Image Super-Resolution Exercise - Report

We present here a comparison of the different models in the exercise.

Six different models were trained, using a custom PSNR loss function. The loss was calculated on a batch wide basis in a weighted fashion, with the loss of the high resolution images weighted as 0.75 and the loss of the medium resolution images weighted as 0.25:

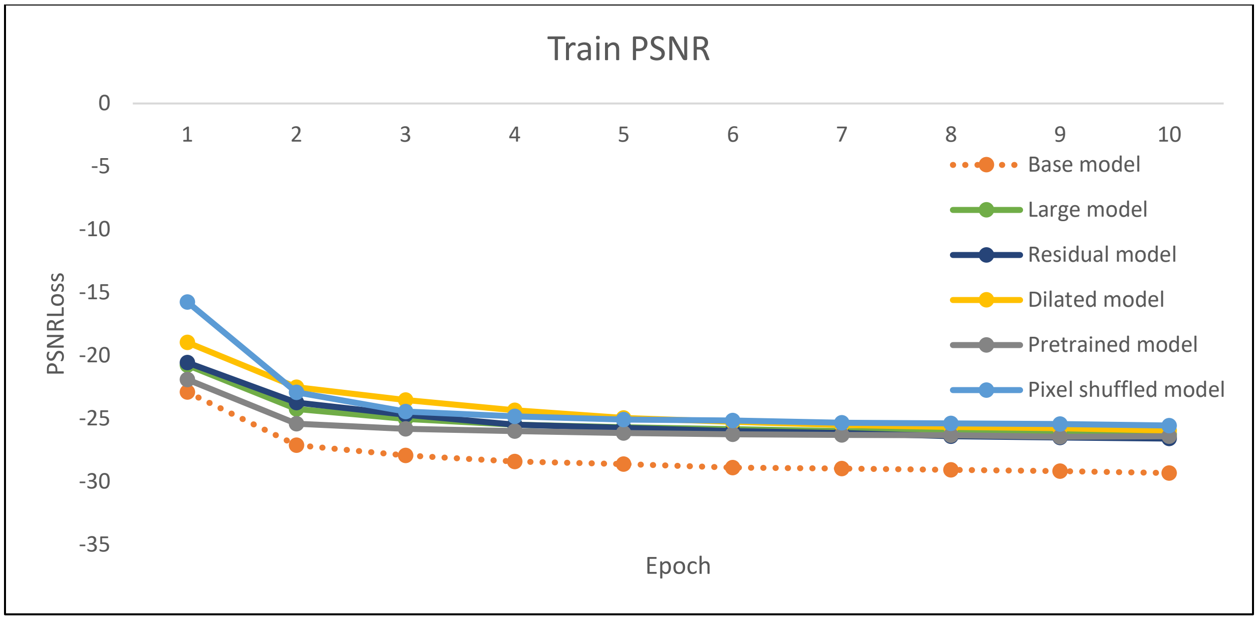
1. Base model – trained only on low resolution images and producing medium resolution output, according to the architecture outlined in the exercise.
2. Large model – an extended version of the base model which produces both medium and high resolution output (by adding a second upsampling block to the previous, base architecture).
3. Residual model – this model utilized three residual blocks, as outlined in the exercise, but kept the basic Large model architecture.
4. Dilated model – same architecture as the residual model, but using dilated (Atrous) convolutional blocks instead of residual blocks.
5. Pretrained model – this model used the same architecture as the Large model, however prior to the first upsampling, features from the second convolutional layer in the first block of VGG16 were added as additional channels to the results of the second convolutional layer in the model. Data in the VGG channel was normalized to mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]. The VGG layer weights were frozen in this model.
6. Pixel shuffled model – same architecture as the pretrained model, but upsampling layers were replaced by pixel shuffling.

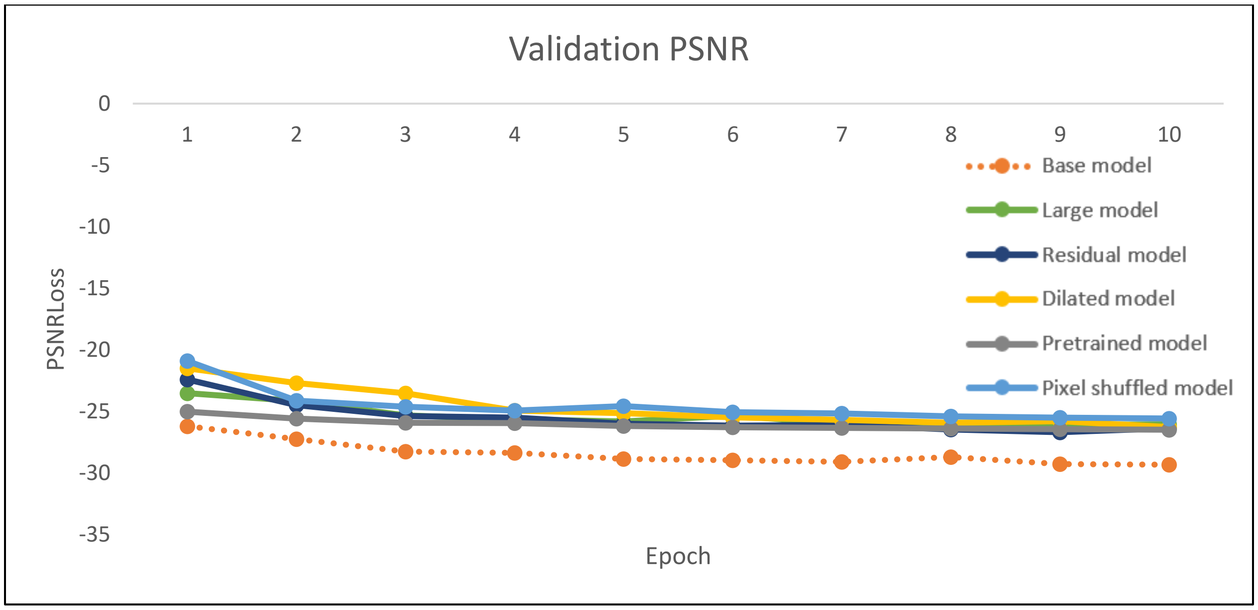
In all models leaky relu was used as the activation function after each convolutional block except for the final, output convolutions, which were followed by a sigmoid activation function.

All models, but the pixel shuffle model, were trained with learning rate of 0.0005. For the pixel shuffle model a learning rate of 0.001 was used.

Overall, in the current scope the models reached very similar performance overall, as can be seen both in the reported measures and in the attached visual examples. It is possible that additional hyperparameter tuning and optimization for each model would reveal differences between them.

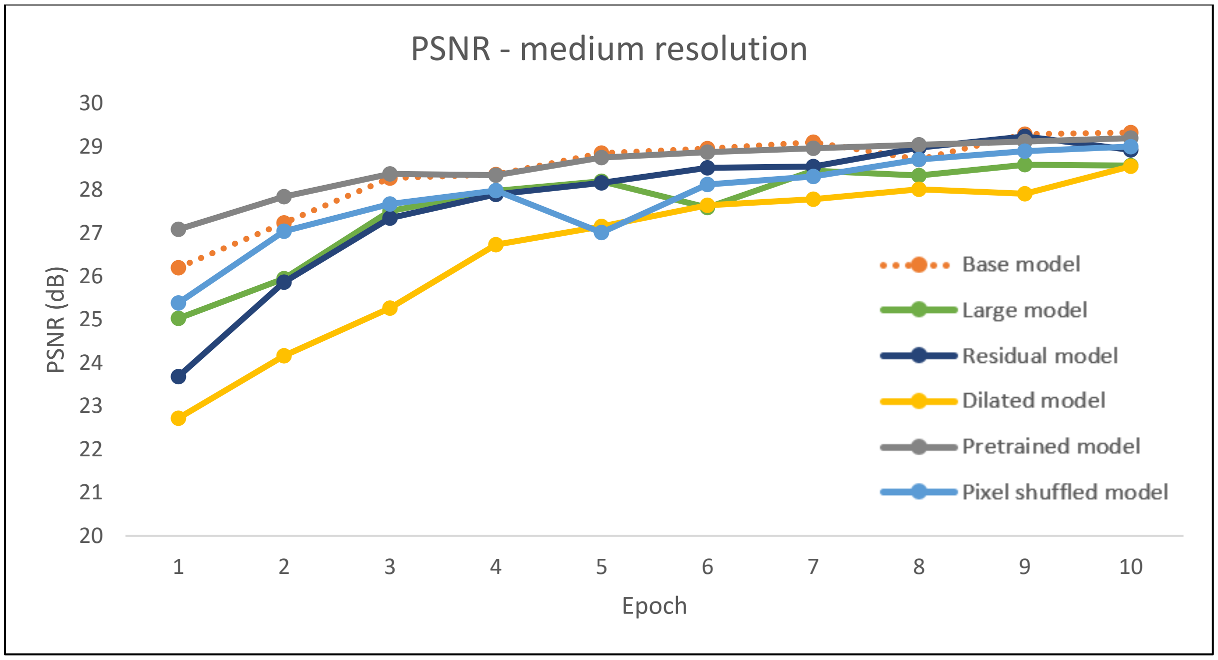
First, we show the training and validation loss for the models. A custom PSNR loss function was used in this case, as described above. Note that from all models except for the base model, both training and validation losses reflect a weighted average of the PSNR in the medium and high resolution output. Thus, the comparison with the base model for this measure is not relevant, and the base model is included in these figures only for visual assessment of the convergence rate. Furthermore, note that the negative weighted PSNR appears in these graphs, as the negative weighted PSNR was minimized during training. Although we trained the models on 100 epochs, we display only the first 10, were the greatest differences between models were observed.

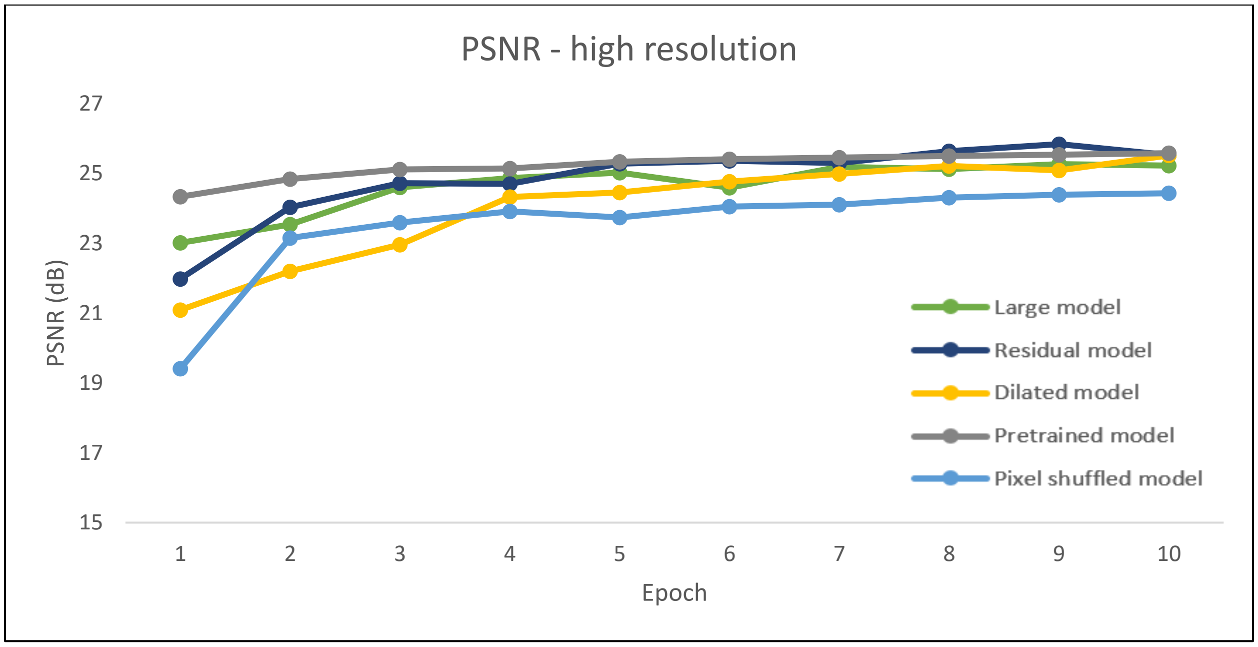




Plots comparing train and validation losses directly for each model can be found in the notebook. Here we present only the model comparison.

Next we display the PSNR of the resulting images from the validation set (non-weighted), separately for the medium and high resolution images (therefore, the base model, which was trained to produce only medium resolution images, appears in the first figure only).

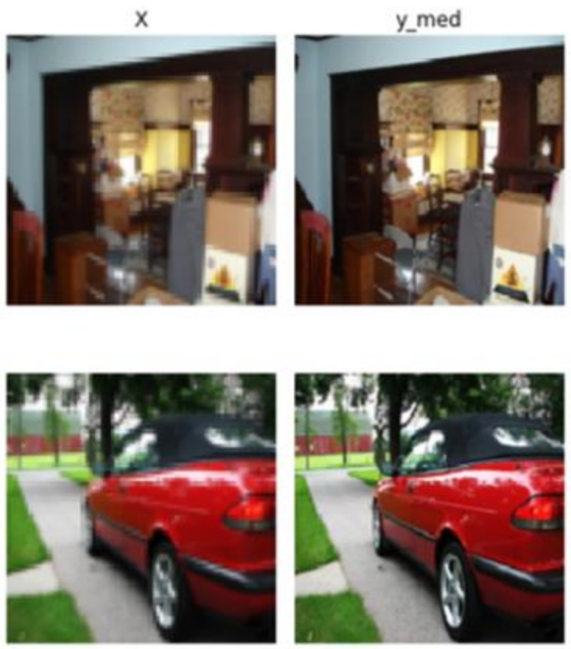


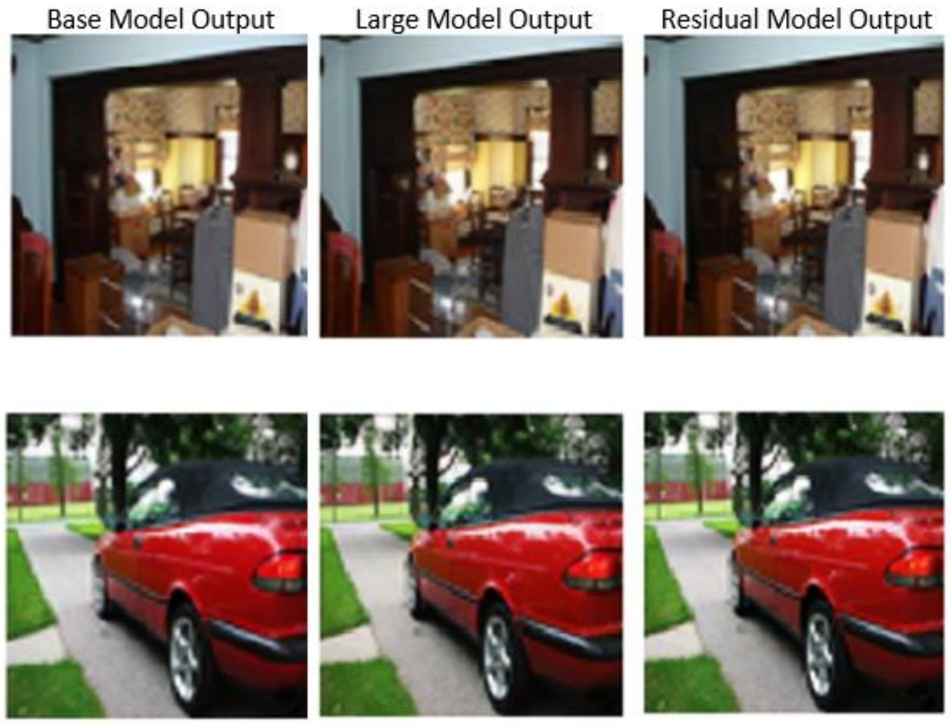


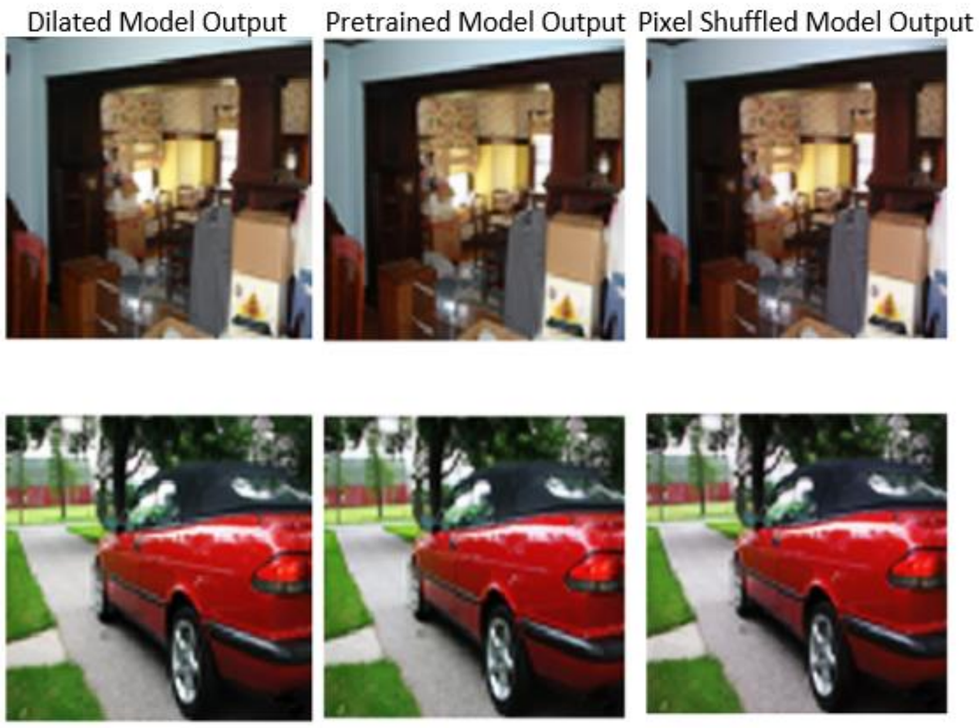
In both of these figures it seems that the model which included pretrained features from VGG16 converged a bit quicker than the other models, and was one of the top preforming models for both medium and high resolution output. Nonetheless, the differences between the models presented here are small overall.

Finally, we present visual examples from all of the models, for medium and high resolution outputs (where applicable) on images from the validation set. Overall, visually, is seems to us that it is difficult to observe differences between the models with the naked eye:

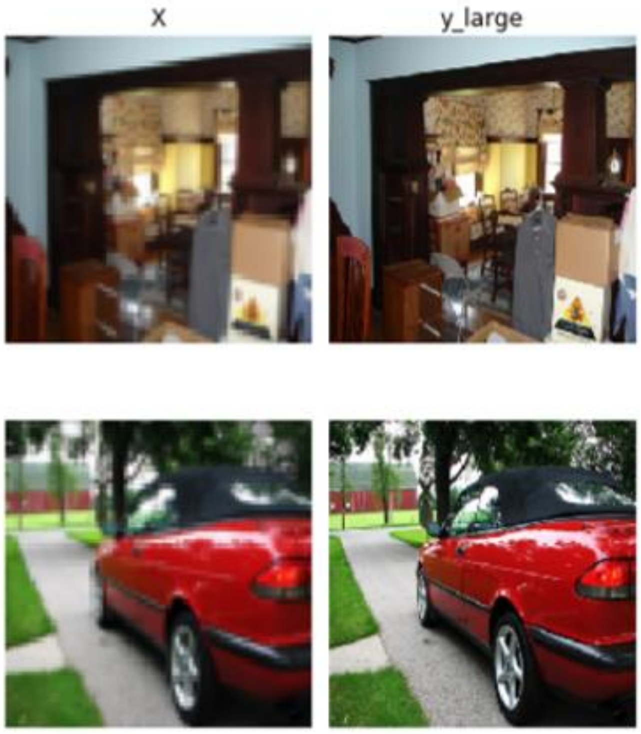
Medium results:

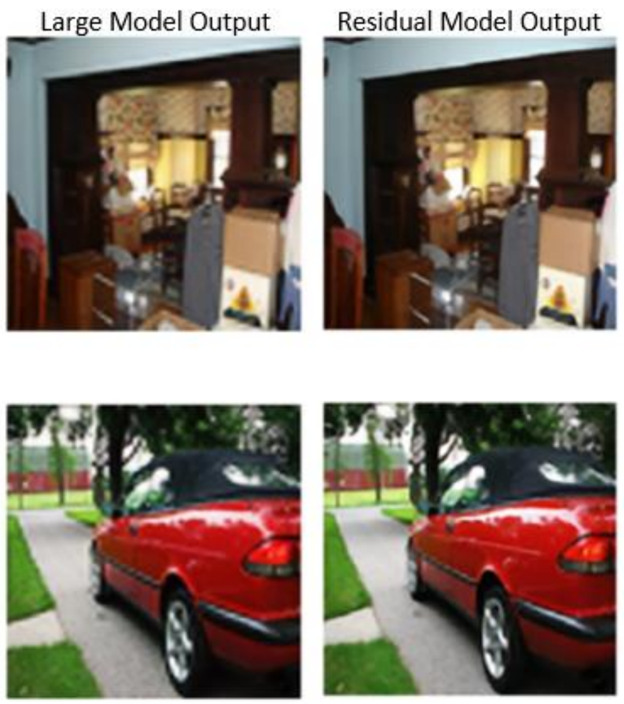


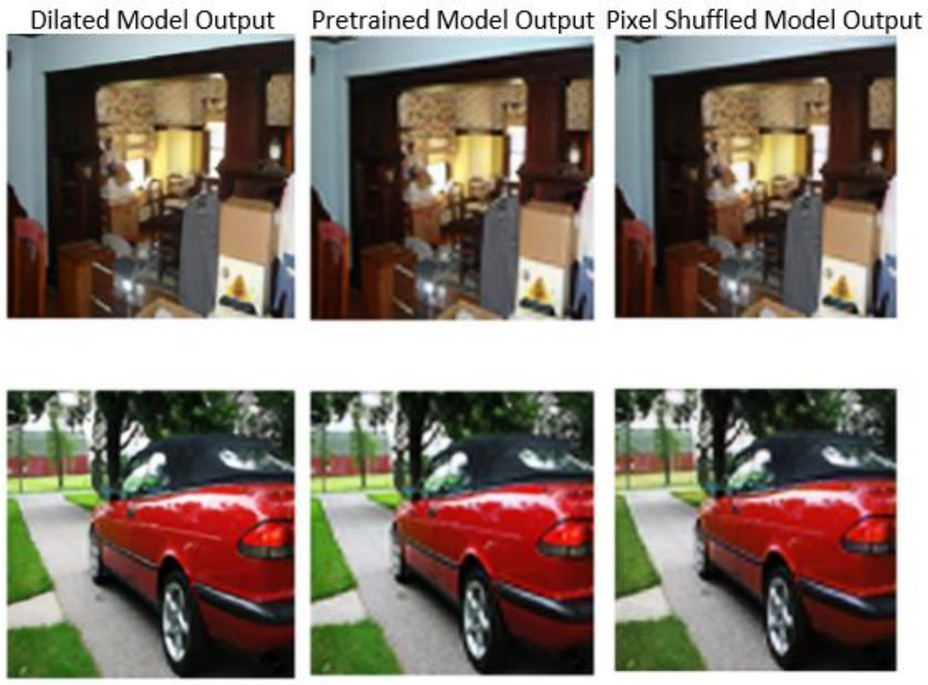




High results:







In summary, as mentioned, all models perform relatively well on the task and produce results that look very good visually. Additional hyperparameter tuning may outline differences between the models in a better way, and perhaps optimize performance as well.

During the exercise we experimented in the preliminary stages (on a small subset of the trainset, for a small number of epochs) both the MSE and MAE Losses and tried different combinations of static weighted averages (not adaptive). In our setting none yielded better performance than the others.

We also experimented with the relu activation function during development, but then switched to the leaky relu. We did observe noticeable improvements when using leaky relu.

During development we faced the issue of “burnt pixels” (pixels with values out the range [0.0, 1.0]). Trying to clip the out of bounds values within the models only degraded performance, probably because the resulting error was smaller than it was supposed to be). We didn’t want to clip the values outside of the models, so in the end we added the sigmoid function to force the values to be in the desire range. Although it improved the perceptual performance observed by us, it wasn’t manifested in the metrics.

As stated above, we trained all models for 100 epochs. Additional epochs would have improved performance as evident by the slopes of the learning curves.