Micro-lens image stack upsampling for hyperspectral light fields

Paul Hasenbusch

October 30, 2025

Contents

1 Models

1.1 Prelimanries

1.1.1 Neural Networks

Definition 1 (Fully Connected Layer) *Let $A \in \mathbb{R}^{n\times m}$ and $b \in \mathbb{R}^m$. We call the mapping*

$$F : \mathbb{R}^n \to \mathbb{R}^m \ , \quad F(x) = Ax + b \ ,$$

a fully connected layer.

In order to refer to the architecture, thats is a fully connected layer with input dimension $n \in \mathbb{N}$ and output dimension $m \in \mathbb{N}$, whose weights are not fixed but subject to optimization, we write $F(n, m)$.

Definition 2 (Paralelization) *Let X, Y be two sets, the parallelization operation P is defined by*

$$P : f(X,Y) \times f(X,Y) \to f(X,Y^2) \ , \quad P(f,g)(x) = \begin{pmatrix} f(x) \\ g(x) \end{pmatrix} \ .$$

1.1.2 Convolutional Layers

Definition 3 (Convolution over multiple feature maps) *Let $X \in \mathbb{R}^{C\times n_1\times\ldots\times n_d}$ and $k \in \mathbb{R}^{C\times d_1\times\ldots\times d_d}$. The convolution of X with the kernel k, denoted by $k * X$ is given by*

$$(k * X)(p_0) = \sum_{i=1}^{c} \sum_{p\in R} k(c,p)X(c, p_0 + p) \ ,$$

where $R = \prod_{i=1}^{d}[0, d_i] \cap \mathbb{N}$, for all $p_0 \in \prod_{i=1}^{d} \mathbb{N} \cap [1, n_i - d_i]$.

Definition 4 (Convolutional Layer) *Let $X \in \mathbb{R}^{C\times n_1\times\ldots\times n_d}$, $k_1, ..., k_{C'} \in \mathbb{R}^{C\times d_1\times\ldots\times d_d}$ and $b \in \mathbb{R}^{C'}$. We call the mapping*

$$C : \mathbb{R}^{C\times n_1\times\ldots\times n_d} \to \mathbb{R}^{C'\times n_1\times\ldots\times n_d} \ , \quad C(X) = [k_1 * X, ..., k_{C'} * X] + b \ ,$$

a convolutional layer.

In order to refer to the architecture, thats is a convolutional layer with input channel dimension $C \in \mathbb{N}$ and output channel dimension $C' \in \mathbb{N}$, kernels $k_1, ..., k_{C'} \in \mathbb{R}^{C\times d_1\times\ldots\times d_d}$ and $b \in \mathbb{R}^{C'}$, which are not fixed but subject to optimization, we write

$$C(n, m, \text{kernel-size} = (d_1, ..., d_d), \text{padding} = p, \text{padding-mode} = m) \ .$$

Definition 5 (Residual Connection) *Let X be some set, $F = \{f : X \to X | f \text{is a function}\}$ be the set of functions on X mapping back on X. The operation*

$$R : F \to F \ , \quad R(f)(x) = f(x) + x \ ,$$

for all $x \in X$, is called the residual mapping.

Pooling operations, as for example max-pooling, reducing the spatial dimension of feature maps by utilizing strides larger than 1 are widely known.

On the other side there are also variations of convolutional operations to increase the size of feature maps. Most notably transposed convolutions and sub-pixel convolutions.

1.1.3 Transformers

Transformers operate on sequences of data $(x_k)_{k=1}^n$, where $x_k \in \mathbb{R}^d$. In the literature the elements of the input sequence are commonly referred to as tokens. In the following we denote the set of sequence over a set A by $S(A)$. Central to Transformer models is the so-called attention mechanism. The tokens x_k are embedded into three different subspaces using linear mappings $Q, K, V \in \mathbb{R}^{d',d}$, to obtain queries $(q_k)_{k\in\mathbb{N}}$, keys $(k_k)_{k\in\mathbb{N}}$ and values $(v_k)_{k\in\mathbb{N}}$, where

$$q_k = Qx_k \ , \quad k_k = Kx_k \text{ and } v_k = Vx_k \ ,$$

for all $k = 1, ..., n$. The queries q_k and keys k_k are used to compute the attention scores among the tokens, measuring the level of relevance of their respective information for each other

$$A_{ij} = \frac{\exp(k_i^T q_j)}{\sum_{k=1}^n \exp(k_k^T q_j)} \ . \tag{1}$$

The outputs are then computed for all $j = 1, ..., n$ by

$$y_j = \sum_{i=1}^n A_{ij} v_i \ . \tag{2}$$

Note that by construction for all $j = 1, ..., n$ holds

$$\sum_{i=1}^n A_{ij} = 1 \ .$$

Definition 6 (Self-Attention) *Let $Q, K, V \in \mathbb{R}^{d',d}$. The operation described in equations (1), (2), that is*

$$SA : S(\mathbb{R}^d) \to S(\mathbb{R}^d) \ , \quad SA(Q, K, V)\left((x_k)_{k=1}^n\right) = (y_k)_{k=1}^n \ ,$$

is called self-attention.

To increase the capacity of Transformer models, multiple self-attention operations, called heads in the literature, are used in parallel to process the input sequence.

Definition 7 (Multi Headed Self-Attention) *Let* $Q_h, K_h, V_h \in \mathbb{R}^{d',d}$ *for* $h = 1, ..., H$ *and let* $Q = (Q_1, ..., Q_H), K = (K_1, ..., K_H), V = (V_1, ..., V_H)$. *The operation*

$$MSA(Q, K, V)\left((x_k)_{k=1}^n\right) = \left[SA(Q_1, K_1, V_1)\left((x_k)_{k=1}^n\right), ..., SA(Q_H, K_H, V_H)\left((x_k)_{k=1}^n\right)\right],$$

is called multi headed self-attention.

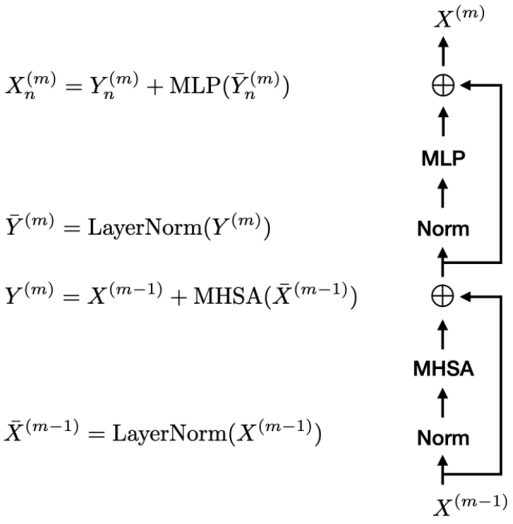

Figure 1: Image taken from [**?**], Transformer Block architecture.

In order to refer to the architecture, that is Multi Headed Self-Attention with dimension $d \in \mathbb{N}$ and number of heads $H \in \mathbb{N}$, whose weights are not fixed but subject to optimization, we write $\mathrm{MSA}(d, H)$.
Given a number of heads $H \in \mathbb{N}$ typically the embedding dimension of each head is chosen as $\frac{d}{H}$.
Another key ingredient for Transformer models is layer normalization.

Definition 8 (Layer Normalization) *Let* $\gamma \in \mathbb{R}$ *and* $\beta \in \mathbb{R}^d$. *The operation given by*

$$LN : \mathbb{R}^d \to \mathbb{R}^d \ , \quad LN(x) = \gamma \bar{x} + \beta \ \textit{where} \ \bar{x}_{ki} = \frac{1}{\sqrt{var(x_k)}}(x_{ki} - \sum_{j=1}^{d} x_{kj})$$

is called layer normalization.

Typically Multi Headed Self-Attention is used in transformer blocks, the architecture is outlined in figure 1. We describe this operation formally in the next definition.

Definition 9 (Transformer Block) *Let $d, H \in \mathbb{N}$ and $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ be some mapping. The architecture*

$$T(d, H, \Phi) = R\big(\Phi \circ LN\big) \circ R\big(MSA(d, H) \circ LN\big)$$

is called a Transformer Block.

Typically the mapping Φ is some neural network, with only a few number of layers. The tokens are processed individually by the mapping Φ, so the sequence is treated as a batch.

Transformers operate on sequential data, so how to apply these models to images? In their 2020 paper *An image is worth 16 × 16 words: Transformers for image recognition at scale*, Dosovitskiy et al. [**?**] propose to partition image data $X \in \mathbb{R}^{C \times H \times W}$ into a sequence of patches, in order to bridge the gap between the two domains. An image is partitioned into patches sized $P \times P$, for some $P \in \mathbb{N}$, these are flattened and embedded using a linear mapping to obtain a sequence $(x_k)_{k=1}^N \in S(R^{C \cdot P^2})$, where $N = \frac{HW}{P^2}$. Let $w = \frac{W}{P}$, mathematically the partitioning can be described as follows

$$\hat{x}_k(i_c P^2 + i_h P + i_w) = X\left(i_c, \left\lfloor \frac{k}{w} \right\rfloor + i_h, k \mod w + i_w\right) , \tag{3}$$

for $k = 1, ..., N$, $i_c = 1, ..., C$ and $i_h, i_w = 1, ..., P$. The flattened patches are embedded to obtain the sequence of tokens $(x_k)_{k=1}^N$, that is

$$x_k = E\hat{x}_k ,$$

for some $E \in \mathbb{R}^{D \times C \cdot P^2}$. We implicitly assume that $H \mod P = W \mod P = 0$, i.e. both the height and the width are devisable by the patch size P. The approach is also visualized in figure 2.

Liu et al. [**?**] point out, that unlike in language, where a word naturally offers itself as the atomic unit, visual elements vary in scale, making the fixed patch sizes unsuitable for tasks requiring predictions at pixel level, as for example semantic segmentation. Simply treating each individual pixel as a token would solve the problem, but at the same time introduce immense computational complexity. For a full HD image of size 1920×1080 this leads to a sequence length of $2.0736 \cdot 10^6$. Thus to reduce computational complexity, but at the same time maintain a global receptive field, Liu et al. [**?**] propose Hierarchical Shifted Window Transformers (SWinT).

As before the image is partitioned into non-overlapping patches, as described in equation (3), Liu et al. [**?**] opt for a patch size $P = 4$, to obtain a sequence of tokens $(x_k)_{k=1}^N$. The tokens are then partitioned into subsequences

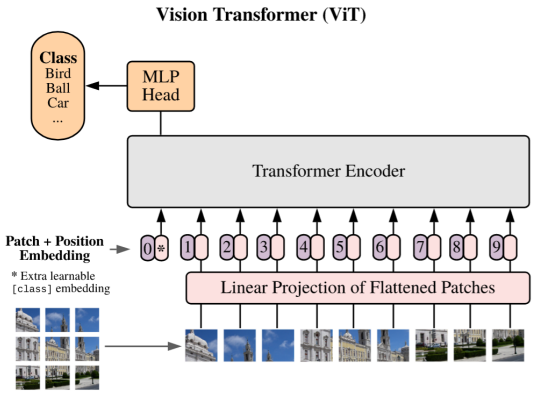

Figure 2: Image taken from [**?**], Vision Transformer.

$$\left(x_{k_l^{(1)}}\right)_{l=1}^{M^2}, ..., \left(x_{k_l^{(N')}}\right)_{l=1}^{M^2}$$

where $N' = \frac{HW}{4^2M^2}$. The subsequences $(k_l^{(p)})_{l=1}^{M^2}$ are chosen so that neighboring patches form a super patch of size $M \times M$, for some $M \in \mathbb{N}$, formally that is

$$k_l^{(p)} = \underbrace{M^2 \cdot \left\lfloor \frac{p}{\left(\frac{W}{4M}\right)} \right\rfloor}_{\text{inter patch row}} + \underbrace{M \cdot \left(p \mod \frac{W}{4M}\right)}_{\text{inter patch column}} + \underbrace{\left\lfloor \frac{l}{M} \right\rfloor \cdot \left(\frac{W}{4M} - 1\right)}_{\text{intra patch row}} + \underbrace{l}_{\text{intra patch column}}, \tag{4}$$

for all $p = 1, ..., N'$ and $l = 1, ..., M^2$, we are enumerating the super patches from left to right, top to bottom, note that $\frac{W}{4P}$ returns the number of super patches along the horizontal axis. Self-attention is then performed locally inside of each super patch

$$(y_{k_l^{(p)}})_{l=1}^{M^2} = \text{MSA}(Q, K, V)\left((x_{k_l^{(p)}})_{l=1}^{M^2}\right), \tag{5}$$

for $p = 1, ..., N'$, for some $Q, K, V \in \mathbb{R}^{H \times \frac{d}{H} \times d}$.

If equation (5) would be used repeatedly to update features, information is restricted to flow only inside individual super patches. To establish information flow amongst super patches, Liu et al. [**?**] introduce the shifted window mechanism. Consecutive operations of self-attention use different partitionings of the sequence $(x_k)_{k=1}^N$. We consider the partitioning described in equation (**??**) as the unshifted variant, for the shifted partitioning the borders of the patches are moved down and to the right by $s = \lfloor \frac{P}{2} \rfloor$ units. In order to achieve the shift, while keeping the same partitioning, a cyclic shift is applied to the feature map,

it is visualized in figure 3. Formally, this can be described by processing an auxilliary feature map $X' \in \mathbb{R}^{C\times H\times W}$, defined by

$$X'(c,i,j) = X\big(c,(i+s) \mod H,(j+s) \mod W\big)\ , \tag{6}$$

for $c = 1,...,C$, $i = 1,...,H$ and $j = 1,...,W$. Instead also padding techniques could be applied, but Liu et al. [**?**] report achieving better results, whilst saving computational complexity by employing the cyclic shift.

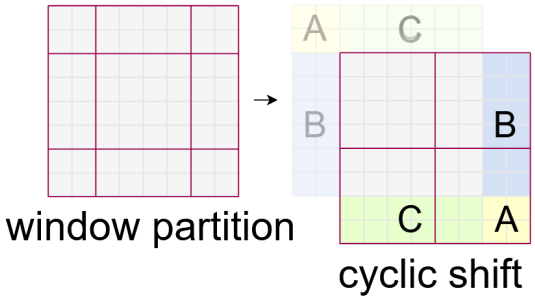

Figure 3: Image taken from [**?**], visualizing the cyclic shift. Here the pixels in the image are rearranged, and then the unshifted partitioning is used, to obtain the cyclic shift.

Definition 10 (Cyclic shift) *We denote the function implementing equation (6) by*

$$CS : \mathbb{R}^{C\times H\times W} \to \mathbb{R}^{C\times H\times W}\ ,\quad CS(X) = X'\ .$$

To enable a global receptive field, Liu et al. [**?**] propose to merge neighboring patches, after the inputs are processed by a certain number of SWin TransformerBlocks. To this end patches $P_1,...,P_4 \in \mathbb{R}^{C\times P\times P}$, inside a neighborhood of size 2×2 are stacked along the channel dimension, to form a super patch $\hat{P} = [P_1,...,P_4] \in \mathbb{R}^{4\cdot C\times P\times P}$. To fuse the features a convolutional layer is applied, halving the channel dimension of the super patch

$$P = C(4\cdot C, 2\cdot C, \text{kernel-size} = 3, \text{padding} = 1)(\hat{P})\ .$$

This process is repeated until the entire feature map is of size $P\times P$. We implicitly assume that $H = 2^{n_1}P$ and $W = 2^{n_2}P$ for some $n_1, n_2 \in \mathbb{N}$. The overall architecture of the SWin Transformer is shown in figure 4.

Definition 11 (Shifted window Transformer Block) *Let $d, H \in \mathbb{N}$ and $\Phi_1, \Phi_2 : \mathbb{R}^d \to \mathbb{R}^d$. The architecture given by*

$$SWinT(d,H,\Phi_1,\Phi_2) = T(d,H,\Phi_2)\circ CS\circ T(d,H,\Phi_1)\ .$$

is called Shifted Window Transformer Block.

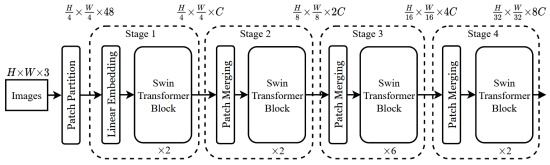

Figure 4: Image taken from [**?**], SWin Transformer.

Chen et al. [**?**] introduce overlapping Cross Attention (OCA), a modification of SWin Transformers. Whilst in equation (4) the patches are constructed to partition the feature map, OCA establishes cross patch connections, by constructing the patches so that they overlap. Formally, let $\gamma \in (0,1)$ be the factor of overlap and $p = 1, ..., N$ index the patches. Let $M_o = \lfloor (1+2\gamma)M \rfloor$, the subsequences associated to patch p is given by

$$k_l^{(p)} = \underbrace{M^2 \cdot \left\lfloor \frac{p}{\left(\frac{W}{4M}\right)} \right\rfloor}_{\text{inter patch row}} + \underbrace{M \cdot \left(p \mod \frac{W}{4M} \right)}_{\text{inter patch column}} + \underbrace{\left\lfloor \frac{l}{M_o} \right\rfloor \cdot \left(\frac{W}{4M_o} - 1 \right)}_{\text{intra patch row}} + \underbrace{l - \gamma M}_{\text{intra patch column}} + \underbrace{\gamma M}_{\text{padding}} , \tag{7}$$

for $l = 1, ..., M_o^2$, note padding was added and only the terms responsible for intra patch indexing were manipulated, this way the overlap is guaranteed. Super patches sharing an edge have an overlap of $\lfloor \gamma M \rfloor M$ pixels, whereas super patches sharing only a corner have an overlap of $\lfloor \gamma M \rfloor^2$ pixels. For query, key and value matrices $Q, K, V \in \mathbb{R}^{d',d}$, for a patch $p = 1, ..., P$, the attention scores are computed in the following way

$$A_{ij} = \frac{\exp({x_i^{(p)}}^T K^T Q x_j^{(p)})}{\sum_{k=1}^{M_o} \exp({x_k^{(p)}}^T K^T Q x_j^{(p)})} , \tag{8}$$

for $i = 0, ..., M_o$ and $j = \lfloor \gamma M \rfloor, ..., M + \lfloor \delta M \rfloor$. The outputs are then computed for all $j = \lfloor \gamma M \rfloor, ..., M + \lfloor \gamma M \rfloor$ by

$$y_j = \sum_{i=1}^{n} A_{ij} V x_i . \tag{9}$$

Thereby every token is updated exactly once, while capturing information from tokens belonging to other partitions. Note that equation (1) and (8) differ only by the indices attained by j, leading to the queries only coming from the part of the super patch, which is not shared by others. The operation is visualized in figure 5.

We conclude this chapter by introducing definitions for overlapping cross-attention, multi headed overlapping cross-attention and overlapping cross-attention block analogously to the definitions 10, 6, 7 and 9.

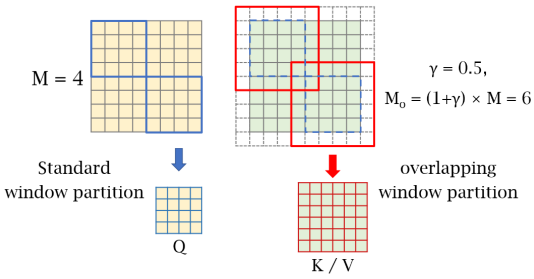

Figure 5: Image taken from [**?**], Overlapping Cross-Attention Block.

Definition 12 (Overlapping Cross-Attention) *Let* $Q, K, V \in \mathbb{R}^{d',d}$. *The operation described in equations (8), (9), that is*

$$OCA : S(\mathbb{R}^d) \to S(\mathbb{R}^d) \ , \quad OCA(Q,K,V)\big((x_k)_{k=1}^N\big) = (y_k)_{k=1}^N \ ,$$

is called overlapping cross-attention.

Analogously to standard self-attention, we also introduce multi headed overlapping cross-attention.

Definition 13 (Multi Headed Overlapping Cross-Attention) *Let* $Q_h, K_h, V_h \in \mathbb{R}^{d',d}$ *for* $h = 1, ..., H$ *and let* $Q = (Q_1, ..., Q_H), K = (K_1, ..., K_H), V = (V_1, ..., V_H)$. *The operation*

$$MOCA(Q,K,V)\left((x_k)_{k=1}^n\right) = \left[OCA(Q_1,K_1,V_1)\left((x_k)_{k=1}^n\right), ..., OCA(Q_H,K_H,V_H)\left((x_k)_{k=1}^n\right)\right] ,$$

is called multi headed overlapping cross-attention.

Definition 14 (Overlapping Cross-Attention Block) *Let* $\delta \in (0,1)$, $d, H \in \mathbb{N}$ *and* $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ *be some mapping. The architecture*

$$OCAB(d,H,\Phi) = R\big(\Phi \circ LN\big) \circ R\big(MOCA(d,H) \circ LN\big) \circ P(\delta)$$

is called a Overlapping Cross-Attention Block.

1.2 General Model Architecture

Independent of the domain of application, a general architectural choice that can be observed in all super resolution models, is that the architecture is made up of three components. A shallow feature extraction module H_S, a deep feature extraction module H_D and an image reconstruction module H_{IR}. Typically the model architecture is conceptualized as follows

$$H = H_{IR} \circ R(H_D) \circ H_S \ . \tag{10}$$

The shallow feature extraction module H_S scales the channel dimension of the input to a higher dimension, which is used throughout the majority of the network. Additionally it extracts low frequency features. The module is usually composed of only one or few convolutional layers.
The deep feature extraction module H_D forms the main part of the model. It is supposed to recover high frequency information. Here is where different architectures proposed in the literature vary the most, convolutional layers, transformer models and various combinations thereof have been tried out.
Note the residual connection in equation (10), the rational behind this being that this way the low frequency features extracted by H_S can bypass the deep feature extraction module H_D. The image reconstruction module H_{IR} maps the input back to the original channel dimension and scales the spatial dimension to the desired size. It has been experimentally confirmed that better results are achieved when scaling is done at the end, rather than processing the already spatially upsampled input. To this end usually transposed convolutional layers or pixel sshuflling layers are employed.

1.3 Single Image Super Resolution

1.3.1 Deep Residual Channel Attention Network

The Deep Residual Channel Attention Network (DRCAN) proposed by Zhang et al. [**?**], the channel attention mechanism is introduced to single image super resolution. Channel Attention enables the network to dynamically assess which feature maps / channels are more important or need more refinement. This is achieved by processing the globally pooled average of the feature maps using a lightweight network and then reweighing the feature maps based thereon.

The overall model architecture is depicted in figure 6. The input image X is first processed via an initial convolutional layer

$$F_0 = C(3, 64, \text{kernel-size} = 3, \text{padding} = 1)(X) \ .$$

The following convolutional layers used in the architecture of the DRCAN are of the form

$$C = C(64, 64, \text{kernel-size} = 3, \text{padding} = 1, \text{padding-mode} = \text{zero}) \ ,$$

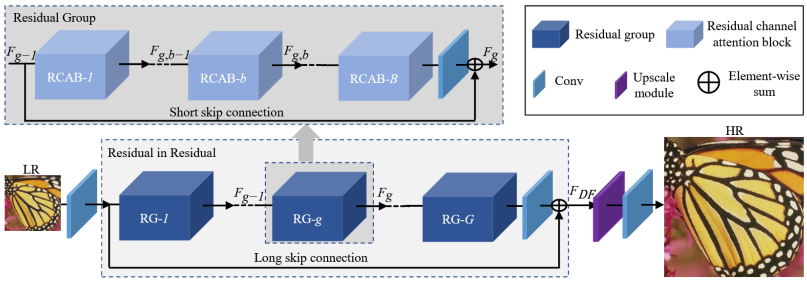

Figure 6: Image taken from [**?**], DRCAN model architecture.

that is 64 ingoing feature maps processed by 64 quadratic kernels of size 3×3 with zero-padding of size 1, so that feature map sizes are conserved throughout the model. The inial features F_0 are then further processed by a network with a residual in residual architecture

$$F_1 = H_D(F_0) \ .$$

For low-frequency features to bypass the deep feature extraction, a residual connection is used before the upsampling is performed

$$F_2 = F_0 + F_1 \ .$$

The final features F_2 are then upsampled using transposed convolutional layers.
The H_{RIR} network is composed of 10 Residual Groups followed by a final convolutional layer, that is

$$H_D = C \circ H_{RG} \circ ... \circ H_{RG} \ .$$

The Residual Groups (RG) are again composed of 20 Residual Channel Attention Blocks followed as well by a convolutional layer, the structure is encapsuled in a residual connection

$$H_{RG} = R\big(C \circ H_{RCAB} \circ ... \circ H_{RCAB}\big) \ .$$

The Residual Channel Attention Block (RCAB) depicted in figure 7, is made up of two convolutional layer, with a ReLU activation function in between, followed by a channel attention module, the output is then added back to the input again via a residual connection

$$H_{RCAB} = R\big(H_{CA} \circ C \circ \mathrm{ReLU} \circ C\big) \ . \tag{11}$$

The channel attention mechanism depicted in 8. The information of a feature map is first condesated into a single value by using global pooling

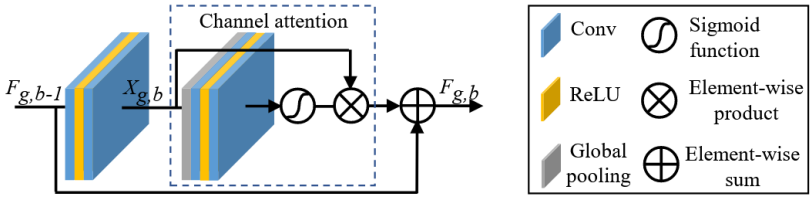

Figure 7: Image taken from [**?**], architecture of RCAB module.

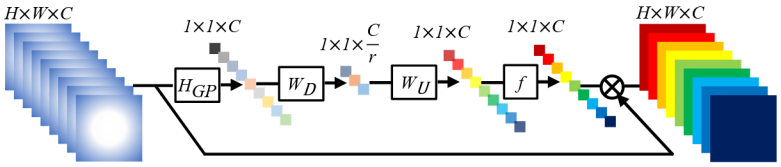

Figure 8: Image taken from [**?**], Channel Attention machanism.

$$z_c = H_{GP}(x_c) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W} x_c(i,j) \ ,$$

with the input $X = [x_1, ..., x_C] \in \mathbb{R}^{C\times H\times W}$. The vector $z \in \mathbb{R}^C$ is then processed by a two-layer neural network

$$\Phi = \sigma \circ F(4, 64) \circ \text{ReLU} \circ F(64, 4) \ ,$$

the sigmoid activation function is applied at last, in order to squash the attention scores into the interval $[0, 1]$. Channel attention the weights the inputs according to the attention scores

$$H_{CA}(X) = \Phi \circ H_{GP}(X) \cdot X \ . \tag{12}$$

1.3.2 Shifted Window Transfomer Image Restoration

The SWinIR model proposed by Liang et al. [**?**], makes use of the shifted window transfomer architecture introduced by Liu et al. [**?**]. While the model does not employ the hierarchical structure of the original architecture, it makes extensive use of the shifted window mechanism. The model architecture is depicted in figure 9.

The broader architectural design follows that described in section 1.2. Given inputs $X \in \mathbb{R}^{3\times H\times W}$, the shallow feature extraction is performed via a single convolutional layers

$$F_0 = C(3, 180, \text{kernel-size} = 3, \text{padding} = 1)(X) \ .$$

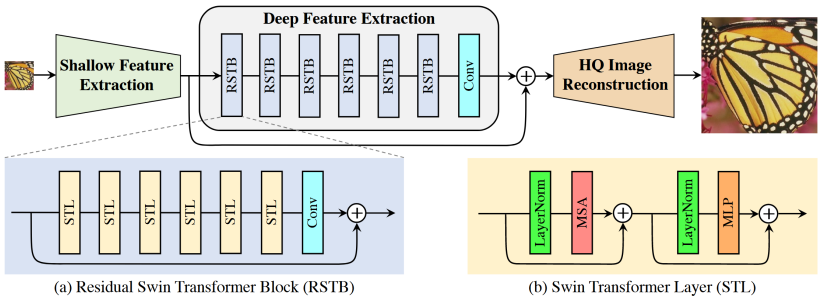

Figure 9: Image taken from [**?**], architecture of SWinIR model.

The features are then further processed by the deep feature extraction module

$$F_1 = H_D(F_0) + F_0 \ ,$$

before being upsampled using the image reconstruction module $I_{SR} = H_{IR}(F_1)$. To this end the authors employ a sub-pixel convolutional layer.

The deep feature extraction module is composed of 6 Residual Swin Transformer Blocks (RSTBs), followed by a last convolutional layer

$$H_D = C(180, 180) \circ H_{RSTB} \circ ... \circ H_{RSTB} \ .$$

Each RSTB is consists of 6 Transformer layers where every second makes use of the shifted window mechanism, or using the notation introduced in section 1.1.3 3 Shifted Window Transformer Blocks SWinT, these are suceeded by a final convolutional layer

$$H_{RSTB} = C(180, 180) \circ \text{SWinT}(180, 6, \Phi, \Phi) \circ ... \circ \text{SWinT}(180, 6, \Phi, \Phi) \ .$$

The network $\Phi : \mathbb{R}^{180} \to \mathbb{R}^{180}$ is given by

$$\Phi = F(360, 180) \circ \text{GELU} \circ F(180, 360) \ . \tag{13}$$

1.3.3 Hybrid Attention Transfomer

The Hybrid Attention Transformer proposed by Chen et al. [**?**], combines the shifted window mechanism as used in SWinIR by Liang et al. [**?**] and Channel Attention [**?**], in a Hybrid Attention Block where both operations are performed in parallel. On top of that they introduce Overlapping Cross-Attention, which we already discussed in section 1.1.3.

The overall architecture is shown in figure 10. The architecture follows the general scheme described in section 1.2, initial feature extraction is performed

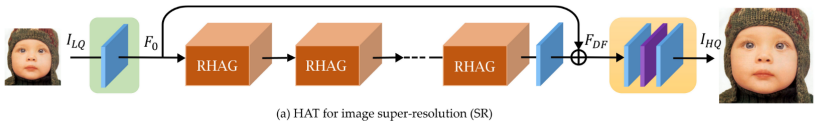

Figure 10: Image taken from [**?**], architecture of HAT model.

via a single convolutional layer, spatial upsampling is achieved using subpixel convolution. The deep feature extraction module proposed by Chen et al. [**?**] is composed of 6 cascaded Residual Hybrid Attention Groups (RHAG)

$$H_D = H_{RHAG} \circ ... \circ H_{RHAG} \ .$$

A RHAG is made up of 6 Hybrid Attention Blocks (HABs) followed by one OCA-Block as defined in 14 and a convolutional layer

$$H_{RHAG} = C(180, 180) \circ \text{OCAB}(180, 6, \Phi) \circ H_{HAB} ... \circ H_{HAB} \ .$$

A HAB is a modification of a Shifted Window Transformer Block, where in parallel to the Shifted Window Multi Headed Attention, Channel Attention Block (CAB) H_{CAB} is employed

$$H_{CAB} = H_{CA} \circ C \circ \text{ReLU} \circ C \ , \tag{14}$$

with H_{CA} defined as in equation 12. Let $s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d, s(x, y) = x + y$ be the element-wise summation

$$H_{HAB} = R(\Phi \circ \text{LN}) \circ R(s \circ P(H_{CAB}, \text{SWinMSA}) \circ \text{LN}) \ .$$

The exact architecture of the network Φ is not specified by Chen et al. [**?**]. A construction as in equation 13 can be used. The modules H_{RHAG}, H_{HAB}, OCAB and H_{CAB} are visualized in figure 11.

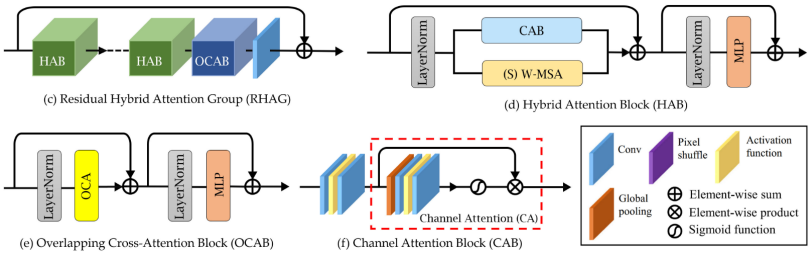

Figure 11: Image taken from [**?**], Residual Hybrid Attention Group.

1.4 Hyper Spectral Image Super Resolution

1.4.1 Full 3D U-Net

Liu et al. [**?**] propose the Full 3D U-Net (F3DUN), for hyperspectral image super resolution. The model implements a U-Net architecture composed solely of three-dimensional convolutional layers. Because of the multi-band property of hyperspectral images, 3d convolutions come as a natural candidate for processing this type of data. On the other side, previously pure 3d CNN models were considered difficult to train. As the combination of increased model capacity, compared to 2d convolutional networks, and the the absence of large datasets for HSI related tasks, the risk of overfitting is present. Liu et al. [**?**] disprove this common assumption in their work, by introducing a more carefully designed model architecture, outperforming previous state-of-the-art methods. The overall architecture is displayed in figure 12.

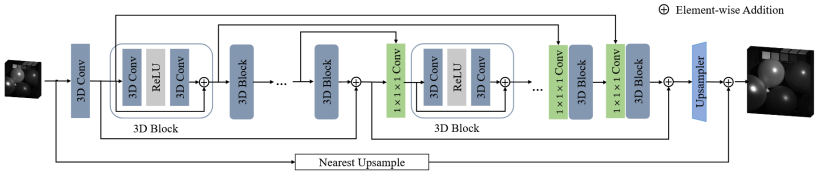

Figure 12: Image taken from [**?**], architecture of F3DUN.

For shallow feature extraction one 3d convolutional layer $H_S = C_3(31, 64, \text{kernel-size} = 3, \text{padding} = 1)$ is used. The features are scaled to the desired spatial dimension using a transposed 3d convolution.
Throughout the architecture the following convolution is mostly used

$$C = C_3(64, 64, \text{kernel-size} = 3, \text{padding} = 1) \ .$$

The basic building block of the model is the 3d Block, 10 of such block are employed

$$H_{3dBlock,i} = R(C \circ \text{ReLU} \circ C) \ , \tag{15}$$

for $i = 1, ..., 10$. Given an input $X \in \mathbb{R}^{31 \times H \times W}$, the shallow features $F_0 \in \mathbb{R}^{64 \times H \times W}$ are computed by

$$F_0 = H_S(X) \ .$$

The features are then passed through the contracting path of the U-Net

$$F_i = H_{3dBlock,i}(F_{i-1}) \ ,$$

for $i = 1, ..., 4$. In contrary to the standard U-Net architecture, the authors introduce two more skip connections, reaching from the beginning of each path to its end. For the contracting path this is

$$F_5 = H_{3dBlock,5}(F_4) + F_0 \ . \tag{16}$$

Next the features are further processed by the expanding path. As it is common in U-Nets, the features from the previous block are concatenated with the features put out by the block at the same level in the contracting path. To this end, the following module is used

$$H_{fuse,i} = C_3(128, 64, \text{kernel-size} = 1) \ , \tag{17}$$

for $i = 6, ..., 10$. To map the concatenated features down from 128 to 64 channels, the fuse module defined in (17) is employed before the 3d block

$$F_i = H_{3dBlock,i} \circ H_{fuse,i} \left(\left[F_{10-i}, F_{i-1} \right] \right) \ .$$

For $i = 6, ..., 9$, note F_{10-i} is the feature, coming from the same level of the contracting path. Analogously to the contracting path, as in equation (16), a skip connection is also used in the expanding path

$$F_{10} = H_{3dBlock,10} \circ H_{fuse,10} \left(\left[F_9, F_0 \right] \right) + F_5 \ .$$

The features F_{10} are the final output of the deep feature extraction module.

1.4.2 Spatial Spectral Aggregation Transfomer

Wang et al. [**?**] propose the Spatial-Spectral Aggregation Transformer (SSAformer), for the task of hyperspectral image super resolution. Similarly to the Hybrid Attention Transformer introduced by Chen et al. [**?**], the Spectral-Spatial Attention Block, the basic building block of the SSAformer architecture, employs channel attention and overlapping cross-attention in parallel. This design is motivated by the need to comprehensively fuse spatial and spectral information, overcoming the limitations of CNNs' locality and conventional Transformers' imbalance between the two domains. The architecture is visualized in figure 13.

For initial feature extraction a single convolutional layer is used. To scale features to the desired size, a sub-pixel convolution is performed.
The deep feature extraction module H_d is composed of 4 Spatial-Spectral Attention Groups (SSAGs), followed by a final convolutional layer

$$H_d = C \circ H_{SSAG} \circ ... \circ H_{SSAG} \ .$$

A SSAG is formed by 4 Spectral-Spatial Attention Blocks (SSABs), succeeded as well by convolutional layer

$$H_{SSAG} = C \circ H_{SSAB} \circ ... \circ H_{SSAB} \ .$$

The SSAB forms the basic building block of the architecture proposed by Wang et al. [**?**]. Overlapping Cross-Attention is employed in parallel, to a Channel Attention Block H_{CAB}, defined in equation (14)

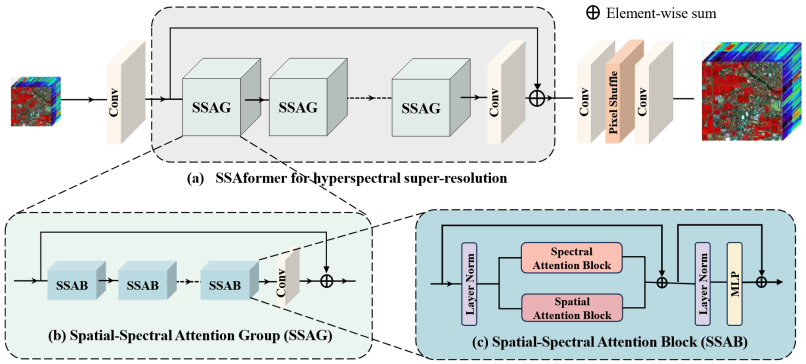

Figure 13: Image taken from [**?**], architecture of RCAB module.

$$H_{SSAB} = R(\Phi \circ \mathrm{LN}) \circ R(s \circ P(\mathrm{OCA}(180, 6), H_{CAB}) \circ \mathrm{LN}) \ ,$$

where $\Phi : 180 \to 180$ is a neural network, the architecture is not specified by the authors. Again, a model as in equation (13) could be used.

1.5 Spectral Super Resolution

1.5.1 Multi-stage Spectral-wise Transformer

Cai et al. [**?**] propose the Multi-stage Spectral-wise Transformer (MST), for the task of spectral super resolution. The idea the authors introduce, is instead of forming tokens by partitioning the spatial domain, entire spectral-bands are being treated as tokens. Given a feature map $X \in \mathbb{R}^{C \times H \times W}$, the spatial domain is flattened, leading to the shape $C \times HW$, each entry in the channel dimension is then treated as a token, yielding $(x_k)_{k=1}^{C} \in \mathcal{S}(\mathbb{R}^{HW})$. The model architecture is depicted in figure 14.

The basic building block of the MST forms the Spectral Attention Block (SAB). A SAB is a variation of a Transformer Block as described in definition 9. Instead of standard multi-headed self-attention, spectral-wise self-attention is used. Given an input $X \in \mathbb{R}^{C \times H \times W}$ tokens $(x_k)_{k=1}^{C} \in \mathcal{S}(\mathbb{R}^{HW})$ are formed, as described in the introduction. The tokens are embedded using matrices $Q_h, K_h, V_h \in \mathbb{R}^{d_h \times C}$, for $h = 1, ..., H$, as in standard multi-headed self-attention, to obtain provisional queries and keys $(\hat{q}_k^{(h)})_{k=1}^{HW}, (\hat{k}_k^{(h)})_{k=1}^{HW}$ as well as values $(\hat{v}_k^{(h)})_{k=1}^{HW} \in \mathcal{S}(\mathbb{R}^{C})$, that is

$$\hat{q}_k^{(h)} = Q_h x_k, \hat{k}_k^{(h)} = K_h x_k \text{ and } \hat{v}_k^{(h)} = V_h x_k \ . \tag{18}$$

In contrary to standard self-attention the tokens are then transposed in

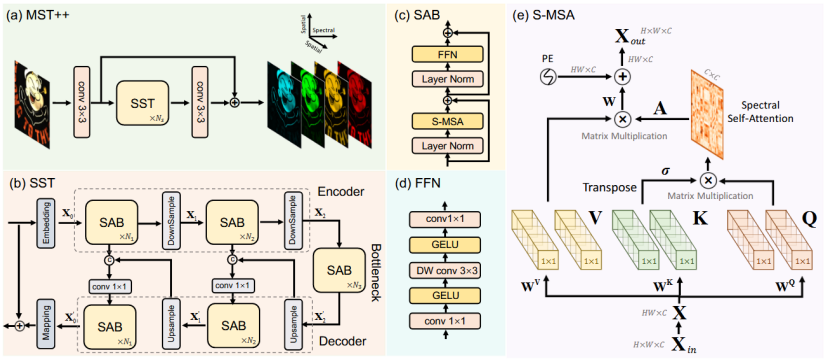

Figure 14: Image taken from [**?**], architecture of MST model.

some sense [1], to obtain the final queries, keys $(q_k^{(h)})_{k=1}^C, (k_k^{(h)})_{k=1}^C$ and values $(v_k^{(h)})_{k=1}^C \in \mathcal{S}(\mathbb{R}^{HW})$ given by

$$q_k^{(h)}(i) = \hat{q}_i^{(h)}(k) \ , \quad k_k^{(h)}(i) = \hat{k}_i^{(h)}(k) \text{ and } v_k^{(h)}(i) = \hat{v}_i^{(h)}(k) \ , \tag{19}$$

for $i = 1, ..., HW$ and $k = 1, ..., C$. To the computation of attention scores an optimizable scalar $\sigma_j \in \mathbb{R}$ is introduced, replacing by the factor $\frac{1}{\sqrt{d}}$ commonly used. The authors report that spectral density varies strongly with respect to the wavelength, which is thereby accounted for

$$A_{ij}^{(h)} = \frac{\exp\left(\sigma_j {k_i^{(h)}}^\top q_j^{(h)}\right)}{\sum_{k=1}^n \exp\left(\sigma_j {k_k^{(h)}}^\top q_j^{(h)}\right)} \tag{20}$$

for $i, j = 1, ..., d_h$. Finally the values are weighted by the attention scores

$$\hat{y}_j^{(h)} = \sum_{i=1}^{C} A_{ij}^{(h)} v_j^{(h)} \ , \tag{21}$$

Using an optimizable matrix $W \in \mathbb{R}^{C \times C}$, the heads are merged back together

$$\hat{y}_k = W[\hat{y}_k^{(1)}, ..., \hat{y}_k^{(H)}] \ , \tag{22}$$

for $k = 1, ..., HW$. One more peculiar design decision the authors made, positional encodings are added as a last step and computed using a shallow network ϕ from the value embeddings $\hat{v}_k$, that is

[1] If one represents queries, keys and values as matrices, i.e. $X_Q^{(h)} = Q_h X, X_K^{(h)} = K_h X$ and $X_V^{(h)} = V_h X$ for $h = 1, ..., H$, where $X \in \mathbb{R}^{C \times HW}$, the attention scores are computed by $A_h = \text{softmax}(\sigma_h {X_K^{(h)}}^\top X_Q^{(h)}) \in \mathbb{R}^{C \times C}$. Note, for standard self-attention they are computed by $A_h = \text{softmax}(\frac{1}{\sqrt{d_h}} X_Q^{(h)} {X_K^{(h)}}^\top) \in \mathbb{R}^{HW \times HW}$.

$$y_k = \hat{y}_k + \phi\left(\left[\hat{v}_k^{(1)}, ..., \hat{v}_k^{(H)}\right]\right) \ , \tag{23}$$

where ϕ is defined by

$$\phi = F(H \cdot W, H \cdot W) \circ \text{GELU} \circ F(H \cdot W, H \cdot W) \ .$$

The difference between spectral and standard self-attention is visualized once more in figure 15.

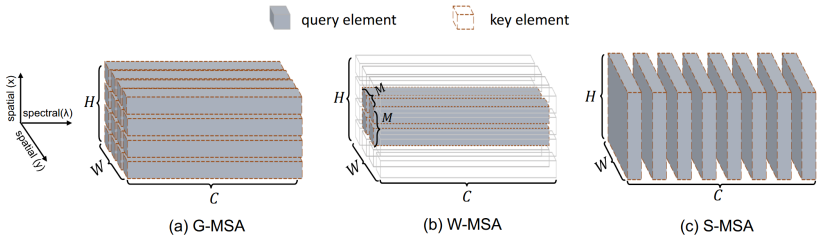

Figure 15: Image taken from [**?**], Spectral Multi-headed Self-Attention.

We capture the operation in the following definition.

Definition 15 (Spectral Multi-headed Self-Attention) *The operation defined through equations (18), (19), (20), (21), (22) and (23)*

$$\textit{S-MSA} : \mathcal{S}(\mathbb{R}^d) \to \mathcal{S}(\mathbb{R}^d) \ , \quad \textit{S-MSA}\left((x_k)_{k=1}^N\right) = (y_k)_{k=1}^N \ ,$$

is called Spectral Multi-Headed Self-Attention.

As already mentioned in the beginning, the Spectral Multi-Headed Self-Attention Block H_{SAB} is a variation of the transformer block

$$H_{SAB} = R(\phi \circ \text{LN}) \circ R(\text{S-MSA} \circ \text{LN}) \ ,$$

here $\phi : \mathbb{R}^{C \times H \times W} \to \mathbb{R}^{C \times H \times W}$ is defined as a three layer CNN

$$\phi = C(4d, d, \text{kernel-size} = 1) \circ \text{GELU} \circ C(4d, 4d, \text{kernel-size} = 3, \text{padding} = 1) \circ \text{GELU} \circ C(d, 4d, \text{kernel-size} = 1) \ .$$

We have introduced the SAB, we can now move on to describe how the architecture builds upon this module. The deep feature extraction module H_d consists of 3 cascaded Single-stage Spectral-wise Transformers (SSTs), followed by a final convolutional layer

$$H_d = C \circ H_{SST} \circ ... \circ H_{SST} \ .$$

An SST admits a U-Net like architecture, displayed in part (b) of figure 14. In order to describe an SST we are missing the downsampling and upsampling modules. The downsampling module is given by a strided convolution

$$H_{down}(d_{in}, d_{out}) = C(d_{in}, d_{out}, \text{kernel-size} = 4, \text{stride} = 2, \text{padding} = 1) \ ,$$

whereas upsampling is achieved using a transposed convolution

$$H_{up}(d_{in}, d_{out}) = C^{\top}(d_{in}, d_{out}, \text{kernel-size} = 2, \text{stride} = 2, \text{padding} = 0) \ .$$

Given initial features $F_0 \in \mathbb{R}^{31 \times H \times W}$, these are first processed using a convolutional layer

$$F_1 = C(31, 31, \text{kernel-size} = 3, \text{padding} = 1)(F_0) \ .$$

The features are then passed through the contracting path, were they are processed using a SAB followed by downsampling

$$F_i = H_{down}(2^{i-1} \cdot 31, 2^i \cdot 31) \circ H_{SAB}(2^{i-1} \cdot 31, H)(F_{i-1}) \ ,$$

for $i = 2, 3$. The feature then undergo the bottleneck, which consists of a single SAB

$$F_4 = H_{SAB}(122, H)(F_3) \ .$$

Afterwards the features are passed through the expanding path, as it is common practice in U-Net architectures, features from the same levels are concatenated

$$F_i = H_{up}(2^{7-i} \cdot 31, 2^{6-i} \cdot 31) \circ H_{SAB}(2^{7-1} \cdot 31, H) \left(\left[F_{i-1}, (F_{8-i}) \right] \right) \ ,$$

for $i = 5, 6$. The features are then passed through a final convolutional layer

$$F_1 = C(31, 31, \text{kernel-size} = 3, \text{padding} = 1)(F_0) \ .$$

This is the final output of the SST module.

1.6 Light Field Super Resolution

1.6.1 Epipolar Transformer

Liang et al. [**?**] propose the Epipolar Transformer model (EPIT), for the task of light field super resolution. The authors aim to improve the exploitation of spatial-angular correlation. To this end a 4d light field $X \in \mathbb{R}^{U \times V \times W \times H \times C}$ is mapped onto multiple 2d epipolar plane images $E^{(b)} \in \mathbb{R}^{V \times W \times C}$, for $b = 1, ..., HW$. The EPI is flattened along the first and second dimension and treated as a sequence $(x_k^{(b)})_{k=1}^{VW} \in \mathcal{S}(\mathbb{R}^C)$. Then the self-attention mechanism is employed, in order to learn the long range spatial-angular correlation along the epipolar line. The overall architecture is depicted in figure 16.

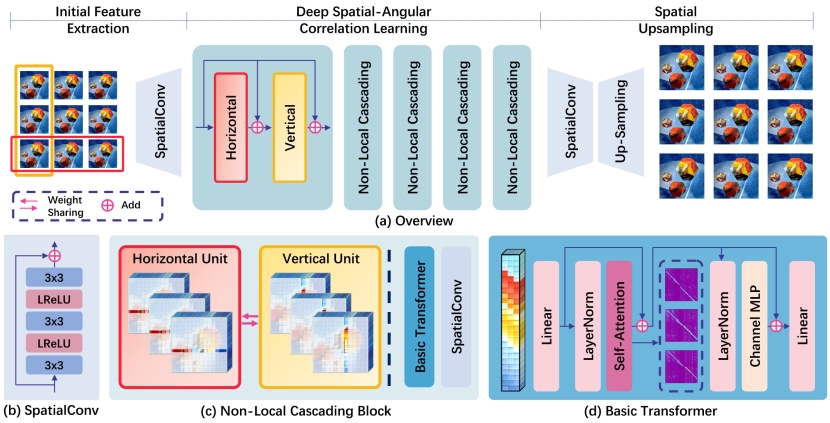

Figure 16: Image taken from [**?**], architecture of EPIT model.

The deep feature extraction model H_d proposed by Liang et al. [**?**], consists of 5 Non-Local Cascading Blocks (NLCBs)

$$H_d = H_{NLCB} \circ ... \circ H_{NLCB} \ .$$

The NLCB is composed of a horizontal and a vertical unit, to extract features along the horizontal epipolar line, and the vertical epipolar line respectively. Additionally, each unit is followed by a shallow convolutional network, both encapsuled inside of a residual connection, that is

$$H_{NCLB} = R(C_{SpaConv} \circ H_v) \circ R(C_{SpaConv} \circ H_h) \ .$$

The spatial convolution is made up of three cascaded convolutions, interspersed by leaky ReLU activation functions with a negative slope of 0.2

$$C_{SpaConv} = C \circ \mathrm{LReLU}(0.2) \circ C \circ \mathrm{LReLU}(0.2) \circ C \ ,$$

where $C = C(64, 64)$. We explain the construction of the horizontal unit H_h in greater detail, that of H_v is analog to it. Given an input $X \in \mathbb{R}^{U \times V \times H \times W \times C}$, the second and third dimension are transposed, and it is reshaped to $UH \times VW \times C$, capturing this in form of an operation gives us

$$\pi_h = \Pi^{UH \times VW \times C}_{U \times H \times V \times W \times C} \circ P_{\sigma_{SAI \to EPIH}} \ .$$

The first dimension is treated as the batch dimension, while the second as the sequence length. The sequences are then processed by a transformer block with 8 heads and a dimension of 128, formally that is

$$H_h = \pi_h^{-1} \circ T(128, 8, \Phi) \circ \pi_h \ .$$

For the mapping Φ a two layer neural network is employed

$$\Phi = F(128, 64) \circ \text{ReLU} \circ F(64, 128) \ .$$

The vertical unit is constructed analogously

$$H_v = \pi_v^{-1} \circ T(128, 8, \Phi) \circ \pi_v \ ,$$

where $\pi_v = \Pi^{VW \times UH \times C}_{V \times W \times U \times H \times C} \circ P_{\sigma_{SAI \to EPIV}}$. The weights of the transformer block T of the horizontal and the vertical unit of each NLCB are shared.

1.6.2 Disentaglement Net

Wang et al. [**?**] propose the light field disentanglement mechanism, for processing light field images. The light field disentanglement mechanism is composed of a spatial, angular and two epipolar feature extractors, these work along different 2d subspaces of the 4d light field and gather complementary information from it. The model scores the best PSNR and SSIM compared to other state of the art methods for light field super resolution. Moreover, Wang et al. [**?**] report in their ablation study, that by replacing the Disentanglement Blocks (31) by 4d convolutional residual blocks no improvement could be achieved. The architecture is visualized in figure 17.

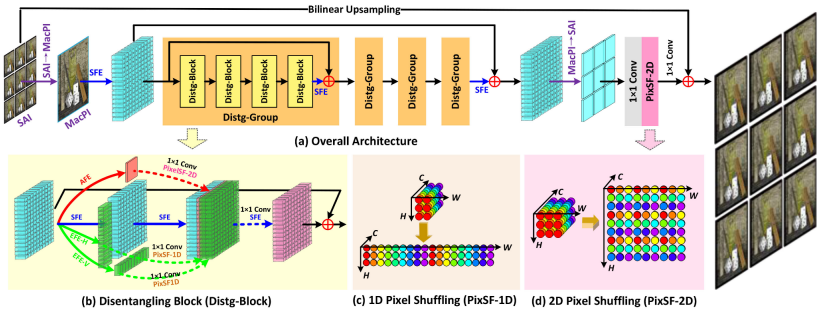

Figure 17: Image taken from [**?**].

Inputs $X \in \mathbb{R}^{C \times U \times V \times H \times W}$ are reshaped to $C \times HU \times WV$, i.e.

$$X'(c, h \cdot U + u, w \cdot V + v) = X(c, u, v, h, w) \ , \tag{24}$$

for $c = 1, ..., C$, $u = 1, ..., U$, $v = 1, ..., V$, $h = 1, ..., H$ and $w = 1, ..., W$, so that the macro pixels form tiles in a 2d plane. In the following we are going to assume that the angular dimensions are the same, i.e. $U = V = A$ for some $A \in \mathbb{N}$. The spatial feature extractor only incorporates information belonging to the same SAIs, to this end a convolution with dilation set to A is employed

$$C_{sfe} = C(64, 64, \text{kernel-size} = 3, \text{padding} = A, \text{dilation} = A, \text{stride} = 1) \ . \tag{25}$$

The spatial feature extraction module is composed of two spatial feature extractors, with the leaky ReLU activation function in between

$$H_{sfe} = C_{sfe} \circ \text{LReLU}(0.1) \circ C_{sfe} \ . \tag{26}$$

The angular feature extractor gathers information only from the same MacPi, this is achieved by using quadratic kernel of size A, i.e. the size of the MacPi itself and stride set to A

$$C_{afe} = C(64, 64, \text{kernel-size} = A, \text{padding} = 0, \text{dilation} = 1, \text{stride} = A) \ . \tag{27}$$

The feature extraction module makes use of only a single angular feature extraction, due to the stride of A the resulting feature map shrinks by a factor of A. To scale the feature map back to the same size as that produced by H_{sfe} pixel shuffling is employed

$$H_{afe} = \text{PixelShuffle2D}(A) \circ C(16, A^2 \cdot 16, 1) \circ \text{LReLU}(0.1) \circ C_{afe} \ . \tag{28}$$

Note that when MacPis are tiled in a 2d plane as done by the operation described in (24), the epipolar lines are exactly the rows and the columns. Hence for epipolar feature extraction a convolution with a kernel of shape $1 \times A^2$ is used, with a vertical stride of 1 and a horizontal stride of A, padding of $\left\lfloor \frac{A(A-1)}{2} \right\rfloor$ is added to all sides

$$C_{efe} = C(64, 64, \text{kernel-size} = (1, A^2), \text{padding} = (0, \left\lfloor \frac{A(A-1)}{2} \right\rfloor), \text{dilation} = A, \text{stride} = A) \ . \tag{29}$$

Similarly to the angular feature extraction module H_{afe} after the application of the epipolar feature extractor the feature map needs to be scaled up again

$$H_{efe} = \text{PixelShuffle1D}(A) \circ C(32, A \cdot 32, 1) \circ \text{LReLU}(0.1) \circ C_{efe} \tag{30}$$

Note that by transposing the input image, H_{efe} can be used to extract features along the vertial epipolar lines. The Disentanglement Block H_{Distg}, shown in part (b) of figure 17, forms the basic building block of the architecture proposed by Wang et al. [**?**]. The input is processed in parallel by the spatial, angular and epipolar feature extractors, the outputs are concatenated and fused by convolution with kernel size 1 followed by a spatial feature extractor

$$H_{Distg} = R(C_{sfe} \circ C(144, 64, \text{kernel-size} = 1) \circ P(H_{sfe}, H_{afe}, H_{efe}, H_{efe} \circ P_\sigma)) \tag{31}$$

where $\sigma = (1, 3, 2)$. Four cascaded disentanglement blocks followed by a spatial convolution encapsuled inside of a residual connection make up one disentanglement block

$$H_{Distg-Group} = R(C_{sfe} \circ H_{Distg} \circ ... \circ H_{Distg}) \ .$$

Finally, the deep feature extraction model is composed of 4 disentanglement groups followed by a final spatial feature extraction

$$H_d = C_{sfe} \circ H_{Distg-Group} \circ ... \circ H_{Distg-Group} \ .$$

1.6.3 Disentaglement U-Net

Chao et al. [**?**] propose a U-Net architecture for processing light field data. The architecture makes use of the disentanglement mechanism introduced by Wang et al. [**?**].

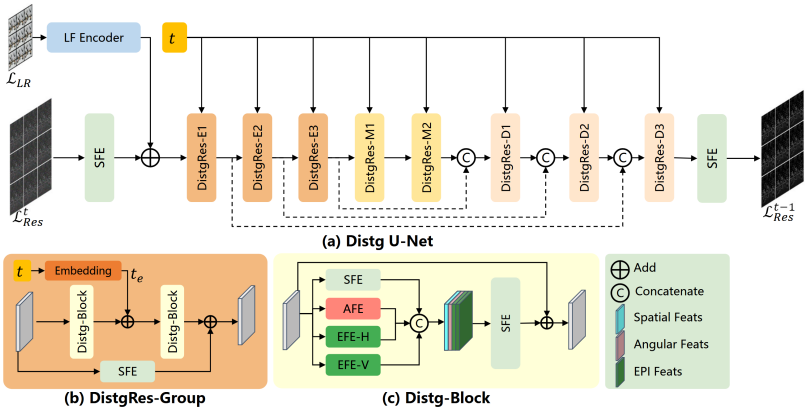

Figure 18: Image taken from [**?**], visualization of the model.

The basic building block of the network forms the disentanglement block, shown in part (c) of figure 18, it is constructed the same way as in the network described in 1.6.2 in equation 31, but the residual connection is omitted[2]

$$H_{Distg}(d_{in}, d_{out}) = C_{sfe}(d_{in}, d_{out}) \circ C(144, 64, \text{kernel-size} = 1) \circ P(H_{sfe}, H_{afe}, H_{efe}, H_{efe} \circ P_\sigma) \ ,$$

all the modules follow the same definitions as in section 1.6.2, apart from $C_{sfe}(d_{in}, d_{out})$. Different from the implementation in section 1.6.2, it might return outputs differing in channel dimension compared to its inputs

$$C_{sfe}(d_{in}, d_{out}) = C(d_{in}, d_{out}, \text{kernel-size} = 3, \text{padding} = A, \text{dilation} = A, \text{stride} = 1) \ .$$

The Disentaglement Residual Group builds upon the Distg Block

[2]Despite figure 18 (c) displaying a residual connection, it is not used in the actual implementation.

$$H_{DistResGroup}(d_{in}, d_{out}) = s\big(H_{main}(d_{in}, d_{out}), C_{sfe}(d_{in}, d_{out})\big) \ ,$$

as the channel dimension of input and output might differ, a spatial convolution is used as a replacement for a residual connection. The main branch of the group is made up of two cascaded disentanglement blocks, where in between the information of the time step of the diffusion process is inferred via sinusoidal positional embeddings

$$H_{main}(d_{in}, d_{out}) : \mathbb{R}^{d_{in}\times H\times W}\times\mathbb{N} \to \mathbb{R}^{d_{out}\times H\times W} \ , H_{main}(d_{in}, d_{out})(X, t) = H_{Distg}(d_{out}, d_{out})\big(H_{Distg}(d_{in}, d_{out})(\tag{32}$$

The mapping ϕ is a shallow neural network given by

$$\phi = F(32, 32) \circ \text{Mish} \circ F(32, 32) \circ \text{Mish} \ .$$

The addition in (**??**) is to be interpreted channelwise, to each element of the ith channel of $H_{Distg}(d_{in}, d_{out})(X)$ the ith element of $\phi \circ p_{\sin}(t)$ is added. The last pieces missing to describe the U-Net architecture are the upsampling and downsampling modules. Downsampling is achieved by performing a strided convolution

$$H_{down} = C(32, 32, \text{kernel-size} = 3, \text{stride} = 2, \text{padding} = 1) \ .$$

On the other side, upsampling is done by using a transposed convolution

$$H_{up} = C^{\top}(32, 32, \text{kernel-size} = 4, \text{stride} = 2, \text{padding} = 1) \ .$$

Given initial features $F_0 \in \mathbb{R}^{32\times H\times W}$, the features in the contracting path are processed by two sequential disentanglement residual groups, followed by a downsampling operation

$$F_i = H_{down} \circ H_{DistResGroup}(32, 32) \circ H_{DistResGroup}(32, 32)(F_{i-1}) \ ,$$

for $i = 1, 2$. The last level of the contracting path ($i = 3$) as well as the bottleneck ($i = 4, 5$) omit the downsampling

$$F_i = H_{DistResGroup}(32, 32) \circ H_{DistResGroup}(32, 32)(F_{i-1}) \ .$$

As usual for U-Net architectures, features from the same level are concatenated in the expanding path

$$F_i = H_{up} \circ H_{DistResGroup}(32, 32) \circ H_{DistResGroup}(64, 32) \left(\left[F_{i-1}, F_{9-i}\right]\right) \ ,$$

for $i = 7, 8$. Analogously to the contracting path, the upsampling operation is omitted for the last stage of the expanding path

$$F_9 = H_{DistResGroup}(32, 32) \circ H_{DistResGroup}(64, 32) \left(\left[F_8, F_1\right]\right) \ .$$

2 Training

2.1 Preprocessing the Data

2.2 Training Methods

2.2.1 Single Image Super Resolution

2.2.2 Light Field-, Hyperspectral Image- and Spectral Super Resolution Methods

2.2.3 Diffusion Models