On-Board Crowd Counting and Density Estimation Using Low Altitude Unmanned Aerial Vehicles—Looking beyond Beating the Benchmark

Bartosz Ptak, Dominik Pieczyński, Mateusz Piechocki and Marek Kraft

Institute of Robotics and Machine Intelligence, Poznan University of Technology, Piotrowo 3A, 60-965 Poznan, Poland; bartosz.ptak@doctorate.put.poznan.pl (B.P.); dominik.piezynski@put.poznan.pl (D.P.); mateusz.piechocki@doctorate.put.poznan.pl (M.P.)

*Correspondence: marek.kraft@put.poznan.pl

Abstract: Recent advances in deep learning-based image processing have enabled significant improvements in multiple computer vision fields, with crowd counting being no exception. Crowd counting is still attracting research interest due to its potential usefulness for traffic and pedestrian stream monitoring and analysis. This study considered a specific case of crowd counting, namely, counting based on low-altitude aerial images collected by an unmanned aerial vehicle. We evaluated a range of neural network architectures to find ones appropriate for on-board image processing using edge computing devices while minimising the loss in performance. Through experiments on a range of neural network architectures, we also showed that the input image resolution significantly impacts the prediction quality and should be considered an important factor before going for a more complex neural network model to improve accuracy. Moreover, by extending a state-of-the-art benchmark with more in-depth testing, we showed that larger models might be prone to overfitting because of the relative scarcity of training data.

Keywords: deep learning; crowd counting; image processing; edge processing; embedded; UAV

1. Introduction

Object counting is one of the central problems in computer vision, and one can frame many problems in the field as such. Numerous applications, ranging from the microscopic (cells and microorganisms) to the macroscopic (persons, animals, and cars) scale, with a large variety of counted objects, have been presented. One of the most prominent applications involving counting the instances of a specific object type is crowd counting [1–3]. As the name suggests, the goal of the application is to estimate the count of the number of people, usually in a crowded scene, with the best possible accuracy. The crowd counting problem is often formulated alternatively as a crowd density estimation problem. The density estimation’s result is a density map image in which each pixel encodes the number of people in the given pixel’s location (usually a fractional number) [4,5]. Given the density map, the crowd count can be computed with a simple summation of all of the map’s components (pixel values). Aside from the simple numerical information, the count and density information can be used to derive an even more in-depth knowledge of the observed scene, enabling the detection of violent acts, behavioural analysis, and others [6–8]. The estimated size of crowds participating in public gatherings can be required for multiple reasons. Interestingly, it is oftentimes used during demonstrations and protests as an argument to settle debates [9].

While most of the literature and applications are dedicated to crowd counting with images coming from typical surveillance cameras, more and more attention is drawn to crowd counting using low-altitude aerial imagery, typically captured using unmanned aerial vehicles (UAVs). Compared to standard surveillance, using UAVs offers broader...
scope, less viewpoint variability, and more flexibility in deployment, making the captured footage a helpful tool in public safety and urban management applications. On the other hand, there are many domain-specific issues associated with the use of UAVs. Since most state-of-the-art image processing is performed using deep learning techniques [10], it is associated with a relatively high computational cost [11]. Deep learning-related workloads are usually deployed in the cloud or powerful workstations. In the case of the UAV applications, this would create the need for streaming the image data to a remote server, which is not a preferable solution since it forfeits the flexibility that comes with UAV use. Moreover, establishing the high-bandwidth communication between the UAV and the remote server might be problematic in real-world use cases. An alternative approach involving on-board processing is therefore preferable. This imposes some severe constraints on the UAV-mounted computational platform—it has to be powerful enough to handle the computational workload yet small and lightweight enough to enable mounting it on the UAV. Fortunately, the recent introduction of deep learning-capable embedded devices and computational accelerators enables on-board data processing, subject to these constraints.

This study evaluated a range of CNN-based methods for crowd counting in images registered by a UAV. The main contributions of the study are as follows. First, we evaluated a range of solutions with a range of neural network architectures using a variety of feature extractor backends to explore the trade-off between complexity and person density and count estimation accuracy. Second, all the aforementioned network architectures were evaluated with different resolution input images to answer whether and to what degree using higher resolution images can compensate for using a simpler neural network. All the benchmarks were performed using embedded devices dedicated to efficient neural network inference. The benchmarks reveal that the demonstrated simpler neural networks could successfully be implemented using on-board embedded hardware and perform the crowd counting directly on the UAV, in the edge processing mode. Third, we demonstrated through experiments that the larger, more complex models designed to beat the current state-of-the-art performance using the available benchmark data might be overfitting to the available datasets and would, therefore, be an inferior choice in terms of generalisation. Moreover, this suggests that the application calls for more diverse and more comprehensive benchmarks.

2. Related Work

Early approaches to the crowd counting task were based mainly on the detection of characteristic features (head and shoulders) using a classic sliding window approach to account for every person present in the image [12–15]. Although rapid progress has been made in object detection in images with the advent of deep learning and although state-of-the-art methods offer notable improvements in the quality of results over the classic methods [16–18], their performance is still sub-par in realistic conditions with dense crowds and background clutter. The issues with detection-based approaches were dealt with using methods that rely on direct regression of the number of persons in the image, usually based on some form of image feature descriptors [19,20]. The current state-of-the-art approaches are based on the estimation of the crowd density map directly from the image. The density map can be used to calculate the number of persons by adding its components. The function that relates the input image to the density map is determined using machine learning methods. The pioneering approach presented in [21] relied on linear mapping of the extracted features to the resulting density map, but more complex methods were soon to follow [22].

The introduction of convolutional neural networks (CNNs) has significantly improved the performance of a vast range of computer vision applications [23]. The capability for using the trained feature extraction to suit the target applications is a key performance driver when compared with approaches based on handcrafted features [24]. Crowd counting is not an exception here, and current state-of-the-art methods are generally based on convolutional neural networks performing the estimation of crowd density in the form of
a density image. However, the first approaches performed mostly density estimation by classification into a number of predefined density level classes [25] or adapted early CNN architectures to use their feature extractors in a framework that performs direct regression of person count from the input image [26]. The first use of CNN for direct estimation of a crowd density map and count was reported in [27]. The method also introduced a multi-column network—a network in which information from multiple independent feature extraction paths is fused to obtain the final result. The method was developed and expanded in multiple later works to deal with the problem of multiple scales at which the counted objects occur with a shallow and deep path usage [28], scale-specialised multiple paths and a pyramid of input image patches [29], specialised paths and a trained switching mechanism for path selection [30] or attention mechanisms [31]. While the multi-column networks are a significant step in the development of CNN-based crowd counting, enhancing prior art with scale awareness, they suffer from two prominent disadvantages. First, the more complex multi-column structure is more challenging to train than the corresponding single-column networks. Second, the operation on shared input information inevitably causes redundancy in extracted features across columns, increasing both the complexity of the networks and the computational cost. As a result, single-column networks have gained more attention in recent years due to their architectural simplicity and simpler training. The architectures of single-column networks use a range of mechanisms to deal with the problem of persons appearing at multiple scales. The solution presented in [32] is based on dilated convolutions to expand the receptive field of feature detectors. In contrast, other solutions rely on multi-scale deformable convolutions paired with attention [33] or architecture based on the U-Net encoder-decoder architecture [34] with an additional specialised decoder [35]. To improve the global detection context, visual transformers were adopted for the crowd counting task. The algorithm presented in [36] exploits Swin Transformer as a feature extractor and connects it with the traditional convolutional decoder. Similarly, [37] used the transformer encoder with a multi-scale regression head to build a density map, achieving state-of-the-art performance.

Until recently, the research dealt mostly with ground-perspective crowd counting, and applications involving UAV-registered data were uncommon. However, the increased popularity and recent affordability of UAVs have drawn the attention of researchers, resulting in the publication of dedicated datasets [38–41] and research. Crowd counting with UAVs brings forth unique challenges—perspective distortion is not as significant as scale, environment, and lighting changes. While dedicated, lightweight solutions tailored to UAVs exist [42], they are mostly designed to operate on images only. The best-performing solutions still rely on multi-column networks, feature pyramids, and complex backbones for feature extraction [40], hindering their deployment on embedded hardware. UAV crowd counting approaches may rely on additional information in the form of image difference or optical flow to integrate temporal information in the prediction pipeline for added robustness. This is possible since selected public UAV crowd-counting datasets contain consecutive frames from video sequences. However, in order to achieve the state-of-the-art performance, the architectures rely on relatively complex neural network architectures. For example, the solution given in [43] contains two independent paths for image and flow information processing. Each individual path is essentially a ResNet50 neural network with an added head. On top of that, the solution uses a dedicated head block for information fusion and final prediction generation.

While crowd counting using UAVs has certain advantages, such as flexibility in terms of deployment and the choice of an optimal point of view, it also brings forth certain challenges. Due to the size, weight, and power consumption constraints, the UAVs are usually fitted with embedded hardware that is not designed to handle significant computational workloads imposed by the use of CNNs. Additionally, solutions constructed with transformers are at an early stage of research and are not well optimised for inference on edge AI devices yet. On the other hand, transmitting the data to a ground station for processing is not always an option—offline processing may not be acceptable since results need to be
available in real-time, the monitored area may not provide the necessary communication infrastructure, etc. The problems may be solved using on-board processing, and the recent surge of interest in deep learning solutions has facilitated the development of embedded accelerators [44], often using hardware-specific techniques such as network quantisation, pruning, or binarisation [45]. These techniques result in more hardware-efficient, embedded solutions with low power consumption but often at the cost of performance. Nevertheless, the on-board processing of UAV images is gaining traction in a variety of applications as the most natural option.

3. Materials and Methods

In this section, we first overview the high-level overview of the research methodology. Then, we proceed with the details and present the dataset used in experiments, followed by detailed descriptions of the key elements of the described crowd-counting method.

3.1. Context and Methodology

Counting crowds in UAV-view images is a more challenging task than with CCTV camera images. The UAV-captured images may exhibit large variability in terms of the environment, zoom factor, perspective, and others. In addition, a flying robot is a machine that is itself in motion, introducing additional acquisition drift. This makes it hard to consider this a direct counting by object detection or regression task. Therefore, a method was adopted that is based on density maps and formulates the counting problem as a density regression task. For each location of a person present in the image, a Gaussian kernel whose sum of elements equals to one is added. This gives rise to a density map whose sum of individual elements is equal to the number of persons in the image. The location of each person is represented by a local maximum in the density map. The nature of the dataset and the sample density masks are shown in Figure 1. Each represents different features: left—low luminance, high altitude, high and compressed density (person count: 255); centre—low altitude, very high density (person count: 417); right—high luminance and low density (person count: 32). For this specific task it was shown that including optical flow information along with the input images improves the estimation performance. Regardless of the conditions, the main goal is to estimate the good quality feature map corresponding to people’s head position.

As the task is rather specific and relies on a deep neural network, the training requires a large amount of data to assure proper generalisation and prevent overfitting. In order to tackle this challenge, while ensuring the solution is capable of running using embedded on-board hardware, we explored a variety of architectures, including ones featuring a lightweight feature extraction encoder. Moreover, the evaluation is performed using a range of input image resolutions and includes a cross-validation analysis to gain insight on the generalisation performance using a recently proposed dataset [40]. As we are interested in a high processing framerate, we decided not to employ the sliding window...
approach and made a single prediction for the complete image acquired by the UAV. Aside from the increased processing time arising from the necessity of making multiple predictions for a single input image, the sliding window approach brings forth additional problems with merging the prediction results. Moreover, since we used the optical flow data to augment the input image information for improving the prediction accuracy, long processing times associated with the sliding window approach might also degrade the quality of the computed optical flow due to the lower sampling frequency. The sliding window approach might still be useful if the accuracy is of paramount importance and the associated additional computational cost is not a problem. Off-line processing of UAV-registered videos or image sequences might be an example of such an application [46].

The following is a detailed description of the elements of the solution.

3.2. Dataset

To investigate the effectiveness of the proposed method in UAV-based crowd counting, we used video sequences collected from real-world scenarios. We used the VisDrone 2020 Crowd Counting challenge dataset [40] that provides drone-view sequences of images with annotations for the task. The dataset consists of 112 challenging sequences, including 82 sets dedicated for training (2420 frames in total) and 30 intended for testing (900 frames in total). These groups of images are various due to luminance and the number of people, as shown in Figure 2.

Additionally, the videos from relatively low and high altitudes were significantly differentiated. These sequences were taken by UAV in urban environments—mainly showing streets, crossroads, sidewalks, and large squares. Since crowds are dynamically scattered across video frames, each crowd is surrounded by different backgrounds with different levels of clutter. As a ground truth, the authors provide human head position coordinates for each image, with over 400 k annotations in total. All images were registered with the same resolution equal to 1920 × 1080, which, with the high altitude of the drone, makes it extremely challenging to extract features that describe the object.

![Figure 2. Average person count in each of the VisDrone 2020 CC sequences.](image)

3.3. Our Approach

This subsection explains our approach in the context of neural network inference on edge devices and the usage of sequential aspects of images.

3.3.1. Evaluation Hardware

Power consumption, small dimensions, and low weight are essential for UAV-mounted data processing units. With this in mind, the following computational platforms were evaluated.

**Nvidia Xavier NX** is a system-on-chip designed for embedded applications, integrating multi-core embedded 64-bit CPU and an embedded GPU in a single integrated circuit [47]. The CPU is a six-core, high-performance 64-bit Nvidia Carmel ARM-compatible processor clocked at up to 1.9 GHz (with two cores active, 1.4 GHz otherwise). The GPU is based on Volta architecture, with 384 CUDA cores. Moreover, it is equipped with 64 tensor cores and two Nvidia deep learning accelerator (NVDLA, Santa Clara, CA, USA) engines dedicated to deep learning workloads. The CPU and the GPU both share the system
memory. The development platform is fitted with 8 GB low power DDR4 RAM clocked at 1600 MHz with a 128-bit interface, which translates to 51.2 GB/s bandwidth.

**Neural Compute Stick 2 (NCS2)** is a fanless device in a USB flash drive form factor used for parallel processing acceleration. It is powered by a Myriad X VPU (vision processing unit), which can be applied to various tasks from the computer vision and machine learning domains. The high performance is facilitated by using 16 Streaming Hybrid Architecture Vector Engine (SHAVE) very long instruction word (VLIW) cores within the SoC and an additional neural computing engine. Combining multi-core VLIW processing with ample local storage for data caching for each core enables high throughputs for SIMD workloads. It is directly supported by the OpenVINO toolkit, which handles neural network numerical representation and code translation [48,49]. The USB deep learning accelerators were mounted on the Raspberry Pi 4 single-board computer with a quad-core ARM Cortex A72 processor and 4 GB of RAM.

### 3.3.2. Models

The basic idea of our approach was to deploy an end-to-end CNN-based model to generate the density estimation map on edge AI devices. This requires a balance between performance accuracy and model size—both the number of parameters and memory consumption are important during inference. With limited resources on board the UAV, single-column models are the preferred option. Therefore, in our research, we focused on the segmentation encoder–decoder architectures: UNet [34], UNet++ [50], and DeepLabV3+ [51]. In general, the encoder–decoder architecture can be considered as a U-Net structure. The encoder aims to reduce the spatial dimension of feature maps and extract more high-level semantic features. The decoder intends to emphasise the details of the feature and reconstruct the original spatial dimension. Therefore, after each dimension reduction in the encoder, higher-level features are passed to the decoder to preserve a higher amount of spatial information. As an encoder, we explored popular possibilities that satisfy the performance requirements. We used the ResNet [52] family as a starting point, which has the widest generalisation capabilities for various applications. Additionally, we looked at EffcientNet [53], which has a state-of-the-art claim for certain tasks. MNASNet [34] in the variant with Squeeze-and-Excitation (SEMNASNet) and MixNet [55] have been tested because they achieve high metrics on mobile devices. This is possible due to the use of depthwise convolutions that perform the convolution kernel for each channel separately, which reduces the number of parameters and cost. All models’ implementations are provided by the Segmentation Models PyTorch library [56].

### 3.3.3. Optical Flow

Image sequences provide a wide range of additional information. Analysis of motion in the image is one of them. We adopted the Dense Inverse Search [57] (DIS) method from the OpenCV library. While a number of methods based on deep learning have been proposed for optical flow estimation [58], they are typically demanding in terms of computation, especially when high-resolution optical flow information is needed. Due to on-board hardware limitations, a decision was made to devote all parallel processing capabilities to the execution of crowd density estimation and counting. DIS can run at sufficient speed even on embedded processors, leaving enough headroom for other functionality. Moreover, the optical flow estimation error of DIS is relatively low, especially considering its speed. For reference, the processing times for embedded devices with ARM processors are shown in Table 1. The algorithm allows us to select the speed vs. precision trade-off according to the expected time constraints.

The output of the above algorithm is dense optical flow matrices providing information about the motion of objects in both the x and y axes. We exploited these features as extra information by concatenating them alongside the RGB image as additional channels. The DIS values need to be calculated for each image in sequence as a part of the data
preprocessing function. We provided a graphical interpretation for the example sequences in Figure 3.

Table 1. Effect of changing the optical flow generation mode for 1280 × 736 resolution.

| Flow Mode | NVIDIA Jetson Xavier NX | Raspberry Pi 4 |
|-----------|-------------------------|----------------|
|           | Average Time (s) | Performance (img/s) | Average Time [s] | Performance (img/s) |
| Medium    | 0.130 ± 0.009 | 7.747 ± 0.426 | 0.405 ± 0.002 | 2.471 ± 0.013 |
| Fast      | 0.034 ± 0.008 | 30.814 ± 4.346 | 0.091 ± 0.001 | 10.968 ± 0.118 |
| Ultrafast | 0.011 ± 0.001 | 93.011 ± 7.286 | 0.028 ± 0.001 | 35.326 ± 0.325 |

Figure 3. Example frames of using DIS optical flow algorithm. The left column presents the original input images. The middle column represents the x-axis component, whereas the y-axis component of flow is illustrated on the right column.

3.3.4. Data Augmentation

In the computer vision tasks, image augmentation is an indispensable regularisation technique for dealing with overfitting in deep neural networks. In our crowd counting pipeline, we applied a set of image augmentations to deal with the small dataset size in order to improve the generalisation capability. The operation on sequences of images requires extra care. Geometric augmentations applied across all images in the sequence must be coherent. Moreover, using additional information (such as optical flow) aside of image data must be treated separately, with a subset of selected augmentations. For example, applying gamma shift or channel shuffle augmentations to the optical flow data might render it useless. For this purpose of applying a set of transformations to the input images, Albumentations [59], an open-source library for fast and flexible processing, was used. The augmentations were grouped into three following sub-blocks.

Values level augmentations. These augmentations were only applied to RGB channels and affected the pixel values. RandomGamma and ColorJitter were used. These operations help to reduce the gaps between different illumination and time of the day in the sequences. Colour Jitter also improves the model generalisation for location variations.

Transforms level augmentations. This technique was applied to all input image channels. An Affine Transform, which preserves points, straight lines and planes, is used. The operation regulates the algorithm for dynamic movements of the UAVs: rotation, scaling, and translation. Additionally, as suggested in [60], geometric transformations strengthen the confidence of the models’ responses.
Sample level augmentations. Fusing multiple samples together has a positive impact on the training process and allows for efficient use of a dataset. We used the Mosaic method that was proposed in YOLOv4 [16]. The method mixes four training images to make the sample more generalisable than the standard context. The advantage of this augmentation is the use of natural images to improve model regularisation.

3.4. System Architecture

The proposed system was successfully implemented for real-world cases on an embedded UAV on-board device. Figure 4 presents the architecture of a vision system addressed to the challenge of counting crowds in near real-time constraints. It contains three stages:

- **Camera image grabbing.** The image is read from a video source and converted to greyscale.
- **Optical flow calculation.** Based on the current image and frames history, the system computes the DIS Optical Flow. The new optical flow is then wrapped with its previous values. For the next value, this means that not only is the past frame relevant, but the previous motion is also important.
- **Neural network inference.** In the final stage, the input information is concatenated into a single image with additional channels. Next, the tensor is scaled to the resolution of the convolutional model input and forwarded to inference. This step is necessary since the experiments are performed using a range of neural network input image resolutions. The output mask values are summed to provide the current outcome.

![Figure 4. Crowd-counting model inference pipeline.](image)

4. Experiments

Initially, only the three increasingly complex encoder architectures (ResNet18, ResNet34, and ResNet50) were tested with three neural network architectures (DeepLabV3+, UNet, and UNet++). Since the UNet-based versions consistently offered good processing speed vs. accuracy trade-off across all variants, additional UNet and lightweight encoder variants that were developed with the application on mobile devices were tested in addition to the aforementioned ResNet encoders. A detailed description of the full set of experiments is given below, assuring the ability to use the converted embedded devices for on-board processing. For this reason, we have conducted a series of measurements on popular hardware setups. The model architecture and weights were converted from the PyTorch format to an Open Neural Network Exchange (ONNX) [61] representation (using OPSET (ONNX runtime groups layers compatibility and specific version standards into operator sets called OPSET) 11), which allows high portability between platforms. Next, we used the ONNXRuntime framework (ver. 1.10.0, Linux Foundation, San Francisco, CA, USA) [62], which provides inference using device native libraries: TensorRT (ver. 8.0.1.6, NVIDIA,
Santa Clara, CA, USA) for the Jetson NX and OpenVINO (ver. 2021.4.2, Intel, Santa Clara, CA, USA) for the MYRIAD device. The conversion pipeline is illustrated in Figure 5.

Figure 5. Crowd-counting model conversion pipeline.

Both native libraries provide FP32 and FP16 precision (Jetson NX also enables quantisation to INT8); however, FP16 was adopted since model fitting was performed with this precision setting. The measurements were repeated for the tested models and input resolutions. The model response time (latency) and the image processing ability (performance) were evaluated. Table 2 shows the results for NVIDIA Jetson NX. Meanwhile, the results for NCS2 are shown in Table 3. The obtained values exclude both preprocessing and postprocessing process time. The information E1 indicates the error that occurred during the inference arising from the host not having enough memory (“FIFO0 due to not enough memory on device” error). The E2 indicates that the MYRIAD device has no memory left (“Not enough memory to allocate intermediate tensors on remote device”). Figure 6 compares the performance of the different model versions depending on the resolution. Note that there are two different axes: the left for the NVIDIA Jetson NX and the right for the NCS2 connected to the Raspberry Pi 4.

Table 2. One-batch inference speed (images per second) measurements of crowd-counting models for NVIDIA Jetson Xavier NX.

| Model       | Encoder       | Input Resolution |
|-------------|---------------|------------------|
|             |               | 480 × 288 | 640 × 384 | 960 × 540 | 1280 × 736 |
| U-Net       | resnet18      | 60.367 ± 0.688 | 37.127 ± 0.349 | 18.793 ± 1.445 | 10.879 ± 0.863 |
|             | resnet34      | 45.546 ± 0.481 | 27.970 ± 0.376 | 14.464 ± 1.013 | 8.367 ± 0.640 |
|             | resnet50      | 57.265 ± 1.490 | 35.131 ± 0.171 | 17.916 ± 1.191 | 10.333 ± 0.812 |
|             | mixnet_s      | 51.561 ± 0.729 | 31.341 ± 0.135 | 16.257 ± 1.229 | 9.338 ± 0.764 |
|             | semnasnet_075 | 82.018 ± 1.233 | 50.846 ± 0.636 | 25.949 ± 0.188 | 15.016 ± 0.049 |
|             | semnasnet_100 | 78.074 ± 1.904 | 46.919 ± 0.412 | 23.816 ± 2.031 | 13.892 ± 1.175 |
|             | efficientnet-b0 | 91.058 ± 7.687 | 63.451 ± 0.707 | 33.104 ± 2.919 | 19.241 ± 1.888 |
| U-Net++     | resnet18      | 35.856 ± 0.416 | 21.419 ± 0.129 | 10.368 ± 0.636 | 5.762 ± 0.431 |
|             | resnet34      | 32.417 ± 0.373 | 19.239 ± 0.085 | 9.537 ± 0.487 | 5.270 ± 0.384 |
|             | resnet50      | 14.195 ± 0.058 | 8.045 ± 0.118 | 3.652 ± 0.021 | 1.982 ± 0.023 |
| DeepLabV3+  | resnet18      | 91.058 ± 7.687 | 63.451 ± 0.707 | 33.104 ± 2.919 | 19.241 ± 1.888 |
|             | resnet34      | 72.688 ± 1.980 | 47.103 ± 0.353 | 24.402 ± 1.884 | 14.272 ± 1.236 |
|             | resnet50      | 53.308 ± 1.790 | 33.778 ± 0.216 | 17.368 ± 1.103 | 10.055 ± 0.810 |
Table 3. One-batch inference speed (images per second) measurements of crowd counting models for Raspberry Pi 4 with Intel Neural Compute Stick 2.

| Model   | Encoder | Input Resolution |
|---------|---------|------------------|
|         |         | 480 × 288  | 640 × 384  | 960 × 540  | 1280 × 736 |
| UNet    | resnet18| 5.450 ± 0.050| 3.160 ± 0.021| 1.305 ± 0.002| 0.801 ± 0.002|
|         | resnet34| 4.320 ± 0.030| 2.536 ± 0.010| 1.034 ± 0.001| 0.612 ± 0.001|
|         | resnet50| 2.646 ± 0.012| 1.709 ± 0.016| 0.637 ± 0.001| E1           |
|         | mixnet_s| 4.372 ± 0.030| 2.732 ± 0.021| 1.179 ± 0.004| 0.700 ± 0.002|
|         | semnasnet_075| 6.217 ± 0.053| 3.665 ± 0.051| 1.500 ± 0.010| 0.913 ± 0.006|
|         | semnasnet_100| 5.839 ± 0.066| 3.462 ± 0.023| 1.469 ± 0.006| 0.867 ± 0.006|
|         | efficientnet-b0| 3.851 ± 0.027| 2.351 ± 0.018| 1.006 ± 0.001| 0.619 ± 0.002|
| UNet++  | resnet18| 2.528 ± 0.012| 1.368 ± 0.002| E2           | E2           |
|         | resnet34| 2.269 ± 0.012| 1.236 ± 0.001| E2           | E2           |
|         | resnet50| 0.715 ± 0.001| E2           | E2           | E2           |
| DeepLabV3+ | resnet18| 2.427 ± 0.013| 1.844 ± 0.008| 0.730 ± 0.034| 0.337 ± 0.001|
|         | resnet34| 1.567 ± 0.094| 1.220 ± 0.005| 0.436 ± 0.021| 0.224 ± 0.005|
|         | resnet50| 1.082 ± 0.136| 0.805 ± 0.121| 0.344 ± 0.023| 0.159 ± 0.012|

We adopted two widely used crowd counting metrics to evaluate the proposed method. The Mean Absolute Error (MAE, M1, Equation (1)) was mainly used, which was computed between the number of people estimated and the ground-truth, as the evaluation metric. To cope with the large variance in the number of people in the sequences, the additional relative metric Normalized Absolute Error [63] (NAE, M2, Equation (2)) was included.

\[
\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| 
\]

(1)

\[
\text{NAE} = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{\hat{y}_i} 
\]

(2)

A series of experiments were performed to investigate the accuracy achieved by all the models and configurations. The range of configurations included various model structures, encoder backends, and input resolutions. Basic and mosaic augmentation was used for all the cases. Models input tensors contained five channels images: three colour channels and two optical flow channels. Table 4 reports the metrics achieved in our research. To obtain representative results, we used five-fold cross-validation. Figure 7 illustrates the changes in the MAE metric with the increase in model input resolution. The number of folds was not increased beyond that, since it would reduce the size of the validation set too much.
Figure 6. The model performance visualisation (given in frames per second) for various resolutions and both devices NVIDIA Jetson NX (left scale, blue) and Raspberry Pi4 + NCS2 (right scale, red). R1–R4 represent the following input sizes: 480 × 288, 640 × 384, 960 × 540, and 1280 × 736.
Figure 7. The visualisation of model results (People MAE) for various input image resolutions benchmarked on the Jetson NX platform. R1–R4 represent the following input sizes: 480 × 288, 640 × 384, 960 × 540, and 1280 × 736. All the subdiagrams are represented on the same axis scale.
Table 4. Metrics for researched crowd-counting models.

| Model      | Encoder | Input Resolution | M1     | M2     | M1     | M2     | M1     | M2     | M1     | M2     |
|------------|---------|------------------|--------|--------|--------|--------|--------|--------|--------|--------|
|            |         | 480 × 288 (R1)   |        |        |        |        |        |        |        |        |
| UNet       | resnet18| 33.85 ± 9.30     | 0.27 ± 0.07 | 26.30 ± 7.56 | 0.20 ± 0.05 | 23.47 ± 6.31 | 0.17 ± 0.05 | 20.23 ± 2.78 | 0.16 ± 0.05 |
|            | resnet34| 29.34 ± 9.35     | 0.24 ± 0.10 | 27.90 ± 6.10 | 0.22 ± 0.06 | 22.75 ± 3.55 | 0.18 ± 0.05 | 19.14 ± 4.66 | 0.14 ± 0.03 |
|            | resnet50| 28.92 ± 7.78     | 0.22 ± 0.04 | 27.37 ± 5.95 | 0.21 ± 0.05 | 20.73 ± 3.59 | 0.16 ± 0.06 | 22.53 ± 7.80 | 0.16 ± 0.02 |
|            | mixnet_s| 28.90 ± 8.72     | 0.20 ± 0.04 | 24.23 ± 5.08 | 0.19 ± 0.04 | 24.36 ± 5.60 | 0.19 ± 0.06 | 20.38 ± 5.41 | 0.15 ± 0.04 |
|            | semnasnet_075 | 26.99 ± 6.08 | 0.18 ± 0.04 | 26.62 ± 8.49 | 0.21 ± 0.10 | 25.19 ± 10.40 | 0.164 ± 0.02 | 21.83 ± 6.04 | 0.16 ± 0.02 |
|            | semnasnet_100 | 27.53 ± 5.57 | 0.21 ± 0.05 | 25.46 ± 6.14 | 0.17 ± 0.04 | 21.67 ± 8.77 | 0.15 ± 0.02 | 18.85 ± 5.58 | 0.13 ± 0.02 |
|            | efficientnet-b0 | 26.89 ± 6.06 | 0.21 ± 0.05 | 24.28 ± 2.27 | 0.18 ± 0.03 | 20.75 ± 4.01 | 0.15 ± 0.02 | 20.43 ± 4.66 | 0.14 ± 0.03 |
| UNet++     | resnet18 | 29.27 ± 6.47     | 0.22 ± 0.06 | 26.13 ± 6.92 | 0.21 ± 0.06 | 19.79 ± 5.18 | 0.14 ± 0.03 | 19.26 ± 3.79 | 0.15 ± 0.03 |
|            | resnet34 | 30.56 ± 6.53     | 0.23 ± 0.07 | 29.01 ± 6.79 | 0.21 ± 0.06 | 28.47 ± 5.06 | 0.21 ± 0.07 | 20.42 ± 2.86 | 0.15 ± 0.05 |
|            | resnet50 | 30.85 ± 9.68     | 0.22 ± 0.07 | 26.22 ± 4.03 | 0.21 ± 0.04 | 20.53 ± 5.67 | 0.15 ± 0.01 | 20.66 ± 3.05 | 0.15 ± 0.03 |
| DeepLabV3+ | resnet18 | 30.82 ± 9.24     | 0.27 ± 0.09 | 27.43 ± 1.69 | 0.24 ± 0.08 | 23.97 ± 4.89 | 0.18 ± 0.06 | 23.63 ± 6.24 | 0.17 ± 0.04 |
|            | resnet34 | 30.03 ± 11.91    | 0.24 ± 0.08 | 23.30 ± 5.76 | 0.18 ± 0.07 | 21.07 ± 5.15 | 0.17 ± 0.05 | 19.27 ± 4.33 | 0.15 ± 0.03 |
|            | resnet50 | 26.27 ± 5.91     | 0.20 ± 0.05 | 27.51 ± 7.83 | 0.21 ± 0.06 | 21.00 ± 4.79 | 0.15 ± 0.04 | 20.55 ± 6.56 | 0.14 ± 0.03 |

The scatterplot visualising the mean accuracy and processing speed of each neural network variant at the highest tested resolution is shown below.

5. Discussion

The extensive testing of various combinations of network architectures and input resolutions results in multiple findings. According to this research, on-board crowd counting is possible, and lightweight approaches can yield satisfactory results. The general rule is: the better the input resolution, the better the model performance in terms of accuracy. This trend was observed for all the tested architecture and encoder variants. The choice of the encoder and overall neural network architecture was less impactful in comparison with image size selection but can still significantly influence the metrics. This shows that one might forego using complex neural network architectures and achieve a similar performance using simpler ones just by using higher-resolution images.

Even at the highest resolution, the Xavier NX device is capable of performing the predictions with a speed of over 10 frames per second for the UNet configuration with a lightweight ResNet encoder while achieving good accuracy on average. Such processing speed is sufficient for this type of application due to the slow relative movement of the objects in the frame. Please note that one also has to account for the time necessary to compute the optical flow. The fastest model at the highest resolution overall was the DeepLabV3+ combined with the ResNet18 encoder. Unfortunately, this was also the worst model in terms of accuracy. Of the additional lightweight, edge processing-oriented encoder architectures used as a part of UNet architecture, the SemiNasNet 100 achieved the best average accuracy among the tested models while also being faster than all the ResNet variants. The UNet++ variants performed consistently well in terms of average accuracy. Interestingly, the densely connected decoder architecture that is characteristic of this model seems to reduce the standard deviation of the predictions. Unfortunately, this comes at a steep increase in processing times—all the tested UNet++ models were the slowest among the tested architectures (Figure 8).
Figure 8. The comparison of models and encoders in terms of achieved performance and accuracy. The chart presents the results for the highest tested resolution (1280 × 736). The performance metric was measured during inference on the Jetson NX platform.

Most of the findings regarding the inference speed using the Xavier NX translate to the Intel Neural Compute Stick 2 device. Although the overall processing speed is strongly limited, the SemNasNet encoders still outperform ResNet architectures. The major difference is the degraded throughput of the DeepLabV3+ model. This suggests that the Neural Compute Stick’s implementation of the dilated convolution algorithm is suboptimal. Nevertheless, such hardware configuration is suited only for the lowest input image resolution. Sample input images and their prediction for the highest evaluated input image resolution on the Xavier NX platform are shown in Figure 9.

One of the notable differences between this work and related work utilising the VisDrone dataset is the five-fold validation used to obtain the final performance metrics for different models. As the results present a large standard deviation between different splits, we conclude that the VisDrone dataset’s size makes it unsuitable for benchmarking complex people counting algorithms, especially without using k-fold cross-validation. The differences in performance across splits can be quite striking. For example, as seen in Table 4 and Figure 7, the span between the maximum and the minimum MAE for the UNet semnasnet 75 variant at R3 resolution is over 20. For the UNet semnasnet 100 variant at R3 resolution, the lower MAE bound is on par with the leading approaches whose accuracy was reported for the VisDrone benchmark. However, the upper MAE bound of this design would be considered a model that fares considerably worse. Overall, the lightweight encoder networks seem to perform better in terms of the best average MAE. We believe that this phenomenon stems from the fact that they are less susceptible to overfitting, but this effect can still be observed in all tested architectures.

In [64], the authors discussed the risks and challenges associated with the use of UAVs, including the safety and privacy risks. In the crowd-counting activity, a critical security issue for humans is the potential loss of privacy. For this reason, a continuous software examination is recommended. However, images taken by UAVs at altitudes comparable to the dataset prevent any extraction of biometrical features or attempts of personal identification.
6. Conclusions

In this article, an exploratory study of machine learning algorithms for the crowd counting task with a specific focus on the capability for processing the input image stream on board of a UAV was provided. We showed that using the Xavier NX platform provides sufficient computational power to deal with this task with an accuracy that is close to the current state-of-the-art solutions. Moreover, we showed that the input image resolution is an important factor affecting the final prediction accuracy. This is certainly an issue that one should consider before deciding on the use of a more complex neural network model. The aforementioned factors further emphasise the need for on-board image processing, as the streaming of relatively high-resolution video for processing in the ground base station might not be a tractable solution in many real-world applications.

By using a more in-depth evaluation with multiple folds rather than the typical, suggested training and validation dataset split of the benchmark dataset, we showed that the widely used benchmarks for UAV-based person counting might be too small (in terms of the number of images) for the task as it is complex. This is hinted at by the significant standard deviation of mean average error achieved across multiple folds. Another factor supporting this hypothesis is the general worse performance of the more complex (in terms of the number of parameters) neural network models, which might indicate overfitting.

In the future, we plan on using a larger variant of VisDrone – the DroneCrowd dataset [41], along with pretraining using synthetically generated data. This approach may mitigate, at least to some extent, the problem of overfitting. Another promising approach is to directly utilise the temporal information from more frames, which can be achieved using 3D convolutions. Synthetic data pretraining becomes even more critical for this case, as 3D neural networks require more data to train. This would also address an important issue in crowd counting based on UAV imagery—the need for tedious data labelling, as, in a controlled, simulated environment, the labels can be generated automatically.
Author Contributions: Conceptualization, B.P., M.P. and M.K.; methodology, B.P., D.P., M.P. and M.K.; software, B.P., D.P. and M.P.; validation, B.P., D.P. M.P. and M.K.; formal analysis, B.P., D.P. and M.K.; investigation, B.P., D.P. M.P. and M.K.; resources, D.P. and M.K.; data curation, B.P., D.P. and M.P.; writing—original draft preparation, B.P., M.P. and M.K.; writing—review and editing, B.P., D.P., M.P. and M.K.; visualization, B.P. and M.P.; supervision, D.P. and M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Polish Ministry of Science and Higher Education (Grant No. 0214/SBAD/0233).

Conflicts of Interest: The authors declare no conflict of interest.

References
1. Sindagi, V.A.; Patel, V.M. A survey of recent advances in cnn-based single image crowd counting and density estimation. Pattern Recognit. Lett. 2018, 107, 3–16. [CrossRef]
2. Gao, G.; Gao, J.; Liu, Q.; Wang, Q.; Wang, Y. Cnn-based density estimation and crowd counting: A survey. arXiv 2020, arXiv:2003.12783.
3. Ilyas, N.; Shahzad, A.; Kim, K. Convolutional-neural network-based image crowd counting: Review, categorization, analysis, and performance evaluation. Sensors 2020, 20, 43. [CrossRef] [PubMed]
4. Perko, R.; Klopschitz, M.; Almer, A.; Roth, P.M. Critical Aspects of Person Counting and Density Estimation. J. Imaging 2021, 7, 21. [CrossRef]
5. Li, B.; Huang, H.; Zhang, A.; Liu, P.; Liu, C. Approaches on crowd counting and density estimation: A review. Pattern Anal. Appl. 2021, 24, 853–874. [CrossRef]
6. Shao, J.; Kang, K.; Change Loy, C.; Wang, X. Deeply learned attributes for crowded scene understanding. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4657–4666.
7. Yi, S.; Li, H.; Wang, X. Understanding pedestrian behaviors from stationary crowd groups. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3488–3496.
8. Marsden, M.; McGuinness, K.; Little, S.; O’Connor, N.E. Resnetcrowd: A residual deep learning architecture for crowd counting, violent behaviour detection and crowd density level classification. In Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–7.
9. McPhail, C.; McCarthy, J. Who counts and how: Estimating the size of protests. Contexts 2004, 3, 12–18. [CrossRef]
10. Jiao, L.; Zhao, J. A survey on the new generation of deep learning in image processing. IEEE Access 2019, 7, 172231–172263. [CrossRef]
11. Thompson, N.C.; Greenewald, K.; Lee, K.; Manso, G.F. The computational limits of deep learning. arXiv 2020, arXiv:2007.05558.
12. Li, M.; Zhang, Z.; Huang, K.; Tan, T. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In Proceedings of the 2008 19th International Conference on Pattern Recognition, Tampa, FL, USA, 8–11 December 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 1–4.
13. Sim, C.H.; Rajmadhan, E.; Ranganath, S. Using color bin images for crowd detections. In Proceedings of the 2008 15th IEEE International Conference on Image Processing, San Diego, CA, USA, 12–15 October 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 1468–1471.
14. Subburaman, V.B.; Descamps, A.; Carincotte, C. Counting people in the crowd using a generic head detector. In Proceedings of the 2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance, Beijing, China, 18–21 September 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 470–475.
15. Topkaya, I.S.; Erdogan, H.; Porikli, F. Counting people by clustering person detector outputs. In Proceedings of the 2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Seoul, Korea, 26–29 August 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 313–318.
16. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv 2020, arXiv:2004.10934.
17. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, Seattle, WA, USA, 14–19 June 2020; pp. 10781–10790.
18. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 213–229.
19. Chan, A.B.; Vasconcelos, N. Counting people with low-level features and Bayesian regression. IEEE Trans. Image Process. 2011, 21, 2160–2177. [CrossRef]
20. Chen, K.; Loy, C.C.; Gong, S.; Xiang, T. Feature mining for localised crowd counting. In Proceedings of the British Machine Vision Conference, 2012, Surrey, UK, 3–7 September 2012; Volume 1, p. 3.
21. Lempitsky, V.; Zisserman, A. Learning to count objects in images. Adv. Neural Inf. Process. Syst. 2010, 23, 1324–1332.
47. Franklin, D.; Hariharapura, S.S.; Todd, S. Bringing Cloud-Native Agility to Edge AI Devices with the NVIDIA Jetson Xavier NX Developer Kit. 2020. Available online: https://developer.nvidia.com/blog/bringing-cloud-native-agility-to-edge-ai-with-jetson-xavier-nx/ (accessed on 11 August 2021).

48. Gorbachev, Y.; Fedorov, M.; Slavutin, I.; Tugarev, A.; Fatekhov, M.; Tarkan, Y. OpenVINO deep learning workbench: Comprehensive analysis and tuning of neural networks inference. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea, 27–28 October 2019.

49. Libutti, L.A.; Igal, F.D.; Pinuel, L.; De Giusti, L.; Naiouf, M. Benchmarking performance and power of USB accelerators for inference with MLPerf. In Proceedings of the 2nd Workshop on Accelerated Machine Learning (AccML), Valencia, Spain, 31 May 2020.

50. Zhou, Z.; Siddiquee, M.M.R.; Tajbakhsh, N.; Liang, J. U-Net++: A Nested U-Net Architecture for Medical Image Segmentation. arXiv 2018, arXiv:1807.10165.

51. Chen, L.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. arXiv 2018, arXiv:1802.02611.

52. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. arXiv 2015, arXiv:1512.03385.

53. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv 2019, arXiv:1905.11946.

54. Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Le, Q.V. MnasNet: Platform-Aware Neural Architecture Search for Mobile. arXiv 2018, arXiv:1807.11626.

55. Tan, M.; Le, Q.V. MixConv: Mixed Depthwise Convolutional Kernels. arXiv 2019, arXiv:1907.09595.

56. Yakubovskiy, P. Segmentation Models Pytorch. 2020. Available online: https://github.com/qubvel/segmentation_models.pytorch (accessed on 13 March 2022).

57. Kroeger, T.; Timofte, R.; Dai, D.; Van Gool, L. Fast optical flow using dense inverse search. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 471–488.

58. Hur, J.; Roth, S. Optical flow estimation in the deep learning age. In Modelling Human Motion; Springer: Berlin/Heidelberg, Germany, 2020; pp. 119–140.

59. Buslaev, A.V.; Parinov, A.; Khvedchenya, E.; Iglovikov, V.I.; Kalinin, A.A. Albumentations: Fast and flexible image augmentations. arXiv 2018, arXiv:1809.06839.

60. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J. Big Data 2019, 6, 60. [CrossRef]

61. Bai, J.; Lu, F.; Zhang, K. ONNX: Open Neural Network Exchange. 2019. Available online: https://github.com/onnx/onnx (accessed on 13 March 2022).

62. Developers, O.R. ONNX Runtime. Version: 1.10.0. 2021. Available online: https://onnxruntime.ai/ (accessed on 13 March 2022).

63. Wang, Q.; Gao, J.; Lin, W.; Li, X. NWPU-Crowd: A Large-Scale Benchmark for Crowd Counting. arXiv 2020, arXiv:2001.03360.

64. Iqbal, S. A Study on UAV Operating System Security and Future Research Challenges. In Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), Online, 27–30 January 2021; pp. 759–765. [CrossRef]