Semantic Image Segmentation of Human Faces

Daniyar Zhakyp
School of Engineering and Digital Sciences Electrical & Computer Engineering Department
Nazarbayev University
daniyar.zhakyp@nu.edu.kz

Danish Alenov
School of Engineering and Digital Sciences Electrical & Computer Engineering Department
Nazarbayev University
danish.alenov@nu.edu.kz

Azhar Adilkyzy
School of Engineering and Sciences Electrical & Com Engineering Department
Nazarbayev University
azhar.adilkyzy@nu.

Abstract

Based on the DeepLabV3+ model, this project presents a novel and practical architecture for facial recognition technology and addresses one of the most significant problems in computer vision: semantic segmentation. Although the variety of neural network architectures were systematically compared to the satisfactory approaches in the study, the atrous convolution model has shown efficient performance, which allows the utilization of several atrous convolutions at different dilation rates for a larger Field-of-View. One central issue in this methodology is an intrinsic problem in placing atrous convolution blocks with various rates. To solve this problem, we introduce the DeepLabV3+ model, which addresses the constraints caused by the loss of resolution inherited from unpooling operations. Experimental results on the DeepLabV3+ model indicate that our network performs either favorably or comparably in terms of the validation dataset for all metrics above 70%. It is apparent that there are a few insignificant disparities in certain facial features, such as the hair, but overall, the model's performance may be deemed satisfactory.

1. Introduction

A facial recognition system technology identifies a human face from a digital image or a video frame and matches it with a database of faces. Mostly used to authenticate users through ID verification services. Works by pinpointing and measuring facial features from a given image. Apple Face ID is a great example. Such systems must be able to understand the physiology of the human face, in order to increase reliability and security. One of the important problems such systems encounter is semantic segmentation, which is the task of labeling every pixel in an image with a predefined object category. In object detection we try to know what all objects that are present in an image, the location at which the objects are present with the help of bounding boxes. Image segmentation tries to find out accurately the exact boundary of the objects in the image. Array of images going to the input will give a grayscale image as an output (segmentation mask) with the same shape. Collection of images and pixel labeled images are used to train a semantic segmentation network. A pixel labeled image is an image where every pixel value represents the categorical label of that pixel. This project aims to label each pixel of an image with a corresponding class of what is being represented and the objects displayed in the image are grouped according to predefined classes or categories. The reduced feature resolution and segmenting objects at multiple scales are the main challenges of this project. Data augmentation is going to be implemented in this project, since it can be helpful for small datasets like this, and allows to raise the accuracy of the datasets' system for portrait segmentation. DeepLab V3 Model architecture was used for the project implementation, since it shows good performance and high efficiency.

2. Related Work

In this section, various models that make use of deep learning and CNNs have been proposed to tackle semantic image segmentation tasks. It has been shown that models based on Fully Convolutional Networks (FCNNs) have demonstrated significant improvement on several segmentation benchmarks [1, 2, 3], demonstrate effective feature generation and end-to-end training, and consequently have become the most popular choice for semantic segmentation. There are two studies [4, 5] that propose the benefits of global features or contextual interactions in accurately classifying pixels for semantic segmentation. In this project, there are several types of FCNN that are used to utilize context information for semantic segmentation [6, 7], including taking into account multi-scale inputs [8,9,10,11] or adopting a general framework that constructs and uses probabilistic graphical models. In order to address one of the issues of low-resolution predictions, a variety of techniques have been demonstrated to obtain high-resolution semantic segmentation. The general idea behind the image pyramid approach is that the model is typically implemented with shared weights for multi-scale inputs, with feature responses beginning with small-scale inputs encoded for the long-range context and large-scale inputs retaining the small-scale input object details. As one of the concerns in

scene parsing is how to take a wide context into account to make a local decision, in [12], the authors proposed a much simpler approach, where a feature extractor is applied densely to an image pyramid. Thus, the feature maps are upsampled to match the finest scale. Several studies [13, 14, 15] involve using families of segmentations or trees to generate candidate segments by aggregating elementary segments. Despite the cited approaches using engineered features, in [12], features are extracted densely from a multiscale pyramid of datasets using a convolutional network (ConvNet). Their work showed preliminary results that a convolutional network could be trained to perform scene parsing with high accuracy. Several good works have used the encoder-decoder model, such as deconvolutional networks by Zeiler *et al.* [16], in which the spatial dimension of feature maps is gradually reduced and thus larger image pictures can be captured in the deeper encoder output and the decoder, in which the spatial dimension and details of the object are moderately restored. Long *et al.* [7] and Noh *et al.* [17] also learned to use deconvolution for upsampling of low-resolution feature responses. More recent approaches have used SegNet [17], which aims to reuse the encoder's pooling indices and acquire information from additional convolutional layers to make dense the feature responses. Ronneberger *et al.* [18] applied U-Net to different segmentation tasks, which resulted in notably better results compared to the sliding-window convolutional network. Another common network architecture, RefineNet, which significantly differs from other methods, showed good performance in [19], by employing various ranges of residual connections with identity mapping that allows efficient end-to-end training of the system. Atrous convolution model that also enables explicit control of feature response computation density in FCCN was proposed by Chen *et al.* [14] and employs dilated or atrous convolutions to account for larger receptive fields without downscaling the image. Despite the fact that there are numerous existing works on how to effectively exploit multi-scale features, atrous convolution for image semantic segmentation was chosen as one of the most effective approaches by providing competitive performance in this project. Comprehensive empirical results clearly show the effectiveness of the method chosen for this project.

3. Methodology

3.1. Dataset Exploration

The dataset provided for the human face parsing task contained 4000 images, each segmented into eighteen different parts, or masks. Each image is a *128×128* RBG image, and each mask is a binary mask of the same size as the images. Figure 1 shows several typical images from the dataset, with all eighteen masks applicable to the images plotted simultaneously. Figure 2 shows the eighteen grayscale masks plotted separately. The masks include 'skin', 'nose', 'l_eye', 'r_eye', 'l_brow', 'r_brow', 'l_ear', 'r_ear', 'eye_g', 'u_lip', 'l_lip', 'hair', 'hat', 'ear_r', 'mouth', 'neck', 'cloth', and 'background' parts. The mask is empty when there is no such part in the image, like an earring (i.e., an 'ear_r' mask) in a picture of a man.

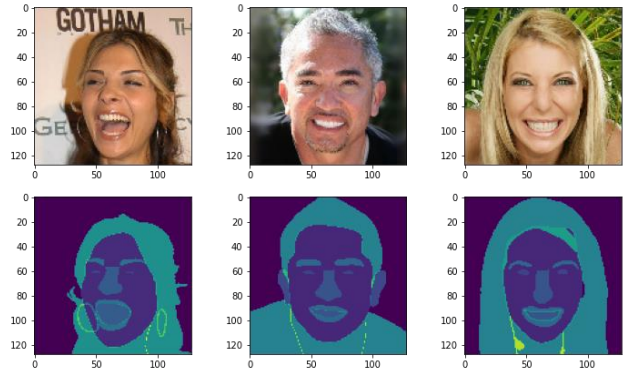

Figure 1. **The images from the training set** with 18 binary masks plotted together.

Figure 2. **The eighteen binary masks plotted** separately in grayscale**.**

3.2. Dataset Preprocessing

The dataset has been divided into training and validation sets in the proportion of 80/20, where 3200 images have been devoted to model training and 800 images for validation. Thus, the shapes of the training set for images and masks are *3200×128×128×3* and *3200×128×128×18* respectively, and the shapes for the validation set are *800×128×128×3* and *800×128×128×18* respectively.

The range of the images' values is from 0 to 255; the images' pixels have been scaled down to the range between 0 and 1 to avoid any bias for higher values during classification. The masks consist of "True" and "False" boolean values, which have been converted to the corresponding 0 and 1 numbers. Thus, images' and masks' values have been cast to the *float32* datatype during the preprocessing stage.

Data Augmentation To prevent the model from overfitting and increase its learning capability, the training set can be artificially increased through different data augmentation techniques. Figure 3 shows the original image with the "skin" mask and their horizontally flipped versions. The horizontal flipping has been applied to all 3200 images and their masks as the only data augmentation

method. The data augmentation has been done using the "Albumentations" library, which makes the process fast and convenient. It has been experimentally proven that horizontal flipping has a better effect on training and validation scores than other typical spatial-level transformations like random rotation or translation. Among some pixel-level transformations, we tried to execute channel shuffle and random brightness and contrast change augmentation procedures; however, Google Colab's RAM has given up on such "costly" operations, even when performed separately. Thus, by having all the images and their masks horizontally flipped, we increased the capacity of our training dataset up to 6400 units (i.e., 6400 images and masks). Later, during the training process, it was decided to train the model with and without augmented images to check the performance in both cases.

Figure 3. **The horizontal flipping augmentation procedure** applied to the images and their corresponding masks.

3.3. Model Architecture

Inspired by the robust performance of the DeepLabV3 model on the Cityscapes [20] and PASCAL VOC 2012 [21] datasets for semantic image segmentation, we have built the DeepLabV3+ model for the given task. Figure 3 shows the schematic of DeepLabV3+. It consists of the encoder, where the spatial dimensions of feature maps are gradually reduced and the deeper information is extracted, and the decoder, where these spatial dimensions are gradually recovered to retrieve the object boundaries. In the encoder, there is a model pre-trained on a large image dataset – in our case, it is ResNet-50. It is followed by a pyramid pooling layer consisting of both conventional and dilated (atrous) convolution operations with various rates, as well as the image pooling part. The encoder output is up-sampled and concatenated with the input in the decoder, which then goes through 3×3 convolutions and final up-sampling to produce the output mask.

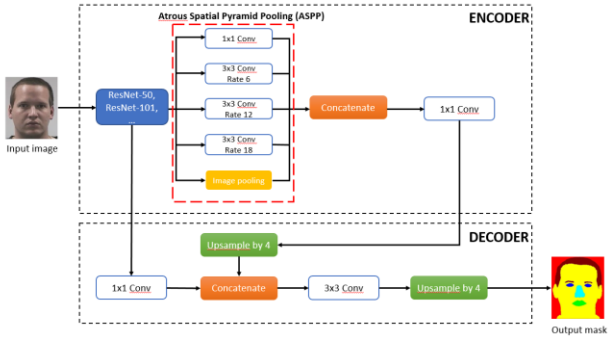

Figure 3. **TheDeepLabV3+ model** with the backbone as ResNet-50 (or other models), consisting of the encoder and decoder parts, with the Atrous Spatial Pyramid Pooling (ASPP) in the encoder.

Atrous Convolution Depending on the stride and kernel size, conventional convolution tends to reduce the spatial dimensions of a feature map. Normally, the desired shape of the output in binary or multiclass segmentation has the same size as the input image. While models like U-Net deploy transpose convolution operations to recover the input's spatial dimensions, the DeepLabV3+ model utilizes dilated or atrous convolutions, which satisfy the given requirement [22]. Figure 4 depicts an example of the mutable filter size of the atrous convolution due to rate change [22]. The conventional convolution operation is dilated convolution with rate equal to 1. By changing the rate, we can control how densely we want to extract our features. The zeroes padded the filter weights with a rate greater than 1.

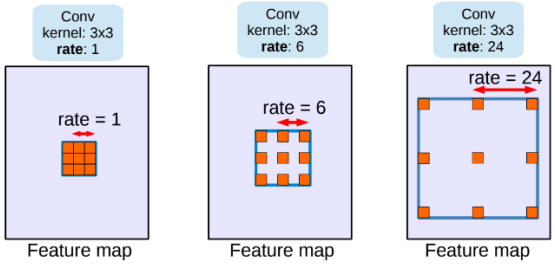

Figure 4. **Atrous convolution** with kernel size 3 x 3 and different rates. Zeros are padded between corresponding filter weights.

The output from the atrous convolution operation (consider a one-dimensional case) can be calculated as follows:

$$y[i] = \sum_{k} x[i + r \cdot k]w[k] \qquad (1)$$

where x is an input feature map, w are convolution filter's weights, r is the rate of dilation due to which $r - 1$ zeros are padded between two consecutive filter values.

Spatial Pyramid Pooling There is an inherent problem in placing atrous convolution blocks with different rates consecutively. As the dilation rate gradually increases, the valid (i.e., non-zero) values of the feature map become smaller due to the multiplications with the padded zeros of the atrous filters after each step [22]. Hence, the DeepLabV3+ model places those convolution blocks in parallel, which has received the name Atrous Spatial Pyramid Pooling (ASPP). To capture the global context information, image pooling has been introduced in ASPP, where the input feature map passes through the global average pooling and 1×1 convolution layers, and is then bi-linearly up-sampled to meet the dimensions of the data passed through other parallel blocks of ASPP [22].

Squeeze and Excitation It is a channel-wise operation, that squeezes the number of filters by some ratio after a convolution operation. This operation also excites certain features by allocating more weight to them in the depth dimension and suppresses their effect by allocating less weight. In our case, the squeeze ratio is equal to 8. This block was placed after the 1×1 convolution in the encoder and before the up-sampling operation in the decoder

Activation function Considering the structure of the dataset, the activation function of the last layer of the model is the 'sigmoid' function. The sigmoid function gives the probabilities between 0 and 1 for all eighteen binary masks individually, which can be considered as eighteen separate binary segmentation problems. From the model summary, we observe the output dimensions to be *Batch size×128×128×18*.

3.4. Model Training

The DeepLabV3+ model training has been performed in batches of size 4. The Adam optimizer with an initial learning rate of 10^{-3} has been chosen as the model optimizer. As for the metric, using which we monitored the performance of our model on training and validation sets, we used the mean intersection over union (mIoU) with a threshold of 0.5 retrieved from the "Segmentation Models" open-source library. It calculates the average IoU score across all the batches and all the classes. The loss function has been chosen to be the dice loss function, which works among the best for binary and multiclass image segmentation tasks. It also works qualitatively in tandem with the IoU score, where the dice loss equals the 1-dice coefficient, which in turn equals the IoU score with a double-counted intersection region. The model, with approximately 18 million parameters in total, has been compiled with the abovementioned arguments.

Learning Rate Scheduling The learning rate (LR) scheduling has been part of the model training process. Starting with an initial LR of 10^{-3}, the scheduling reduced the learning rate by 0.1 every five epochs, within which the validation loss remained unchanged. It is one of the overfitting prevention techniques that attempts to find the optimal learning rate, allowing the model to converge properly and learn to recognize small changes in various parameters. It is also recommended not to put the learning rate too low; therefore, the minimum possible LR due to scheduling has been 10^{-7}.

Early Stopping Early stopping is another model overfitting prevention techniques that has been used in our model training process. This method implies the early stoppage of model training after the pre-defined number of epochs where the model's validation loss (or other parameter) did not improve. In our training process, the number of epochs after which the early stopping happens has been set to 10.

As it has been said earlier, the same model has been trained on the data with and without augmentation for 80 epochs to compare the difference in performance (i.e., validation dice loss and mIoU). The augmentation process has not been performed on the validation set, hence it remained unchanged in both cases. If the validation loss has been refined after each epoch, the corresponding model has been saved to Google Drive.

4. Experimental Evaluation

As the size of the dataset was small (4000 images in total and 3200 images for training), it was decided to perform the data augmentation (i.e., horizontal image flipping) at first to enlarge the dataset and train the model for 80 epochs. Figure 5 shows the results for the training/validation loss and IoU for the model trained on 6400 images for 80 epochs. At those settings, the patience (i.e., the number of epochs with no improvement in validation loss after which the model stops training) of early stopping has been equal to 20. Overall, the model has shown robust performance on the validation set, with the validation loss almost equal to the training loss. However, there are some "unhealthy spikes" at the beginning of the training process, and the overall validation IoU score is not as high as it has been expected.

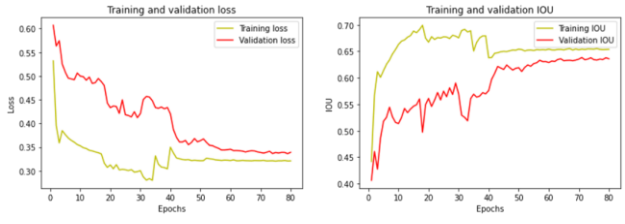

Figure 5. **Training and validation loss and IoU** values for the model trained on the data with augmentation (horizontal flipping).

Thus, we removed the augmentation, which might be paradoxically harmful for our dataset, to train the model on the original dataset, and set the patience for early stopping

to 10. Figure 6 shows the two graphs, but for the model trained on the original training dataset with 3200 images and masks inside. As we can see, the model has a better training and validation loss and IoU score even with the higher difference, and has less ripples and no random transients throughout all 80 epochs.

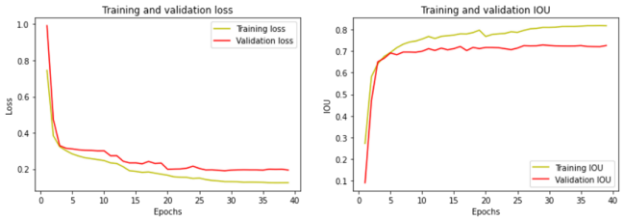

Figure 6. **Training and validation loss and IoU** values for the model trained on the original dataset (without augmentation).

Considering the model trained on the original data as the best model out of two, we assessed its performance based on three additional metrics besides the IoU score. Table 1 summarizes the results of the average IoU, precision, recall, and f1 scores across all the classes and also depicts these metrics' summaries for each individual class or mask. As it can be seen, the skin mask has the highest percent of recognition across all four metrics, and the hat mask has a nearly zero percent correct classification. The reason for that may lie in the density of pixels corresponding to the masks, where the skin has the highest density and the hat mask is almost always "empty" for numerous images.

	IoU	**Precision**	**Recall**	**F1 score**
Skin	**0.9304**	**0.9667**	**0.9612**	**0.9639**
Nose	0.8578	0.9128	0.9344	0.9234
Background	0.7455	0.9080	0.8063	0.8541
L_eye	0.7530	0.8636	0.8547	0.8591
R_eye	0.7482	0.8640	0.8481	0.8559
L_brow	0.6814	0.7955	0.8262	0.8105
R_brow	0.6739	0.7841	0.8273	0.8051
L_ear	0.6803	0.8315	0.7890	0.8097
R_ear	0.6646	0.8109	0.7865	0.7985
Eye_g	0.7456	0.8673	0.8415	0.8542
U_lip	0.6937	0.8129	0.8254	0.8191
L_lip	0.7585	0.8597	0.8656	0.8626
Hair	0.8665	0.9274	0.9295	0.9284
Hat	≈ 0	≈ 0	≈ 0	≈ 0
Ear_r	0.3522	0.6528	0.4333	0.5209
Mouth	0.0327	0.4979	0.0338	0.0633
Neck	0.7785	0.8849	0.8661	0.8754
Cloth	0.6780	0.8667	0.7569	0.8081
Mean (from model.predict)	0.7255	0.8654	0.8134	0.8386

Table 1. **The summary of the performance** of the model based on four different metrics for evaluating the prediction within all masks and within each individual mask.

In general, the model has shown robust performance on the validation dataset, with all four metrics above 70%. Figures 6 and 7 show the comparative plots of all masks of two random images from the validation dataset. Visually, it seems that the model accurately predicts all the masks: it plots nothing for the empty true masks and a similar shape for the present masks.

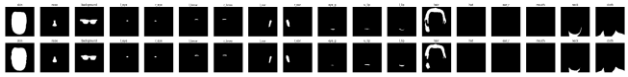

Figure 6. **The difference between the true (top) and predicted (bottom) binary masks** of one random image from the validation dataset.

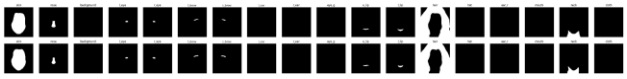

Figure 7. **The difference between the true (top) and predicted (bottom) binary masks** of another random image from the validation dataset.

Figure 8 shows three random images from the validation set, in which all the masks are plotted together. Again, it can be seen that there are some minor discrepancies in some parts of the face, like the hair, but the total model performance can be considered satisfactory.

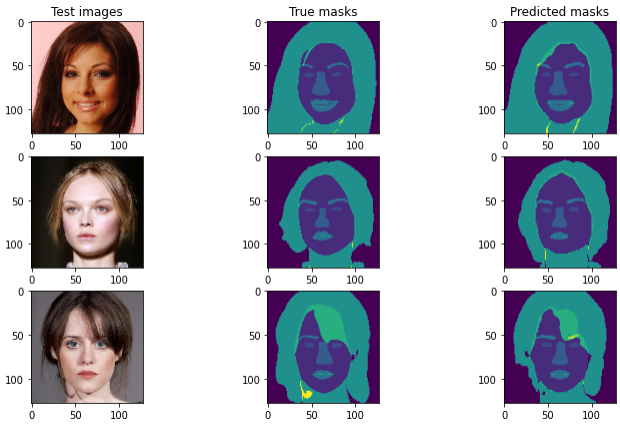

Figure 8. **The true and predicted masks** for three random images plotted together.

5. Conclusion

In general, the model has shown a high level of performance. As it can be seen from Figure 5, the validation dataset with the data augmentation have all four metrics above 70%. With the removed augmentation, the model has a better training and validation loss and IoU score. Validation loss is almost equal to the training loss. However, there are some "unhealthy spikes" at the beginning of the training process, and the overall validation IoU score is not as high as it has been expected. Figures 6 and 7 clearly illustrate that the model accurately predicts all the masks, with the comparative plots of all masks of two random images from the validation dataset. System plots nothing for the empty true masks and a similar shape for the present masks. In Figure 8 there are some minor discrepancies in some parts of the face, like the hair, but the total model performance is very good. As it can be seen from Table 1, the skin mask has the highest percent of recognition across all four metrics, and the hat mask has a nearly zero percent correct classification. In general we get good results with the model with high accuracy of object identification. All the goals are accomplished.

References

[1] Everingham, M., Eslami, S.M.A., Gool, L.V., Williams, C.K.I., Winn, J., Zisser- man, A.: The pascal visual object classes challenge a retrospective. IJCV (2014)

[2] Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: CVPR. (2014)

[3] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR. (2016).

[4] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 2009.

[5] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012.

[6] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv:1312.6229*, 2013.

[7] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[8] Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV. (2015)

[9] Lin, G., Shen, C., van den Hengel, A., Reid, I.: Efficient piecewise training of deep structured models for semantic segmentation. In: CVPR. (2016)

[10] Chen, L.C., Yang, Y., Wang, J., Xu, W., Yuille, A.L.: Attention to scale: Scale aware semantic image segmentation. In: CVPR. (2016)

[11] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. TPAMI (2017)

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks." in NIPS 2012.

[13] J. Fu, J. Liu, Y. Wang, and H. Lu. Stacked deconvolutional network for semantic segmentation. *arXiv:1708.04943*, 2017.

[14] L. C. Chen, G.Papandreou, I.Kokkinos, K.Murphy,andA.L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016.

[15] L. C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016.

[16] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvoutional networks for mid and high-level feature learning. In *ICCV*, 2011.

[17] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[18] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[19] G. Lin, A. Milan, C. Shen, and I. Reid. RefineNet: Multipath refinement networks with identity mappings for high-resolution semantic segmentation. *arXiv:1611.06612*, 2016.

[20] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V. and Garcia-Rodriguez, J., 2022. *A Review on Deep Learning Techniques Applied to Semantic Segmentation*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1704.06857>

[21] Chen, L.-C. *et al.* (2018) "Encoder-decoder with atrous separable convolution for Semantic Image segmentation," *Computer Vision – ECCV 2018*, pp. 833–851. Available at: https://doi.org/10.1007/978-3-030-01234-2_49.

[22] Chen, L., Papandreou, G., Schroff, F. and Adam, H., 2022. *Rethinking Atrous Convolution for Semantic Image Segmentation*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1706.05587>.