Analysis of Computer Vision Application in The Context of SAR Missions

João Custódio de Faria Filho
Instituto Tecnológico de Aeronáutica, São José dos Campos/SP - Brasil

***Abstract*— This work includes a general feasibility and performance analysis of some image classification models for Search And Rescue (SAR) operations. Since SAR missions are typically high-wear and high-risk situations for both the victims and the SAR crew, some teams around the globe have been seeking to use technology to speed up rescues and reduce damages. The Croatian Mountain Rescue Service (CMRS) and other SAR teams have been using Unmanned Aerial Systems (UAS) to obtain bird's eye captures of the search area, contrasting with typical low-altitude manned flies. This paper uses the HERIDAL dataset images to train, validate, and test models. We used two different neural network architectures and eight different training parameters. Accuracy above 98% was achieved, but it doesn't necessarily mean that the models are appropriate for real-life use, so several considerations were made.**

***Keywords*— Search and Rescue (SAR) Missions. Computer Vision. Performance Analysis.**

I. Introduction

Search and rescue (SAR) missions are defined by the Canadian Forces and United States Defense Department as the use of available resources, including machinery, personnel, and other facilities, to search for people or property that are, or are feared to be, in distress or imminent danger, and rescue them to a safe place, besides providing their medical or other needs [1], [2].

Considering the scope of the given definition, SAR missions may need to be performed in different contexts and peculiarities. Commonly these missions have a large search area and can last for days. Furthermore, the hostility of the search environment makes it dangerous for the committed team and urgent for the victims. Therefore, the use of unmanned aircraft systems, UAS, or simply drones, equipped with high-resolution cameras, is a direct solution to conducting an aerial search, reducing the risks and costs of the mission.

In fact, some SAR agencies have been using drones to search for people. According to [3], from 2017 to 2020, at least 59 individuals were rescued by drones from life-threatening situations in 18 different incidents around the world. In Croatia, the use of UASs is largely carried out by the Croatian Mountain Rescue Service, CMRS, which has been using this tool for over the last eight years, at least. It was initially restricted to terrain reconnaissance and brief searches for areas of interest. However, they soon began to be used in all phases of SAR missions and in different types of terrain.

The success of using drones for SAR missions does not depend only on well-trained pilots. In fact, the analysis of captured aerial images also requires specialized teams, to look for people or relevant evidence.

J.C. Faria F., joaocfariaf@gmail.com

In the case of the CMRS, each searching UAS captures images every 4 seconds of flight, which means more than 2000 images per day [3]. Also, each captured image corresponds to a large area, so that a person present would occupy less than 0.1% of the photographed area. In addition, the victims are often camouflaged, due to the variation of shadows, colors, reliefs, and elements [3], [4].

The CMRS makes use of its own cloud computing system to make the captured images available for analysis by specialized personnel, taking from 5 to 45 seconds to verify each image individually, according to [3]. Unfortunately, this may not be enough. In 2020, there was at least one case of an unsuccessful CMRS mission even after 10 days of search, using 6 aircraft and 340 people [3].

In the scope of image analysis, the large number of captures and the inherent meticulousness make this task prone to human failure due to fatigue and attention loss. Therefore, the use of computer vision for the purpose of SAR missions might be promising.

The main goals of this paper are to train image classification models in the context of SAR missions, make a critical review of the considered performance measures and visualization, and suggest new ones.

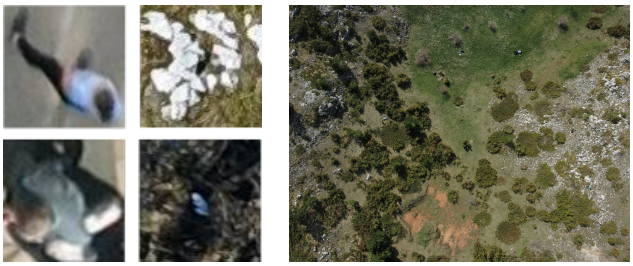
The next sections are described as follows. Section II we provide a brief of works with similar objectives. Section III describes HERIDAL, which is the data set used. Section IV describes the considerations taken and the methodology used to prepare and test the models to achieve the results discussed in Section V. Section VI describes the final considerations about this work.

II. Related Work

In fact, in recent years, there were made some research work publications about the use of computer vision in the context of SAR missions, such as [3], [4], [5], [6], which are focused in detecting people in the given images. In the case of the first two listed works, the scenario is the wilderness while the last two are focused on SAR missions in the sea.

There are some works in this context that use infrared cameras in an attempt to detect people by the emission of this frequency which is related to the characteristic heat of mammals [6]. Other works sought the use of multispectral cameras, seeking to combine particular emissions in different bands [5].

According to [3], in warmer scenarios, such as tropical forests or the one that operates the CMRS in the summer, it's not worth using infrared images or multispectral cameras, since the environment usually has quite similar radiation emissions when compared to people. Thus, it lasts to process common images, obtained by visible radiation.



(a) Patches. (b) UAS capture.

Fig. 1: Samples of images from HERIDAL.

Despite having been listed here some scientific works involving computer vision and SAR missions, there are not quite many works with operational application analysis. In addition, many works lack images of real missions, due to the difficulty of obtaining them for research as well as standardizing them for this purpose. To overcome the lack of real data, as well as the inherent ethical questions, some works make use of images of situations simulated by actors, as in [3], and [4], which uses the images from [7].

Nevertheless, the use of computer vision can be promising to help in these missions. Not being susceptible to physical or emotional fatigue and being able to have improved processing speed with the use of specific hardware.

The present work focuses on the analysis of the application of computer vision for the detection of people in high-resolution aerial images obtained by UASs, in the visible spectrum.

III. Dataset Description

The HERIDAL dataset is used in this work and was first published by [7]. It was created aiming to tackle the difficulty of finding specific image datasets for SAR purposes. This dataset contains bird's eye captures of simulations of SAR missions and has already been used by [3] and [4].

In the HERIDAL dataset there are originally three folders: "*trainImages*"; "*testImages*"; and "*patches*". In the first and second folders there are respectively $3,131$ and 205 "*.jpg*" files each one related to a "*.xml*" file in a separated folder named "*labels*". In the third folder, there are two directories: "*negative*" and "*positive*" containing cuts of the images in "*trainImages*" and "*testImages*", each one of size 81 x 81 *pixel*. In "*positive*" directory, there are $29,050$ images with a person on it. In "*negative*" directory, there are $39,700$ images with no person on them. Examples of files in the "*patches*" folder are illustrated in Fig. 1a.

The images in HERIDAL were all captured using a drone of the model *DJI Phantom 3*, flying at a height of 50 m and equipped with a camera of resolution of 4000 x 3000 pixel.

These captures were taken from simulated scenes which were made considering statistical data involving clothing, positioning, and pose in which children, the elderly, people with mental illness, hunters, mountaineers, and climbers are usually found in the area covered by CMRS. All the 36 simulations were done in open areas in the wilderness. With the guidance of experts, only 4 of them were chosen to be part of the HERIDAL dataset.

It is worth noting that for the purpose of optimizing the dataset for training, in each aerial capture, there is at least one person to be found.

In a real scenario of SAR missions, most of the images will probably have no person in it. For the purpose of optimizing the dataset for training models, in each aerial capture in HERIDAL, there is at least one person to be found, as shown in Fig. 1b.

IV. Methodology

In this section we describe the procedure and considerations taken to explore the images from HERIDAL in terms of training, validating, and testing classification models and also other aspects inherently to the considered application.

A. Image Classification

In this subsection, we explain the peculiarities of the work of building the classification models, which were trained, validated, and tested using the images in the folder "*patches*" of the HERIDAL data set.

1) Network Architectures: The Architecture #1 was already used by [8] for another classification task. In that context, the network is trained for image classification solving an example problem of differentiating images, 150 x 150 pixels, as containing cats or dogs. For the context of this paper, the input of the architecture was changed to images with resolution 81 x 81 pixels.

Fig. 2: CNN architecture from [8].

This network architecture starts with three Convolutional layers, all having *ReLU* as its activation function. Also, each one of these layers is followed by a Max Pooling Layer. This set of layers is then followed by a fully connected layer, which also has *ReLU* as its activation function, and which, in turn, is followed by the fully connected output layer, with a single neuron and a sigmoid as its activation function. Fig. 2 illustrates this architecture.

As illustrated, the first and second convolutional layers have 32 filters each, while the third has 64, the same for the hidden fully connected layer. The max pooling layers have kernels with dimensions of 2 x 2 pixels, and 1 pixel of stride.

The Architecture #2 was implemented following the example proposed by [4]. It starts with a set of two convolutional layers each one followed by a Max Pooling Layer. This set of layers is then followed by two other convolutional layers and a fully connected layer, which is connected to the fully connected output layer, with a single neuron. All the convolutional layers and the hidden fully connected layer have *ReLu* as activation function. The output layer has a sigmoid as its activation function. Fig. 3 illustrates this architecture.

As illustrated, the first and second convolutional layers have 32 filters each, while the third and fourth have 64 filters, as the hidden fully connected layer. The max pooling layers have kernels with dimensions of 3 x 3 pixels, and 3 pixels of stride.

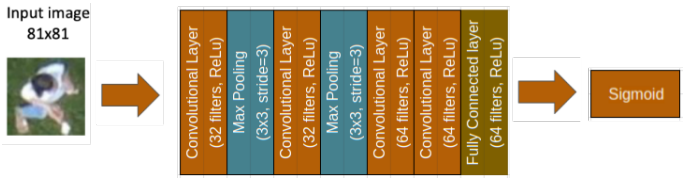


Fig. 3: CNN architecture from [4].

2) Model Fitting and Evaluating: The set of patches was divided in three subsets: *train*, *validation* and *test*, each one with 60%, 20%, 20%, respectively. The split of patches between each one of these subsets was made maintaining the original proportion between "negative" and "positive".

In terms of the learning task, each architecture is explored using two learning algorithms. The first one is the Root Mean Square Propagation Algorithm with $\rho = 0.9$ and a learning rate of 0.001 as used by [8]. The second one is the Stochastic Gradient Descent Algorithm with the same learning rate, which was also used by [4] to achieve an accuracy of 99.21% with Architecture # 2.

Given these two learning algorithms, we also need to define the batch size used to perform the learning task. In this paper, we test some different values of batch size, which due to hardware limitations was $m' \in \{16, 32, 128, 256\}$.

After that, we need to define the number of steps per epoch and the number of epochs. To guarantee that the training process uses the entire training set, the number of steps is $\lfloor m/m' \rfloor$, with m being the number of examples. The number of epochs was fixed at 70, since it does not cause overfitting and also provides a suitable training time. Comparing these training variations, one best model is chosen for each architecture taking the accuracy as the parameter. In this case, since the classes are almost balanced, accuracy is a reasonable performance measure for the classification task.

3) Real life considerations: In HERIDAL there are approximately 3 positives for each 4 negatives patches, which would mean that looking for a lost person in any area corresponding to a patch will be successful approximately 42% of the time. However, as mentioned by [3], in real SAR missions, the number of negative patches is usually much greater than that of positive ones. That is why the search part is also hard in a SAR mission, and why works like this one can be really helpful.

Considering the data set limitations and the fact that the expected proportion of "positives"and "negatives"can be very different depending on the causes of the mission, in this paper we explore different proportions for testing the models. Thus, there are extra sets with the same negative patches but with a number of positive ones 10^{γ} times smaller. The sets are named "*test*_γ", with $\gamma \in \{0.5, 1.0, 1.5, 2.0\}$.

In these contexts, accuracy is no more an adequate performance measure, since a model that classifies all inputs as negative would also achieve a high rate. Thus, the total of true negatives is not a very relevant measure and we might consider a trade-off between precision and recall.

A high recall rate would mean that almost no victim would be left behind by the algorithm. However, this could result in a model that indicates almost all inputs as positive, which signifies time wasted in false alarms which diminishes the success chances of the mission wasting resources and prolonging the time that the victims are exposed to danger.

A high precision rate indicates that if the algorithm had classified an input as positive, it is probably true. This means less resource wasting in false alarms, but can also mean that some victims would be left behind in the mission, as the model will neglect them.

F_β is a performance measure that is a trade-off between precision and recall weighted by the parameter β,

$$F_\beta := (1 + \beta^2) \times \frac{\text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}. \quad (1)$$

The ideal value of β is not easily determinable and should depend on the nature of the application and operation particularities that are beyond the scope of this work.

If the model is, for example, supposed to be embedded in a UAS that can drop a medical kit if the region is classified as positive, then the priority should be precision and β should have a small value, since false positives would imply in great losses for the mission. However, if the model is supposed to recommend areas for an operator at the base and the search is for only one missing person, then, a greater value of β should be considered, as the operator action limits the losses. Thus, in this part of this work, we explore different values of β using a graphical approach.

B. Sliding Window

After training and testing different neural network models to classify patches and choosing the best one, we need to consider a methodology to extract the patches from the original images if we want to consider their use in a real situation. In the context of this work, the strategy chosen was the sliding window method which is used in the images from in the folder "*trainImages*" and "*testImages*".

The Sliding Window approach consists of classifying pieces of the original image individually using a model. The name *window* comes from the fact that at each classification step, only a part of the original image is considered, while the rest is ignored, just like if the image is seen through a window. This window has a fixed dimension, which in this case is 81 x 81 pixel as it is the size of the input of the architectures described in IV-A.1. As the window slides, other parts of the image are seen and classified by a classification model. The way that the window slides is determined by the strides parameters in both of the image dimensions. In this case, the stride parameters are 80 pixels and 75 pixels in the horizontal and vertical directions, as it takes 50 steps to slide horizontally and 40 steps to slide vertically, considering the dimension of the images, as described in the section III. Thus, for each image, there will be necessary 2000 steps.

In terms of the applying operations at separated regions in the image, the Sliding Window is quite similar to the convolution operation. However, in this case, the operation does not use padding and the local operation is a classification instead of a matrix multiplication. As one or more patches of the image are classified as positive, the localization is also informed, since the coordinates of the window are given.

In this part of the work, the evaluation consists only in terms of the time taken to classify all the regions proposed by the Sliding Window approach. Also, since all the variations explored in IV-A.2 do not interfere with the complexity of each classification operation, except for the architecture of

the network, the comparison relies only on the differences of time considering the two .

V. Results and Discussion

All algorithms used in this paper run on a notebook equipped with an Intel® Core™ i7-8550U 1.8 GHz with Turbo Boost up to 4.0 GHz, 8 GB DDR4 memory, and NVIDIA® GeForce® MX130 with 2GB VRAM, which is a hardware configuration quite less powerful than the used by [4]. In that case, the available hardware was a workstation equipped with an Intel® Xeon® E5-2640v4 of 3.40 GHz, 4 x 16 GB DDR4 memory, and multi-GPU 4 x NVIDIA® GeForce® GTX 1080Ti Turbo with 11 GB memory.

A. Image Classification

As detailed in IV-A.2, both the architectures described IV-A.1 were tested considering the accuracy after training the model with different parameters. Results referring to the tests with both architectures are presented in tables I and II. The column "*Time*" refers to the time spent to train the correspondent model using the correspondent training parameters.

TABLE I:
Tests from IV-A using Architecture $\#1$.

Optimizer	Batch Size	Time (hh:mm:ss)	Accuracy (%)
SGD	256	00:56:39	87.14
	128	00:57:16	90.65
	32	01:06:21	96.57
	16	01:18:42	97.81
RMSProp	256	00:57:27	98.69
	128	00:58:07	98.25
	32	01:08:05	95.38
	16	01:21:49	90.78

TABLE II:
Tests from IV-A using Architecture $\#2$.

Optimizer	Batch Size	Time (hh:mm:ss)	Accuracy (%)
SGD	256	00:43:45	88.63
	128	00:44:38	87.52
	32	00:53:19	97.50
	16	01:04:08	98.36
RMSProp	256	00:43:47	98.86
	128	00:44:23	98.71
	32	00:53:09	96.33
	16	01:05:51	85.74

Considering an operational application, as the scenario of the mission is much variable, the team should consider, if possible, training the pre-trained model with some inputs of the current mission or that of a similar mission, so that the model becomes more adjusted to the current purpose. Thus, the time spent to train the model could be a relevant parameter to consider. Analyzing each architecture individually, the time spent is very influenced by the batch size. This makes sense since a smaller batch size means more steps per epoch as $\lfloor m/m' \rfloor$ gets a higher value as m' gets a lower one. With a fixed number of epochs, the number of total steps gets higher and, also, the time to train the model, since more gradient descent steps mean more mathematical operations to estimate the gradient and update the parameters by back-propagating, thus more time spent. Furthermore, a smaller batch could mean a higher cost of time even without meaning a higher computational cost, since it reduces the possibilities of parallel processing. The choice of the optimizer does not seem to impact much of the time spent, this is true because the cost difference between them is caused by a few multiplications.

In terms of accuracy, considering the use of SGD for both architectures, the number of steps seems to be quite relevant. Thus, the accuracy tends to be higher as the batch is lower. This makes sense since SGD algorithm runs with a learning rate of 0.001 which means that each parameters' update is a tiny step in the direction of decreasing the estimated gradient. As the batch gets smaller the statistical relevance of the m' considered elements also decreases and becomes more probable to get a bad estimation for the gradient and to struggle to converge. However, the small learning rate makes each step have reduced impacts individually, preventing this problem from happening.

As the RMSProp consists of a strategy of "incrementing"the learning rate by using the historical data of the gradient so that the algorithm converges more rapidly in bowl-shaped regions, it is more sensitive to the possible abrupt variations in the estimation of gradient considering small values of m'. Thus, it might struggle to converge to a higher accuracy with a small batch size, even spending many steps. However, considering greater batch sizes, this optimizer makes it possible to achieve high accuracy values with a relatively low cost of time. The choice of a good batch size should consider the hardware limitations and the equilibrium between the statistical relevance of each batch and the time needed to achieve high accuracy. For each situation, there should be at least an "optimum interval"for batch size values, so that smaller or bigger sizes would mean, respectively, much more time to train or much less accuracy, due to the reduced number of steps per epoch. In this case, the interval from 128 to 256 appears to be a good choice for an "optimum region"for batch sizes, since there is not much difference in terms of accuracy or time, however. Also, it's important to point out that these results are subject to fluctuations due to the randomization of the choice of batches and depend on the characteristics of the dataset and the hardware. In this case, larger batch sizes could not be tested due to memory limitations.

It's notable that thanks to the RMSProp algorithm, the both best models in terms of accuracy had taken a time to train quite close to the smallest time taken for each architecture. Also, in terms of accuracy achieved after training, both architectures are basically equivalent, the values measured will probably fluctuate due to the random choices of batches.

However, it is noticeable that in terms of time to train, the Architecture $\#2$ is clearly faster than the Architecture $\#1$. In fact, considering tables I and II, and comparing the time to train between cases with the same optimizer and batch size, on average, the time to train of Architecture #2 is $(21.30 \pm 2.01)\%$ smaller. This difference in time can be due to the reduction of dimensionality that is more accentuated in the pooling layers of Architecture $\#2$, since the strides are bigger.

B. Real Life Considerations

Since the ideal value of β to consider in F_β is not trivially deduced, in this section, we propose a graphical visualization of F_β as a function of β^2 (see 1), so we can visualize the variation of the function for different choices of β^2.

Since $0 \leq \beta^2 < \infty$, we display the graphics in two separate intervals and display them side-by-side. On the left side, we

TABLE III:
A_β for each domain, γ and architecture.

Architecture	β^2	$\gamma = 0.5$	$\gamma = 1.0$	$\gamma = 1.5$	$\gamma = 2.0$
#1	$[0, 1[$	0.973	0.934	0.933	0.602
	$[1, \infty[$	0.977	0.961	0.960	0.774
#2	$[0, 1[$	0.969	0.915	0.914	0.535
	$[1, \infty[$	0.978	0.953	0.949	0.726

plot the graph of F_β vs β^2, considering $\beta^2 \in [0, 1[$. On the right side, we plot the graph of F_β *vs* $1/\beta^2$, considering $1/\beta^2 \in]0, 1]$ with the *x-axis* inverted so that the combination of the plots looks continuous.

The segregation of the β^2 values in the given intervals has also meaning considering the signification of β^2 being in one of these. If $\beta^2 < 1$, then the value of F_β depends more on precision than on recall, which is equivalent to saying that in the operations at which the ideal value of β^2 is smaller than 1, precision is more relevant than recall. Analogously, for $\beta^2 > 1$, the value of F_β depends more on recall than on precision. In terms of limits, from 1:

$$\begin{cases} \lim_{\beta^2 \to 0} F_\beta = \text{Precision} \\ \lim_{\beta^2 \to \infty} F_\beta = \text{Recall} \end{cases}.$$

To provide a single value to evaluate the performance of a model in terms of F_β in a given interval of variation of β^2, we also calculated the area under the curve of F_β as a function of β^2 or $1/\beta^2$ (it might be helpful to consider A_β as the area under the curve of F_β divided by the length of the considered interval of β^2 or $1/\beta^2$, but as both intervals considered here have length 1, we don't need to care about it). This score is referred to here as A_β and is estimated by the sum of the areas of the trapezoids defined in the graph by taking 200 equidistant points at it $]0, 1[$. For each plot, $0 \leq A_\beta \leq 1$.

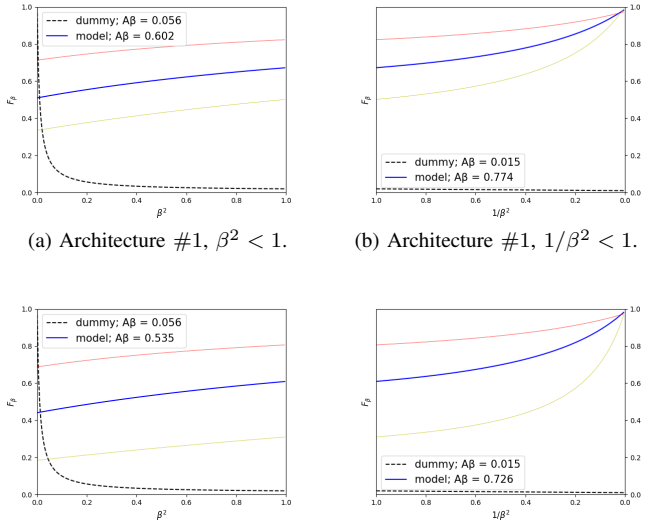
The closer A_β is to 1, the better the model. If A_β is high-valued, then the model achieved a high-valued F_β for all β^2 in the given interval. However, if the value of A_β is not high enough for the considered task, we should investigate the reasons for it since F_β could have remained almost constant in a low value, or it can have assumed high values in a sub-interval and very low values in another.

In these terms, A_β is similar to F_β which also does not have a defined meaning when it gets low values, since it can mean low precision, low recall, or even both. On the other hand, a high value of F_β means high precision and recall. Thus, a high value of A_β means high enough values of F_β, then also precision and recall, in the considered interval of variation of β^2.

The table III presents the calculated values of A_β for each architecture in the variety of domains and values of γ. The Fig. 4 shows the proposed visualization of F_β for $\gamma = 2.0$.

In Fig. 4, the dashed black line stands for a dummy model that classifies the inputs as positive randomly with probability equal to the relative frequency of positives in the data set. The thin yellow and red lines represent the trained model with thresholds equal to 0.005 and 0.995, respectively. The blue line represents the trained model with a threshold equal to 0.5 and is the one that is considered by default.

Comparing the results of F_β achieved by the best models of the two network architectures with that of the dummy model, the latter performs much worse than both of the best models



(a) Architecture #1, $\beta^2 < 1$. (b) Architecture #1, $1/\beta^2 < 1$.

(c) Architecture #2, $\beta^2 < 1$. (d) Architecture #2, $1/\beta^2 < 1$.

Fig. 4: Plots of F_β with $\gamma = 2.0$ for the most accurate models.

in all the scenarios proposed except for $(1/\beta^2 \to 0$, which implies $F_\beta \to Recall = 1$, for the dummy model. However, in operational terms $(1/\beta^2 \to 0$ makes no sense at all since it would mean that a model that gives no false negatives but lots of false positives, which costs time and human resources in a real SAR mission, is better than another model that gave only one false negative. In terms of A_β, the dummy model performs better in the interval $(1/\beta^2) < 1$, since it means more relevance to the recall, and also performs better in scenarios with more positive inputs, as it implies higher precision. Even though, in average, considering all the plots, the A_β for the best models of both architectures is around 9 times bigger than that of the dummy model, which is a first, but not too meaningful, sanity check for the use of these models.

Both architectures achieved $A_\beta > 0.90$ except for $\gamma = 2.0$. This indicates that for any optimal choice of β the performance of the algorithm should be, at least, around 0.90 as measured by F_β, which indicates some robustness of the models in all but one of the tested scenarios.

As we got $\gamma = 2.0$, the A_β value drops significantly, as can be seen in Figs. 4a, 4b, 4c and 4d. This indicates that both models will probably perform poorly with $\gamma > 2.0$, for most choices of β, except for $1/\beta^2 < 0.1$, which is probably a bad choice, since $F_\beta \geq 0.9$ can be achieved with $Precision = 0.45$, for example.

According to [3], more than 2,000 aerial images are taken on a typical day in a search mission. Each image of HERIDAL has a resolution of 4,000 x 3,000 pixels, which means a minimum of 1,900 patches with resolution 81 x 81 pixels to cover up all the images completely.

Considering that all the patches classified as positive by the model are inspected by an operator that checks the given classification in only 2 s, the estimated time spent with false positives is given by $t = \text{FPR} \times 2{,}000 \times 1{,}900 \times 2s$, with FPR denoting the false positive rate. We express FPR in terms of precision and γ as:

$$\text{FPR} = \frac{1}{10^\gamma + 1} \times \left(\frac{1}{\text{Precision}} - 1 \right). \quad (2)$$

TABLE IV:
Time to verify one image as in IV-B.

Architecture	Average Time (s)	STD (s)
#1	101.07	1.21
#2	100.69	1.32

Thus, with $\gamma = 2.0$ and $Precision = 0.45$ the working time wasted with false positives will be around 26 hours per day. As A_β diminishes as γ increases, we can expect that both the models will not be good enough for $\gamma \geq 2.0$.

In real situations, the corresponding value of γ is probably greater than 2.0. In the context of the CMRS, there are $1{,}900\times$ $2{,}000$ patches per day, with $\gamma = 2.0$, this means that if the total area corresponds to 1% of the pictures, there will be 380 missing people, which is much more than the maximum occupancy of most airplanes.

Also, airplane accidents are not a common cause for SAR missions in the context of CMRS. so both models appear not to be a good choice in this context. However, this result should be evaluated again with a larger test set, with more positive and negative examples, so that the conclusions could have more statistical relevance.

C. Sliding Window

As described in IV-B, in this section we take the most accurate model of each architecture tested in the previous section and use it in a sliding window approach in the images captured by the UAS available in the HERIDAL dataset. The objective of this stage is to measure the time taken to evaluate an image using a sliding window and the classification models trained in the previous section in order to compare with the time taken by a human operator and that of [4]. Table IV gives the results achieved in this stage.

Considering the results in table IV, none of the architectures performed significantly better than the other one. In fact, considering the error margin of the measure, given by the standard deviation, both architectures have similar times to analyze a picture completely. However, the average time taken for visual inspection of one image is 43.68 s, and that of the proposed model by [4] is less than 15 s. Thus, this approach is approximately 2.3 times slower than a human operator and 6.7 times slower than the model proposed by [4].

However, this difference is not sufficient to discard this approach for operations. In terms of operational costs, the time taken is relevant but also is the price of the hardware used, since this time taken can be reduced using a better hardware unit or multiple units with cheap hardware working in parallel. Thus, for a better comparison with the result achieved by [4] we should consider testing both approaches in similar hardware.

For operational purposes, the approach needs to be tested with the available devices and considering the specific application that is been proposed. For example, in the case of the work of the CMRS the requirements of cost of time and memory are probably easier than the ones that would be needed if the model was supposed to be embedded.

VI. Conclusion

In terms of accuracy, the models achieved similar results compared to [4] and are quite equivalent, even with all the variations considered. However, the models based on [4], needed an average training time $(21.30 \pm 2.01)\%$ less than that based on [8], which is a significant advantage if more numerous training data sets are considered. In terms of optimizer, significantly less time was taken to achieve high accuracy rates using RMSProp as compared to using SGD.

Considering the evaluation of $4{,}000$ x $3{,}000$ pixel images, the time taken was about 2.3 times greater than that by a human operator and about 6.7 times greater than that by the approach of [4]. It is worth noting the difference in the hardware available for this work and that one for [4]. In real life, the analysis should be done using the available hardware for a SAR team. Thus, future work can be done in this direction or aiming to determine minimum hardware requests to achieve operational objectives.

The analysis carried out using F_β concluded that the models are not suitable for real situations. However, this affirmative needs to be tested with greater statistical relevance, since the measure done is very susceptible to noise since there are only tens of positive inputs in the entire test set.

Even with a better test set, we would still need to find a good β^2 value or interval, which can be analyzed with A_β. To determine β^2, the victims, the accident, the equipment, and the personnel, must be considered. Suppose a data set like HERIDAL, built through sufficiently realistic simulations with 100 actors of an accident in conditions that at every hour the victims' chances of survival reduce by 1%. Considering $2{,}000$ pictures per search day and knowing the proportion of positive patches, we can make an estimation of the time spent with false positives and the decay in life expectancy of the victims. The "ideal" β^2 can calculated with the ratio between these expected life/time costs.

In general, works like this can be quite relevant for improving SAR operations. However, for the evaluation of models, definition of metrics, and feasibility studies, more specific studies would be needed with previously calculated statistical data for the considered contexts.

References

[1] C. Forces, "National sar manual," 1998. [Online]. Available: https://web.archive.org/web/20080830015913/http://www.casaraontario.ca/~webmaster1/Manuals/NationalSARmanual_full_english.pdf

[2] U. D. of Defense, "U.s. department of defense directive number 3003.01," 2011. [Online]. Available: https://www.dco.uscg.mil/Portals/9/CG-5R/nsarc/DoDI%203003.01%20DoD%20Support%20to%20Civil%20SAR.pdf

[3] S. Gotovac, D. Zelenika, Ž. Marušić, and D. Božić-Štulić, "Visual-based person detection for search-and-rescue with uas: Humans vs. machine learning algorithm," *Remote Sensing*, vol. 12, no. 20, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/20/3295

[4] M. Kundid Vasić and V. Papić, "Multimodel deep learning for person detection in aerial images," *Electronics*, vol. 9, no. 9, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/9/1459

[5] E. Lygouras, N. Santavas, A. Taitzoglou, K. Tarchanidis, A. Mitropoulos, and A. Gasteratos, "Unsupervised human detection with an embedded vision system on a fully autonomous uav for search and rescue operations," *Sensors*, vol. 19, p. 3542, 08 2019.

[6] A. J. Gallego, A. Pertusa, P. Gil, and R. Fisher, "Detection of bodies in maritime rescue operations using unmanned aerial vehicles with multispectral cameras: Gallego et al." *Journal of Field Robotics*, vol. 36, 12 2018.

[7] D. Božić-Štulić, Ž. Marušić, and S. Gotovac, "Deep learning approach in aerial imagery for supporting land search and rescue missions," *International Journal of Computer Vision*, vol. 127, pp. 1–23, 09 2019.

[8] F. Chollet, "Building powerful image classification models using very little data," 2016. [Online]. Available: blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html