**Application of Image restoration task based on UNet3+ inspired Architecture - Project Report – Group C3**

*Marcus Pertlwieser*

*Utsaha Joshi*

*Alfi Xavier*

*Tamas Scheidl*

*Vivekanand Ramakrishnan*

**Abstract**

**Introduction**

The following report briefly summarizes the proposed algorithm, outlines the results and discusses the limitations of the suggested UNet3+ approach applied by Group C3 for the given Image Restoration task using integrated images.

**Preprocessing**

The first stage of our implementation is aimed at preprocessing the given input data. This step can be broadly distinguished into Filtering, AOS Integrating and Train-Test splitting.

*Filtering:* The Filtering step is handled based on the completeness of the given input samples. If the provided input data is complete, i.e., if the 11 images, the GT (ground truth) and the parameter file are existent, the code will create a new folder for each pair (based on their “row-number”) in the specified destination folder, and consequently copy the data into the corresponding folders.

*AOS Integration:* Afterwards, the AOS integrator takes the pre-processed folders and generates images for the focal plane lengths of 0, -1.5 and -3m. The resulting images are subsequently merged and saved together to the destination folder along with the associated GT and parameter files.

*Splitting:* Finally, the code for the for splitting the dataset is then used to load the integrated images and group the files in the source directory based on their row\_number. Subsequently, these groups will be randomly shuffled and partitioned into a training, evaluation and testing set, which are split into 3 different directories called “Train”, “Eval and “Test”. The splitting is performed in a ratio of 70%-15%-15% (Train, Test, Eval) and is then used to train the model.

+ Include that we used Max-Normalization

**Architecture**

*Encoder*: The architecture consists of an Encoder like VGG16D, except it does only contain the first three 512-channeled stacks at the 4-th Encoder stage. For the last Encoder stage strided Maxpooling with 3x3 Convolutions were used to generate three 1024 feature stacks subsequently.

*Decoder*: Like UNet3+ each of the Encoder feature stacks were convolved to 64 channels of equal resolution to build up a skip-stack.

The first stage of the Decoder (seen from bottom to up) utilizes all skip-stacks. Entries of skip-stacks from stages above get nearest sampled down to the appropriate resolution, stacks of equal stage are not sampled since they already have appropriate size. Stages from below get up-sampled accordingly.

The first and following Decoder stages receive 320 channels as inputs and convolve them to 64 channels.

After the first Decoder stage, instead of reusing the same 64 channels of the Encoder which were already incorporated, the 64 output channels of previous Decoder stages are reused. This should result in more enriched feature maps the deeper we get into the Decoder.

After the last Decoder we arrive at 64 channels, which are consequently linearly combined with a 1x1 convolution to 1 channel.

*Guidance*: Since ground temperature seems to be one of the key values which should pose as a good reference value for removing unnecessary background information this was used as additional guidance. We used max-normalisation on the ground temperature values. These values were copied into the pixel values of the same size as the original input images. We concatenate this temperature image with the original focal stack input. After a 3x3 convolution to 4 channels we arrive at a feature map which should incorporate information to give the model the capability of correctly assessing the correct ground temperature values. After concatenation with the output of the last 1x1 concatenation of the UNet3+-like part we arrive at 5 channels. Now we should have information on correct values and position of the background and target, therefore another 3x3 convolution to 1 channel should do the trick (for a 1x1 convolution A diagram of a computer

Description automatically generatedcould be argued).

**Training**

A graph of a train loss

Description automatically generated

*Initial Training:* Initial Training was done using a batch size of 8, Adam optimizer with initial learning rate of 0.0001, an initialized VGG16 Encoder which was trained on ImageNet-1K and MSE loss. We chose the batch size of 8 for initial learning because it ensures sufficient exploration of the loss function space. It should also be noted that since this is a regression task with high dimensional output (512x512 in our case), it means that the stochastic effect can significantly be dampened simply for the fact that averages over large spacial dimensions are taken. The idea for the rather small learning rate of the optimizer is to give the Encoder sufficient time to adjust to the new data with momentum increasing the effective learning rate after a few iterations. Initial training on an MSE-SSIM combination was also tried but did not yield good results as it made the architecture prone to get stuck in local minima which focused on ignoring the target and removing the background.

A graph of a graph with a line

Description automatically generated with medium confidence

*Finetuning:* The Finetuning was done using a batch size of 128 Adam Optimizer with a learning rate of 0.00001 and an MSE-SSIM combination of the following form:

**Results**

**Some pics, also of different temperatures…**

**The model shows really good results for the task of image restoration on the test set, for limitations see next section.**

**Limitations**

**Limitations of the model seems to be that it does not well in extrapolating to completely occluded areas of the target. This is to be expected, because generative capabilities were never the focus of our approaches in architecture design, training and evaluation.**