UNIVERSITAT POLITÈCNICA DE CATALUNYA - FACULTAT D'INFORMÀTICA DE BARCELONA

UNIVERSITAT DE BARCELONA - FACULTAT DE MATEMÀTIQUES

UNIVERSITAT ROVIRA I VIRGILI - ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA

MASTER THESIS

Multi-Scale Super Resolution With Blind De-noising Using Residual Learning For Digital Art

---

Author:
Santiago MAZAGATOS PÉREZ

Advisor:
Sergio ESCALERA

April 24, 2019

Abstract

Master in Artificial Intelligence

Multi-Scale Super Resolution With Blind De-noising Using Residual Learning For Digital Art

by Santiago MAZAGATOS PÉREZ

In the fields of illustration and digital art, it is imperative that an image be of high quality when being used for printing, as a wallpaper or for another kind of decorative purpose, however, high resolution, clean images are often unavailable, either because the form in which the original work was rendered is deemed subpar by today's standards, or because all available copies have been subject to lossy compression or downsampling. Super Resolution is an increasingly active field of research in machine learning with the aim of providing a computational model that can achieve what is impractical or impossible with human means, the recreation of available images or other kind of visual data in the highest possible quality.

Super-Resolution, Residual Learning, Denoising

Contents

List of Figures

Chapter 1

Introduction

Technical limitations are as common to technology as technology itself, what was a vast amount of storage a decade ago is little more than what one would expect from a basic cloud storage plan today, that being the case one often comes face to face with yesterday's limitations, chief among them processing power and storage.

In an era in which the most common removable storage medium had no more than 1.44 Megabytes of space and the average processor had its performance measured in Megahertz and whether or not it had a math co-processor, image use was accompanied by lossy compression, like in the case of JPEG (Hamilton, 1992) files, to save the most amount of space, and cheap downsampling methods that wouldn't overburden the CPU.

Nowadays storing images in PNG (Boutell, 1997) format with its lossless compression and downsizing using methods such as Lanczos resampling (Wikipedia contributors, 2019) to prevent aliasing is a reasonable proposition, but the previous methods of saving space and cycles are still present, and there's no way of knowing whether an illustration created by a digital artist today with a high definition display will end up being dwarfed by screens 10 years into the future.

If one wanted to print an illustration onto a T-shirt, or a poster, or use it as a wallpaper on a 4K display, such file would need to have a very high resolution and no compression artifacts or noise to look good, however such files can often only be sourced from the artist, if they even exist; recreating the work in a way that achieves the required quality would require a substantial amount of time and effort by someone with the skill set to do so, while it would be expensive for someone who lacks it to commission someone for it, furthermore the result might still be unsatisfactory, with the recreated image having a slightly different aesthetic feel to it.

1.1 Waifu2x

Waifu2x (nagadomi, 2019) is a web based application that performs upscaling and denoising of anime style images and photos, however, it uses different models for different levels of noise, relying on the user to select the correct denoising level, and it only allows upscaling up to 2 times the original image size.

While there is a Caffe port of waifu2x (lltcggie, 2019) that supports upscaling beyond 2 times and has an auto-denoising feature, it is not explained how the software determines the best denoising level for a particular image, this port runs on the user's computer and is able to take advantage of any CUDA enabled graphics cards present, making it well suited for batch processing images since the original

web application has no such feature and requires a captcha to be resolved for every processed image.

1.2 Goals

The goals of this project are simple, to create a model that is able of upscaling and denoising an image (digital art) without any prior knowledge of level or type of noise (if any) present on the input image/s with results on par or superior to those of the best available alternative (Waifu2x).

Chapter 2

State of the art

2.1 Super-Resolution

Initial works in the world of Super-Resolution relied primarily on Dictionary based approaches, with low resolution and high resolution dictionaries in which a high resolution image is obtained by matching the low resolution representation to items in the LR dictionary and translating those similarities using the HR dictionary to the final HR image.

Dictionary based methods have undergone a great deal of transformation, with a wide array of improvements designed to improve the dictionary learning process (He, Qi, and Zaretzki, 2013), or the LR to HR mapping in terms of speed and memory with approaches like Anchor Neighborhood Regression (Timofte, De, and Gool, 2013), with its subsequent improvements(Timofte, De Smet, and Van Gool, 2015, Timofte, Rothe, and Gool, 2015, Perez-Pellitero, Salvador, Ruiz-Hidalgo, and Rosenhahn, 2016), Sparse Coding (Yang, Wright, Huang, and Ma, 2010, Wang, Liu, Yang, Han, and Huang, 2015) and Super-Resolution Forests (Salvador and Pérez-Pellitero, 2015, Schulter, Leistner, and Bischof, 2015).

Along with all the methods reliant on external information came the ones reliant in information intrinsic to the images being processed (Huang, Singh, and Ahuja, 2015) and more hybrid approaches with external pre-training and internal fine tuning (Wang, Yang, Wang, Chang, Han, Yang, and Huang, 2015.

However, present works rely on less engineered solutions with maps of LR and HR dictionaries and instead have networks learn the mapping functions directly in a non-linear way, the specifics still vary depending on which mapping function has to be learned, the primary example of this new avenue of research is SRCNN (Dong, Loy, He, and Tang, 2016), which has been the subject of some improvements (Shi, Caballero, Huszár, Totz, Aitken, Bishop, Rueckert, and Wang, 2016, Dong, Loy, and Tang, 2016).

As models were made deeper in order to learn more complex mappings, skip connections like the ones displayed in ResNet (He, Zhang, Ren, and Sun, 2015) were added to avoid exploding/vanishing gradients (Tai, Yang, and Liu, 2017, Kim, Lee, and Lee, 2015a, Ledig, Theis, Huszar, Caballero, Aitken, Tejani, Totz, Wang, and Shi, 2016, Lim, Son, Kim, Nah, and Lee, 2017), recursive architectures have also been successfully implemented, among other things as a way to mitigate overfitting (Kim, Lee, and Lee, 2015b), which have then been improved upon (Tai, Yang, and Liu, 2017)

Additionally, new models have been using Generative Adversarial Networks as a way of guiding networks towards producing results belonging to the natural image manifold, avoiding results that are good number-wise (as in having a low Mean Squared Error, or high PSNR and SSIM), but look "soft" and artificial to humans. (Ledig, Theis, Huszar, Caballero, Aitken, Tejani, Totz, Wang, and Shi, 2016, Wang, Yu, Dong, and Loy, 2018, Yuan, Liu, Zhang, Zhang, Dong, and Lin, 2018).

2.1.1 Denoising

Modern denoising methods often double as SISR models, such as TNRD (Chen and Pock, 2017) and RED30 (Mao, Shen, and Yang, 2016), and despite its age, BM3D (Dabov, Foi, Katkovnik, and Egiazarian, 2007) is still relevant and being revisited, either to propose an alternative (Burger, Schuler, and Harmeling, 2012), or a modern adaptation (Yang and Sun, 2018).

2.2 Deep Learning libraries

There are a variety of Open Source machine learning libraries for Python with CUDA support:

- **TensorFlow**: Developed by the Google Brain Team, TensorFlow is a very popular library, especially when used as a backend with Keras, it is in fact taught in this Master's Deep Learning course. (tensorflow, 2019)
- **PyTorch**: Based on Torch and merged with Caffe2, it is developed by Facebook's AI research group. (pytorch, 2019)
- **Theano**: While no longer in development and a relatively minor player, the fact that it was developed by the Montreal Institute for Learning Algorithms might attract those who might be displeased with the corporate ties of other libraries. (Theano, 2019)
- **Microsoft Cognitive Toolkit**: Microsoft's CNTK library has both a Python API and a C# API, which makes it much easier to integrate with Universal Windows Platform, Windows Forms and ASP.NET applications. (Microsoft, 2019)

Chapter 3

Methodology

3.1 Choice: EDSR vs. CinCGAN

At the start of this master thesis project my advisor gave me two papers as good state of the art methods that would do nicely EDSR (Lim, Son, Kim, Nah, and Lee, 2017) and CinCGAN (Yuan, Liu, Zhang, Zhang, Dong, and Lin, 2018) offer fundamentally different approaches to Super Resolution:

EDSR builds on SRResNet (Ledig, Theis, Huszar, Caballero, Aitken, Tejani, Totz, Wang, and Shi, 2016), which is an adaptation of ResNet (He, Zhang, Ren, and Sun, 2015) for SISR, removing Batch Normalization layers, thus reducing the computational and memory burden of the network on hardware and allowing for a deeper architecture and more complex non-linear model to be learned for LR/HR mapping.

CinCGAN uses CycleGAN's (Zhu, Park, Isola, and Efros, 2017) idea of image to image translation to create an unsupervised model that can denoise and upsample images without training with LR/HR pairs by first denoising the input in LR space, and then upscaling it with a state of the art SISR model, in this case EDSR.

While CinCGAN achieves performance comparable to SRGAN, I personally find CinCGAN's results visually un-appealing since it seems to introduce artifacts of its own.

(a) ground truth	(b) bicubic	(c) EDSR$^+$	(d) SRGAN$^+$ [16]	(e) BM3D+EDSR	(f) **CinCGAN (ours)**
PSNR/SSIM	26.81/0.83	30.28/0.88	29.05/0.85	26.84/0.86	28.26/0.84

FIGURE 3.1: While SRGAN+'s result is not as sharp as the ground truth or CinCGAN's result, the lines in the latter case look distorted (Taken from the CinCGAN paper)

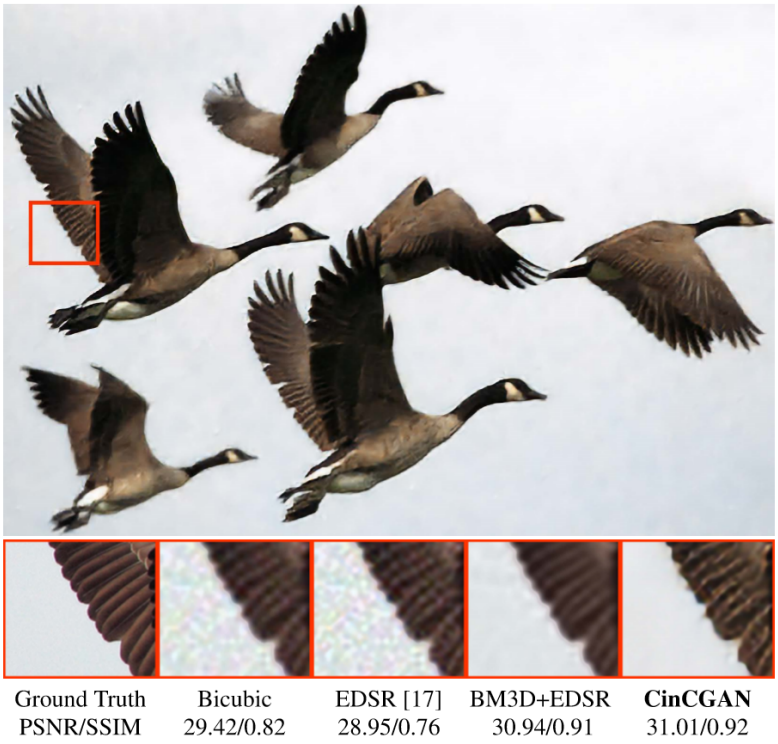


FIGURE 3.2: While Bicubic and EDSR models can't denoise the image and BM3D+EDSR's result looks soft and blurry, the CinCGAN seems to introduce its own noise (Taken from the CinCGAN paper)

Since EDSR already performed the SR task inside CinCGAN and I didn't like the visual quality of the results given by the latter I chose to work with EDSR as a SR method, and would later think of a way of performing denoising.

3.2 Choice: Denoising

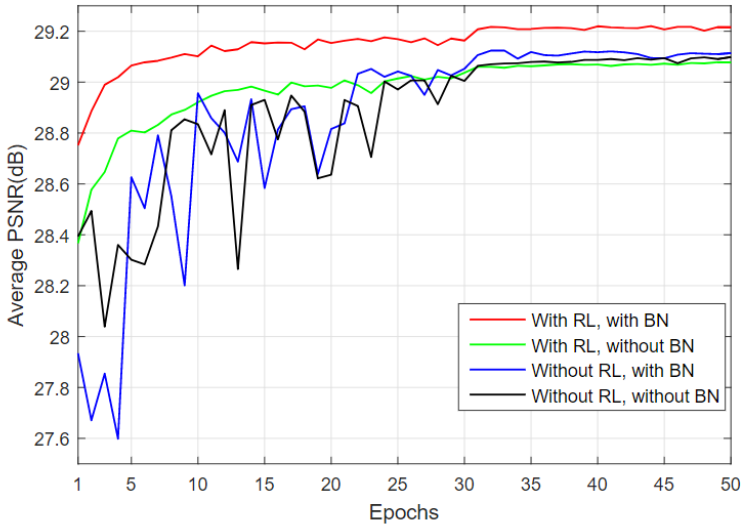
Denoising is a much more established and "stable" field, so when faced with the need to choose a denoising method I chose to look for extensively cited papers first instead of prioritizing state-of-the-art brand new ones.

DnCNN (Zhang, Zuo, Chen, Meng, and Zhang, 2017) is a residual neural network capable of handling gaussian noise and JPEG compression, its architecture according to the authors is a modified VGG (Simonyan and Zisserman, 2015) network, adapting it for denoising instead of image recognition and implementing residual learning (He, Zhang, Ren, and Sun, 2015), the result is strikingly similar to SRResNet, and therefore, to EDSR, although SR tasks in DnCNN are performed by upscaling a LR image to HR size using bicubic interpolation and feeding it to the network.

The way training data is created for DnCNN is to create patches with noise in the ranges of [0,55]σ for gaussian noise and [5,99] quality for JPEG deblocking.

I hypothesized that I could take the training method in DnCNN and use it with the EDSR architecture, since the only major differences were the lack of Batch Normalization layers, and the deconvolution and transpose process at the very end of EDSR, making its output larger than its input.

DnCNN had been tested without Batch Normalization and it was shown that with Adam (Kingma and Ba, 2015), the training and performance impact of not using BN layers was small, so that wouldn't be a significant issue.



(b) Adam

FIGURE 3.3: Impact of training DnCNN with and without BN (Taken from the DnCNN paper)

3.3 Final decision

I chose to use the EDSR architecture with a mixed training approach, the images in the training set would be split into overlapping patches, augmented with 90 degree rotations and then treated randomly in one of three ways (in all cases the HR patch is saved as a PNG intact):

- **Clean save**: the extracted patch would be downscaled using Lanczos resampling and saved as a PNG to preserve its quality.
- **JPEG save**: the extracted patch would be downscaled using Lanczos resampling and saved as a JPEG with a quality varying from 5 to 99.
- **Gauss save**: the extracted patch would be downscaled using Lanczos resampling, gaussian noise would be added to the patch with a standard deviation value between 0 and 55 and then saved as a PNG file.

The network will follow the EDSR architecture and be trained with the Adam optimizer (learning rate: 1e-4, with it being halved every 200.000 updates, momentum: 0.9, variance momentum: 0.999, ϵ: 1e-8) for 300.000 updates with a minibatch size of 16, for the loss function, L2 is used (as in SRResNet) instead of L1 (EDSR).

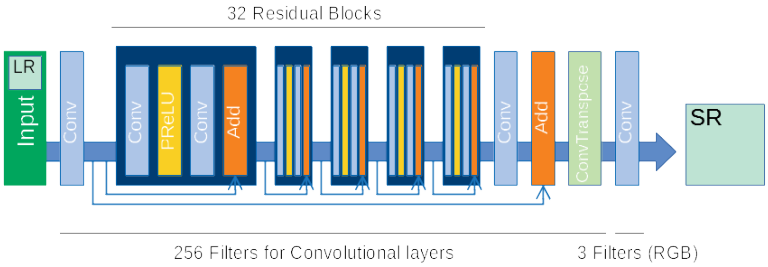


FIGURE 3.4: Implemented Architecture for 2x and 3x scales

This for 4X upscaling has one more ConvolutionTranpose layer, this is especially useful for reusing the 2X model's weights to train a 4X model since the parameters of the layer remain the same.

Chapter 4

Development

4.1 Dataset

To train the model to work with digital art and illustrations a dataset other than the usual photograph based datasets like the ones used on the EDSR paper (DIV2K, Set5, Set14, B100, Urban100), to do that I considered a variety of digital art imageboards: Danbooru, Gelbooru, Yandere, e621, Sankaku Channel and Derpibooru.

I needed images that were in PNG format, for practicality reasons the selected board had to work well with an automatic downloader (Bionus, 2019), the board from which the images were going to be downloaded also needed to have reliable tagging and preferably, an active community that would use the often underutilized scoring system present in (nearly) all imageboards.

Danbooru and e621 were the best candidates among all imageboards considered but due to the prevalence of japanese artists in Danbooru and their infamous censorship practices I chose e621 over Danbooru to ensure no mosaic filters ended up in the training patches.

I downloaded the 1,600 highest scored images on e621 with the assumption that due to their popularity they would be more representative of what the final use case would be.

4.2 Network implementation

Microsoft's CNTK has a greater amount of instructional materials, including examples and tutorials than TensorFlow (in my experience), and has a guide on how to implement various SISR models such as VDSR, DRRN, SRResNet and SRGAN and how to use them later to upscale images (Vukorepa, 2017)

The reason why 1,600 images were downloaded (an excessive amount) is because I initially considered adding a discriminator to EDSR, effectively applying EDSR's improvements over SRResNet to SRGAN and I would use 800 images to train the generator, and 800 to train the GAN, however SRGAN relies on a pre-trained VGG network which is trained on different data (photographs) and training the discriminator from scratch, fine tuning the balance in the loss function between discriminator and generator was a dangerous timesink.

Due to memory constraints on my GTX 1060 (6 GB), the LR patch size for the network had to be reduced to 32 x 32, down from EDSR's LR patch size of 48 x 48.

FIGURE 4.1: Training curve for 2X scale, Training time was: 1 day, 22 hours, 53 minutes

Geometric Self Ensemble as used in Lim, Son, Kim, Nah, and Lee, 2017 and explained in Timofte, Rothe, and Gool, 2015 was implemented, however I haven't used it for testing purposes, my concerns being that each patch needs to be predicted 8 times and the same concerned expressed by Ledig, Theis, Huszar, Caballero, Aitken, Tejani, Totz, Wang, and Shi, 2016 in section 1.1.3, being that averaging all possible solutions for a specific patch might cause a loss of detail and overly smooth the image as a whole, since EDSR doesn't have a discriminator to ensure a good perceptual loss value, this issue is especially relevant.

4.3 Graphical User Interface

Using waifu2x-caffe as an inspiration I created a graphical user interface designed not only to use my EDSR models, but any CNTK model that takes low resolution patches and outputs high resolution ones, the code only needs to know the output patch size and the upscaling coefficient and it will deconstruct input images into appropriately sized patches and reconstruct a high resolution image from the results, additionally, unlike waifu2x-caffe, this GUI presents a real time preview of the upscaling process, a lack of feedback can often be upsetting to users especially if the model is being run on slow hardware.

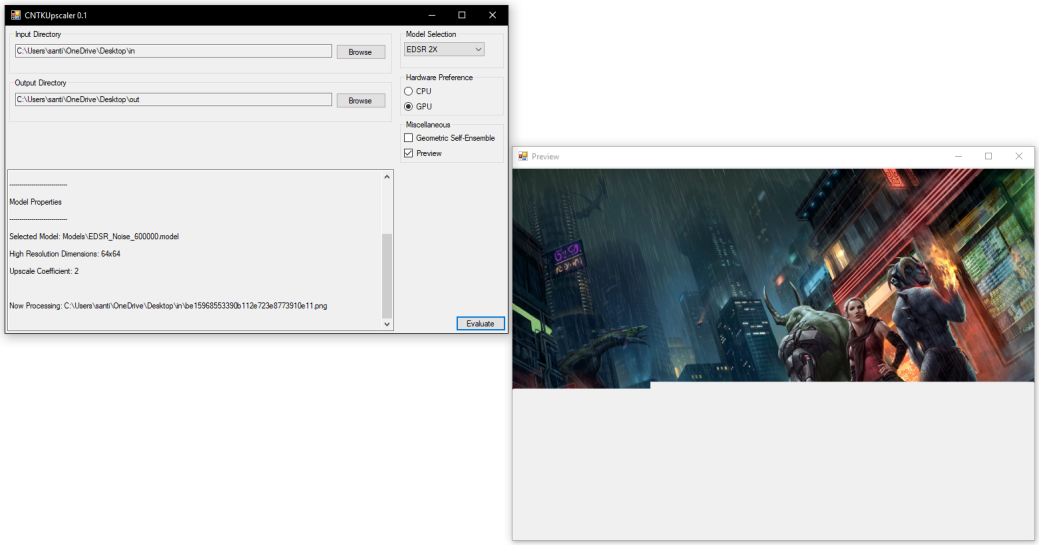


FIGURE 4.2: Screenshot of the CNTK Upscaler GUI

The upscaler also supports using Geometric Self Ensemble since it isn't dependent on the model, and models can be added by placing them on the 'Models' folder of the application along with a JSON file with the necessary information, the application automatically loads all models present in the folder at startup.

```
{
  "ModelName": "EDSR 2X",
  "File": "EDSR_Noise_600000.model",
  "OutDimensions": 64,
  "UpscaleCoefficient": 2
}
```

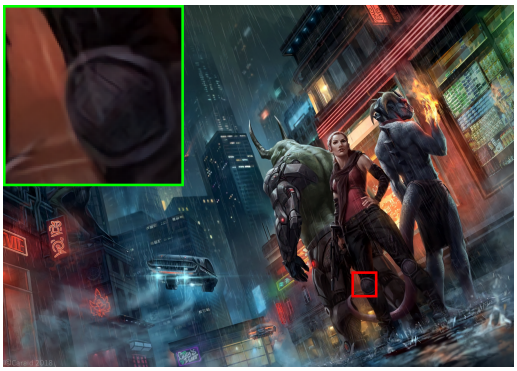
FIGURE 4.3: An example JSON file

Chapter 5

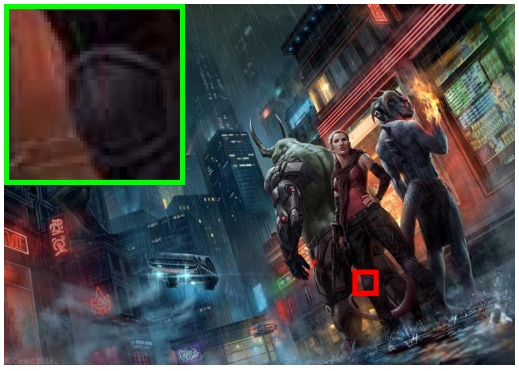
Evaluation

To evaluate the trained EDSR with blind-denoising I hand picked 7 pictures with varying styles and complexity and created low resolution versions with specific gaussian noise levels and JPEG qualities; for Gaussian noise the σ values were: 0, 15, 25 and 50. And for JPEG quality levels, the values were: 25, 50, 75 and 100, in total that is 56 low resolution images per model to test.

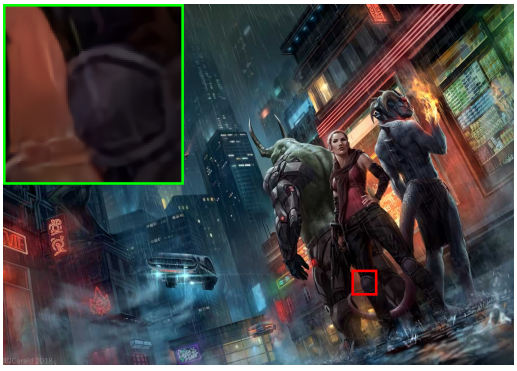
(A) LR image with gaussian noise (25σ)



(B) EDSR result

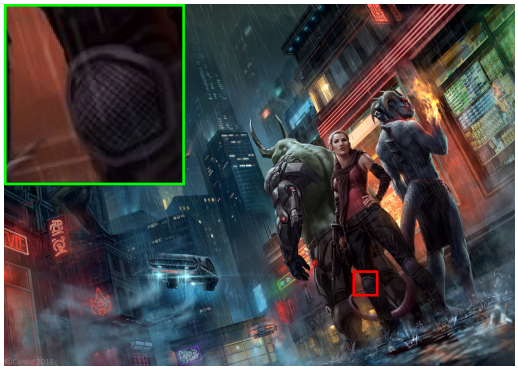


(C) LR image with JPEG noise (50)

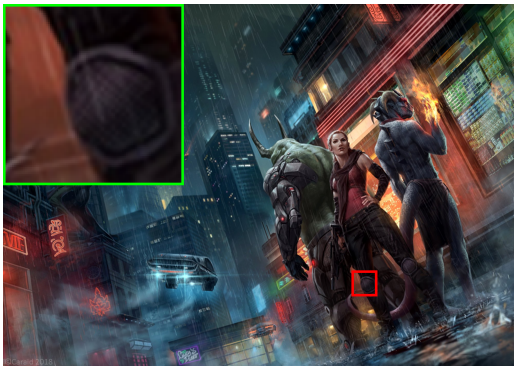


(D) EDSR result

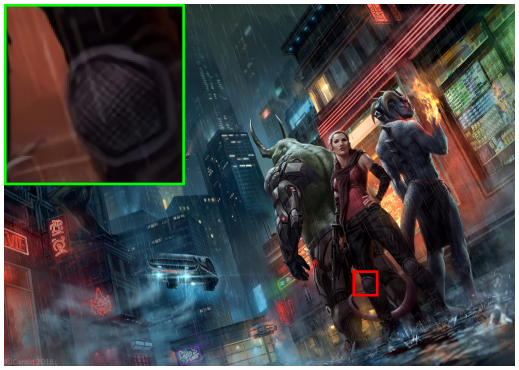
FIGURE 5.1: trained denoising capabilities



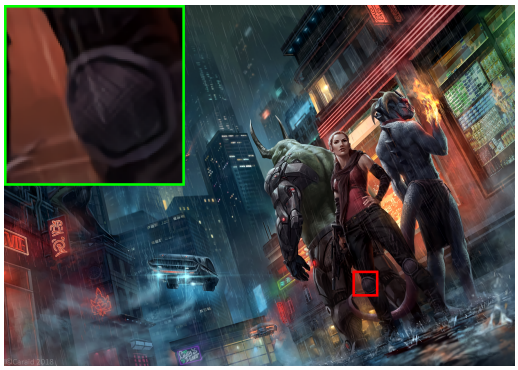
(A) High Resolution



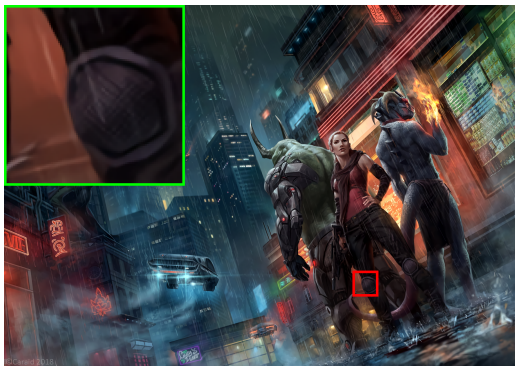
(B) Bicubic



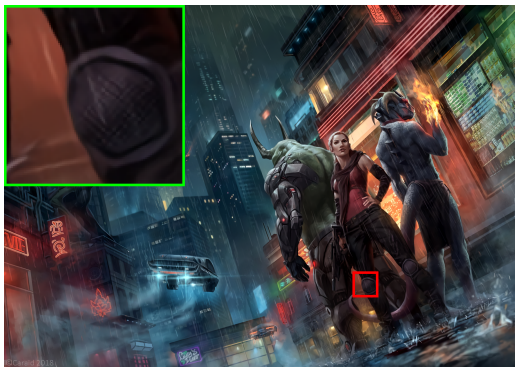
(C) EDSR



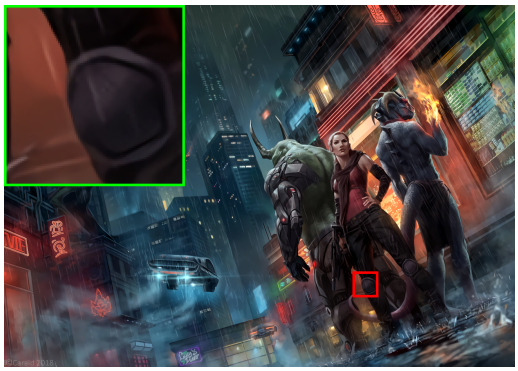
(D) Waifu2x (level 0)



(E) Waifu2x (level 1)



(F) Waifu2x (level 2)



(G) Waifu2x (level 3)

FIGURE 5.2: Results of Waifu2x and my trained model on a clean version (GAUSS_0) of one of the images

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	35.654	34.420	33.751	32.423	34.062
BICUBIC	33.468	26.475	23.640	18.774	25.589
Waifu2X (level 0)	34.854	25.066	21.715	16.359	24.499
Waifu2X (level 1)	34.222	25.146	21.966	16.744	24.520
Waifu2X (level 2)	33.739	25.185	22.152	17.493	24.642
Waifu2X (level 3)	32.656	27.014	24.529	19.467	25.916

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	35.183	32.176	30.895	29.394	31.912
BICUBIC	30.696	31.013	29.886	28.657	30.063
Waifu2X (level 0)	34.726	31.775	30.182	28.543	31.306
Waifu2X (level 1)	34.090	32.093	30.672	28.950	31.451
Waifu2X (level 2)	33.563	31.898	30.880	29.443	31.446
Waifu2X (level 3)	32.613	31.832	30.923	29.507	31.218

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.960	0.940	0.929	0.898	0.932
BICUBIC	0.938	0.750	0.582	0.316	0.647
Waifu2X (level 0)	0.948	0.647	0.435	0.193	0.556
Waifu2X (level 1)	0.938	0.647	0.448	0.207	0.560
Waifu2X (level 2)	0.924	0.629	0.448	0.238	0.560
Waifu2X (level 3)	0.899	0.856	0.742	0.410	0.727

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.956	0.892	0.857	0.807	0.878
BICUBIC	0.861	0.869	0.832	0.785	0.837
Waifu2X (level 0)	0.948	0.887	0.843	0.785	0.866
Waifu2X (level 1)	0.939	0.892	0.854	0.798	0.871
Waifu2X (level 2)	0.927	0.882	0.855	0.811	0.869
Waifu2X (level 3)	0.899	0.880	0.856	0.813	0.862

(E) SSIM results with JPEG noise

FIGURE 5.3: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	35.058	33.250	32.416	30.911	32.909
BICUBIC	32.431	25.006	21.857	17.508	24.201
Waifu2X (level 0)	34.572	23.360	19.743	15.036	23.178
Waifu2X (level 1)	34.421	23.542	20.019	15.399	23.345
Waifu2X (level 2)	34.421	23.830	20.535	16.336	23.780
Waifu2X (level 3)	31.610	25.531	22.492	18.001	24.409

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	33.678	30.792	29.490	28.053	30.503
BICUBIC	28.621	28.908	27.900	26.841	28.068
Waifu2X (level 0)	33.589	29.977	28.418	26.865	29.712
Waifu2X (level 1)	32.980	30.537	29.049	27.330	29.974
Waifu2X (level 2)	32.589	30.462	29.401	27.931	30.096
Waifu2X (level 3)	31.411	30.429	29.471	28.053	29.841

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.948	0.916	0.902	0.870	0.909
BICUBIC	0.921	0.736	0.573	0.318	0.637
Waifu2X (level 0)	0.939	0.670	0.457	0.207	0.569
Waifu2X (level 1)	0.938	0.675	0.476	0.223	0.578
Waifu2X (level 2)	0.938	0.653	0.472	0.251	0.578
Waifu2X (level 3)	0.878	0.829	0.726	0.409	0.711

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.929	0.865	0.833	0.788	0.854
BICUBIC	0.805	0.811	0.780	0.741	0.784
Waifu2X (level 0)	0.924	0.844	0.799	0.743	0.827
Waifu2X (level 1)	0.914	0.859	0.820	0.763	0.839
Waifu2X (level 2)	0.905	0.855	0.830	0.789	0.845
Waifu2X (level 3)	0.875	0.854	0.833	0.793	0.839

(E) SSIM results with JPEG noise

FIGURE 5.4: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	36.827	34.407	33.307	31.247	33.947
BICUBIC	32.417	26.025	22.425	17.809	24.669
Waifu2X (level 0)	36.389	24.495	20.316	15.295	24.124
Waifu2X (level 1)	36.244	24.770	20.623	15.674	24.328
Waifu2X (level 2)	36.147	25.005	21.210	16.599	24.740
Waifu2X (level 3)	32.560	27.203	23.405	18.331	25.375

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	33.761	30.740	29.345	27.767	30.403
BICUBIC	28.043	28.344	27.327	26.265	27.495
Waifu2X (level 0)	34.206	29.636	27.851	26.232	29.481
Waifu2X (level 1)	33.763	30.504	28.728	26.849	29.961
Waifu2X (level 2)	33.211	30.492	29.190	27.565	30.114
Waifu2X (level 3)	32.113	30.606	29.371	27.732	29.955

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.974	0.950	0.938	0.907	0.942
BICUBIC	0.943	0.759	0.600	0.337	0.660
Waifu2X (level 0)	0.971	0.686	0.477	0.221	0.589
Waifu2X (level 1)	0.969	0.686	0.491	0.235	0.595
Waifu2X (level 2)	0.969	0.660	0.483	0.261	0.593
Waifu2X (level 3)	0.921	0.871	0.758	0.426	0.744

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.950	0.896	0.863	0.815	0.881
BICUBIC	0.831	0.836	0.803	0.760	0.808
Waifu2X (level 0)	0.953	0.872	0.821	0.759	0.851
Waifu2X (level 1)	0.946	0.893	0.851	0.787	0.869
Waifu2X (level 2)	0.940	0.890	0.863	0.817	0.878
Waifu2X (level 3)	0.916	0.892	0.868	0.822	0.875

(E) SSIM results with JPEG noise

FIGURE 5.5: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	43.314	41.144	40.155	38.120	40.683
BICUBIC	39.634	23.934	20.738	16.880	25.296
Waifu2X (level 0)	44.156	21.552	18.299	14.210	24.554
Waifu2X (level 1)	43.405	21.656	18.493	14.519	24.518
Waifu2X (level 2)	43.050	21.717	18.865	15.439	24.768
Waifu2X (level 3)	39.541	23.404	20.546	16.995	25.122

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	40.456	37.690	36.033	33.793	36.993
BICUBIC	34.407	34.713	33.529	32.103	33.688
Waifu2X (level 0)	41.605	36.635	34.488	32.247	36.244
Waifu2X (level 1)	40.980	37.706	35.559	32.938	36.796
Waifu2X (level 2)	40.289	37.749	36.218	33.884	37.035
Waifu2X (level 3)	39.143	37.810	36.425	34.174	36.888

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.987	0.977	0.972	0.959	0.974
BICUBIC	0.981	0.690	0.472	0.202	0.586
Waifu2X (level 0)	0.987	0.548	0.309	0.102	0.487
Waifu2X (level 1)	0.985	0.557	0.329	0.114	0.496
Waifu2X (level 2)	0.985	0.516	0.319	0.141	0.490
Waifu2X (level 3)	0.971	0.879	0.710	0.297	0.714

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.977	0.961	0.950	0.927	0.954
BICUBIC	0.927	0.929	0.916	0.897	0.917
Waifu2X (level 0)	0.980	0.952	0.931	0.901	0.941
Waifu2X (level 1)	0.978	0.961	0.946	0.916	0.950
Waifu2X (level 2)	0.974	0.962	0.953	0.934	0.956
Waifu2X (level 3)	0.970	0.964	0.956	0.939	0.957

(E) SSIM results with JPEG noise

FIGURE 5.6: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	44.286	41.665	40.638	38.430	41.255
BICUBIC	37.321	20.818	18.697	15.851	23.172
Waifu2X (level 0)	44.030	19.043	16.753	13.554	23.345
Waifu2X (level 1)	43.801	19.122	16.873	13.772	23.392
Waifu2X (level 2)	43.417	19.897	17.751	14.937	24.000
Waifu2X (level 3)	39.475	21.239	19.128	16.215	24.014

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	40.389	37.379	35.621	33.383	36.693
BICUBIC	33.277	33.611	32.465	31.091	32.611
Waifu2X (level 0)	41.436	36.296	33.664	31.298	35.674
Waifu2X (level 1)	41.048	37.536	35.065	32.196	36.461
Waifu2X (level 2)	40.384	37.567	35.931	33.487	36.842
Waifu2X (level 3)	39.033	37.719	36.163	33.714	36.657

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.992	0.985	0.981	0.970	0.982
BICUBIC	0.982	0.682	0.476	0.216	0.589
Waifu2X (level 0)	0.992	0.549	0.323	0.117	0.495
Waifu2X (level 1)	0.991	0.549	0.335	0.127	0.500
Waifu2X (level 2)	0.991	0.516	0.334	0.156	0.499
Waifu2X (level 3)	0.981	0.883	0.714	0.306	0.721

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.986	0.972	0.962	0.941	0.965
BICUBIC	0.937	0.939	0.924	0.903	0.926
Waifu2X (level 0)	0.988	0.964	0.940	0.906	0.949
Waifu2X (level 1)	0.986	0.974	0.959	0.926	0.961
Waifu2X (level 2)	0.984	0.975	0.968	0.950	0.969
Waifu2X (level 3)	0.980	0.976	0.970	0.953	0.970

(E) SSIM results with JPEG noise

FIGURE 5.7: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	43.120	40.287	39.476	37.806	40.172
BICUBIC	40.625	27.655	23.968	18.709	27.739
Waifu2X (level 0)	43.537	25.698	21.776	16.284	26.824
Waifu2X (level 1)	43.449	25.812	22.052	16.701	27.003
Waifu2X (level 2)	43.335	25.905	22.354	17.584	27.294
Waifu2X (level 3)	39.142	28.687	25.123	19.670	28.156

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	39.804	37.603	36.554	34.790	37.188
BICUBIC	35.678	35.934	34.766	33.299	34.919
Waifu2X (level 0)	40.554	37.232	35.498	33.421	36.676
Waifu2X (level 1)	39.960	37.622	36.284	34.034	36.975
Waifu2X (level 2)	39.744	37.579	36.649	34.824	37.199
Waifu2X (level 3)	38.635	37.618	36.817	35.044	37.029

(C) PSNR results with JPEG noise

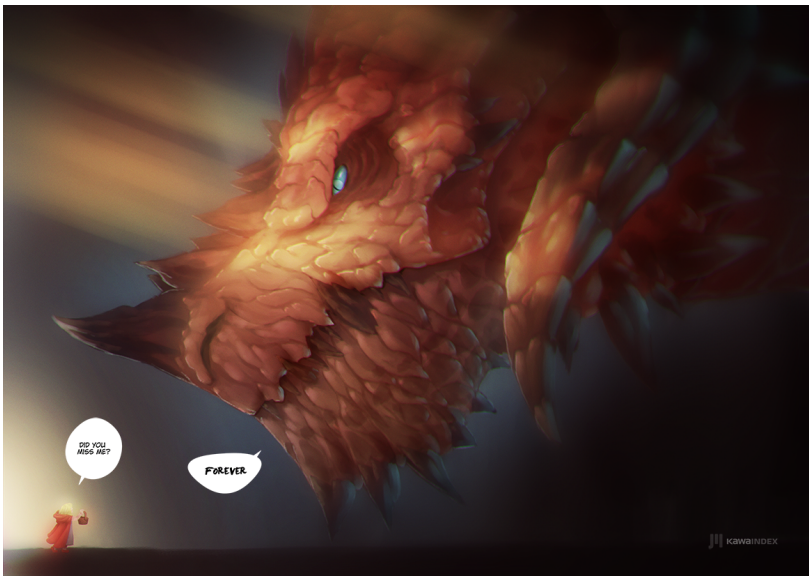
	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.980	0.965	0.960	0.945	0.963
BICUBIC	0.977	0.704	0.493	0.219	0.598
Waifu2X (level 0)	0.980	0.555	0.323	0.114	0.493
Waifu2X (level 1)	0.981	0.563	0.341	0.127	0.503
Waifu2X (level 2)	0.981	0.528	0.333	0.153	0.499
Waifu2X (level 3)	0.957	0.898	0.747	0.339	0.735

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.970	0.952	0.940	0.920	0.945
BICUBIC	0.927	0.929	0.915	0.893	0.916
Waifu2X (level 0)	0.971	0.945	0.926	0.896	0.934
Waifu2X (level 1)	0.968	0.951	0.937	0.909	0.941
Waifu2X (level 2)	0.967	0.950	0.942	0.924	0.946
Waifu2X (level 3)	0.957	0.951	0.944	0.927	0.945

(E) SSIM results with JPEG noise

FIGURE 5.8: PSNR and SSIM results on the shown image



(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	38.657	37.307	36.989	35.834	37.197
BICUBIC	36.640	23.974	22.144	18.174	25.233
Waifu2X (level 0)	38.728	22.545	20.320	15.860	24.363
Waifu2X (level 1)	38.678	22.568	20.497	16.203	24.487
Waifu2X (level 2)	38.538	22.437	20.615	17.108	24.674
Waifu2X (level 3)	36.834	23.932	22.645	18.879	25.573

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	37.850	36.340	35.332	33.671	35.798
BICUBIC	34.438	34.563	33.744	32.538	33.821
Waifu2X (level 0)	37.972	36.134	34.784	32.862	35.438
Waifu2X (level 1)	37.762	36.429	35.267	33.326	35.696
Waifu2X (level 2)	37.465	36.273	35.492	34.005	35.809
Waifu2X (level 3)	36.707	36.162	35.513	34.130	35.628

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.968	0.956	0.952	0.938	0.954
BICUBIC	0.967	0.683	0.472	0.202	0.581
Waifu2X (level 0)	0.968	0.525	0.298	0.100	0.473
Waifu2X (level 1)	0.968	0.530	0.314	0.112	0.481
Waifu2X (level 2)	0.969	0.491	0.304	0.144	0.477
Waifu2X (level 3)	0.949	0.879	0.707	0.308	0.711

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.956	0.941	0.931	0.910	0.934
BICUBIC	0.924	0.925	0.914	0.893	0.914
Waifu2X (level 0)	0.959	0.940	0.925	0.897	0.930
Waifu2X (level 1)	0.957	0.944	0.932	0.908	0.935
Waifu2X (level 2)	0.954	0.943	0.936	0.921	0.938
Waifu2X (level 3)	0.948	0.943	0.937	0.923	0.938

(E) SSIM results with JPEG noise

FIGURE 5.9: PSNR and SSIM results on the shown image

Chapter 6

Conclusion

My model achieves performance comparable to the best waifu2x model match for a given noise level for JPEG tasks, while dominating in gaussian denoising, I suspect the reason why it doesn't consistently outperform it in JPEG denoising (apart from the average score) is that the patch size is overly small (32 x 32), I used the L2 metric as a "safe" option since I was adding new functionality to the network apart from just uscaling, despite L1 having empirically better results (Lim, Son, Kim, Nah, and Lee, 2017) and the lack of Batch Normalization layers, which are proven to degrade performance slightly (Zhang, Zuo, Chen, Meng, and Zhang, 2017), however the fact that my model performs substantially better when the picture has a lot of fine detail 5.2 suggest that there might actually be an overfitting issue, after all EDSR is a highly complex model, and artwork is generally not as complex as photographs.

6.1 Future work

I have found a small handful of images with compression artifacts that differ from JPEG compression, I will try to find the optimizer that generated those artifacts and include it in the training data, as well as other downsampling methods to fix aliasing or other issues that might occur on low resolution images.

List of Abbreviations

CNN	**C**onvolutional **N**eural **N**etwork
GAN	**G**enerative **A**dversarial **N**etwork
GUI	**G**raphical **U**ser **I**nterface
JSON	**J**ava **S**cript **O**bject **N**otation
SR	**S**uper **R**esolution
HR	**H**igh **R**esolution
LR	**L**ow **R**esolution
SISR	**S**ingle **I**mage **S**uper **R**esolution
PSNR	**P**eak **S**ignal to **N**oise **R**atio
SSIM	**S**tructural **SIM**ilarity Index

Acknowledgements

I thank my advisor for pointing me in the right direction with this project when he talked to me about the NITRE challenge and gave me the EDSR and CinCGAN papers to read, as a researcher in the field of Computer Vision he would know about the latest developments in Super-Resolution much better than a master degree student searching on Google Scholar.

I also want to thank my colleague Jordan that left the program after the first semester for making that trial by fire bearable enough for me to get this far.

Appendix A

Results for 3X scale

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	32.134	31.236	30.693	29.601	30.916
BICUBIC	31.204	25.352	22.588	17.728	24.218
Waifu2X (level 0)	31.642	24.363	21.109	15.770	23.221
Waifu2X (level 1)	31.367	24.427	21.368	16.188	23.337
Waifu2X (level 2)	31.239	24.465	21.538	16.947	23.547
Waifu2X (level 3)	30.010	26.195	23.975	19.126	24.826

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	31.701	29.639	28.680	27.463	29.371
BICUBIC	30.525	29.032	28.101	26.988	28.662
Waifu2X (level 0)	31.351	29.175	27.997	26.739	28.816
Waifu2X (level 1)	30.995	29.487	28.422	27.061	28.991
Waifu2X (level 2)	30.732	29.393	28.625	27.441	29.048
Waifu2X (level 3)	29.954	29.341	28.687	27.533	28.879

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.912	0.880	0.863	0.822	0.869
BICUBIC	0.893	0.672	0.506	0.270	0.585
Waifu2X (level 0)	0.898	0.609	0.414	0.189	0.527
Waifu2X (level 1)	0.887	0.605	0.424	0.203	0.530
Waifu2X (level 2)	0.879	0.587	0.421	0.231	0.529
Waifu2X (level 3)	0.833	0.796	0.700	0.405	0.683

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.904	0.824	0.785	0.738	0.813
BICUBIC	0.872	0.803	0.763	0.717	0.789
Waifu2X (level 0)	0.894	0.814	0.766	0.713	0.797
Waifu2X (level 1)	0.882	0.822	0.780	0.727	0.803
Waifu2X (level 2)	0.871	0.812	0.783	0.742	0.802
Waifu2X (level 3)	0.832	0.809	0.784	0.744	0.792

(E) SSIM results with JPEG noise

FIGURE A.1: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	30.691	29.816	29.277	28.178	29.490
BICUBIC	29.982	23.673	20.664	16.412	22.683
Waifu2X (level 0)	30.462	22.624	19.184	14.504	21.693
Waifu2X (level 1)	30.434	22.784	19.472	14.901	21.898
Waifu2X (level 2)	30.394	23.020	19.956	15.858	22.307
Waifu2X (level 3)	28.736	24.529	21.873	17.656	23.198

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	29.890	28.095	27.153	26.031	27.792
BICUBIC	28.675	27.113	26.244	25.226	26.814
Waifu2X (level 0)	29.892	27.426	26.269	25.044	27.158
Waifu2X (level 1)	29.519	27.860	26.777	25.444	27.400
Waifu2X (level 2)	29.275	27.836	27.074	25.923	27.527
Waifu2X (level 3)	28.583	27.860	27.125	26.016	27.396

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.886	0.853	0.836	0.802	0.844
BICUBIC	0.873	0.662	0.504	0.276	0.579
Waifu2X (level 0)	0.885	0.636	0.437	0.200	0.539
Waifu2X (level 1)	0.883	0.634	0.453	0.217	0.547
Waifu2X (level 2)	0.883	0.611	0.446	0.244	0.546
Waifu2X (level 3)	0.816	0.774	0.688	0.405	0.671

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.864	0.796	0.763	0.721	0.786
BICUBIC	0.823	0.752	0.719	0.679	0.743
Waifu2X (level 0)	0.864	0.775	0.729	0.677	0.761
Waifu2X (level 1)	0.851	0.793	0.752	0.698	0.774
Waifu2X (level 2)	0.841	0.789	0.764	0.723	0.779
Waifu2X (level 3)	0.812	0.790	0.767	0.728	0.774

(E) SSIM results with JPEG noise

FIGURE A.2: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	31.133	30.088	29.401	28.023	29.661
BICUBIC	29.610	24.416	21.114	16.678	22.954
Waifu2X (level 0)	30.971	23.548	19.647	14.735	22.225
Waifu2X (level 1)	30.922	23.756	19.957	15.131	22.441
Waifu2X (level 2)	30.868	23.926	20.445	16.032	22.818
Waifu2X (level 3)	28.937	25.733	22.628	17.923	23.805

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	29.775	27.750	26.759	25.515	27.450
BICUBIC	28.016	26.472	25.551	24.488	26.132
Waifu2X (level 0)	29.849	26.921	25.545	24.262	26.644
Waifu2X (level 1)	29.567	27.528	26.248	24.770	27.028
Waifu2X (level 2)	29.288	27.546	26.663	25.351	27.212
Waifu2X (level 3)	28.620	27.623	26.766	25.473	27.120

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.918	0.887	0.871	0.834	0.878
BICUBIC	0.893	0.684	0.525	0.289	0.598
Waifu2X (level 0)	0.919	0.652	0.455	0.212	0.560
Waifu2X (level 1)	0.918	0.647	0.467	0.227	0.565
Waifu2X (level 2)	0.918	0.620	0.456	0.250	0.561
Waifu2X (level 3)	0.851	0.809	0.713	0.417	0.698

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.895	0.824	0.788	0.738	0.811
BICUBIC	0.845	0.770	0.730	0.681	0.757
Waifu2X (level 0)	0.894	0.798	0.741	0.678	0.778
Waifu2X (level 1)	0.884	0.822	0.775	0.708	0.797
Waifu2X (level 2)	0.876	0.818	0.789	0.740	0.806
Waifu2X (level 3)	0.845	0.819	0.792	0.744	0.800

(E) SSIM results with JPEG noise

FIGURE A.3: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	38.733	37.466	36.791	35.204	37.049
BICUBIC	37.091	22.718	19.650	15.750	23.802
Waifu2X (level 0)	38.937	21.150	17.932	13.708	22.932
Waifu2X (level 1)	38.583	21.240	18.120	14.024	22.992
Waifu2X (level 2)	38.440	21.247	18.455	14.950	23.273
Waifu2X (level 3)	36.262	22.782	20.144	16.598	23.947

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	37.236	34.849	33.488	31.678	34.313
BICUBIC	34.989	32.992	31.892	30.480	32.588
Waifu2X (level 0)	37.709	33.982	32.235	30.419	33.586
Waifu2X (level 1)	37.271	34.787	33.100	31.012	34.043
Waifu2X (level 2)	36.889	34.854	33.617	31.804	34.291
Waifu2X (level 3)	36.017	34.907	33.809	32.054	34.197

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.974	0.963	0.959	0.945	0.960
BICUBIC	0.970	0.638	0.429	0.187	0.556
Waifu2X (level 0)	0.976	0.556	0.328	0.114	0.494
Waifu2X (level 1)	0.974	0.559	0.345	0.128	0.502
Waifu2X (level 2)	0.974	0.518	0.333	0.160	0.496
Waifu2X (level 3)	0.957	0.874	0.728	0.335	0.724

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.965	0.944	0.931	0.909	0.937
BICUBIC	0.942	0.912	0.898	0.878	0.908
Waifu2X (level 0)	0.968	0.933	0.911	0.882	0.924
Waifu2X (level 1)	0.965	0.944	0.927	0.898	0.934
Waifu2X (level 2)	0.961	0.946	0.936	0.917	0.940
Waifu2X (level 3)	0.956	0.948	0.939	0.922	0.941

(E) SSIM results with JPEG noise

FIGURE A.4: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	37.880	36.654	35.982	34.413	36.232
BICUBIC	34.389	19.802	17.668	14.803	21.666
Waifu2X (level 0)	37.442	18.719	16.379	13.101	21.410
Waifu2X (level 1)	37.382	18.806	16.511	13.345	21.511
Waifu2X (level 2)	37.234	19.618	17.416	14.547	22.204
Waifu2X (level 3)	34.941	20.935	18.831	15.933	22.660

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	36.149	33.927	32.543	30.841	33.365
BICUBIC	32.988	31.535	30.514	29.200	31.059
Waifu2X (level 0)	36.205	32.915	30.952	29.094	32.292
Waifu2X (level 1)	35.949	33.800	32.109	29.903	32.940
Waifu2X (level 2)	35.568	33.771	32.717	30.987	33.261
Waifu2X (level 3)	34.661	33.849	32.863	31.139	33.128

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.982	0.973	0.969	0.957	0.970
BICUBIC	0.967	0.632	0.434	0.199	0.558
Waifu2X (level 0)	0.982	0.559	0.339	0.127	0.502
Waifu2X (level 1)	0.982	0.552	0.349	0.139	0.505
Waifu2X (level 2)	0.981	0.520	0.347	0.172	0.505
Waifu2X (level 3)	0.966	0.878	0.731	0.340	0.729

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.975	0.954	0.940	0.915	0.946
BICUBIC	0.947	0.914	0.895	0.869	0.906
Waifu2X (level 0)	0.976	0.942	0.910	0.872	0.925
Waifu2X (level 1)	0.974	0.957	0.936	0.898	0.941
Waifu2X (level 2)	0.971	0.957	0.948	0.927	0.951
Waifu2X (level 3)	0.965	0.959	0.950	0.930	0.951

(E) SSIM results with JPEG noise

FIGURE A.5: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	38.020	36.761	36.234	35.060	36.519
BICUBIC	37.114	26.324	22.804	17.670	25.978
Waifu2X (level 0)	37.785	24.984	21.162	15.729	24.915
Waifu2X (level 1)	37.817	25.094	21.447	16.168	25.132
Waifu2X (level 2)	37.729	25.121	21.713	17.064	25.407
Waifu2X (level 3)	35.864	27.893	24.597	19.358	26.928

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	36.768	35.228	34.303	32.817	34.779
BICUBIC	35.523	33.962	32.979	31.579	33.511
Waifu2X (level 0)	36.734	34.588	33.184	31.457	33.991
Waifu2X (level 1)	36.322	34.904	33.889	32.077	34.298
Waifu2X (level 2)	36.305	34.885	34.176	32.814	34.545
Waifu2X (level 3)	35.556	34.926	34.282	32.982	34.436

(C) PSNR results with JPEG noise

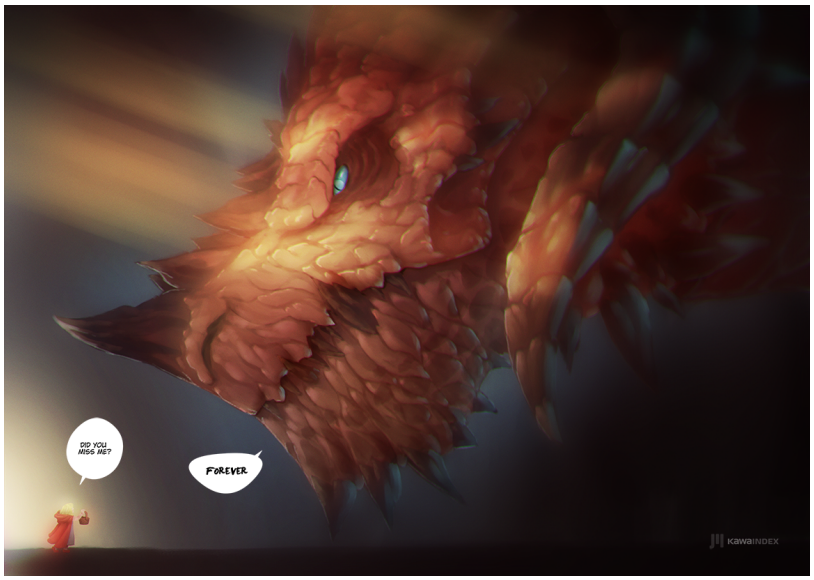
	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.964	0.949	0.944	0.929	0.946
BICUBIC	0.964	0.646	0.443	0.201	0.563
Waifu2X (level 0)	0.964	0.555	0.338	0.128	0.496
Waifu2X (level 1)	0.964	0.558	0.353	0.143	0.505
Waifu2X (level 2)	0.964	0.520	0.340	0.170	0.498
Waifu2X (level 3)	0.943	0.886	0.753	0.373	0.739

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.957	0.937	0.923	0.898	0.929
BICUBIC	0.940	0.911	0.893	0.866	0.902
Waifu2X (level 0)	0.958	0.927	0.902	0.867	0.913
Waifu2X (level 1)	0.955	0.935	0.917	0.883	0.923
Waifu2X (level 2)	0.953	0.935	0.924	0.903	0.929
Waifu2X (level 3)	0.942	0.935	0.926	0.906	0.928

(E) SSIM results with JPEG noise

FIGURE A.6: PSNR and SSIM results on the shown image



(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	36.631	35.677	35.186	34.493	35.497
BICUBIC	35.346	23.382	21.349	17.229	24.326
Waifu2X (level 0)	36.535	22.422	19.967	15.379	23.576
Waifu2X (level 1)	36.500	22.467	20.177	15.761	23.726
Waifu2X (level 2)	36.420	22.532	20.422	16.749	24.031
Waifu2X (level 3)	34.920	24.123	22.646	18.726	25.104

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	36.201	34.693	33.674	32.196	34.191
BICUBIC	34.512	33.409	32.564	31.275	32.940
Waifu2X (level 0)	36.004	34.278	32.895	31.290	33.617
Waifu2X (level 1)	35.842	34.556	33.470	31.784	33.913
Waifu2X (level 2)	35.630	34.455	33.785	32.422	34.073
Waifu2X (level 3)	34.835	34.410	33.827	32.539	33.903

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.959	0.948	0.942	0.929	0.945
BICUBIC	0.959	0.631	0.426	0.186	0.550
Waifu2X (level 0)	0.960	0.531	0.315	0.116	0.480
Waifu2X (level 1)	0.960	0.528	0.329	0.130	0.487
Waifu2X (level 2)	0.960	0.488	0.315	0.162	0.481
Waifu2X (level 3)	0.940	0.876	0.722	0.346	0.721

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.951	0.930	0.919	0.897	0.925
BICUBIC	0.937	0.915	0.901	0.879	0.908
Waifu2X (level 0)	0.952	0.927	0.910	0.882	0.918
Waifu2X (level 1)	0.949	0.932	0.920	0.894	0.924
Waifu2X (level 2)	0.946	0.931	0.924	0.908	0.927
Waifu2X (level 3)	0.939	0.932	0.925	0.910	0.927

(E) SSIM results with JPEG noise

FIGURE A.7: PSNR and SSIM results on the shown image

Appendix B

Results for 4X scale

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	30.715	29.929	29.432	28.413	29.622
BICUBIC	29.740	24.931	22.427	17.660	23.689
Waifu2X (level 0)	30.381	24.220	21.191	15.915	22.927
Waifu2X (level 1)	30.221	24.277	21.435	16.325	23.064
Waifu2X (level 2)	30.159	24.304	21.595	17.049	23.277
Waifu2X (level 3)	28.841	25.712	23.801	19.068	24.356

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	30.245	28.438	27.464	26.462	28.152
BICUBIC	29.176	27.833	27.013	26.009	27.508
Waifu2X (level 0)	30.054	28.016	26.879	25.751	27.675
Waifu2X (level 1)	29.738	28.250	27.204	26.039	27.808
Waifu2X (level 2)	29.476	28.164	27.389	26.409	27.859
Waifu2X (level 3)	28.770	28.142	27.462	26.492	27.717

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.868	0.834	0.815	0.776	0.823
BICUBIC	0.840	0.655	0.510	0.290	0.574
Waifu2X (level 0)	0.853	0.609	0.432	0.210	0.526
Waifu2X (level 1)	0.846	0.604	0.440	0.225	0.529
Waifu2X (level 2)	0.842	0.588	0.436	0.251	0.529
Waifu2X (level 3)	0.784	0.755	0.679	0.422	0.660

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.856	0.774	0.737	0.700	0.767
BICUBIC	0.822	0.751	0.715	0.677	0.741
Waifu2X (level 0)	0.845	0.763	0.717	0.672	0.749
Waifu2X (level 1)	0.833	0.771	0.731	0.685	0.755
Waifu2X (level 2)	0.823	0.763	0.735	0.700	0.755
Waifu2X (level 3)	0.782	0.760	0.736	0.702	0.745

(E) SSIM results with JPEG noise

FIGURE B.1: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	29.681	28.756	28.211	27.102	28.438
BICUBIC	28.721	23.389	20.613	16.370	22.274
Waifu2X (level 0)	29.610	22.593	19.342	14.634	21.545
Waifu2X (level 1)	29.549	22.739	19.641	15.026	21.739
Waifu2X (level 2)	29.532	23.025	20.081	15.930	22.142
Waifu2X (level 3)	27.764	24.268	21.835	17.627	22.873

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	28.726	26.986	26.048	24.975	26.684
BICUBIC	27.547	26.040	25.202	24.250	25.760
Waifu2X (level 0)	28.866	26.360	25.209	24.082	26.129
Waifu2X (level 1)	28.482	26.807	25.675	24.421	26.346
Waifu2X (level 2)	28.150	26.742	25.928	24.850	26.418
Waifu2X (level 3)	27.561	26.766	25.974	24.910	26.303

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.853	0.818	0.800	0.765	0.809
BICUBIC	0.828	0.651	0.512	0.296	0.572
Waifu2X (level 0)	0.853	0.641	0.458	0.220	0.543
Waifu2X (level 1)	0.852	0.636	0.473	0.238	0.550
Waifu2X (level 2)	0.852	0.617	0.464	0.265	0.549
Waifu2X (level 3)	0.778	0.739	0.669	0.420	0.652

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.826	0.756	0.724	0.687	0.748
BICUBIC	0.782	0.710	0.677	0.641	0.702
Waifu2X (level 0)	0.827	0.733	0.686	0.638	0.721
Waifu2X (level 1)	0.812	0.751	0.709	0.658	0.733
Waifu2X (level 2)	0.799	0.747	0.721	0.682	0.737
Waifu2X (level 3)	0.773	0.748	0.723	0.686	0.732

(E) SSIM results with JPEG noise

FIGURE B.2: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	29.455	28.482	27.838	26.571	28.086
BICUBIC	27.853	23.899	20.842	16.569	22.291
Waifu2X (level 0)	29.384	23.438	19.680	14.828	21.832
Waifu2X (level 1)	29.328	23.642	19.985	15.217	22.043
Waifu2X (level 2)	29.310	23.814	20.476	16.107	22.427
Waifu2X (level 3)	27.426	25.067	22.305	17.817	23.154

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	28.231	26.507	25.498	24.415	26.163
BICUBIC	26.625	25.244	24.385	23.418	24.918
Waifu2X (level 0)	28.233	25.711	24.398	23.234	25.394
Waifu2X (level 1)	27.972	26.212	24.973	23.642	25.700
Waifu2X (level 2)	27.739	26.211	25.307	24.137	25.849
Waifu2X (level 3)	27.119	26.261	25.401	24.250	25.758

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.874	0.839	0.822	0.782	0.829
BICUBIC	0.835	0.661	0.522	0.303	0.580
Waifu2X (level 0)	0.875	0.652	0.469	0.226	0.555
Waifu2X (level 1)	0.874	0.646	0.479	0.242	0.560
Waifu2X (level 2)	0.875	0.621	0.468	0.267	0.558
Waifu2X (level 3)	0.796	0.765	0.688	0.427	0.669

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.851	0.777	0.739	0.695	0.765
BICUBIC	0.794	0.715	0.677	0.635	0.705
Waifu2X (level 0)	0.845	0.745	0.687	0.634	0.728
Waifu2X (level 1)	0.831	0.768	0.719	0.659	0.745
Waifu2X (level 2)	0.825	0.764	0.734	0.688	0.753
Waifu2X (level 3)	0.790	0.765	0.737	0.692	0.746

(E) SSIM results with JPEG noise

FIGURE B.3: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	37.422	36.014	35.372	33.800	35.652
BICUBIC	35.571	22.697	19.601	15.724	23.398
Waifu2X (level 0)	37.632	21.286	18.038	13.834	22.698
Waifu2X (level 1)	37.283	21.361	18.219	14.145	22.752
Waifu2X (level 2)	37.183	21.390	18.547	15.067	23.047
Waifu2X (level 3)	34.906	22.900	20.207	16.666	23.670

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	35.876	33.576	32.356	30.812	33.155
BICUBIC	33.771	31.757	30.713	29.417	31.414
Waifu2X (level 0)	36.361	32.641	31.018	29.354	32.344
Waifu2X (level 1)	35.887	33.387	31.755	29.872	32.725
Waifu2X (level 2)	35.482	33.368	32.193	30.555	32.899
Waifu2X (level 3)	34.594	33.442	32.366	30.772	32.793

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.966	0.953	0.949	0.935	0.951
BICUBIC	0.957	0.672	0.480	0.234	0.586
Waifu2X (level 0)	0.968	0.606	0.384	0.146	0.526
Waifu2X (level 1)	0.966	0.606	0.400	0.163	0.534
Waifu2X (level 2)	0.965	0.569	0.387	0.201	0.531
Waifu2X (level 3)	0.946	0.875	0.752	0.392	0.741

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.957	0.936	0.925	0.907	0.931
BICUBIC	0.931	0.900	0.886	0.868	0.896
Waifu2X (level 0)	0.959	0.921	0.898	0.872	0.913
Waifu2X (level 1)	0.955	0.933	0.914	0.887	0.922
Waifu2X (level 2)	0.951	0.934	0.923	0.905	0.928
Waifu2X (level 3)	0.945	0.935	0.926	0.909	0.929

(E) SSIM results with JPEG noise

FIGURE B.4: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	36.469	35.160	34.448	32.981	34.764
BICUBIC	32.509	19.806	17.651	14.783	21.187
Waifu2X (level 0)	36.227	18.888	16.516	13.234	21.216
Waifu2X (level 1)	36.142	18.964	16.653	13.474	21.308
Waifu2X (level 2)	36.017	19.753	17.561	14.663	21.998
Waifu2X (level 3)	33.511	21.041	18.944	16.011	22.377

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	34.765	32.561	31.413	29.945	32.171
BICUBIC	31.451	30.111	29.184	27.989	29.684
Waifu2X (level 0)	34.744	31.547	29.658	27.918	30.967
Waifu2X (level 1)	34.461	32.327	30.717	28.654	31.540
Waifu2X (level 2)	34.139	32.291	31.312	29.643	31.846
Waifu2X (level 3)	33.223	32.365	31.470	29.799	31.714

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.973	0.964	0.959	0.946	0.960
BICUBIC	0.947	0.661	0.478	0.241	0.582
Waifu2X (level 0)	0.974	0.609	0.392	0.157	0.533
Waifu2X (level 1)	0.973	0.600	0.401	0.171	0.536
Waifu2X (level 2)	0.973	0.570	0.397	0.210	0.538
Waifu2X (level 3)	0.953	0.876	0.752	0.391	0.743

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.967	0.944	0.930	0.911	0.938
BICUBIC	0.929	0.892	0.871	0.847	0.885
Waifu2X (level 0)	0.966	0.926	0.888	0.851	0.908
Waifu2X (level 1)	0.963	0.942	0.918	0.877	0.925
Waifu2X (level 2)	0.961	0.942	0.931	0.909	0.936
Waifu2X (level 3)	0.952	0.944	0.934	0.911	0.935

(E) SSIM results with JPEG noise

FIGURE B.5: PSNR and SSIM results on the shown image

(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	38.122	36.451	35.743	34.339	36.164
BICUBIC	35.990	26.366	22.823	17.664	25.711
Waifu2X (level 0)	37.956	25.285	21.394	15.917	25.138
Waifu2X (level 1)	37.886	25.407	21.683	16.354	25.333
Waifu2X (level 2)	37.838	25.498	21.949	17.232	25.629
Waifu2X (level 3)	35.490	28.149	24.754	19.432	26.956

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	36.923	34.996	33.813	32.297	34.508
BICUBIC	34.740	33.008	31.993	30.575	32.579
Waifu2X (level 0)	36.917	34.168	32.339	30.499	33.481
Waifu2X (level 1)	36.456	34.673	33.214	31.152	33.874
Waifu2X (level 2)	36.357	34.632	33.645	31.975	34.152
Waifu2X (level 3)	35.348	34.659	33.776	32.158	33.985

(C) PSNR results with JPEG noise

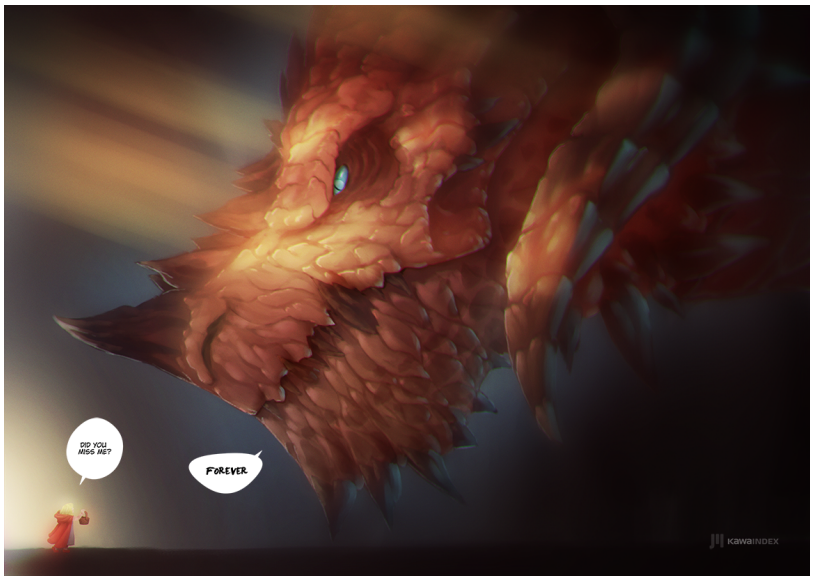
	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.957	0.941	0.933	0.917	0.937
BICUBIC	0.952	0.679	0.491	0.248	0.593
Waifu2X (level 0)	0.957	0.604	0.393	0.166	0.530
Waifu2X (level 1)	0.957	0.605	0.408	0.184	0.538
Waifu2X (level 2)	0.957	0.570	0.393	0.214	0.533
Waifu2X (level 3)	0.931	0.883	0.771	0.429	0.754

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.949	0.927	0.912	0.891	0.920
BICUBIC	0.930	0.894	0.874	0.848	0.886
Waifu2X (level 0)	0.950	0.913	0.884	0.848	0.899
Waifu2X (level 1)	0.947	0.922	0.902	0.866	0.909
Waifu2X (level 2)	0.945	0.922	0.910	0.887	0.916
Waifu2X (level 3)	0.931	0.923	0.912	0.890	0.914

(E) SSIM results with JPEG noise

FIGURE B.6: PSNR and SSIM results on the shown image



(A) High Resolution

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	34.780	34.003	33.756	33.096	33.909
BICUBIC	33.847	23.316	21.330	17.240	23.933
Waifu2X (level 0)	34.251	22.401	20.052	15.564	23.067
Waifu2X (level 1)	34.307	22.458	20.256	15.932	23.238
Waifu2X (level 2)	34.309	22.590	20.481	16.834	23.554
Waifu2X (level 3)	33.245	24.122	22.686	18.714	24.692

(B) PSNR results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	34.379	33.242	32.424	31.394	32.860
BICUBIC	33.266	32.096	31.338	30.217	31.729
Waifu2X (level 0)	33.941	32.665	31.531	30.157	32.073
Waifu2X (level 1)	33.856	32.894	32.055	30.606	32.353
Waifu2X (level 2)	33.770	32.890	32.384	31.207	32.563
Waifu2X (level 3)	33.179	32.841	32.414	31.320	32.438

(C) PSNR results with JPEG noise

	GAUSS_0	GAUSS_15	GAUSS_25	GAUSS_50	Average
EDSR	0.950	0.937	0.933	0.919	0.935
BICUBIC	0.945	0.662	0.474	0.236	0.580
Waifu2X (level 0)	0.950	0.579	0.370	0.155	0.513
Waifu2X (level 1)	0.949	0.573	0.382	0.172	0.519
Waifu2X (level 2)	0.949	0.536	0.366	0.204	0.514
Waifu2X (level 3)	0.929	0.871	0.742	0.405	0.737

(D) SSIM results with gaussian noise

	JPEG_100	JPEG_75	JPEG_50	JPEG_25	Average
EDSR	0.942	0.922	0.912	0.896	0.918
BICUBIC	0.928	0.901	0.888	0.868	0.896
Waifu2X (level 0)	0.943	0.914	0.898	0.871	0.906
Waifu2X (level 1)	0.939	0.920	0.908	0.883	0.913
Waifu2X (level 2)	0.936	0.920	0.912	0.896	0.916
Waifu2X (level 3)	0.929	0.921	0.914	0.898	0.915

(E) SSIM results with JPEG noise

FIGURE B.7: PSNR and SSIM results on the shown image

Bibliography

[1] E. Hamilton, "Jpeg file interchange format", C-Cube Microsystems, Milpitas, CA, USA, Tech. Rep., Sep. 1992. [Online]. Available: http://www.w3.org/Graphics/JPEG/jfif3.pdf.

[2] T. Boutell, *PNG (Portable Network Graphics) Specification Version 1.0*, RFC 2083, Mar. 1997. DOI: 10.17487/RFC2083. [Online]. Available: https://rfc-editor.org/rfc/rfc2083.txt.

[3] Wikipedia contributors, *Lanczos resampling — Wikipedia, the free encyclopedia*, [Online; accessed 8-April-2019], 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Lanczos_resampling&oldid=887426730.

[4] nagadomi, *Waifu2x*, https://github.com/nagadomi/waifu2x, 2019.

[5] lltcggie, *Waifu2x-caffe*, https://github.com/lltcggie/waifu2x-caffe, 2019.

[6] L. He, H. Qi, and R. Zaretzki, "Beta process joint dictionary learning for coupled feature spaces with application to single image super-resolution", in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 345–352. DOI: 10.1109/CVPR.2013.51.

[7] R. Timofte, V. De, and L. V. Gool, "Anchored neighborhood regression for fast example-based super-resolution", in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1920–1927. DOI: 10.1109/ICCV.2013.241.

[8] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution", in *Computer Vision – ACCV 2014*, D. Cremers, I. Reid, H. Saito, and M.-H. Yang, Eds., Cham: Springer International Publishing, 2015, pp. 111–126, ISBN: 978-3-319-16817-3.

[9] R. Timofte, R. Rothe, and L. J. V. Gool, "Seven ways to improve example-based single image super resolution", *CoRR*, vol. abs/1511.02228, 2015. arXiv: 1511.02228. [Online]. Available: http://arxiv.org/abs/1511.02228.

[10] E. Perez-Pellitero, J. Salvador, J. Ruiz-Hidalgo, and B. Rosenhahn, "Psyco: Manifold span reduction for super resolution", in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, Nevada, USA, 2016. [Online]. Available: http://perezpellitero.github.io/.

[11] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation", *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010, ISSN: 1057-7149. DOI: 10.1109/TIP.2010.2050625.

[12] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior", in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 370–378. DOI: 10.1109/ICCV.2015.50.

[13] J. Salvador and E. Pérez-Pellitero, "Naive bayes super-resolution forest", in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 325–333. DOI: 10.1109/ICCV.2015.45.

[14] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests", in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3791–3799. DOI: 10.1109/CVPR.2015.7299003.

[15] J. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars", in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5197–5206. DOI: 10.1109/CVPR.2015.7299156.

[16] Z. Wang, Y. Yang, Z. Wang, S. Chang, W. Han, J. Yang, and T. S. Huang, "Self-tuned deep super resolution", *CoRR*, vol. abs/1504.05632, 2015. arXiv: 1504.05632. [Online]. Available: http://arxiv.org/abs/1504.05632.

[17] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2015.2439281.

[18] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network", *CoRR*, vol. abs/1609.05158, 2016. arXiv: 1609.05158. [Online]. Available: http://arxiv.org/abs/1609.05158.

[19] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network", *CoRR*, vol. abs/1608.00367, 2016. arXiv: 1608.00367. [Online]. Available: http://arxiv.org/abs/1608.00367.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[21] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2790–2798. DOI: 10.1109/CVPR.2017.298.

[22] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks", *CoRR*, vol. abs/1511.04587, 2015. arXiv: 1511.04587. [Online]. Available: http://arxiv.org/abs/1511.04587.

[23] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network", *CoRR*, vol. abs/1609.04802, 2016. arXiv: 1609.04802. [Online]. Available: http://arxiv.org/abs/1609.04802.

[24] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution", *CoRR*, vol. abs/1707.02921, 2017. arXiv: 1707.02921. [Online]. Available: http://arxiv.org/abs/1707.02921.

[25] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution", *CoRR*, vol. abs/1511.04491, 2015. arXiv: 1511.04491. [Online]. Available: http://arxiv.org/abs/1511.04491.

[26] X. Wang, K. Yu, C. Dong, and C. C. Loy, "Recovering realistic texture in image super-resolution by deep spatial feature transform", *CoRR*, vol. abs/1804.02815, 2018. arXiv: 1804.02815. [Online]. Available: http://arxiv.org/abs/1804.02815.

[27] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks", *CoRR*, vol. abs/1809.00437, 2018. arXiv: 1809.00437. [Online]. Available: http://arxiv.org/abs/1809.00437.

[28] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1256–1272, 2017, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2596743.

[29] X. Mao, C. Shen, and Y. Yang, "Image denoising using very deep fully convolutional encoder-decoder networks with symmetric skip connections", *CoRR*, vol. abs/1603.09056, 2016. arXiv: 1603.09056. [Online]. Available: http://arxiv.org/abs/1603.09056.

[30] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering", *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007, ISSN: 1057-7149. DOI: 10.1109/TIP.2007.901238.

[31] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?", in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2392–2399. DOI: 10.1109/CVPR.2012.6247952.

[32] D. Yang and J. Sun, "Bm3d-net: A convolutional neural network for transform-domain collaborative filtering", *IEEE Signal Processing Letters*, vol. 25, no. 1, pp. 55–59, 2018, ISSN: 1070-9908. DOI: 10.1109/LSP.2017.2768660.

[33] tensorflow, *Tensorflow*, https://github.com/tensorflow/tensorflow, 2019.

[34] pytorch, *Pytorch*, https://github.com/pytorch/pytorch, 2019.

[35] Theano, *Theano*, https://github.com/Theano/Theano, 2019.

[36] Microsoft, *Cntk*, https://github.com/Microsoft/CNTK, 2019.

[37] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks", *CoRR*, vol. abs/1703.10593, 2017. arXiv: 1703.10593. [Online]. Available: http://arxiv.org/abs/1703.10593.

[38] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising", *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017, ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2662206.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[41] Bionus, *Imgbrd-grabber*, https://github.com/Bionus/imgbrd-grabber, 2019.

[42] B. Vukorepa, *Cntk 302 part b: Image super-resolution using cnns and gans*, 2017. [Online]. Available: https://cntk.ai/pythondocs/CNTK_302B_Image_Super-resolution_Using_CNNs_and_GANs.html.