Context-Aware Crowd Counting

Ananya Sankaranarayanan - MDS202105
Pragya Jaiswal - MDS202129
Prashant Bajpai - MDS202130

April, 2023

Abstract

Deep networks are used to estimate crowd density in contemporary approaches for counting people in crowded scenes. Usually, they apply the same filters to huge image portions or the entire image. They don't estimate local scale to correct for perspective distortion until after that. This is often accomplished by teaching an auxiliary classifier to pick the best kernel size out of a constrained list of options for predefined picture patches. As a result, these methods are limited in the range of contexts they can use and are not end-to-end trainable.
In this study, we provide a trainable end-to-end deep architecture that combines features collected from various receptive field widths and learns the relative weights of each feature at various image locations. In order to effectively anticipate crowd density, this method adaptively encodes the magnitude of surrounding information. Modern crowd counting techniques are outperformed by this approach, especially when there are significant perspective effects.

Contents

1 Introduction

Crowd counting is the process of estimating the number of individuals or objects present in a given area or scene from visual data, such as images or videos. Crowd counting is important for applications such as public safety, transportation, event planning, video surveillance, etc. The applications outweigh the challenges and hence methods of crowd counting have developed significantly over time. One of the earliest methods of Crowd counting, *Counting by Detection* which involved techniques like Monolithic Detection, Part-based detection and Shape matching. Next, came *Counting by regression*. This method, *counting-by-density* consists of two parts, one, low level features are extracted like edges, textures etc and two, these features are mapped to a regression function. Modern day counting techniques use the density of crowd in an image to estimate the number of people present. In this method, spatial information is taken into account by learning the mapping between local features and the density map. Gaussian Kernels are used for smoothing and reducing noise while creating density maps generally. The latest methods of crowd counting are a combination of some form of CNN and density detection.
One should change the receptive field size across images but these methods use same filter and pooling operations over the entire image and rely on the same receptive field everywhere. There have been attempts to fix this in the past by combining density maps extracted from image patches at different resolutions. These methods ignore the fact that scale varies continuously across the image. The resulting methods can only use a narrow range of receptive fields for the networks to remain of a manageable size, are not end-to-end trainable, cannot accommodate for quick scale changes because they assign a single scale to relatively large patches.
In this report, we have used the methods stated in Context Aware Crowd Counting that explicitly extracts features over multiple receptive field sizes and learns the importance of each such feature at every image location, thus accounting for potentially rapid scale changes.
Therefore, this is a method that integrates multi-scale contextual data into a trainable end-to-end pipeline for crowd counting and learns to take advantage of the appropriate context at each image location. We have worked on the Shanghai Tech dataset as well as a Penguin crowd dataset to test the performance of this algorithm.

2 Main Results

This paper introduces a new deep net architecture that takes into account the contextual information in the scene and encodes it into features that are then regressed to a density map. In notational terms, for a set of N training images $\{I_i\}_{1\leq i\leq N}$ with corresponding ground truth density maps $\{D_i^{gt}\}$, the goal is to learn a non-linear mapping $\mathcal{F}$ that maps an input image I_i to an estimated density map $D_i^{est}(I_i) = \mathcal{F}(I_i, \theta_i)$ such that the L^2 norm between the ground truth and the estimated density map is minimum.
The network structure used can be broadly divided into a front-end network, context network and a back-end/decoder network.

2.1 Network Architecture

2.1.1 Front-End Network

The first 10 layers of VGG-16 network makes up the front-end network of the architecture. VGG-16 network consists of 16 learnable layers: 13 convolutional layers and three fully connected layers. It also consists of 5 maxpool layers.
The front-end network uses the first 10 convolutional layers along with 3 maxpool layers.

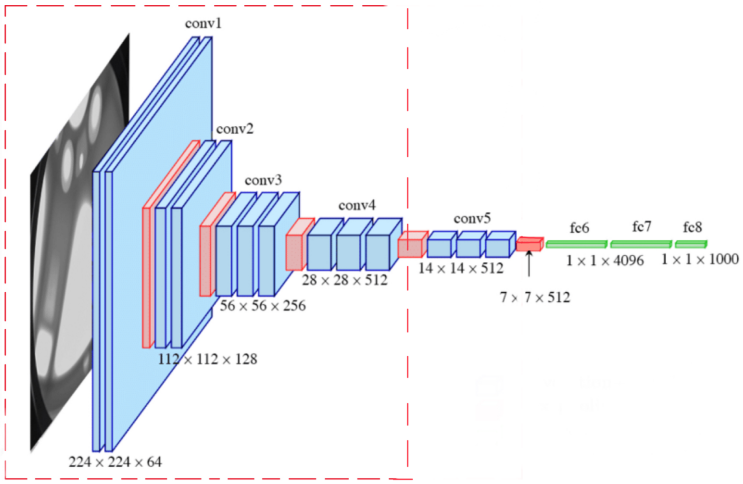

Figure 2 *: VGG-16 Network*
The dotted red line indicates the layers used in the context-aware architecture as a front-end network

Given an image I, the first 10 layers of the VGG-16 network outputs features of the form

$$f_v = \mathcal{F}_{vgg}(I) \qquad \dots (1)$$

The limitation of $\mathcal{F}_{vgg}$ is that it encodes the same receptive field over the entire image.
To account for different scales, the context network is used to extract multi-scale context information from the VGG features.

2.1.2 Context Network

The Context Network is designed to capture contextual information that can help improve crowd counting accuracy.
In order to extract the multi-scale features from the VGG features, we compute scale-aware features for each scale j, using Spatial Pyramid Pooling as:

$$s_j = U_{bi}(\mathcal{F}_j(P_{ave}(f_v, j), \theta_j)) \qquad \dots (2)$$

$P_{ave}(., j)$ is the spatial pyramid average pooling layer that divides the input feature map into a fixed number of sub-regions. The average of each sub-region is computed to finally get $k(j)$ x $k(j)$ blocks. The resulting context features are passed through a 1x1 convolutional layer $\mathcal{F}_j$ with weight parameters θ_j in order to combine the features across channels without a reduction in the dimension. The context features need to be combined as the spatial pyramid layer keeps the features independent for each channel, thereby lowering representational power of each scale. The resulting features are then up-sampled using bilinear-interpolation U_{bi} to get it to the same dimension as f_v.
Here, 4 different scales are used with corresponding block sizes $k(j) = \{1, 2, 3, 6\}$.
These scale aware features are then used to find the contrast features by subtracting from them the features from the front-end network. It is defined as:

$$c_j = s_j - f_v \qquad \dots (3)$$

This is done to capture the differences between the features at a specific location and those in the neighborhood in order to identify saliency in a given image. This is what helps us understand the

local scale of each region.
To have multi-scale context, different scales need to be weighted differently for each region. To this extent, the contrast features are passed through a 1x1 convolutional layer $\mathcal{F}_{sa}^{j}$ with weight parameters θ_{sa}^{j} followed by a sigmoid function to compute the weights ω_j for each scale. The resulting scale-specific weight map is of the form:

$$\omega_j = \mathcal{F}_{sa}^{j}(c_j, \theta_{sa}^{j}) \qquad \dots (4)$$

The final contextual features are computed as:

$$f_I = [f_v | \frac{\Sigma_{j=1}^{S} \omega_j \odot s_j}{\Sigma_{j=1}^{S} \omega_j}] \qquad \dots (5)$$

where $[\cdot|\cdot]$ denotes the channel-wise concatenation operation, and $\odot$ is the element-wise product between a weight map and a feature map. The contextual features are then passed on to the decoder network to obtain the estimated density maps.

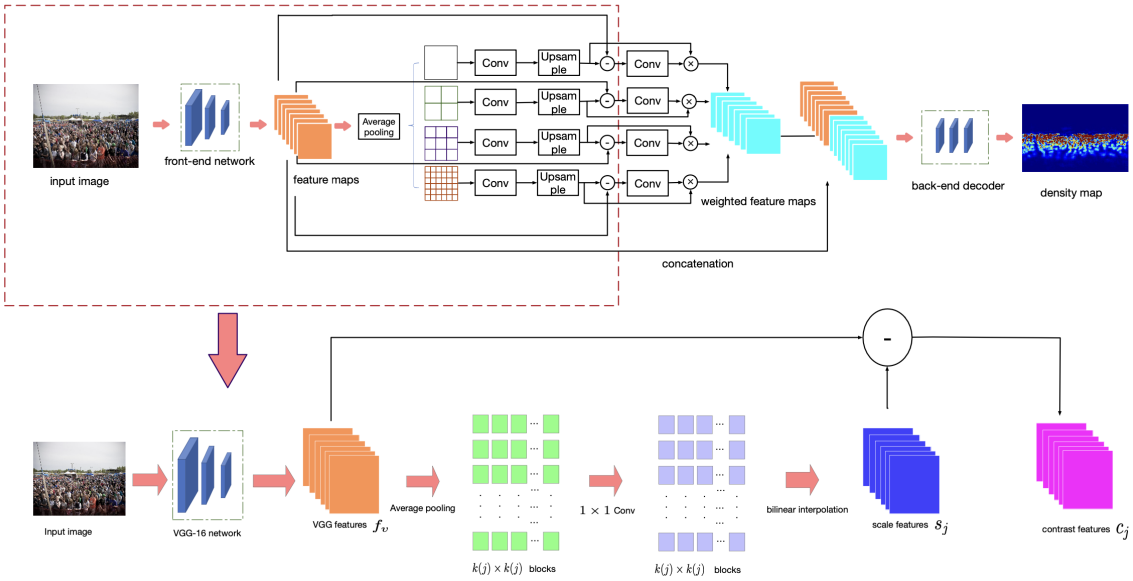

***Figure 2** : Context-Aware Network*
Images are fed to a font-end network (the first 10 layers of the VGG-16 network). The resulting local features are grouped in blocks of different sizes by average pooling followed by a 1×1 convolutional layer. To get it back to the original feature size, they are up-sampled and are used to form contrast features. Contrast features are further used to learn the weights for the scale-aware features that are then fed to a back-end network to produce the final density map. ***(Bottom)*** *The contrast features are the difference between local features and context features.*

2.1.3 Decoder Network

The contextual features f_I are passed on to the decoder/back-end network which consists of of several dilated convolutions.
Dilated convolutions introduce gaps or spacing between the values in the kernel used for convolution. This spacing is determined by the dilation rate. It can be thought of as "stretching" the filter by inserting empty spaces (or zeros) between the values in the filter which allows the filter to

cover a larger receptive field, without increasing the size of the filter. Up-sampling is not required with dilated convolutions because the dilated convolution operation itself can effectively enlarge the receptive field, allowing the network to capture context and multi-scale information without the need for additional up-sampling operations. In a regular convolutional operation, the filter (or kernel) is moved across the input data in a sliding window manner, and the size of the receptive field is determined by the size of the filter. However, in dilated convolutions, the spacing or gaps introduced in the filter allows it to cover a larger receptive field without increasing its size. This means that a dilated convolutional layer can capture context and features from a larger area of the input data compared to a regular convolutional layer, without the need for up-sampling.
The resulting output from this network is the required density map D_i^{est} for an input image I_i.

2.2 Geometry-Guided Context Learning

The contextual information from each region greatly varies due to perspective distortion which is further reflected in the scene geometry. When the scene geometry for an image becomes available, a modified architecture is used.
The scene geometry of an image I_i is represented by a perspective map M_i that encodes the number of pixels per meter in the image plane. This map is used as input to the front-end network however, in this case the input is of a single channel.

$$f_g = \mathcal{F}'_{vgg}(M_i, \theta_g) \qquad \dots (6)$$

To initialize the weights corresponding to this channel, the average the of the original three RGB channels are used. The perspective map is also normalized to lie within the same range as RGB images before passing it through the network.
The computation of the weight maps (Eq.4) is also modified to include the modified VGG features described above. The weight map for each scale is computed as:

$$\omega_j = \mathcal{F}^j_{gc}([c_j|f_g], \theta^j_{gc}) \qquad \dots (7)$$

These are then used in Eq.5 to get the corresponding contextual features and then passed through the decoder network to obtain the density map.

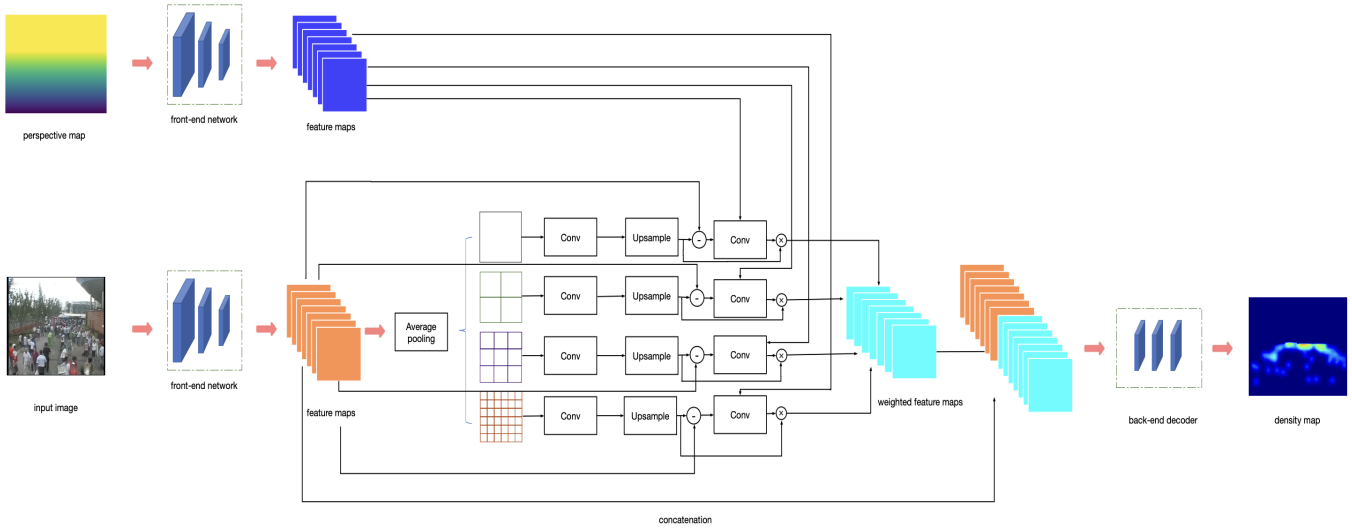

***Figure 3** : Modified Context-Aware Network*

A perspective map that encodes local scale is fed as input to the truncated VGG network (single channel input). Its output is concatenated to the original contrast features and the resulting scale-aware features are used to estimate density map.

2.3 Details of Implementation

To evaluate the output of the network, ground truth density maps are required. The ground truth density maps D_i^{gt} are obtained using manually annotated data. That is, each image I_i is associated to a set of points P_i^{gt} that denote the position of each human head in the scene. An image with ones in these positions and zeroes elsewhere are convolved with a Gaussian kernel $N^{gt}(p|\mu, \sigma^2)$ to get the ground truth density map.
The loss function used to train the network is the L^2 loss function. For batch size B, it is given by

$$L(\theta) = \frac{1}{2B}\Sigma_{i=1}^{B}||D_i^{gt} - D_i^{est}||_2^2$$

During training, random image patches of $1/4^{th}$ the size of the original image are cropped at different locations. These patches are mirrored to double the training set.
This method is applied on two datasets : Part B of Shanghai Tech dataset, and Penguin crowd dataset.

2.3.1 Shanghai Tech Dataset

The Shanghaitech dataset is a large-scale crowd counting dataset. It consists of 1198 annotated crowd images. The dataset is divided into two parts, Part-A containing 482 images and Part-B containing 716 images. Part-A is split into train and test subsets consisting of 300 and 182 images, respectively. Part-B is split into train and test subsets consisting of 400 and 316 images. Each person in a crowd image is annotated with one point close to the center of the head. In total, the dataset consists of 330,165 annotated people. Images from Part-A were collected from the Internet, while images from Part-B were collected on the busy streets of Shanghai.
We implemented the network on Part-B of the dataset. The ground truth was created for each of the 716 images and stored as .h5py files which took approximately 30 minutes. These ground truth images were used as target in the neural network that was trained. We created two .json files containing a list of image files names that were used for training and validation while training the network. Then the network was trained for 100 epochs (as compared to 1000 epochs as done by the authors of the paper). The batch size was 16, which is lesser than what is mentioned in the paper, due to GPU constraints. We used Adam optimizer as was suggested in the paper. The learning rate was 10^{-4} and the weight decay for Adam optimizer was set to $5\text{x}10^{-4}$. Training the network on this dataset took around 2.5 hours.

2.3.2 Penguin Crowd Dataset

The penguin dataset is a collection of images of penguin colonies in Antarctica coming from the larger penguin watch project, which was setup with the purpose of monitoring their changes in population. The images are taken by fixed cameras in over 40 different locations, which have been capturing an image per hour for several years. In order to track the colony sizes, the number of penguins in each of the images in the dataset is required. The penguin counts have been done with the help of citizens, where interested users can place dots on top of the penguins.
The annotations mainly consist of raw X-Y coordinates of dots placed by each of the volunteers.
We trained on 120 images corresponding to 6 different cameras, having an equal number of images from each camera. Ground truth and .json files were created in a manner similar to the one used for Shanghai dataset. We trained the model for 50 epochs with batch size 4. The learning rate used

was 10^{-4} and the weight decay for Adam optimizer was set to $5x10^{-4}$. Training the network on this dataset took around 1 hour.

3 Results and Observations

The evaluation metrics used are mean absolute error (MAE) and root mean squared error (RMSE).

3.1 Shanghai Tech Dataset

Evaluating the trained model on the test set containing 316 images, we got MAE as 9.86 and RMSE as 15.18. This differs from the metrics the authors got which were 7.8 and 12.2 respectively. This small difference in performance can be attributed to the difference in the number of epochs used for training (the authors used 1000 epochs which is 10 times the number of epochs we used for implementation).

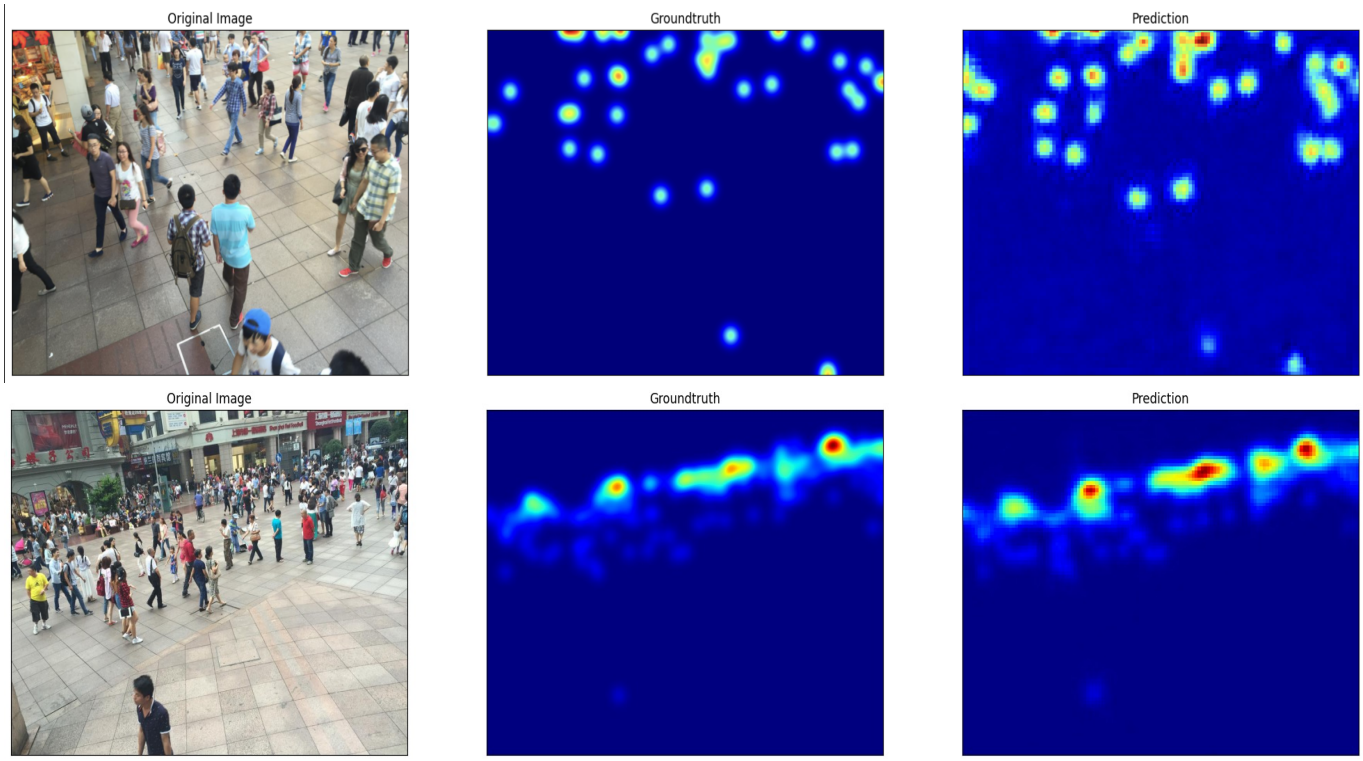

Figure 4 *: Crowd density estimation on ShanghaiTech*
Left to right : Input image, Ground truth, our prediction

We can see that the model performs fairly well at predicting the density map.

3.2 Penguin Crowd Dataset

The results from the trained model are as follows:

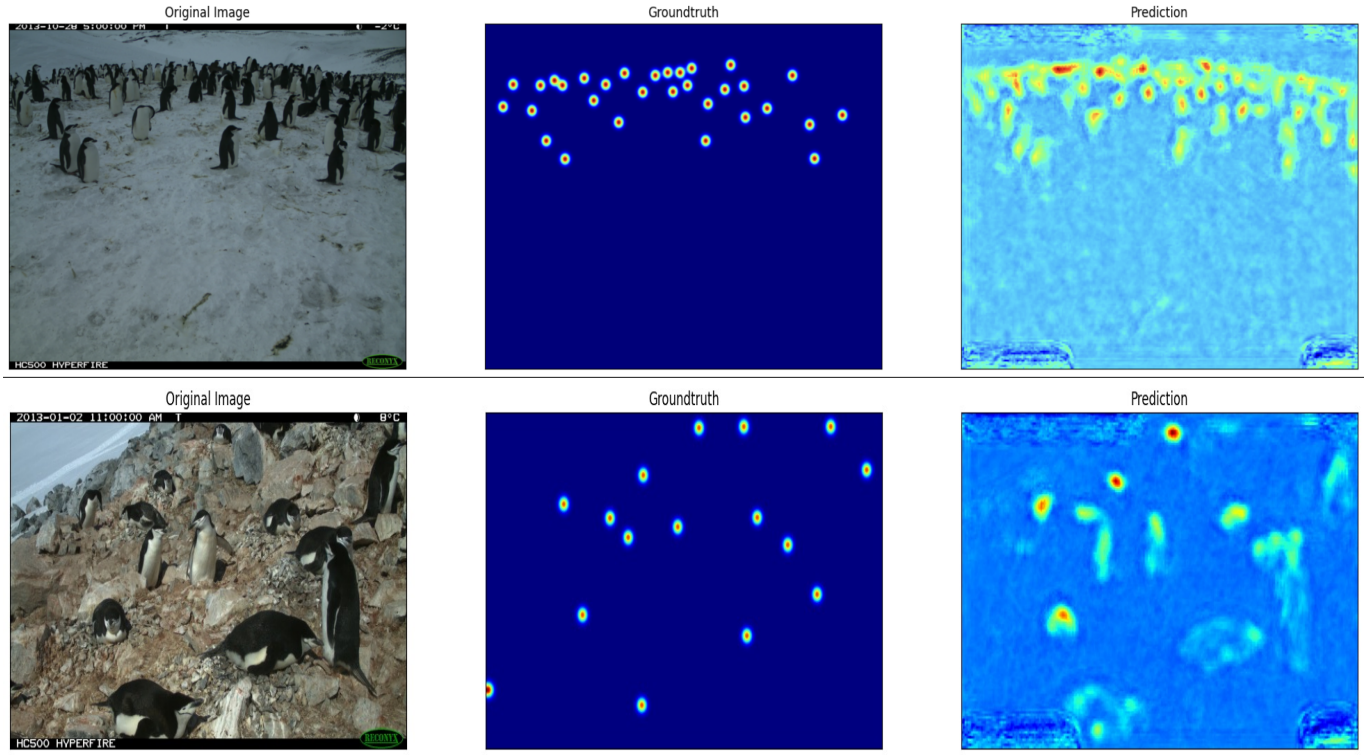

Figure 5 *: Crowd density estimation on Penguin Dataset*
Left to right : Input image, Ground truth, our prediction

It can be observed that the model identifies not just the head but also the feet (and body in some cases). Note that the model is not just detecting black regions, as in some cases the white torsos of penguins are also detected. However, the model's performance is still commendable considering the fact that it was only trained on 120 images for 50 epochs.
The model was also tested on a random image taken from the internet, outside the ones from the cctv cameras as shown below.

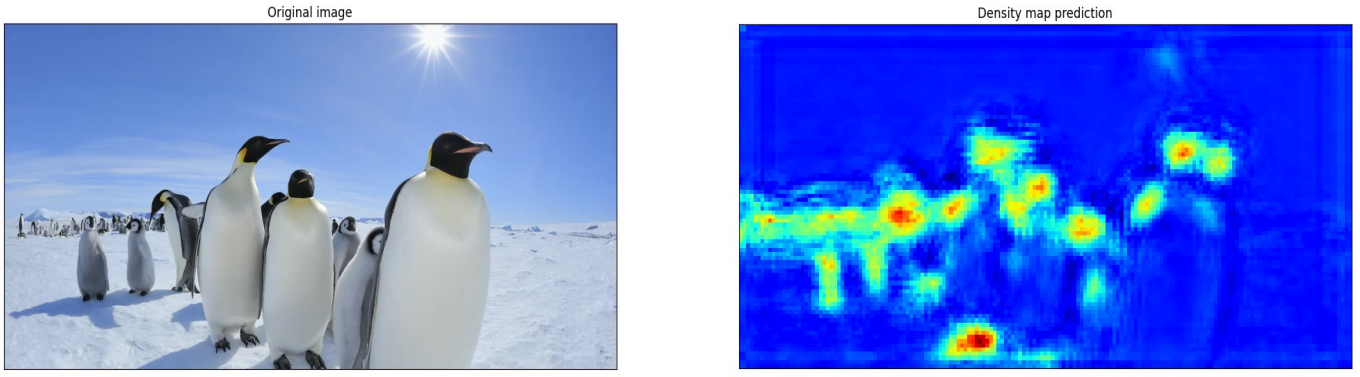

Figure 6
Left to right : Input image, our prediction

4 Current state of matters

The results of this context-aware crowd counting method outperforms all other methods mentioned in the literature of the paper, in terms of the evaluation metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The method is referred to as OURS-CAN.

Model	MAE	RMSE
Zhang et al.	32.0	49.8
ic-CNN	10.7	16.0
CSRNet	10.6	16.0
SANet	8.4	13.6
OURS-CAN	**7.8**	**12.2**

This shows that encoding multi-scale context adaptively, along with providing an explicit model of perspective distortion effects as input to a deep net, substantially increases crowd counting performance.
The paper further mentions that this method could be used in mobile cameras, such as those in drones after expanding the approach to process consecutive images simultaneously and enforce temporal consistency, which requires correcting the ground truth densities to account for perspective distortions.

5 References

1. Liu, W., Salzmann, M., & Fua, P.V. (2018). Context-Aware Crowd Counting. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 5094-5103.
2. Penguin Dataset - `https://www.robots.ox.ac.uk/~vgg/data/penguins/`
3. `https://github.com/weizheliu/Context-Aware-Crowd-Counting`
4. A Regression Based Model to Count Pedestrians in Crowds with Area, Shape and Texture Feature
5. Liu, Jiang & Gao, Chenqiang & Meng, Deyu & Hauptmann, Alexander. (2017). DecideNet: Counting Varying Density Crowds Through Attention Guided Detection and Density Estimation.
6. Ilyas, Naveed & Shahzad, Ahsan & Kim, Kiseon. (2019). Convolutional-Neural Network-Based Image Crowd Counting: Review, Categorization, Analysis, and Performance Evaluation. Sensors (Basel, Switzerland). 20. 10.3390/s20010043.
7. Elbishlawi, Sherif & Abdelpakey, Mohamed & Eltantawy, Agwad & Shehata, Mohamed & Mohamed, Mostafa. (2020). Deep Learning-Based Crowd Scene Analysis Survey. Journal of Imaging. 6. 95. 10.3390/jimaging6090095.
8. `https://encyclopedia.pub/entry/25356`
9. `https://www.analyticsvidhya.com/blog/2021/06/crowd-counting-using-deep-learning/`
10. Shanghai Tech Dataset - `https://drive.google.com/open?id=16dhJn7k4FWVwByRsQAEpl9lwjuV03jVI`