accurate segmentation and efficient network architecture with fewer parameters.

The architecture is characterized by its U-shaped design, consisting of a contracting path (encoder) followed by an expansive path (decoder) allowing for precise localization while capturing contextual information (Fig. 2). The EfficientNetB3 was used as the encoder [33], [34]. It consists of convolutional(Fig.2 blue blocks) and bottleneck layers (Fig.2 2, yellow blocks) [35]. In the middle column of Fig.2 2, dashed arrows denote skip connections between layers in the encoder and the decoder which help the decoder part recover spatial information lost during downsampling in the encoder part. In the research, we used Segmentation Models, a Python library with Neural Networks for Image Segmentation based on well-known Keras and TensorFlow frameworks [36].

The input images must have 3 channels (e. g. RGB) and have the same size. To effectively use the network, we resize initial images and the corresponding ground truth segmentations to 256x256. This allows us to get the results quickly without significant quality loss. However, other sizes could also be used, e. g. 224x224 which is a standard input size for EfficientNetB3, as well as bigger sizes. Also, the following parameters were used during the training: batch size is 5, optimizer function is Adam, the learning rate is 0.0001, the activation function on the last layer is softmax, and the number of epochs is 50. Besides, we consider two loss functions that take into consideration the spatial characteristics of segments. Dice loss is widely used as a metric showing how much two images are similar to each other [23], [37], [38]:

$$GDL = 1 - \frac{2\sum_{i=1}^{N} w_i \sum_{n=1}^{C}(r_{in} \cdot p_{in} + \epsilon)}{\sum_{i=1}^{N} w_i \sum_{n=1}^{C}(r_{in} + p_{in}) + \epsilon}, \quad (1)$$

where N is the number of pixels on each of two compared images, C is the number of classes, pln and rln are probabilities for the n-th pixel from both images to be in the l-th class, wl is a normalizing coefficient, $\in$ is a term to avoid division by zero. In the paper, value $\in$ = 107 is used. Focal loss is the metric widely used in image classification and segmentation tasks [37], [39]. It derives from the cross entropy concept and addresses the one-stage object detection scenario in which there is an extreme imbalance between foreground and background classes during training. The loss function is calculated according to the formula:

$$FL(p) = -(1-p)^{\gamma} log(p), \quad (2)$$

where p is the model's estimated probability for a pixel to belong to a certain class, and γ is the focusing parameter to down-weight easy examples and focus on training on hard ones. We used the default value which is γ = 2. Both functions are suitable for binary and ternary semantic segmentation. Besides, they require only a few parameters to define, so the models don't become too hard to tune. *C. CRF* In the research, we use PyDense-

CRF, a CPython-based Python wrapper for a fully connected CRF with a highly efficient approximate inference algorithm implemented in which the pairwise edge potentials are defined by a linear combination of Gaussian kernels [28], [40]. Concepts of appearance and smoothness are used in the network to calculate a posteriori probabilities for each pixel belonging to each class. Appearance is the property of a segmentation map to have nearby pixels of the same color likely belonging to the same class. In smooth models, large classes must absorb small isolated regions nearby. The formalization of both concepts can be expressed by the formula:

$$k(f_i, f_j) = \mu_1 exp\left[\frac{-||p_i - p_j||^2}{2\sigma_p^2}\right] - \left[\frac{-||l_i - l_j||^2}{2\sigma_l^2}\right] + \quad (3)$$

$$\mu_2 exp\left[\frac{-||p_i - p_j||^2}{2\sigma_p^2}\right], \quad (4)$$

where p_i and p_j - positions of two pixels, l_i and l_j - their colors, f_i and f_j - their features vectors, k is the similarity function to be maximized, u_1 and u_2 - linear combination weights, σ_p, σ_l are initially defined parameters. In the research, we used the following values: σ_p = 10, σ_l = 20, u_1 = 0, u_2 = 1.

D. The Evaluating Process

After evaluating the final segmentation maps, some metrics functions are calculated to obtain pixel-wise comparison ground truth maps with performed results. For binary segmentation, we calculated four values for each pair:

$$TP = |\{(i,j) : p_{ij} = 1 \wedge b_{ij} = 1\}|, \quad (5)$$

$$FP = |\{(i,j) : p_{ij} = 1 \wedge b_{ij} = 0\}|, \quad (6)$$

$$TN = |\{(i,j) : p_{ij} = 0 \wedge b_{ij} = 0\}|, \quad (7)$$

$$FN = |\{(i,j) : p_{ij} = 0 \wedge b_{ij} = 1\}|, \quad (8)$$

where (i, j) stands for the position of two pixels to be compared, p_{ij} is the value for the ground truth pixel (1 stands for crowded region, and 0 means the pixel belongs to non-crowded area), and p_{ij} is the value for the pixel on predicted map. After that, accuracy, crowd predictive value, and non-crowded predictive value are calculated:

$$acc = \frac{TP + TN}{TP + FP + TN + FN}, \quad (9)$$

$$cpi = \frac{TP}{TP + FP}, \quad (10)$$

$$npi = \frac{TN}{TN + FN}. \quad (11)$$

Besides, for each image, we calculated the number of annotation labels that fell into each class (according to ground truth and predicted segmentation maps):

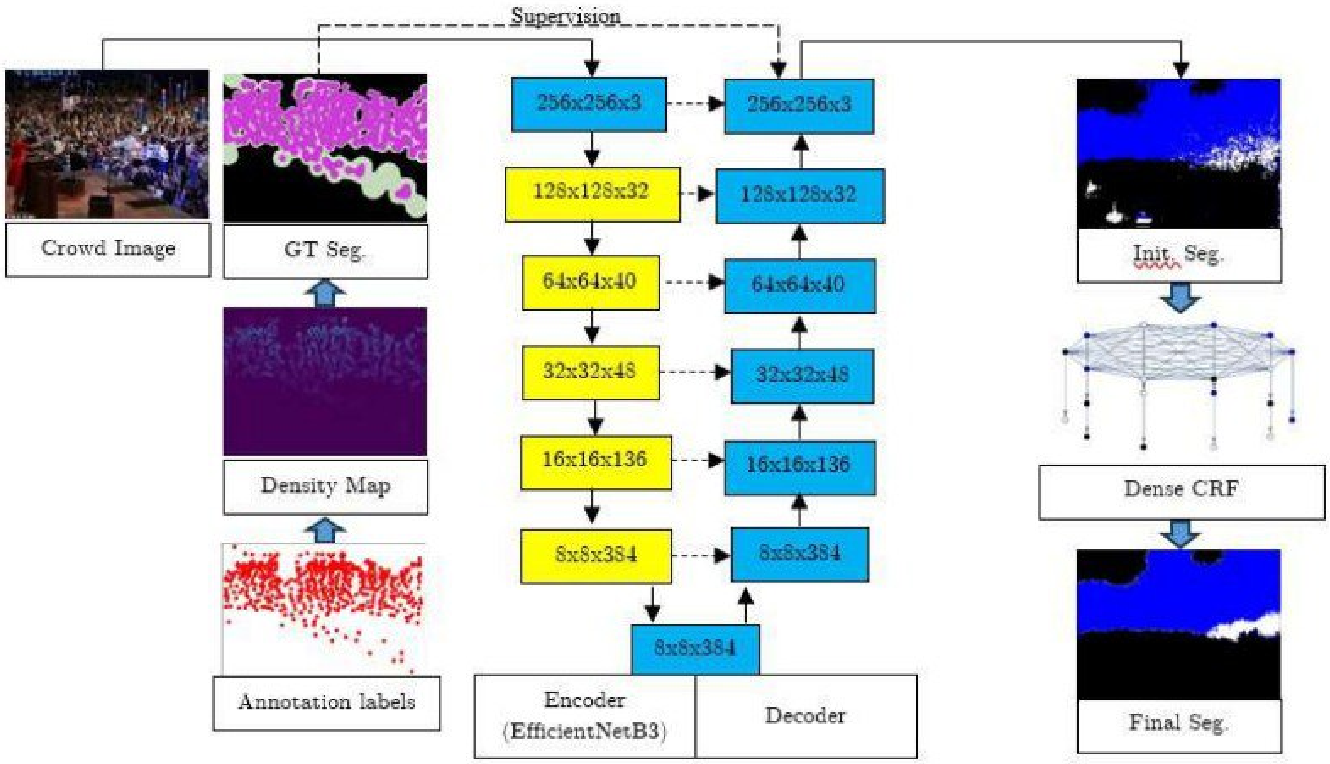

Figure 2: The crowd segmentation prediction framework consisting of image preprocessing (left column), the UNet network (middle column), and dense CRF to refine the predicted segmentation (right column)

$$N_0 = |\{i : p_{hi} = 0\}|$$
$$\mathrm{N}_0 = |\{i : p_{hi} = 0\}|,$$
$$\hat{\mathrm{N}}_1 = |\{i : p_{hi} = 1\}|,$$
$$\hat{\mathrm{N}}_1 = |\{i : p_{hi} = 1\}|,$$

where N stands for the number of annotation labels felt in the class labeled by the number i, and hi is the location of the i-th annotation label. Then, to estimate the rate of the crowd detection, we calculated the N1/$\hat{\mathrm{N}}$. For the ternary segmentation, the following characteristics are calculated:

DD=—(i,j):p$_{ij} = 2\hat{p}_{ij} = 2|$,
DS=—(i,j):p$_{ij} = 2\hat{p}_{ij} = 1|$,
DN=—(i,j):p$_{ij} = 2\hat{p}_{ij} = 0|$,
SD=—(i,j):p$_{ij} = 1\hat{p}_{ij} = 2|$,
SS=—(i,j):p$_{ij} = 1\hat{p}_{ij} = 1|$,
SN=—(i,j):p$_{ij} = 1\hat{p}_{ij} = 0|$,
ND=—(i,j):p$_{ij} = 0\hat{p}_{ij} = 2|$,
NS=—(i,j):p$_{ij} = 0\hat{p}_{ij} = 2|$,
NN=—(i,j):p$_{ij} = 0\hat{p}_{ij} = 0|$,

where 0 stands for non-crowded pixels, 1 is sparse crowd, and 2 is dense crowd. Based on these values, we calculate metrics that we call accuracy, dense crowd predictive value, crowd predictive value, and non-crowded predictive value:

$$acc = \frac{DD+SS+NN+}{DD+DS+DN+SD+SS+SN+ND+NS+NN},$$
$$dvp = \frac{DD}{DD+DS+DN},$$
$$cpv = \frac{DD+DS+SD+SS}{DD+DS+DN+SD+SS+SN},$$
$$npv = \frac{NN}{ND+NS+NN},$$

In the same manner, as for binary segmentation, we calculate the number of annotation labels corresponding to all three predicted classes:

$$\hat{N}_0 = |\{i : \hat{p}_{hi} = 0\}|,$$
$$\hat{\mathrm{N}}_0 = |\{i : \hat{p}_{hi} = 0\}|,$$
$$\mathrm{N}_1 = |\{i : \hat{p}_{hi} = 1\}|,$$
$$\hat{\mathrm{N}}_1 = |\{i : \hat{p}_{hi} = 1\}|,$$
$$\mathrm{N}_2 = |\{i : \hat{p}_{hi} = 2\}|.$$
$$\hat{\mathrm{N}}_2 = |\{i : \hat{p}_{hi} = 2\}|.$$

Based on such characteristics, we can calculate the rate of dense crowd detection equalling *N1/$\hat{N}$*.

E. Crowd Dense Semantics

After obtaining and refining the segmentation maps (Fig. 3a), we clusterize the people in crowd images based on their positions provided as annotations for the considered dataset. For this, we divide all annotation points according to the corresponding connected areas of resultant segmentations (Fig. 3b) and clusterize each group separately (Fig. 3c). We don't clusterize points falling into non-crowded areas. Any clustering method for 2D points can be used. We decided to use DBSCAN as it demonstrated its benefit in various researches in logistics, spatial analysis, and behavior patterns detection [41], [42].

After the clustering, all points are divided into multiple clusters represented as convex hulls of the points inside them (Fig. 3c). Each cluster is characterized by its location, density, people count, as well as spatial connections

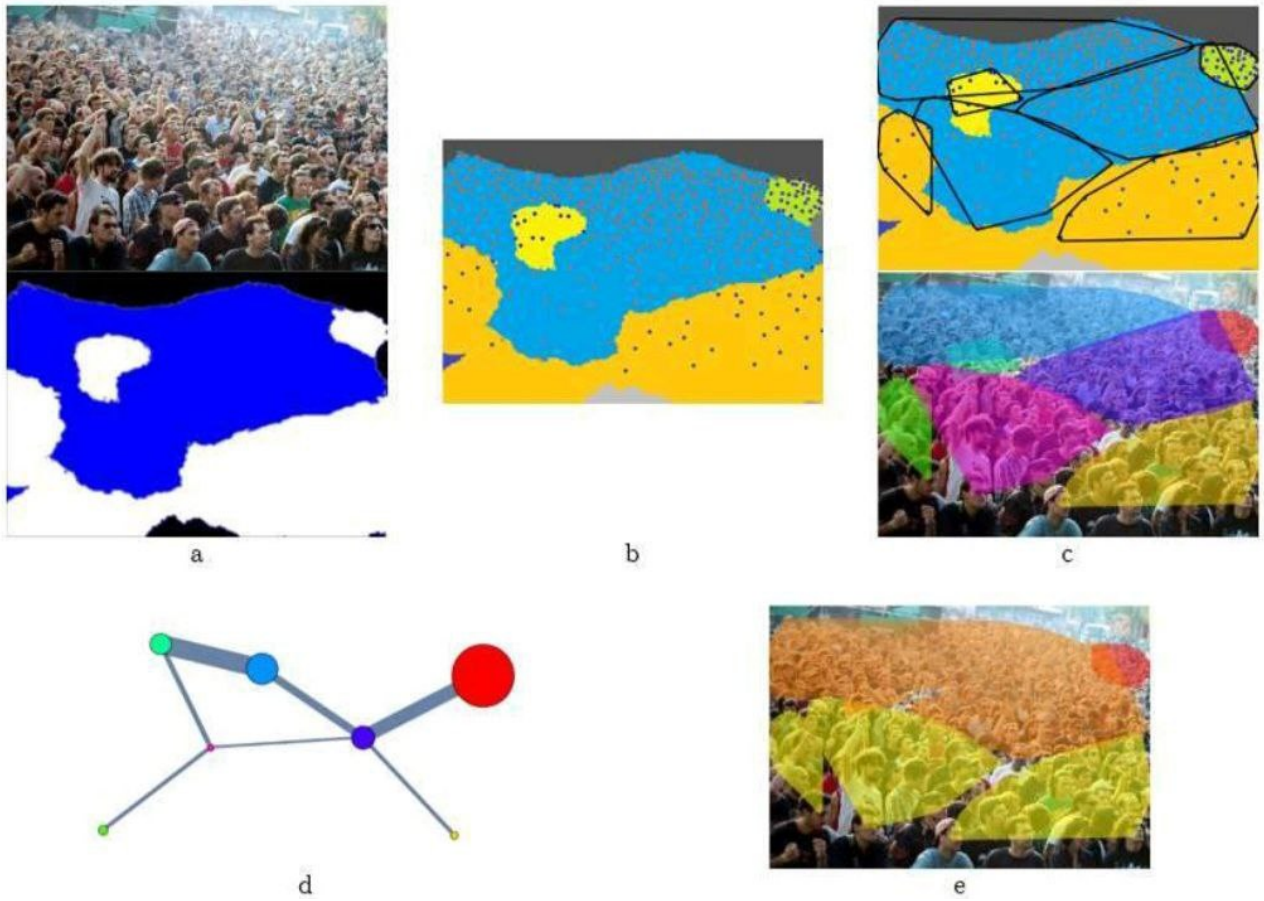

Figure 2: Figure 3. Semantics in a crowd: the initial image and the segmentation map for it (a), connected areas in the segmentation map (b), clusterization of annotation labels within connected areas (c), the clusters connectivity graph (d), crowd semantic clusters (e)

with other clusters. All clusters and such connections are presented as the graph where each vertex is assigned to a cluster, and two vertices are adjacent if only two clusters overlap or share a border (Fig. 3d). The size of a vertex is proportional to the density of the crowd in the corresponding cluster, and the thickness of each edge is proportional to the area shared between two adjacent clusters. Based on the separate clusters' properties and the graph's connectivity, we can interpret semantics for the depicted crowd in the image. In Fig. 3e, the red cluster is a dense crowd (more than 1 individual per 1000 pixels), the orange one is a regular crowd (0.5- 1.5 individuals), and the yellow clusters present a sparse crowd (less than 0.7 individuals). Different clusters can share their borders, overlap, or even be a part of another cluster. Considering such facts, we can evaluate the crowd semantics.

IV *R*esults and Discussion

A. Segmentation Evaluation

After training neural networks and obtaining predicted segmentation maps for images in testing samples (Fig. 1, 3a), we received the results presented in Table I. Statistics on calculated accuracies and predictive values through the testing samples (from ShanghaiTech dataset's part B for binary segmentation and part A for ternary segmentation) are presented there.

As we can see, the CRF refinement significantly improves the ternary segmentations in terms of overall quality (acc.) and dense crowd detection (DPV). In other cases, it didn't demonstrate better results. Comparing dice and focal loss functions' results, we can conclude that the neural network with focal loss can predict crowd regions slightly better (according to the CPV). Nevertheless, dice loss allows us to predict dense crowds slightly better (DPV). Also, binary segmentation gave better results than ternary one in terms of overall performance. However, it takes place due to its ability to detect non- crowded areas (NPV) whereas the ternary segmentation model is more successful in detecting crowded areas (CPV).

Statistics on crowd detection rates are presented in Table II. Here we can see the poor performance of the model using the focal loss function in detecting sparse crowds (low values of N1/Nˆ1), but for dense crowds detection, both dice and focal functions results are approximately equal (dice function demonstrates slightly better results though). On average, the model is prone to overlook some parts of sparse and dense crowds (from 3% to 22% according to average and median results).