Deep Learning – Assignment 4
Image Super-Resolution

Submitted by: Itay Bouganim, 305278384

Problem Statement

The purpose of this assignment is to get familiar with construction and training of fully convolutional networks.

We will specifically use the task of image super-resolution, and we'll construct several different architectures and compare the results achieved by each of them.

1. **Step 1 - Loading, Preprocessing and Helper functions**
   At first step we will load a small portion from our image data in order to experiment with different type of network compositions (Residual, Dilated etc.).

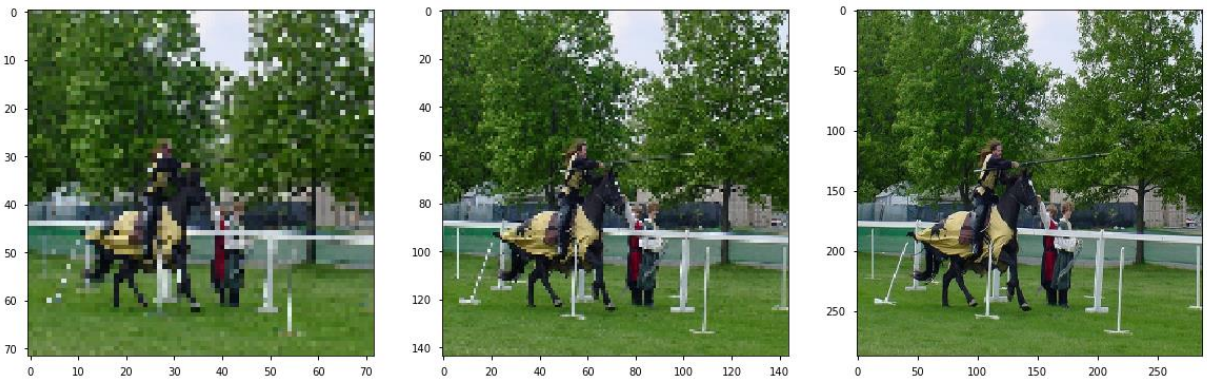
Data preprocessing

The data preprocessing steps will include loading the images using OpenCV in an RGB format (not BGR).
We will normalize the pixel values to range between 0 and 1 for faster learning process.
Next we will scale our training images to 72x72x3 dimensions.
For our output we will use two different scales: x2 and x4 from the 72x72x3 X image sizes.
We will get:

```
Image downscale dims: (72, 72)
Image upsacle x2 dims: (144, 144)
Image upsacle x4 dims: (288, 288)
```

Initially we will load only 100 images from out training set.
As validation we will split 20% from our training data.

```
(X: (100, 72, 72, 3), y_mid: (100, 144, 144, 3), y_large: (100, 288, 288, 3))
```



Original images sample scaled (left to right) 72x72, 144x144, 288x288

Training Callbacks

In order to better visualize our results we will use two custom callbacks:
SavePredictionCallback – will be used to save predicted images for each epoch in the model training process. The saved images will be from a predefined image set (3 images).
PlotPredictionCallback– will be used to predict and plot a specific image output in the training process to see a live preview of our model progress mid training process.

Custom Metrics and Loss functions

During our experiments we are going to measure the progress using the known **MSE and MAE known metrics with the additional PSNR and SSIM metrics.**

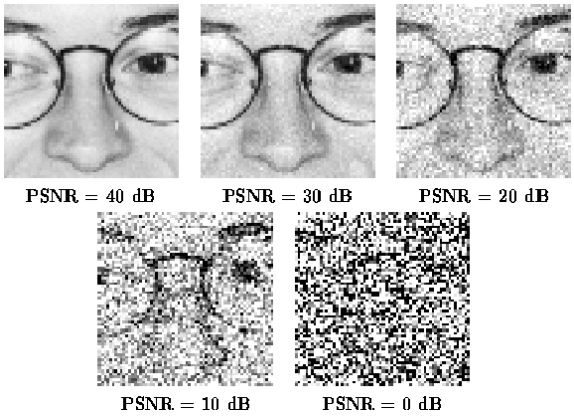
PSNR and SSIM Metrics Explanations

PSNR Metric

PSNR is most easily defined via the mean squared error (MSE).
eak signal-to-noise ratio (PSNR) is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.
Peak signal-to-noise ratio definition (PSNR) is most commonly used as a quality estimation for the loss of quality through different codecs and image compression where the signal is the original image and the noise is error created by compressing the image.
PSNR is very common for evaluating image enhancement techniques, such as Super resolution where the signal is the original/ground truth image and the noise is the error not recovered by the model.
Although PSNR is a logarithm based metric, it is based on the MSE.

PSNR Loss

Since higher PSNR metric value results in less noisy image, we will want to minimize it as a loss function.
Therefore the loss function will evaluate the negative value of the PSNR metric and try to minimize it.

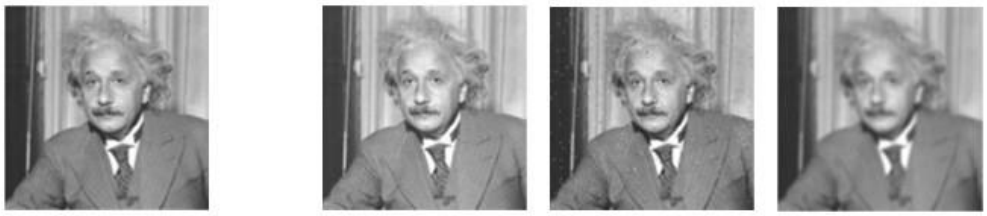


$$
\begin{aligned}
PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\
&= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\
&= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE)
\end{aligned}
$$

SSIM Metric

SSIM (structural similarity index measure) is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms.
The difference with other techniques such as MSE or PSNR is that these approaches estimate absolute errors.
SSIM is based on visible structures in the image, Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close.
These dependencies carry important information about the structure of the objects in the visual scene. Luminance masking is a phenomenon whereby image distortions (in this context) tend to be less visible in bright regions, while contrast masking is a phenomenon whereby distortions become less visible where there is significant activity or "texture" in the image.

The usage of SSIM for image enhancement evaluation came about as for some researchers PSNR is no longer regarded as a reliable indicator of image quality degradation. It is a perceptual metric that quantifies image quality degradation caused by processing.



SSIM Loss

Since in the SSIM metric 1.0 represents the best value and 0.0 the worst, we will want out loss function to converge to 1.0.
Therefore we will use 1 - SSIM as our loss function.

PSNR and SSIM Loss

Since SSIM does not account for multiple color channels and only accounts for luminance, contrast, and structure, we will combine it with the PSNR loss for color correction. Another loss function we can use is the combination of PSNR with SSIM.
Since PSNR is unbounded and in SSIM best is 1.0 we will use -PSNR - SSIM as our loss function to try and maximize both.

MSE and SSIM Loss

Since SSIM does not account for multiple color channels and only accounts for luminance, contrast, and structure, we will combine it with the MSE loss for color correction. Another loss function we can use is the combination of MSE with SSIM.
Since in MSE best is 0.0 and in SSIM best is 1.0 we will use MSE - SSIM as our loss function to try and minimize MSE and maximize SSIM.

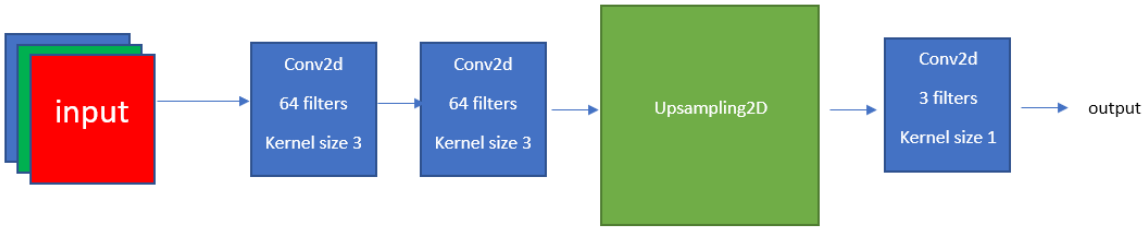
In the following section we are going to perform model training experiments on different architectures on a small part of the data.

Training Experimental Models

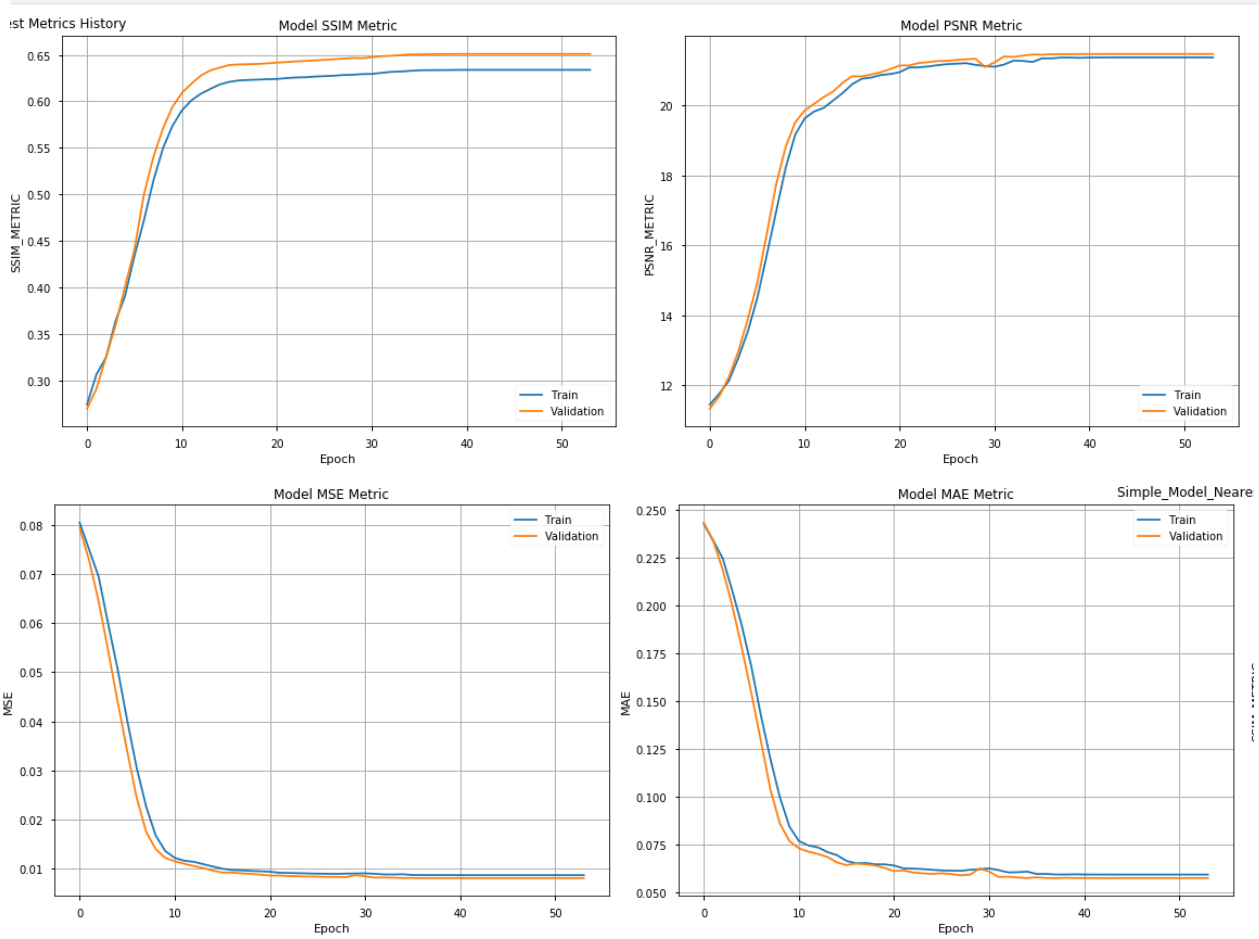
In the following steps we will train our models, for better quality images.
For GIF images of the training process please refer to the attached Jupyter notebook or to the attached ‘gifs’ zip file.

Step 2 - Create an initial fully convolutional model

At first we will use UpSampling2D layer with ‘nearest’ interpolation.

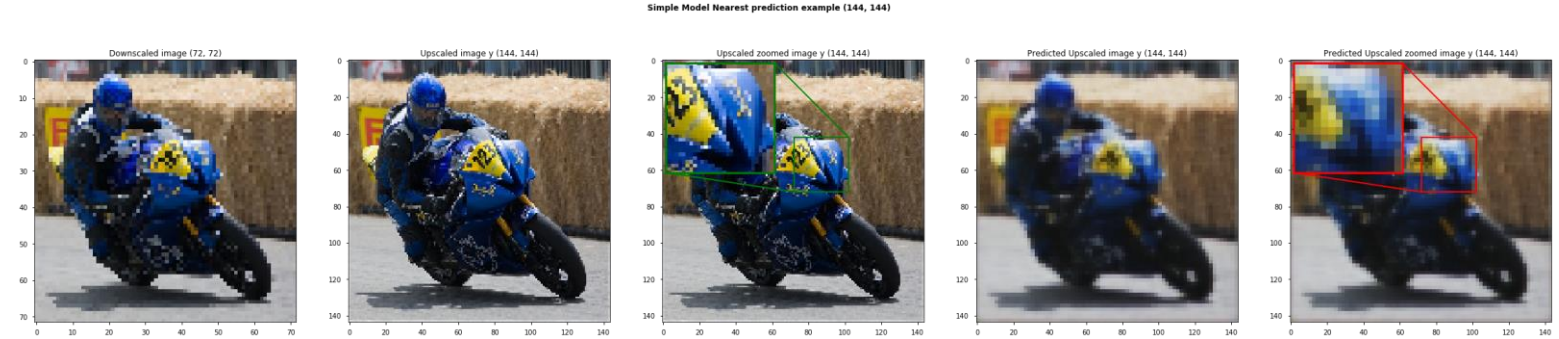



```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None, None, 3)]   0
_________________________________________________________________
conv2d (Conv2D)              (None, None, None, 64)    1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, None, None, 64)    36928
_________________________________________________________________
up_sampling2d (UpSampling2D) (None, None, None, 64)    0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, None, None, 3)     195
=================================================================
Total params: 38,915
Trainable params: 38,915
Non-trainable params: 0
```



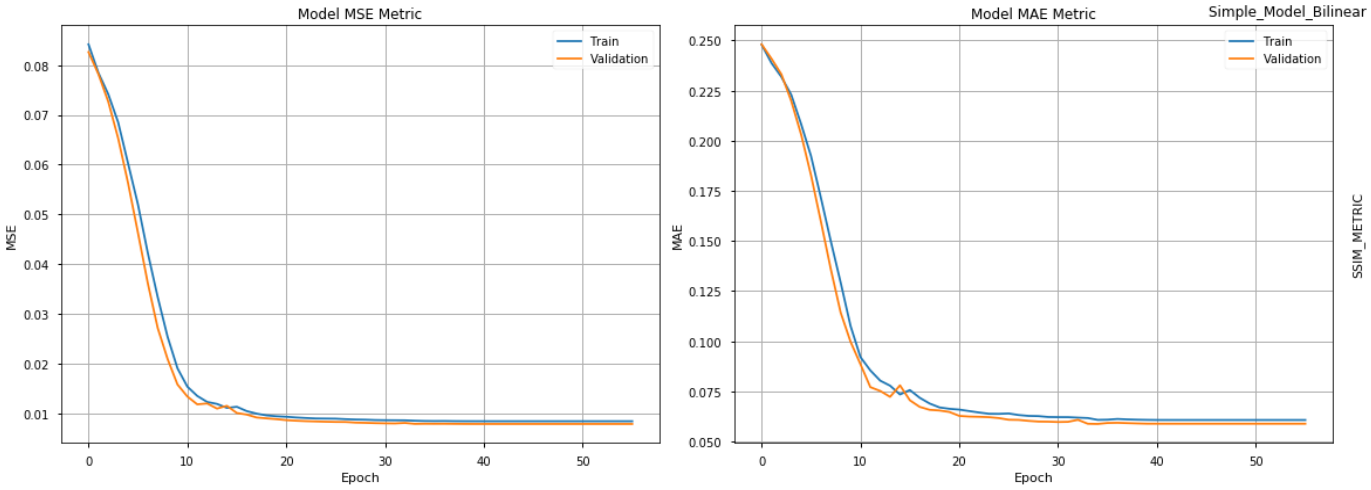
Mean Model Statistics:

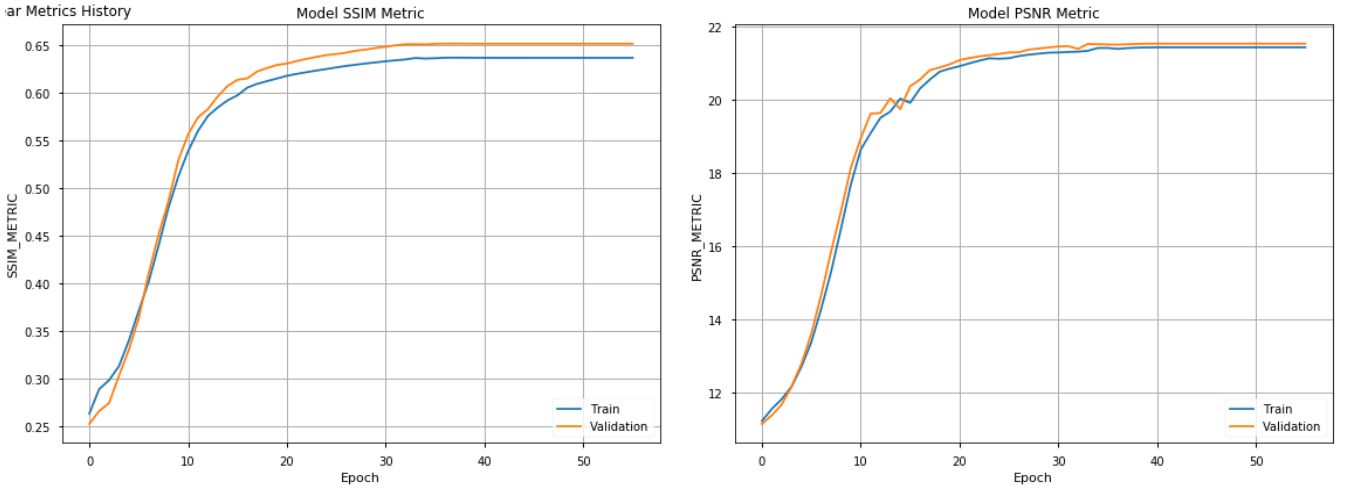
	MSE	MAE	PSNR	SSIM
0	0.008042	0.057368	21.477512	0.650988



Highest MSE metric predicted images:

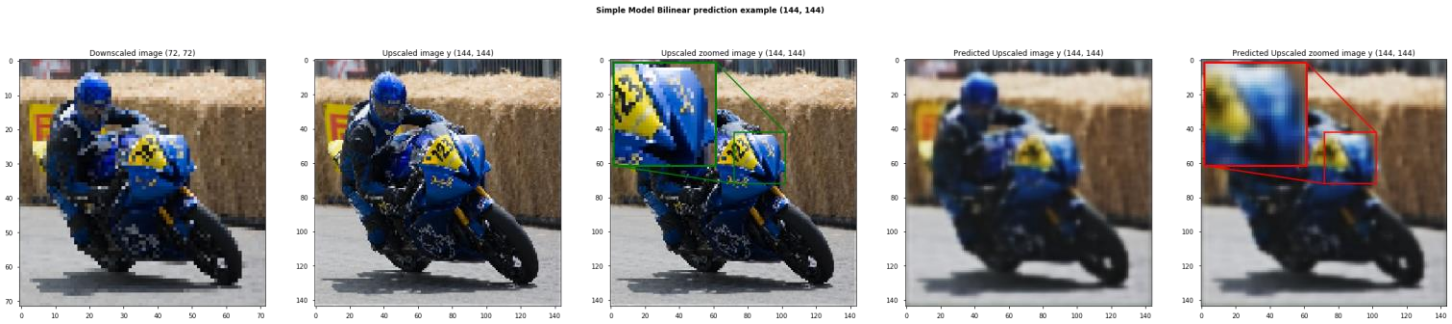
We can see that the results achieved are very pixilated and blurry.
We will try training the same model with ‘Bilinear’ interpolation method.





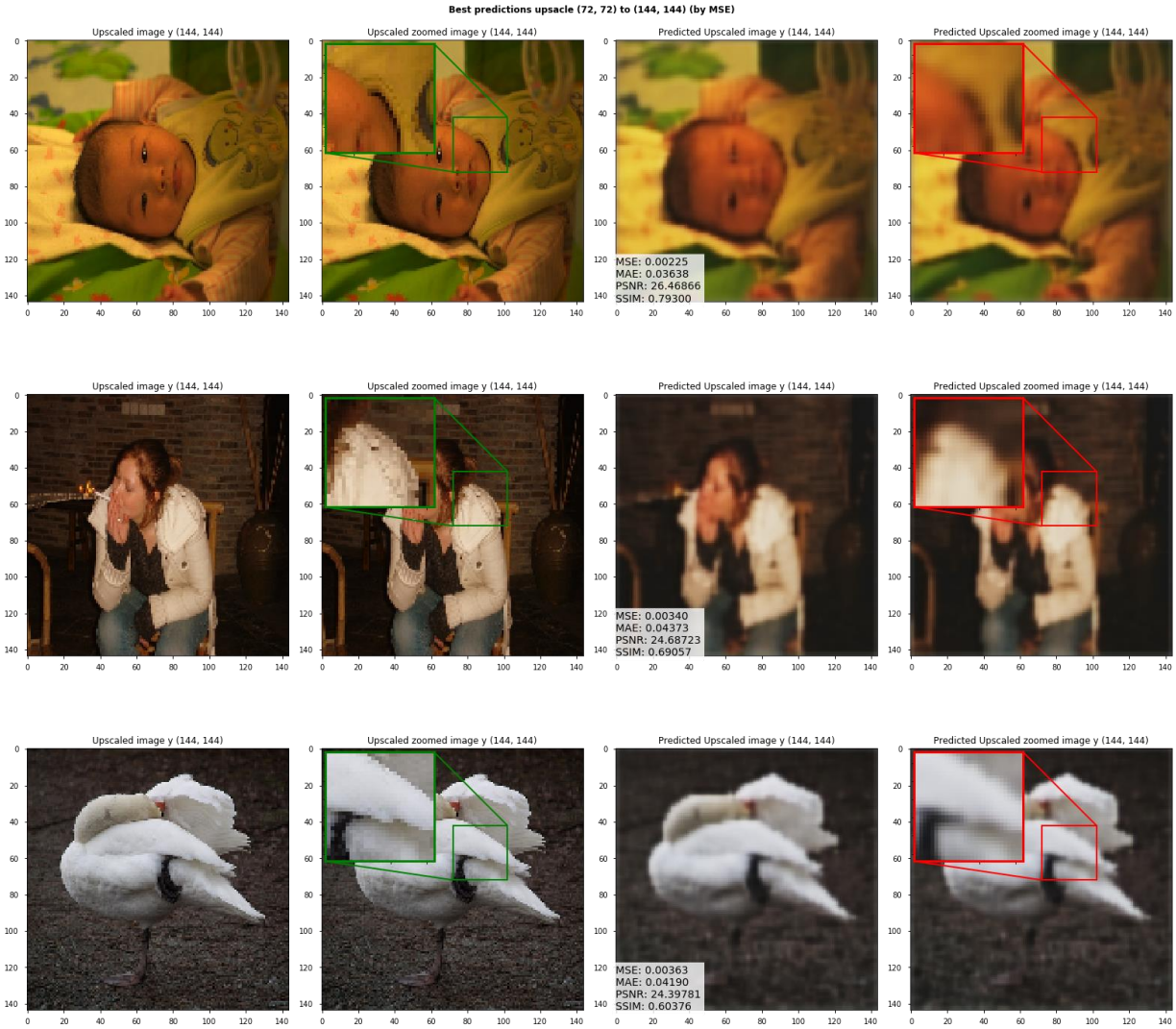
Mean Model Statistics:

	MSE	MAE	PSNR	SSIM
0	0.00791	0.058852	21.545467	0.651205



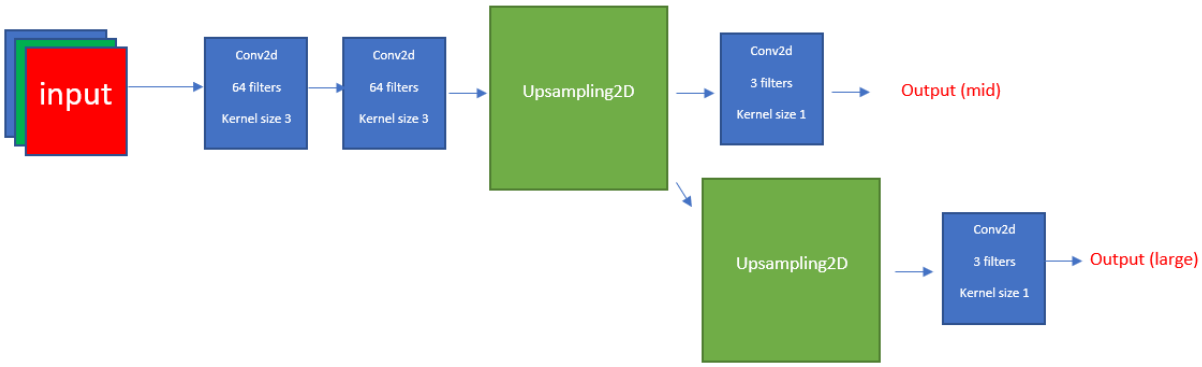
Highest MSE metric predicted images:

Images are still blurry but are less pixilated than the ‘nearest’ interpolation variation.

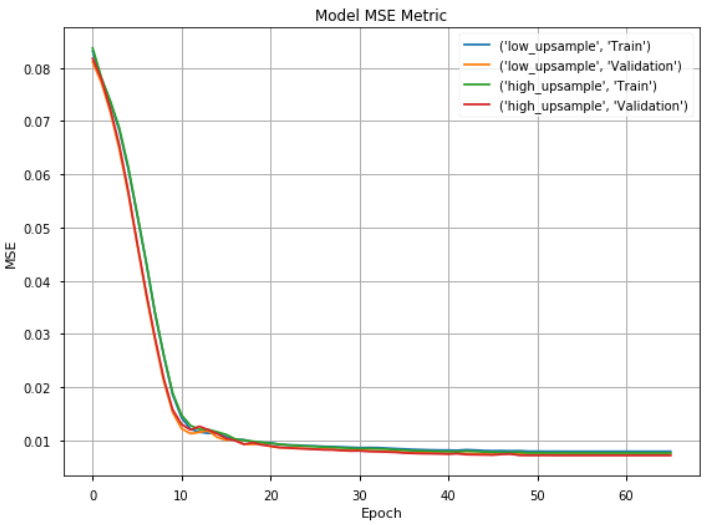


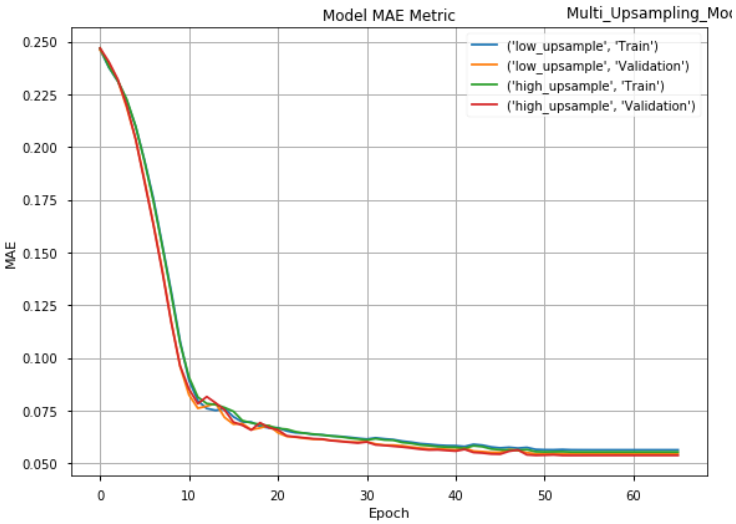
Step 3 - Add another upsampling layer and output (144, 144) upsample and (288, 288) upsample

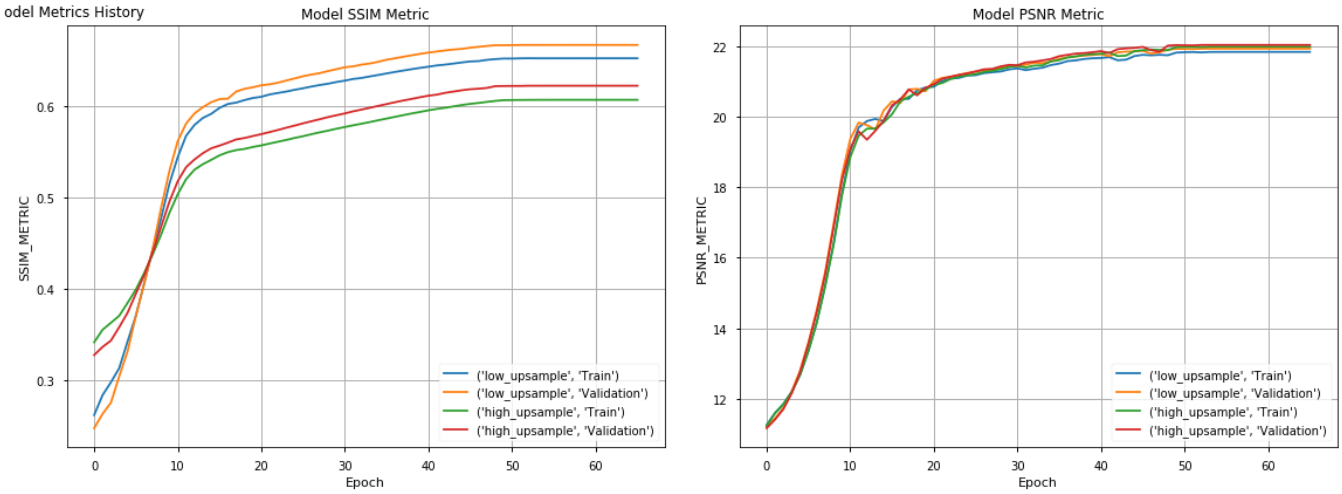
Add block to your model so that we'll have both 144x144x3 output along with 288x288x3 output as follows:




```
Model: "functional_39"
_____
___
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
===
input_20 (InputLayer)           [(None, None, None,  0
_____
___
conv2d_32 (Conv2D)              (None, None, None, 6 1792        input_20[0][0]
_____
___
conv2d_33 (Conv2D)              (None, None, None, 6 36928       conv2d_32[0][0]
_____
___
up_sampling2d_14 (UpSampling2D) (None, None, None, 6 0           conv2d_33[0][0]
_____
___
up_sampling2d_15 (UpSampling2D) (None, None, None, 6 0           up_sampling2d_14[0][0]
_____
___
low_upsample (Conv2D)           (None, None, None, 3 195         up_sampling2d_14[0][0]
_____
___
high_upsample (Conv2D)          (None, None, None, 3 195         up_sampling2d_15[0][0]
==================================================================================================
===
Total params: 39,110
Trainable params: 39,110
Non-trainable params: 0
_____
___
```







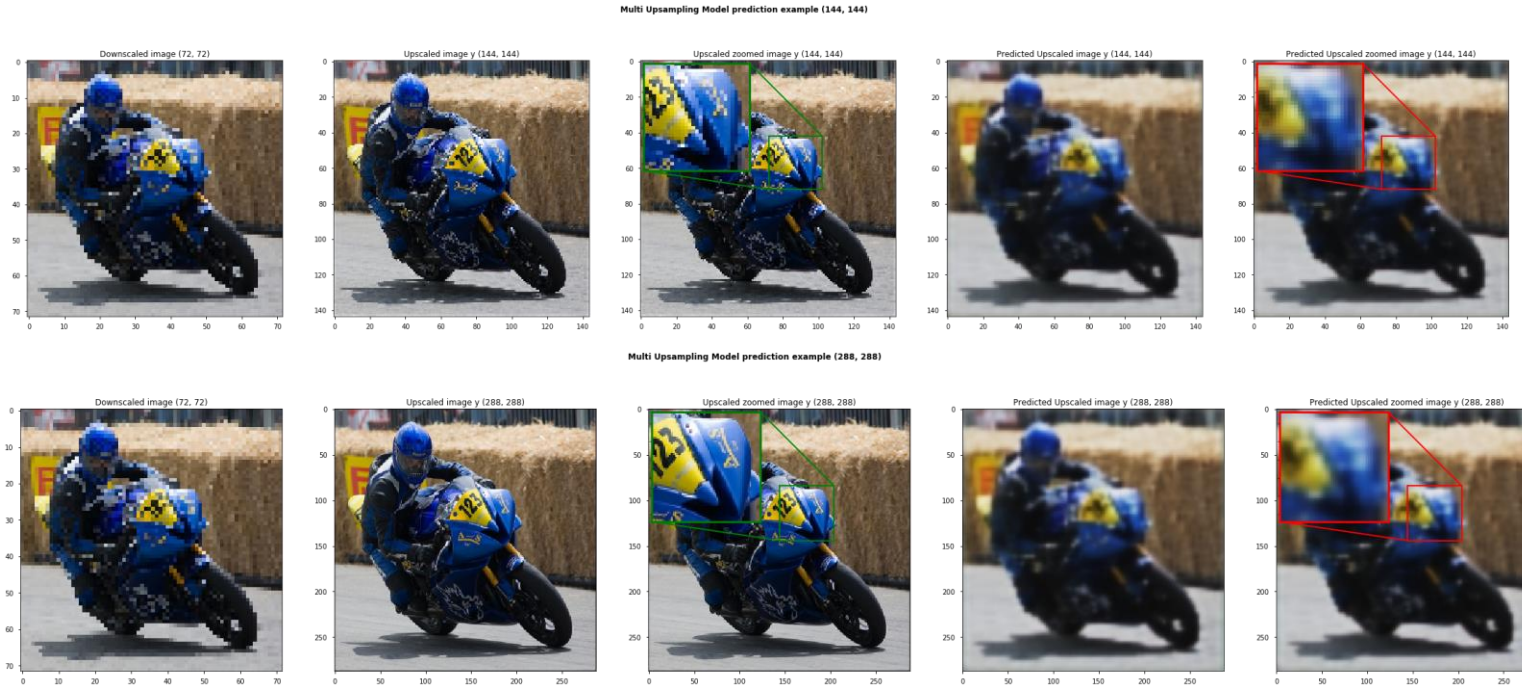
Mean Model Statistics:

`multi_upsampling_modelr Mean Validation Metrics (144, 144)`

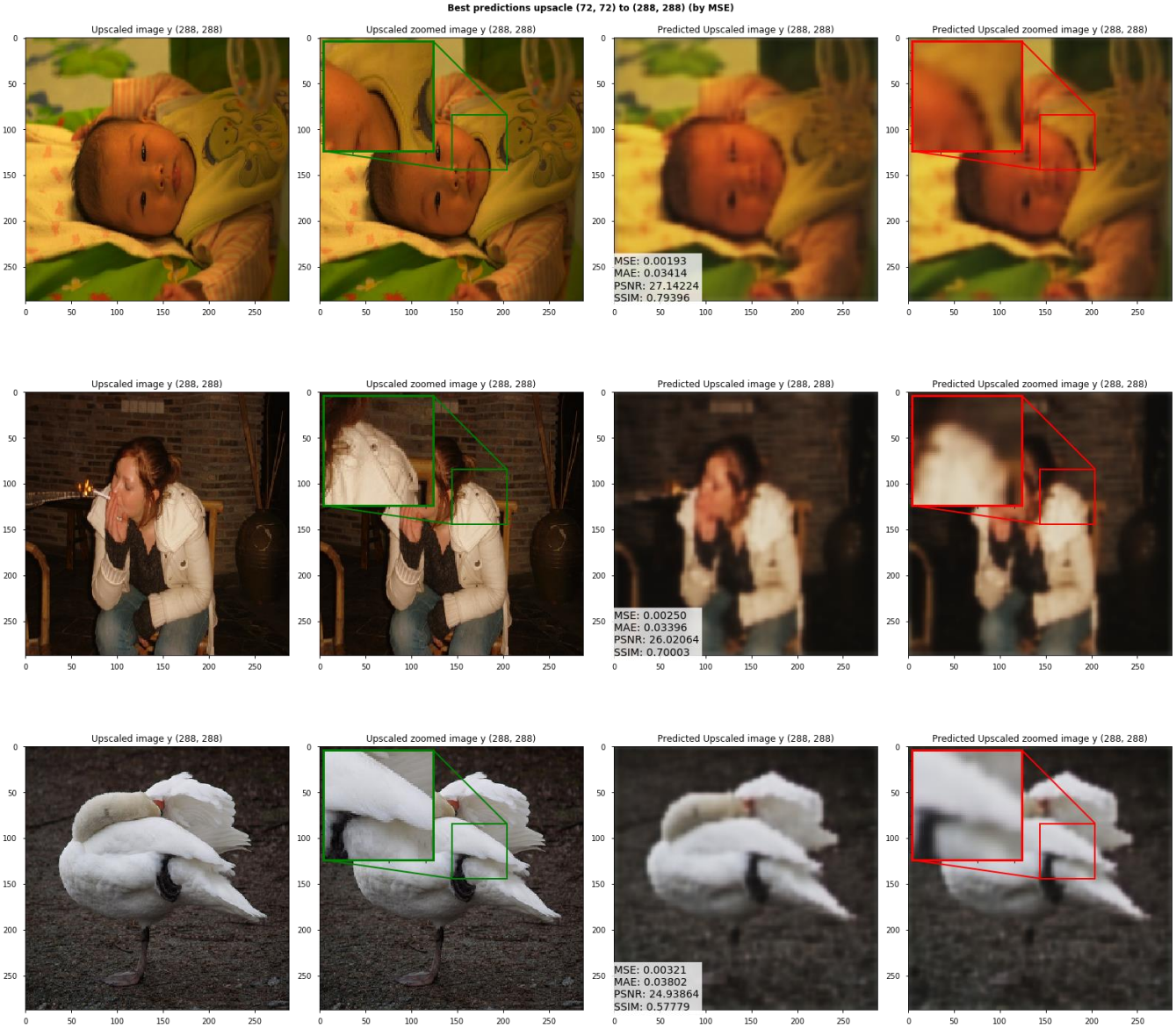
	MSE	MAE	PSNR	SSIM
0	0.007371	0.054544	21.931559	0.667144

`multi_upsampling_modelr Mean Validation Metrics (288, 288)`

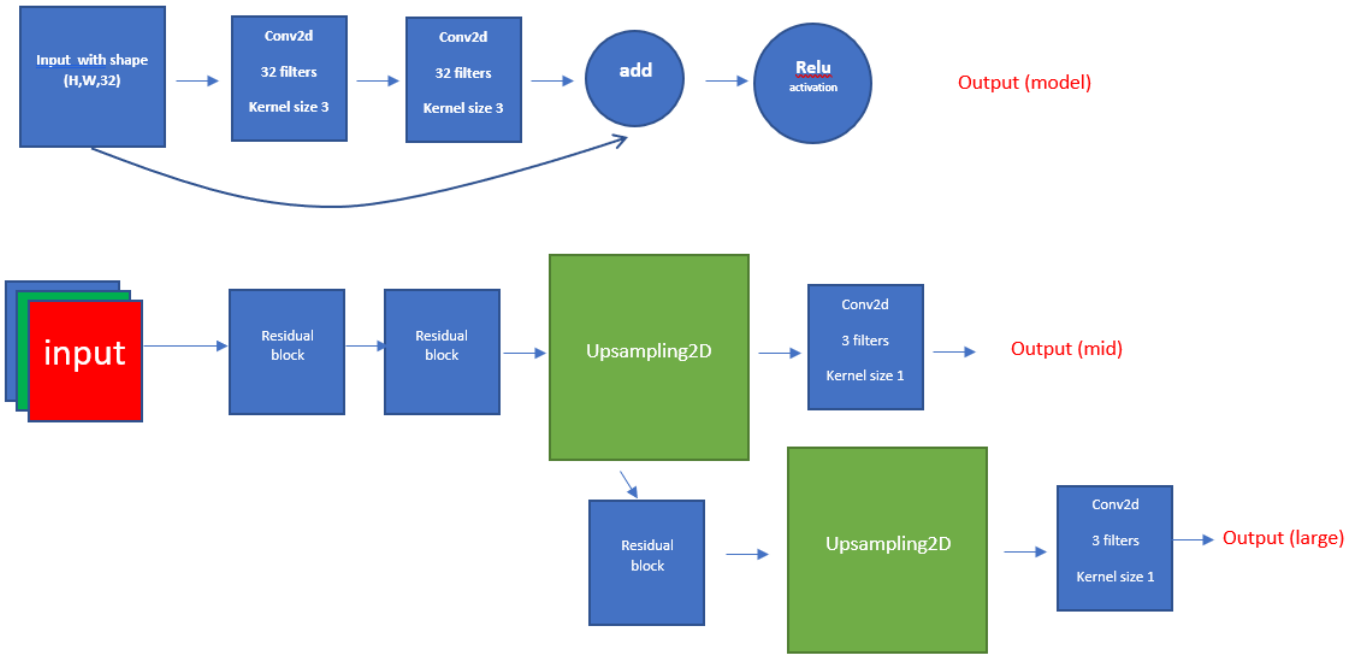
	MSE	MAE	PSNR	SSIM
0	0.007163	0.05371	22.037392	0.622454



Highest MSE metric predicted images:



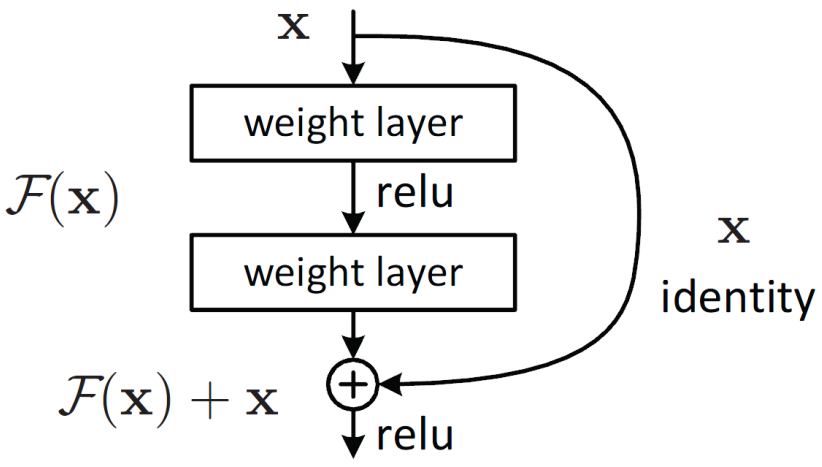
Step 4 - Add residual blocks into the process



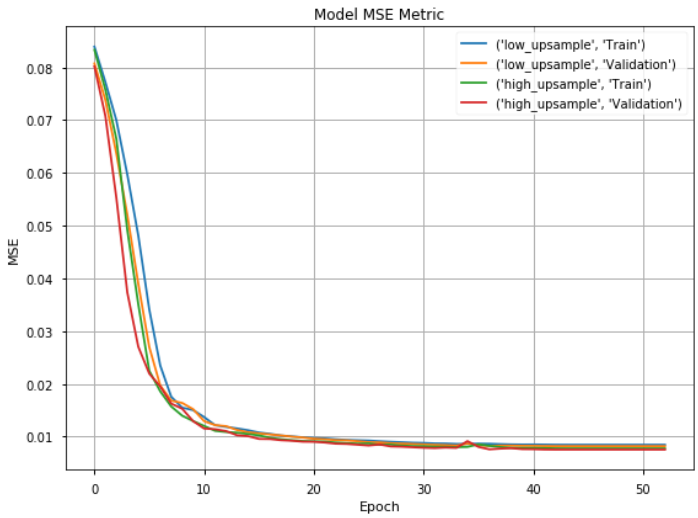
Residual blocks

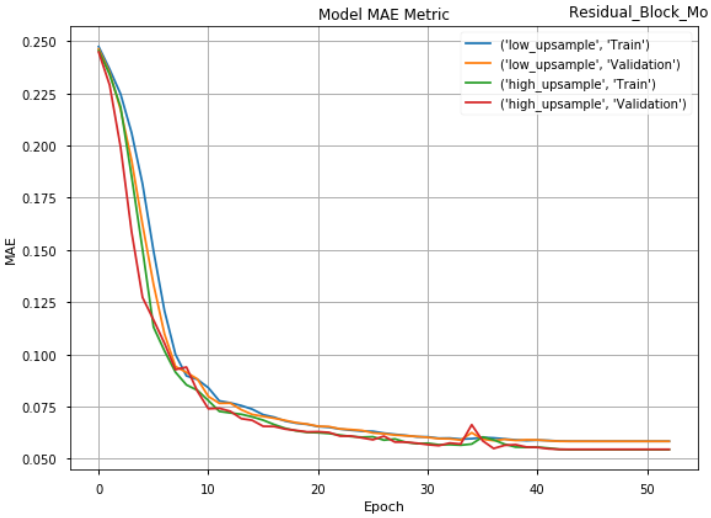
Residual Blocks are skip-connection blocks that learn residual functions with reference to the layer inputs, instead of learning unreferenced functions.
They were introduced as part of the ResNet architecture.

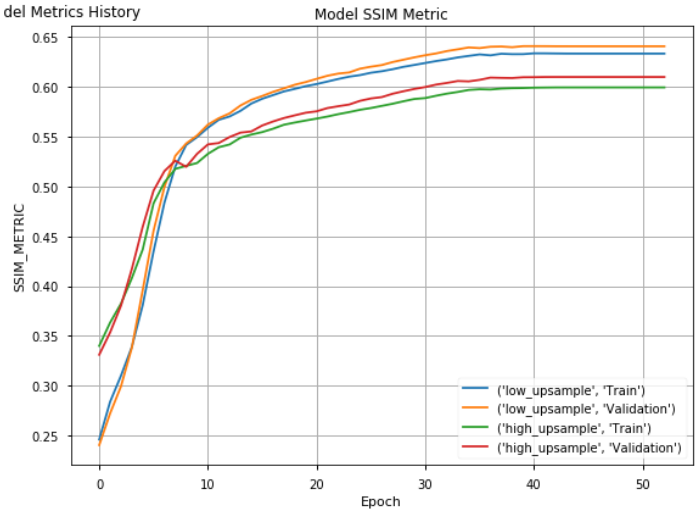
The intuition is that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping.
To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers.
Having skip connections allows the network to more easily learn identity-like mappings.

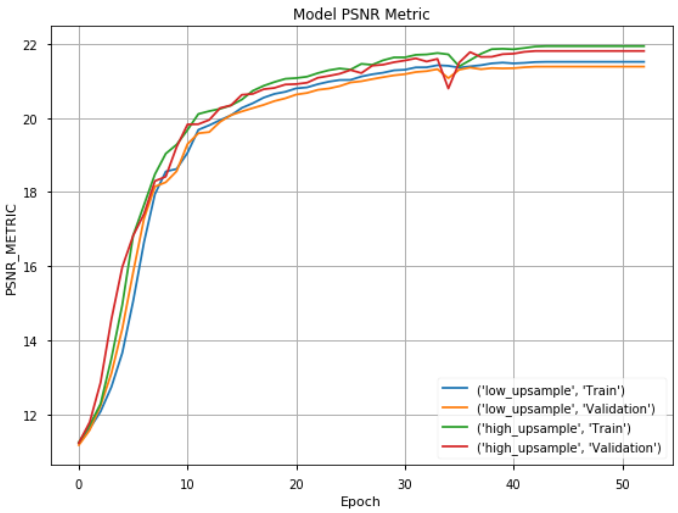


```
Model: "functional_49"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_22 (InputLayer)        [(None, None, None,  0
_________________________________________________________________________________________________
conv2d_36 (Conv2D)           (None, None, None, 3 128         input_22[0][0]
_________________________________________________________________________________________________
functional_43 (Functional)   (None, None, None, 3 18496       conv2d_36[0][0]
_________________________________________________________________________________________________
functional_45 (Functional)   (None, None, None, 3 18496       functional_43[0][0]
_________________________________________________________________________________________________
up_sampling2d_18 (UpSampling2D) (None, None, None, 3 0           functional_45[0][0]
_________________________________________________________________________________________________
functional_47 (Functional)   (None, None, None, 3 18496       up_sampling2d_18[0][0]
_________________________________________________________________________________________________
up_sampling2d_19 (UpSampling2D) (None, None, None, 3 0           functional_47[0][0]
_________________________________________________________________________________________________
low_upsample (Conv2D)        (None, None, None, 3 99          up_sampling2d_18[0][0]
_________________________________________________________________________________________________
high_upsample (Conv2D)       (None, None, None, 3 99          up_sampling2d_19[0][0]
=================================================================================================
Total params: 55,814
Trainable params: 55,814
Non-trainable params: 0
_________________________________________________________________________________________________
```









Mean Model Statistics:

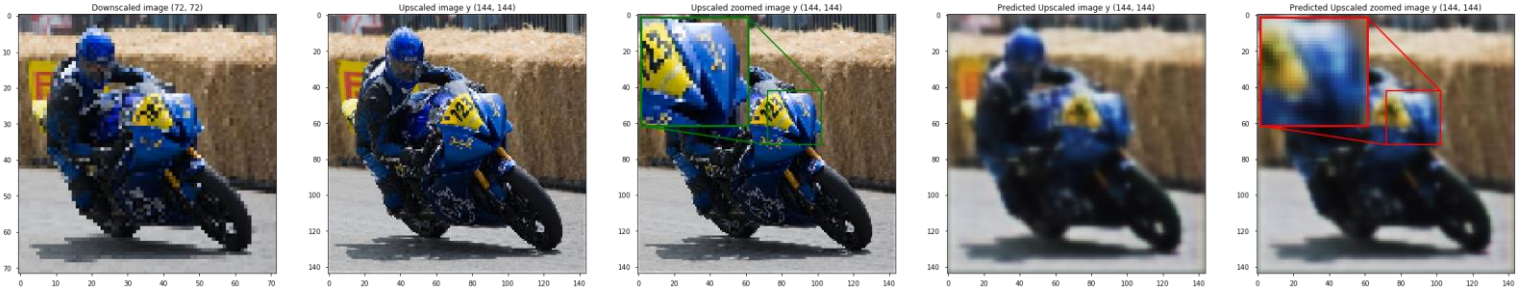
`residual_block_model Mean Validation Metrics (144, 144)`

	MSE	MAE	PSNR	SSIM
0	0.008201	0.058387	21.378817	0.640398

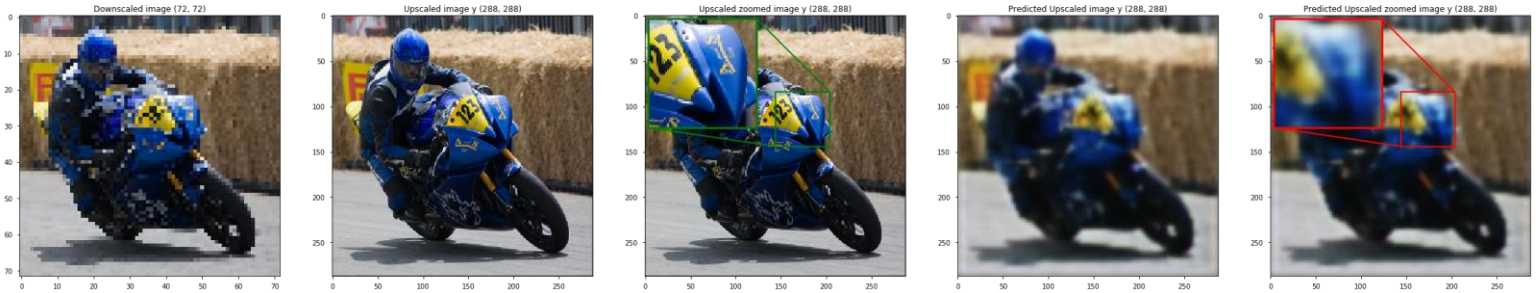
`residual_block_model Mean Validation Metrics (288, 288)`

	MSE	MAE	PSNR	SSIM
0	0.007551	0.054517	21.8011	0.609584

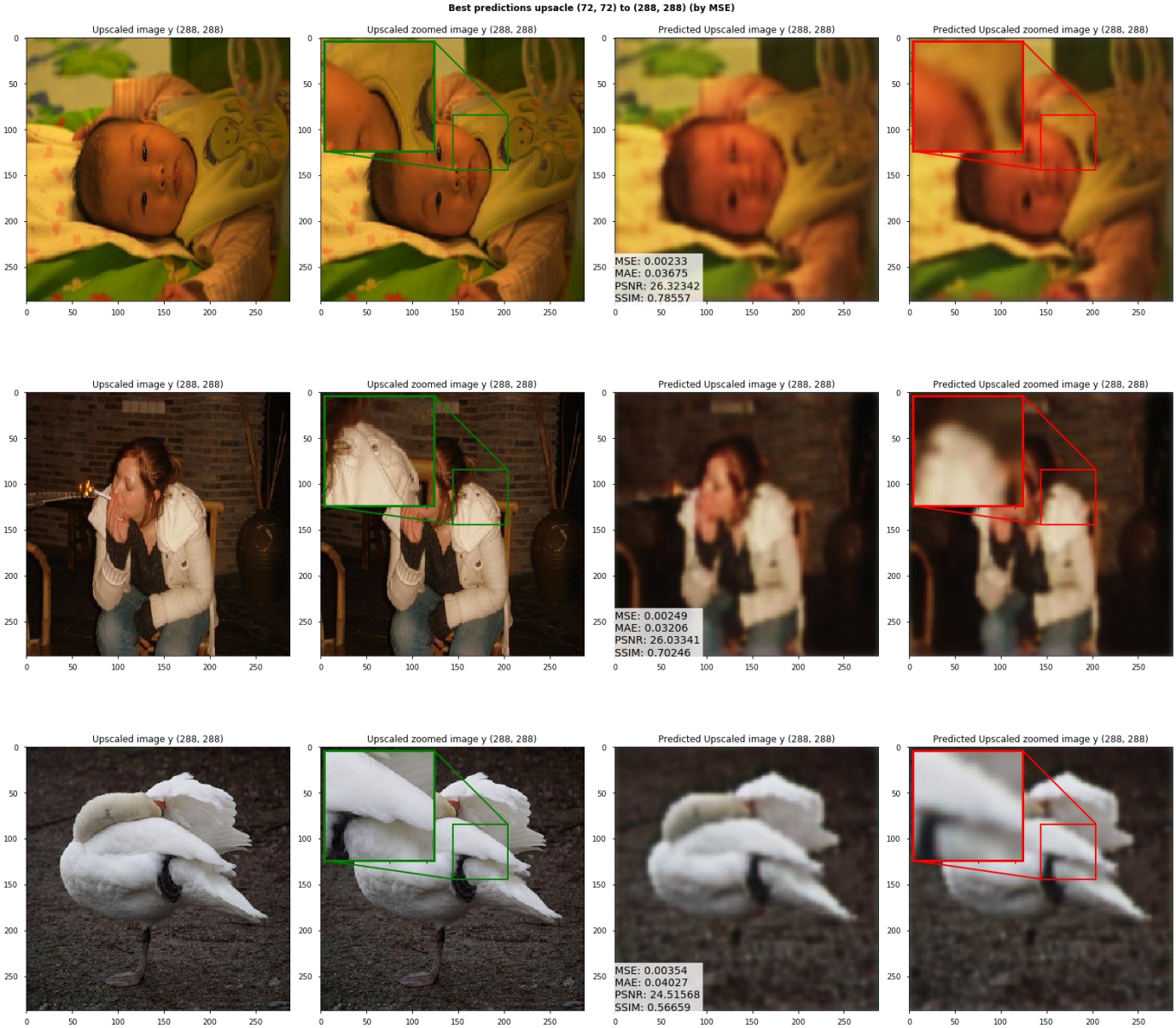
Residual Block Model prediction example (144, 144)



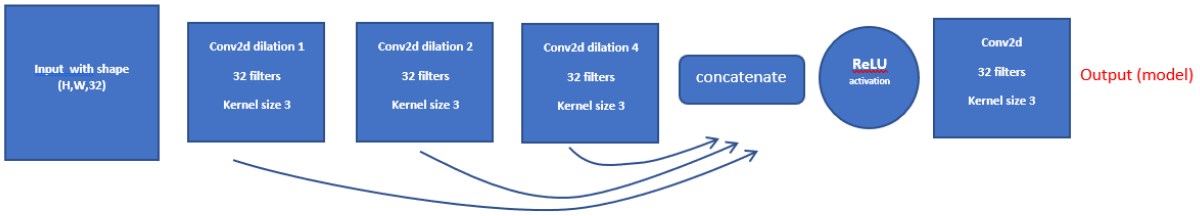
Residual Block Model prediction example (288, 288)

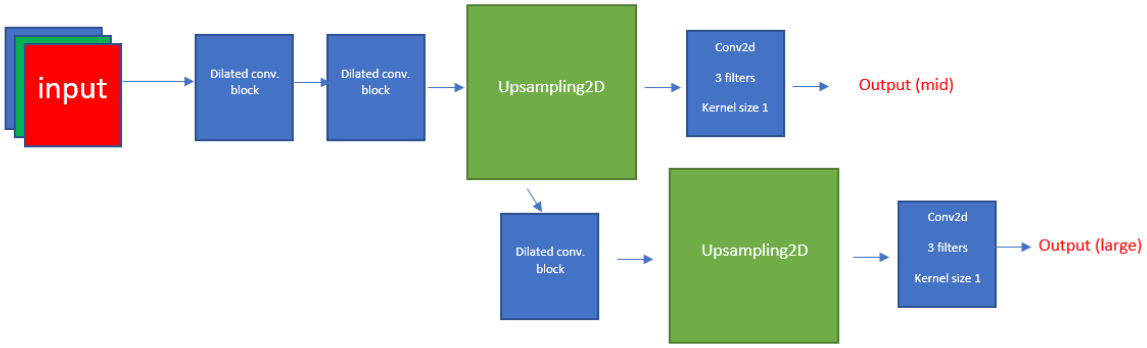


Highest MSE metric predicted images:



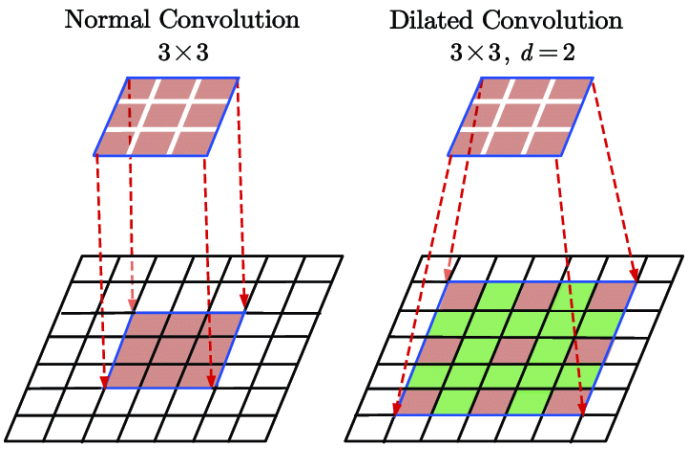
Step 5 - Replace residual blocks with a dilated (Atrous) convolutional block

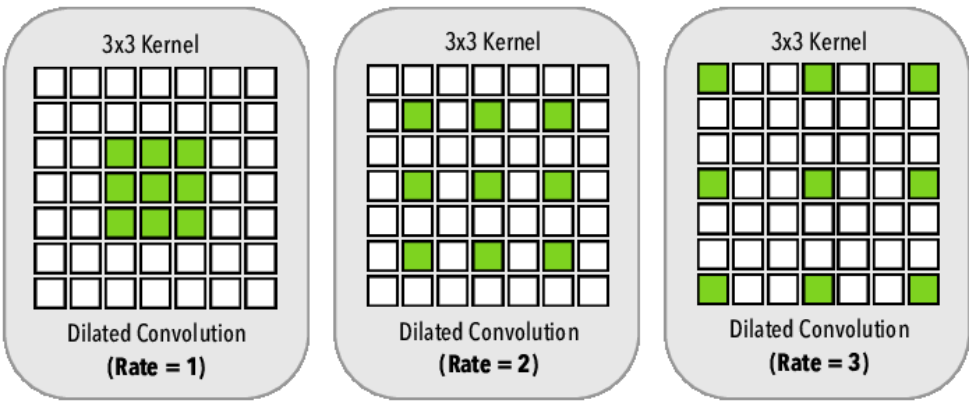




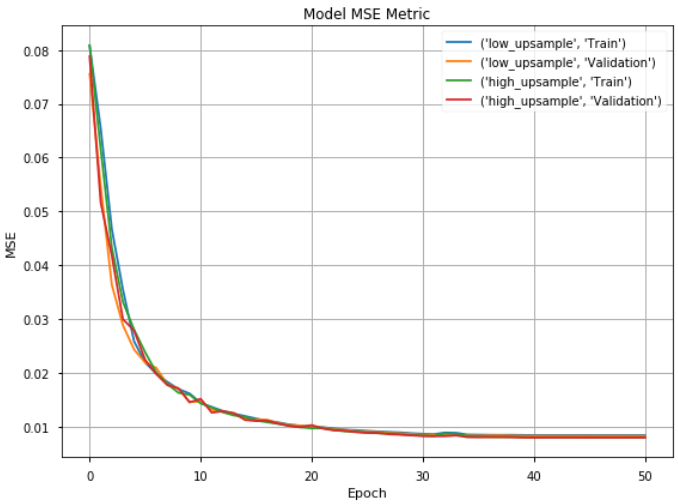
Dilated convolutional blocks

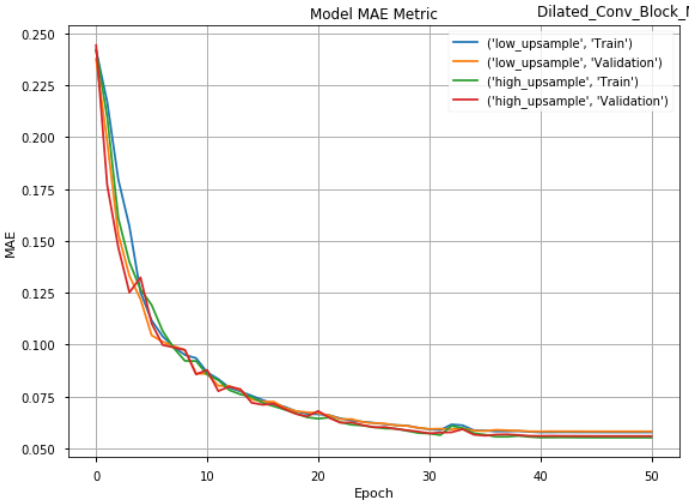
Dilated Convolutions (or atrous convolution) are a type of convolution that “inflate” the kernel by in serting holes between the kernel elements (pixel skipping).
An additional parameter l (dilation rate) indicates how much the kernel is widened.
There are usually l - 1 spaces inserted between kernel elements.
Dilated convolution skips pixels with 2 directions: width and height and as a result it takes much bi gger part of the image for every step of the convolution.
If you put dilation rate 2, then it will take 2 times bigger part of the image for every step. it will take every second pixel of that part.

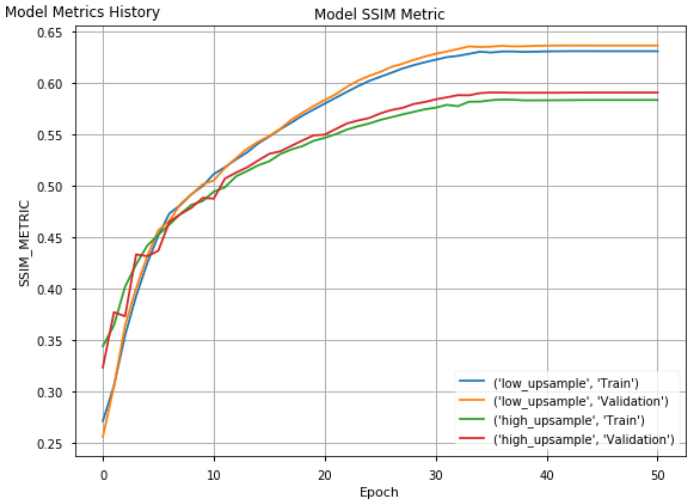


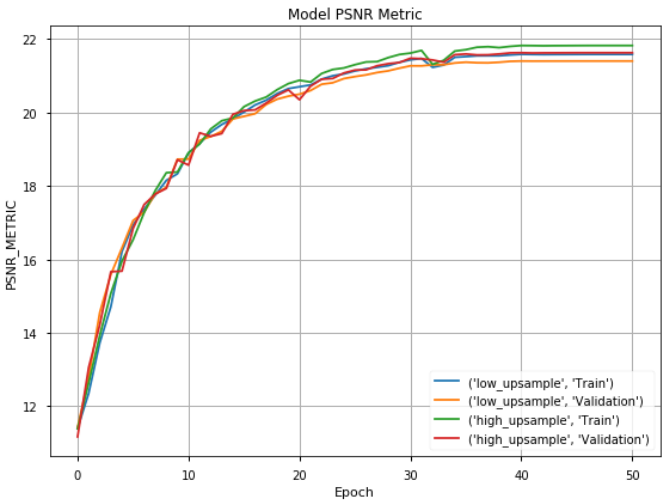


```
Model: "functional_65"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_30 (InputLayer)        [(None, None, None,  0
_________________________________________________________________________________________________
conv2d_50 (Conv2D)           (None, None, None, 3 128         input_30[0][0]
_________________________________________________________________________________________________
functional_59 (Functional)   (None, None, None, 3 55424       conv2d_50[0][0]
_________________________________________________________________________________________________
functional_61 (Functional)   (None, None, None, 3 55424       functional_59[0][0]
_________________________________________________________________________________________________
up_sampling2d_22 (UpSampling2D) (None, None, None, 3 0           functional_61[0][0]
_________________________________________________________________________________________________
functional_63 (Functional)   (None, None, None, 3 55424       up_sampling2d_22[0][0]
_________________________________________________________________________________________________
up_sampling2d_23 (UpSampling2D) (None, None, None, 3 0           functional_63[0][0]
_________________________________________________________________________________________________
low_upsample (Conv2D)        (None, None, None, 3 99          up_sampling2d_22[0][0]
_________________________________________________________________________________________________
high_upsample (Conv2D)       (None, None, None, 3 99          up_sampling2d_23[0][0]
=================================================================================================
Total params: 166,598
Trainable params: 166,598
Non-trainable params: 0
_________________________________________________________________________________________________
```









Mean Model Statistics:

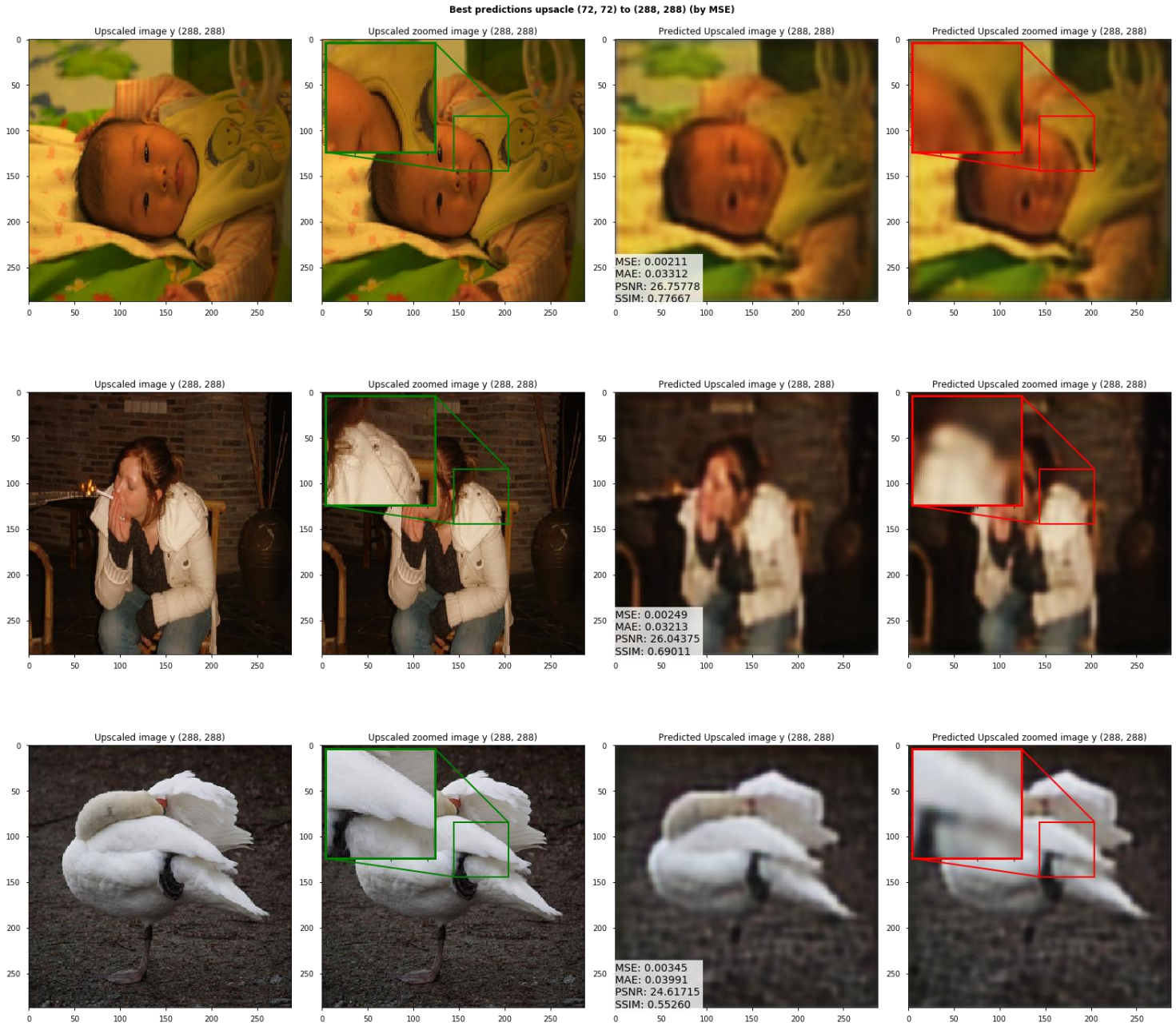
`dilated_conv_block_model Mean Validation Metrics (144, 144)`

	MSE	MAE	PSNR	SSIM
0	0.008272	0.058157	21.39794	0.636006

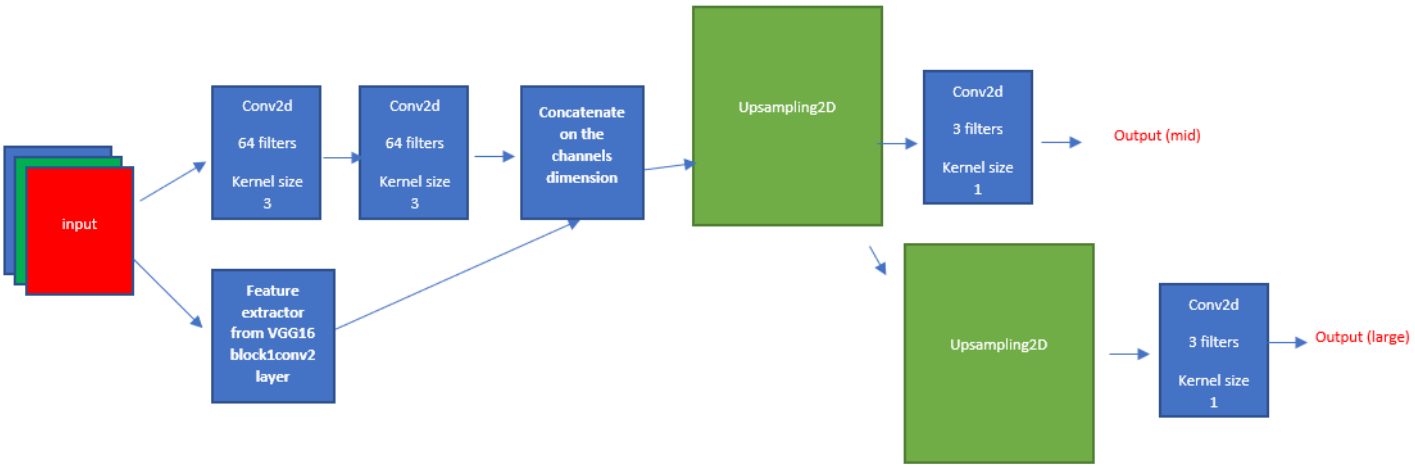
`dilated_conv_block_model Mean Validation Metrics (288, 288)`

	MSE	MAE	PSNR	SSIM
0	0.007961	0.055945	21.623392	0.590215

Highest MSE metric predicted images:



Step 6 - Add VGG16 pretrained network feature extractor to the network

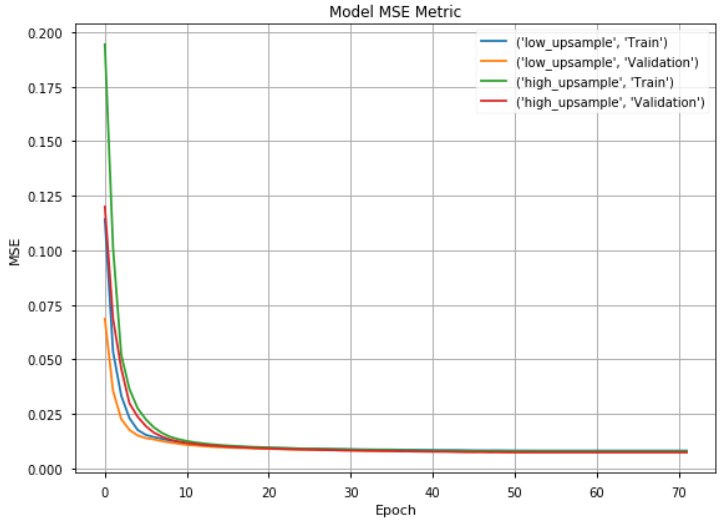


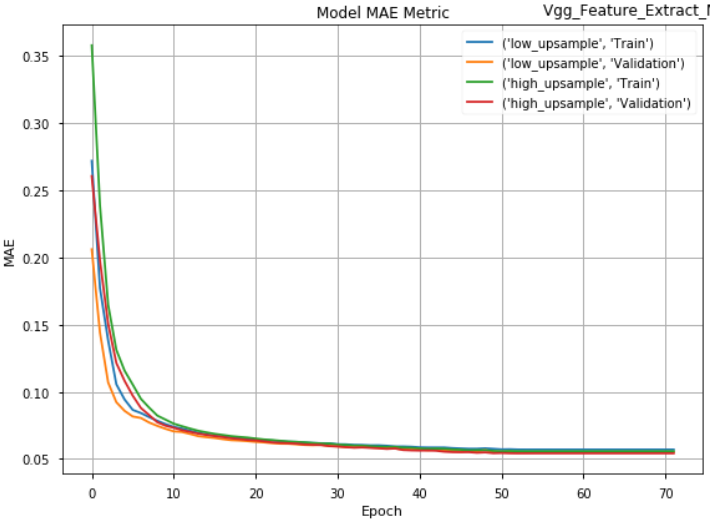
VGG Feature Extractor:

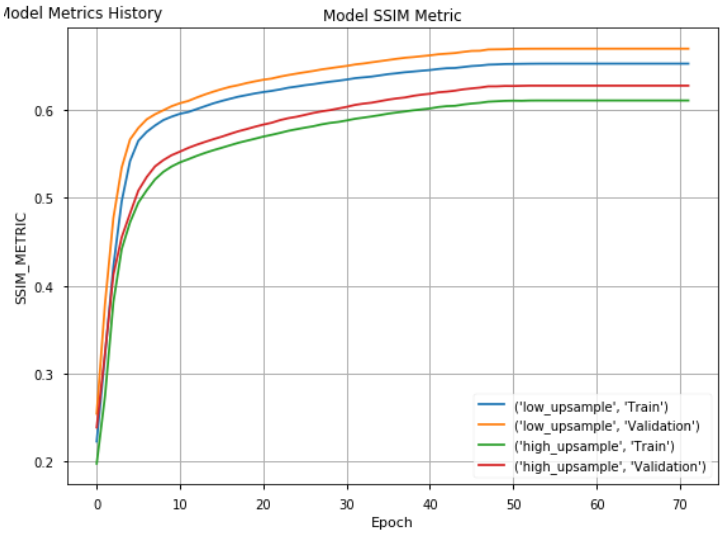
```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, None, None, 3)]   0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
=================================================================
Total params: 38,720
Trainable params: 0
Non-trainable params: 38,720
_________________________________________________________________
```

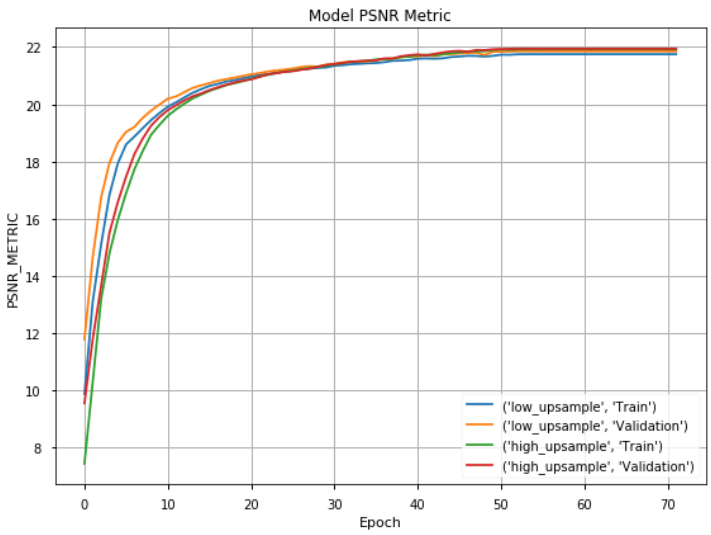
Model:

```
Model: "functional_79"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_39 (InputLayer)           [(None, None, None,  0
__________________________________________________________________________________________________
conv2d_76 (Conv2D)              (None, None, None, 6 256         input_39[0][0]
__________________________________________________________________________________________________
conv2d_77 (Conv2D)              (None, None, None, 6 4160        conv2d_76[0][0]
__________________________________________________________________________________________________
functional_77 (Functional)      (None, None, None, 6 38720       input_39[0][0]
__________________________________________________________________________________________________
concatenate_12 (Concatenate)    (None, None, None, 1 0           conv2d_77[0][0]
                                                                 functional_77[0][0]
__________________________________________________________________________________________________
up_sampling2d_26 (UpSampling2D) (None, None, None, 1 0           concatenate_12[0][0]
__________________________________________________________________________________________________
up_sampling2d_27 (UpSampling2D) (None, None, None, 1 0           up_sampling2d_26[0][0]
__________________________________________________________________________________________________
low_upsample (Conv2D)           (None, None, None, 3 387         up_sampling2d_26[0][0]
__________________________________________________________________________________________________
high_upsample (Conv2D)          (None, None, None, 3 387         up_sampling2d_27[0][0]
==================================================================================================
Total params: 43,910
Trainable params: 43,910
Non-trainable params: 0
__________________________________________________________________________________________________
```

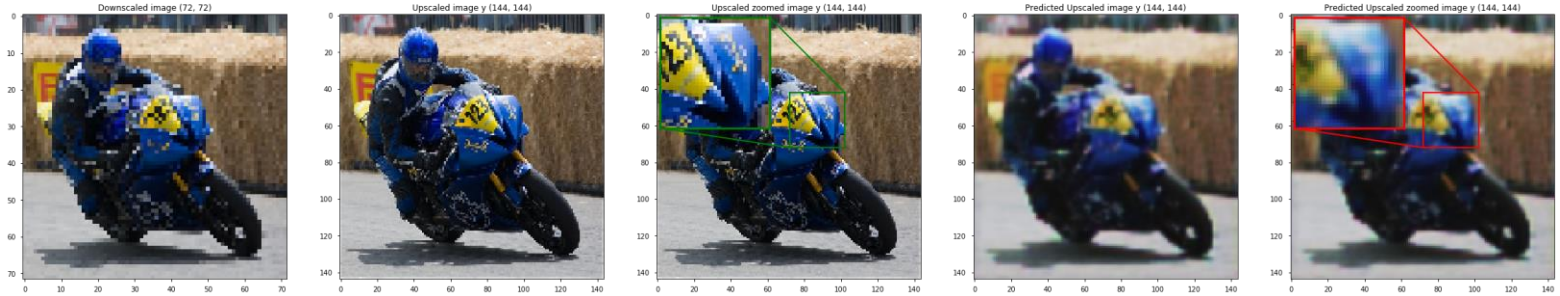




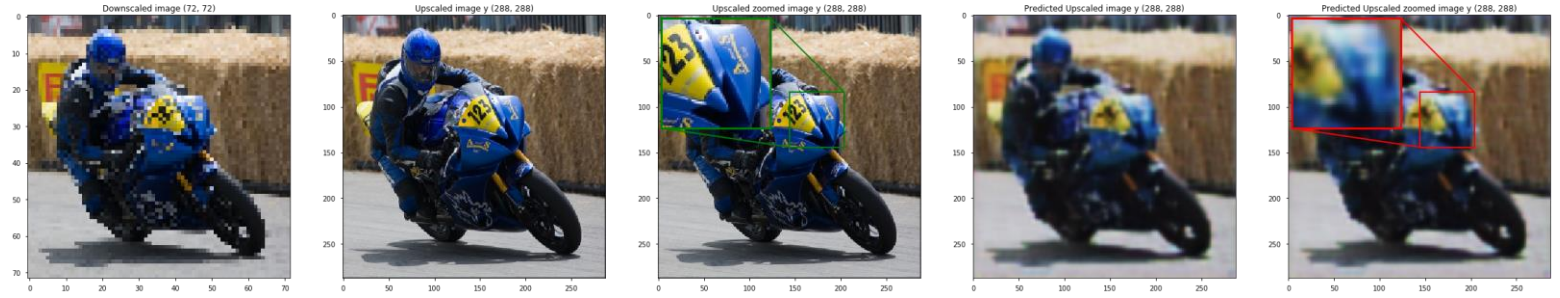




Vgg Feature Extract Model prediction example (144, 144)



Vgg Feature Extract Model prediction example (288, 288)



Mean Model Statistics:

vgg_feature_extract_model Mean Validation Metrics (144, 144)

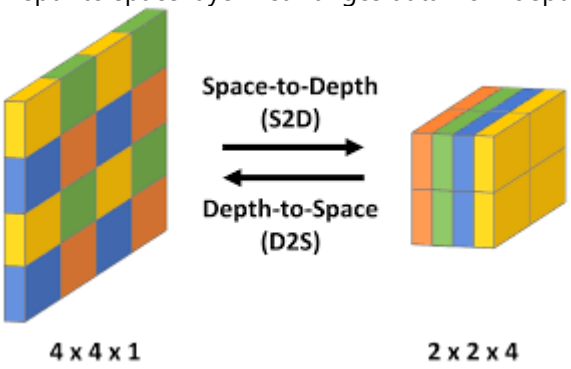
	MSE	MAE	PSNR	SSIM
0	0.007484	0.055156	21.85358	0.669549

vgg_feature_extract_model Mean Validation Metrics (288, 288)

	MSE	MAE	PSNR	SSIM

Step 7 - Replace the Upsampling2D layer with depth_to_space

Depth to space layer Rearranges data from depth into blocks of spatial data.



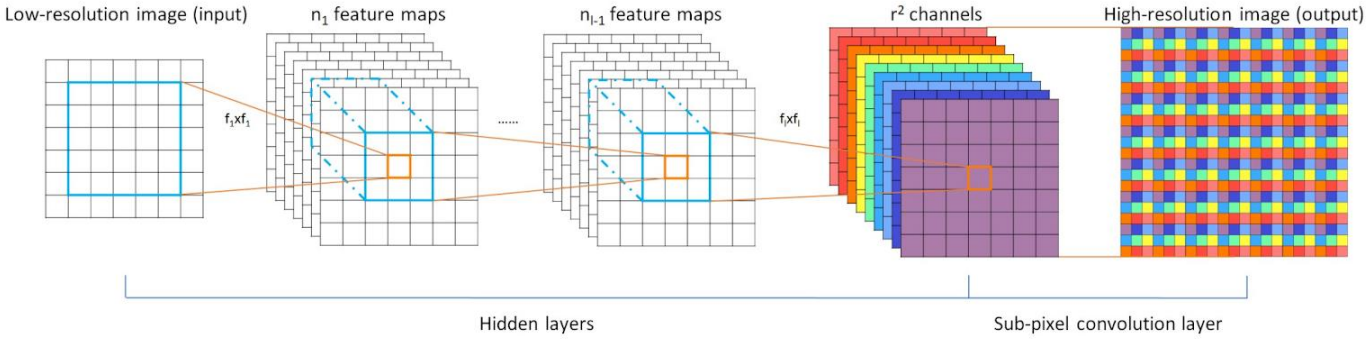
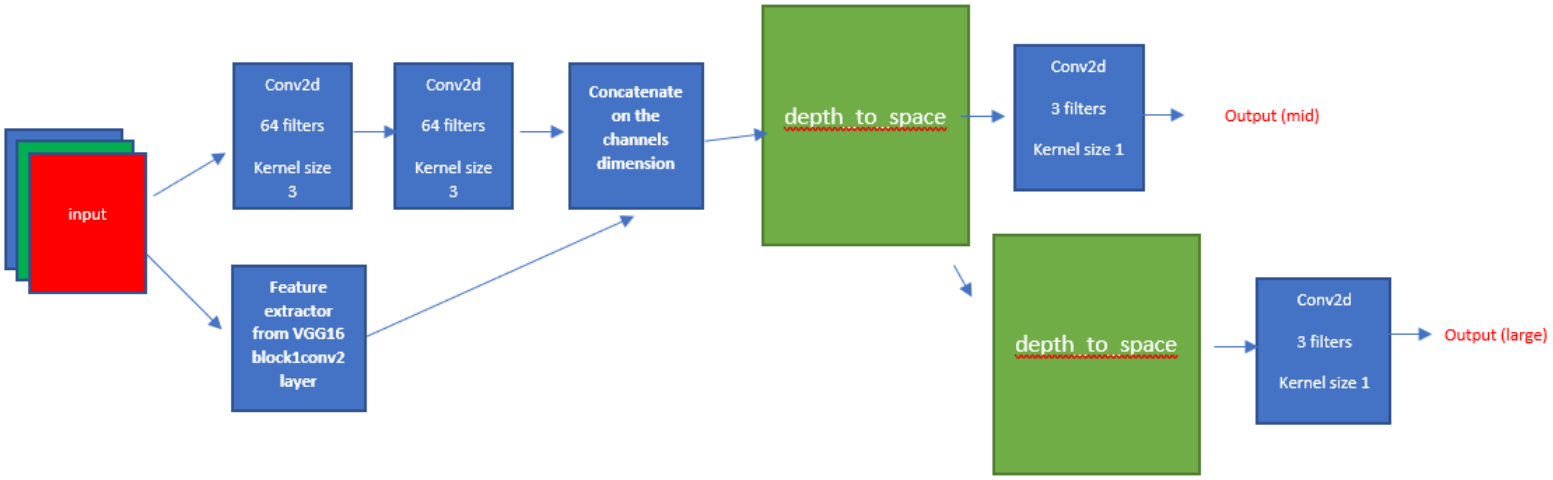
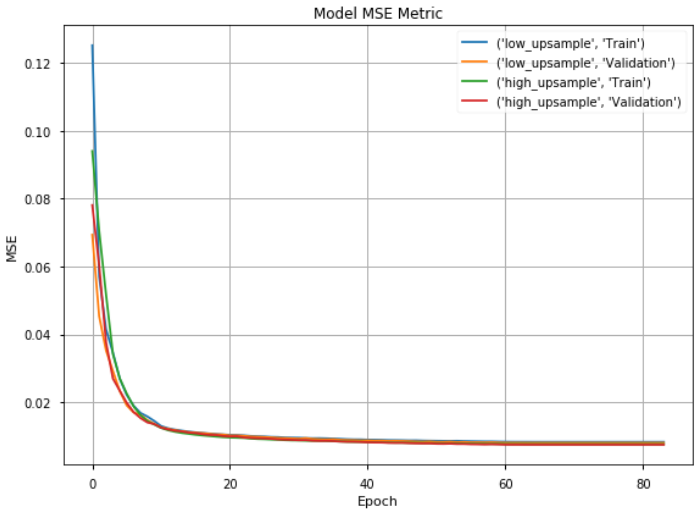
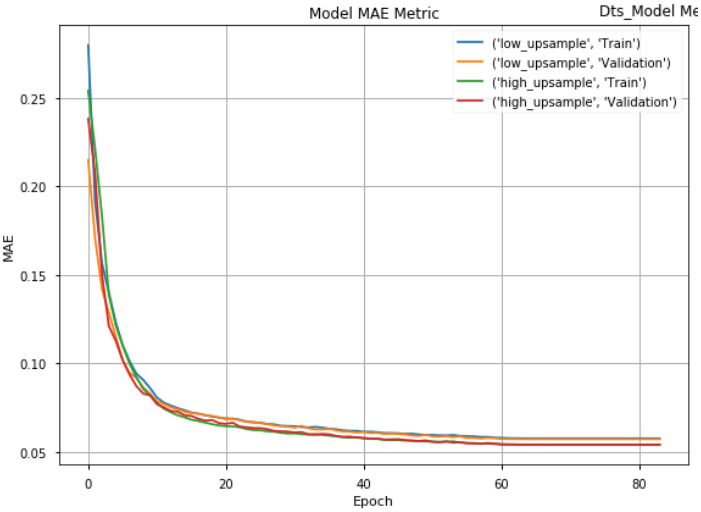


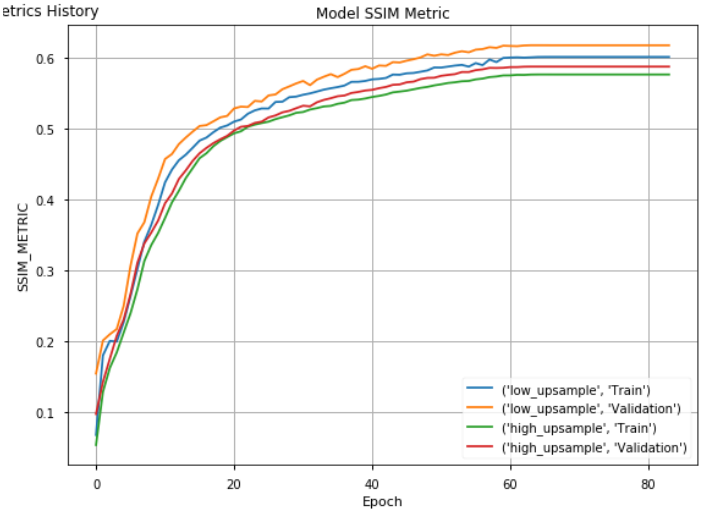
Figure 1. The proposed efficient sub-pixel convolutional neural network (ESPCN), with two convolution layers for feature maps extraction, and a sub-pixel convolution layer that aggregates the feature maps from LR space and builds the SR image in a single step.

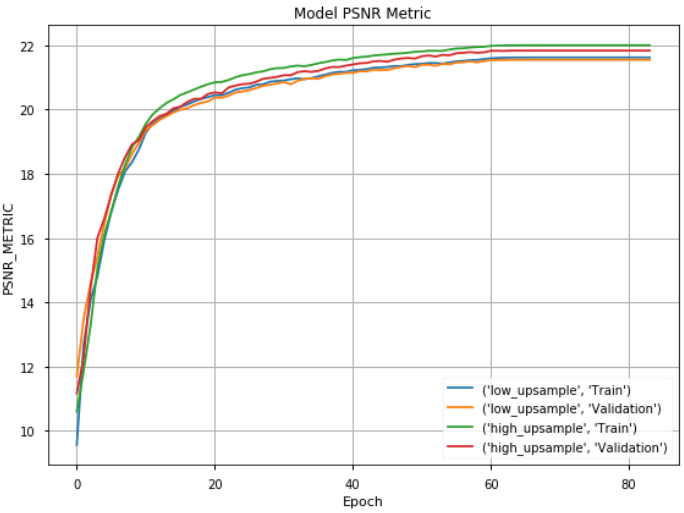


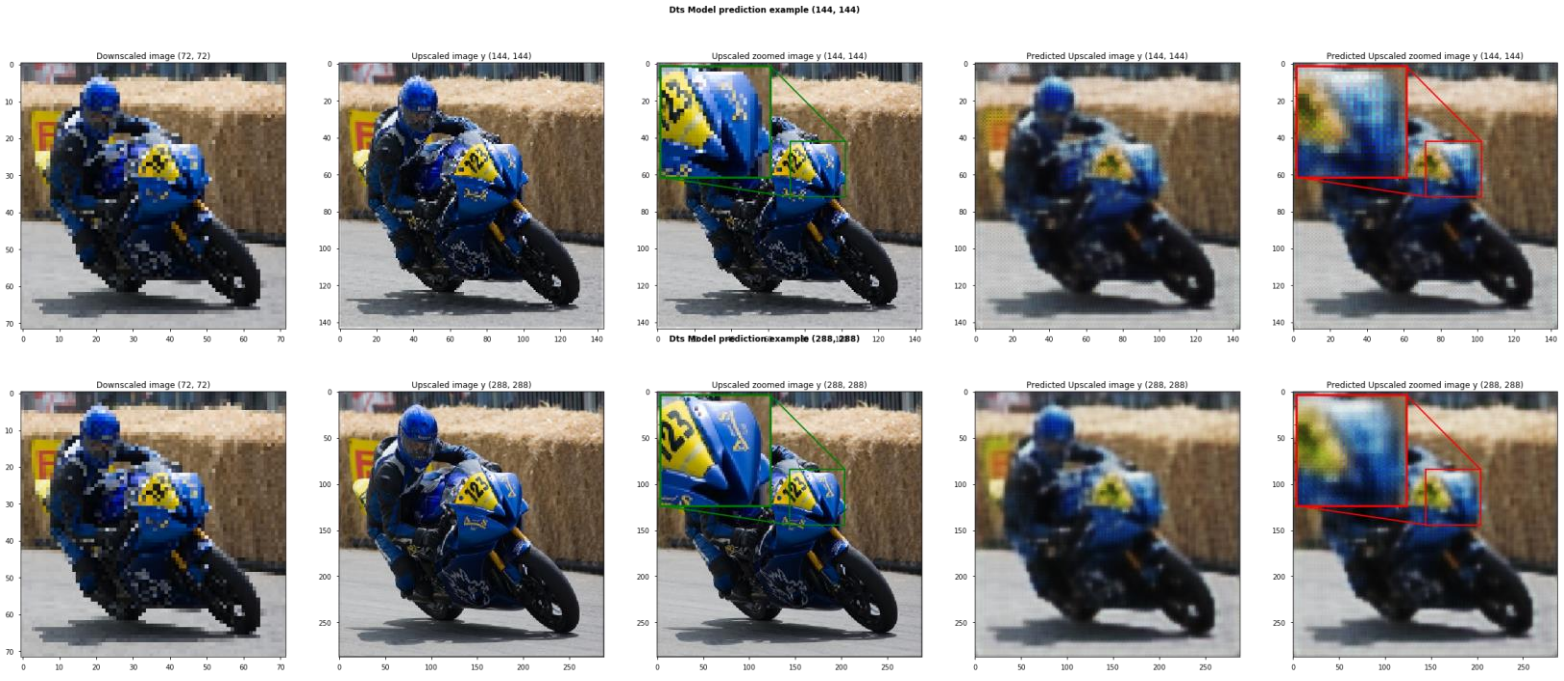
```
Model: "functional_93"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_46 (InputLayer)        [(None, None, None,  0
_________________________________________________________________________________________________
conv2d_120 (Conv2D)          (None, None, None, 6 1792      input_46[0][0]
_________________________________________________________________________________________________
conv2d_121 (Conv2D)          (None, None, None, 6 36928     conv2d_120[0][0]
_________________________________________________________________________________________________
functional_91 (Functional)   (None, None, None, 6 38720     input_46[0][0]
_________________________________________________________________________________________________
concatenate_31 (Concatenate) (None, None, None, 1 0         conv2d_121[0][0]
                                                            functional_91[0][0]
_________________________________________________________________________________________________
conv2d_122 (Conv2D)          (None, None, None, 6 73792     concatenate_31[0][0]
_________________________________________________________________________________________________
tf_op_layer_DepthToSpace_8 (Ten [(None, None, None,  0      conv2d_122[0][0]
_________________________________________________________________________________________________
conv2d_123 (Conv2D)          (None, None, None, 6 9280      tf_op_layer_DepthToSpace_8[0][
0]
_________________________________________________________________________________________________
tf_op_layer_DepthToSpace_9 (Ten [(None, None, None,  0      conv2d_123[0][0]
_________________________________________________________________________________________________
low_upsample (Conv2D)        (None, None, None, 3 51        tf_op_layer_DepthToSpace_8[0][
0]
_________________________________________________________________________________________________
high_upsample (Conv2D)       (None, None, None, 3 51        tf_op_layer_DepthToSpace_9[0][
0]
=================================================================================================
Total params: 160,614
Trainable params: 121,894
Non-trainable params: 38,720
_________________________________________________________________________________________________
```


Model MSE Metric
MSE
Epoch
('low_upsample', 'Train')
('low_upsample', 'Validation')
('high_upsample', 'Train')
('high_upsample', 'Validation')


Model MAE Metric
MAE
Epoch
('low_upsample', 'Train')
('low_upsample', 'Validation')
('high_upsample', 'Train')
('high_upsample', 'Validation')


Model SSIM Metric
SSIM_METRIC
Epoch
('low_upsample', 'Train')
('low_upsample', 'Validation')
('high_upsample', 'Train')
('high_upsample', 'Validation')


Model PSNR Metric
PSNR_METRIC
Epoch
('low_upsample', 'Train')
('low_upsample', 'Validation')
('high_upsample', 'Train')
('high_upsample', 'Validation')


Dts Model prediction example (144, 144)
Downscaled image (72, 72)
Upscaled image y (144, 144)
Upscaled zoomed image y (144, 144)
Predicted Upscaled image y (144, 144)
Predicted Upscaled zoomed image y (144, 144)
Dts Model prediction example (288, 288)
Downscaled image (72, 72)
Upscaled image y (288, 288)
Upscaled zoomed image y (288, 288)
Predicted Upscaled image y (288, 288)
Predicted Upscaled zoomed image y (288, 288)

Mean Model Statistics:

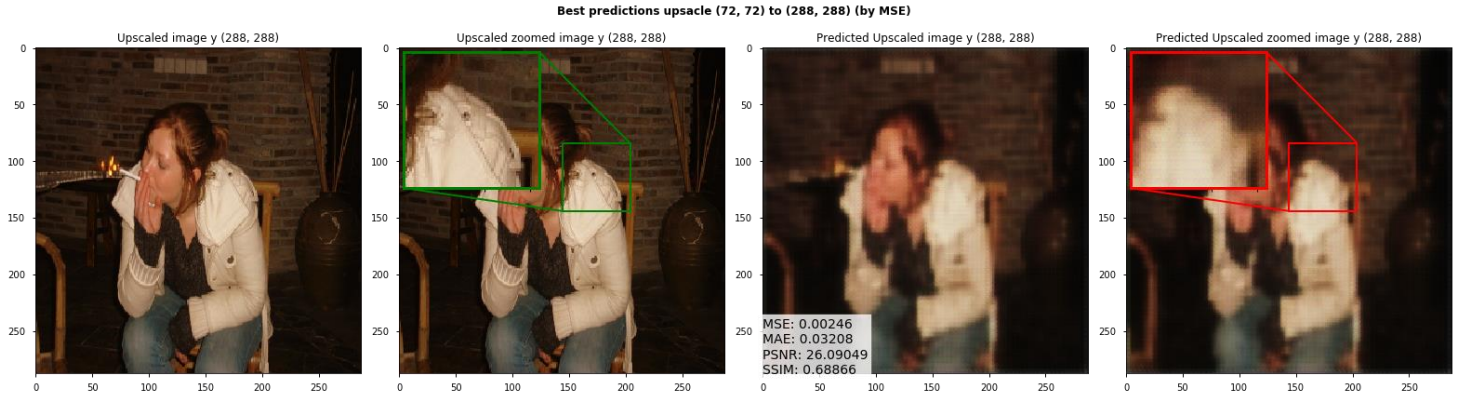
dts_model Mean Validation Metrics (144, 144)

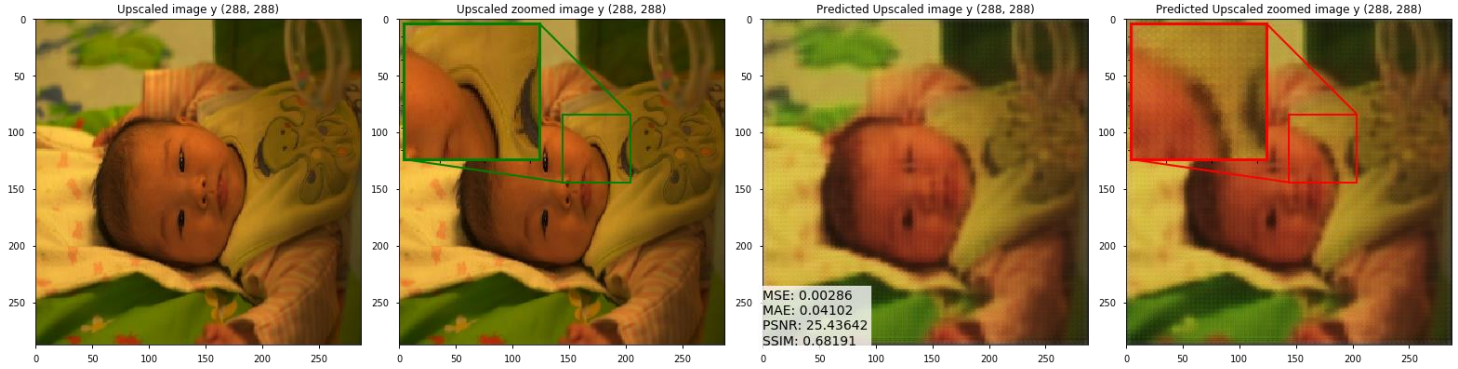
	MSE	MAE	PSNR	SSIM
0	0.00802	0.057105	21.548134	0.616974

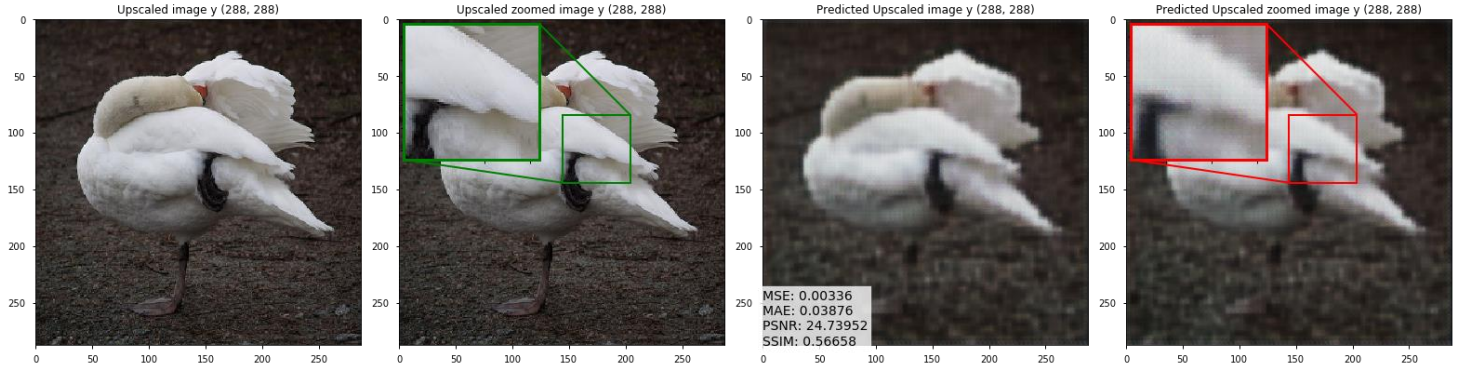
dts_model Mean Validation Metrics (288, 288)

	MSE	MAE	PSNR	SSIM
0	0.007464	0.054004	21.831583	0.586844

Highest MSE metric predicted images:







Result Models

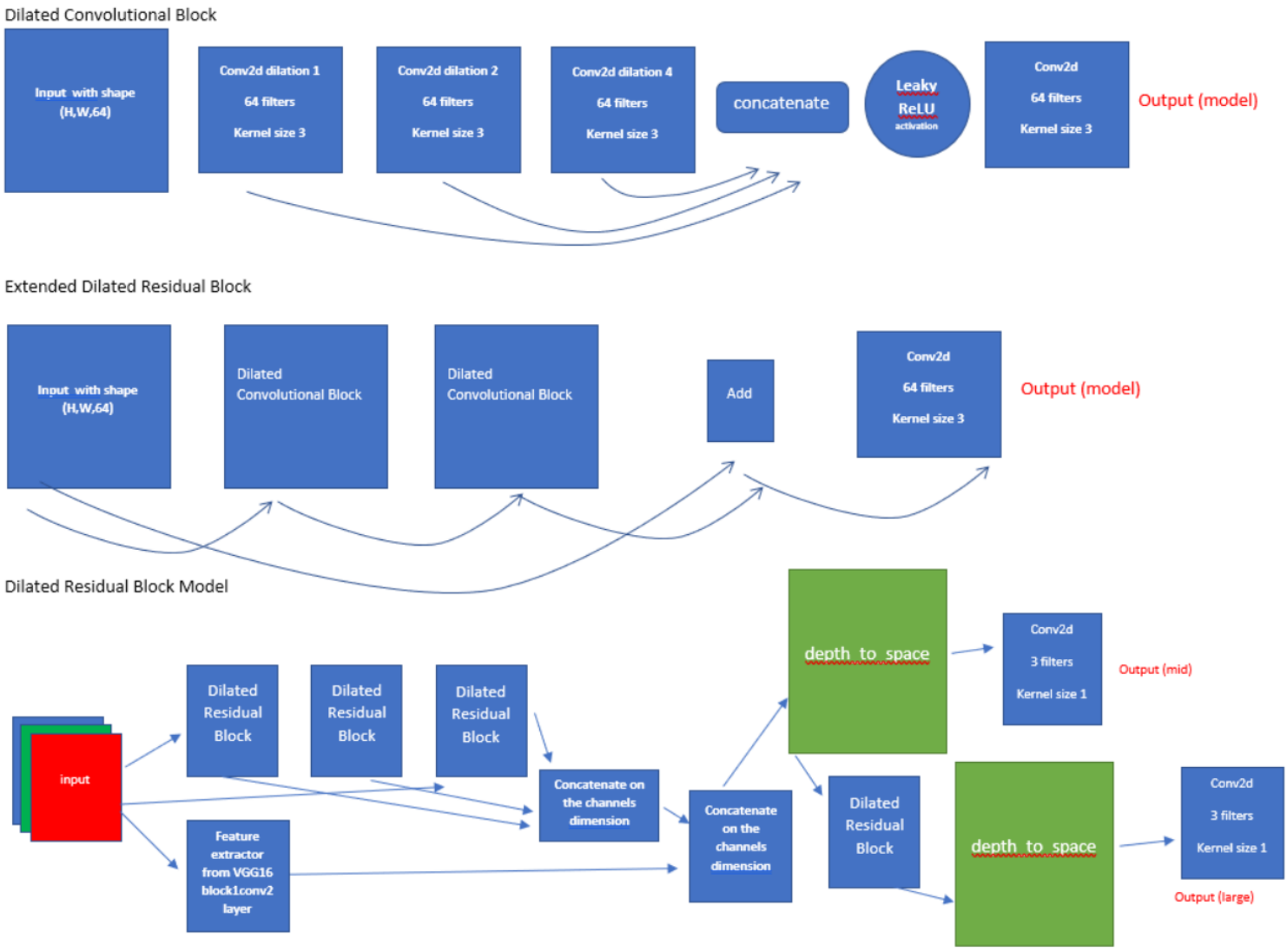
In the following model implementations, a larger portion of the data was used in the training process in order to try and achieve the best results possible.
Additionally, a few experiments made regarding models architecture, loss functions and optimizers to achieve the best results possible while not diverging.

Size of training data = 2500 images

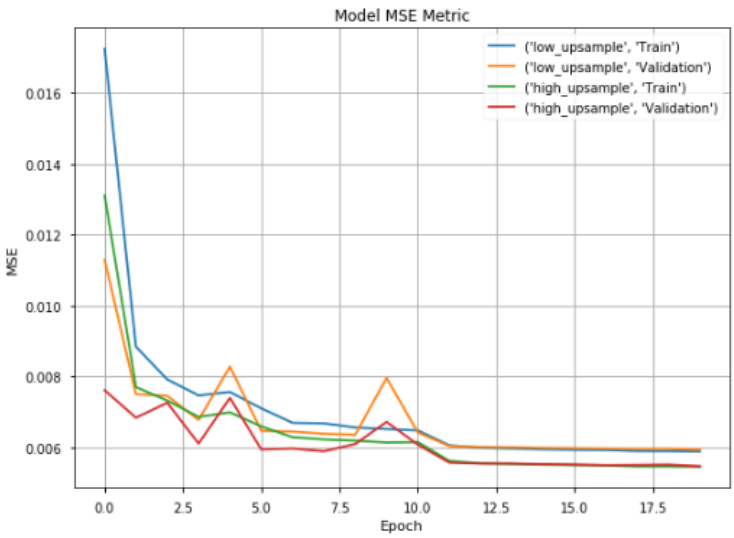
Model 1

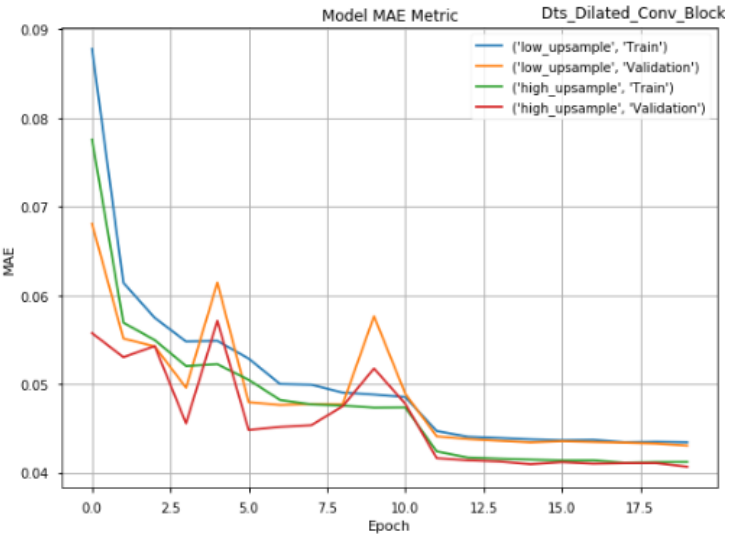
Model Name: dts_dilated_conv_block_model
Loss function used: $-PSNR$
Model architecture:
The idea behind the architecture is to use "dialated residual block" – residual blocks that are made of 2 dialated convolutional blocks as shown previously in this report.
All the dialated residual block are concatenated and then concatenated again using a VGG16 feature extractor.

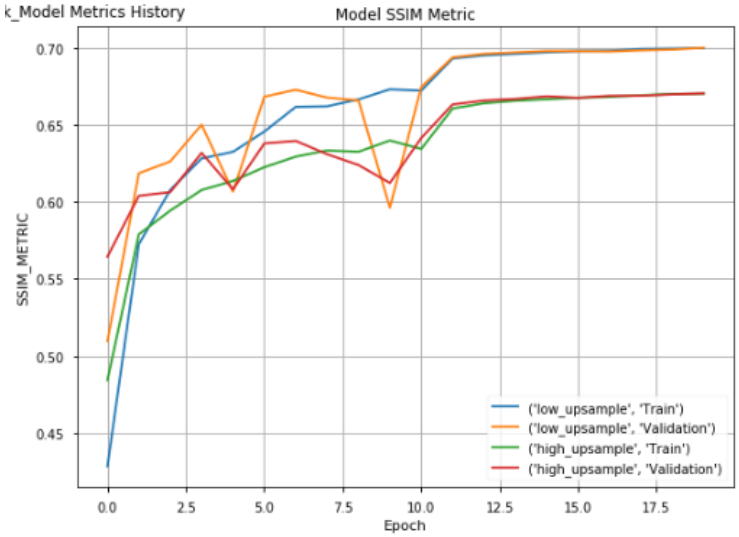
Model diagram:

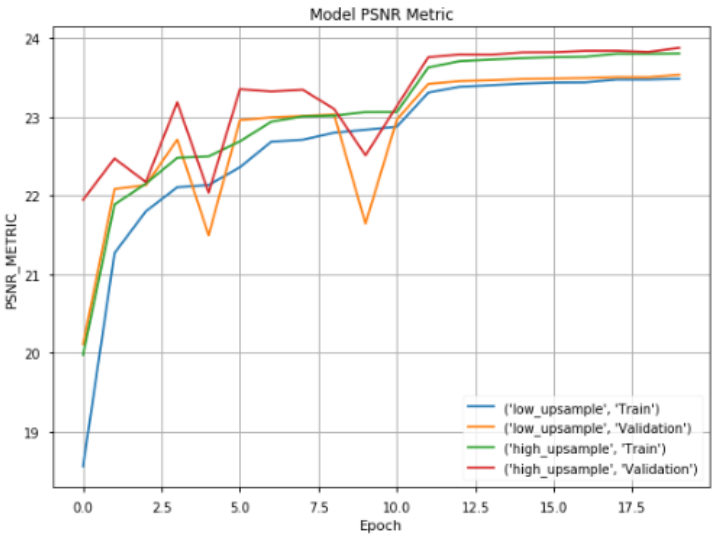


```
Model: "functional_349"
_____
___
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
===
input_162 (InputLayer)       [(None, None, None,  0
___
conv2d_440 (Conv2D)          (None, None, None, 6 1792        input_162[0][0]
_____
___
functional_327 (Functional)  (None, None, None, 6 479808      conv2d_440[0][0]
___
functional_333 (Functional)  (None, None, None, 6 479808      conv2d_440[0][0]
_____
___
functional_339 (Functional)  (None, None, None, 6 479808      conv2d_440[0][0]
___
concatenate_119 (Concatenate)  (None, None, None, 1 0         functional_327[0][0]
                                                              functional_333[0][0]
                                                              functional_339[0][0]
___
functional_341 (Functional)  (None, None, None, 6 38720       input_162[0][0]
_____
___
concatenate_120 (Concatenate)  (None, None, None, 2 0         concatenate_119[0][0]
                                                              functional 341[0][0]
_____
___
conv2d_468 (Conv2D)          (None, None, None, 6 147520      concatenate_120[0][0]
_____
___
tf op layer DepthToSpace 28 (Te [(None, None, None,  0        conv2d 468[0][0]
_____
___
conv2d_469 (Conv2D)          (None, None, None, 6 9280        tf_op_layer_DepthToSpace_28[0]
[0]
_____
___
functional_347 (Functional)  (None, None, None, 6 479808      conv2d_469[0][0]
_____
___
tf_op_layer_DepthToSpace_29 (Te [(None, None, None,  0        functional_347[0][0]
_____
___
low_upsample (Conv2D)        (None, None, None, 3 51          tf_op_layer_DepthToSpace_28[0]
[0]
_____
___
high upsample (Conv2D)       (None, None, None, 3 51          tf op layer DepthToSpace 29[0]
[0]
==================================================================================================
===
Total params: 2,116,646
Trainable params: 2,077,926
Non-trainable params: 38,720
```









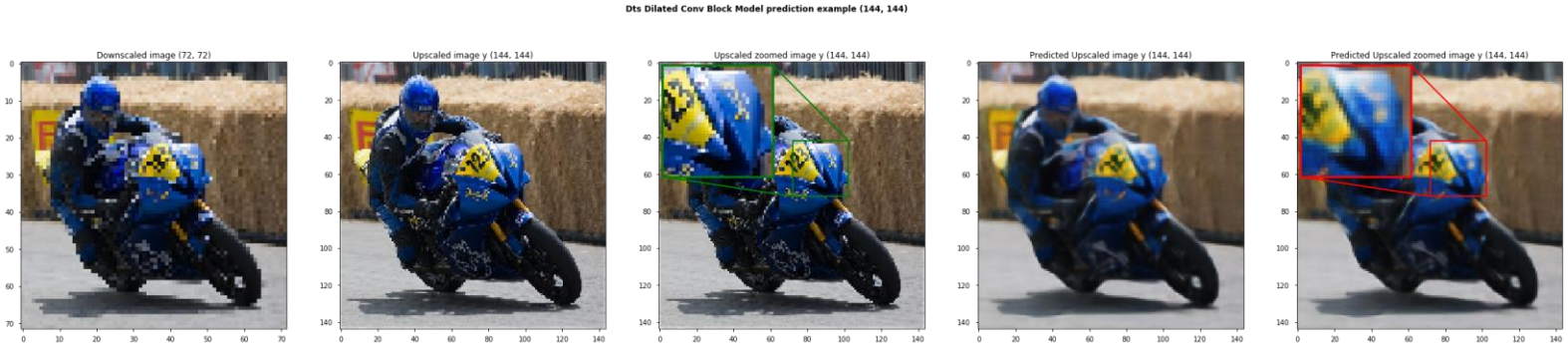
Mean Model Statistics:

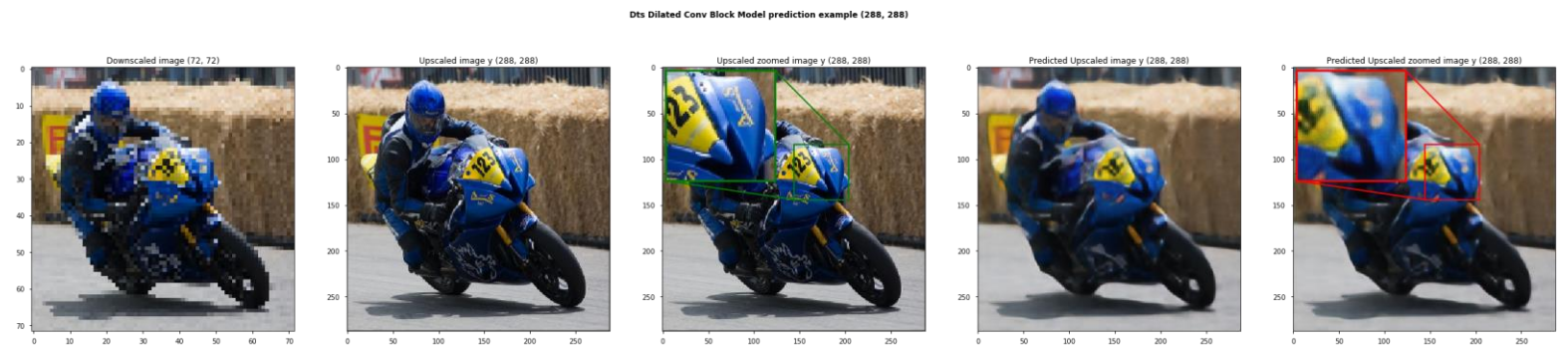
`dts_dilated_conv_block_model Mean Validation Metrics (144, 144)`

	MSE	MAE	PSNR	SSIM
0	0.005536	0.042337	23.492382	0.725306

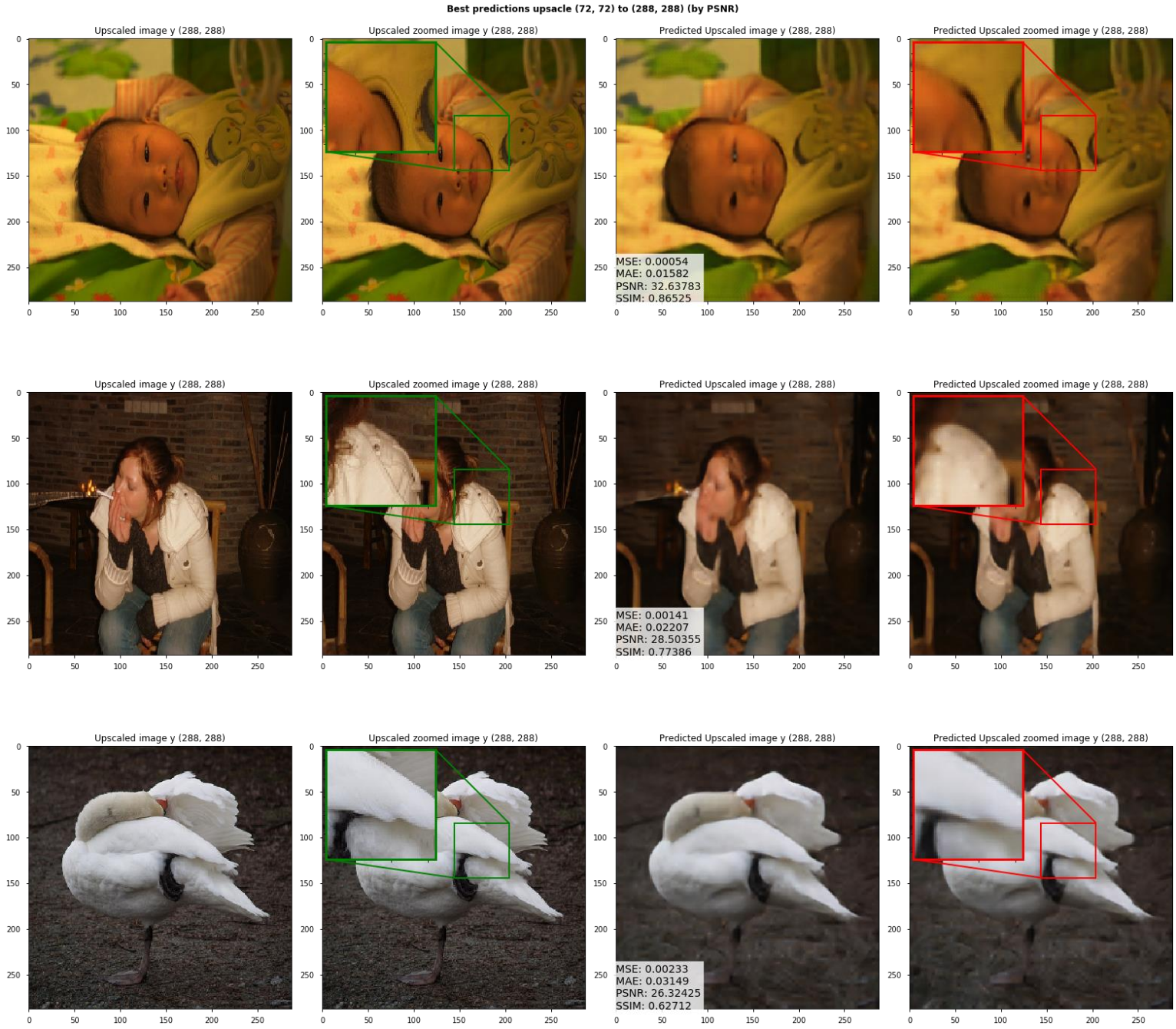
`dts_dilated_conv_block_model Mean Validation Metrics (288, 288)`

	MSE	MAE	PSNR	SSIM
0	0.005126	0.039911	23.820621	0.693821





Highest MSE metric predicted images:



Model 2

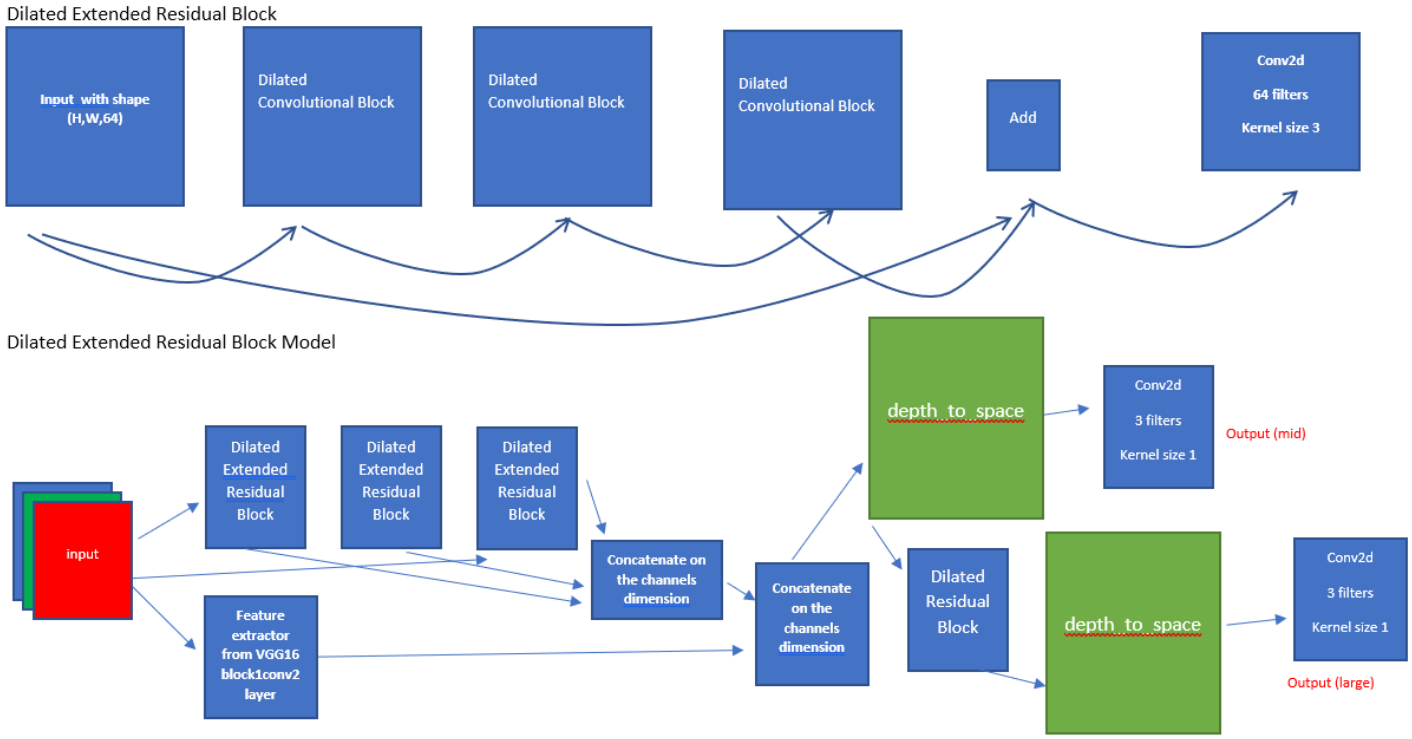
Model Name: dts_dilated_conv_ext_block_model

Loss function used: $-PSNR - SSIM$

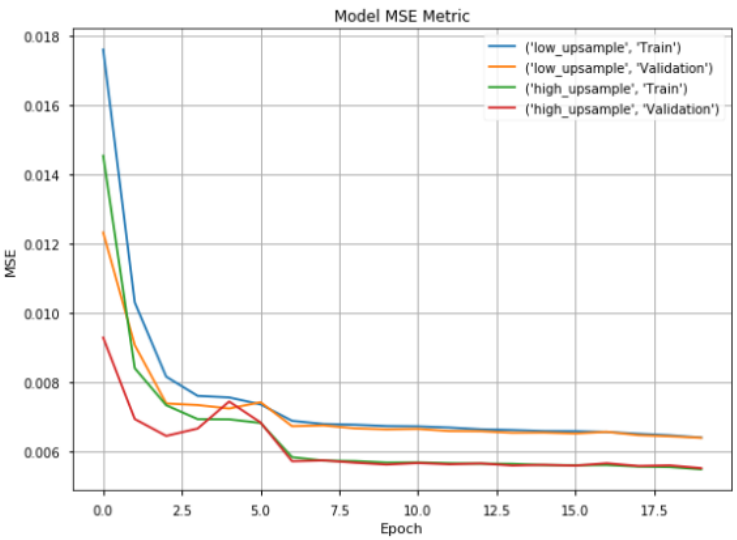
Model architecture:

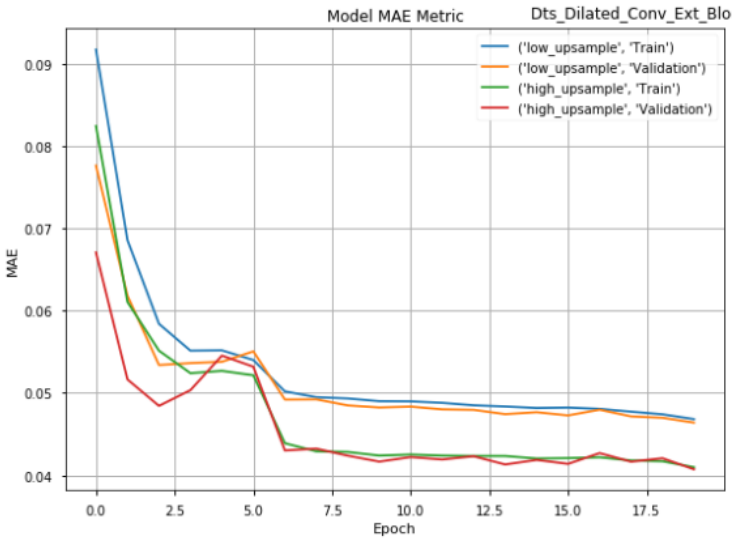
The idea behind the architecture is to use "dialated extended residual block" – residual blocks
that are made of **3** dialated convolutional blocks as shown previously in this report.
All the dialated residual block are concatenated and then concatenated again using a VGG16 feature extractor.

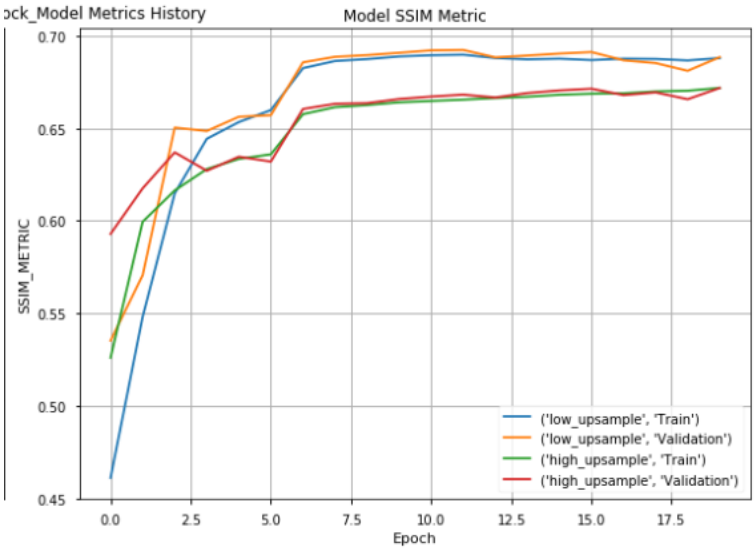
Model diagram:

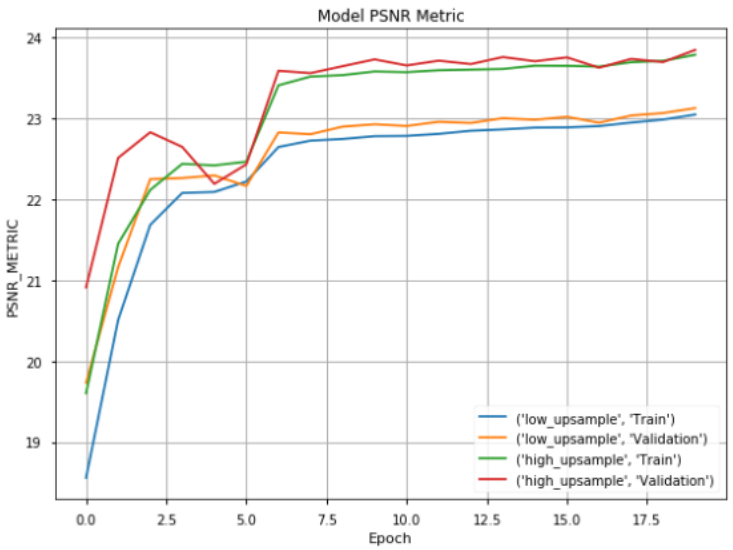


```
Model: "functional_35"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            [(None, None, None,  0
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, None, None, 6 1792        input_4[0][0]
__________________________________________________________________________________________________
functional_9 (Functional)       (None, None, None, 6 701248      conv2d_2[0][0]
__________________________________________________________________________________________________
functional_17 (Functional)      (None, None, None, 6 701248      conv2d_2[0][0]
__________________________________________________________________________________________________
functional_25 (Functional)      (None, None, None, 6 701248      conv2d_2[0][0]
__________________________________________________________________________________________________
concatenate_9 (Concatenate)     (None, None, None, 1 0           functional_9[0][0]
                                                                 functional_17[0][0]
                                                                 functional_25[0][0]
__________________________________________________________________________________________________
functional_27 (Functional)      (None, None, None, 6 38720       input_4[0][0]
__________________________________________________________________________________________________
concatenate_10 (Concatenate)    (None, None, None, 2 0           concatenate_9[0][0]
                                                                 functional_27[0][0]
__________________________________________________________________________________________________
conv2d_42 (Conv2D)              (None, None, None, 6 147520      concatenate_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_DepthToSpace (Tenso [(None, None, None,  0           conv2d_42[0][0]
__________________________________________________________________________________________________
conv2d_43 (Conv2D)              (None, None, None, 6 9280        tf_op_layer_DepthToSpace[0][0]
__________________________________________________________________________________________________
functional_33 (Functional)      (None, None, None, 6 479808      conv2d_43[0][0]
__________________________________________________________________________________________________
tf_op_layer_DepthToSpace_1 (Ten [(None, None, None,  0           functional_33[0][0]
__________________________________________________________________________________________________
low_upsample (Conv2D)           (None, None, None, 3 51          tf_op_layer_DepthToSpace[0][0]
__________________________________________________________________________________________________
high_upsample (Conv2D)          (None, None, None, 3 51          tf_op_layer_DepthToSpace_1[0][
0]
==================================================================================================
Total params: 2,780,966
Trainable params: 2,742,246
Non-trainable params: 38,720
__________________________________________________________________________________________________
```









Mean Model Statistics:

`dts_dilated_conv_ext_block_model Mean Validation Metrics (144, 144)`

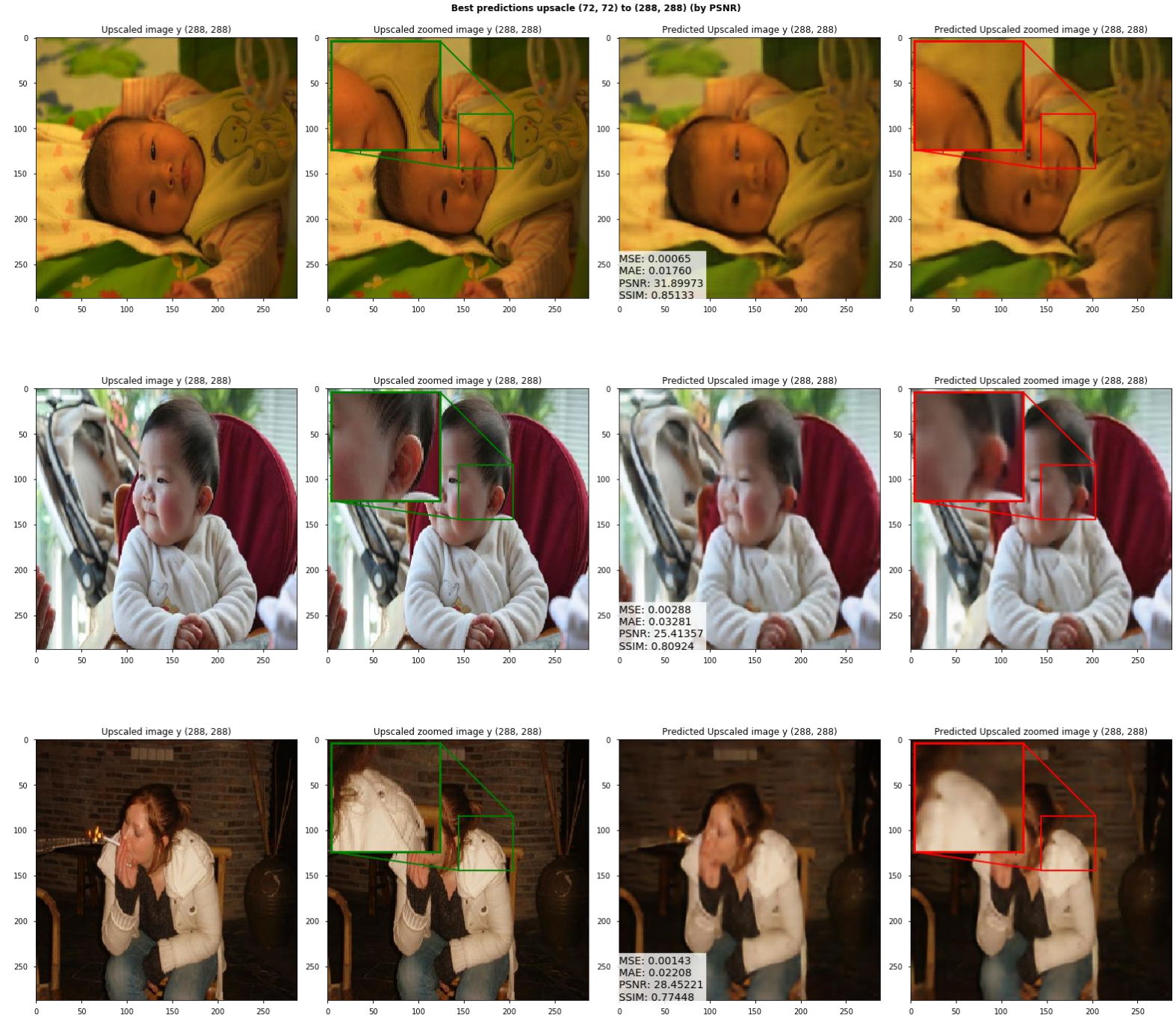
	MSE	MAE	PSNR	SSIM
0	0.006113	0.046698	22.786966	0.707361

`dts_dilated_conv_ext_block_model Mean Validation Metrics (288, 288)`

	MSE	MAE	PSNR	SSIM
0	0.005233	0.040234	23.695541	0.69315

We can see that the lower resolution (144x144x3) output we received in this case does not match the original image color space.

Highest MSE metric predicted images:



Model 3

Model Name: dts_dilated_conv_ext_block_ssim_model

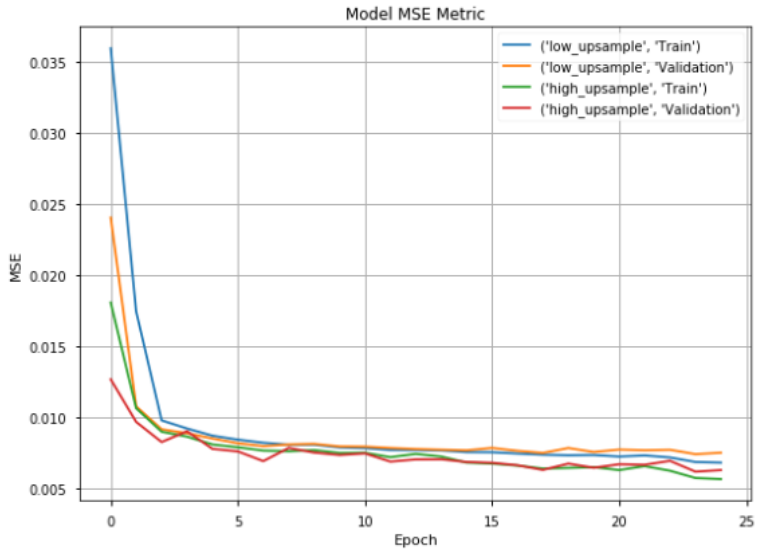
Loss function used: $MSE - SSIM$

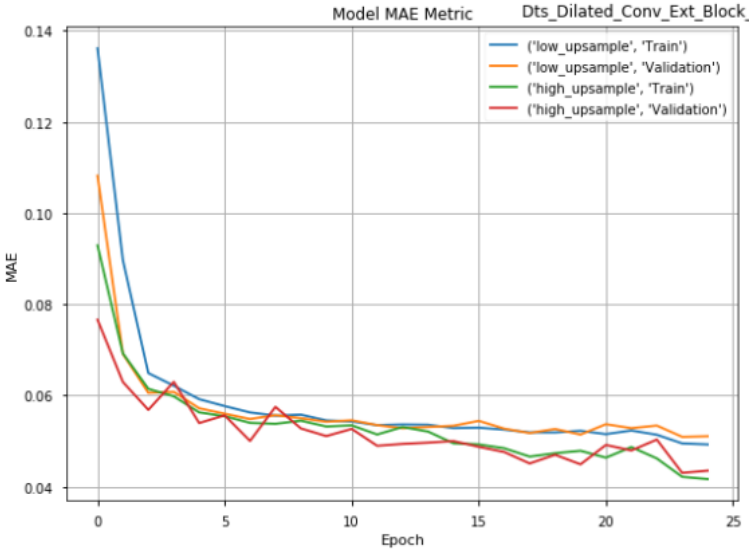
Model architecture:

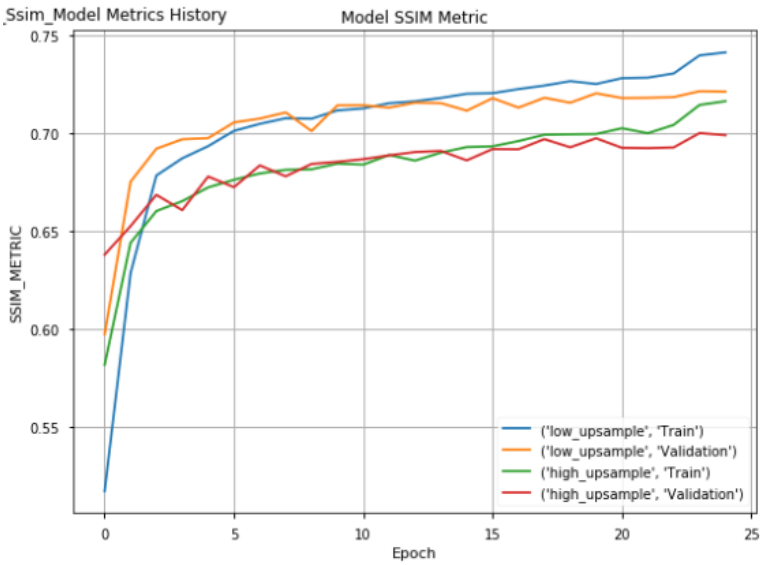
Same architecture as the previous model.

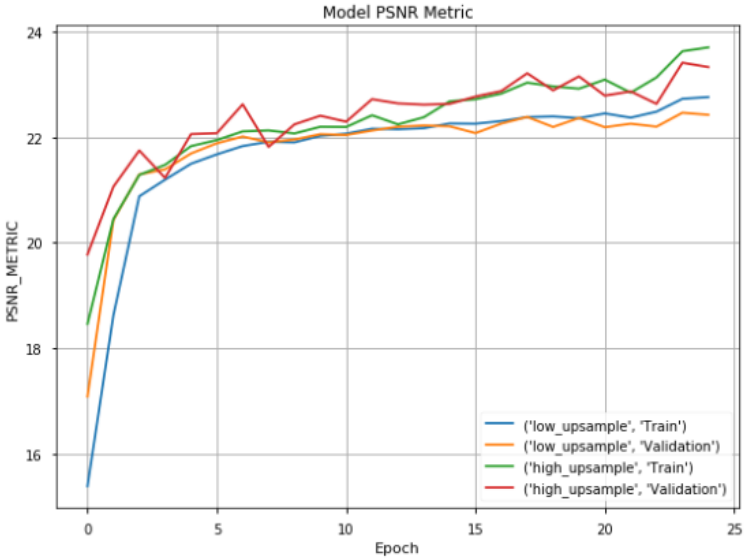
The only change is the loss function (from $-PSNR - SSIM$ to $MSE - SSIM$)

```
Model: "functional_177"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_75 (InputLayer)           [(None, None, None,  0
__________________________________________________________________________________________________
conv2d_218 (Conv2D)             (None, None, None, 6 1792        input_75[0][0]
__________________________________________________________________________________________________
functional_151 (Functional)     (None, None, None, 6 701248      conv2d_218[0][0]
__________________________________________________________________________________________________
functional_159 (Functional)     (None, None, None, 6 701248      conv2d_218[0][0]
__________________________________________________________________________________________________
functional_167 (Functional)     (None, None, None, 6 701248      conv2d_218[0][0]
__________________________________________________________________________________________________
concatenate_64 (Concatenate)    (None, None, None, 1 0           functional_151[0][0]
                                                                 functional_159[0][0]
                                                                 functional_167[0][0]
__________________________________________________________________________________________________
functional_169 (Functional)     (None, None, None, 6 38720       input_75[0][0]
__________________________________________________________________________________________________
concatenate_65 (Concatenate)    (None, None, None, 2 0           concatenate_64[0][0]
                                                                 functional_169[0][0]
__________________________________________________________________________________________________
conv2d_258 (Conv2D)             (None, None, None, 6 147520      concatenate_65[0][0]
__________________________________________________________________________________________________
tf_op_layer_DepthToSpace_8 (Ten [(None, None, None,  0           conv2d_258[0][0]
__________________________________________________________________________________________________
conv2d_259 (Conv2D)             (None, None, None, 6 9280        tf_op_layer_DepthToSpace_8[0][
0]
__________________________________________________________________________________________________
functional_175 (Functional)     (None, None, None, 6 479808      conv2d_259[0][0]
__________________________________________________________________________________________________
tf_op_layer_DepthToSpace_9 (Ten [(None, None, None,  0           functional_175[0][0]
__________________________________________________________________________________________________
low_upsample (Conv2D)           (None, None, None, 3 51          tf_op_layer_DepthToSpace_8[0][
0]
__________________________________________________________________________________________________
high_upsample (Conv2D)          (None, None, None, 3 51          tf_op_layer_DepthToSpace_9[0][
0]
==================================================================================================
Total params: 2,780,966
Trainable params: 2,742,246
Non-trainable params: 38,720
__________________________________________________________________________________________________
```









Mean Model Statistics:

`dts_dilated_conv_ext_block_ssim_model Mean Validation Metrics (144, 144)`

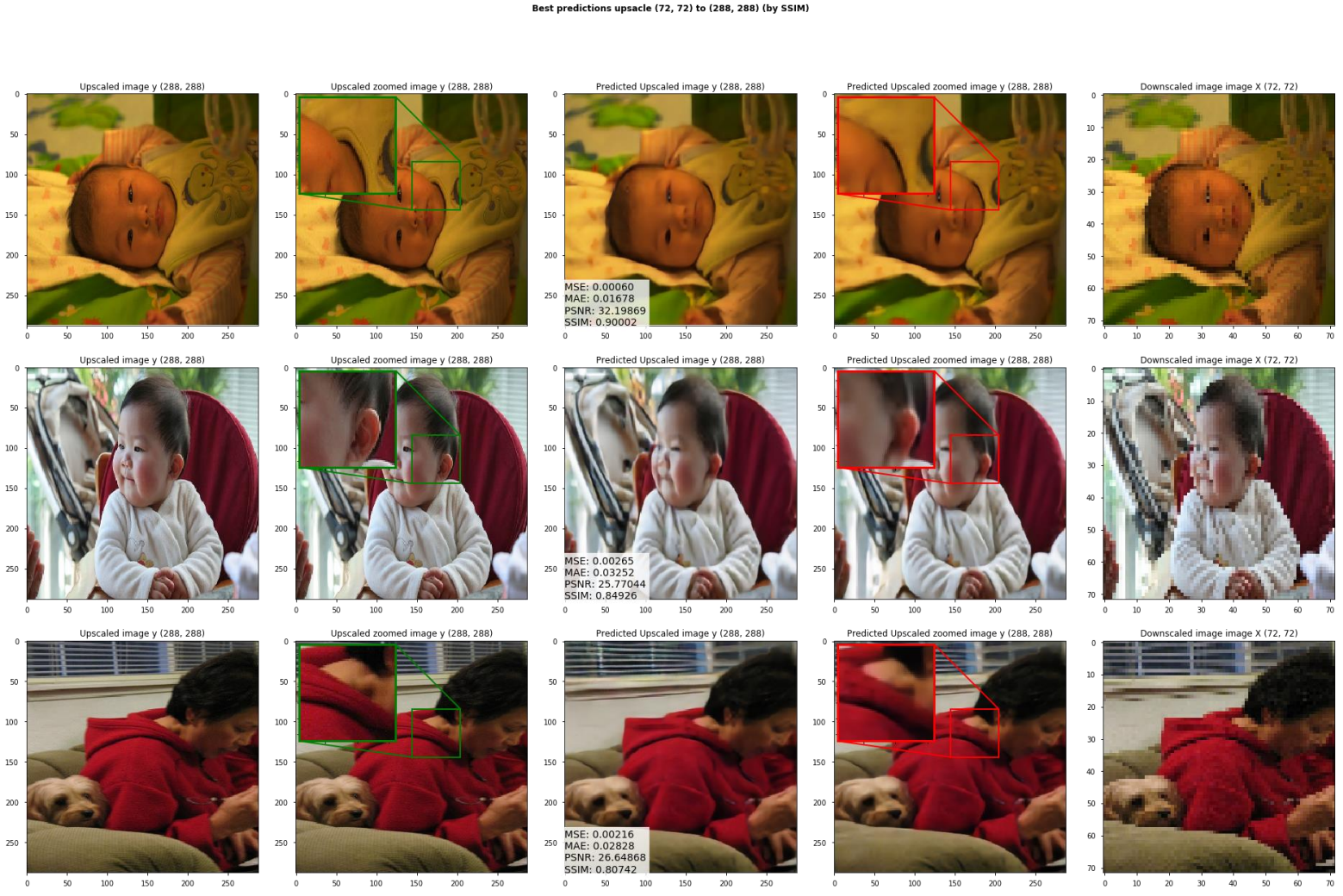
	MSE	MAE	PSNR	SSIM
0	0.006735	0.049803	22.326284	0.763142

`dts_dilated_conv_ext_block_ssim_model Mean Validation Metrics (288, 288)`

	MSE	MAE	PSNR	SSIM
0	0.005131	0.039843	23.797699	0.743382

We can see that the lower resolution (144x144x3) output we received in this case does not match the original image color space.
Although the higher resolution image is much clearer then the previous iterations.

Highest MSE metric predicted images:

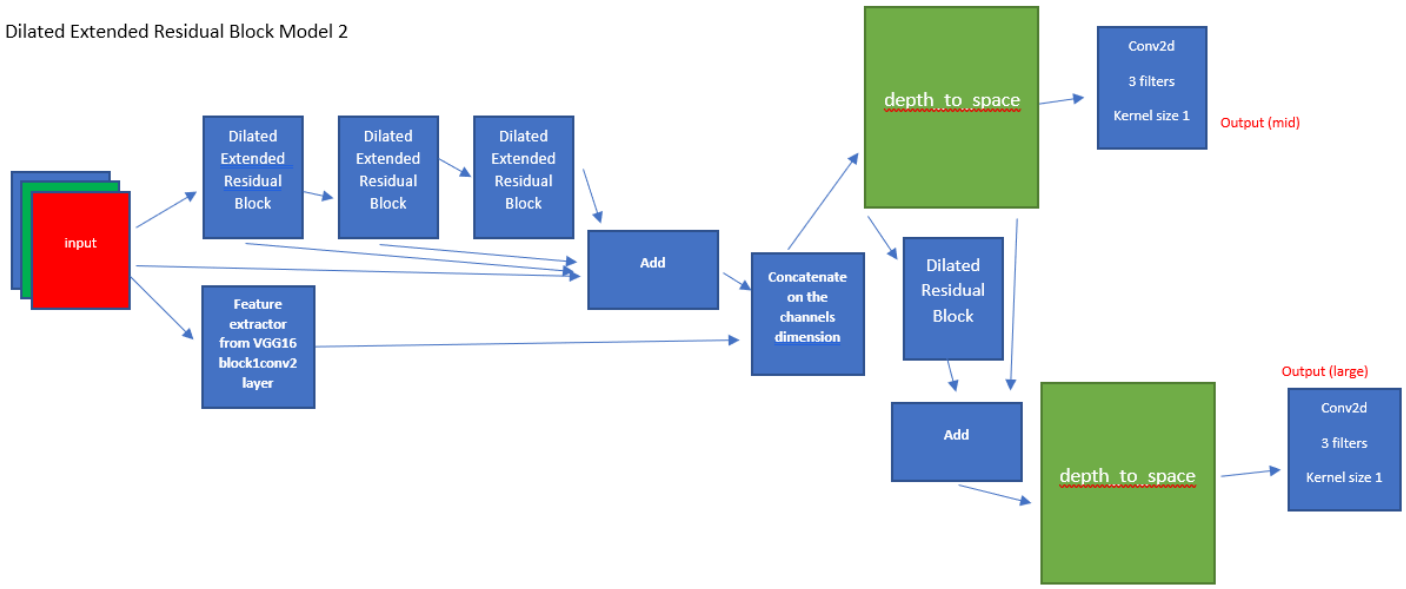


Model 4

For the final model the architecture was changes to Addition of the extended residual dialated bloc ks instead of concatenating them.
Additionally, the blocks are now connected to each other.
Loss functions changed to consider the PSNR and SSIM in a more uniform way.

Model Name: dts_dilated_conv_ext_block_ssim_psnr_model
Loss function used: $-(\boldsymbol{PSNR} / (\mathbf{1} - \boldsymbol{SSIM}))$
Model architecture:
Added addition instead of concatenation and skip connections between the dialated extended residual blocks.

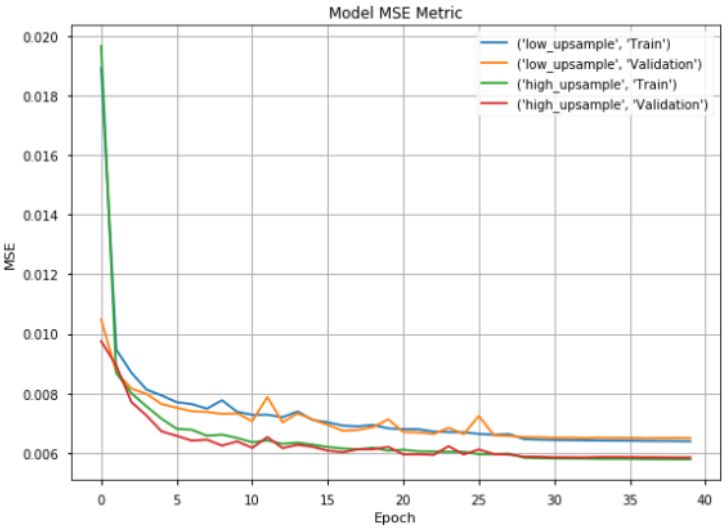
Model diagram:

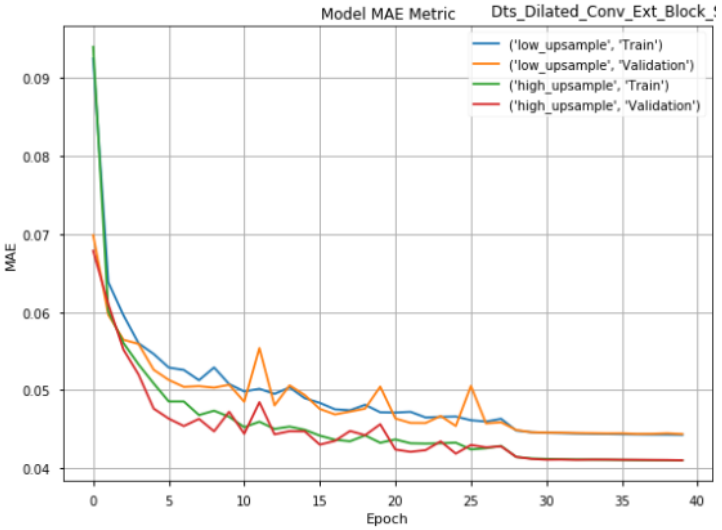


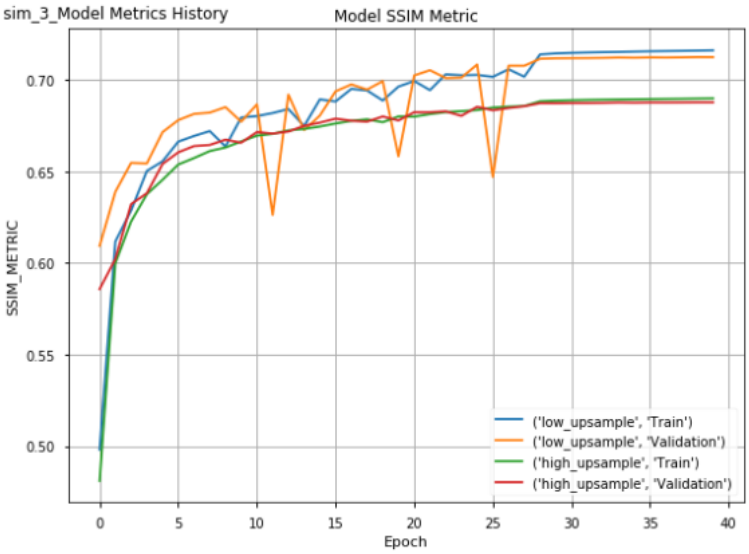
Model: "functional_251"

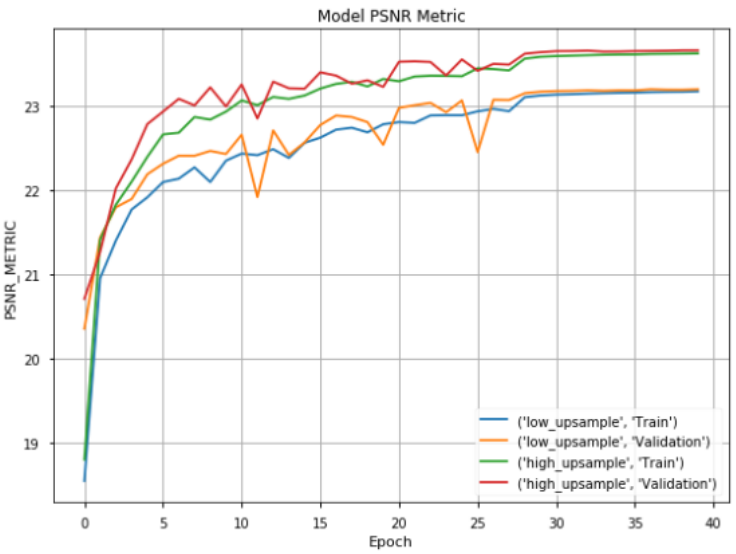
Layer (type)	Output Shape	Param #	Connected to
input_109 (InputLayer)	[(None, None, None,	0	
conv2d_331 (Conv2D)	(None, None, None, 6	1792	input_109[0][0]
functional_223 (Functional)	(None, None, None, 6	701248	conv2d_331[0][0]
functional_231 (Functional)	(None, None, None, 6	701248	functional_223[0][0]
functional_239 (Functional)	(None, None, None, 6	701248	functional_231[0][0]
add_31 (Add)	(None, None, None, 6	0	conv2d_331[0][0] functional_223[0][0] functional_231[0][0] functional_239[0][0]
functional_241 (Functional)	(None, None, None, 6	38720	input_109[0][0]
concatenate_89 (Concatenate)	(None, None, None, 1	0	add_31[0][0] functional_241[0][0]
conv2d_371 (Conv2D)	(None, None, None, 6	73792	concatenate_89[0][0]
tf_op_layer_DepthToSpace_12 (Te	[(None, None, None,	0	conv2d_371[0][0]
conv2d_372 (Conv2D)	(None, None, None, 6	9280	tf_op_layer_DepthToSpace_12[0][0]
functional_249 (Functional)	(None, None, None, 6	701248	conv2d_372[0][0]
add_33 (Add)	(None, None, None, 6	0	conv2d_372[0][0] functional_249[0][0]
conv2d_386 (Conv2D)	(None, None, None, 6	36928	add_33[0][0]
tf_op_layer_DepthToSpace_13 (Te	[(None, None, None,	0	conv2d_386[0][0]
low_upsample (Conv2D)	(None, None, None, 3	51	tf_op_layer_DepthToSpace_12[0][0]
high_upsample (Conv2D)	(None, None, None, 3	51	tf_op_layer_DepthToSpace_13[0][0]

Total params: 2,965,606
Trainable params: 2,926,886
Non-trainable params: 38,720

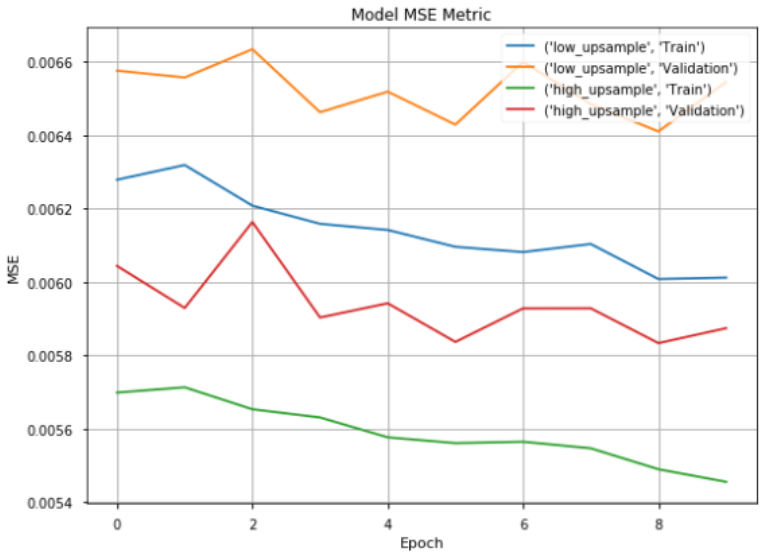


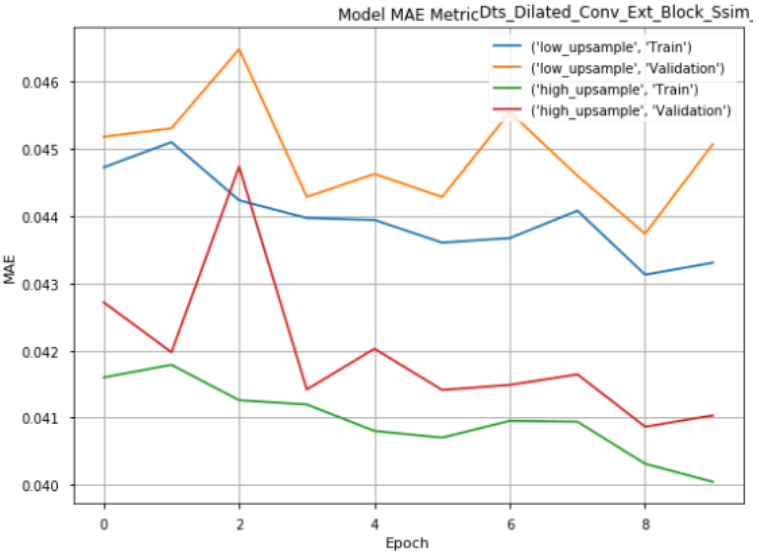


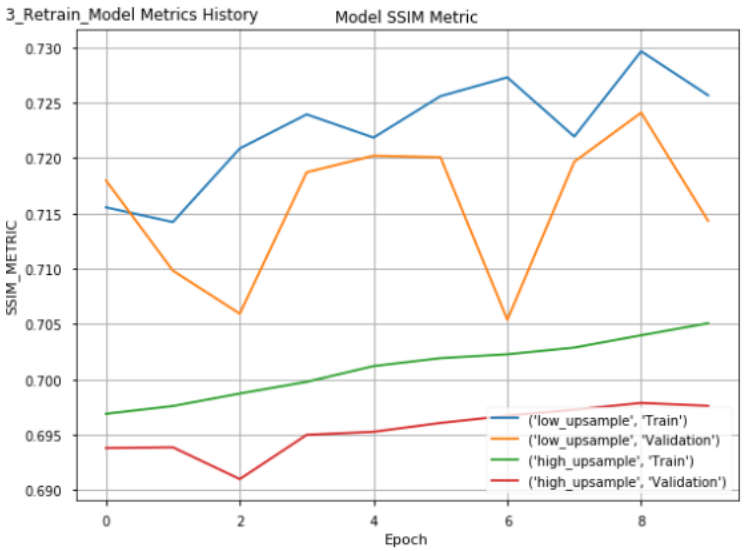


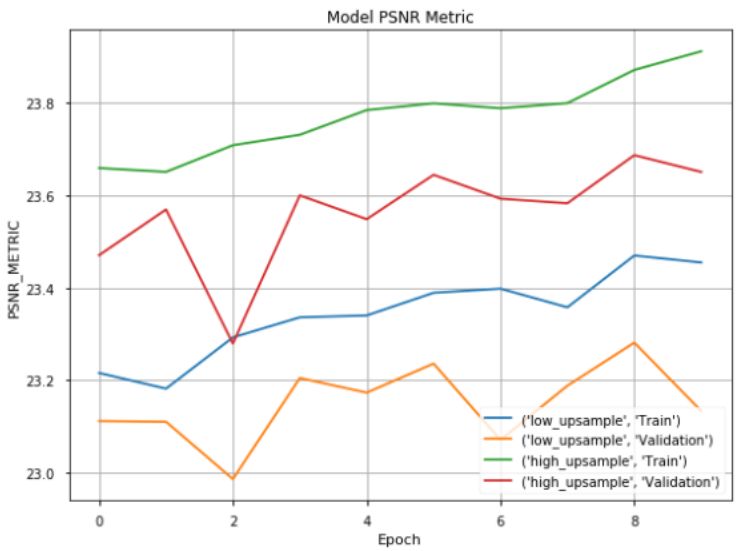


You can see that the model can still improves at this stage so we will retrain it for 10 more epochs. This time we will load 5000 images as our training data to achieve the best results possible out of the model.









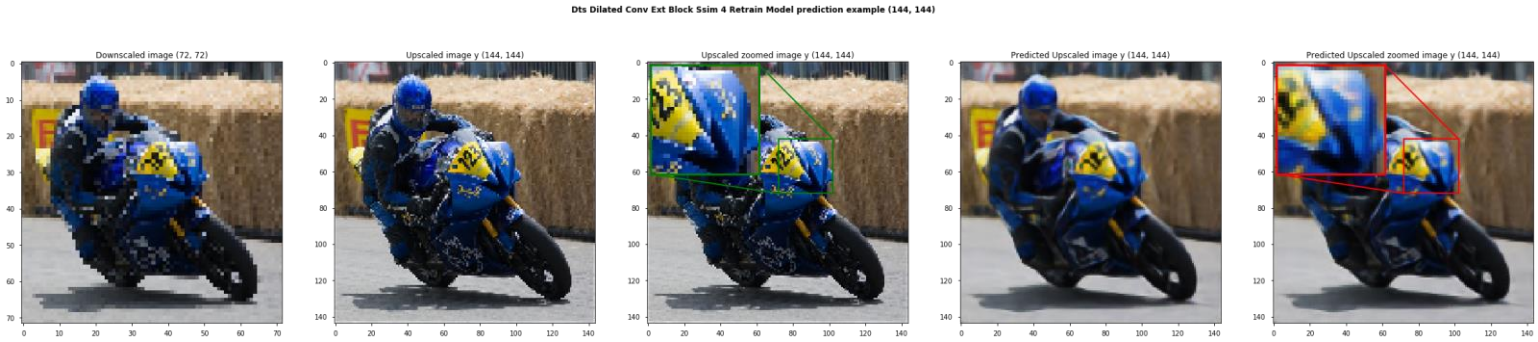
Mean Model Statistics:

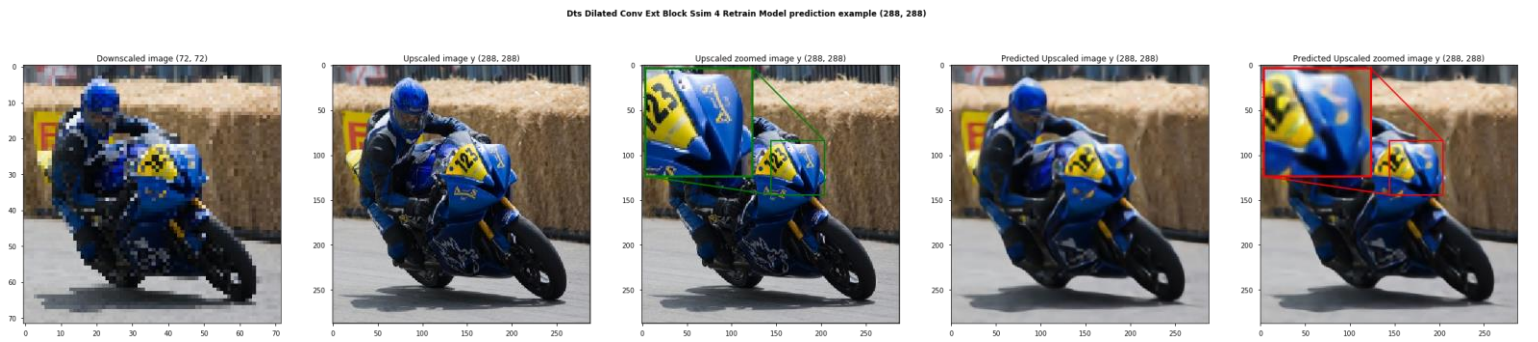
d dts_dilated_conv_ext_block_ssim_psnr_model Mean Validation Metrics (144, 144)

	MSE	MAE	PSNR	SSIM
0	0.005022	0.039502	23.981213	0.774663

dts_dilated_conv_ext_block_ssim_psnr_model Mean Validation Metrics (288, 288)

	MSE	MAE	PSNR	SSIM
0	0.004559	0.036308	24.422482	0.745073





Highest MSE metric predicted images:

Summary and Final Results

In this assignment we touched the subject of image super-resolution. Multiple techniques were shown in this notebook that eventually were combined in the final 4 models in order to achieve the best results possible.
we can determine that we managed to achieve good results overall.
The most substantial factor that caused the biggest improvement was using PSNR and SSIM losses combined in order to restore the most accurate image in terms of color, saturation, luminance and structure.

We can clearly see that by improving PSNR and SSIM metrics also resulted in improvements in MSE and MAE metrics.
By adding more images to our training data, the learning rate of the optimizer had to be lowered to 0.0001 since the default learning rate caused the model to diverge.

Below you can see the final metric results on each individual model and the final visual results compared of the last 4 models that were trained on a substantial amount of the training data.
Additionally, a zip file named 'gifs' was attached to this assignment containing GIF images visualizing the training process by showing the predicted image for a set of 3 constant images with the number of epoch the prediction was made in.

Future Improvements

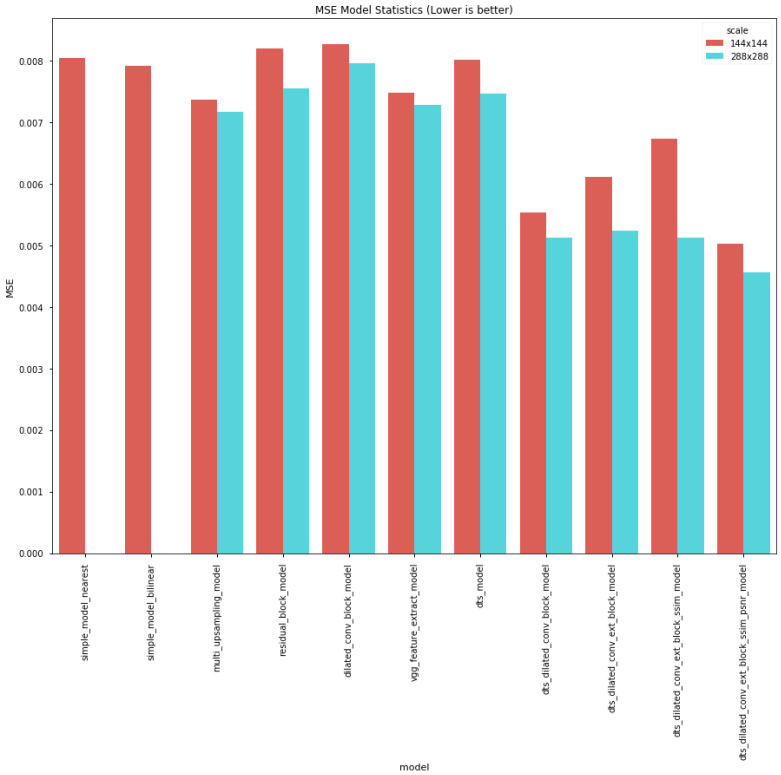
Ideas for further exploration in order to achieve even better results that can be taken:

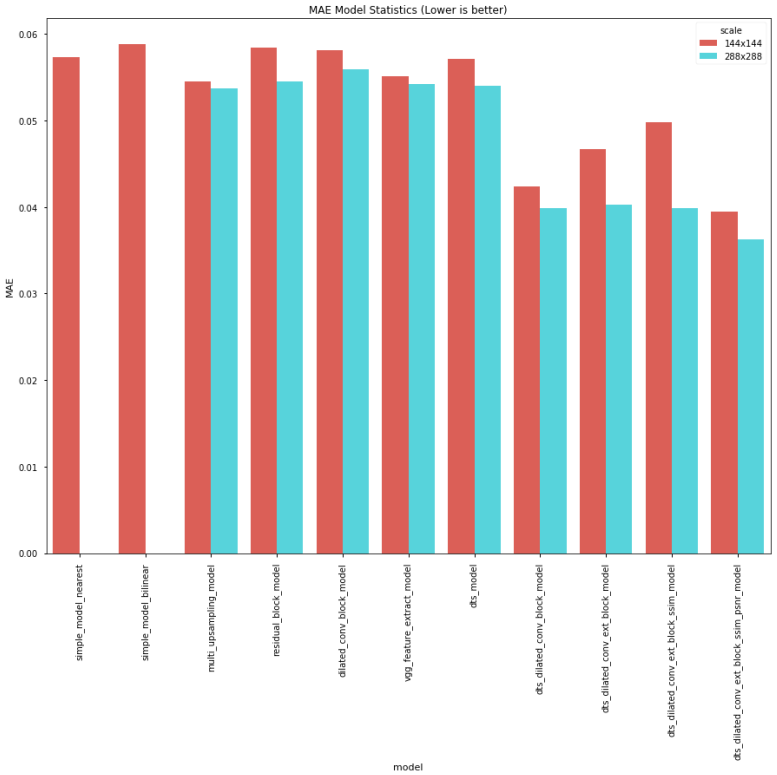
- Increasing the training data and training for more epochs.
- Add more residual blocks with skip connections to the model.
- Try using a different feature extraction block instead of VGG16 (e.g. ResNet) or using a different layer from the existing FE model for feature extraction.
- Try using no feature extraction model.
- Try Generative adversarial network (GAN) for the task since it was used in many cases for the super-resolution task on had good outcomes (e.g. SRGAN)

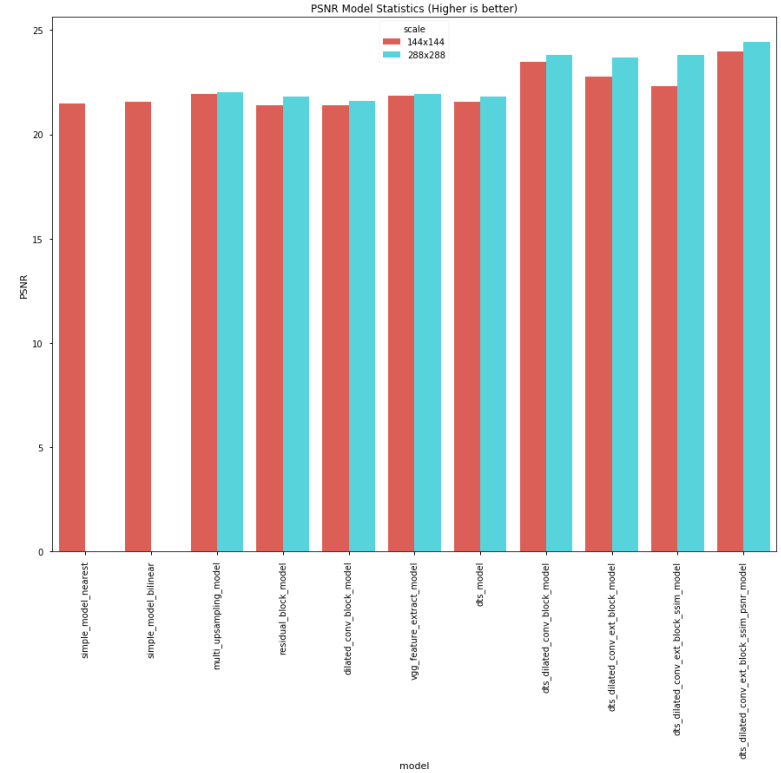
Metric statistics table

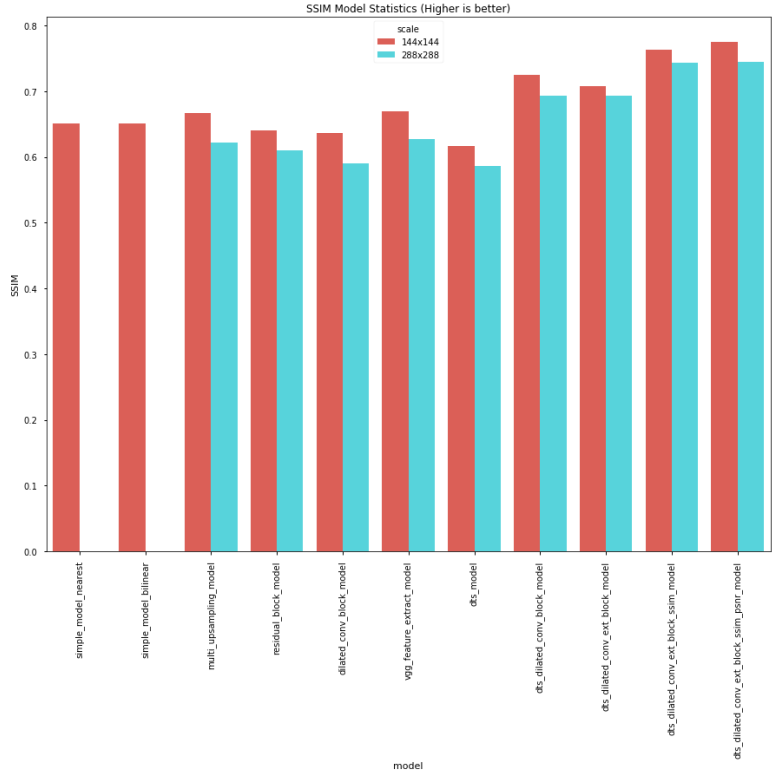
	MSE	MAE	PSNR	SSIM	model	scale
0	0.008042	0.057368	21.477512	0.650988	simple_model_nearest	144x144
1	0.007910	0.058852	21.545467	0.651205	simple_model_bilinear	144x144
2	0.007371	0.054544	21.931559	0.667144	multi_upsampling_model	144x144
3	0.007163	0.053710	22.037392	0.622454	multi_upsampling_model	288x288
4	0.008201	0.058387	21.378817	0.640398	residual_block_model	144x144
5	0.007551	0.054517	21.801100	0.609584	residual_block_model	288x288
6	0.008272	0.058157	21.397940	0.636006	dilated_conv_block_model	144x144
7	0.007961	0.055945	21.623392	0.590215	dilated_conv_block_model	288x288
8	0.007484	0.055156	21.853580	0.669549	vgg_feature_extract_model	144x144
9	0.007276	0.054209	21.941778	0.627485	vgg_feature_extract_model	288x288
10	0.008020	0.057105	21.548134	0.616974	dts_model	144x144
11	0.007464	0.054004	21.831583	0.586844	dts_model	288x288
12	0.005536	0.042337	23.492382	0.725306	dts_dilated_conv_block_model	144x144
13	0.005126	0.039911	23.820621	0.693821	dts_dilated_conv_block_model	288x288
14	0.006113	0.046698	22.786966	0.707361	dts_dilated_conv_ext_block_model	144x144
15	0.005233	0.040234	23.695541	0.693150	dts_dilated_conv_ext_block_model	288x288
16	0.006735	0.049803	22.326284	0.763142	dts_dilated_conv_ext_block_ssim_model	144x144
17	0.005131	0.039843	23.797699	0.743382	dts_dilated_conv_ext_block_ssim_model	288x288
18	0.005022	0.039502	23.981213	0.774663	dts_dilated_conv_ext_block_ssim_psnr_model	144x144
19	0.004559	0.036308	24.422482	0.745073	dts_dilated_conv_ext_block_ssim_psnr_model	288x288

Metric statistics charts



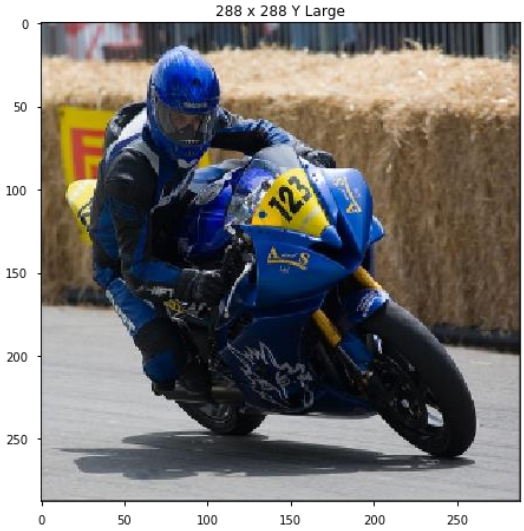


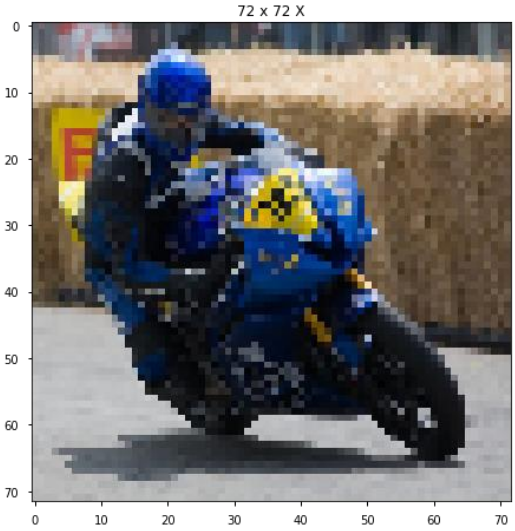




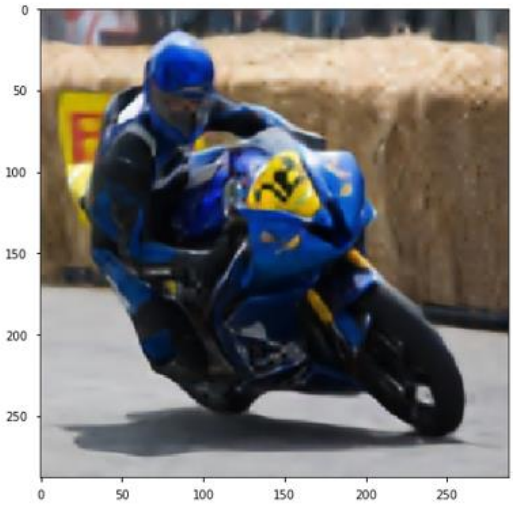
Best visual and metrics outcome

Original images (left – 288x288x3 resolution, right – downscaled 72x72x3)





Predicted reconstructed image outcome (288x288)

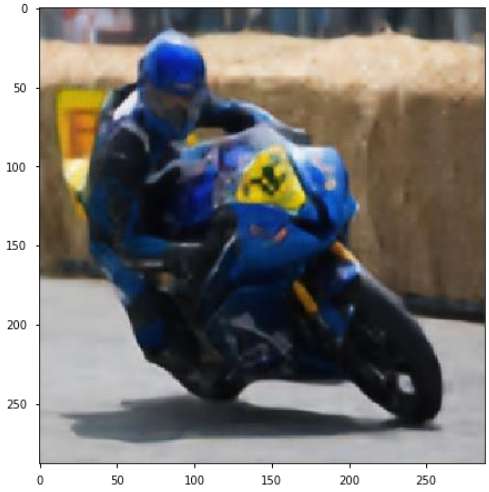


dts_dilated_conv_ext_block_ssim_psnr_model

MSE: 0.00456
MAE: 0.03631
PSNR: 24.42248
SSIM: 0.74507

Loss: $-(PSNR\ /\ (1\ -\ SSIM))$

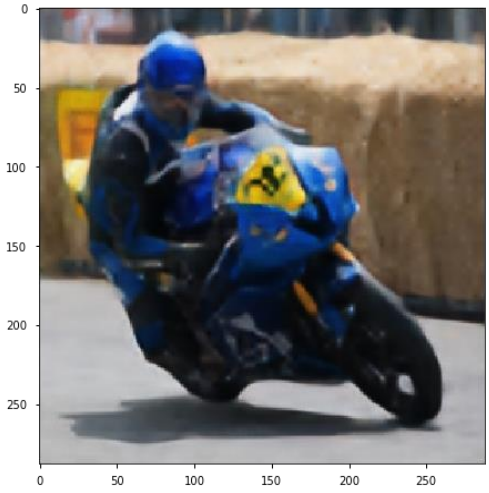
Model iteration process results



dts_dilated_conv_block_model

MSE: 0.00513
MAE: 0.03991
PSNR: 23.82062
SSIM: 0.69382

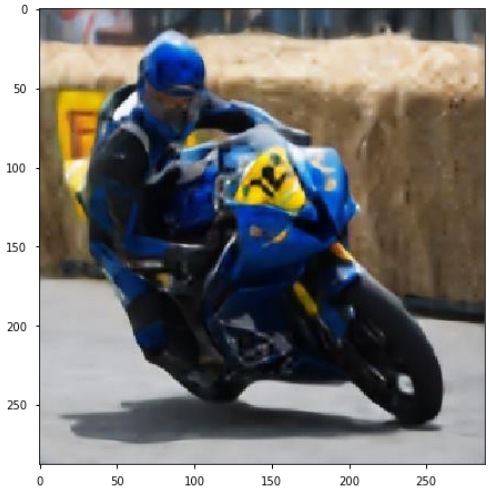
Loss: $-PSNR$



dts_dilated_conv_ext_block_model

MSE: 0.00523
MAE: 0.04023
PSNR: 23.69554
SSIM: 0.69315

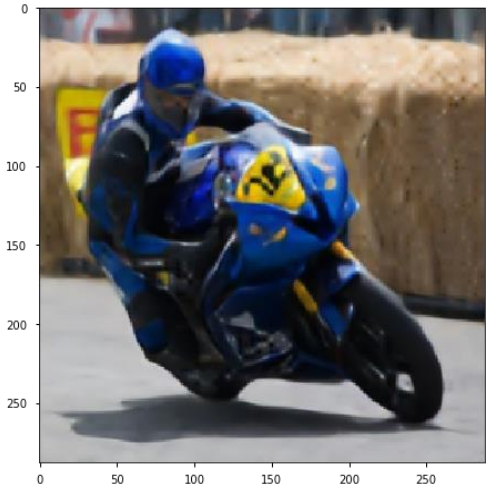
Loss: $-PSNR - SSIM$



dts_dilated_conv_ext_block_ssim_model

MSE: 0.00513
MAE: 0.03984
PSNR: 23.79770
SSIM: 0.74338

Loss: $MSE - SSIM$



dts_dilated_conv_ext_block_ssim_psnr_model

MSE: 0.00456
MAE: 0.03631
PSNR: 24.42248
SSIM: 0.74507

Loss: $-(PSNR\ /\ (1\ -\ SSIM))$