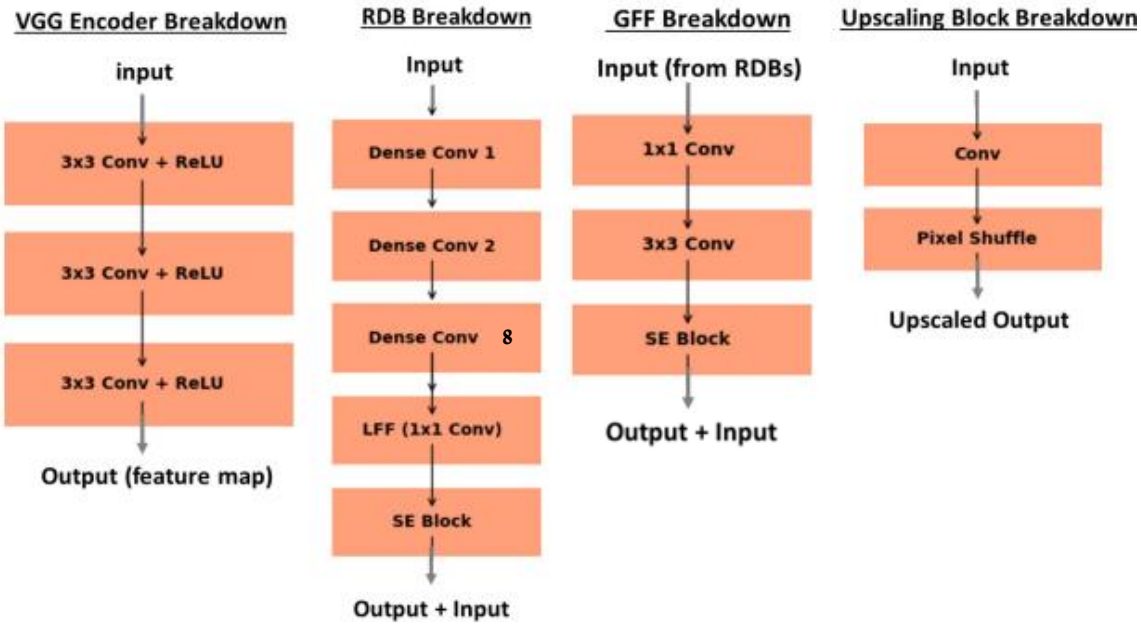
SISR project summary

- Reconstruct a high-definition image from a single LR counterpart.
- This model provides good performance across various **scaling factors, even ones not trained on (also real valued ones).**
- My changes were to the encoder & decoder architecture with emphasis on:
    - Improving the receptive field.
    - Improving interactions between the branches of the decoder.
    - Robustness.
- **Original Encoder: (produces deep feature map)**
    - Original is of **RDN** architecture (residual dense network) followed by local feature fusion (LFF) and global feature fusion (GFF). The RDN performs features extraction to produce feature maps.
    - Shallow feature extraction-> RDBs with LFF -> GFF
    - Key importance of RDN is the use of dense blocks which utilize dense connections between all layers within the block as well as local and global residual learning (where the **output is added to the input**) to allow for maximum gradient flow through the net and overcome vanishing gradient. Instead of learning direct mapping $y = f(x)$, residual learning formulates the target as $y = x + f(x)$ where f is the residual learnt.

- **Modified Encoder**



- Changed to a **VGG** inspired block for the initial feature extraction, then used the RDB blocks, with the addition of **SE** blocks (squeeze and excitation for channel-wise feature attention).
- Used **12 RDB blocks**. Outputs of each RDB block are appended to an output array.
- Each RDB block performs 8 sequential convolutional layers, each followed by a RELU and **concatenation (dense connection)** of the output and the input.
- **LFF** performs 1x1 convolution across the channels of the output from the convolutions.
- **SE blocks** include two parts:
    - Squeeze part: global average pooling with an output size of 1 (for each channel) – channel-wise averages. Meaning, a scalar representing the information averaged across the channel, for each channel, organized into a vector $\boldsymbol{z}$.
    - Excitation part: FC->RELU->FC->Sigmoid. Designed to learn the attention weight for each channel.

      $$s = \sigma\big(W_2 \cdot RELU(W_1 \cdot z)\big)$$

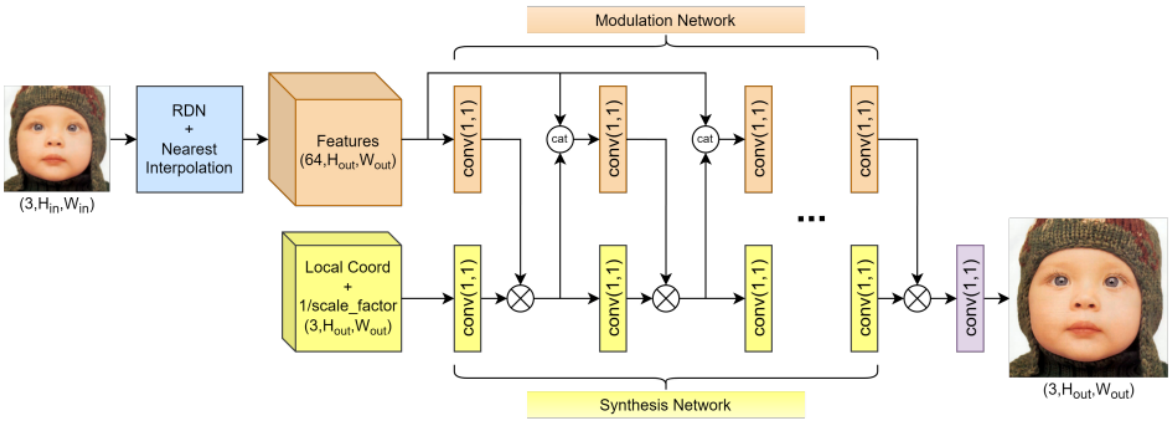
      Where W_1, W_2 are the weights of the first and second FC layers respectively.

      z is the global averaged vector across all channels.

      s is the channel weights vectors (where each element $\epsilon R[0,1]$)
- The output of the SE block is the input to the block where each channel is multiplied (element wise) by the corresponding channel weight.

- **GFF** includes 1x1 convolution across the concatenated output features of the 12 RDB blocks into a single tensor followed by 2d convolution and another SE block.
- The output of the GFF is added (**residual connection**) to its input.
- Upscale block using pixelshuffle function (involves learning 2dconv whose output is bigger by a scale factor **2, then pixelshuffle organizes the data to be the width*factor and height*factor of the input. Used factor of 2.

- **Decoder (dual interactive implicit decoder):**



  - The implicit decoder predicts the signal (i.e., the (r, g, b) values) at any query location within the image space, conditioned on the associated features provided by the encoder.
  - Receives the feature maps produced by the encoder alongside a grid out the desired output image's size, the scale factor and positional features. (called for each pixel in the output grid).
  - There is a **decoupling** of the content features & positional features in the decoder. Where the content features flow through the modulation branch and the positional features flow through the synthesis branch.
  - The decoder is built of 2 branches: modulation and synthesis. Each is implemented by a MLP of 256 neurons and 4 layers, interaction is allowed between the decoder's branches as shown. Each layer is implemented using convolution with 256 kernels of size 1.
  - At the end of the end dimensionality reduction is performed (1x1 convolution) to the desired 3 channel output (RGB).
  - Each layer in the modulation branch makes use of RELU activation while each layer in the synthesis branch makes use of a SIN activation.
  - **Net's uniqueness**: the net is learning implicit neural representation (A.K.A coordinate based representation) for the pixel level in the image, such that the output image synthesis of each pixel is done using the features supplied into the decoder around the specific desired pixel query location. Meaning, it implicitly models the image as a continuous function that can be queried at any coordinate, enabling resolution independent representation (allowing for arbitrary scale factors).

Modified decoder:

- The decoder receives output of encoder: 64, in channels, built of 4 hidden layers of size 256 each.
- Used **adaptive modulation** (for more context aware synthesis). Done by introducing a new branch **(M branch)** for the decoder allowing the model to dynamically scale the activations of the synthesis branch based on the activations of the **modulation branch (K branch)**.
- The learnt modulation factors are applied with element-wise multiplication to the output activations of **synthesis branch (Q branch)** before passing them through the final layers of the decoder.
- This allows the model to dynamically emphasize certain features in the synthesis branch based on the modulation branch.
- The decoder implementation step in the hidden layers i looks like so:

$$k = K[i]\big(cat(q, x)\big)$$
$$q = k \cdot Q[i](q)$$
$$m = m + M[i](k)$$

- As one can see, the m branch is affected by the activations of each previous modulation (k branch) output. It is then applied to the final layer of the q (synthesis) branch before applying the last layer, by element wise multiplication:

$$Output = LastLayer(q \cdot m)$$

  $LastLayer$ is implemented by a 1x1 convolution taking in the last number of hidden dimensions as input and outputs 3 RGB channels.
- Each Q,K,M branch's layer is built of: conv2d->Silu activation->dropout (if applied).
- **Dropout** regularization (tested several probabilities: 0,0.1,0.3,0.5). After increasing the model size, we wanted to test if dropout is needed to preserve the model robustness and prevent overfitting (although in the end it wasn't needed since the net went through relatively small training procedure).
- **Dilated convolution,** a task which was suggested by the article's authors was to increase the model's receptive field. Using dilated kernels allows the decoder to grasp broader contextual information of the image. Factors used: (1,2,4,8). It was used in the convolutions performed in the different branches.
- **SILU** activation. SiLU(x)=x·σ(x), is a smooth differentiable function, which helps convergences effectively. Tends to perform well in deep architectures as it mitigates issues like vanishing gradient.

    - **Trained and tested** a bunch of different combinations of these implementations and changes.
- **Evaluation metrics:**
    - PSNR - It quantifies the level of noise or distortion present in the super-resolved image by calculating the ratio of the peak signal (maximum possible pixel value) to the mean squared error (MSE) between the super-resolved and ground truth images
    - SSIM - considers three components of image quality: luminance (brightness), contrast, and structure, and computes a similarity index based on these components.
- **Data:**
    - Training and validation: DIV2K (consists of 1000 HR images).
    - Test on set5, set14, urban100, b100.
    - For training, we took the HR images, and down-sampled them by different scale factors, then applied random rotations. Each minibatch of 4 HR images produces 12 pairs of LR and HR images for training. The recovery error is calculated and trained upon between the HR image and the corresponding SR image.
- **Train**
    - Training procedure is similar to the original article with a few changes made due to memory and computation power limitations. For each scale $factors \in \{2, 4\}$ and HR image in the minibatch, randomly cropped a 36×36 patch and down sample it using bicubic interpolation. Then, randomly applied horizontal, vertical, and\or diagonal flips, each with a probability of 0.5. The training was done where each epoch is a full pass through 800 HR images in the DIV2K training set times the number of trainsets repeat defined. The initial models were trained for approximately 5 epochs (changing depends on memory limitations) with 10 trainsets repeats (8000 images each epoch), for a total of about 40,000 training images. Later we switched to training with 5 trainsets repeats (4000 images each epoch) for 5 to 20 epochs (depending on the model trained), for a total of about 20,000 to 80,000 training images (again, depending on the model). Training was done using the Adam optimizer with the default hyper parameters provided by PyTorch. The learning rate was initialized at 10 to 4. We have used the L1 loss to train our network. Each model was evaluated at the end of each epoch using an evaluation batch size of 3000.

- **Results:**

Figure 6: model H results, LR image on the left, SRx1.5 on the middle and SRx2 on the right

-

Figure 4: Results of super resolved image with upfront scale 4 (SRx4) and gradually scaled image by factors of 1.25, then 1.6 and then 2 (SRx1.25x1.6x2)

-

	Upfront (x4)	Gradual (x1.25x1.6x2)
PSNR	21.0311	17.8846
SSIM	0.6638	0.5848

Table 1: Evaluation of super resolved image with upfront scale 4 (SRx4) and gradually scaled image by factors of 1.25, then 1.6 and then 2 (SRx1.25x1.6x2).

Model	SSIM 3.14	SSIM 4	SSIM 8	PSNR 3.14	PSNR 4	PSNR 8
A	0.5707	0.5028	0.3634	23.8673	23.1125	21.3797
B	0.6188	0.5643	0.4096	23.0040	22.5127	20.5930
C	0.6733	0.6078	0.4721	23.2912	22.6600	21.1770
D	0.6819	0.6031	0.4577	23.5244	22.6884	20.9663
E	0.6594	0.5960	0.4554	23.0031	22.4609	21.0288
F	0.6691	0.6044	0.4597	23.4170	22.7795	21.1399
G	0.7093	0.6466	0.4826	24.6562	23.7387	21.7813
H	0.7369	0.6712	0.5240	26.0928	25.0048	22.6406

Table 2: benchmark results on B100 dataset