Ensemble Transfer Learning for Emergency Landing Field Identification on Moderate Resource Heterogeneous Kubernetes Cluster

Andreas Klos, Marius Rosenbaum, Wolfram Schiffmann
Department of Computer Science
FernUniversität in Hagen
Germany, North Rhine-Westphalia 58097
Andreas.Klos,Marius.Rosenbaum,Wolfram.Schiffmann@fernuni-hagen.de

Abstract

The full loss of thrust of an aircraft requires fast and reliable decisions of the pilot. If no published landing field is within reach, an emergency landing field must be selected. The choice of a suitable emergency landing field denotes a crucial task to avoid unnecessary damage of the aircraft, risk for the civil population as well as the crew and all passengers on board. Especially in case of instrument meteorological conditions it is indispensable to use a database of suitable emergency landing fields. Thus, based on public available digital orthographic photos and digital surface models, we created various datasets with different sample sizes to facilitate training and testing of neural networks. Each dataset consists of a set of data layers. The best compositions of these data layers as well as the best performing transfer learning models are selected. Subsequently, certain hyperparameters of the chosen models for each sample size are optimized with Bayesian and Bandit optimization. The hyperparameter tuning is performed with a self-made Kubernetes cluster. The models outputs were investigated with respect to the input data by the utilization of layer-wise relevance propagation. With optimized models we created an ensemble model to improve the segmentation performance. Finally, an area around the airport of Arnsberg in North Rhine-Westphalia was segmented and emergency landing fields are identified, while the verification of the final approach’s obstacle clearance is left unconsidered. These emergency landing fields are stored in a PostgreSQL database.

1 Introduction

The loss of thrust depicts a major issue for every pilot. In such a stressful situation quick and focused action is required. There are several reasons why the engines of an aircraft can fail completely. For example, a bird strike on all engines such as in case of flight UA1549 in 2009 and the subsequent forced landing in the Hudson or a technical problem on the single engine of a general aviation aircraft. In those emergency cases each aircraft becomes a glider and the pilot must choose a suitable glide path so that the aircraft arrives at an appropriate altitude at the beginning of the selected landing field e. g. as described in [1,2]. Thereby, the reachability of landing fields is limited by the residual altitude of the aircraft at the time of incident.

Furthermore, it is not guaranteed that an published landing field – in the best case paved – is within reach. In that case, the pilot is compelled to select an emergency landing field which is mainly based on his experience and the emergency guidelines. During the choice of a suitable emergency landing field several terrain properties like the size, shape, slope, surface, surrounding and civilization as well as the current conditions e. g. the wind, season of the year, rainfall etc. have to be considered.
The selection of an appropriate emergency landing field is a crucial task and influences the degree of possible damage of the aircraft and viability of the crew members as well as the passengers. For that reason, our objective is the acceleration of the pilots decision process by providing a database with appropriate emergency landing fields for the specific aircraft type.

For the autonomous identification of emergency landing fields (ELFs), many machine vision and machine learning techniques have been proposed in recent years which can be subdivided in three categories: 1) processing real-time images obtained by on-board sensors; 2) processing pre-acquired data; 3) processing multi-modal images sources.

In [3] an embeddable Convolutional Neural Networks (CNNs) trained on synthetic data to estimate the obstacle cleanness and safeness of regions for landing an UAV is proposed. The approach is tested with UAV footage. Every pixel is assigned to one of the categories: horizontal (landable), vertical (obstacles) and others (safer for a landing). Unfortunately, the classification of flat areas as suitable for an landing leads also to classifications of highways and water aerials as appropriate. In [4] an $k$-Nearest Neighbor approach that considers a feature vector of data acquired by an UAV camera and measures of a light intensity sensor is proposed. Regrettably, only small areas within the field of view are analyzed depending on meteorological conditions. Thus, only a highly restricted number of suitable ELFs can be found. These drawbacks are revised in [5] by aircraft-mounted cameras oriented to the front and a horizon detection algorithm to identify the ground in the images. Besides, they apply a nonlinear retinex image-enhancement method to revamp the environmental effects and improve the contrast and sharpness. The results depending on the resolution of the aircraft-mounted cameras and the altitude of the aircraft. In [6] a surface classification of ELFs is introduced. The classification is performed by a multi-class Support Vector Machine (SVM). Other terrain classifications are proposed in [7] – SVM and AdaBoost for multi-spectral images, in [8] – SVM and multi layer perceptron, and in [9] – premised on SVM and Random Forrests processing monocular camera data. In [10] an algorithm is introduced which applies standard image processing techniques and artificial neural networks to verify obstacle cleanness.

In [11] a two-stage segmentation approach based on satellite imagery is proposed. First, an initial segmentation is performed by analyzing the corresponding histogram of the satellite imagery to estimate the number of different classes. Afterwards, a structure preserving segmentation is performed in the spectral domain. This approach lacks in its ability of edge detection and disregards the suitability for an emergency landing due to its length and width. In [12] another two step processing algorithm is presented which first performs a sectioning of the considered region – Canny edge, line growing – and subsequently a geometric check to ensure the suitability regarding to the shape as well as dimension of the examined region. These image processing steps omit to analyze the slope and bumpiness which are required to guarantee the suitability of located ELFs. In [13] a digital elevation model processing approach is introduced which performs the examination by the quadtree data structure. The metric of variance and average altitude may be insufficient for the selection of a suitable ELF because outliers – caused e. g. by buildings – might be pruned which could lead to a false-negative classification of the corresponding region. Besides, the lack of investigation of the ELF’s surface could hide e. g. water areas which depicts also a weakness to the algorithm proposed in [14] where only elevation data is processed regarding to predefined slope restrictions. In [15] a patch based segmentation approach is proposed based on a CNN aerial images acquired from Google Maps. The classification is performed into the following three categories: Safe, Not recommended, Other. The CNNs performance is evaluated by considering the precision scores achieved for the safe areas (63.8%) and for both, the safe and the not recommended regions (87.1%). The exclusive processing of satellite imagery neglects the bumpiness and might result in false negative classification regarding the landability. A semi-automated emergency landing site selection algorithm is proposed in [16] operating on Google Maps satellite imagery, digital elevation models and a human settlement layer. The segmentation is based on standard image processing methods. Besides, the safety estimation of the ELF considers five different measures. Furthermore, a reachability analysis is performed.

Hence, a combination of processing images from an aircraft-mounted camera and pre-acquired DEMs is shown in [17]. The authors investigated the processing of 2D geodata and reconstructed 3D model. In [18] another multi-modal processing algorithm is proposed. First, preliminary processing steps are performed as mentioned earlier in [19]. Afterwards, man-made and natural objects were distinguished by considering the intensity values. Subsequently, the geometric shape, the surface type and the slope are considered. Unfortunately, the obstacle clearance of the final approach is left unconsidered.
In this paper, we present a patch segmentation of multi-modal geodata based on pre-trained artificial neural networks (ANNs) by the application of ensemble learning. Our implementations are mainly realized in Python with the usage of the PyTorch API and a self-made heterogeneous moderate resource Kubernetes cluster. The manual segmentation of the regions has been evaluated by thresholding so that false-negative classifications are avoided or at least reduced. The datasets are manual labeled by the utilization of QGIS. Our approach utilizes data fusion of the digital surface model and orthophotos to train, validate and test the selected ANNs. After training the selected ANNs, the ensemble model is created and applied to an area around the airport of Arnsberg, in North Rhine-Westphalia. Subsequently, the areas identified as suitable for an emergency landing by the patch segmentation are stored in a PostgreSQL database as georeferenced polygons. Further geographic queries are performed by the usage of the extension PostGIS – a spatial database extender of PostgreSQL – to identify runways. These runways are also stored in the database. In the prior mentioned area we identified 1.590 ELFs and were able to segment 28.021 m² as suitable and 221.979 m² as unlandable.

The paper is structured as follows: In Sec. 2 the utilized dataset, its generation as well as the constructed deeplearning infrastructure are proposed. Afterwards, in Sec. 3 our investigations are proposed and the achieved results are presented as well as discussed. In Sec. 4 the paper finalizes with a conclusion and an outlook on our future works.

2 Dataset generation and deeplearning infrastructure

2.1 Dataset generation

To the knowledge of the authors, due to the time of our research, no public available dataset did exist for the training, validation and test of ANNs regarding classification of multi-modal geodata as landable or unlandable. Therefore, we created different datasets for supervised learning.

The unlabeled and raw geodata – spatial reference system: EPSG 25832 (ETRS89 / UTM Zone 32N) – is downloaded from [20] and [21]. This data is composed of digital orthographic photos (DOP) with four channels (red, green, blue, near infrared with horizontal resolution of 0.2 m per pixel as shown in Fig. 1 (a) and (b)) and digital surface models (DSM) (point clouds with X, Y position and altitude with horizontal resolution of four points m² and 0.2 m in vertical direction).

The point clouds are interpolated by inverse distance weighting with the gdal Python package configured as follows: "invdistnn:power=2.0:radius=1.415:max_points=16:nodata=-2147483648.0". The resulting resolution of the interpolated raster is 1 m per pixel and 0.2 m in altitude. The interpolated DSMs and the DOPs were stored in a PostgreSQL database with the PostGIS extension.

Subsequently, the data became labeled by quadratic polygons with various sizes (32 m², 64 m², 128 m², 256 m²). The polygons are labeled with 0 and 1. Thereby, 0 denotes unlandable and 1 landable areas. Afterwards, the individual examples are queried from the database with respect to the corresponding polygons. Each polygon determines an area for which the corresponding DOP and DSM are queried from the database. The elevation data is interpolated to the same horizontal resolution as the DOP by the bilinear interpolation algorithm [22, P. 88]. Additionally, the roughness and slope is calculated by the usage of PostGIS standard functions (ST_Roughness, ST_Slope) as illustrated in Fig. 1 (e) and (f). The color ranges from white to red where the darkness of red determines the elevation difference and the slope. Besides, the normalized difference vegetation index (NDVI) is computed: ((NIR − Red)/(NIR + Red)). An NDVI image is shown in Fig. 1 (c).
where the color ranges from white to green. The hue of the green color determines the biomass at the considered position.

Each queried area is subsampled in different search windows (SW) of 8 m$^2$, 16 m$^2$ and 32 m$^2$ and stride $\frac{SW}{2}$. Furthermore, each example tagged as landable is evaluated regarding their slope by thresholding. The results are stored in a *.hdf5 file. The sample count of the various generated dataset is as follows: SW 8 m$^2$: [train: 380928 with {0: 190464, 1: 190464}, test: 76288 with {0: 38152, 1: 38136}], SW 16 m$^2$: [train: 84992 with {0: 42498, 1: 42494}, test: 17024 with {0: 8516, 1: 8508}], SW 32 m$^2$: [train: 16768 with {0: 8424, 1: 8344}, test: 3328 with {0: 1672, 1: 1656}].

2.2 Deeplearning infrastructure

The subsequent results are achieved by the utilization of a heterogeneous cluster with high requirements regarding availability and reliability. Therefore, we build a cluster based on Kubernetes (k8s) with the Docker engine and the NVIDIA Container Toolkit. To facilitate the usage of GPUs, the default runtime of Docker is changed to NVIDIA. The nodes are composed of a master, several workers and a network file system (NFS) server. The Master is depicted by a personal computer (PC) dedicated for constantly serving the cluster. The PCs of the chair’s employees as well as one PC provided for always serving the cluster – further denoted as Worker 1 – are the workers. Every worker is configured with a dual boot operating system (OS). If the employee decides, that the PC is currently free for doing some other task, the PC can be booted as Ubuntu 18.04 LTS OS and the worker will be automatically tagged as ready in the k8s cluster. Subsequently, a Pod is launched on the new, ready node. The NFS-Server is launched on Worker 1 to provide the NFS as volume mount on each worker. The hardware configuration of the k8s cluster is detailed in Tab. I.

| Node   | NVIDIA GPU       | CPU              | RAM         |
|--------|------------------|------------------|-------------|
| Master | GeForce GTX 1080 | Intel(R) Core(TM) 17-2600K | 12 GiB DDR3 |
| Worker 1 | GeForce GTX 1080 | Intel(R) Core(TM) 17-8700K | 32 GiB DDR4 |
| Worker 2 | GeForce RTX 2080 Ti | Intel(R) Core(TM) 19-9900K | 32 GiB DDR4 |
| Worker 3 | GeForce GTX 1080 | Intel(R) Core(TM) 17-7700K | 32 GiB DDR4 |
| Worker 4 | GeForce GTX 1080 | Intel(R) Core(TM) 17-7700K | 32 GiB DDR4 |

Additionally, a RabbitMQ message broker is deployed as StatefulSet. The message broker is used for delivering tasks from a queue to its consuming clients. Inside the k8s cluster, the message broker becomes accessible by a dedicated service. This service guarantees the reachability through the assigned DNS entry and the required ports at anytime. The message broker exploits a persistent volume claim to request a persistent volume with ReadWriteOne access mode and a capacity of 1 GiB. To facilitate a high data reliability, the persistent volume is mounted at RabbitMQ’s data storage location in /var/lib/rabbitmq. The message broker is used for distributing the tasks to each worker node. The message broker is configured as follows: 1) Manual message acknowledgments to make sure that a message (task) is never lost e.g. caused by a closed connection, 2) Durable queues and persistent messages which ensures the survival of the queues as well as the messages even if RabbitMQ restarts, 3) Fair message dispatching by setting the prefetch count equal one which configures RabbitMQ to give only one unacknowledged message to a worker at a time.

Furthermore, a PostgreSQL database server is deployed as a StatefulSet. A connection to the database can be established by the usage of the corresponding service as described before. The database uses a persistent volume claim to request a persistent volume. To provide high data reliability, the persistent volume is mounted at the storage location of the database at /var/lib/postgresql/data which guarantees a high reliability of the data. The purpose of the database is the recording of the current state of each worker regarding the task in process accompanied by the storage of the best result achieved for each task during validation and test phase.

The message broker and database are assigned to Worker 1. Other relevant deployments launch their Pods on those node, e.g. CoreDNS, otherwise the shutdown of one of the other workers (Worker n with n ∈ 2, ..., 4) might result in an error.

1 For further information the authors refer to [23]
Besides, a deployment is created which provides declarative updates for Pods and the corresponding ReplicaSet. The ReplicaSet launches a Pod on each worker. Each process – a containerized task, further called worker-process – running in the container inside the Pod subscribes the message broker. As soon as a queue with tasks is created in the message broker, an individual task is sent to each worker-process. The next task will be first assigned to one of the listening worker-processes, if a successful completion confirmation is sent to the message broker. The results achieved by the worker-process are stored in the aforementioned database. Each task is depicted by a string with the following information: "searchWindow: dataComposition: modelName". The last task in the queue triggers an automatic selection of the task, with the best test results regarding the accuracy for each search size. Subsequently a new task queue is created and sent to the message broker. After receiving the new queue, the novel tasks are distributed as before mentioned to each worker-process by the message broker.

The monitoring of the k8s cluster is performed with the Helm charts Prometheus and Grafana. Periodic backups are done with Velero and a Minio S3 Object Store.

3 Results and discussion

3.1 Best input data composition

It is assumed, that each search size benefits from a certain input data composition. For proofing the assumption, we made a analysis of the following input data compositions on the basis of AlexNet: RGB, NIR, SLOPE, ROUGHNESS, NDVI, DOM, RGB-NIR, RGB-Slope, RGB-NDVI, NIR-Slope, NDVI-Slope, NDVI-NIR, RGB-NIR-Slope, NDVI-NIR-Slope, RGB-NIR-NDVI-Slope. All models are trained four times, maximum 100 epochs, with batch size 128 and early stopping, after 6 Epochs of no improvement. The mean squared error (MSE) loss function and the Adam optimizer are chosen. The parameterization of the Adam optimizer is equal to the default settings of PyTorch framework except weight decay (10^{-5}). No data augmentation is applied to the input data so that the evaluation of the best data composition is more meaningful. The best results achieved regarding accuracy on test dataset are shown in Tab. 2.

Table 2: The results achieved for training, validation and test of AlexNet on the corresponding input data composition. The abbreviation SW denotes the search size and T the duration time.

| SW [m^2] | Data composition | Accuracy [%] | Loss [10^{-3}] | T [s] |
|---------|-----------------|-------------|----------------|------|
|         |                 | train | valid | test | train | valid | test |      |
| 8       | RGB-NIR-Slope   | 99.398 | 99.603 | 99.565 | 9.185 | 6.811 | 7.027 | 552   |
| 16      | NDVI-Slope      | 99.728 | 99.811 | 99.666 | 6.163 | 4.433 | 5.420 | 54    |
| 32      | RGB-Slope       | 99.854 | 99.917 | 99.970 | 5.735 | 5.886 | 5.644 | 19    |

As proposed earlier, different SWs seem to benefit from distinct data compositions. For a SW of 8 m^2 the best test result is achieved on RGB-NIR-Slope data composition. About 332 test samples were classified wrong. Interestingly, the validation accuracy as well as the test accuracy reach higher values than the accuracy achieved during training. We tried to avoid this by a uniform distribution of suitable and unsuitable samples in each dataset. It might be, that the training dataset is harder to classify than the validation and test dataset. Another reason could be the usage of regularization methods like dropout during training, e.g. AlexNet utilizes dropout in its linear classifier layers. In Fig. 2 the maximum accuracy and its corresponding loss values scored on test dataset are plotted.

Figure 2 (a) shows the best results for each data composition, except for DOM which is also omitted in Fig. 2 (b) and (c) regarding the low accuracy values scored by AlexNet. This might be caused by the normalization and the properties inherited by itself. For normalization we used the highest and lowest elevation value occurring in the dataset. If we would apply the trained model on a different area (with other minimum and maximum elevation values) the classification might be even worse. The minimum loss value is achieved by the RGB-Slope data composition. Nevertheless, the accuracy has proven better for all four runs on the RGB-NIR-Slope data composition which results also in a higher average accuracy.

\[\text{If an out of memory (OOM) exception is caught, the batch size becomes halved till the training of the model fits into memory of the GPU.}\]
For a SW of 16 m² the NDVI-Slope data composition results in the peak accuracy of 99.665% on the test dataset which sums up to only 57 false classifications. Furthermore, the necessary computation time is 10 times faster for training, validation and testing the neural network compared to the measured summed time achieved for the RGB-NIR-Slope data composition with a SW of 8 m². This is caused by the lower dimensional dataset (NDVI-Slope: 2 layers, RGB-NIR-Slope: 5 layers) and the total sizes of the datasets. The dataset for the SW of 16 m² is about four and half times smaller. Moreover, the calculated loss values are smaller in all considered phases for the NDVI-Slope data composition. Figure 2(b) shows that the minimum loss is about two times smaller with the RGB-NIR-Slope data composition. However, the accuracy achieved by all four iterations on the NDVI-Slope data composition is higher. Thus, we selected the NDVI-Slope data composition in our further analysis.

The RGB-Slope data composition with SW 32 m² achieved a test accuracy of 99.97 % which means, that only one example has been wrongly classified. Figure 2(c) can be obtained that the RGB-NDVI data composition reached the lowest loss values ($2.531 \cdot 10^{-3}$) with a relative high accuracy. We further investigated the classification results with layer-wise relevance propagation (LRP)\cite{26} as shown in Fig 3.
The input data layers are presented in Fig. 3 (a) - (c). In Fig. 3 (d) and (e) LRP heatmaps for AlexNet trained on RGB-NDVI and RGB-Slope for a landable sample are illustrated. The darkness of red color determines how much the corresponding input pixels were relevant for the classification. If a pixel appears in blue color, these pixels contribute to the contrary prediction. AlexNet trained on RGB-NDVI predicted the sample as unsuitable for an emergency landing (false-positive) and the same model trained on RGB-Slope forecast the sample as landable (true-positive). Obviously, the model trained with RGB-NDVI data composition have issues with input samples covered by shadows. The shadows in the sample contribute with a undeniable share to the prediction (unlandable), while the models forecast trained with RGB-Slope data composition seems to be uninfluenced by the occurrence of shadows. For that reason, we have chosen the RGB-Slope data composition for our subsequent investigations. Furthermore, both heatmaps show that especially the right input data border inhibit the prediction confidence. This might be caused by the applied zero padding (2 × 2) in AlexNets architecture.

### 3.2 Best transfer learning models selection

The selection of the appropriate model depicts a crucial task for the classification performance. The following models are considered: ResNet-18, ResNet-50, ResNet-101 [27], AlexNet [24], VGG16 [28], DenseNet-161 [29], Inception v3 [30], GoogLeNet [31], ShuffleNet v2 with 1.0 × output channels [32], MobileNet v2 [33], ResNeXt-50 32x4d, ResNeXt-101 32x8d [34], Wide-ResNet-50-2 [35]. The training, validation and testing is configured as described in Sec. 3.1. The best results achieved regarding accuracy on test dataset is shown in Tab. 3.

| SW [m²] | Model name               | Accuracy [%] | Loss [10⁻²] | T [s] |
|---------|--------------------------|--------------|--------------|-------|
|         | train | valid | test | train | valid | test |       |
| 8       | ResNet-18            | 99.555 | 99.726 | 99.709 | 10.612 | 14.966 | 14.980 | 702   |
| 16      | Wide-ResNet-50-2     | 99.726 | 99.852 | 99.801 | 11.018 | 40.142 | 39.979 | 382   |
| 32      | AlexNet              | 99.854 | 99.917 | 99.970 | 5.735  | 5.886  | 5.644  | 19    |

Obviously, the ResNet-18 model achieved the highest test accuracy with 99.709% for the SW 32 m². Compared to the results proposed in Tab. 2 for AlexNet, about 110 test samples are less incorrectly classified by ResNet-18. The loss values during investigating ResNet-18 increased compared to AlexNet and is more than twice as high as the loss value computed for VGG16 (see Fig. 2 (d)). Nevertheless, the high accuracy during test leads to the selection of ResNet-18 for our subsequent hyperparameter optimization.

The Wide-ResNet-50-2 model reached the best accuracy of 99.801% on the test dataset for SW 16 m², about 34 (ca. 60%) test samples are less wrong classified compared to the results stated in Tab. 2. However, the loss achieved by Wide-ResNet-50-2 is more than seven times higher than the value achieved by AlexNet, which depicts the lowest loss as shown in Fig. 2 (e). Due to the high margin of the accuracy between AlexNet and Wide-ResNet-50-2, the latter is chosen for the hyperparameter optimization.

For the SW 32 m², ResNet-50, ResNet18, ResNeXt-50 32x4d and AlexNet achieved the same accuracy of 99.97%. The calculated loss for these neural networks differ dramatically as shown in Fig. 2 (f). The calculated loss for AlexNet is much smaller compared to all other networks. Additionally, the processing time required by AlexNet depicts the smallest demand. For that reason, AlexNet is selected for the subsequent hyperparameter optimization.

### 3.3 Hyperparameter optimization

The hyperparameter optimization of the selected models is performed by the utilization of Ax-API for the following hyperparameter: learning rate ∈ [10⁻⁷, 0.5], weight decay ∈ [10⁻⁸, 0.5], optimizer ∈ [Adadelta [36], Adagrad [37], Adam, Adamax [25], AdamW [38], ASGD [39], RMSprop [40, p. 303-305], SGD [40, p. 290-292]], loss function ∈ [BCELoss, MSELoss]. The framework offers off-the-shelf Bayesian Optimization [41] and Bandit Optimization based on Thompson sampling [42].
The objective was the optimization of the validation accuracy. The best hyperparameter configuration for each SW with the number of trials is shown in Tab. 4.

Table 4: Hyperparameter configuration identified by Bandit and Bayesian optimization.

| SW | Trials | Learning rate | Weight decay | Optimizer | Criterion | Feature extraction |
|----|--------|---------------|--------------|-----------|-----------|--------------------|
| 8  | 100    | 1.318·10⁻³    | 7.661·10⁻⁵  | Adadelta  | BCELoss   | False              |
| 16 | 100    | 8.178·10⁻⁶    | 3.140·10⁻⁴  | AdamW     | BCELoss   | False              |
| 32 | 500    | 3.087·10⁻⁹    | 2.115·10⁻²  | AdamW     | MSELoss   | False              |

Obliviously, adjusting all weights results in the highest training and validation accuracy values during the hyperparameter optimization. The achieved results are reported in Tab. 5.

Table 5: Results achieved by the hyperparameter optimization.

| SW [m²] | Accuracy [%] | Loss [10⁻²] | Precision [%] | T [s] |
|---------|---------------|-------------|---------------|-------|
|         | train | valid | test | train | valid | test | train | valid | test |
| 8 8 m²  | 99.998 | 99.965 | 99.958 | 0.135 | 1.973 | 1.691 | 99.999 | 99.967 | 99.984 | 524 |
| 16 16 m² | 100    | 99.959 | 99.959 | 0.027 | 1.023 | 1.334 | 100    | 99.967 | 99.976 | 764 |
| 32 32 m² | 99.937 | 99.958 | 99.940 | 0.626 | 0.592 | 0.508 | 100    | 100    | 99.940 | 17  |

The hyperparameter optimization is composed of 100 trials for SW 8 m² and 16 m² and due to the low computational demand 500 trials of hyperparameter optimization are conducted for SW 32 m². The results show, that the accuracy has increased dramatically for SW 8 m² and 16 m² during the hyperparameter optimization. Considering the accuracy reached during test for SW 8 m², the number of false classification has shrunken to 32 examples. Compared to the results achieved by ResNet-18 reported in Tab. 3 the number of false classified examples is about 7 times smaller. The loss values presented in Tab. 5 are more than 7 times reduced against the values stated in Tab. 3. Additionally, we consider the precision for monitoring the reliability of our classification as suitable for an emergency landing. Fortunately, the precision during training, validation and test is close to 100% for each SW. The accuracy improvement regarding SW 16 m² is quite high. On the test dataset the model classified about 7 samples wrong. Besides, the loss during training, validation and test dropped at least about 30 times compared to the values reported in Tab. 3. The accuracy values for SW 32 m² are comparable to the results in Tab. 3 however, the loss values decreased at least 9 times against the presented once in Tab. 3. The newly trained models with the best identified hyperparameter configuration are used for facilitating the subsequent ensemble learning.

3.4 Ensemble transfer learning model and runway identification

The proposed ensemble transfer learning model is build up in an hierarchical matter. It is composed of the best performing input data composition (Sec. 3.1), the corresponding model (Sec. 3.2) and its hyperparameters (Sec. 3.3) for each SW. The model consists of three siblings which are already powerful by themselves. The predictions of each model are weighted by their confidence. The confidence is calculated as follows: \[
\frac{\text{max}(\text{softmax}(x)) - 0.5}{0.5}.
\] The inference of the model trained for SW 32 m² is performed on the whole area of 250 km² e. g. see Fig. 4 (a).

Figure 4: Ensemble transfer learning model segmentation and ELF identification.
Afterwards, the regions are selected with a lower prediction confidence than 99%. For these areas the inference of the model trained for SW 16 m² is conducted. Thus, the area of interest for the inference shrinks to ≈ 7.171 km² so that the computational demand is reduced about 35 times. Subsequently, model trained for SW 8 m² is applied for areas where the averaged prediction confidence is still < 99%. The resulting averaged prediction is presented in Fig 4(b). Obviously, the road is better identified by our hierarchical ensemble transfer learning model, compared to the segmentation performed by one single model. We were able to segment 250 km² in 28.021 km² as landable and 221.979 km² as unlandable.

Subsequently, 1.590 ELFs are identified with the algorithm shown in Alg. 1. The number of identified ELFs can be heavily influenced by the analyzed rotation_angles and stride during the search. In our investigation, the rotation_angles∈ [0, 10.25, ..., 179] with stride of $\frac{\text{width}}{2}$.

Algorithm 1: Identifying rectangular shaped ELFs.

| Line | Code                                                                 |
|------|----------------------------------------------------------------------|
| 1    | rotation_angles ← [i · \frac{\pi}{179}], where i ∈ [0, 1, ..., 179] |
| 2    | centroid ← get_centroid(polygon)                                     |
| 3    | elf ← get_elf(elf_length, elf_width)                                |
| 4    | stride ← elf.width                                                  |
| 5    | foreach rotation_angle ∈ rotation_angles do                         |
| 6    |     if rotation_angle! = 0 then                                       |
| 7    |         poi ← rotate(polygon, rotation_angle, centroid)              |
| 8    |     else                                                            |
| 9    |         poi ← polygon                                               |
| 10   |     if not check_dimensions(poi) then                                |
| 11   |         continue                                                    |
| 12   |     y_min, y_max, x_min, x_max ← get_polygon_limits(poi)            |
| 13   |     \( \Delta y \leftarrow (y_{\text{max}} - y_{\text{min}}) \)    |
| 14   |     y_start_positions ← ((i · stride), where i ∈ [0, 1, ..., \frac{\Delta y}{\text{stride}} + 1])|
| 15   | foreach y_start_position ∈ y_start_positions do                      |
| 16   |     x_shift ← 0.0                                                   |
| 17   |     while x_max = x_shift >= elf_length                             |
| 18   |         shifted_elf ← shift(elf, x_shift, y_start_position)          |
| 19   |         if contains(poi, shifted_elf) then                           |
| 20   |             resize ← 1                                               |
| 21   |             while contains(poi, shifted_elf) do                       |
| 22   |                 shifted_elf ← optimize_length(shifted_elf, resize)    |
| 23   |                 resize ← resize + 1                                  |
| 24   |             x_shift ← x_shift + resize + 1                           |
| 25   |             output_row(rotate(shifted_elf, -rotation_angle, centroid), resize - 1) |
| 26   |     else                                                           |
| 27   |         x_shift ← x_shift + 1                                      |

4 Conclusion and future works

We proposed the generated dataset as well as the deployed deep learning infrastructure. Furthermore, we performed an in-depth analysis regarding the best input dataset configuration with AlexNet. For distinct SW different dataset configuration have proven the best results. Subsequently, we investigated the best models for the corresponding dataset configuration. The hyperparameters of each selected model are highly improved. The trained models are applied as ensemble transfer learning model to a area of 250 km². Hence, we were able to identify 1.590 ELFs and saved our results in our database.

In future works we will develop our own neural network architectures, verify the obstacle clearness of the final approach and build a recommendation system for the best suitable ELFs.

Acknowledgments and Disclosure of Funding

The authors want to thank Mr. Florian Fromm for his recommendations regarding Kubernetes, RabbitMQ, Prometheus and Grafana as well as for the proofreading the corresponding subsection.
References

[1] S. Izuta and M. Takahashi, “Path planning to improve reachability in a forced landing,” *J. Intell. Robotics Syst.*, vol. 86, p. 291–307, May 2017.

[2] M. Klein, A. Klos, J. Lenhardt, and W. Schiffmann, “Moving target approach for wind-aware flight path generation,” *International Journal of Networking and Computing*, vol. 8, no. 2, pp. 351–366, 2018.

[3] A. Marcu, D. Costea, V. Licăreț, M. Pîrvu, E. Slușanschi, and M. Leordeanu, “Safeuav: Learning to estimate depth and safe landing areas for uavs from synthetic data,” in *European Conference on Computer Vision (ECCV)* (L. Leal-Taixé and S. Roth, eds.), (Cham), pp. 43–58, Springer International Publishing, 2019.

[4] F. M. Rao, S. Aziz, A. Khalid, M. Bashir, and A. Yasin, “UAV Emergency Landing Site Selection System using Machine Vision,” *Journal of Machine Intelligence*, vol. 1, no. 1, pp. 13–20, 2016.

[5] Y. F. Shen, Z. U. Rahman, D. Krušienki, and J. Li, “A vision-based automatic safe landing-site detection system,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 1, pp. 294–311, 2013.

[6] L. Mejias, “Classifying natural aerial scenery for autonomous aircraft emergency landing,” in *International Conference on Unmanned Aircraft Systems*, pp. 1236–1242, 2014.

[7] S. T. Namin and L. Petersson, “Classification of materials in natural scenes using multi-spectral images,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1393–1398, 2012.

[8] T. Y. Kim, G. Y. Sung, and J. Lyou, “Robust terrain classification by introducing environmental sensors,” *8th IEEE International Workshop on Safety, Security, and Rescue Robotics*, 2010.

[9] J. Chetan, M. Krishna, and C. V. Jawahar, “Fast and spatially-smooth terrain classification using monocular camera,” *Proceedings - International Conference on Pattern Recognition*, pp. 4060–4063, 2010.

[10] D. Fitzgerald, R. Walker, and D. Campbell, “A Vision Based Forced Landing Site Selection System for an Autonomous UAV,” *International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pp. 397–402, 2005.

[11] T. Levora, B. Ondřej, and P. Pačes, “Emergency Landing Site Location using Aerial Image Segmentation,” in *29th Congress of the International Council of the Aeronautical Sciences*, (St. Petersburg, Russia), pp. 1–6, 2014.

[12] L. Mejias and D. L. Fitzgerald, “A multi-layered approach for site detection in uas emergency landing scenarios using geometry-based image segmentation,” in *International Conference on Unmanned Aerial Systems*, (Atlanta, Georgia), pp. 366–372, IEEE Control Society, 2013.

[13] M. Garg, A. Kumar, and P. B. Sujit, “Terrain-based landing site selection and path planning for fixed-wing uavs,” in *International Conference on Unmanned Aircraft Systems*, pp. 246–251, 2015.

[14] F. Eckstein, B. Wittich, and W. Schiffmann, “Emergency landing field recognition based on elevation data using parallel processing,” (London), Digital Avionics Systems Conference, DASC, 2018.

[15] I. Funahashi, Y. Umeki, T. Yoshida, and M. Iwahashi, “Safety-level estimation of aerial images based on convolutional neural network for emergency landing of unmanned aerial vehicle,” in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 886–890, Nov 2018.

[16] B. Ayhan, C. Kwan, Y.-B. Um, B. Budavari, and J. Larkin, “Semi-automated emergency landing site selection approach for uavs,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. PP, pp. 1–1, 11 2018.

[17] M. Warren, L. Mejias, X. Yang, B. Arain, F. Gonzalez, and B. Upcroft, “Enabling Aircraft Emergency Landings Using Active Visual Site Detection,” in *Field and Service Robotics SE - 12* (L. Mejias, P. Corke, and J. Roberts, eds.), vol. 105 of *Springer Tracts in Advanced Robotics*, pp. 167–181, Springer International Publishing, 2015.
[18] L. Mejias, D. Fitzgerald, P. Eng, and X. Liu, “Forced landing technologies for unmanned aerial vehicles: Towards safer operations,” in Aerial Vehicles (T. M. Lam, ed.), ch. 21, Rijeka: IntechOpen, 2009.

[19] L. Mejias and D. L. Fitzgerald, “A multi-layered approach for site detection in uas emergency landing scenarios using geometry-based image segmentation,” in International Conference on Unmanned Aerial Systems, (Atlanta, Georgia), pp. 366–372, IEEE Control Society, 2013.

[20] “Digitales oberflächenmodell (dom).” https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/hoehenmodelle/oberflaechenmodell/index.html Accessed: 2019-10-01.

[21] “Digitale orthophotos (dop).” https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/luftbilderzeugnisse/digitale_orthophotos/index.html Accessed: 2019-10-01.

[22] R. C. Gonzalez and R. E. Woods, Digital Image Processing (3rd Edition). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[23] M. Luksa, Kubernetes in Action. Manning Publications, 2018.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems 25 (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[25] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” International Conference on Learning Representations, 12 2014.

[26] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, Layer-Wise Relevance Propagation: An Overview, pp. 193–209. Cham: Springer International Publishing, 2019.

[27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” CoRR, vol. abs/1512.03385, 2015.

[28] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv 1409.1556, 09 2014.

[29] G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” CoRR, vol. abs/1608.06993, 2016.

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” CoRR, vol. abs/1512.03385, 2015.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” CoRR, vol. abs/1409.4842, 2014.

[32] N. Ma, X. Zhang, H. Zheng, and J. Sun, “Shufflenet V2: practical guidelines for efficient CNN architecture design,” CoRR, vol. abs/1807.11164, 2018.

[33] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation,” CoRR, vol. abs/1801.04381, 2018.

[34] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” CoRR, vol. abs/1611.05431, 2016.

[35] S. Zagoruyko and N. Komodakis, “Wide residual networks,” CoRR, vol. abs/1605.07146, 2016.

[36] M. D. Zeiler, “ADADELTA: an adaptive learning rate method,” CoRR, vol. abs/1212.5701, 2012.

[37] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” Journal of Machine Learning Research, vol. 12, no. 61, pp. 2121–2159, 2011.

[38] I. Loshchilov and F. Hutter, “Fixing weight decay regularization in adam,” CoRR, vol. abs/1711.05101, 2017.

[39] B. T. Polyak and A. B. Juditsky, “Acceleration of stochastic approximation by averaging,” SIAM Journal on Control and Optimization, vol. 30, no. 4, pp. 838–855, 1992.

[40] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org
[41] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, “Constrained bayesian optimization with noisy experiments,” *Bayesian Anal.*, vol. 14, pp. 495–519, 06 2019.

[42] D. Russo, B. V. Roy, A. Kazerouni, and I. Osband, “A tutorial on thompson sampling,” *CoRR*, vol. abs/1707.02038, 2017.