Skip-Connected Deep Convolutional Autoencoder for Image Restoration

project report

Raphael Viehauser (k51811030) - David Lehrner (k11833349) - Waad Al-Awamleh (k12137178) - Maximilian Burger (k51849415) - Moritz Haderer (k11774793)

1. INTRODUCTION

Airborne Optical Sectioning (AOS) is a viable technique for imaging occluded targets from an aerial perspective [2]. Employing a line-camera setup mounted on a drone, AOS captures multiple images at different focal lengths and then uses light-field technology to synthesize integral images. The integration of these images results in a synthetic wide aperture, allowing for revelation of occluded targets from above. Beyond its applications in search-and-rescue missions, AOS may find utility in diverse fields, such as environmental monitoring, and agricultural assessment.

The challenge of de-occlusion, a term used here to describe the removal of obstructing elements from an image, presents an interesting problem in image restoration. To solve this task, training a neural network to discern and eliminate occlusions holds significant promise. The processed images leave behind only a reconstructed, de-occluded target. This task poses inherent difficulties for the neural network, as it must learn intricate features and patterns to effectively address occlusions.

2. METHODS

The project was implemented with Keras 3 [3]. Keras is a higher level framework offering GPU acceleration, a flexible models API and cross-framework support (pytorch, tensorflow, jax). For the final training we used a paid version of Google Colab with an A100 GPU.

3. MODEL

In this project, we adopt the RedNet [1] architecture to address the problem at hand (see figure 1). The RedNet is a deep convolutional autoencoder with skip-connections, a simple yet powerful architecture that has demonstrated good results in various image restoration tasks, such as denoising and super-resolution. The use of skip-connections facilitates the propagation of high-level semantic information across different layers of the network, preserving information of crucial details needed to solve the restoration problem. Further it prevents gradient vanishing which would naturally occur with deep networks.

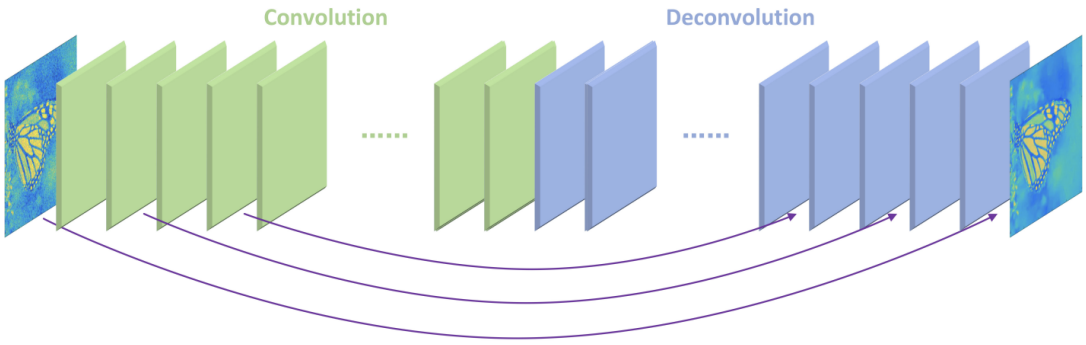

Figure 1: The RedNet model architecture sketch consists of an encoder and decoder, which contain convolutional and deconvolutional layers respectively and skip connections for information flow. Typically, the architecture is symmetric, meaning that there is the same number of layers in the encoder and decoder [1].

Each layer in the RedNet uses the same number of kernels (64) with the same shape (3x3). What had to be adapted from the original paper to our use case, is that the input channels do not match the output channels. In our case, the input of the encoder gets a 6-channel image and the decoder outputs a 1-channel image. To enable the outermost skip-connection, it was changed from an addition to a convolution with a 1x1 kernel. This simple trick turns the 6-channel image into a 1-channel image, which then can be added to the output of the network, as a residual connection.

As in the original paper, we use ReLU activations and an interval of 2 for the skip connections, meaning that every other layer there is a skip connection to its corresponding layer in the decoder.

Our final architecture consists of 11 layers in the decoder as well as in the encoder. Overall the model then has 22 layers. For that reason we will refer to it as REDNet22. Overall the model has a very pleasantly small footprint of only 762.664 trainable parameters, amounting to around 2.8Mb in size. A detailed model output as given by keras is included in the appendix.

4. DATA

Simulated drone images of occluded forest patches were provided in the form of bundles, each comprising 11 images. Each of the 11 images in a bundle corresponded to an individual simulated shot captured by one of the 11 cameras within the line-camera setup. Accompanying the image bundles were ground-truth images showing the de-occluded targets, as well as parameter files specifying the density of occlusion and the number and pose of target individuals present. This results in a total of 13 files per bundle. A large dataset of 33,000 such bundles was made available for our study.

To process these bundles, an Airborne Optical Sectioning (AOS) integrator script was provided. This script leverages light-field technology to integrate the information from the 11 images obtained by the line-camera setup into a single, integral image. The integral image contains information from all 11 images. Bundles that are incomplete, meaning where at least one of the 13 files is missing, were filtered out. Around 2.8% of the bundles were incomplete, and thus, removed.

The most important hyperparameter for us, when creating an integral image is the height of the focal plane of the resulting integrated image. In the context of our methodology, we opted to produce six images at varying heights of the focal plane, measured from the ground. The selected focal stack is as follows: 0cm(ground), 40cm, 80cm, 120cm, 160cm, and 200cm.

The chosen focal stack was theorized to allow for a high-resolution representation of any possible human position in the images. If they are laying (0cm-40cm), sitting (0cm-120cm) or standing (0-200cm).

poses	number of samples		trees per ha	number of samples
laying	8213		0	2552
sitting	8237		100	14709
idle	8182		200	7371

Table A: Training dataset sample size, showing balanced distribution of scenarios sorted by pose and number of trees per hectare.

All samples were then split randomly into a training set and images used for validation and testing. For the training set, we take 85% of all samples. This amounts to 27281 samples that can be used to update the weights. From the 15% (4815 samples) that are not used for training, we use 128 images to validate the model during training. This leaves 4687 images for testing, which the model never sees until final evaluation.

The 27281 training set consists of many images that do not contain a person. Since the main focus was to reconstruct people and not an empty forest, we decided to remove the samples without a person for training. The reasoning was that the model will probably learn how to

reconstruct an empty forest without explicit examples, as it is a large part of the other images as well. This reduced the training dataset to 24632 [see table A].

Since this is a quite large number of images and we only have limited compute resources, we chose to crop the images to a smaller size. Specifically we cropped a 128x128 pixel window around the person. The person was located with classical thresholding on the ground truth image. Then, a 128x128 pixel window was fit, so the person is in the center. In cases where the person is close to the edge of the image, the 128x128 window is shifted to be inside the boundaries in the image, resulting in the person not being centered in the cropped image.

Our reasoning for why cropping would be beneficial is the following. Since we work with a convolutional network, the model should learn similar features on the cropped images as on the full images. Because convolution is translation invariant, it does not matter if the person is in the center, top left or right of the image. Only the features count. Further, we theorized that the hard part of the reconstruction is indeed the reconstruction of the person and not turning the image of trees into a uniform background. Therefore, we think it is efficient to increase the ratio of the person in the image, compared to the background; in the 128x128 window, there is still a significant amount of background where the model can learn. The cropping effectively reduces our dataset size by ~93% and speeds up computation substantially.

The REDNet Architecture can handle input images of various dimensions, and we hypothesize that when the model learns to reconstruct the cropped images, it can also perform decent reconstruction on the full sized images.

5. TRAINING

For training, we leveraged Keras 3 for high-level training tools. Keras 3 offers seamless incorporation of model checkpoints, validation processes, CSV logging, and early stopping callbacks. To better understand the model's performance in varying scenarios, we implemented multiple validation callbacks, evaluating its performance across images without occlusion, moderate occlusion, heavy occlusion, and images without a target. For each of the validation subsets, loss curves were plotted.

We trained the model for a total of 112 epochs in several stages. In the first stage, we used an exponentially decaying learning rate that went from 0.001 to 0.0004 in the first 40 Epochs. After that, the learning rate was fixed to 0.0001 and the model was further trained for 20 epochs. The model had already reached a decent Mean Squared Error of 0.0051 on our cropped validation set. However, the performance was not yet satisfying.

We attempted to train the model further, but now swapping out the MSE loss for the mean absolute error. This idea came from the thought that the loss values are already so small that learning is difficult. Due to the squaring, most values get too small to produce significant gradients. Therefore we use the Absolute instead of the square, to further train the model.

For 40 more epochs with a learning rate of 0.0001 the model increased in performance, ending up at around 0.0043 Mean Squared error. We trained for 12 more epochs with an even lower learning rate of 0.00001 and an increased batch size of 512, however the loss stagnated, at which point we stopped training [see figure 2].

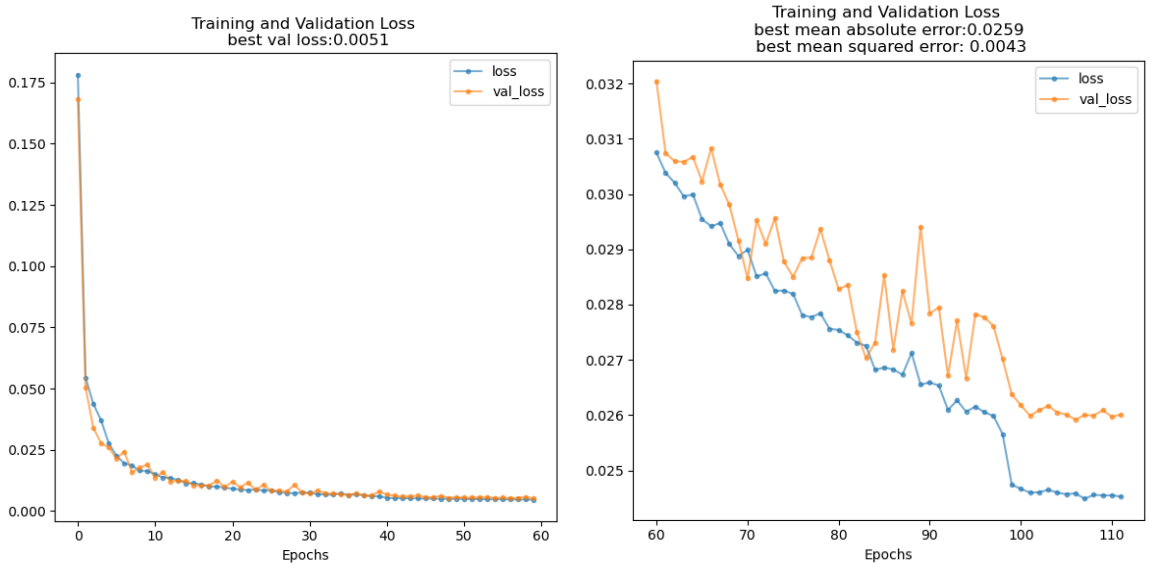

Figure 2: Training and validation loss during training. The final model reaches a loss of 0.0043 on the validation set. On the left, the first 60 epochs are depicted, showing an exponential decline in loss. On the right, the model was trained after the 60th epoch with mean absolute Error, showing linear decline and ending up at around 0.0043 Mean squared error.

6. RESULTS

The final model is evaluated on a selection of images from the test set for all possible scenarios e.g. ‘100 trees per ha, sitting person’. Each combination of the poses (no person, laying, sitting, and standing) and number of trees (0, 100, and 200) per hectare results in 12 test sets of 45 images each (540 images in total) for the final evaluation. They are randomly selected from the full test set and have the full original dimensions of (512x512) pixels. To evaluate the model we compare the model’s prediction on the final test set and the ground truth, this is done by calculating the Peak Signal-to-Noise Ratio (PSNR) and the mean squared error [see table B].

	no person	laying	sitting	idle (standing)
0 trees per ha	0.00002 48.16854 dB	0.00004 43.90106 dB	0.00004 44.52174 dB	0.00004 44.43128 dB
100 trees per ha	0.00314 25.03689 dB	0.00364 24.39364 dB	0.00345 24.62316 dB	0.00328 24.83675 dB
200 trees per ha	0.00913 20.39679 dB	0.01083 19.65303 dB	0.00906 20.42896 dB	0.00782 21.07045 dB

Table B: Showing the mean squared error on top and the Peak Signal-to-Noise Ratio (PSNR) on the bottom for all possible combinations of scenarios in the final testing set.

A clear and expected trend is the growing loss where there are more trees occluding the person [see figure 3]. A slight difference in error for the different poses is observed, (except in the case of 200 trees per ha). However this might be due to stochasticity in the selection of the test sets.

With a MSE of ~0.004 over all the 12 test sets, the performance is comparable with that on the small and cropped validation set. This is good, as it can be interpreted that our assumptions are correct; learning on the cropped images can be transferred to the full size images.

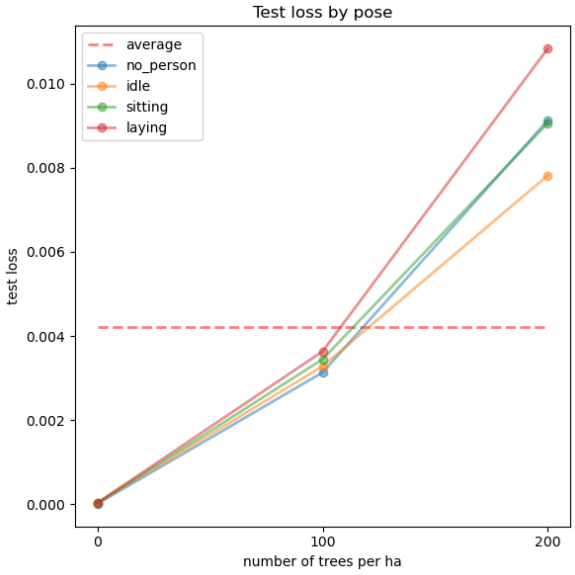

Figure 3: visualizes the mean squared error, where each line represents a pose and the average is indicated with the red line.

It is important to acknowledge the fact that there is less to no information about the person in some of the datasets due to heavy occlusion. Therefore, it is to be expected that our model will fail in some of these scenarios. This highlights the inherent complexity of the de-occlusion task.

In a general term, our model does a decent job reconstructing images, where the background is dark, in comparison to a light background. Whereas, the model is limited in discriminating trees and persons due to the low contrast [see figure 4].

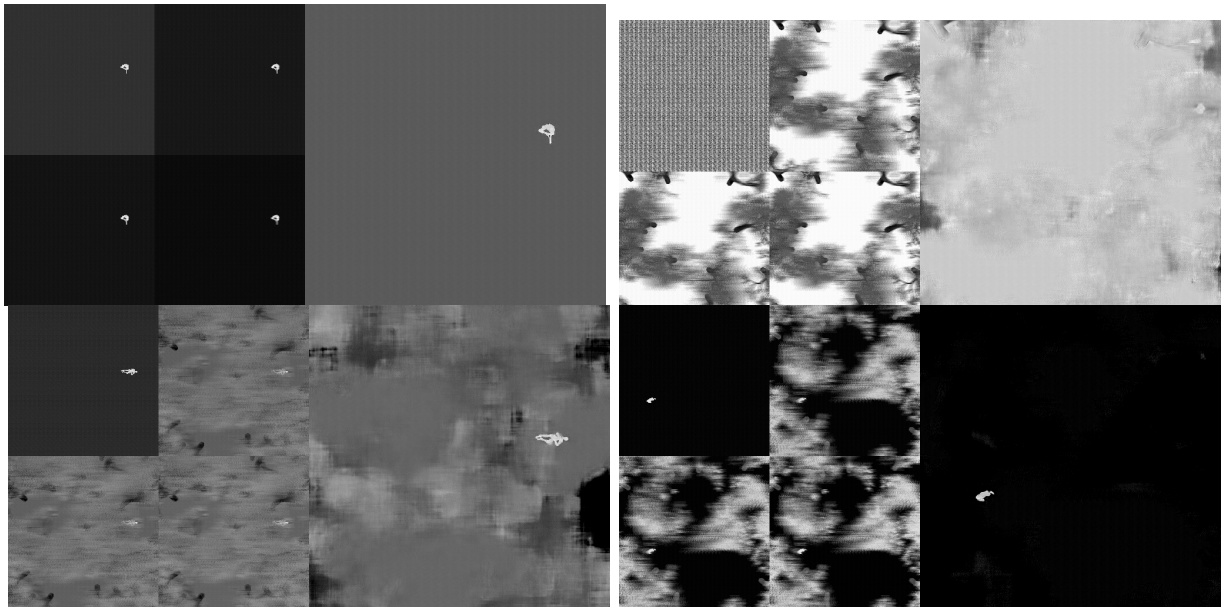

Figure 4: depicts the scenarios that showcase the model's strengths and limitations across different scenarios. Each part of the figure shows the ground truth (a), the stacks with focal sizes; 0.0 (b), 0.8 (c), 1.6 (d) and the model's prediction using the test set (e). Scenarios presented in the figure are; sitting person with no trees (top left), person laying down with 200 trees per ha (bottom left), no person with 100 trees per ha (top right), and standing person with 100 trees per ha (bottom right).

7. THE REAL FOCAL STACK

Our model exhibited a good result on the real focal stack that was provided. It is important to note that this image is different from the one presented, as there was an error when resizing the image. For completeness, the 'wrong' prediction that was presented last time is shown [see figure 5]. Even though this might not count, we still want to include it as it correctly represents the capabilities of our model. The person is clearly visible in the reconstruction, however the background could not be removed completely.

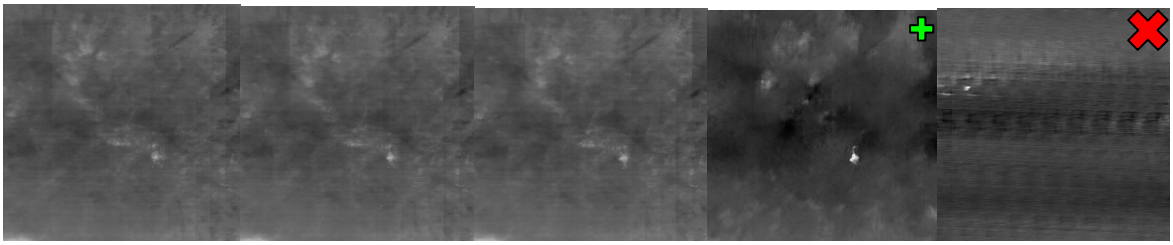

Figure 5: shows the stack of different focal lengths; From left to right 0.0, 0.80 and 1.60 meters, the model prediction marked in green (second from the right) and the prediction presented on Monday the 22.01.24 during the final project presentations marked in red (on the far right side).

8. DISCUSSION

While our RedNet22 convolutional autoencoder performed well in scenarios with low to moderate occlusion, challenges remained in handling heavily occluded targets. This limitation underlines the complexity of de-occlusion in densely obscured scenarios. Potential improvements to our approach could be achieved using a more detailed focal stack and image augmentations. The current use of six focal lengths aims to capture high-resolution views of potential human positions while keeping the dataset size within limits. Expanding the focal stack could improve the model's ability to discern heavily occluded targets. Furthermore, image augmentations are known to add to the robustness of vision models by introducing certain invariances.

It is also worth mentioning that cropping the dataset by approximately 93% significantly accelerates computation, and despite the reduction in size, the model's performance, as indicated by a Mean Squared Error (MSE) of approximately 0.004 suggests that learning on the cropped images can effectively transfer to the full-size images.

One vital limitation of our approach comes from the fact that images without occluded target individuals showed an anomalous distribution of pixels, leading to a checkered pattern [see figure 4.a, top right]. This pattern anomaly clearly poses a challenge to the model, as it has to learn an entirely different mapping to solve this particular subtask.

Overall, the model selection was decent, as it provided us with a small footprint and manageable training resources that are required. It clearly demonstrates the capability of reconstructing the images, although a bigger model with more advanced data augmentation and training procedure might be necessary to be on par with Transformer models.

9. REFERENCES

[1] Mao, X.-J., Shen, C., & Yang, Y.-B. (2016). *Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections* (arXiv:1606.08921). arXiv. http://arxiv.org/abs/1606.08921

[2] Kurmi, I., Schedl, D.C. & Bimber, O. Combined person classification with airborne optical sectioning. *Sci Rep* 12, 3804 (2022). https://doi.org/10.1038/s41598-022-07733-z

[3] Chollet, F., Keras, (2015), GitHub repository, https://github.com/keras-team/keras

10. APPENDIX

The full model description as given when calling keras model.summary() method. Names have been shortened for better readability. DeConv2D is originally named Conv2DTransposed.

```
Model: "REDNet22"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
input_18 (InputLayer)       [(B, W, H, 6)]      0         []
L147 (Conv2D)          (B, W, H, 64)       3520      ['input_18[0][0]']
L148 (Conv2D)          (B, W, H, 64)       36928     ['L147[0][0]']
L149 (Conv2D)          (B, W, H, 64)       36928     ['L148[0][0]']
L150 (Conv2D)          (B, W, H, 64)       36928     ['L149[0][0]']
L151 (Conv2D)          (B, W, H, 64)       36928     ['L150[0][0]']
L152 (Conv2D)          (B, W, H, 64)       36928     ['L151[0][0]']
L153 (Conv2D)          (B, W, H, 64)       36928     ['L152[0][0]']
L154 (Conv2D)          (B, W, H, 64)       36928     ['L153[0][0]']
L155 (Conv2D)          (B, W, H, 64)       36928     ['L154[0][0]']
L156 (Conv2D)          (B, W, H, 64)       36928     ['L155[0][0]']
L157 (Conv2D)          (B, W, H, 64)       36928     ['L156[0][0]']
L164 (DeConv2D)        (B, W, H, 64)       36928     ['L157[0][0]']
add_76 (Add)                (B, W, H, 64)       0         ['L164[0][0]','L157[0][0]']
activation_147 (Activation  (B, W, H, 64)       0         ['add_76[0][0]'])
L165 (DeConv2D)  (B, W, H, 64)       36928     ['activation_147[0][0]']
activation_148 (Activation  (B, W, H, 64)       0         ['L165[0][0]'])
L166 (DeConv2D)  (B, W, H, 64)       36928     ['activation_148[0][0]']
add_77 (Add)                (B, W, H, 64)       0         ['L166[0][0]', 'L155[0][0]']
activation_149 (Activation  (B, W, H, 64)       0         ['add_77[0][0]'])
L167 (DeConv2D)  (B, W, H, 64)       36928     ['activation_149[0][0]']
activation_150 (Activation  (B, W, H, 64)       0         ['L167[0][0]'])
L168 (DeConv2D)  (B, W, H, 64)       36928     ['activation_150[0][0]']
add_78 (Add)                (B, W, H, 64)       0         ['L168[0][0]', 'L153[0][0]']
activation_151 (Activation  (B, W, H, 64)       0         ['add_78[0][0]'])
L169 (DeConv2D)  (B, W, H, 64)       36928     ['activation_151[0][0]']
activation_152 (Activation  (B, W, H, 64)       0         ['L169[0][0]'])
L170 (DeConv2D)  (B, W, H, 64)       36928     ['activation_152[0][0]']
add_79 (Add)                (B, W, H, 64)       0         ['L170[0][0]', 'L151[0][0]']
activation_153 (Activation  (B, W, H, 64)       0         ['add_79[0][0]'])
L171 (DeConv2D)  (B, W, H, 64)       36928     ['activation_153[0][0]']
activation_154 (Activation  (B, W, H, 64)       0         ['L171[0][0]'])
L172 (DeConv2D)  (B, W, H, 64)       36928     ['activation_154[0][0]']
add_80 (Add)                (B, W, H, 64)       0         ['L172[0][0]', 'L149[0][0]']
activation_155 (Activation  (B, W, H, 64)       0         ['add_80[0][0]'])
L173 (DeConv2D)  (B, W, H, 64)       36928     ['activation_155[0][0]']
activation_156 (Activation  (B, W, H, 64)       0         ['L173[0][0]'])
L174 (DeConv2D)  (B, W, H, 1)        577       ['activation_156[0][0]']
L175 (DeConv2D)  (B, W, H, 1)        7         ['input_18[0][0]']
add_81 (Add)                (B, W, H, 1)        0         ['L174[0][0]','L175[0][0]']
activation_157 (Activation  (B, W, H, 1)        0         ['add_81[0][0]'])
=================================================================================================
Total params: 742664 (2.83 MB)
Trainable params: 742664 (2.83 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________________________________________
```