NHL STENDEN
computer vision & data science

Drone Safe Space Landing Detection

NHL Stenden Professorship in Computer Vision & Data Science

Boas Prins, Vlad Veresciaghin

Supervisors: Maya Aghaei Gavari, Jaap van de Loosdrecht

Abstract—This technical paper focuses on creating a failure-safe landing system for BLOVS UAVs, to achieve fully autonomous drones which comply with the European Union Aviation Safety Agency regulations. This research tackled a specific part of a larger project, which aims to find a suitable segmentation model which can detect and avoid ground obstacles in the scenario of an emergency landing performed by an unmanned aerial vehicle. A deep learning approach is used, involving two segmentation architectures, U-Net and U-Net++, supported by different experiments to improve the performance of these models and in the end determine the best performing architecture. From the results, was concluded that using a segmentation approach is a suitable method to apply in this project, although a few limitations must be first settled to test this method in a real-life scenario.

Index Terms—Semantic segmentation, UAV, fail-safe landing, BVLOS, U-Net, U-Net++

1 Introduction

The last few years show a spike in the growth of drone usage thus, the future of the drones industry looks promising as the usage of unmanned aircraft expands, from products delivery to agriculture, building inspections, security, law enforcement, and many more[1]. Yet, unmanned aerial vehicle (UAV) usage has been impeded by strict lawful measures. However, according to the European Union Aviation Safety Agency (EASA) as of 31 Dec 2020, the regulations for civil drones usage were revised in the European Union, thus allowing more freedom for the use of Beyond Visual Line of Sight (BVLOS) UAVs for diverse purposes [2]. Even so, the unmanned aircraft (UA) must comply with strict safety regulations implemented by the Specific Operations Risk Assessment department (SORA) which are correlated with the EASA standards[3]. Hence, BVLOS UA must develop high safety features in case of failure or an emergency landing to extend those boundaries and be able to achieve commercial use at a large scale.

The current issue is that there is no reliable and fully accurate autonomous fail-safe landing system for unmanned aircraft operating beyond the visual line of flight. Thus, the 'Towards the first and Best EU approved Autonomous Security drone for BVLOS flighT ' (The BEAST) project emerged as a collaboration between Saxion University and NHL Stenden University of Applied Sciences, to create a reliable fail-safe landing system for drones. The main goal of this project is to detect and avoid hazardous impediments on the ground which might interfere with a safe landing maneuver performed by the BVLOS, thus achieving a successful touchdown without producing any harm to the environment or the aircraft.

A UAV can be flown in three different scenarios regarding the pilot's line of sight, the flight that occurs at the visual line of sight (VLOS) imply that the pilot maintains continuous independent visual contact with the UA, thus allowing different safety and avoidance maneuvers to be performed within the visual line. Secondly, the extended visual line of sight (EVLOS) requires the remote pilot to be assisted by a trained observer who must always have a good visual of the drone and communicate essential flight information to the pilot. Lastly, the beyond visual line of sight (BVLOS) drones are operated by the pilot without retaining a visual line of sight upon the aircraft at all times, thus the only visual available is the bird's eye view provided by the drone's camera. [4] Hence, the UA is deprived of the hands-on ability of the pilot to detect and avoid obstacles, such a system must be developed to aid the drone along its trajectory. However, this implementation must be light enough to go onboard a UAV. This implementation should also consider computation speed and fast response which are important for this task as each second is crucial when performing a fail-safe landing and object avoidance at the same time, luckily achieving those standards has been facilitated by the evolution of deep learning in recent years. Although the final goal of The BEAST project is to achieve state-of-the-art performance in all these categories, this specific paper is focused on implementing an algorithm to detect and avoid ground obstacles in the scenario of an emergency landing, while minimizing the risks of crashing and inflicting damage. As the other study done on this project used an object detection approach to solve the issue, for this implementation we decided to make use of a segmentation approach, to predict the outliers in a specific area and assess if it is suitable to perform a safe landing maneuver. Thus, from the Aeroscapes dataset [5], the twelve classes were divided in two, as follows. The safe class contains vegetation and roads, while the unsafe class contains everything else, such as constructions, people, obstacles, cars. See Section 3.1.

This research paper will address the implementation of segmentation using two architectures known to yield high results, U-Net [6] and U-Net++ [7], for fail-safe landing in drones operating BVLOS by answering the research question: *How can semantic segmentation be implemented in the failure-safe landing system of Unmanned Aerial Vehicles?*
To help solve the above statement three research questions were derived:

- *How suitable is the segmentation approach in implementing the fail-safe landing system in UAVs?*
- *How does a different segmentation algorithm compare with the baseline U-Net architecture?*
- *Which hyperparameters have the most impact on the performance of the segmentation model?*

- *Vlad Veresciaghin is a Hospitality Management student at the NHL Stenden University of Applied Sciences, E-mail: vlad.veresciaghin@student.nhlstenden.com.*
- *Boas Prins is an Electrical Engineering student at the NHL Stenden University of Applied Sciences, E-mail: boas.prins@student.nhlstenden.com.*
- *Maya Aghaei Gavari is a senior researcher at the NHL Stenden Professorship in Computer Vision & Data Science, E-mail: maya.aghaei.gavari@nhlstenden.com.*
- *Jaap van de Loosdrecht is a professor at the NHL Stenden Professorship in Computer Vision & Data Science, E-mail: jaap.van.de.loosdrecht@nhlstenden.com.*

2 State of the Art

Recent studies have shown different approaches to solving fail-safe landings. In their paper from 2018, Rabah et al. [8], used image processing to solve the problem. After they converted the image received from the drone video footage from RGB to HSV format, they apply a threshold resulting in a binary image. Afterward, morphology transformers are used on the output image to get rid of any noise in it. Finally, to calculate the centroid of the image, 1st order spatial moments around the x-axis, y-axis, and the 0th order central moments of the binary image are calculated. 0th order central moments of the binary image represent the pixels in the image where the binary value is equal to 1.

More recently, Esteban et al. [9], starts with applying the transfer learning technique to another model. This way, one can use the weights of a previously trained model as a starting point, instead of randomly initializing them. This usually gives a better training process since the previous model already learned to extract important features from the samples. Although the data he used was representable, it lacked in size. To solve this problem, the author decided to use data augmentation to improve the performance of the model. Choosing eight augmentation techniques from the Albumentations library [10], resulted in different images to train the model with. In the current state, this study might not be ready for real-life applications yet, but it is the first step towards a fail-safe landing mechanism.

In their paper from 2021, Guérin et al. [11], use semantic segmentation to identify landing zones and Bayesian neural network theory to monitor the predictions of the emergency landing system at runtime. Here the authors aim at building a semantic segmentation model able to determine if a given pixel is part of a busy road or not. To do this, a model called Multi-Scale-Dilation net (MSDnet) proposed in [12], is used. Then, the emergency landing system can use the predicted segmentation to identify a suitable landing spot.

Among other studies on autonomous UAV landing systems, Lee et al. [13], made use of optical flow. The study describes the two modules installed in the fail-safe landing system, the optical flow magnitude map generator, and obstacle analyzer. The optical flow magnitude map generator calculates optical flows between the input images obtained from the camera. An obstacle analyzer does, as the name implies, determine the existence of obstacles. From the optical flow magnitude map, two criteria can be acquired: the magnitude of the optical flow, and the feature point. With these criteria, an obstacle can be determined by the magnitude. If an obstacle exists, the magnitude is larger than normal, and the closer it is, the larger it becomes. In addition, an image that includes an obstacle generates more feature points than a flat image.

3 Materials and Methods

This section will further detail the materials used throughout the project and explain the methods involved in this paper, from the experiments and the fine-tuning of the hyperparameters to the results of the models used.

3.1 Dataset

This project is using a public shared dataset, named Aeroscapes [5]. The aerial dataset comprises 3269 RGB images, captured by a commercial drone from an altitude range of 5 to 50 meters. The pictures have a resolution of 720px by 1280px, are semantically segmented, and divided into 12 classes, from which the respective ground-truth masks are provided [5]. For this project, the classes will be combined in the classes 'safe' and 'unsafe'. The classes amount is not decisive, since the purpose of this project is to identify safe landing spots, and the model does not have to distinguish the difference between a person or a car.

Moreover, the dataset consists of a folder that contains the visualization for each image and another one with the text file names of the training and validation splits for data. To better process the data and feed the model, the images were separated into 3 portions, 60% representing the amount allocated for training the architecture, which makes up 1962 pictures of the total, 653 samples signifying 20%, were assigned for validating the model, while another 20% were reserved for testing the architecture.

Lastly, while doing the research and data exploration, two other datasets were determined to be relevant for this project, the first one being UAVid which contains 300 high-resolution images extracted from 30 video sequences recorded in 4k [12]. The second dataset is the Semantic Drone dataset cumulating a total of 400 images with a resolution of 6000 by 4000px [14]. Those two datasets are fairly different in comparison with Aeroscapes, by having the majority of the pictures taken in densely populated urban areas including busy roads and buildings. Moreover, the image resolution is different in all three of them as well as the segmentation classes. Thus, Aeroscapes being the larger of the three, also includes a higher percentage of pictures containing vegetation and free roads which are the main focus for our model prediction, making this dataset the most suitable one for the given project. However, the Semantic Drone dataset is still being used in the experiments part to measure how well the algorithm performs on out of dataset images.

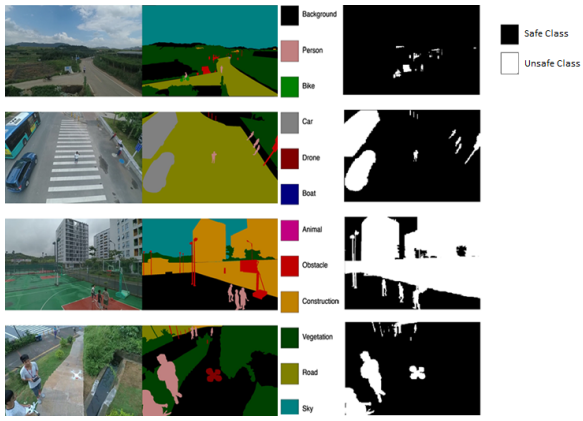

Fig. 1: Aeroscapes Dataset with combined ground truths

3.2 Model architectures

As mentioned earlier, we opt for solving the previously presented research questions using an image segmentation pipeline, and the integrated U-Net architecture [6]. The U-Net model used in this project is composed of 64 start filters, 5 depth layers, and a latent vector of size 1024. The architecture uses a 3x3 convolution, a 2x2 max pooling layer with a kernel size of 3, used for downsampling and stride 2, and a final layer of a 1x1 convolution used to assign each of the 64 component vectors to the desired number of classes [15].

The second architecture is U-Net++ and as its creators mention this algorithm is supposed to be a 'more powerful architecture for image segmentation' [7]. Published in 2018, U-Net++ had a few key differences compared to its predecessor U-Net, as it uses redesigned skip pathways, which help to bridge the prior existing gap between the encoder and decoder sub-networks. Moreover, the U-Net++ architecture is pre-trained on imagenet and contains dense skip connections and deep supervision which improve the segmentation accuracy and the gradient flow while enabling the model to be easily adjustable to find the optimal balance between inference time and performance [7].

Moreover, not only the dataset was tailored to fit the needs of this research, but the segmentation implementation also had to be modified to ensure optimal performance of the model. Thus, the data loader was adapted to the given dataset, and the training runtime of the model was decreased by 35% to boost the operation's speed. This boost was achieved by experimenting with changing different hyper-parameters and finding a suitable number of workers for the dataloader. The parameters which contributed the most to the outcome were the number of workers, the batch size, and the number of tiles, see Section 3.3.

Furthermore, the option to add data augmentations was implemented with the use of the python library named Albumentations [16] to test the impact on the model's performance. As the library offers a wide variety of transforms, for this research only a handful of relevant augmentations were applied, as their relevancy was determined considering the given scenario of detecting the safe landing zones, for a BVLOS UAV, using semantic segmentation. In the next part, the transforms which yielded the best result will be analyzed and explained, see Section 4.4.

3.3 Tiling

Due to the restricted amount of available computing power, we decided to use a method called tiling to split the image into smaller images. This way the model will not lose any data. In addition, tiling can help improve the output of the model.

There are 3 compelling methods of tiling: fixed tiling, random tiling, and positive tiling. Given an image with the implementation of *fixed tiling*, this image will be split in x amount of tiles. These tiles will have a fixed position that stays the same for each image. See Figure 2. *Random tiling* is a variation of tiling that gets tiles from an image at a random position. This position will then change for each image. Indeed the results from random tiling will not be favorable since many tiles will not contain a target (the car in this example). *Positive tiling* however, expands on this method and makes sure that the random tiles that are generator are certain to contain a target. In our case, positive tiling could result in a better pixel distribution between the classes. However, positive tiling may result in unrepresentative data as the Aeroscapes dataset contains some images without the unsafe class. Moreover, due to the limited hardware resources, we decided to only include fixed-, and random tiling in our experiments

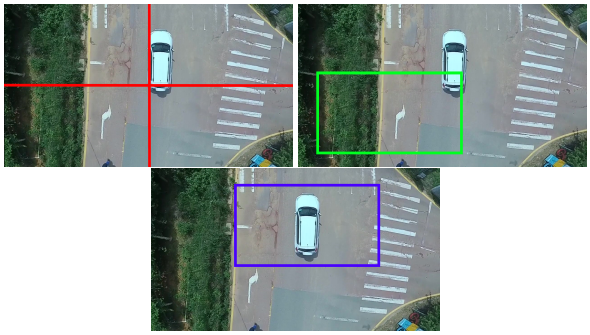

Fig. 2: The procedure of fixed tiling (Red), random tiling (Green), and positive tiling (Blue).

3.4 Hardware and Software

Table 1 shows the hardware specifications of the virtual machine that is used during this project.

Hardware Specifications	
CPU	8 Cores @ 2.4GHz
CPU Memory / RAM	29.2GB
GPU Model	Nvidia GeForce RTX 2070 Super
GPU Memory	8 GB
CUDA Version	11.0

Table 1: Hardware specifications of the virtual machine

The programming language Python is used throughout the project. With the software comes multiple packages, which the most outstanding being:

- Pytorch [17]
- Albumentations [16]

3.5 Evaluation Metrics

To determine the validity of the model's predictions, the following will present the evaluation metrics implemented in the testing part.

3.5.1 Intersection Over Union (IoU)

Being one of the most used and relevant metrics regarding segmentation tasks, the intersection over union, also known as the Jaccard Index, represents the area between the predicted output segmentation and the ground truth, divided by the area of union between the outputted segmentation and the ground truth. The ratio ranges from 0 to 1, where 0 is signifying that there is no relationship between the prediction and the ground truth, and 1 represents a perfect overlapping segmentation, see F 3).

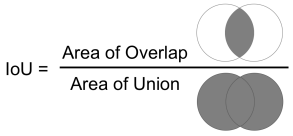

Fig. 3: IoU

3.5.2 Precision and recall

As seen in Figure 4, the precision is calculated by dividing the true positives by the true positives + the false positives and is defined as the ratio of accurately predicted positive observations to the total predicted positive observations. Thus, a high score relates to a lower false-positive ratio. The recall is calculated similarly to the precision metric, the true positives are divided by the true positives + false negatives and it is defined as the rate of correctly predicted positive observations to all observations in the given class. Both metrics are using a range between 0 and 1, as a rule of thumb the higher the score the better is the model.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Fig. 4: Precision & Recall.

3.5.3 F1 score

The F1 score represents the weighted average of precision and recall and it takes into consideration both false positives and false negatives from an operation and is calculated as shown in Figure 5. The F1 score shows the output of the unsafe class predicted by the model, meaning it is class-dependent.

$$F1 - Score = \frac{2 * precision * recall}{precision + recall}$$

Fig. 5: The formula of the F1 score.

4 Experiments & Results

The experiments are categorized to ensure an efficient experiments layout. The different parts of experiments then contain experiments that are focused on the same parameters. For the first set of the experiments, an ensemble of fixed parameters that last throughout all the parts until they encounter their matching part is chosen. The best

parameter that is determined by the experiments is used for future experiments. The dataset used in these experiments is the AeroScapes dataset [5].

For each experiment, the metrics IoU, F1-Score, Recall, and Precision are noted. These metrics are used to decide which model performs better, and thus which parameters are being used for future experiments. The unsafe class can be seen as the positive class in these experiment metrics.

Because some experiments may perform better over a longer period, the experiments are run for a high number of epochs to ensure that the model has enough time to learn. The model is automatically stopped early to prevent overfitting. This method is fine-tuned to ensure that the model can push to its best possible state, yet it saves time by stopping the training when the model starts overfitting. When the model was stopped early, the procedure stays the same, and the model start testing.

4.1 Tiling

The first two experiments are designed to answer the question: ”Does tiling improve the output of the model?”. Without tiling, the input images need to be downscaled to fit the model due to a lack of hardware memory. With tiling, this is not the case, since the images are being separated into smaller chunks that can fit the model. Table 2 shows the first set of experiments, based on fixed tiling. In these experiments, a total of 4 tiles with a height of 360px and a width of 640px is used. In the first experiment, no tiling is applied and the images are being downscaled to fit the model. The second experiment uses fixed tiling with a tile size of 360px by 640px. In experiment 3, random tiling is applied in comparison to experiment 2 where fixed tiling was used. Positive tiling is not applied, since positive tiling is based on a single class, while we are interested in both classes, see Section 3.3. Random tiling is applied to both the training and validation set. The test set remains untouched.

Tiling Experiments

Exp	Model	LR	Optimizer	Tiling Method
1	U-Net	1e-3	Adam	None
2	U-Net	1e-3	Adam	Fixed Tiling
3	U-Net	1e-3	Adam	Random Tiling

Table 2: The first set of experiments, based on tiling.

The results of these three experiments show the impact that tiling can have. Although the model had to train for longer with fixed tiling compared to no tiling, the results improved. The IoU improved by approximately 0.25 see Table 3. From Table 3 we can also conclude that the F1-Score more than doubled when using fixed tiling. The precision increase by 0.063 and the recall increased by almost 0.4. This table also shows a distinct difference between the fixed tiling and random tiling results. The F1-Score, recall, and precision scored lower than with fixed tiling. This is probably due to an unrepresentative and unstable dataset caused by random tiling.

Figure 7 shows the model’s output of the first experiment, where no tiling is applied. The legend of this image is shown in Figure 6. From both, the figures Figure 7 and Figure 8 we can see that the model has improved on detecting contours, and filling in objects.

Tiling Results

Exp	IoU	F1-Score	Precision	Recall
1	0.297	0.338	0.742	0.219
2	**0.548**	**0.689**	**0.805**	**0.603**
3	0.311	0.331	0.622	0.225

Table 3: The results of the first set of experiments, based on tiling.

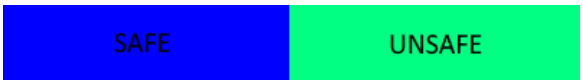

Fig. 6: Legend

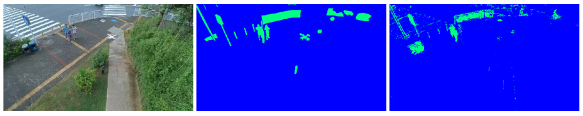

Fig. 7: Experiment 1 - No tiling applied
From left to right: input image, ground truth, and model output

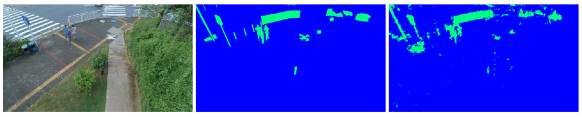

Fig. 8: Experiment 2 - Fixed tiling applied
From left to right: input image, ground truth, and model output

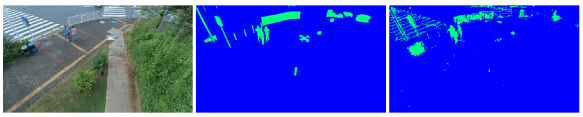

Fig. 9: Experiment 3 - Random tiling applied
From left to right: input image, ground truth, and model output

From both Table 3 and Figures 7, 8, and 9 we can conclude that the output of the model significantly decreases when applying random tiling. With random tiling, the model has a harder time detecting unsafe objects. Hence we continue to use fixed tiling throughout the remaining experiments

4.2 Optimizers

A vital part of getting the best model is choosing an optimizer. In experiments 4 to 7, we take a look at the effect of several optimizers. The optimizers Adam, Adamax, AdamW, SGD, and ASGD were chosen as these seemed most suitable for this project. Table 4 shows the third set of experiments, with various optimizers.

As shown in Table 5, AdamW gave similar results to Adam, while Adamax significantly improved the model compared to Adam. The IoU improved 0.2149 compared to the baseline Adam experiment. Two completely different types of optimizers were also used in this set of the experiments. The SGD optimizer implements stochastic gradient descent, where the ASGD optimizer stands for the averaged stochastic gradient descent. The results found in Table 5 show a significant drop in performance in the ASGD optimizer compared to the Adam optimizer. From the results shown in Table 5 and Figure 10 we can conclude that Adamax is the best performing optimizer for this project and hence will be used for the remaining experiments.

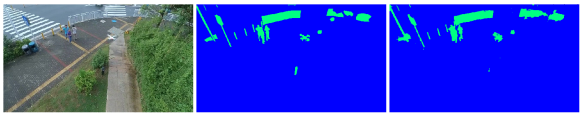

Fig. 10: Experiment 4 - Adamax Optimizer

4.3 Learning rate

With the learning rate fixed at 1e-3 for the previous experiments, we decided to experiment with a few lower learning rates to decrease the spiking in the validation loss which was encountered before. This leaves us with experiments 8 and 9, with a corresponding learning

Optimizer Experiments

Exp	Model	LR	Optimizer	Tiling Method
- 2 -	U-Net	1e-3	Adam	Fixed Tiling
4	U-Net	1e-3	Adamax	Fixed Tiling
5	U-Net	1e-3	AdamW	Fixed Tiling
6	U-Net	1e-3	SGD	Fixed Tiling
7	U-Net	1e-3	ASGD	Fixed Tiling

Table 4: The third set of experiments, with various optimizers.[1]

Optimizer Results

Exp	IoU	F1-Score	Precision	Recall
-2-	0.548	0.689	0.805	0.603
4	**0.763**	**0.880**	**0.922**	**0.842**
5	0.542	0.683	0.801	0.596
6	0.522	0.672	0.785	0.543
7	0.220	0.213	0.384	0.147

Table 5: The results of the third set of experiments, with various optimizers.[1]

rate of 1e-4 and 1e-5, see Table 6. Although only 3 generic learning rates were chosen, the aim is to find a more precise learning rate in a future experiment.

Learning Rate Experiments

Exp	Model	LR	Optimizer	Tiling Method
- 4 -	U-Net	1e-3	Adamax	Fixed Tiling
8	U-Net	1e-4	Adamax	Fixed Tiling
9	U-Net	1e-5	Adamax	Fixed Tiling

Table 6: The fourth set of experiments, with various learning rates. [1]

Table 7 shows the results found from experiments 8 and 9. A learning rate of 1e-3 was used as a baseline. The results from those three different learning rates do not differ that much from each other. This difference can also be seen as noise. Although experiment 9 with a learning rate of 1e-5 shows a small improvement in IoU and F1-Score we decided to use experiment 8 with a learning rate of 1e-4 for further experiments since this model's image output showed qualitatively better results than a learning rate of 1e-5, see Figure 11 and Figure 12. Therefore a learning rate of 1e-4 is used for further experiments.[1]

Learning Rate Results

Exp	IoU	F1-Score	Precision	Recall
- 4 -	0.763	0.880	0.922	0.842
8	**0.764**	**0.877**	**0.833**	**0.828**
9	0.768	0.896	0.910	0.883

Table 7: The results of the fourth set of experiments, with various learning rates.[1]

[1] '- -' indicates the best previous experiment which is used as a baseline for the current set of experiments.

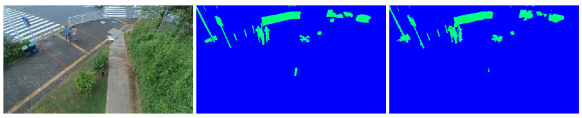

Fig. 11: Experiment 8 - Learning Rate of 1e-4

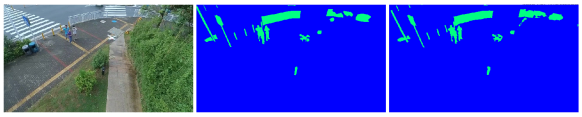

Fig. 12: Experiment 9 - Learning Rate of 1e-5

4.4 Data Augmentations

The next set of the experiments focuses on data augmentations and their impact on the model's performance, as mentioned in Section 3.2. Thus, from all the transforms available in the Albumentations library, six of them were determined to be the most relevant for the experiments of this project [9]. Hence, the optimizer and learning rate which had the better performance in the previous experiments were kept for this one, and gamma, blur, horizontal flip, vertical flip, rain, and fog were tested.
Moreover, after getting the best performing data augmentation, an additional experiment (experiment 16) was done by combining those transforms, to see if the model can improve further. Table 8 shows the fifth set of experiments, with various data augmentations.

Data Augmentation Experiments

Exp	Model	LR	Optimizer	Augmentation(s)
- 8 -	U-Net	1e-4	Adamax	None
10	U-Net	1e-4	Adamax	Gamma
11	U-Net	1e-4	Adamax	Blur
12	U-Net	1e-4	Adamax	Horizontal Flip
13	U-Net	1e-4	Adamax	Vertical Flip
14	U-Net	1e-4	Adamax	Rain
15	U-Net	1e-4	Adamax	Fog
16	U-Net	1e-4	Adamax	Combination

Table 8: The fifth set of experiments, with various data augmentations.[1]

From the results represented in Table 9 we can see that all the data augmentations besides blur managed to improve the models IoU with approximately 0.02 to 0.03. For the combination of the data augmentations, we can see that the model's output improved even further with approximately 0.01 to 0.02 compared to the separate data augmentations.

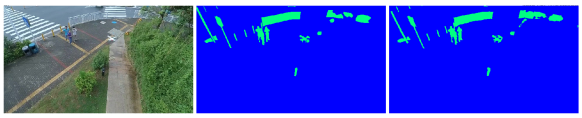

Fig. 13: Experiment 16 - Combination of Augmentations

Figure 13 shows the output of the last and best performing U-Net model. With the IoU being just over 0.8 this model is considered to be a reliable one in predicting the ground truth of the given dataset. The output shows that the model can clearly distinguish objects from the environment.

Data Augmentation Results

Exp	IoU	F1-Score	Precision	Recall
- 8 -	0.764	0.877	0.933	0.828
10	0.794	0.911	0.946	0.878
11	0.763	0.896	0.910	0.882
12	0.797	0.921	0.925	0.918
13	0.789	0.908	0.917	0.900
14	0.792	0.911	0.931	0.892
15	0.774	0.892	**0.940**	0.848
16	**0.809**	**0.927**	0.936	**0.919**

Table 9: The results of the fifth set of experiments, with various data augmentations.[1]

4.5 Model Architectures

The above experiments were realized by using the U-Net architecture presented in Section 3.2. Although to answer the second research question of this paper, we decided to use one more segmentation architecture, U-Net++ Section 3.2. Thus, to compare the two architectures, only the most impactful experiments from U-Net were recreated, using mostly the same parameters to achieve a truthful comparison, as seen in Table 10. The only parameter changed was the size of the tiles, from 360px by 640px to 352px by 640px in order to fit the convolution requirements of the U-Net++ model. Moreover, in S 4.1 it can be seen that tiling drastically improves the model's performance, thus for U-Net++ this parameter was used by default, so no additional testing is necessary.

Model Architecture Experiments

Exp	Model	LR	Optimizer	Augmentation(s)
17	U-Net++	1e-3	Adam	None
18	U-Net++	1e-3	Adamax	None
19	U-Net++	1e-3	SGD	None
20	U-Net++	1e-4	Adamax	None
21	U-Net++	1e-5	Adamax	None
22	U-Net++	1e-4	Adamax	Combination

Table 10: The sixth set of experiments, with U-Net++.[2]

Model Architecture Results

Exp	IoU	F1-Score	Precision	Recall
17	0.318	0.341	0.783	0.218
18	0.677	0.784	0.866	0.716
19	0.542	0.711	0.850	0.612
20	**0.727**	**0.869**	**0.910**	**0.831**
21	0.566	0.759	0.819	0.707
22	0.665	0.814	0.876	0.760

Table 11: The results of the sixth set of experiments, with U-Net++.[3]

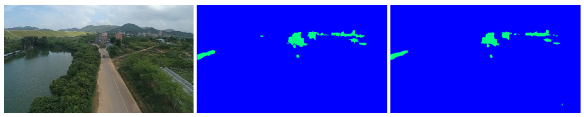

Fig. 14: Experiment 20 - The best model output

From the results seen in Table 11 it can be observed that throughout the experiments, U-Net++ performed worse than U-Net, even with the augmentations applied and using the best optimizer. This might be due to the large structure of the U-Net++ architecture which achieved its pick model early in the training. Even though the model output image seems to match the ground truth quite well, it can be seen that it had difficulties in correctly predicting the edges of the unsafe class, yielding some false negatives and false positives.

5 Discussion, Conclusion & Future Work

This part describes the discussion, based on the research questions and aims to answer the problem statement. The conclusions derived from the results will be detailed and further work will be proposed.

5.1 Discussion

When analyzing the results of the experiments it can be observed that the U-Net architecture yields the highest metrics when augmentations are applied. The IoU has a score of 0.8094, an F1 score of 0.927, and a recall of 0.919, it can be concluded that using a segmentation approach is appropriate when implementing the fail-safe landing system in a UAV.

To answer the research question 'How does a different segmentation algorithm compare with the baseline U-Net architecture?', the results of the experiments conducted in Section 4.5 will be detailed. As shown in tables Table 10 and Table 11, when choosing the most suitable optimizer for both segmentation architectures, it can be concluded that Adamax yielded the best metrics. Although, U-Net++ did not outperform U-Net, achieving only an F1 score of 0.866, IoU being 0.6779 and recall of 0.831, when the F1 score for U-Net was 0.880, IoU of 0.7638, and recall score of 0.842. As for the learning rate, both models performed better when using 1e-4, even though U-Net++ presented a higher discrepancy between the metrics achieved from the three different learning rates, compared to U-Net there the scores were somehow similar. For the best learning rate, U-Net++ achieved better results in terms of precision and recall, scoring 0.910 and 0.831, but performed slightly poorly in terms of IoU and F1-Score. Thus, comparing the results of both architectures, it can be concluded that U-Net++ did not outperform U-Net in this project. This might be due to several factors, such as the large structure of the U-Net++ architecture, the relatively short dataset, and the tuning done to the model which was less for U-Net++ compared with U-Net.

As for the third research question, "Which hyperparameters have the most impact on the performance of the segmentation model?", it can be observed that the improvement for U-Net architecture is exponential from the first experiment Table 4.1 to the last Table 4.4. The first set of experiments in Table 4.1 concludes that using tiling was far more lucrative in this situation and yielded better results than the alternative of not using tiling. Section 4.1 compared the use of a function that let the model pick random tiles. The final output showcased that by not using random tiles the model performs better. The third set of experiments in Table 4.2 compares five different optimizers. While the worst performing one is ASGD with the metrics being lower than what the model achieved without tiling implemented. The next three optimizers, Adam, AdamW, and SGD perform somehow similarly with a low difference margin regarding their test metrics, While Adamax yields outstanding results in comparison with the rest. The model achieved an IoU of 0.763 and an F1-Score of 0.880. The next set of experiments tests the difference between three different learning rates Table 4.3. As the results part presents, the difference in the metrics is extremely slim, thus the learning rate of 1e-4 was chosen for the remaining experiments. The last set of the experiments tested the use of different augmentations from the Albumentations library. After testing, the findings concluded that augmenting the dataset did indeed help improve the model's outcome. The best performing augmentations were gamma which achieved the highest precision of 0.94, horizontal flip with the highest F1 score of 0.921, IoU of 0.797, and recall of 0.91, vertical flip, having metrics slightly lower than the prior two, and lastly, rain which had close results as gamma and horizontal flip. Moreover, the best performing augmentations were combined and the U-Net model

managed to achieve the highest metrics, outputting an IoU score of 0.809 and an F1-Score of 0.927. These being mentioned, it can be concluded that starting from a basic implementation of U-Net, the most impactful hyper-parameters are the use of tiling, the use of an adequate optimizer, and the usage of augmentations.

In terms of the bottlenecks encountered in the process of writing this research, the dataset is the principal one. Namely, the unbalance found among the images, as some classes occur more frequently than others. In terms of total pixels amount, was observed that from the classes belonging to the unsafe landing areas, the classes of animals, bikes, and constructions, occurred significantly less than the ones of cars, people, or obstacles. This might interfere with how the U-Net model is performing. Thus, for further studies, this might be a valuable point to consider, when choosing the dataset.

5.2 Conclusion

The results aim to answer the problem statement of this research, which looks for ways in which semantic segmentation can be implemented in failure-safe landing systems for BVLOS UAVs. Even though no real-time tests were performed, from analyzing the metrics and visual outputs of Section 4, it can be conceded that the U-Net model achieved positive results and it is worth trying it in a real-life situation, under close supervision.

As this is one of the few pieces of research targeting this specific subject and making use of a semantic segmentation model, the project might serve as a comparison point for further research. The findings of this paper contribute to the ultimate goal of creating a failure-safe landing system accepted by EASA, by showing that using a segmentation approach is suitable for this situation and the use of augmented images can replicate real-life scenarios in which a BLOVS UAV can encounter.

Moreover, in the current stage, the project includes a fully functional implementation of the two segmentation architectures, that comes with an aerial dataset of drone imagery, the implementation of tiling, which can be toggled on and off. Also, the code supports the use of the Albumentations library and many other hyper-parameters, which upcoming researchers can experiment with. The last contribution of this project is that it determined the best performing optimizer to be Adamax with a learning rate of 1e-4 and it showcased the most suitable augmentations to use on aerial imagery, parameters which can represent a starting point for further research.

5.3 Future Work

This project aims to inspire further research, thus a few suggestions will be made. Starting with the dataset, the next step is to gather more imagery data to balance the Aeroscapes dataset and achieve an even distribution of the twelve classes. Furthermore, other researchers might want to use an F or F3 score when outputting the metrics and test the best-performing model again, as the focus will lie upon the unsafe class which is the more interesting one when predicting a safe landing zone. Also, a thorough recall comparison will be interesting to be seen, with the focus shifted from a pixel perspective to an objective comparison. Another piece of advice is to make use of the learning rate scheduler, as this project was decided not to use it, to avoid having a larger experiments list, thus future experiments might yield different results. Additionally, another state-of-the-art segmentation architecture can be implemented and different learning rates can be used and also positive tiling can be implemented. Lastly, integrating this approach into a BLOVS UAV and testing it in real-time will represent an important next step and additionally, the segmentation approach can be compared with the object detection one.

Acknowledgements

This project is financially supported by Regieorgaan SIA (part of NWO) under the RAAK MKB project The BEAST A collaboration between Saxion University and NHL Stenden.

References

[1] Drone market outlook in 2021: industry growth trends, market stats and forecast, February 2021. URL: https://www.businessinsider.com/drone-industry-analysis-market-trends-growth-forecasts?international=true&r=US&IR=T.

[2] Civil drones (unmanned aircraft), Jul 2021. URL: https://www.easa.europa.eu/domains/civil-drones-rpas.

[3] Specific operations risk assessment (sora), Jan 2019. URL: https://www.eurocockpit.be/positions-publications/specific-operations-risk-assessment-sora.

[4] The SOARIZON Team. What are vlos, evlos and bvlos? why do they affect drone operations?, Sep 2020. URL: https://www.soarizon.io/news/what-are-vlos-evlos-and-bvlos-why-do-they-affect-drone-operations.

[5] Ishan Nigam, Chen Huang, and Deva Ramanan. Ensemble knowledge transfer for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1499–1508. IEEE, 2018.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[7] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 3–11. Springer, 2018.

[8] Mohammed Rabah, Ali Rohan, Muhammad Talha, Kang-Hyun Nam, and Sung Ho Kim. Autonomous vision-based target detection and safe landing for uav. *International Journal of Control, Automation and Systems*, 16(6):3013–3025, 2018.

[9] Jan Moros Esteban, Jaap van de Loosdrecht, and Maya Aghaei. Obstacle detection for bvlos drones. *arXiv e-prints*, pages arXiv–2106, 2021.

[10] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. Albumentations: fast and flexible image augmentations. *Information*, 11(2):125, 2020.

[11] Joris Guérin, Kevin Delmas, and Jérémie Guiochet. Certifying emergency landing for safe urban uav. *arXiv preprint arXiv:2104.14928*, 2021.

[12] Ye Lyu, George Vosselman, Gui-Song Xia, Alper Yilmaz, and Michael Ying Yang. Uavid: A semantic segmentation dataset for uav imagery. *ISPRS journal of photogrammetry and remote sensing*, 165:108–119, 2020.

[13] Joon Yeop Lee, Albert Y Chung, Hooyeop Shim, Changhwan Joe, Seongjoon Park, and Hwangnam Kim. Uav flight and landing guidance system for emergency situations. *Sensors*, 19(20):4468, 2019.

[14] Semantic drone dataset, Jan 2019. URL: https://www.tugraz.at/index.php?id=22387.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[16] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. URL: https://www.mdpi.com/2078-2489/11/2/125, doi:10.3390/info11020125.

[17] Facebook. Pytorch, 2021. URL: https://pytorch.org/.

A EXPERIMENT 23 - METRICS

The last experiment of this research can be seen in Table 12 and it reflects the results obtained by U-Net architecture on out of our dataset images. The experiment uses the Semantics Segmentation dataset to assess the architecture's performance on a different set of images. As the dataset was used only in the testing function, it can be seen that the metrics are lower than the one obtained by U-Net on the Aeroscapes dataset.

Performance of best model on different datasets

Exp	Model	Dataset	IoU	F1-Score	Precision	Recall
16	U-Net	AeroScapes	0.8094	0.927	0.936	0.919
23	U-Net	Semantic Drone Dataset	0.2943	0.471	0.516	0.487

Table 12: The result & comparison of the final experiment with U-Net.3.2

B EXPERIMENT 23 - IMAGE

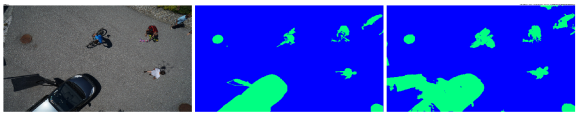

Fig. 15: Experiment 23 - Semantic Drone Dataset