Super Resolution for Satellite Imagery

Mitchell Clark · Nazeef Hamid · Alex Kenna
Limao Chang · Emmanuel Skoufris

1 Introduction

Super-resolution is the generative machine learning task of enhancing image resolution while retaining important visual structures and features. Many state-of-the-art (SOTA) computer vision models, such as Generative Adversarial Networks (GANs) (Ledig et al., 2017), autoencoders (Dong et al., 2015), and diffusion models (Ho, Jain, and Abbeel, 2020), have been successfully applied to super-resolution tasks. These tasks have applications in many domains, such as medical imaging, computer vision, and satellite imagery (Froede, 2023). In this report, we explore these SOTA models on satellite imagery from the Sentinel-2 and VENµS satellites.

High-resolution satellite imagery can be used for monitoring land cover changes, deforestation, and disaster response, among many other applications (Njambi, 2023). As such, it would be desirable to have a satellite that combines the strengths of both Sentinel-2 and VENµS and captures high-resolution imagery at a global scale. However, deploying a new satellite with these attributes would be costly. Instead, we apply SOTA models to bridge this gap and simulate globally available high-resolution satellite imagery.

The SEN2VENµS dataset is an open-data licensed dataset of landscape images (patches) from the Sentinel-2 and VENµS satellites which have been physically aligned and have undergone pre-processing stages. Sentinel-2 has a global range but limited resolution, whereas VENµS has a limited range, but is capable of capturing images at a higher resolution. More specifically, the Sentinel-2 images have a resolution of 128×128 pixels, whereas the VENµS images have a resolution of 256×256 pixels, so our super resolution task has an upscaling factor of two.

While there are papers in the literature (Lanaras et al., 2017; Lin and Bioucas-Dias, 2020; Paris, Bioucas-Dias, and Bruzzone, 2017) that have applied super-resolution to imagery from these satellites individually, we will need to use imagery from both satellites to upscale the Sentinel-2 imagery to the resolution capability of VENµS as performed between Sentinel-2 and Rapid-Eye (Galar et al., 2019), which is a fairly novel task for our satellites (Michel et al., 2022).

We make our code freely available on GitHub[1].

2 Methods

We explore three SOTA models that have seen success in various super-resolution tasks: an autoencoder, a GAN, and a diffusion model.

2.1 Autoencoder

Substantial progress in super-resolution was made by Dong et al. (2015) with the introduction of Super-resolution Convolutional Neural Network (SRCNN), which we adapt and refer to as the autoencoder in this paper. Unlike previous sparse-coding methods that involve explicitly learning and constructing low-resolution and high-resolution dictionaries, the autoencoder is an end-to-end, feed-forward mapping implemented as a CNN. Figure 1 depicts the autoencoder's architecture. Notably, since SRCNN was introduced, more recent work has led to deeper variants of the network and increased its complexity (Wang et al., 2021a). Nonetheless, we chose to base our autoencoder on the original SRCNN due to its already impressive performance, and because our super-resolution task only has an upscaling factor of two.

We now discuss the details of the autoencoder. Before feeding them through the network, the low-resolution 128×128 images are first prepared by upsampling them to a resolution of 256×256 using bicubic interpolation.

[1] `https://github.com/LimaoC/super-resolution-for-satellite-imagery`

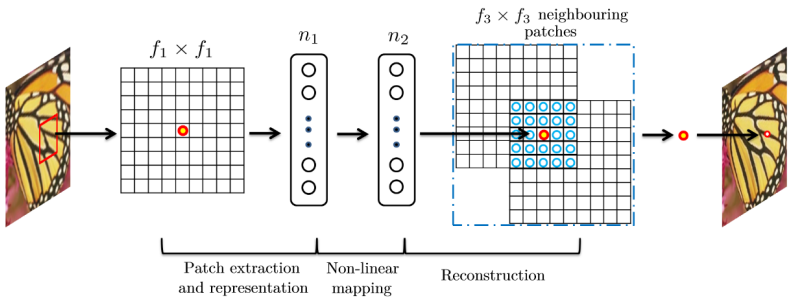


Figure 1: The autoencoder's architecture. Each mapping can be expressed as a convolutional layer (Dong et al., 2015).

1. **Patch embedding.** The first layer consists of n_1 filters with a kernel size of f_1. By convolving the filters with each appropriately sized patch across the three colour channels, the overlapping patches in the image are embedded into a n_1-dimensional space, with each dimension corresponding to a filter. ReLU activation and subsequently a bias term are applied to the output. This is the encoder part of the autoencoder.
2. **Nonlinear embedding.** n_2 filters with kernel size f_2 are applied to the patches in the first feature space. This further embeds the patches into a lower dimensional space (if $n_2 < n_1$). Once again, ReLU activation and bias are applied to the output.
3. **Reconstruction.** In this phase, three filters with filter size f_3 are applied to the previous output to obtain the reconstructions. A bias term is also included. In this way, the predictions in the second feature space are combined in a spatial neighbourhood to produce the final super-resolved image.

For our model, we set $n_1 = 64, f_1 = 9, n_2 = 32, f_2 = 5$, and $f_3 = 5$. For each convolutional layer, sufficient padding is applied to ensure that the output size of each hidden layer remains constant throughout.

Sparsity in super-resolution has been found to improve performance and efficiency. This is because low-resolution images may contain flat regions with little interesting detail. Consequently, it may not be computationally efficient for a model to learn how to upscale these uninteresting regions. The autoencoder does to an extent encourage sparsity through the use of ReLU activation which, combined with its relatively lightweight architecture and impressive performance, motivates its use in this work. Interestingly, recent work has advanced the use of sparsity in SR convolutional networks, with promising results (Wang et al., 2021a). Due to time constraints, such ideas could not be implemented.

2.1.1 Perceptual loss

The naive loss function to use for training the autoencoder is pixel-wise Mean-Squared Error (MSE). More precisely, if $\boldsymbol{y} = \{\boldsymbol{y}_i\}_{i=1}^n$ are the ground truth high-resolution images, and $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^n$ the corresponding super-resolved images, then the MSE between these batches is given by

$$\text{MSE}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2. \tag{1}$$

Although the MSE has highly desirable analytic properties like symmetry, differentiability and convexity, it fails to quantify more global, structural differences between images that influ-

ence their visual quality. For instance, blurring an image causes a significant perceptual change but results in only a small change in MSE, which can be problematic for super-resolution tasks (Zhang et al., 2018).

Johnson, Alahi, and Fei-Fei (2016) proposed an innovative way of circumventing the shortcomings of pixel-wise MSE that leverages the ideas of transfer learning. They suggest that computing MSE between high-level features in the images will more effectively optimise the super-resolution process for human vision. Specifically, the high-level features of the images are extracted by feeding them through a pre-trained VGG-16 network (Figure 2), a deep CNN trained on ImageNet for image classification.

Letting ϕ_j denote the activation of the jth block, the perceptual loss is then computed as:

$$L(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \sum_{j=1}^{4} \frac{1}{C_j H_j W_j} \|\phi_j(\boldsymbol{x}_i) - \phi_j(\boldsymbol{y}_i)\|_2^2, \quad (2)$$

where $C_j \times H_j \times W_j$ is the shape of the jth block's feature map. We only sum over the first four blocks, as the last block does not capture seminal information. As illustrated, the perceptual loss between a batch of images is merely the sum of the MSE between the images' features.

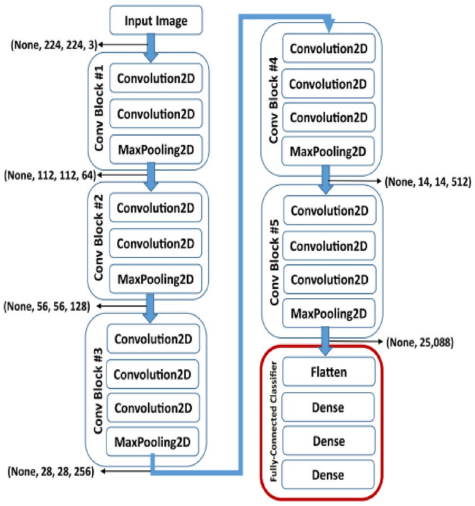


Figure 2: VGG-16 architecture (Theckedath and Sedamkar, 2020)

2.2 SRGAN

The Super-Resolution Generative Adversarial Network (SRGAN) (Ledig et al., 2017) features a CNN generator and discriminator. The generator uses sub-pixel convolution (Shi et al., 2016) to upsample images. Sub-pixel convolution generates the additional pixels required for the super-resolution image in the form of additional output channels and then shuffles these back into the shape of an image. For our problem, we need to map 3×128×128 patch tensors to 3×256×256 corresponding with a factor of two along the width and height. If we apply 3×2×2 filters in a convolutional layer we will acquire some 12×128×128 output. We reshape this output into our high-resolution space, shifting pixel data from the 12 channels into a 3×256×256 tensor.

As in the typical GAN framework, training is equivalent to a mini-max game where the generator tries to create high-resolution images that fool the discriminator into thinking they are from the ground truth set. For $i = 1, 2, \ldots, n$ samples, let $\boldsymbol{x}_i$ be the tensor of the ith super-resolved output patch, $\boldsymbol{y}_i$ be the tensor of the ith ground-truth high-resolution patch, p_i be the discriminator's predicted probability that the super-resolved patch i comes from the ground-truth set. Let β be a hyperparameter weighing the contribution of the discriminator's loss. The generator optimizes the objective function L in Equation (3) that includes an MSE and a binary-cross-entropy term from the discriminator to optimize for both the raw pixel accuracy and image sharpness respectively (Ledig et al., 2017).

$$L = \frac{1}{n} \sum_{i=1}^{n} ||\boldsymbol{x}_i - \boldsymbol{y}_i||_2^2 - \beta \frac{1}{n} \sum_{i=1}^{n} \log(p_i) \quad (3)$$

The discriminator optimizes the standard binary-cross-entropy loss.

Our generator architecture can be seen in Figure 3. In our generator, we use a convolutional pipeline with residual blocks to learn deep features of the images before applying the so-called

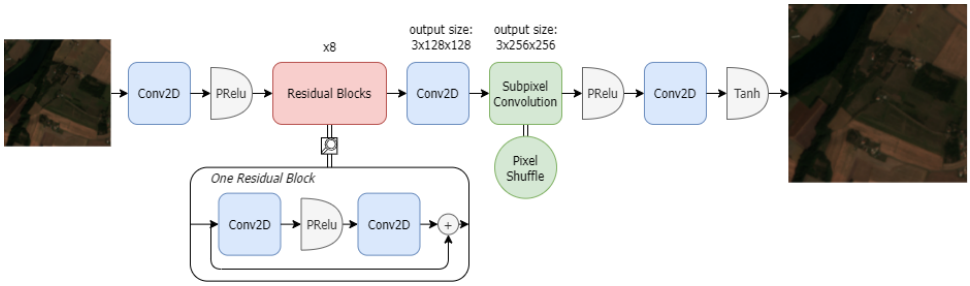


Figure 3: SRGAN Generator Architecture

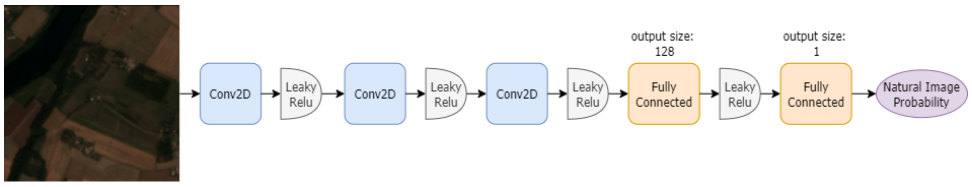


Figure 4: SRGAN Discriminator Architecture

sub-pixel convolution to increase resolution. We use skip-connections in our residual blocks to allow deeper learning without vanishing gradients (He et al., 2015a) and PReLU for the practical benefits of ReLU while avoiding the dying ReLU problem (He et al., 2015b). We learn features in a low-resolution space for computational efficiency (Shi et al., 2016). We have one final convolutional mapping with a tanh activation which produces our super-resolved output.

The final tanh activation layer is a remnant of the original architecture, chosen to prepare the output for the VGG model to use perceptual loss. We could not successfully apply perceptual loss to this model so a sigmoid activation layer may have been more appropriate, but we briefly experimented with this to little success.

We exclude batch normalisation from any generator layers. Batch normalisation generates more artifacts (Wang, Dong, and Shan, 2022) and has been shown empirically to reduce the variability of pixel values which is not typically appropriate for super-resolution (Lim et al., 2017). Excluding batch normalisation improved metrics and eliminated almost all seen artifacts during testing making it the most impactful design choice for this model.

We used a basic CNN for the discriminator which can be seen in Figure 4. Keeping the discriminator simple was important for ensuring it did not gain too much advantage during adversarial training.

2.3 Diffusion

Diffusion models are the current state of the art in image generation tasks, and as such we were interested in the performance of such a model in our application. We make use of a Deep Denoising Diffusion Model (DDPM) (Ho, Jain, and Abbeel, 2020) which is intended for unconditional image generation. During training, the model is given a target image $\boldsymbol{y}_0$, to which it incrementally adds noise in t time-steps according to some schedule, which is referred to as the forward process. The forward process will produce several latent images $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$. While performing the diffusion process, a U-Net (discussed shortly) simultaneously learns how to remove the added noise. This is referred to as the reverse process.

During inference, a diffusion model is given some noisy starting image. By using the learned reverse process, it is able to generate a new image similar to the training examples it has seen previously. To do this, some noisy starting is sampled and used as input for the U-Net reverse

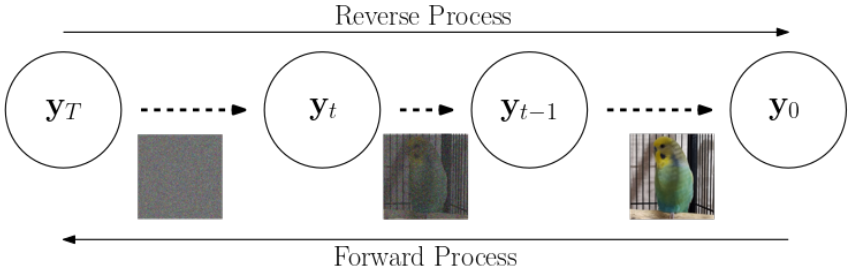


Figure 5: Forward and reverse processes (adapted from Ho, Jain, and Abbeel (2020))

process predictor. The output is cycled back as input for several timesteps until the image is generated.

It should be noted that diffusion models are expensive to train, as for every image, the U-Net must be applied for each noise time step, which results in many more computations than other models. Furthermore, evaluation is also costly for the same reason.

2.3.1 Super Resolution Diffusion Model

To achieve super-resolution using the DDPM, we use a key insight from Saharia et al. (2021), which is to condition the image generation procedure using the low-resolution Sentinel-2 image. This is achieved by first using a traditional method, in our case bicubic interpolation, to upscale the low-resolution image to the target resolution. This is then concatenated to the U-Net's input during training, providing conditioning for the generation procedure. The high-resolution image is provided as the ground truth for training, with the intent for the model to learn how to reproduce high-resolution images given some noisy starting state, and a classically upscaled image.

The U-Net is a type of CNN, and earns the name from its successive downsampling layers, followed by successive upsampling layers which replace the pooling layers commonly found in other CNNs. They are noted for their performance in image segmentation tasks and as such are commonly used in diffusion models (Ronneberger, P.Fischer, and Brox, 2015). Each U-Net block will either increase or decrease the resolution dimension of the input by a factor of two. For example, the first block takes the 256×256 input and reduces it to a 128×128 output. This is facilitated by ResNet sub-blocks within each U-Net block.

Our U-Net architecture can be seen in Figure 6. The input layer is of dimension 256×256×6, corresponding to the two RGB images that are used as inputs. The output layer has the same resolution, but with only 3 channels since it produces a single image. We use six downsampling blocks, with output channels 128, 128, 128, 256, 256 and 256 respectively. We then use 6 upsampling blocks with a mirrored channel layout. All U-Net blocks are comprised of two ResNet sub-blocks. Some literature suggested using three blocks instead, but those sources often considered 4× super-resolution problems rather than our 2× problem (Saharia et al., 2021). A simpler model afforded us a larger batch size in the GPU memory and was better for performance. The 5th downsampling block and the 2nd upsampling block are equipped with spatial attention. The intent was to allow the model to capture features in the image rather than simply observe pixel-wise differences.

The diffusion model was implemented using the HuggingFace `diffusers` library (Platen et al., 2022), which provides convenient tools for creating diffusion models. More details can be found in our code repository.

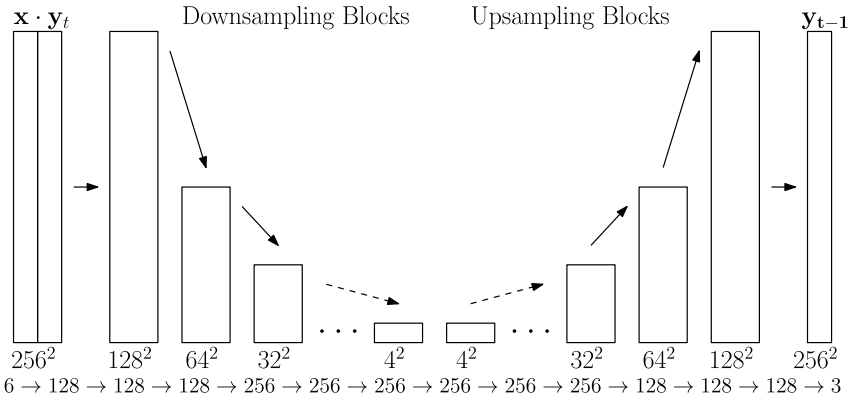


Figure 6: U-Net architecture with block resolution and channels

3 Experiments

We now discuss the specifics of our experimental work, including the data pre-processing steps, the training details, the evaluation metrics used for each model, and our overall findings.

3.1 Data Subset and Pre-Processing

We had access to 130,000 images in the dataset, but due to hardware constraints, we chose a smaller subset of 8,159 training patches and 3,498 test patches. Only RGB channels were used for training due to simplicity and compatibility with existing frameworks. However, wide and narrow infrared, red-edge and atmospheric bands were also available.

We chose these patches from three sites that exhibited sufficient diversity in geographical features, namely FR-BIL, SO2 and NARYN. The proportion of each patch class can be seen in Figure 7.

There are natural inconsistencies that arise between two different satellites. The data had undergone extensive processing to match colour spectrums, align images and more (Michel et al., 2022). However, many image pairs were still not consistent in content. Some inconsistencies include differences in the position of a frame and geographical changes like trees missing in one of the patches. Along with these inter-satellite differences, there were also general irregularities in the data.

The original dataset reported RGB colour bands as surface reflectance in the range $[0, 1]$ (Michel et al., 2022), which is the fraction of light that is reflected from the Earth's surface. However, the data contains values larger than unity, so these are likely surface reflectance *factors*

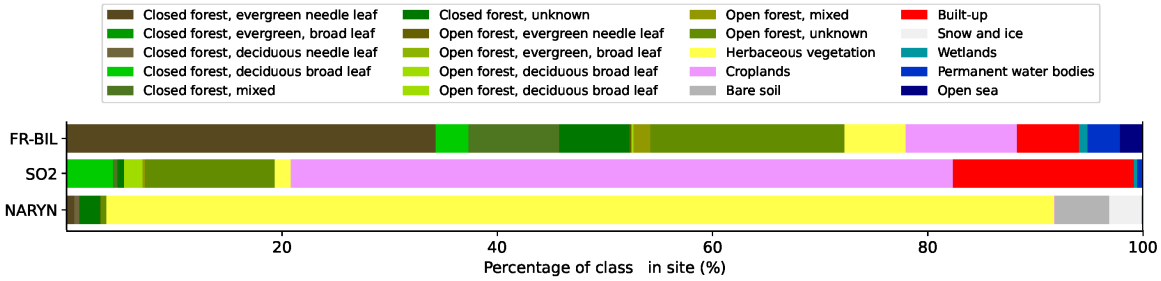


Figure 7: Diversity of each chosen site (Michel et al., 2022).

(a) Reflection from the side of a snowy mountain captured by Sentinel-2.

(b) Reflection from roof windows captured by VENμS.

Figure 8: Examples of high reflectance visual anomalies (Hagolle, 2019).

instead (Schaepman-Strub et al., 2006). The terms reflectance and reflectance factor are often used interchangeably, which can be a source of confusion. Images with reflectance factors larger than one create visual anomalies, and occur due to highly reflective surfaces such as steep snowy mountains and roof windows. Two examples can be seen in Figure 8.

One type of processing used in the original dataset was an atmospheric correction. Atmospheric correction considers the absorbing and scattering effects of the atmosphere to estimate surface reflectance from top-of-atmosphere reflectance measured by remote sensors (Nazeer et al., 2021). This can sometimes be an over-correction which is well known to lead to infeasible negative surface reflectance values (Nazeer et al., 2021). We noted this to be the case with the data. To handle the negative values and outliers of high reflectance, we pre-processed data by clamping values into the range of [0, 1]. We also experimented with standardising reflectance values but this did not yield good performance in our exploratory testing.

3.2 Training Details

3.2.1 Autoencoder

The first two layers of the autoencoder used a learning rate of 10^{-3}, whilst a learning rate of 10^{-4} was used for the last layer. This was noted in the literature to be important for convergence (Dong et al., 2015). The Adam optimizer was used due to its desirable properties. All autoencoder models were trained on an RTX 3060 GPU, which had sufficient memory for a batch size of at most 32. To ensure reasonable training times, we used this maximum batch size. Moreover, the initial weights for the filters are sampled from $\mathcal{N}(0, 10^{-5})$, whilst the bias terms were initialised to zero.

First, we trained an autoencoder using MSE loss for 200 epochs, taking approximately 2 hours in total. To see how perceptual loss influences the metrics and visual quality of the images, we also trained an autoencoder with perceptual loss. The use of perceptual loss was found to significantly slow training compared to MSE loss, due to the need to load the large convolutional blocks onto the GPU, taking approximately 8 hours for 200 epochs.

As seen in Figures 9*a* and 11, the loss rapidly decreases in the early epochs, and then gradually decreases in a stable manner.

3.2.2 SRGAN

The task of the discriminator is much easier in the early stages of training for the SRGAN as super-resolved outputs remain poor until considerable learning has been completed. This can lead to a poor optimization surface for the generator leaving it stuck in local minima (Ham, Jun, and Kim, 2020).

The generator was pre-trained using MSE loss for 75 epochs until the training loss difference started slowing down to avoid these undesirable local minima. During pre-training a batch size of 16 was used. A learning rate of 10^{-4} and weight decay of 10^{-5} were used with the Adam optimizer. All other hyperparameters such as weight and bias initialisation were set to the default PyTorch implementation. Training took 2 hours on an RTX 3060 GPU. On its own, this generator-only model could produce reasonable super-resolved images.

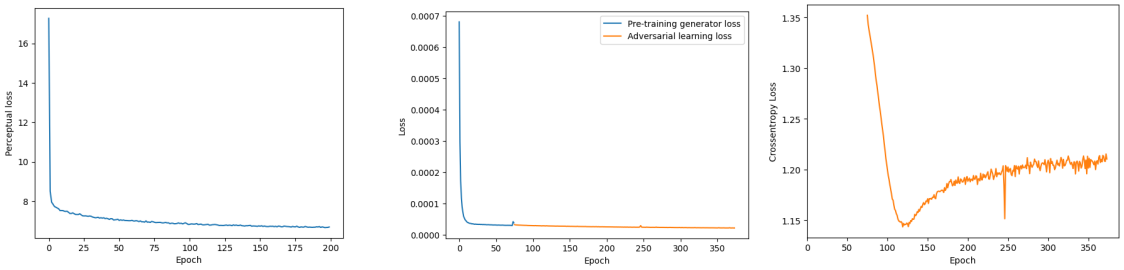
After the generator had been pre-trained we included the discriminator while training for an additional 299 epochs. A batch size of 16 was used again. A learning rate of 10^{-5} and weight decay of 10^{-8} were used for both the generator and the discriminator. We weight the contribution of the discriminator to the generator's loss by $\beta = 10^{-6}$. All other hyperparameters were set to the default PyTorch implementation. Adversarial training took an additional 14 hours.

The learning curves for the SRGAN can be seen in Figures 9b and 9c. The generator first learns at a rapid pace followed by a gradual reduction in loss. It should be noted that visual fidelity still improved greatly even after the initial large reduction in MSE loss. The discriminator also learns at a rapid pace for the first 50 epochs after it is introduced, after which the generator learns to more consistently fool it. We found that making the discriminator more equal in complexity to the generator, for example by increasing the hidden size, often resulted in more unstable learning and frequent failure modes.

3.2.3 Diffusion

The diffusion model was trained using a noise schedule containing 1000 time steps. An Adam optimizer was used with a cosine scheduled learning rate maximised at 10^{-4}, and an MSE loss function. In preliminary testing, it was found that a larger batch size reduced training times and decreased undesirable tinting on generated images. As such, the model was trained using an A100 GPU with a batch size of 16, which was the most that could fit on the 40GB of available GPU memory. The model could only be trained for 13 epochs due to the high compute requirements, amounting to 6 hours of training. Unfortunately, this was insufficient time for training and resulted in an underfitted model.

The loss curve for the diffusion model is omitted in Figure 9 as there was an insufficient number of epochs to observe much movement in the loss.



(a) Autoencoder perceptual loss. (b) SRGAN generator loss. (c) SRGAN discriminator loss.

Figure 9: Learning curves for each model. Learning curves have been separated due to differing scales.

3.3 Metrics

We have selected a suite of model metrics that favour different characteristics of a generated image since there is no universally accepted metric for comparing two images in the field of computer vision (Zhang et al., 2018). Super-resolution is a one-to-many problem in the sense that a low-resolution image could plausibly correspond to multiple high-resolution versions. As such, traditional metrics may penalise generative models for not exactly matching the target image, even when the model has produced a high-quality image (Saharia et al., 2021).

3.3.1 Mean Squared Error (MSE) and Peak Signal to Noise Ratio (PSNR)

The first two metrics in our test suite are MSE (defined in 1) and PSNR. These are both pixel- or element-wise measures of difference, though they differ slightly in what exactly they measure. The PSNR between two images $\boldsymbol{x}_i, \boldsymbol{y}_i$ can be computed with the following formulas:

$$\text{PSNR}(\boldsymbol{x}_i, \boldsymbol{y}_i) = 10 \log_{10} \left(\frac{255^2}{||\boldsymbol{x}_i - \boldsymbol{y}_i||_2^2} \right). \tag{4}$$

The MSE is specifically interested in the difference between two images on a per-element basis, whereas PSNR is useful for comparing images that differ in their dynamic range (Wang and Bovik, 2009). However, we expect the images to have similar dynamic range, so we do not anticipate this PSNR giving us substantially different results to MSE.

It is worth noting that MSE and PSNR are highly localised measures of difference. Consequently, images that minimise the MSE and PSNR should look identical in theory. However, in practice, these metrics do not lend themselves well to image generation tasks (Saharia et al., 2021). As we noted previously, super-resolution is a one-to-many task, and MSE and PSNR may penalise the generative models. We found that two images could have a very small MSE, but the generated image was noticeably blurry compared to the ground truth.

3.3.2 Structural Similarity Index Measure (SSIM)

To alleviate some of the drawbacks of MSE and PSNR, we consider the SSIM, which takes a more macroscopic view of the images. Rather than individual pixels, the measure will examine structural differences across multiple windows across the image. The concept of the structure of an image refers to the idea that nearby pixels are highly related. In this sense, the SSIM is far less localised than MSE or PSNR. However, the SSIM cannot discern any high-level features, and can only make local observations about nearby pixels (Horé and Ziou, 2010). The SSIM formula has been omitted here due to its complexity and can be found in the appendix.

3.3.3 Fréchet Inception Distance (FID)

The FID is the most modern metric we consider. It is designed for evaluating generative models, and to address the shortcomings of the previous metrics, which are not necessarily suited for evaluating generative model performance (Saharia et al., 2021). The FID compares the distribution of a set of generated images to a set of ground truth images. We omit many of the fine details here and only provide a brief overview of how FID is computed. Given two datasets S and S', corresponding to the ground truth and generated images respectively, we consider a distribution function f over these sets of images. A normal distribution is fitted for each of $f_S(s)$ and $f_{S'}(s)$. The FID metric is the Fréchet distance (provided in the appendix) between these two normal distributions (Heusel et al., 2017). All that is left is to select a function f. In the PyTorch implementation of FID, this function f is in fact an InceptionV3 CNN which is pre-trained on ImageNet. This network is designed for image analysis and object detection. A pre-trained network like this will identify higher-level features between the images. As such, we have a metric that is less focused on absolute differences and instead searches for feature similarity, which is more ideal for a one-to-many problem. We found that FID was a better metric for our problem, as models scoring a low FID would produce sharper, high-quality images.

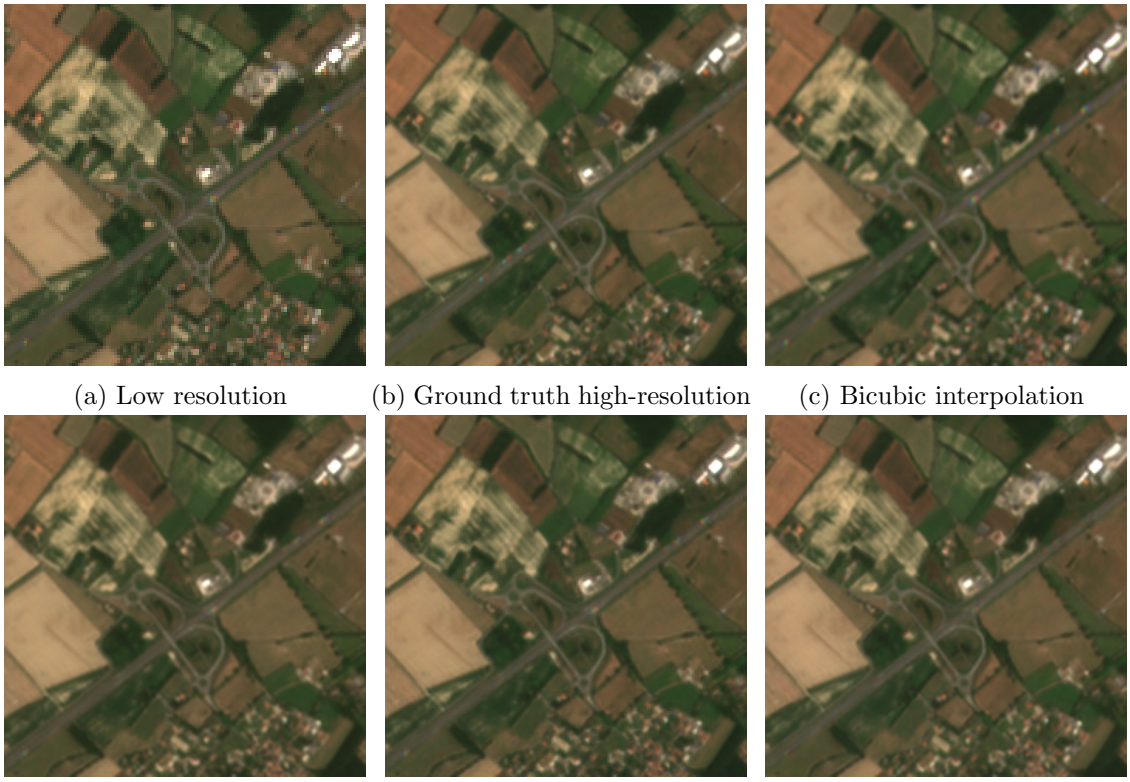
3.4 Results

Table 1: Model performances. ↑ Higher is better. ↓ Lower is better. Best is in bold.

Model	MSE ↓	PSNR ↑	SSIM ↑	FID ↓
Bicubic	4.0248×10^{-5}	43.963	0.98365	1.39640
SRGAN (generator only)	3.1247×10^{-5}	45.068	0.98790	0.97100
SRGAN	$\mathbf{2.5226}\times10^{-5}$	**45.994**	**0.98882**	**0.21432**
Autoencoder (MSE loss)	3.3869×10^{-5}	44.772	0.98684	0.94443
Autoencoder (perceptual loss)	3.7610×10^{-5}	44.319	0.98495	0.39390
Diffusion	2.4000×10^{-2}	16.325	0.41917	8.49360

We compared and evaluated the performance of our models against the baseline bicubic interpolation method on a held-out test set of 3,498 patches from the FR-BIL, SO2, and NARYN sites. Table 1 summarises the quantitative results from these experiments. Specifically, these results demonstrate that our final SRGAN model had the greatest performance across all test metrics and successfully outperformed the baseline bicubic interpolation approach. This included substantially improving upon the FID metric, with a reduction of approximately 84.7% from 1.3964 to 0.21432. Most notably, this suggests that the images produced by the SRGAN model should appear visually sharper and of higher quality than those produced by the bicubic method. Comparing the results of the pre-trained SRGAN generator before adversarial learning to those of the final SRGAN model after adversarial learning, the MSE, PSNR, and SSIM all experienced marginal gains while the FID metric improved significantly from 0.97100 to 0.21432. This may indicate that adversarial learning is an effective paradigm for the super-resolution task.

Although the SRGAN model had the best overall performance on the test set, the autoencoder also outperformed the baseline bicubic interpolation technique on all metrics. Again, the most notable improvement was with respect to the FID metric, which was reduced by approximately 71.8% from 1.3964 to 0.39390. Comparing the results of the two autoencoder models in Table 1, this improvement was largely the result of training with respect to the VGG perceptual loss in place of the standard pixel-wise MSE loss. Although this also led to marginal reductions



(a) Low resolution (b) Ground truth high-resolution (c) Bicubic interpolation

(d) Autoencoder (MSE) (e) Autoencoder (Perceptual) (f) SRGAN

Figure 10: Example high-resolution ground truth and super-resolved images.

in performance for the MSE, PSNR, and SSIM metrics, it is believed that FID is a better indicator of image quality so this trade-off is acceptable. A potential method for overcoming this trade-off is to train the autoencoder to jointly minimise the MSE and VGG loss.

Finally, the diffusion model had the worst performance of the three architectures we explored and was unable to surpass the baseline metrics in testing. This was largely the result of underfitting as the model was too expensive to train sufficiently. Preliminary testing on a smaller data subset showed promising results, which are included in Figure 13 in the appendix.

Although quantitative evaluation is a useful technique for assessing the quality of the images produced by our models, it was also necessary to compare their results qualitatively. Figure 10 displays an example of a ground truth high-resolution image along with the up-scaled images produced by the bicubic interpolation technique and the final autoencoder and SRGAN models. As we expected from the FID metrics in Table 1, the images produced by the two models produce visually sharper and more accurate reconstructions of the ground truth compared to the bicubic method.

4 Conclusion

In this project, we successfully employed deep-learning techniques to perform super-resolution on satellite imagery. For this purpose, we explored several model architectures including autoencoders, generative adversarial networks, and diffusion models. Through evaluation of these models on a range of test metrics designed to assess both pixel- and feature-wise image similarity, we found that the SRGAN and autoencoder models each outperformed the baseline bicubic interpolation technique on the super-resolution task. Although the SRGAN had the greatest

overall performance on the test metrics, qualitatively, the SRGAN and autoencoder models produced images of similarly high visual quality that surpassed that of the bicubic method. These results demonstrate the importance of the FID metric in evaluating images produced via super-resolution.

4.1 Limitations

Although we were able to successfully perform image upscaling using deep-learning techniques, this approach wasn't without limitations. One of the main limitations we encountered was with our dataset. For example, many super-resolution approaches, including the original SRGAN paper, generate their training datasets by downsampling the ground-truth high-resolution images (Ledig et al., 2017). In contrast, our training and test datasets were produced by two different satellites which introduced noise and discrepancies that made learning good models more difficult. Additionally, the fact that the low-resolution images are only upscaled by a factor of 2 further limited the performance of our models. Galar et al. (2019) suggests that most super-resolution networks focus on at least 4x upscaling where the difference between deep-learning techniques and the bicubic interpolation approach is more significant.

Time and hardware constraints were also major limiting factors in this project. For example, our final models were only trained on approximately 8% of the full dataset. Despite this, each model still took between 6 and 14 hours to train completely. These constraints made iterating and trying new approaches significantly more difficult. Most notably, we believe that this is the reason for the poor performance of the diffusion model. Both the SRGAN and autoencoder architectures had the greatest success when trained for a large number of epochs (200 – 400) and with small learning rates (10^{-4} – 10^{-5}). This was not possible for the diffusion model which took 6 hours to complete only 13 epochs with a learning rate of 10^{-4}.

There are also practical limitations to our approach. Most notably, performing image upscaling via deep-learning techniques is more computationally expensive than traditional methods such as bicubic interpolation. This is most evident in our autoencoder architecture, which uses both bicubic interpolation and neural networks to produce the final upscaled image. This introduces a trade-off between accuracy and speed. In some applications, such as real-time image upscaling where efficiency is most important, the use of traditional, non-deep-learning approaches may be preferable.

4.2 Future Work

Throughout this project, we have also identified some potential directions for future work. Firstly, because of the time and hardware constraints we encountered and the success of the SRGAN architecture, we briefly explored transfer learning of a large pre-trained ESRGAN (Enhanced SRGAN) model to the task of satellite image upscaling. Details of this approach are outlined in the appendix and present a possible line of future work. Additionally, although the VGG loss significantly improved the FID metric for the autoencoder architecture, we were unable to successfully implement this approach for the SRGAN and diffusion models due to time constraints. Exploring the use of the VGG loss with these models may result in further improvements to image clarity, although more research is required. Finally, extending our work to the full dataset with the implementation of data augmentation and regularisation techniques is a natural progression of our findings. These extensions would likely be beneficial for model generalisation in practice.

Appendix

Autoencoder Training Loss Curve

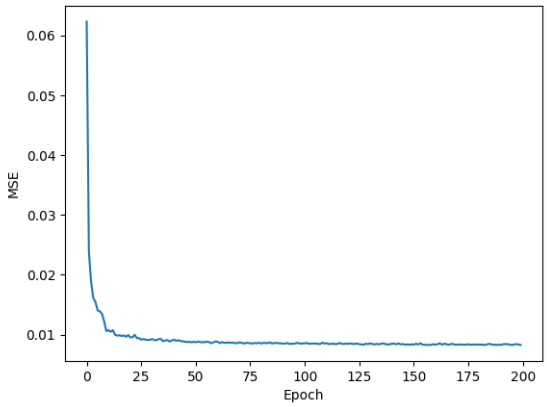


Figure 11: Learning curve of the autoencoder trained with MSE loss.

Structural Similarity Index Measure

For two images $\boldsymbol{x}_i, \boldsymbol{y}_i$ of the same dimensions, the SSIM is defined as follows (Horé and Ziou, 2010):

$$\text{SSIM}(\boldsymbol{x}_i, \boldsymbol{y}_i) = l(\boldsymbol{x}_i, \boldsymbol{y}_i) c(\boldsymbol{x}_i, \boldsymbol{y}_i) s(\boldsymbol{x}_i, \boldsymbol{y}_i) \tag{5}$$

where

$$l(\boldsymbol{x}_i, \boldsymbol{y}_i) = \frac{2\mu_{\boldsymbol{x}_i}\mu_{\boldsymbol{y}_i} + C_1}{\mu_{\boldsymbol{x}_i}^2 + \mu_{\boldsymbol{y}_i}^2 + C_1}, \tag{6}$$

$$c(\boldsymbol{x}_i, \boldsymbol{y}_i) = \frac{2\sigma_{\boldsymbol{x}_i}\sigma_{\boldsymbol{y}_i} + C_2}{\sigma_{\boldsymbol{x}_i}^2 + \sigma_{\boldsymbol{y}_i}^2 + C_2}, \tag{7}$$

$$s(\boldsymbol{x}_i, \boldsymbol{y}_i) = \frac{\sigma_{\boldsymbol{x}_i\boldsymbol{y}_i} + C_3}{\sigma_{\boldsymbol{x}_i}\sigma_{\boldsymbol{y}_i} + C_3}, \tag{8}$$

where μ and σ are the mean and standard deviation of the images, and C_1, C_2, C_3 are constants to avoid a null denominator.

Fréchet Distance

The Fréchet Distance between two normal distributions can be calculated as follows (Dowson and Landau, 1982):

$$d^2(X, Y) = ||\mu_X - \mu_Y||_2^2 + \text{tr}\left(\Sigma_X + \Sigma_Y - 2(\Sigma_X\Sigma_Y)^{\frac{1}{2}}\right), \tag{9}$$

where μ denotes the mean of the respective distributions, and Σ denotes the covariance matrices of the respective distributions.

Additional Diffusion Results

We omitted images from the diffusion model due to their poor quality, but include an example in Figure 12 for completeness.

Figure 12: Diffusion Model Example Image

The final diffusion model did not perform well due to underfitting issues, which is an unfortunate consequence of the more expensive training process. However, there is still potential in the model, and in Figure 2 we present some results from preliminary tests performed on a smaller data subset. Rather than the 3 sites used for testing our final models, these results apply to a diffusion model trained and tested on only the SO2 site.

Table 2: Diffusion metrics on SO2 site.

MSE ↓	PSNR ↑	SSIM ↑	FID ↓
2.5046×10^{-4}	36.107	0.92222	5.48320

While these metrics are not as impressive as those of the other final models, they indicate that with proper training, the full diffusion model may have been able to rival the other models' performance.

Figure 13 shows an example image from this smaller model. There is some blurriness and colour inaccuracy, but the image still shows some there is some promise for diffusion as an upscaling method.

Figure 13: Diffusion Model Example Image trained on only SO2

ESRGAN Model

The ESRGAN model is an extension of the original SRGAN with two predominant changes (Wang et al., 2018):

1. The model was trained using the perceptual VGG loss.
2. Residual blocks were replaced by a proposed Residual in Residual Dense Block (RRDB).

The architecture of this block is shown in Figure 14.

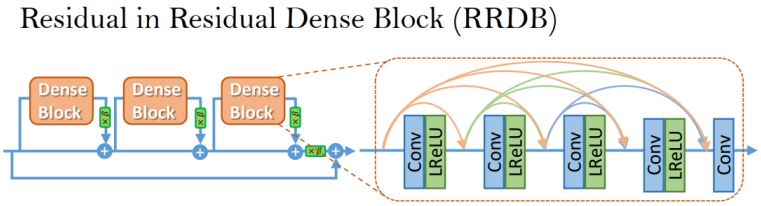


Figure 14: Architecture of the RRDB with residual scaling parameter β.

Wang et al. (2018) found that these changes significantly and consistently improved the visual quality of upscaled images when compared to the original SRGAN architecture. For this reason, it was believed that similar results may be experienced on the SEN2VENµS dataset.

We followed the procedure outlined by Wang et al. (2021b) to fine-tune a large pre-trained ESRGAN model to our satellite imagery dataset. In particular, the model that we used consisted of 23 RRDB blocks that were pre-trained on the DIV2K and Flickr2K datasets. Because of our hardware constraints, the goal of this approach was to learn a richer architecture with less training by transferring the pre-trained features of the ESRGAN model to our dataset. However, this architecture was too large and only 10K of the suggested 400K training iterations could be run, which still took approximately 22 hours. As a result, the ESRGAN model performed poorly on the test metrics which are summarised in Table 3.

Table 3: ESRGAN test metrics.

MSE ↓	PSNR ↑	SSIM ↑	FID ↓
1.1277×10^{-2}	30.987	0.90874	3.10938

References

Dong, C. et al. (2015). *Image Super-Resolution Using Deep Convolutional Networks*. arXiv: 1501.00092 [cs.CV].

Dowson, D. and B. Landau (1982). "The Fréchet distance between multivariate normal distributions". In: *Journal of Multivariate Analysis* 12.3, pp. 450–455. ISSN: 0047-259X. DOI: https://doi.org/10.1016/0047-259X(82)90077-X. URL: https://www.sciencedirect.com/science/article/pii/0047259X8290077X.

Froede, S. (2023). *Discover the differences between classical and AI upscaling methods, their benefits, and applications in enhancing image quality.* URL: https://unimatrixz.com/topics/ai-upscaler/upscaling-methods/.

Galar, M. et al. (2019). "Super-resolution for Sentinel-2 images". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42, pp. 95–102.

Hagolle, O. (2019). *[How it works] Can reflectances be greater than 1?* URL: https://labo.obs-mip.fr/multitemp/16885-2/.

Ham, H., T. J. Jun, and D. Kim (2020). *Unbalanced GANs: Pre-training the Generator of Generative Adversarial Network using Variational Autoencoder*. arXiv: 2002.02112 [cs.LG].

He, K. et al. (2015a). *Deep Residual Learning for Image Recognition*. arXiv: 1512.03385 [cs.CV].

— (2015b). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. arXiv: 1502.01852 [cs.CV].

Heusel, M. et al. (2017). "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

Ho, J., A. Jain, and P. Abbeel (2020). *Denoising Diffusion Probabilistic Models*. arXiv: 2006.11239 [cs.LG].

Horé, A. and D. Ziou (2010). "Image Quality Metrics: PSNR vs. SSIM". In: *2010 20th International Conference on Pattern Recognition*, pp. 2366–2369. DOI: 10.1109/ICPR.2010.579.

Johnson, J., A. Alahi, and L. Fei-Fei (2016). *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. arXiv: 1603.08155 [cs.CV].

Lanaras, C. et al. (2017). "Super-Resolution of Multispectral Multiresolution Images from a Single Sensor". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1505–1513. DOI: 10.1109/CVPRW.2017.194.

Ledig, C. et al. (2017). *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*. arXiv: 1609.04802 [cs.CV].

Lim, B. et al. (2017). *Enhanced Deep Residual Networks for Single Image Super-Resolution*. arXiv: 1707.02921 [cs.CV].

Lin, C.-H. and J. M. Bioucas-Dias (2020). "An Explicit and Scene-Adapted Definition of Convex Self-Similarity Prior With Application to Unsupervised Sentinel-2 Super-Resolution". In: *IEEE Transactions on Geoscience and Remote Sensing* 58.5, pp. 3352–3365. DOI: 10.1109/TGRS.2019.2953808.

Michel, J. et al. (2022). "SEN2VENμS, a Dataset for the Training of Sentinel-2 Super-Resolution Algorithms". In: *Data* 7.7. ISSN: 2306-5729. DOI: 10.3390/data7070096. URL: https://www.mdpi.com/2306-5729/7/7/96.

Nazeer, M. et al. (2021). "Evaluation of atmospheric correction methods for low to high resolutions satellite remote sensing data". In: *Atmospheric Research* 249, p. 105308. ISSN: 0169-8095. DOI: https://doi.org/10.1016/j.atmosres.2020.105308. URL: https://www.sciencedirect.com/science/article/pii/S016980952031245X.

Njambi, R. (2023). *What does "high resolution satellite imagery" mean, anyway?* URL: https://up42.com/blog/what-high-resolution-satellite-imagery-means.

Paris, C., J. Bioucas-Dias, and L. Bruzzone (2017). "A hierarchical approach to superresolution of multispectral images with different spatial resolutions". In: *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 2589–2592. DOI: 10.1109/IGARSS.2017.8127525.

Platen, P. von et al. (2022). *Diffusers: State-of-the-art diffusion models*. https://github.com/huggingface/diffusers.

Ronneberger, O., P.Fischer, and T. Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

Saharia, C. et al. (2021). *Image Super-Resolution via Iterative Refinement*. arXiv: 2104.07636 [eess.IV].

Schaepman-Strub, G. et al. (2006). "Reflectance quantities in optical remote sensing—Definitions and case studies". In: *Remote sensing of environment* 103.1, pp. 27–42.

Shi, W. et al. (2016). *Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network*. arXiv: 1609.05158 [cs.CV].

Theckedath, D. and R. R. Sedamkar (2020). "Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks". In: *SN Computer Science* 1.2, p. 79. DOI: 10.1007/s42979-020-0114-9. URL: https://doi.org/10.1007/s42979-020-0114-9.

Wang, L. et al. (2021a). *Exploring Sparsity in Image Super-Resolution for Efficient Inference*. arXiv: 2006.09603 [cs.CV].

Wang, X., C. Dong, and Y. Shan (2022). *RepSR: Training Efficient VGG-style Super-Resolution Networks with Structural Re-Parameterization and Batch Normalization*. arXiv: 2205.05671 [cs.CV].

Wang, X. et al. (2018). *ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks*. arXiv: 1809.00219 [cs.CV].

Wang, X. et al. (2021b). *Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data*. arXiv: 2107.10833 [eess.IV].

Wang, Z. and A. C. Bovik (2009). "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures". In: *IEEE Signal Processing Magazine* 26.1, pp. 98–117. DOI: 10.1109/MSP.2008.930649.

Zhang, R. et al. (2018). *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. arXiv: 1801.03924 [cs.CV].