# CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes[a]

M.Shedrolosiev[1]

[1]Taras Shevchenko National University of Kyiv, High Energy Physics Department

21 October 2019

- Crowd counting
  - Detection-based approaches
  - Regression-based approaches
  - *Density estimation-based approaches*
  - *CNN-based approach*
- CSRNet architecture
  - VGG-16 (front-end)
  - Dilated convolution
  - Ground truth generation
  - Training details and Evaluation metrics
- Code
- Results
- Project discussion … (current status)

This is overview for a network for Congested Scene Recognition called CSRNet to provide a data-driven and deep learning method that can understand highly congested scenes and perform accurate count estimation as well as present highquality density maps. The proposed CSRNet is composed of two major components: a convolutional neural network (CNN) as the front-end for 2D feature extraction and a dilated CNN for the back-end, which uses dilated kernels to deliver larger reception fields and to replace pooling operations. CSRNet is an easy-trained model because of its pure convolutional structure

a) https://arxiv.org/pdf/1802.10062.pdf

# Crowd counting (motivation)

- Counting the number of people attending a sporting event
- Estimating how many people attended an inauguration or a march (political rallies, perhaps)
- Monitoring of high-traffic areas
- Helping with staffing allocation and resource allotment

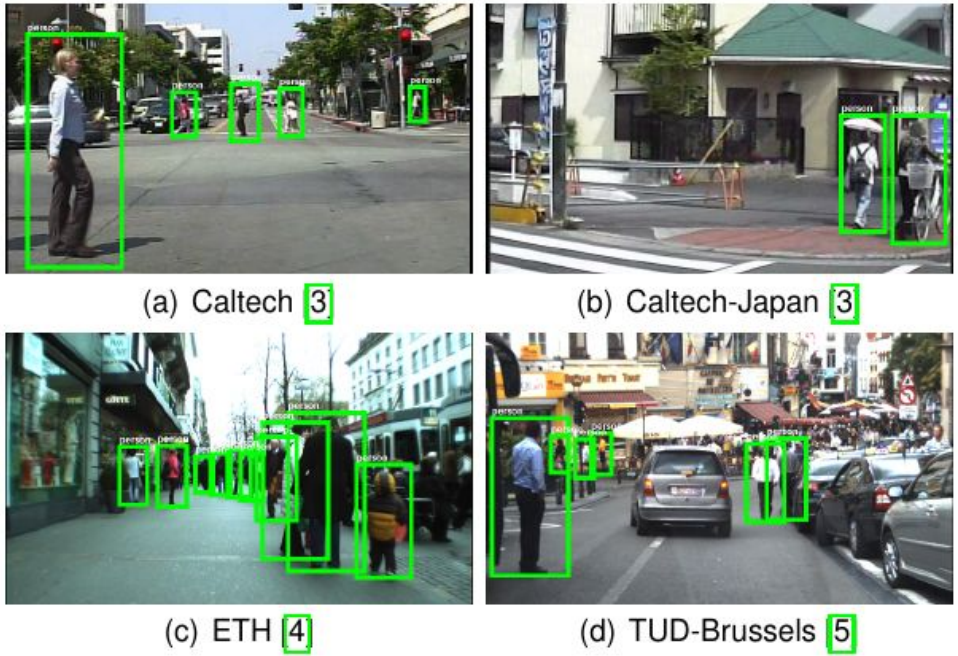
… deliver promising solutions for crowd flows monitoring, assembly controlling, and other security services.

**Methods:**

- **Detection-based approaches**
- **Regression-based approaches**
- **Density estimation-based approaches**
- **CNN-based approach**

# Detection-based approaches

- Moving-window-like detector
- Require well-trained classifiers to extract low-level features (time consuming)
- Perform poorly on highly congested scenes since most of the targeted objects are obscured.




http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.6884&rep=rep1&type=pdf

# Regression-based approaches

- More features, such as foreground and texture features, have been used for generating low-level information
- Idrees et al. [1] propose a model to extract features by employing Fourier analysis
- Perform better on highly congested scenes
- When executing the regression-based solution, one critical feature, called saliency, is overlooked which causes inaccurate results in local regions

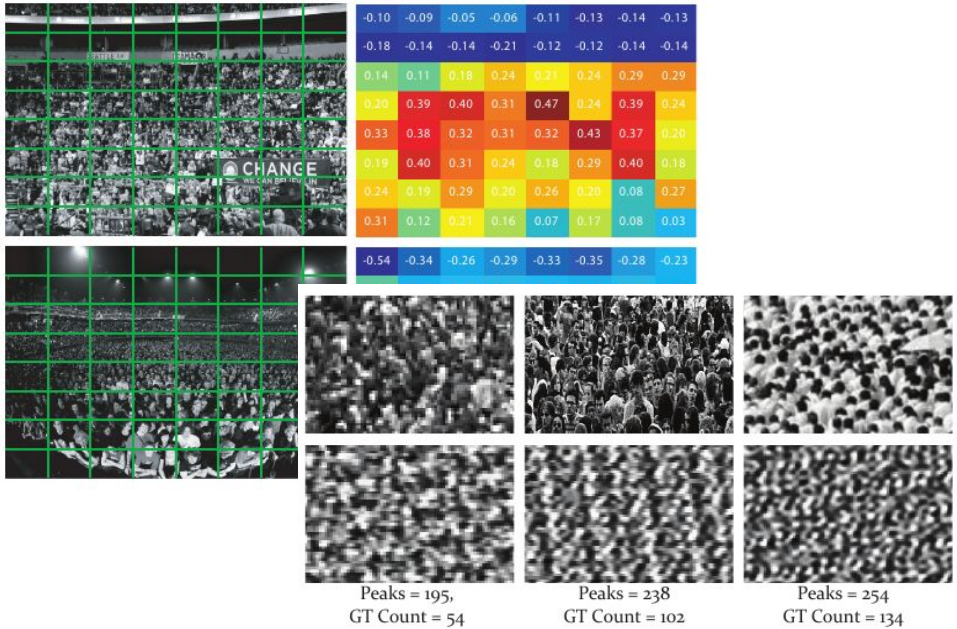
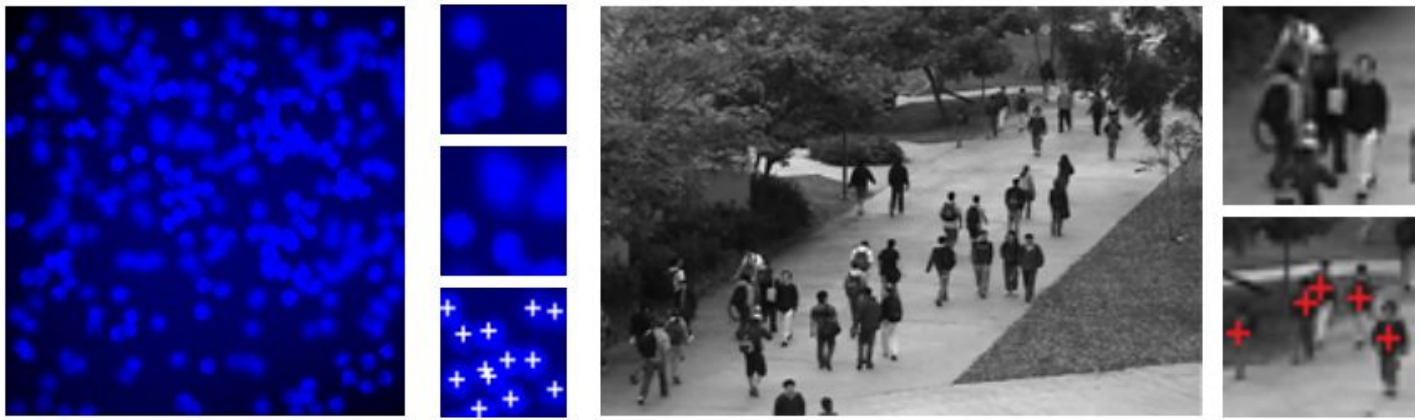



Figure 3: Counting through Fourier Analysis: The first row shows three original patches, while the second row shows corresponding reconstructed patches. The positive correlation is evident from the number of local maximas in the reconstructed patch, and the ground truth counts shown at the bottom.

1) http://www.cs.ucf.edu/~haroon/datafiles/Idrees_Counting_CVPR_2013.pdf

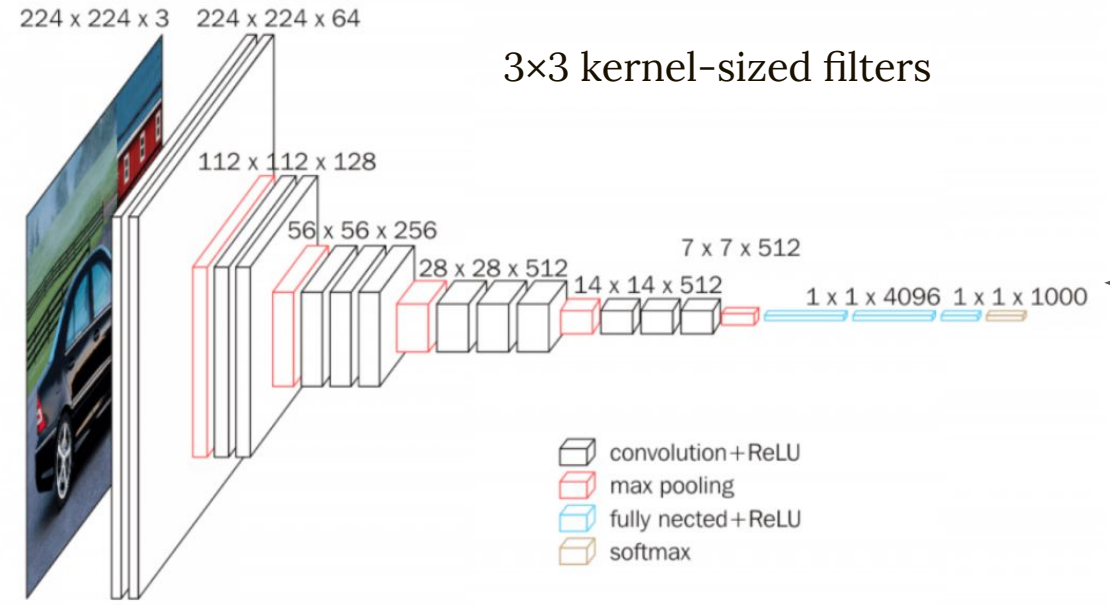
# Density estimation-based approaches

- Learning a linear mapping between features in the local region and its object density maps.
- It integrates the information of saliency during the learning process.



https://www.robots.ox.ac.uk/~vgg/publications/2010/Lempitsky10b/lempitsky10b.pdf

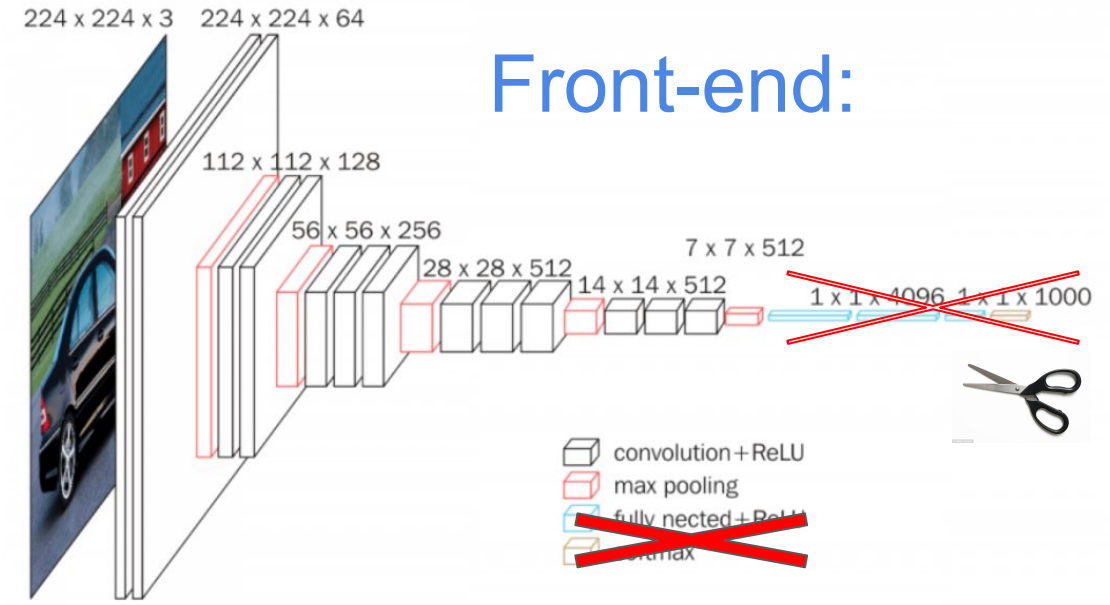
# CSRNet architecture (VGG-16)




Due to its depth and number of fully-connected nodes, VGG16 is over 533MB.

- The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes
- VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's

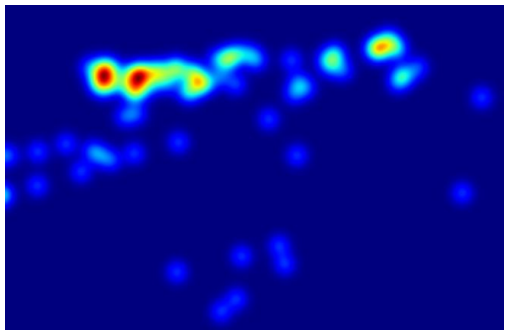
# CSRNet architecture (VGG-16)... transfer learning




## Front-end:

## Back-end:

| back-end (four different configurations) | | | |
|---|---|---|---|
| conv3-512-1 | conv3-512-2 | conv3-512-2 | conv3-512-4 |
| conv3-512-1 | conv3-512-2 | conv3-512-2 | conv3-512-4 |
| conv3-512-1 | conv3-512-2 | conv3-512-2 | conv3-512-4 |
| conv3-256-1 | conv3-256-2 | conv3-256-4 | conv3-256-4 |
| conv3-128-1 | conv3-128-2 | conv3-128-4 | conv3-128-4 |
| conv3-64-1 | conv3-64-2 | conv3-64-4 | conv3-64-4 |
| conv1-1-1 | | | |



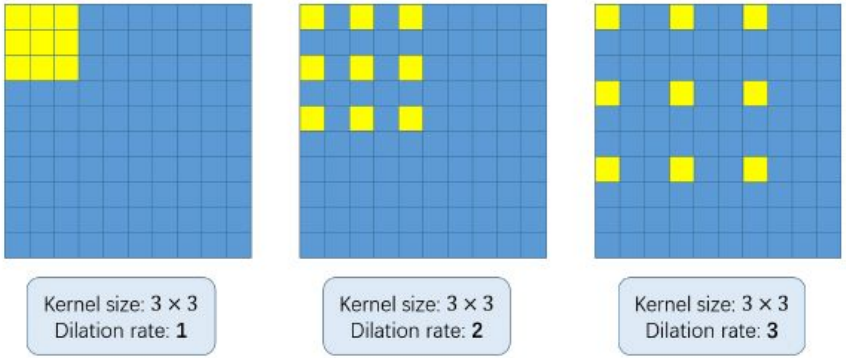
**{ density map }**

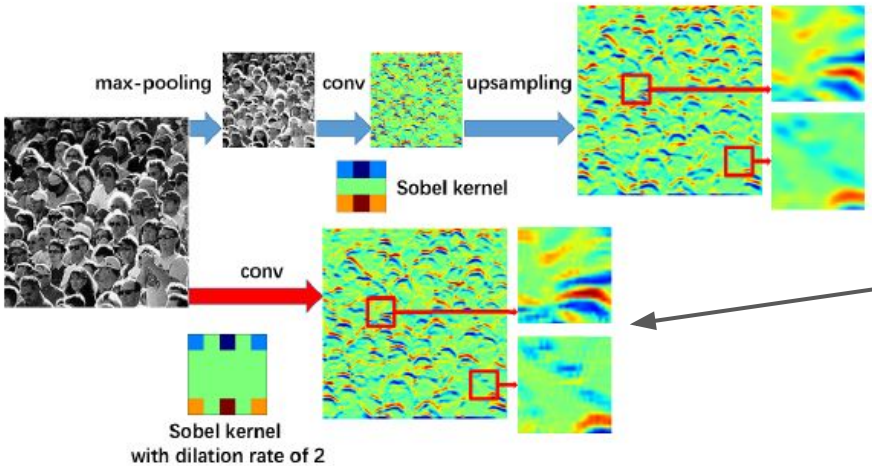
- Extract low level features
- Reduce training time
- extracting deeper information of saliency as well as maintaining the output resolution.

# Dilated convolution (Back-end layers)

$$y(m,n) = \sum_{i=1}^{M}\sum_{j=1}^{N} x(m + r \times i, n + r \times j)w(i,j)$$

dilation rate




Usual pooling dramatically reduce the spatial resolution meaning the spatial information of feature map is lost

# Ground truth generation

the average distance of k nearest neighbors

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_i) \times G_{\sigma_i}(\boldsymbol{x}),\ with\ \sigma_i = \beta \overline{d_i}$$

$\beta$ = 0.3
k = 3

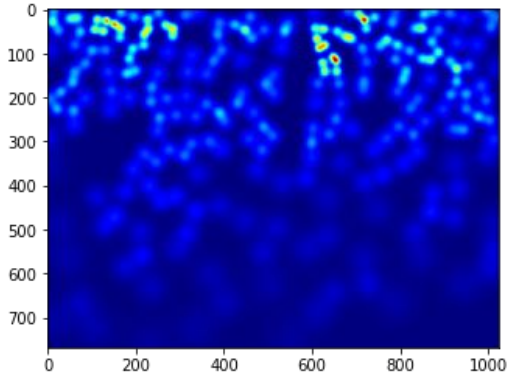
. Gaussian kernel with parameter $\sigma_i$

[ $(x_1, y_1)$, $(x_2, y_2)$, $(x_3, y_3)$ …
]

**generate density**


input

CSRNet




output

# Training details and Evaluation metrics

loss function:

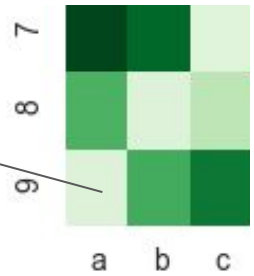
$$L(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \| \ Z(X_i; \Theta) - Z_i^{GT} \ \|_2^2$$

Evaluation matrics: Mean Absolute Error (MAE) and mean squared error (MSE)

$$MAE = \frac{1}{N} \sum_{i=1}^{N} | \ C_i - C_i^{GT} \ |$$

$$MSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} | \ C_i - C_i^{GT} \ |^2}$$

$$C_i = \sum_{l=1}^{L} \sum_{w=1}^{W} z_{l,w}$$



# Code

- https://github.com/leeyeehoo/CSRNet-pytorch/tree/master
- https://github.com/Neerajj9/CSRNet-keras
- https://github.com/tlokeshkumar/CSRNet-tf


- Python 2.7 -> 3.6
- Remove GPU support
- Try to reduce memory usage (not successfully yet)

# Code (data preprocessing)

```python
#this is borrowed from https://github.com/davideverona/deep-crowd-counting_crowdnet
def gaussian_filter_density(gt):
    print(gt.shape)
    density = np.zeros(gt.shape, dtype=np.float32)
    gt_count = np.count_nonzero(gt)
    if gt_count == 0:
        return density

#     pts = np.array(zip(np.nonzero(gt)[1], np.nonzero(gt)[0]))
    pts = list(zip(np.nonzero(gt)[1], np.nonzero(gt)[0]))
    pts_copy = pts.copy()
#     print(pts_copy)
    leafsize = 2048
    # build kdtree
    tree = scipy.spatial.KDTree(pts_copy, leafsize=leafsize)
    # query kdtree
    distances, locations = tree.query(pts, k=4)

    print('generate density...')
    for i, pt in enumerate(pts):
        pt2d = np.zeros(gt.shape, dtype=np.float32)
        pt2d[pt[1],pt[0]] = 1.
        if gt_count > 1:
            sigma = (distances[i][1]+distances[i][2]+distances[i][3])*0.1
        else:
            sigma = np.average(np.array(gt.shape))/2./2. #case: 1 point
        density += scipy.ndimage.filters.gaussian_filter(pt2d, sigma, mode='constant')
    print('done.')
    return density
```

```python
for img_path in img_paths:
    print(img_path)
    mat = io.loadmat(img_path.replace('.jpg','.mat').replace('images','ground-truth').replace('IMG_','GT_IMG_'))
    img= plt.imread(img_path)
    k = np.zeros((img.shape[0],img.shape[1]))
    gt = mat["image_info"][0,0][0,0][0] #possition of the person head on image
    for i in range(0,len(gt)):
        if int(gt[i][1])<img.shape[0] and int(gt[i][0])<img.shape[1]:
            k[int(gt[i][1]),int(gt[i][0])]=1
    k = gaussian_filter_density(k)
    with h5py.File(img_path.replace('.jpg','.h5').replace('images','ground-truth'), 'w') as hf:
            hf['density'] = k
```

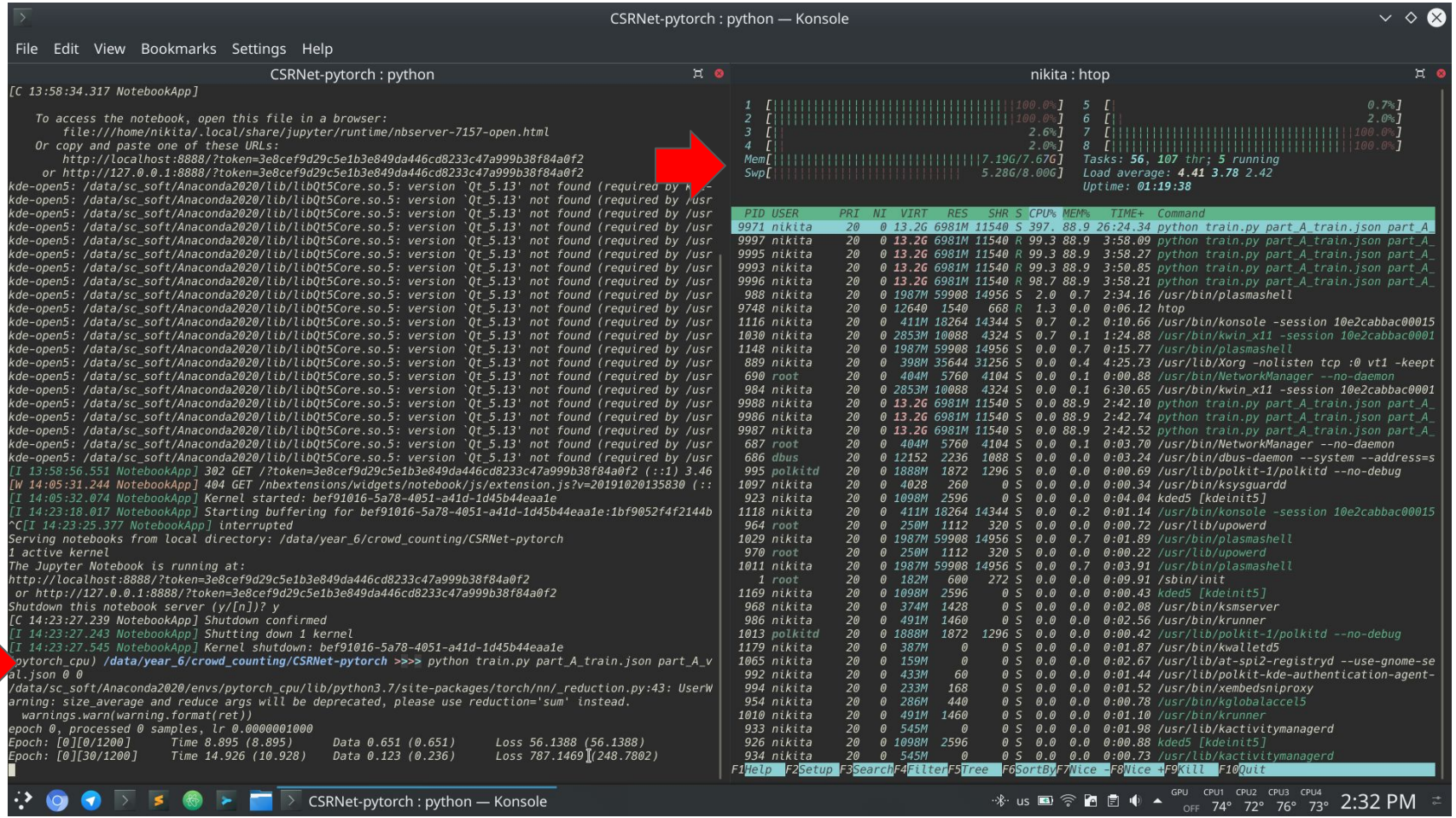
# Code (model)

```
class CSRNet(nn.Module):
    def __init__(self, load_weights=False):
        super(CSRNet, self).__init__()
        self.seen = 0
        self.frontend_feat = [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 'M', 512, 512, 512]
        self.backend_feat  = [512, 512, 512,256,128,64]
        self.frontend = make_layers(self.frontend_feat)
        self.backend = make_layers(self.backend_feat,in_channels = 512,dilation = True)
        self.output_layer = nn.Conv2d(64, 1, kernel_size=1)
        if not load_weights:
            mod = models.vgg16(pretrained = True)
            self._initialize_weights()
            for i in range(len(self.frontend.state_dict().items())):
                list(self.frontend.state_dict().items())[i][1].data[:] = list(mod.state_dict().items())[i][1].data[:]
#                 self.frontend.state_dict().items()[i][1].data[:] = mod.state_dict().items()[i][1].data[:]
                # list(self.frontend.state_dict().items())[i][1].data[:] = list(mod.state_dict().items())[i][1].data[:]
    def forward(self,x):
        x = self.frontend(x)
        x = self.backend(x)
        x = self.output_layer(x)
        return x
    def _initialize_weights(self):
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.normal_(m.weight, std=0.01)
                if m.bias is not None:
                    nn.init.constant_(m.bias, 0)
            elif isinstance(m, nn.BatchNorm2d):
                nn.init.constant_(m.weight, 1)
                nn.init.constant_(m.bias, 0)
```

# Code (image crop)

```
def load_data(img_path,train = True):
    gt_path = img_path.replace('.jpg','.h5').replace('images','ground-truth')
    img = Image.open(img_path).convert('RGB')
    gt_file = h5py.File(gt_path)
    target = np.asarray(gt_file['density'])
    if False:
        crop_size = (img.size[0]/2,img.size[1]/2)
        if random.randint(0,9)<= -1:


            dx = int(random.randint(0,1)*img.size[0]*1./2)
            dy = int(random.randint(0,1)*img.size[1]*1./2)
        else:
            dx = int(random.random()*img.size[0]*1./2)
            dy = int(random.random()*img.size[1]*1./2)



        img = img.crop((dx,dy,crop_size[0]+dx,crop_size[1]+dy))
        target = target[dy:crop_size[1]+dy,dx:crop_size[0]+dx]




        if random.random()>0.8:
            target = np.fliplr(target)
            img = img.transpose(Image.FLIP_LEFT_RIGHT)



    target = cv2.resize(target,(int(target.shape[1]/8),int(target.shape[0]/8)),interpolation = cv2.INTER_CUBIC)*64


    return img,target
```

# Results (memory problem)

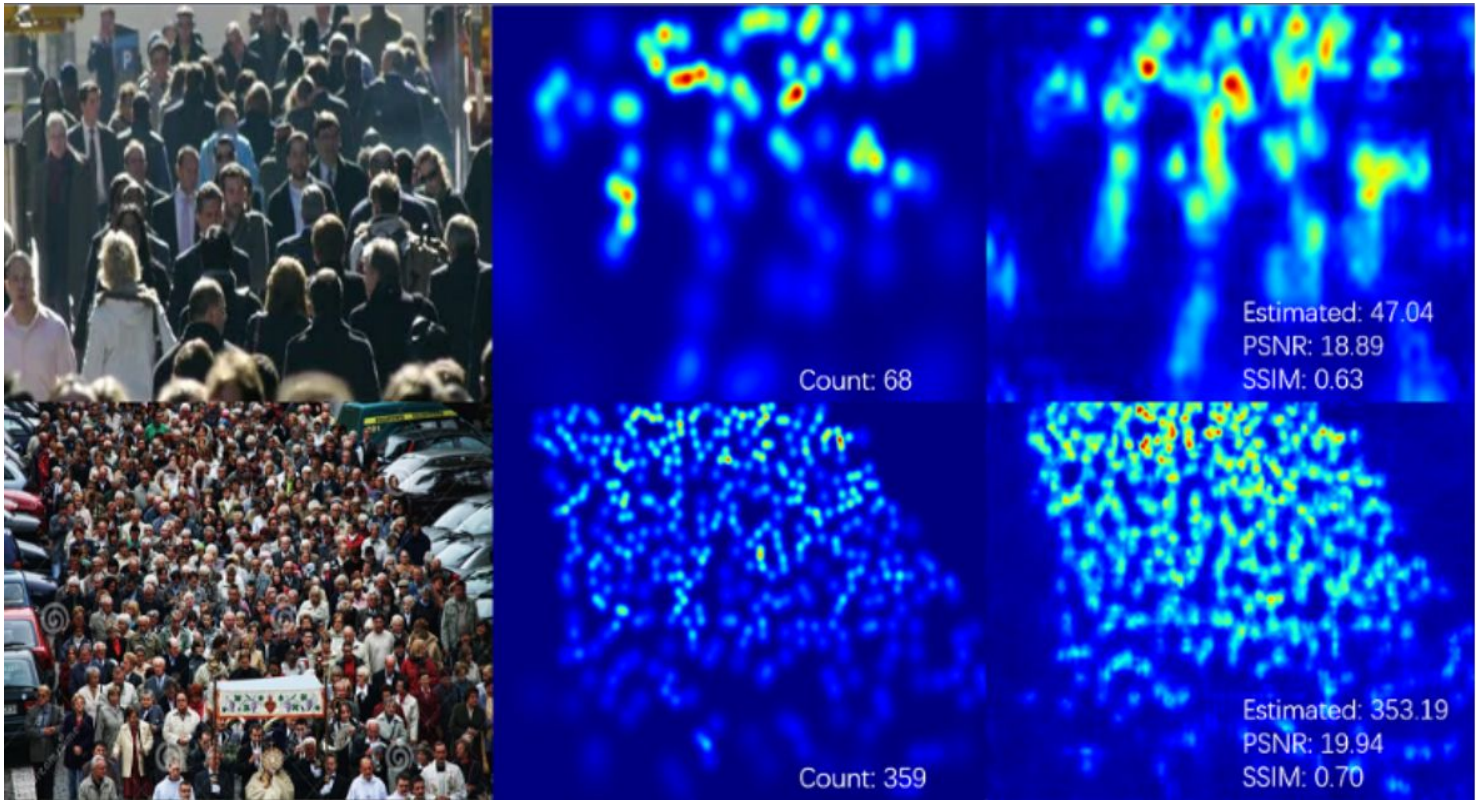


# Results (from arxiv)

| | Part_A | | Part_B | |
|---|---|---|---|---|
| Method | MAE | MSE | MAE | MSE |
| Zhang *et al.* [3] | 181.8 | 277.7 | 32.0 | 49.8 |
| Marsden *et al.* [38] | 126.5 | 173.5 | 23.8 | 33.1 |
| MCNN [18] | 110.2 | 173.2 | 26.4 | 41.3 |
| Cascaded-MTL [39] | 101.3 | 152.4 | 20.0 | 31.1 |
| Switching-CNN [4] | 90.4 | 135.0 | 21.6 | 33.4 |
| CP-CNN [5] | 73.6 | **106.4** | 20.1 | 30.1 |
| CSRNet (ours) | **68.2** | 115.0 | **10.6** | **16.0** |

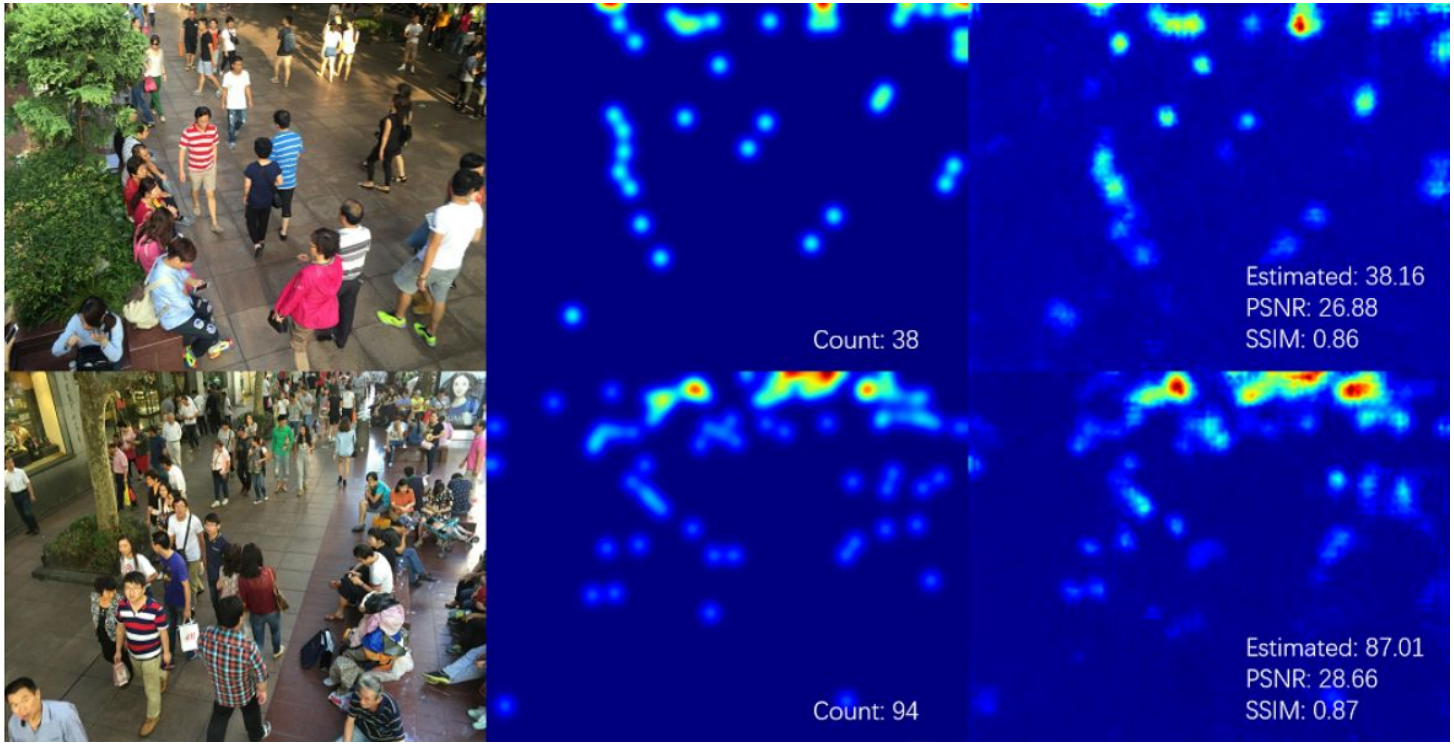
Table 5. Estimation errors on ShanghaiTech dataset

# Results (from arxiv)



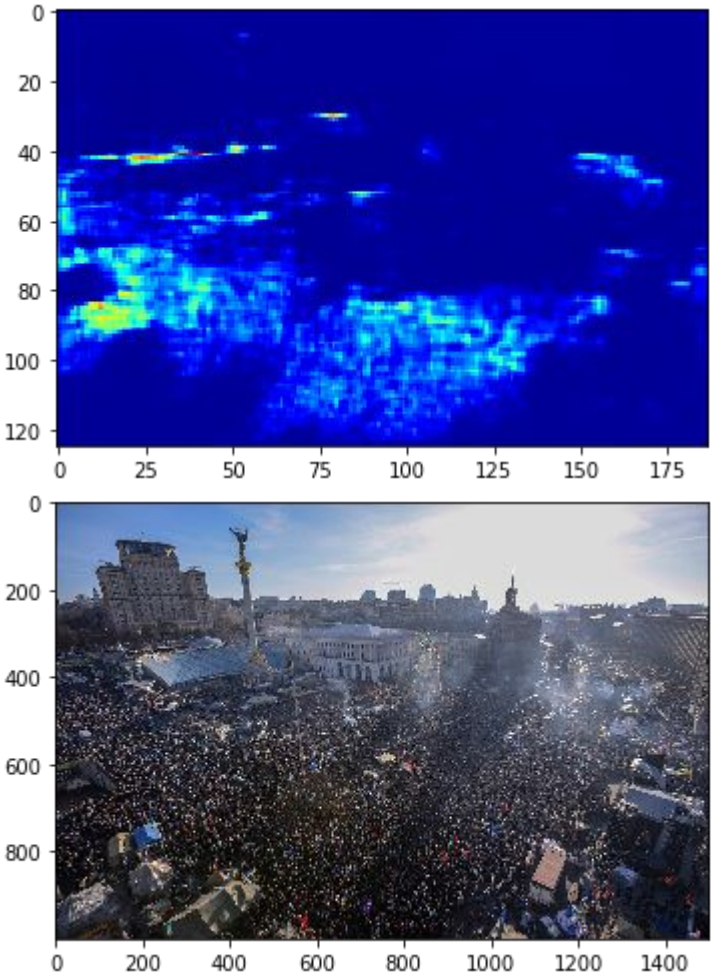

- Постоянный угол фото

# Results (from arxiv)



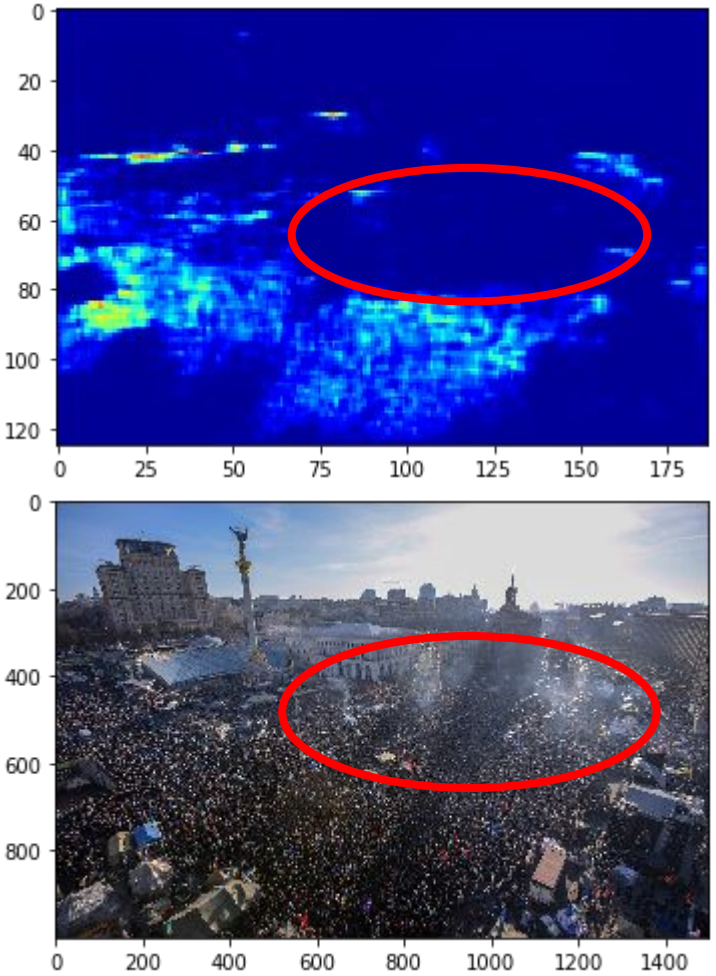

- Постоянный угол фото

# Results (Maidan 2014)




**Predicted Count : 2289**

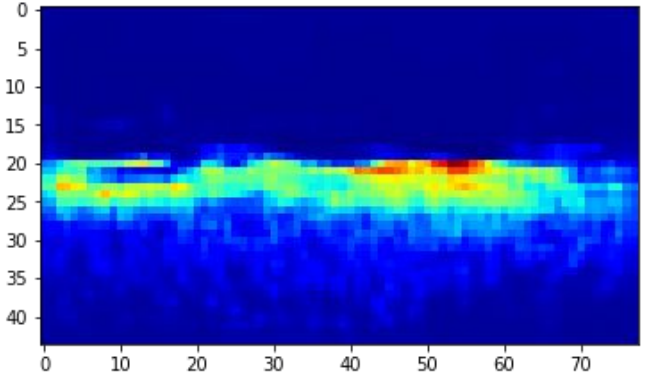
# Results (Maidan 2014)

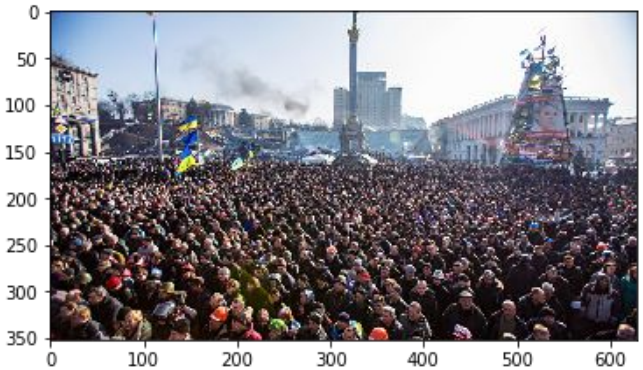




**Predicted Count : 2289**

Ракурс?

# Results (Maidan 2014)

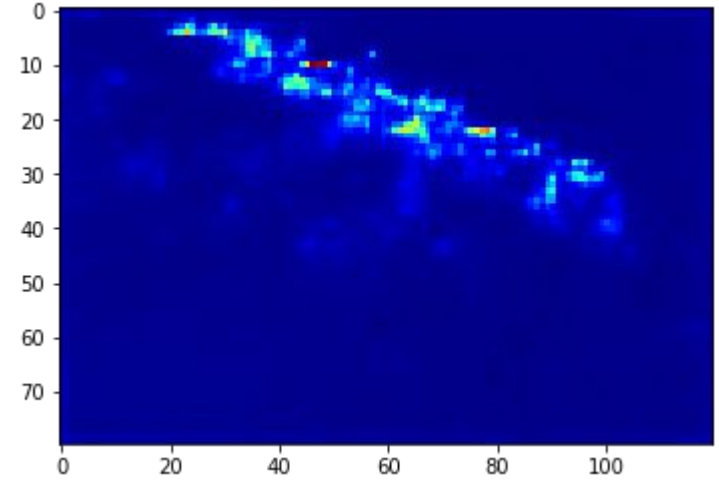



**Predicted Count : 1255**

...better?

# Results (ExPO)

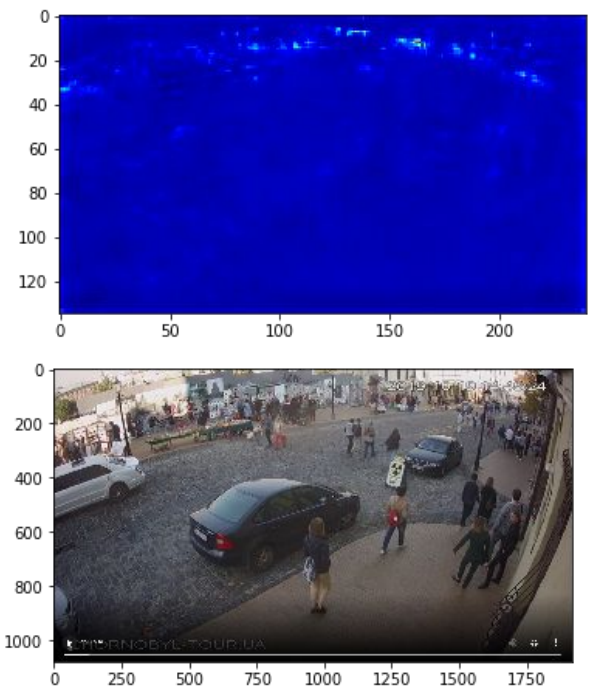




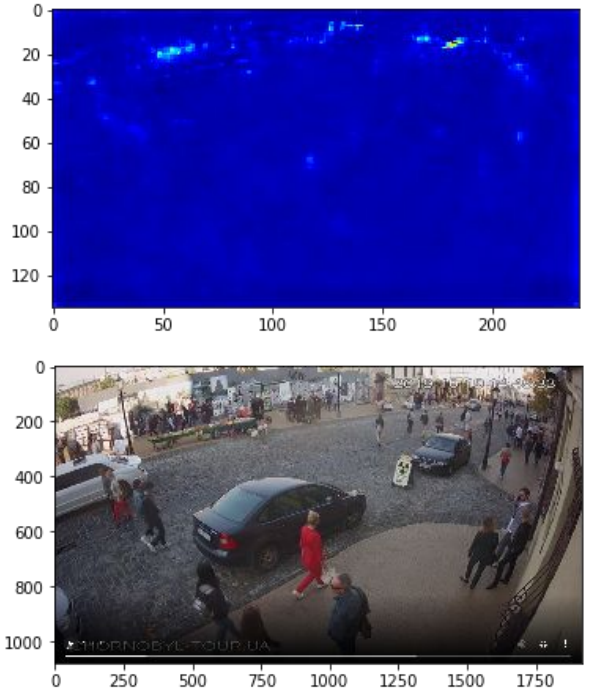
**Predicted Count : 134**

# Results (Kyiv - Андріївський узвіз)

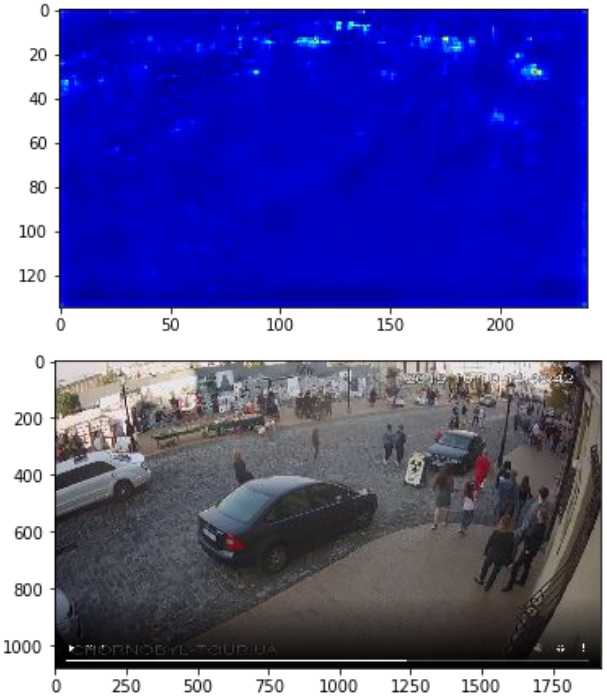
http://webcam.guru.ua/city/kiev/



Predicted Count : 144



Predicted Count : 150

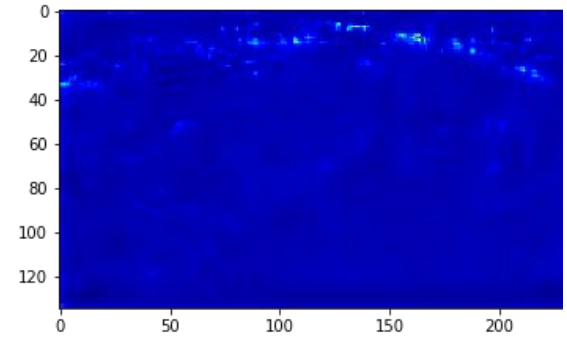


Predicted Count : 142

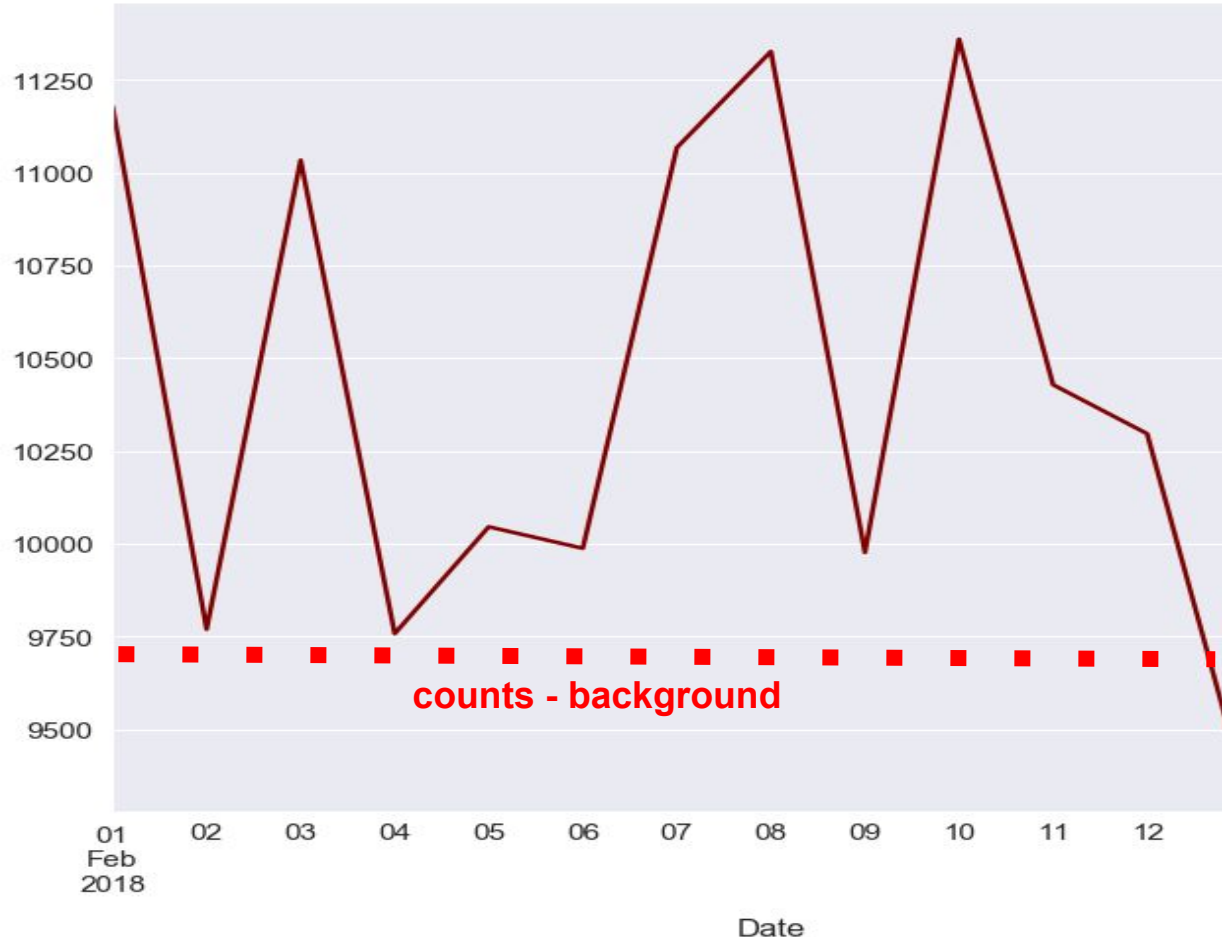
- study time tendencies?
- remove background?

# Results (Kyiv - Андріївський узвіз)

http://webcam.guru.ua/city/kiev/

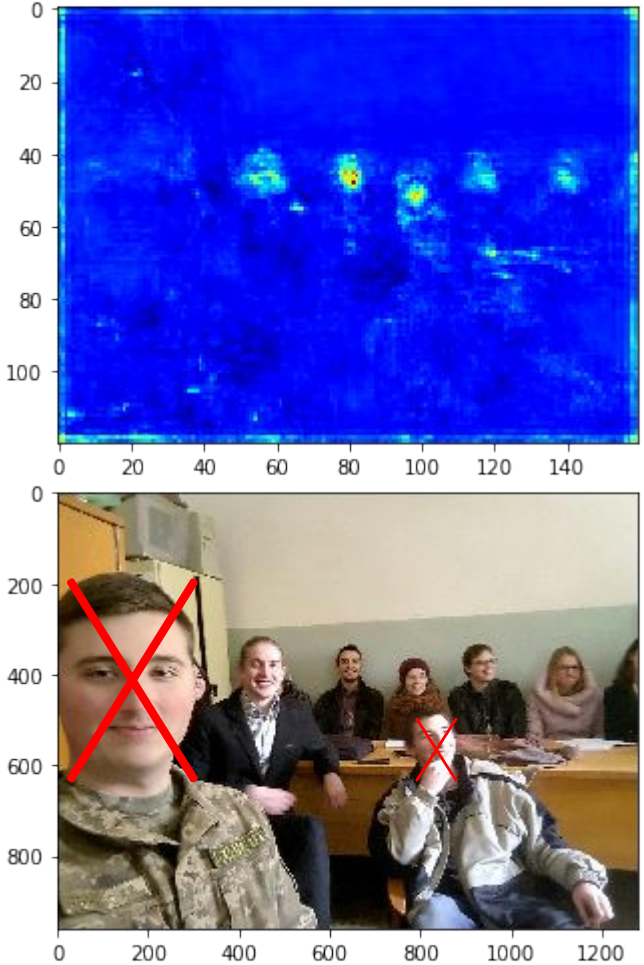



Predicted Count : 144




- study time tendencies?
- remove background?

# Results (scaling problem)


**Predicted Count : 91**

# Project discussion (possible expansion)

1. Try to analyse concentration of people from webcams around Kyiv
   http://webcam.guru.ua/city/kiev/

2. Try to do standalone device and install it somewhere

3. Оцінку кількості живої сили та техніки