameter-constrained and performance will be improved at a later stage.

**Table 3.** Performance of the ResNet architecture using ReLU and PReLU activation functions.

Activation Function	IOU	Precision	Recall	CCC
ReLU	0.795	0.913	<b>0.861</b>	0.822
PReLU	<b>0.799</b>	<b>0.919</b>	0.860	<b>0.830</b>

Bold numbers indicate the best results obtained.

### 3.2.2. Loss Function

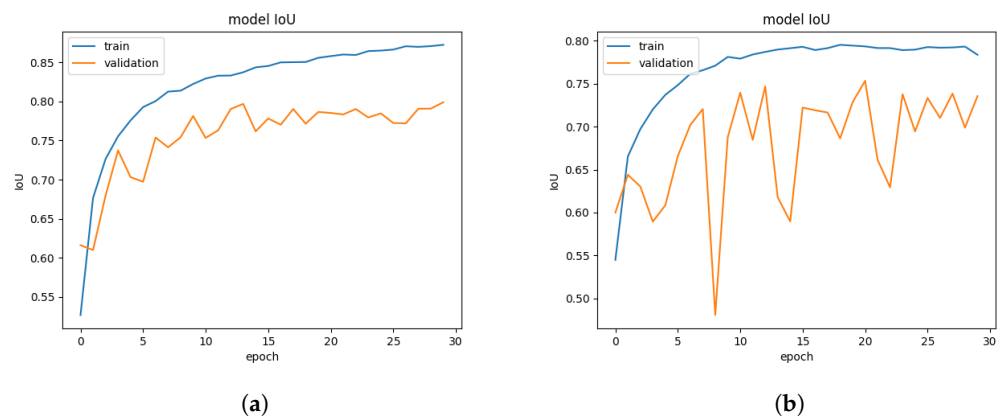
The resulting model has been retrained using DICE Loss and several versions of FL. The average results from three trials are shown in Table 4. The best performance was achieved using FL, with the optimal parameters found  $\alpha = 0.5$  and  $\gamma = 1.1$ . These results are in line with expectations, showing slight improvements over BCE due to FL's greater versatility. Although FL slightly increases the training time due to additional computations, it does not impact the model's inference speed, as it does not alter the model architecture.

Despite the theoretical suitability of Dice Loss for this application, the results indicate poor performance. Dice's stringent nature may lead to noisy evaluations during training, complicating convergence to precise results. Figure 5 illustrates the significant amount of noise observed in the validation IoU curve. This noise likely arises from Dice's sensitivity to small perturbations and its difficulty in adjusting finely to pixel-level details. Pixel weight maps and pseudo-binarization of the inferred segmentations may contribute to the significant noise in the model's evolution during training. Pixels with high weights in the weight maps are inherently challenging, and will tend to be predicted with low confidence. Pseudo-binarization polarizes these predictions, increasing their confidence for better or worse, leading to high variance in the final value with small changes in inferred confidence. Nevertheless, using pseudo-binarization reduces noise from most pixels, and weighting pixels with weight maps is necessary for this application.

**Table 4.** Results obtained with various loss functions in the ResNet architecture with PReLUs.

Loss Function	IOU	Precision	Recall	CCC
BCE (FL $\alpha = 0.5; \gamma = 0$ )	0.799	0.919	0.860	0.831
FL ( $\alpha = 0.5; \gamma = 1.0$ )	0.786	0.886	0.876	0.804
FL ( $\alpha = 0.6; \gamma = 1.0$ )	0.796	0.871	<b>0.903</b>	0.794
FL ( $\alpha = 0.5; \gamma = 1.1$ )	<b>0.802</b>	0.908	0.874	<b>0.834</b>
FL ( $\alpha = 0.5; \gamma = 1.2$ )	0.793	<b>0.920</b>	0.853	0.816
DICE Loss	0.754	0.837	0.885	0.746

Bold numbers indicate the best results obtained.



**Figure 5.** IoU evolution for a ResNet with PReLU using (a) BCE and (b) DICE Loss.

### 3.2.3. Learning Rate Schedule

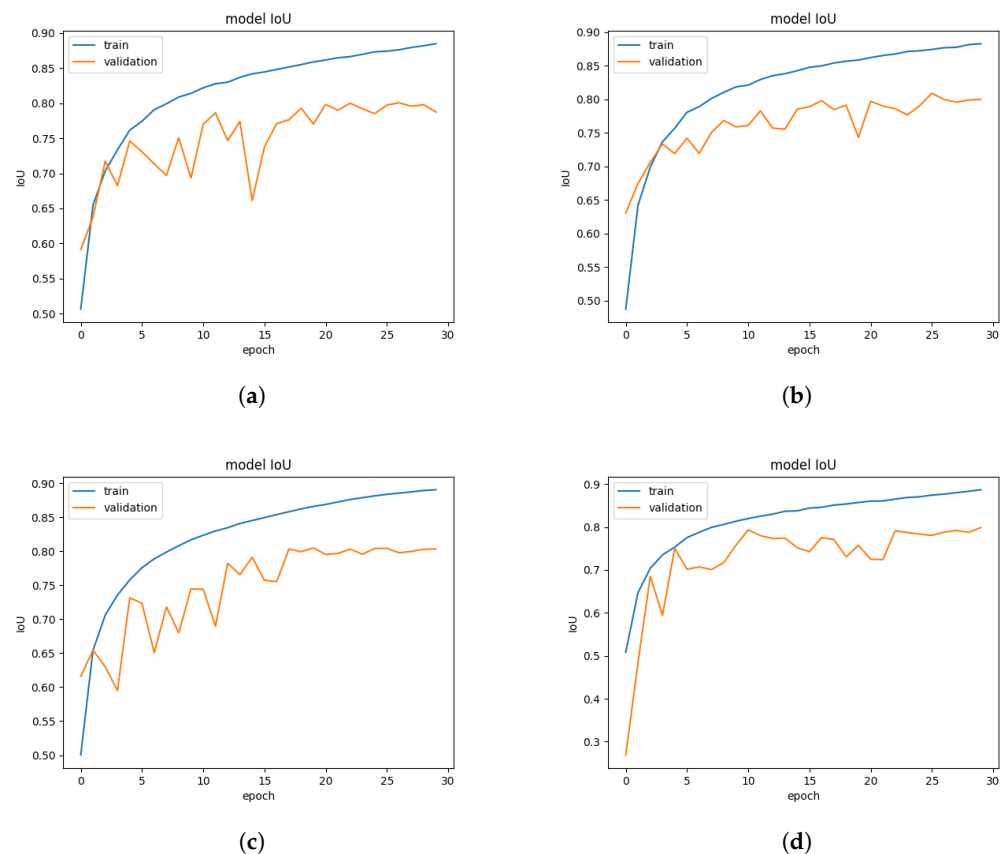
In the early training stages, large learning rate changes help the model explore a wide range of solutions, whereas, as training progresses, reducing fluctuations is crucial for stable convergence. To achieve this, varying the learning rate over epochs is an effective approach. A monotonically decaying learning rate starts high for rapid exploration and gradually decreases for stable convergence. Five learning rate decay schedulers were tested: constant, linear, exponential, cosine, and square root. Performance results are detailed in Table 5, generally showing improvements compared to the constant learning rate. Notably, all schedulers significantly reduced validation IoU noise in later epochs, enhancing stability and convergence, as illustrated in Figure 6.

**Table 5.** Performance metrics obtained using different learning rate schedules, detailing the schedule type, the initial learning rate ( $LR_0$ ), and final learning rate ( $LR_{end}$ ).

Learning Rate Schedule			IoU	Precision	Recall	CCC
Type	$LR_0$	$LR_{end}$				
Constant	0.0010	0.0010	0.802	0.908	0.874	0.834
Linear	0.0017	0.0006	0.795	0.912	0.862	0.831
Exponential	0.0017	0.0007	0.802	0.907	<b>0.877</b>	0.836
Cosine	0.0018	0.0003	<b>0.804</b>	0.914	0.872	<b>0.839</b>
Square root	0.0017	0.0004	0.803	<b>0.919</b>	0.865	0.836

Bold numbers indicate the best results obtained.

Among these, the cosine scheduler demonstrated the most promising results, consistently outperforming a constant learning rate. Its combination of early exploration with a final precise convergence phase proved highly effective. Consequently, it was selected as the preferred model configuration after straightforward hyperparameter optimization. In practice, while further iterations could refine parameters marginally, diminishing returns suggest that more parameter optimization would imply significant overfitting to the training data.



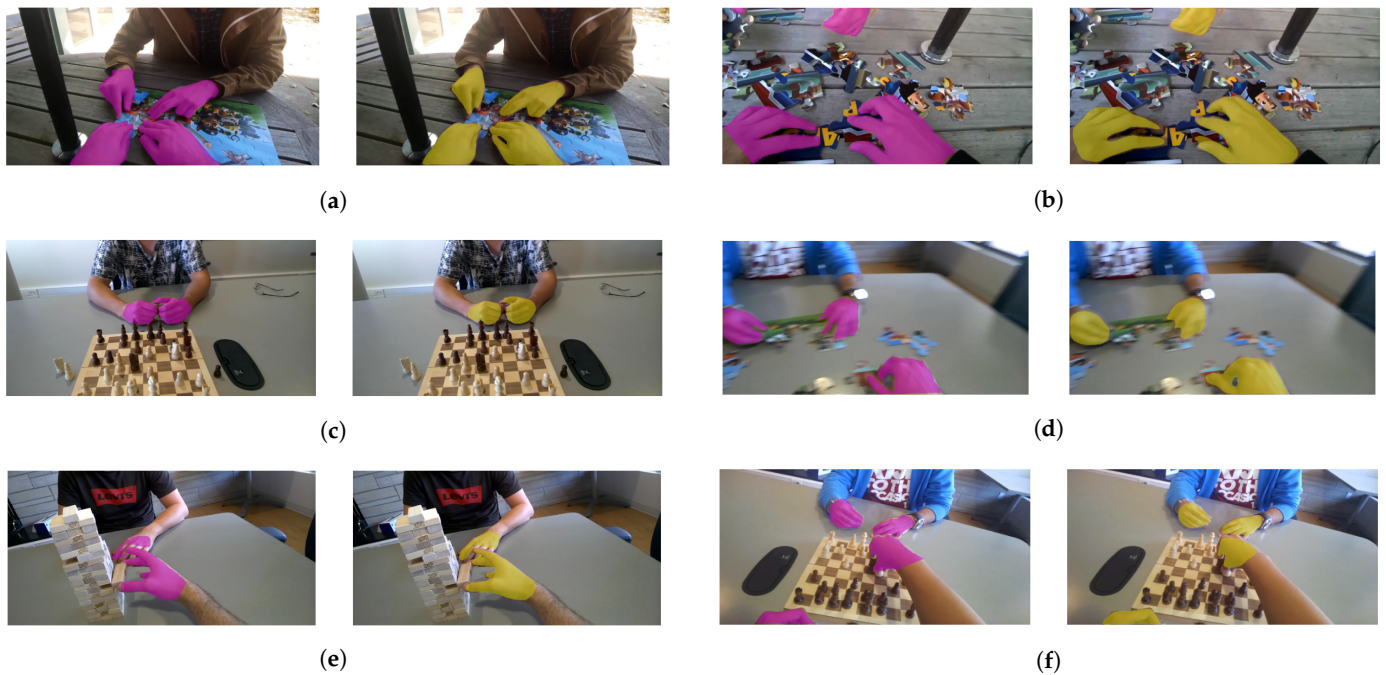
**Figure 6.** Training and validation IoU evolution with four different learning rate schedules: (a) Linear decay, (b) exponential, (c) cosine, and (d) square root.

### 3.3. Visual Analysis

The results achieved are generally of high quality; the model successfully provides independent masks for each hand, delivering precise and well-defined segmentations. Figure 7 displays six well-segmented images from the test dataset, demonstrating that independent masks are effectively obtained even under demanding conditions. This indicates that the model has successfully learned to maintain a significant separation between distinct instances. In even more complex scenarios involving overlapping hand instances, as illustrated in Figure 7e,f, the model continues to perform effectively, delivering accurate independent masks for each instance. This further underscores the model's capability to handle intricate segmentation tasks with high precision. As a reference, the model successfully generated a unique blob for an instance with an IoU above 50% in 79% of cases in the test set. However, this separation has not been achieved in all cases; examples of such failures are shown in Figure 8.

There is a clear trade-off between instance separation and segmentation quality; this complicates the segmentation task and diminishes the expected IoU. This issue is further exemplified by the observed low-quality segmentations, as shown in Figure 9. While the model effectively maintains distinct instance boundaries, its performance is significantly impaired, with an IoU below 0.6. Specifically, some instances are not recognized (Figure 9a–c, while other non-existent instances are erroneously identified (Figure 9d). Hence, IoU no longer accurately reflects the intended objective due to its sensitivity to boundary preservation. As a result, CCC was designed to serve as a complementary metric in evaluating the fulfillment of the requirements. Overall, images in Figure 7 reveal that the model tends to predict smaller areas than the ground truth, particularly around edges. This observation confirms that the model tends to make relatively conservative predictions, and aligns with the observed precision and recall values. This design choice aimed to ensure

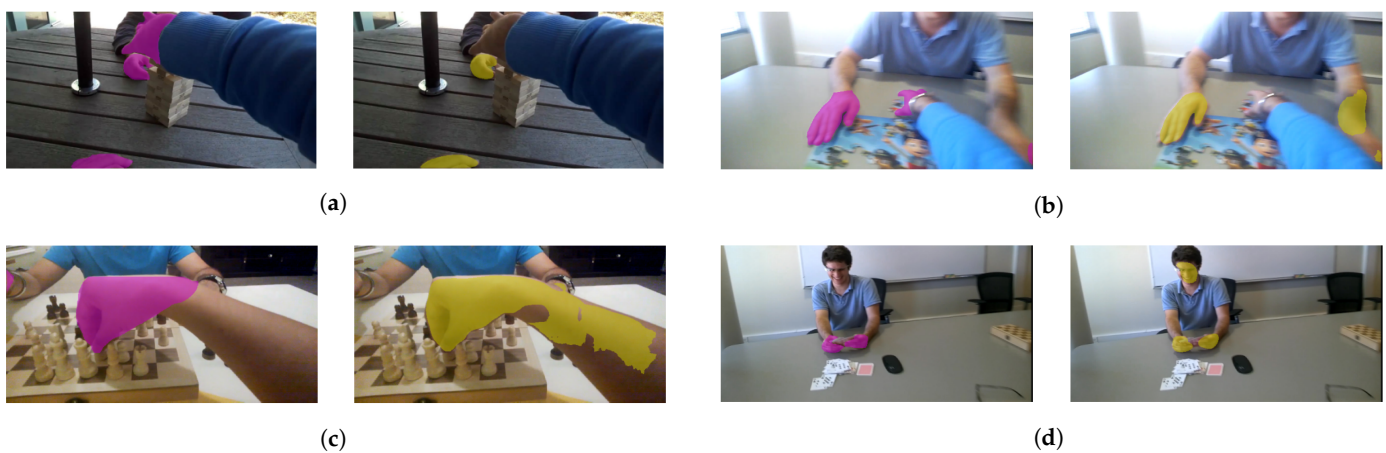
disjoint instance masks, but required balancing precision and recall to avoid significantly reducing IoU.



**Figure 7.** Examples of successful semantic segmentations ( $IoU > 0.8$ ) in challenging scenarios: (a) multiple hands, (b) complex background, (c) overlapping hands, (d) blurry image, (e,f) overlapping hand instances. Ground truth is depicted in magenta, while inferred masks are shown in yellow.



**Figure 8.** Examples of segmentations with a high IoU ( $IoU > 0.8$ ) but that did not provide a distinct mask for each instance: (a) multiple hands, (b) overlapping hand instances. Ground truth is depicted in magenta, while inferred masks are shown in yellow.



**Figure 9.** Examples of low-performance segmentation ( $IoU < 0.6$ ) in challenging cases due to various factors. (a) Poor lighting. (b) Blurry image. (c) Two skin tones. (d) Person at a distance. Ground truth is depicted in magenta, while inferred masks are shown in yellow.

#### 4. Discussion

The integration of the proposed pixel-weight map, combined with morphological operations, allows common semantic segmentation networks to infer instance-level information during runtime without additional computational costs. Designed to support AR rehabilitation applications, this approach achieves real-time CPU-based semantic segmentation inference of multiple hands while preserving instance information. The method was validated on two FCN architectures, ResNet and U-Net, using the EgoHands dataset. Although U-Net is recognized for its ability to retain positional precision, ResNet outperformed it, likely due to its network topology, specifically the modern skip connections and higher parameter count. Li et al. [53] demonstrated that residual connections contribute to convexifying the search space, which significantly enhances training by producing smoother, more optimizable surfaces. This effect helps explain the performance differences between U-Net and ResNet, particularly addressing the convergence issues observed with U-Net (Table 1).

The original ResNet architecture was further optimized by tuning various hyperparameters. The optimal results ( $IoU = 0.804$ ;  $precision = 0.914$ ;  $recall = 0.872$ ;  $CCC = 0.839$ ) were obtained by employing a dropout rate of 0.3, integrating the PReLU activation function, utilizing the FL with  $\alpha = 0.5$  and  $\gamma = 1.1$ , and employing a monotonically decreasing cosine scheduler. The hyperparameters tested aim to enhance specific aspects of training or the model itself that have been deemed suboptimal. Based on the conducted tests and obtained results, the model's performance limit hovers around 0.8 IoU, and consistently surpassing this threshold may require significantly increasing the model's complexity. This can be achieved by adding more layers or increasing feature channels per layer, but constraints imposed by available GPU memory necessitate a more efficient approach. This was finally achieved by using PReLUs, that very efficiently increase model complexity. Introducing FL then allowed tuning the model for this specific application by adjusting precision and recall, which significantly enhanced the preservation of instance-level information (Table 4). Significant fluctuations appear in the validation results. These instabilities were mitigated by replacing the constant learning rate with a learning rate scheduler, which effectively guided the model towards convergence by progressively reducing the learning rate (Figure 6).

To provide context for the performance of the proposed model, Table 6 compares its results with those achieved by other authors using the EgoHands dataset [39]. Bambach et al. [39] reported an IoU of 0.556 for hand detection and segmentation, employing a sliding-window-based CNN approach. Khan and Borji [54] utilized RefineNet for hand segmentation, achieving IoU, precision, and recall scores of 0.814, 0.879, and 0.919, respectively. Wang et al. [55] proposed a method leveraging an RNN for hand segmentation, reporting IoU, precision, and recall values of 0.873, 0.935, and 0.924. Similarly, Li et al. [56] presented a video-based hand segmentation technique using optical flow, achieving an IoU of 0.872. Cai et al. [57] reported an IoU of 0.466. Tsai et al. [58] achieved IoU, precision, and recall values of 0.902, 0.954, and 0.940, respectively. Finally, Lin et al. [59], using RefineNet, reported an IoU of 0.824. Regarding processing efficiency, Li et al. [56] reported a latency of 20.28 ms (equivalent to 49 FPS) using a single NVIDIA P100 GPU. Tsai et al. [58] achieved 200 FPS on an RTX 2080 Ti GPU. Lin et al. [59] recorded a processing time of 50.5 ms on a GeForce RTX 2080 Ti, translating to 19.8 FPS. Roy et al. [60] achieved 65.4 FPS on an NVIDIA GeForce GTX 1080 Ti GPU.

Although the proposed model falls behind some state-of-the-art methods in terms of performance, it successfully meets its primary objective of real-time performance. The model delivers a fast inference time of just under 23 ms per 640x368 image (equivalent to 44 FPS) on an Intel Core i5-11500KF. This demonstrates that, while performance may not be on par with the best-performing models, the optimized speed is a significant accomplishment considering the approach taken to address hardware limitations. On the other hand, while Khan and Borji [54] achieved a slightly superior IoU, their model exhibited lower precision, as they predicted foreground regions more liberally, leading to larger segmentation masks. This approach, while improving overall metrics, is less suitable for

the considered application, where maintaining instance separation requires conservative predictions. Consequently, our model prioritizes precision over recall, striking a balance that maintains acceptable IoU while ensuring effective instance separation. Nevertheless, this method is unable to accurately address instances where occlusions divide an object into multiple fragments, and will struggle if objects to detect are of small size relative to the image. This means that the quality of the instance information obtained through this method is application specific and can be temporally inconsistent. Dedicated instance segmentation networks should be used if instance information is crucial for the overall correctness of the algorithm. However, these issues are not present in the current application, making this approach a preferable solution to using more complex and computationally costly instance segmentation networks.

**Table 6.** Comparison with state-of-the-art models.

Model	IoU	Precision	Recall	FPS
Bambach et al. [39]	0.556	-	-	-
Khan and Borji [54]	0.814	0.879	0.919	-
Wang et al. [55]	0.873	0.935	0.924	-
Li et al. [56]	0.872	-	-	49 <sup>1</sup>
Li and Kitani [61]	0.478	-	-	-
Cai et al. [57]	0.466	-	-	-
Tsai et al. [58]	0.902	0.954	0.940	200 <sup>2</sup>
Lin et al. [59]	0.824	-	-	19.8 <sup>3</sup>
Roy et al. [60]	0.902	0.982	0.968	65.4 <sup>4</sup>
Our model	0.804	0.914	0.872	43.7 <sup>5</sup>

Hardware setup: <sup>1</sup> NVIDIA P100 GPU, <sup>2</sup> NVIDIA RTX 2080 Ti GPU, <sup>3</sup> NVIDIA RTX 2080 Ti, <sup>4</sup> NVIDIA GTX 1080 Ti GPU, <sup>5</sup> Intel i5-11500KF CPU.

While successful in achieving the main goal, it is important to note that instance segmentation in this manner may not be suitable for applications with different constraints or more intricate object boundaries. The approach strikes a balance between semantic and instance segmentation tasks, addressing the challenge of segmenting multiple instances of hands within images effectively. Future research could explore further refinements to enhance boundary delineation and scalability across diverse datasets and object types, thereby extending the applicability of the proposed methodology in broader computer vision tasks. The obtained model has margin for improvements in segmentation performance. Further fine tuning could be performed by testing more modern optimizers like AdamW [62]. Another strategy to be tested is to add a third category (border) other than background and foreground, which was initially rejected for being a discrete version of the presented approach, where the arbitrary definition of the boundary may have an observable effect in the resulting model. Additionally, evaluating the model on additional hand segmentation datasets, such as the Georgia Tech Egocentric Activities (GTEA) dataset [63], would provide further insight into its generalization capabilities across diverse scenarios and environments.

## 5. Conclusions

Precise results comparable to state-of-the-art segmentation algorithms have been achieved with the proposed segmentation strategy. The primary objective of this work was to explore the feasibility of using simple neural network architectures typically employed in semantic segmentation to achieve a reasonable approach to instance segmentation. By preprocessing the dataset to identify critical pixels separating different instances, the model was trained to prioritize these pixels, facilitating the generation of independent segmentation masks for each instance. Additionally, the proposed method does not incur any computational overhead during inference, minimally complicates model design and training, and requires straightforward post-processing of inferred segmentations to detect individual instances. The obtained results demonstrate moderately high accuracy and

the provision of individual masks for each instance, showcasing the model's effectiveness in segmenting complex objects like hands with distinct boundaries. It has limitations regarding the temporal stability and quality of the instance information, so its viability is also application specific. Finally, the model is light-weight, which is crucial for applications such as VR, and it operates in real-time at 44 FPS. This combination of high performance and real-time capability underscores the model's effectiveness in segmenting complex objects like hands with distinct boundaries.

**Author Contributions:** Conceptualization, E.d.l.F.; methodology, M.V. and A.C.; software, M.V.; validation, M.V. and A.C.; formal analysis, M.V., A.C. and E.d.l.F.; writing—original draft preparation, M.V. and A.C.; writing—review and editing, E.d.l.F. and J.C.F.; visualization, M.V.; supervision, E.d.l.F. and J.C.F.; project administration, E.d.l.F. and J.C.F.; funding acquisition, J.C.F. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work has been funded by the Spanish Ministry of Science, Innovation, and Universities, through the research project Shared-control strategies in Laparoscopic Liver Surgery (PID2022-138206OB-C33).

**Data Availability Statement:** The EgoHands dataset used in the study was obtained from the IU Computer Vision Lab and is openly available at <https://vision.soic.indiana.edu/projects/egohands/> [39], accessed on 15 October 2024.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Saposnik, G.; Levin, M.; for the Stroke Outcome Research Canada (SORCan) Working Group. Virtual Reality in Stroke Rehabilitation: A Meta-Analysis and Implications for Clinicians. *Stroke* **2011**, *42*, 1380–1386. [CrossRef] [PubMed]
2. Phan, H.L.; Le, T.H.; Lim, J.M.; Hwang, C.H.; Koo, K.i. Effectiveness of Augmented Reality in Stroke Rehabilitation: A Meta-Analysis. *Appl. Sci.* **2022**, *12*, 1848. [CrossRef]
3. Gu, W.; Bai, S.; Kong, L. A review on 2D instance segmentation based on deep neural networks. *Image Vis. Comput.* **2022**, *120*, 104401. [CrossRef]
4. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
5. Pinheiro, P.O.; Lin, T.Y.; Collobert, R.; Dollár, P. Learning to Refine Object Segments. In *Proceedings of the Computer Vision—ECCV, Amsterdam, The Netherlands, 11–14 October 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 75–91.
6. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8759–8768. [CrossRef]
7. Zhang, H.; Tian, Y.; Wang, K.; Zhang, W.; Wang, F.Y. Mask SSD: An Effective Single-Stage Approach to Object Instance Segmentation. *IEEE Trans. Image Process.* **2020**, *29*, 2078–2093. [CrossRef] [PubMed]
8. Huang, Z.; Huang, L.; Gong, Y.; Huang, C.; Wang, X. Mask Scoring R-CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
9. Chen, L.C.; Hermans, A.; Papandreou, G.; Schroff, F.; Wang, P.; Adam, H. MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4013–4022. [CrossRef]
10. Xu, W.; Wang, H.; Qi, F.; Lu, C. Explicit Shape Encoding for Real-Time Instance Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27–28 October 2019.
11. Arnab, A.; Torr, P.H.S. Bottom-up Instance Segmentation using Deep Higher-Order CRFs. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, 19–22 September 2016*; Wilson, R.C., Hancock, E.R., Smith, W.A.P., Eds.; BMVA Press: Glasgow, UK, 2016. [CrossRef]
12. Pham, T.; Vijay Kumar, B.G.; Do, T.T.; Carneiro, G.; Reid, I. Bayesian Semantic Instance Segmentation in Open Set World. In *Proceedings of the Computer Vision—ECCV 2018, Munich, Germany, 8–14 September 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Cham, Switzerland, 2018; pp. 3–18.
13. Arnab, A.; Torr, P.H.S. Pixelwise Instance Segmentation With a Dynamically Instantiated Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
14. Uhrig, J.; Rehder, E.; Fröhlich, B.; Franke, U.; Brox, T. Box2Pix: Single-Shot Instance Segmentation by Assigning Pixels to Object Boxes. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Suzhou, China, 26–30 June 2018; pp. 292–299. [CrossRef]



15. Uhrig, J.; Cordts, M.; Franke, U.; Brox, T. Pixel-Level Encoding and Depth Layering for Instance-Level Semantic Labeling. In *Pattern Recognition, Proceedings of the 38th German Conference, GCPR 2016, Hannover, Germany, 12–15 September 2016, Proceedings 38*; Rosenhahn, B., Andres, B., Eds.; Springer: Cham, Switzerland, 2016; pp. 14–25.
16. Kong, S.; Fowlkes, C. Recurrent Pixel Embedding for Instance Grouping. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018*; pp. 9018–9028. [[CrossRef](#)]
17. Fathi, A.; Wojna, Z.; Rathod, V.; Wang, P.; Song, H.O.; Guadarrama, S.; Murphy, K.P. Semantic Instance Segmentation via Deep Metric Learning. *arXiv* **2017**, arXiv:1703.10277. [[CrossRef](#)]
18. Chen, K.; Pang, J.; Wang, J.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Shi, J.; Ouyang, W.; et al. Hybrid Task Cascade for Instance Segmentation. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019*; pp. 4969–4978. [[CrossRef](#)]
19. Dai, J.; He, K.; Sun, J. Instance-Aware Semantic Segmentation via Multi-task Network Cascades. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016*; pp. 3150–3158. [[CrossRef](#)]
20. Cai, Z.; Vasconcelos, N. Cascade R-CNN: High Quality Object Detection and Instance Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 1483–1498. [[CrossRef](#)] [[PubMed](#)]
21. Romera-Paredes, B.; Torr, P.H.S. Recurrent Instance Segmentation. In *Computer Vision—ECCV 2016, Proceedings of the 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016, Proceedings, Part VI 14*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 312–329.
22. Ren, M.; Zemel, R.S. End-To-End Instance Segmentation With Recurrent Attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017*.
23. Salvador, A.; Bellver, M.; Baradad, M.; Campos, V.; Marqués, F.; Torres, J.; Nieto, X.G. Recurrent Neural Networks for Semantic Instance Segmentation. In *Proceedings of the ECCV 2018 Women in Computer Vision (WiCV) Workshop, Munich, Germany, 8–14 September 2018*. [[CrossRef](#)]
24. Fang, Y.; Yang, S.; Wang, X.; Li, Y.; Fang, C.; Shan, Y.; Feng, B.; Liu, W. Instances As Queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021*; pp. 6910–6919.
25. Dong, B.; Zeng, F.; Wang, T.; Zhang, X.; Wei, Y. SOLQ: Segmenting objects by learning queries. In *Proceedings of the 35th International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 6–14 December 2024*.
26. Wang, Y.; Xu, Z.; Wang, X.; Shen, C.; Cheng, B.; Shen, H.; Xia, H. End-to-End Video Instance Segmentation with Transformers. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021*; pp. 8737–8746.
27. Cheng, B.; Misra, I.; Schwing, A.G.; Kirillov, A.; Girdhar, R. Masked-Attention Mask Transformer for Universal Image Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022*; pp. 1290–1299.
28. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT: Real-Time Instance Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019*.
29. Wang, X.; Kong, T.; Shen, C.; Jiang, Y.; Li, L. SOLO: Segmenting Objects by Locations. In *Computer Vision—ECCV 2020, Proceedings of the 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part XVIII 16*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer: Cham, Switzerland, 2020; pp. 649–665.
30. Dai, J.; He, K.; Li, Y.; Ren, S.; Sun, J. Instance-Sensitive Fully Convolutional Networks. In *Computer Vision—ECCV 2016, Proceedings of the 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016, Proceedings, Part VI 14*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 534–549.
31. Li, Y.; Qi, H.; Dai, J.; Ji, X.; Wei, Y. Fully Convolutional Instance-Aware Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017*.
32. Chen, X.; Girshick, R.; He, K.; Dollar, P. TensorMask: A Foundation for Dense Object Segmentation. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019*; pp. 2061–2069. [[CrossRef](#)]
33. Ying, H.; Huang, Z.; Liu, S.; Shao, T.; Zhou, K. EmbedMask: Embedding Coupling for One-stage Instance Segmentation. *arXiv* **2019**, arXiv:1912.01954. [[CrossRef](#)]
34. Chen, H.; Sun, K.; Tian, Z.; Shen, C.; Huang, Y.; Yan, Y. BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020*.
35. Tian, Z.; Shen, C.; Chen, H. Conditional Convolutions for Instance Segmentation. In *Computer Vision—ECCV 2020, Proceedings of the 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part I 16*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer: Cham, Switzerland, 2020; pp. 282–298.
36. Xie, E.; Sun, P.; Song, X.; Wang, W.; Liu, X.; Liang, D.; Shen, C.; Luo, P. PolarMask: Single Shot Instance Segmentation With Polar Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020*.

37. Lee, Y.; Park, J. CenterMask: Real-Time Anchor-Free Instance Segmentation. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 13903–13912. [[CrossRef](#)]
38. Wang, X.; Zhang, R.; Kong, T.; Li, L.; Shen, C. SOLOv2: Dynamic and Fast Instance Segmentation. In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 17721–17732.
39. Bambach, S.; Lee, S.; Crandall, D.J.; Yu, C. Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1949–1957. [[CrossRef](#)]
40. Taylor, L.; Nitschke, G. Improving Deep Learning with Generic Data Augmentation. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bengaluru, India, 18–21 November 2018; pp. 1542–1547. [[CrossRef](#)]
41. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Proceedings of the 18th International Conference, Munich, Germany, 5–9 October 2015, Proceedings, Part III 18*; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer: Cham, Switzerland, 2015; pp. 234–241.
42. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [[CrossRef](#)]
43. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [[CrossRef](#)] [[PubMed](#)]
44. Milletari, F.; Navab, N.; Ahmadi, S.A. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. *arXiv* **2016**, arXiv:1606.04797. [[CrossRef](#)]
45. Abdelrahman, A.; Viriri, S. FPN-SE-ResNet Model for Accurate Diagnosis of Kidney Tumors Using CT Images. *Appl. Sci.* **2023**, *13*, 9802. [[CrossRef](#)]
46. Diserud, O.H.; Ødegaard, F. A multiple-site similarity measure. *Biol. Lett.* **2007**, *3*, 20–22. [[CrossRef](#)] [[PubMed](#)]
47. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034. [[CrossRef](#)]
48. Chen, L.; Wu, Y.; Stegmaier, J.; Merhof, D. SortedAP: Rethinking Evaluation Metrics for Instance Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Paris, France, 2–3 October 2023; pp. 3923–3929.
49. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
50. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 37, pp. 448–456. <https://arxiv.org/pdf/1502.03167>.
51. Cai, S.; Shu, Y.; Chen, G.; Ooi, B.C.; Wang, W.; Zhang, M. Effective and Efficient Dropout for Deep Convolutional Neural Networks. *arXiv* **2020**, arXiv:1904.03392. [[CrossRef](#)]
52. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015; pp. 1–15. [[CrossRef](#)]
53. Li, H.; Xu, Z.; Taylor, G.; Studer, C.; Goldstein, T. Visualizing the loss landscape of neural nets. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 6–14 December 2018; NIPS'18; pp. 6391–6401.
54. Khan, A.U.; Borji, A. Analysis of Hand Segmentation in the Wild. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4710–4719. [[CrossRef](#)]
55. Wang, W.; Yu, K.; Hugonot, J.; Fua, P.; Salzmann, M. Beyond One Glance: Gated Recurrent Architecture for Hand Segmentation. *arXiv* **2018**, arXiv:1811.10914. [[CrossRef](#)]
56. Li, M.; Sun, L.; Huo, Q. Flow-guided feature propagation with occlusion aware detail enhancement for hand segmentation in egocentric videos. *Comput. Vis. Image Underst.* **2019**, *187*, 102785. [[CrossRef](#)]
57. Cai, M.; Lu, F.; Sato, Y. Generalizing Hand Segmentation in Egocentric Videos With Uncertainty-Guided Model Adaptation. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 14380–14389. [[CrossRef](#)]
58. Tsai, T.H.; Huang, S.A. Refined U-net: A new semantic technique on hand segmentation. *Neurocomputing* **2022**, *495*, 1–10. [[CrossRef](#)]
59. Lin, F.; Price, B.; Martinez, T. Ego2Hands: A Dataset for Egocentric Two-hand Segmentation and Detection. *arXiv* **2021**. [[CrossRef](#)]
60. Roy, K.; Sahay, R.R. Hand Segmentation With Dense Dilated U-Net and Structurally Incoherent Nonnegative Matrix Factorization-Based Gesture Recognition. *IEEE Trans.-Hum.-Mach. Syst.* **2024**, *54*, 238–249. [[CrossRef](#)]
61. Li, C.; Kitani, K.M. Pixel-Level Hand Detection in Ego-centric Videos. In Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 3570–3577. [[CrossRef](#)]

62. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019. [[CrossRef](#)]
63. Fathi, A.; Ren, X.; Rehg, J.M. Learning to recognize objects in egocentric activities. In Proceedings of the CVPR 2011, Colorado Springs, CO, USA, 20–25 June 2011; pp. 3281–3288. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.