SR3 - Diffusion

Image Super-Resolution via Iterative Refinement

Presented by:

Adir Strack, Rashan shkeer, Ofir Nahshon and Maxim Chanturiay

Based on :

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 45, NO. 4, APRIL 2023 4713

Image Super-Resolution via Iterative Refinement

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi

super resolution
generating a high-resolution image that is consistent with an input low-resolution image.

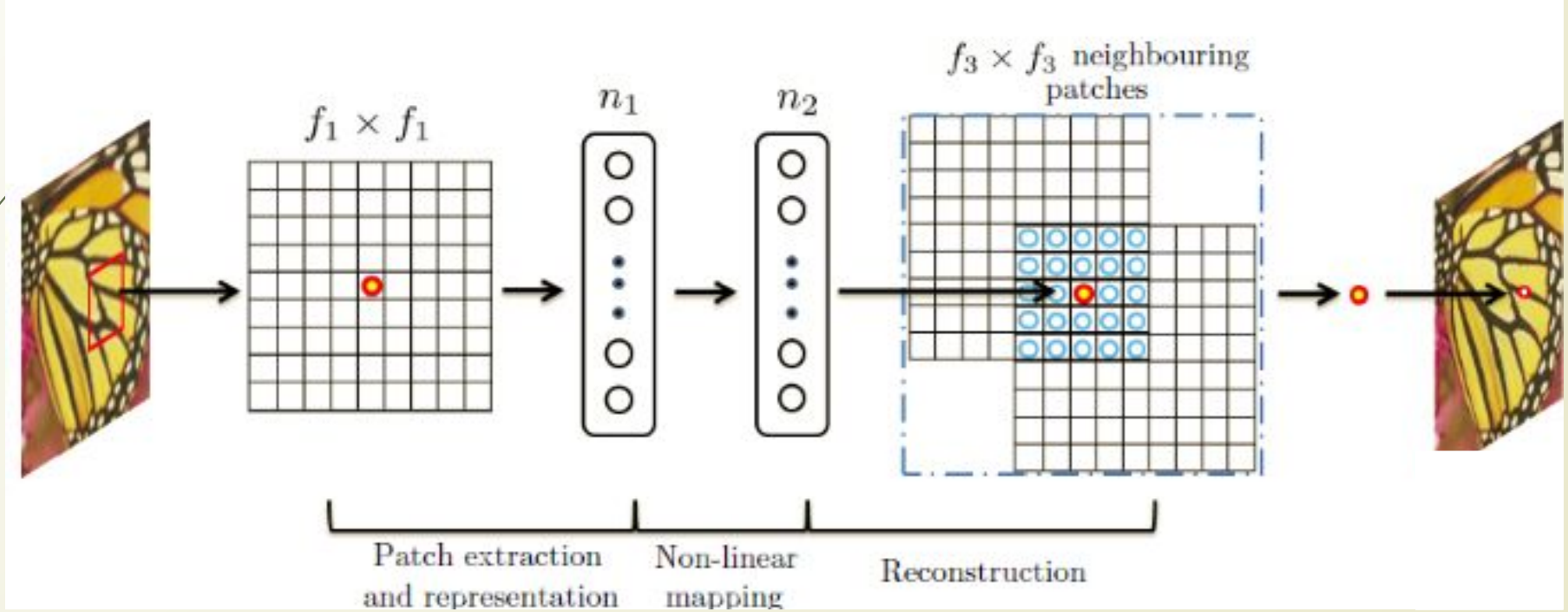
regression-based models

- **regression-based algorithm** is a type of machine learning algorithm used to predict a continuous outcome variable (often called the dependent variable or target) based on one or more predictor variables (also known as independent variables or features).
- The **goal** of regression analysis is to establish a **mathematical relations**hip between the dependent variable and the independent variables, allowing predictions to be made about the dependent variable given new data on the independent variables.

SRCNN: Super resolution Convolution Network

- **SRCNN** takes a low-resolution image as input and generates a high-resolution image.
- It does this by learning a **mapping** from low-resolution images to high-resolution images using a **convolutional neural network** (CNN).
- learns the mapping from low-resolution to high-resolution images **directly from data**.

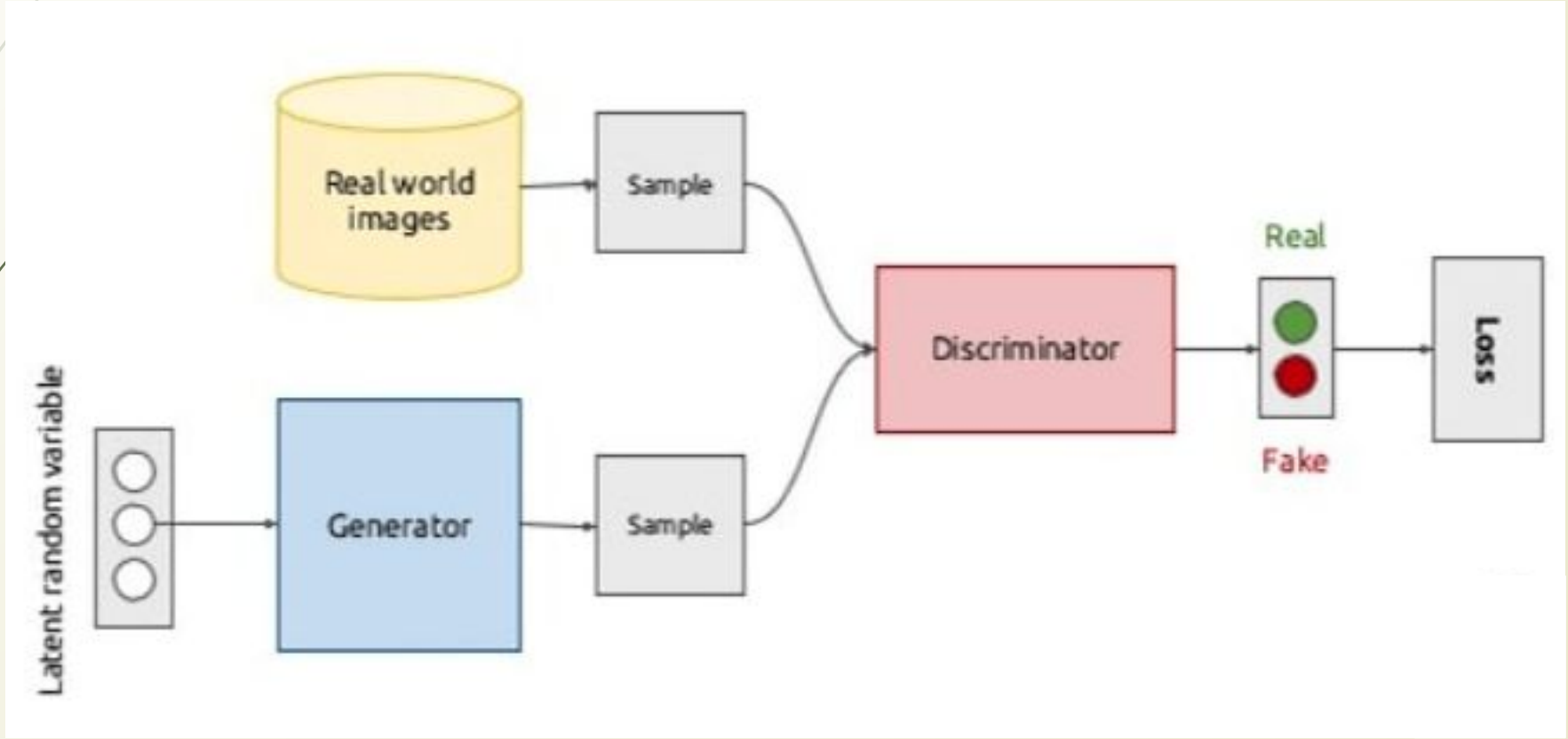
SRCNN


$f_1 \times f_1$
n_1
n_2
$f_3 \times f_3$ neighbouring patches
Patch extraction and representation
Non-linear mapping
Reconstruction

SRGEN: super resolution Generative Adversarial Network

- SRGAN consists of two main components:
  - **Generator**: This network generates high-resolution images from low-resolution inputs.
  - **Discriminator**: This network distinguishes between the generated high-resolution images and real high-resolution images.

SRGEN


Real world images
Sample
Real
Discriminator
Loss
Fake
Latent random variable
Generator
Sample

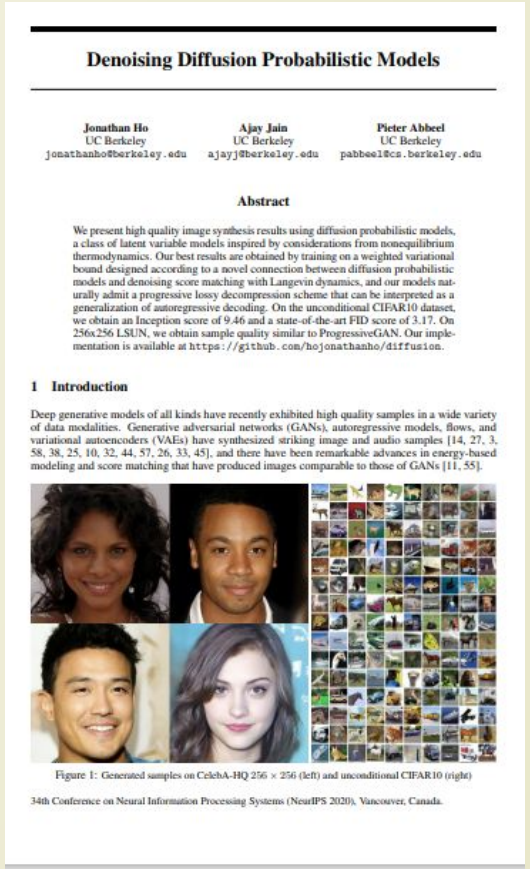
SR3: based on Denoising Diffusion Probabilistic Models (DDPM)

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu



Denoising Diffusion Probabilistic Models

Jonathan Ho UC Berkeley jonathanho@berkeley.edu
Ajay Jain UC Berkeley ajayj@berkeley.edu
Pieter Abbeel UC Berkeley pabbeel@cs.berkeley.edu

Abstract

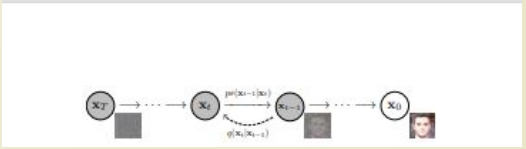
We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

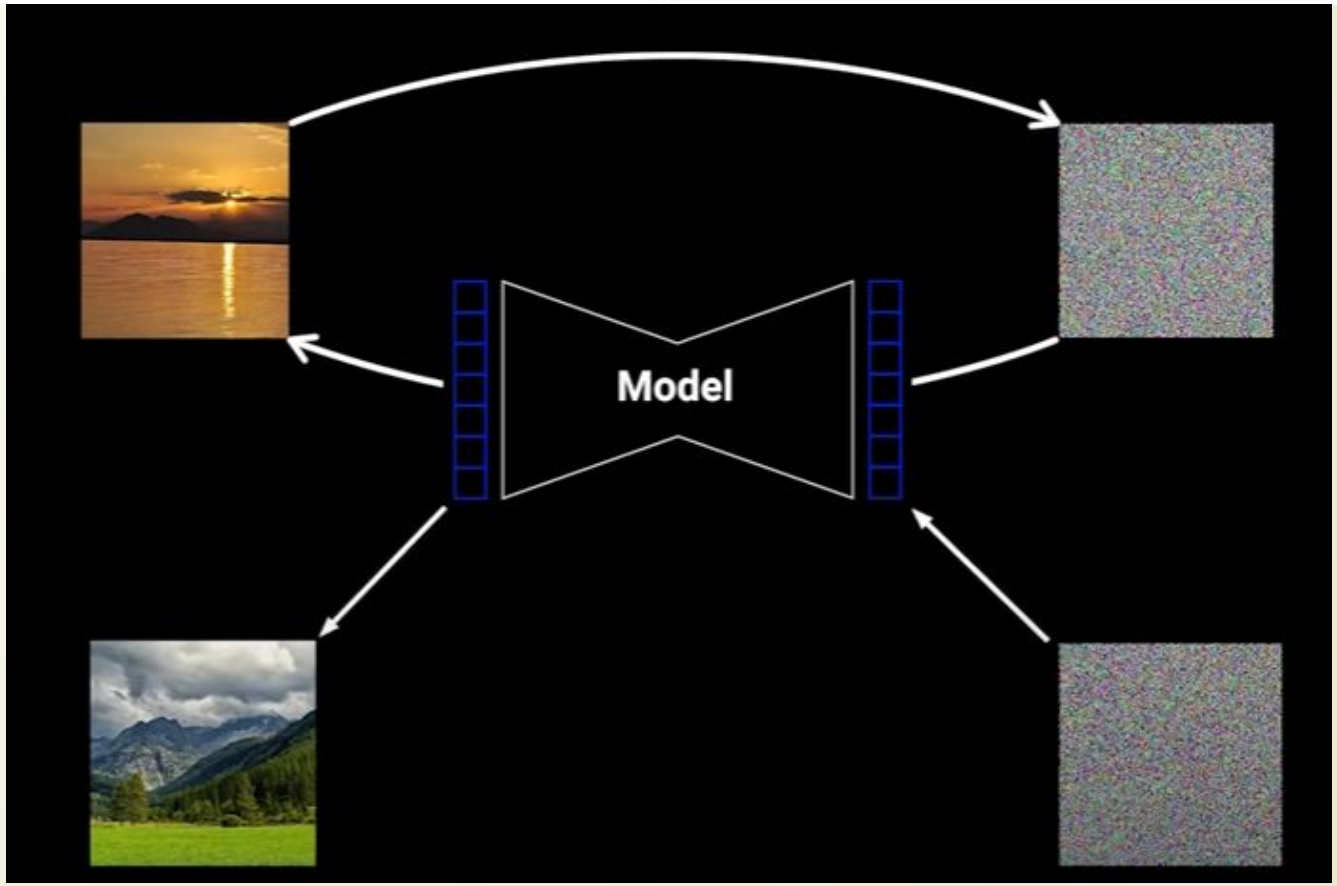


Image Degradation by Gaussian Noise

$$g(x,y) = f(x,y) + n(x,y)$$

$g(x,y)$

$f(x,y)$

Pixel value: [0~1] or [0~255]

$n(x,y)$

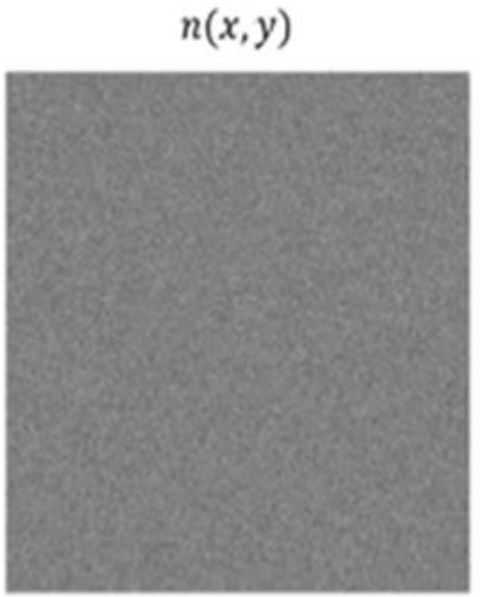
Pixel value: random number

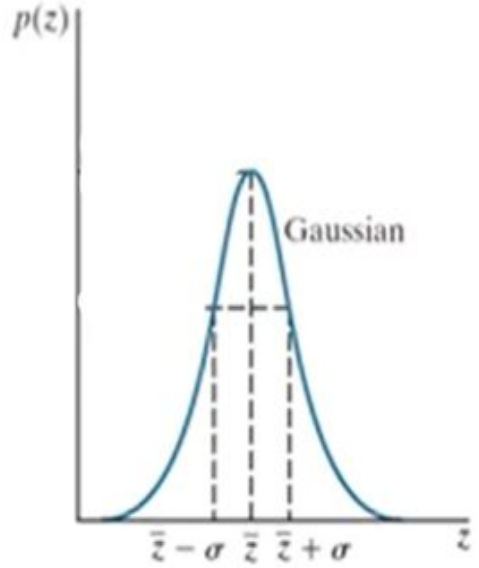
Gaussian Noise Formula:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}} \qquad -\infty < z < \infty$$

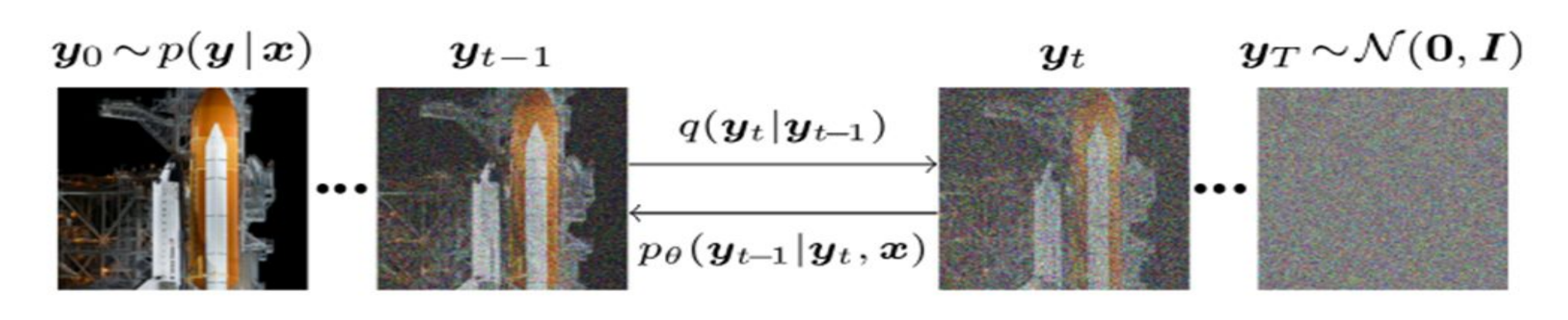
-Basically noise is just an image where the pixel value is a random number.

-when the random number follows the Gaussian distribution the noise is called as a Gaussian noise.





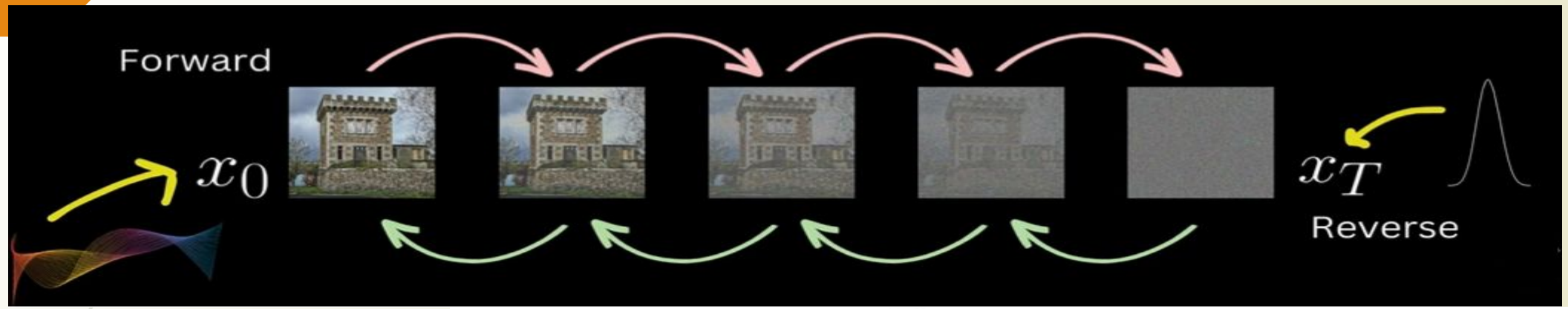
CONDITIONAL DENOISING DIFFUSION MODEL



Denoising diffusion models consist of two processes:

- **Forward diffusion process that gradually adds noise to input**
- **Reverse denoising process that learns to generate data by denoising**

forward diffusion



$$q(\boldsymbol{y}_{1:T} \mid \boldsymbol{y}_0) = \prod_{t=1}^{T} q(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}), \tag{1}$$

$$q(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}) = \mathcal{N}(\boldsymbol{y}_t \mid \sqrt{\alpha_t}\, \boldsymbol{y}_{t-1}, (1-\alpha_t)\boldsymbol{I}), \tag{2}$$

Forward Markovian diffusion process q gradually adds Gaussian noise to a high-resolution image y_0 over T iterations.

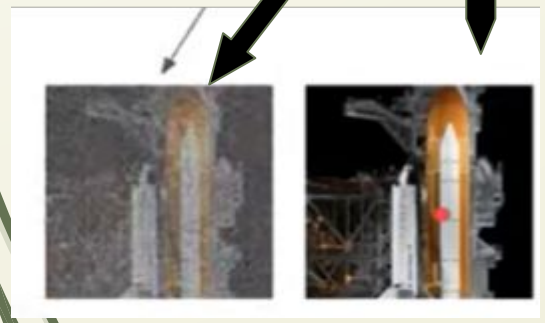
$0 < \alpha_t < 1$

which determine the variance of the noise added at each iteration

To go through the full Markov chain is impractical, as we need to compute every step before y_t in order to derive y_t. A better way is to modify the formula as below:

$$\gamma_t = \prod_{i=1}^{t} \alpha_i$$

$$q(\boldsymbol{y}_t \mid \boldsymbol{y}_0) = \mathcal{N}(\boldsymbol{y}_t \mid \sqrt{\gamma_t}\,\boldsymbol{y}_0, (1-\gamma_t)\boldsymbol{I}),$$



Now, we can obtain y at any timestep based on the y_0 and a certain γ_t.

Backward diffusion

model should 'ideally' predict the amount of noise added at a particular timestep. Continuously denoising the model for 'T' timesteps should yield a clear image.

Optimizing the Denoising Model

we optimize a neural denoising model f_θ that takes as input source image x and a noisy target image $\tilde{y}$

$$\tilde{y} = \sqrt{\gamma}\, y_0 + \sqrt{1-\gamma}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

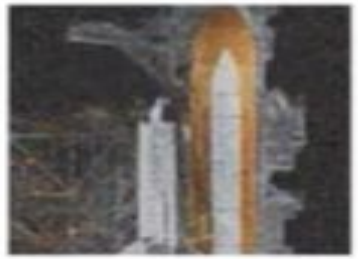
Optimize Denoising

$\boldsymbol{x}$

source

$\tilde{\boldsymbol{y}} = \sqrt{\gamma}\,\boldsymbol{y}_0 + \sqrt{1-\gamma}\,\boldsymbol{\epsilon}$

noisy target

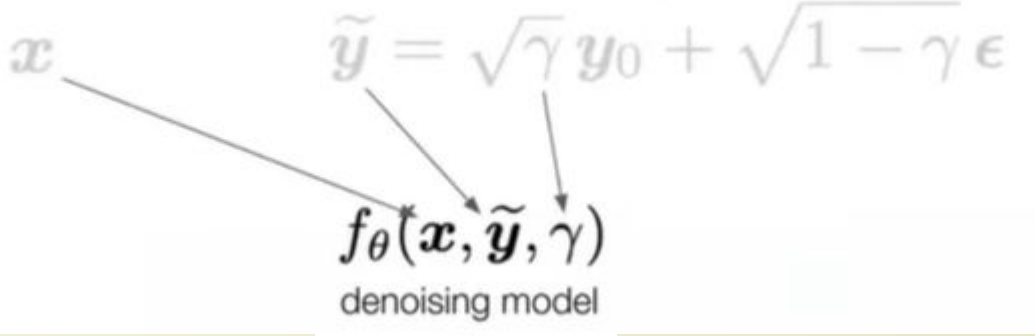


noise variance

$q(\boldsymbol{y}_t \mid \boldsymbol{y}_0) = \mathcal{N}(\boldsymbol{y}_t \mid \sqrt{\gamma_t}\,\boldsymbol{y}_0, (1-\gamma_t)\boldsymbol{I})$

forward distribution

noised step



The proposed objective function for training

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\mathbb{E}_{\boldsymbol{\epsilon},\gamma}\left\| f_\theta(\boldsymbol{x}, \underbrace{\sqrt{\gamma}\,\boldsymbol{y}_0 + \sqrt{1-\gamma}\,\boldsymbol{\epsilon}}_{\tilde{\boldsymbol{y}}}, \gamma) - \boldsymbol{\epsilon} \right\|_p^p$$

Iterative Inference

Inference under our model is defined as a reverse Markovian process,We define the inference process in terms of isotropic Gaussian conditional distributions.

$$p_\theta(\boldsymbol{y}_{0:T}|\boldsymbol{x}) = p(\boldsymbol{y}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{y}_{t-1}|\boldsymbol{y}_t,\boldsymbol{x})$$

$$p(\boldsymbol{y}_T) = \mathcal{N}(\boldsymbol{y}_T\,|\,\boldsymbol{0},\boldsymbol{I})$$

$$p_\theta(\boldsymbol{y}_{t-1}|\boldsymbol{y}_t,\boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}_{t-1}\,|\,\mu_\theta(\boldsymbol{x},\boldsymbol{y}_t,\gamma_t),\sigma_t^2\boldsymbol{I}).$$

learned isotropic Gaussian conditional distributions

covariance matrix $1-\alpha_t$

SR3 – U-Net

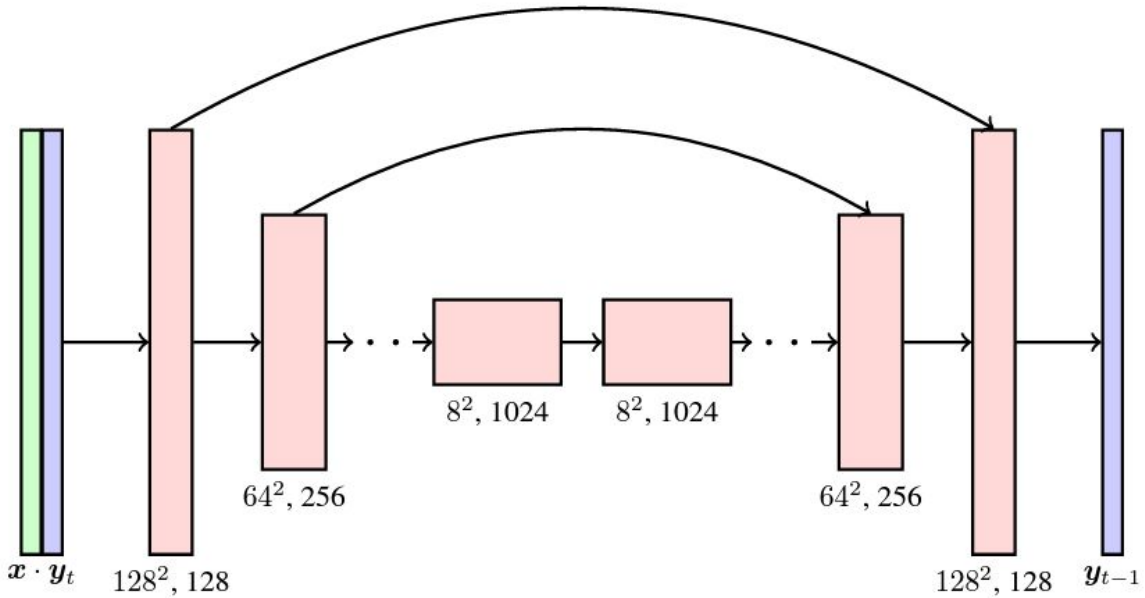


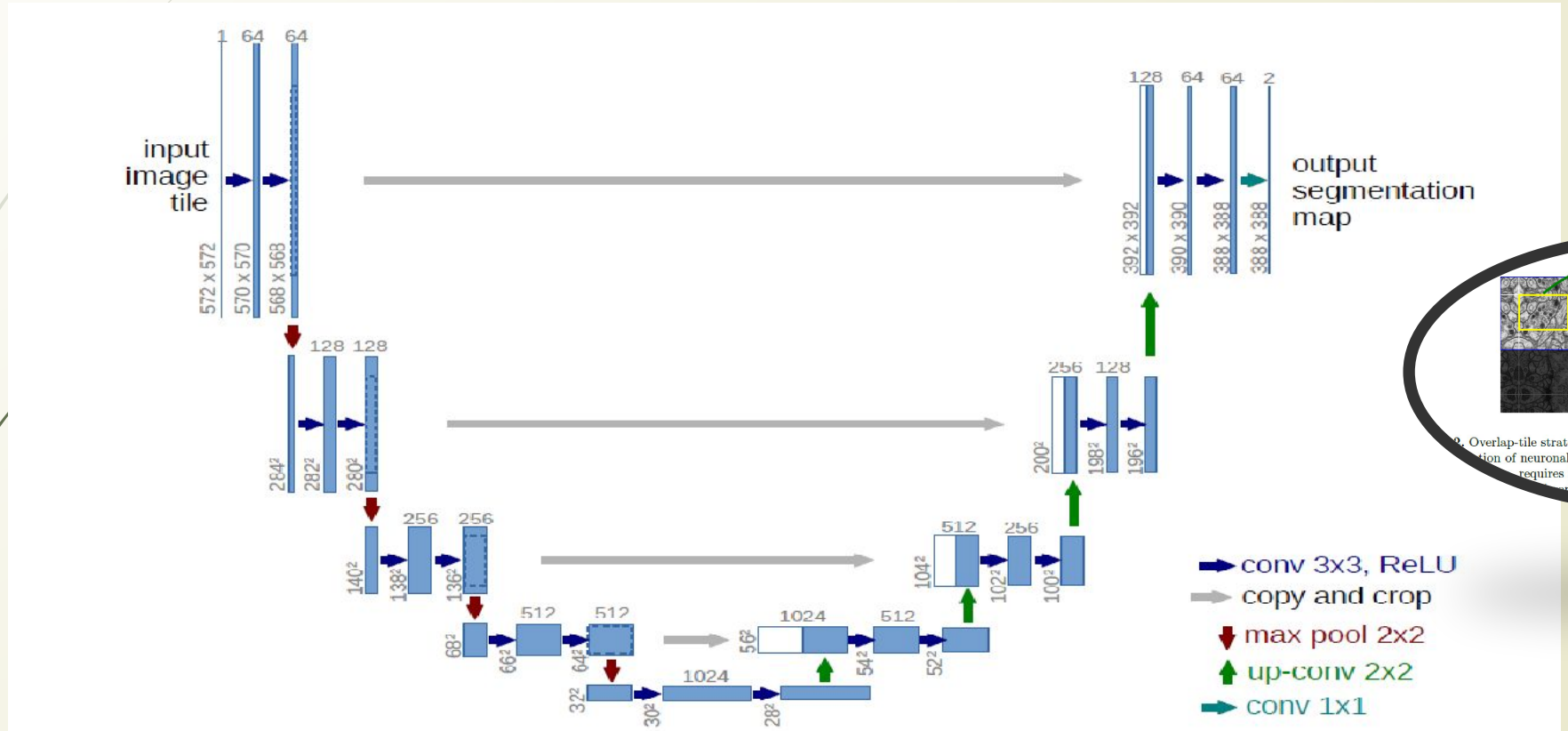
Figure A.1: Description of the U-Net architecture with skip connections. The low resolution input image $\boldsymbol{x}$ is interpolated to the target high resolution, and concatenated with the noisy high resolution image $\boldsymbol{y}_t$. We show the activation dimensions for the example task of $16\times16 \rightarrow 128\times128$ super resolution.

U-Net - 2015

U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/



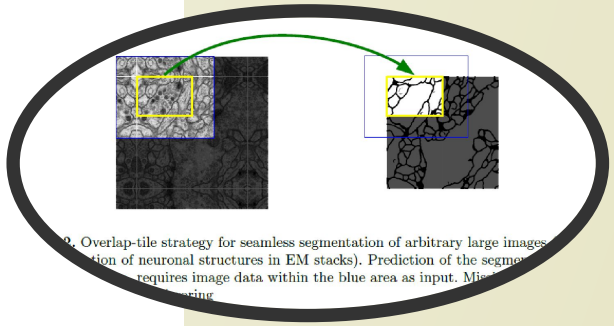


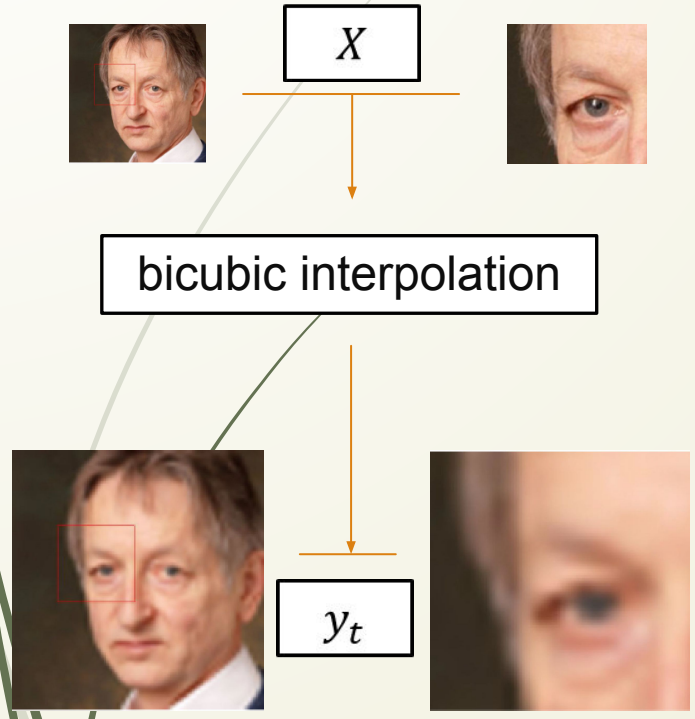
Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

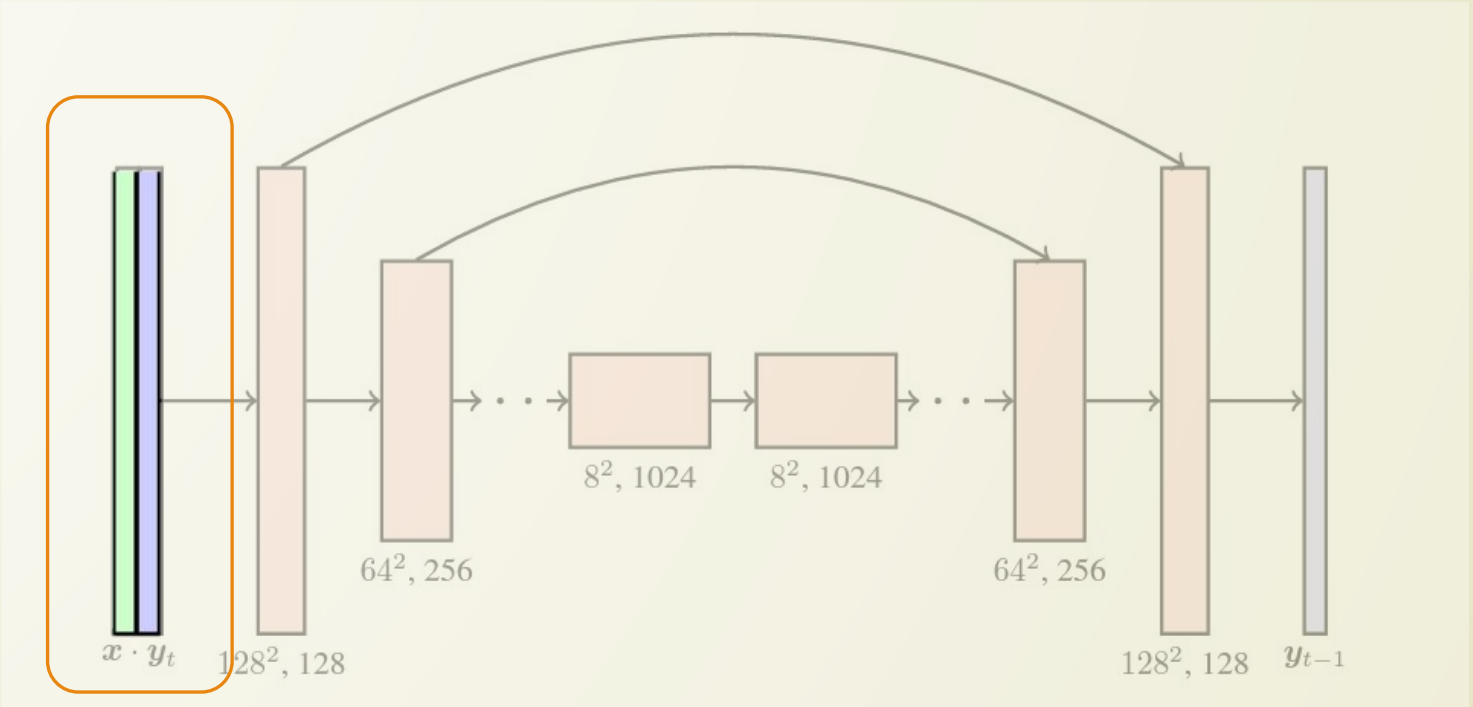
U-Net Architecture

Modifications and Features

1. **Upsampling with Many Feature Channels**:
   - Allows the network to propagate contextual information to the higher resolution layers, improving the quality and detail of the output.
2. **Symmetric Expansive Path**:
   - The expansive path (upsampling) is designed to be more or less symmetric to the contracting path (downsampling).
   - This architecture helps in effectively combining the high-resolution features from the downsampling path with the upsampled features, leading to better segmentation performance.
3. **No Fully Connected Layers**:
   - Allows the network to be applied to input images of arbitrary size.
   - Using only the valid part of each convolution ensures that the segmentation map contains only the pixels for which the full context is available in the input image. This strategy helps in avoiding border effects and ensures more accurate segmentation.
4. **Overlap-Tile Strategy**:
   - To handle arbitrarily large images, the network employs an overlap-tile strategy.
   - For predicting the pixels in the border region of the image, the network uses an extrapolation method by mirroring the input image.

SR3 – U-Net: e.g. 16X16 -> 128X128



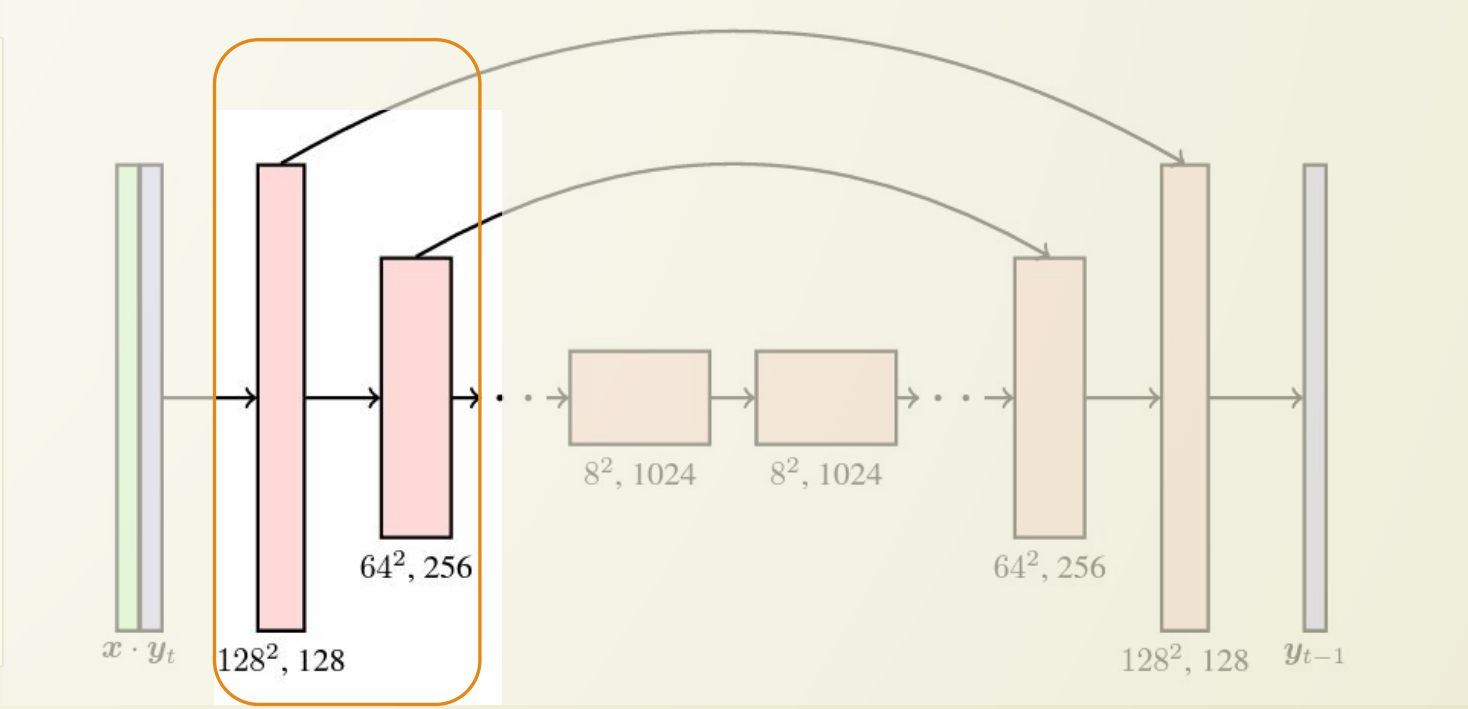


SR3 – U-Net

Contracting Path (Encoder):

The encoder path reduces the spatial dimensions of the image while increasing the depth of the feature maps. It consists of:

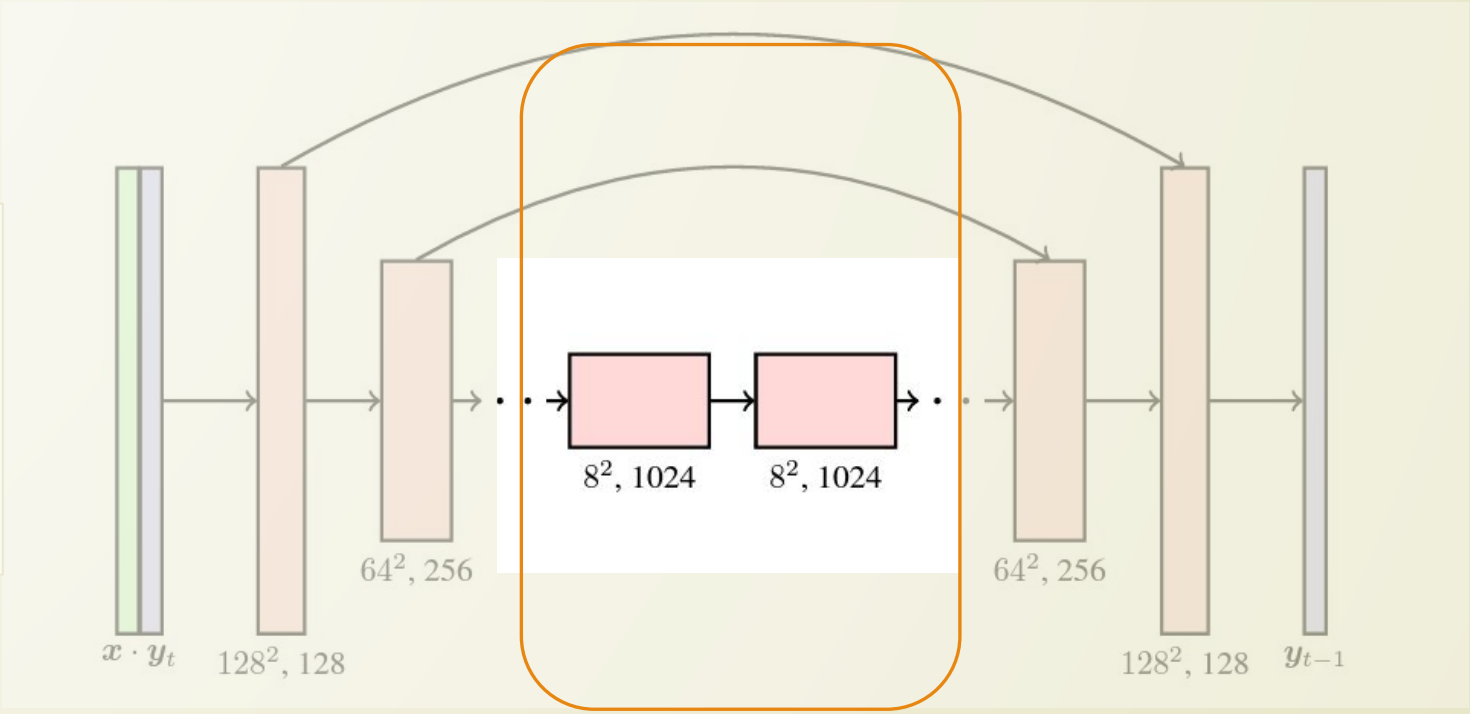
- Convolutions: Each layer applies 3x3 convolutions followed by ReLU activations.
- Pooling: 2x2 max pooling operations reduce the spatial dimensions by half.



SR3 – U-Net

Bottleneck:

At the bottom of the U, the network has the smallest spatial dimensions but the highest number of channels, capturing complex features before starting the upsampling process.

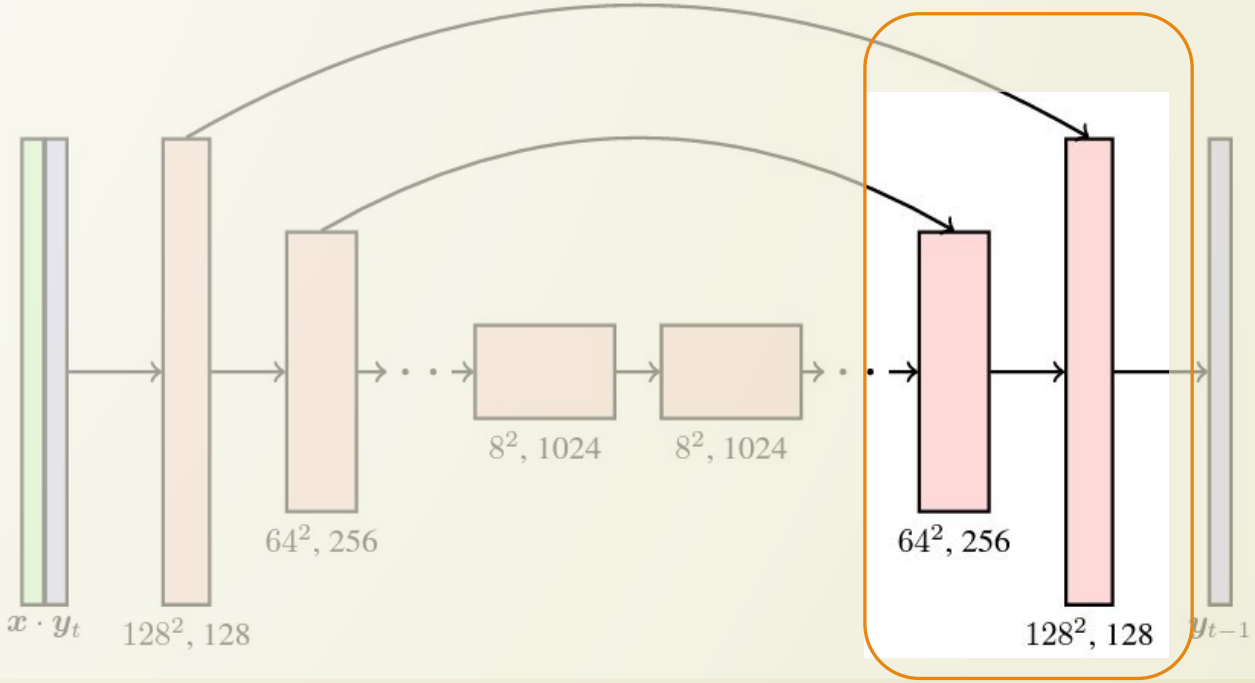


SR3 – U-Net

Expansive Path (Decoder):

The decoder path increases the spatial dimensions of the feature maps while reducing the depth.

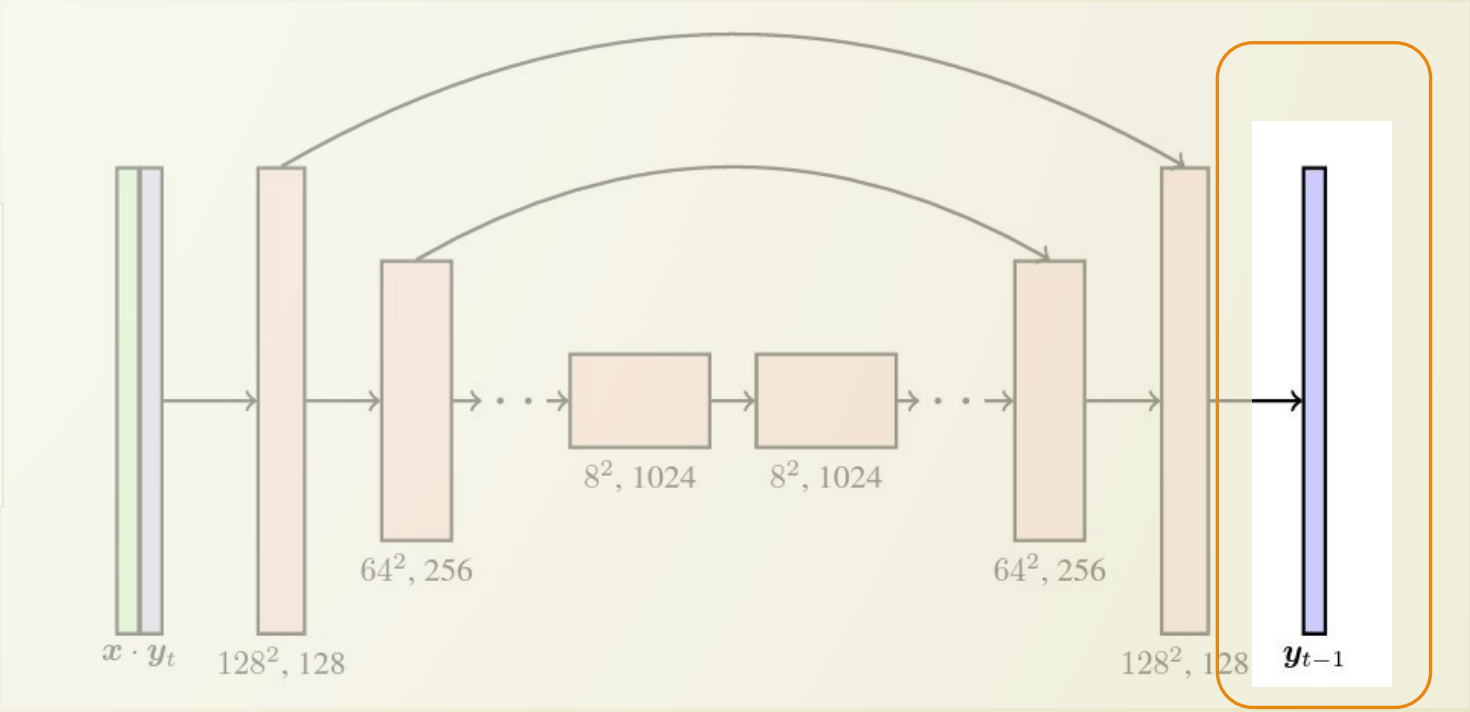
- Upsampling: 2x2 transposed convolutions double the spatial dimensions.
- Concatenation: Skip connections from the corresponding layers in the encoder are concatenated with the upsampled features. This combination helps retain high-resolution details from the input image.
- Convolutions: Additional 3x3 convolutions with ReLU activations refine the features after concatenation.



SR3 – U-Net

Final Layer:

A 1x1 convolution is applied to map the feature maps to the desired number of output channels, producing the high-resolution output image (y_{t-1}).

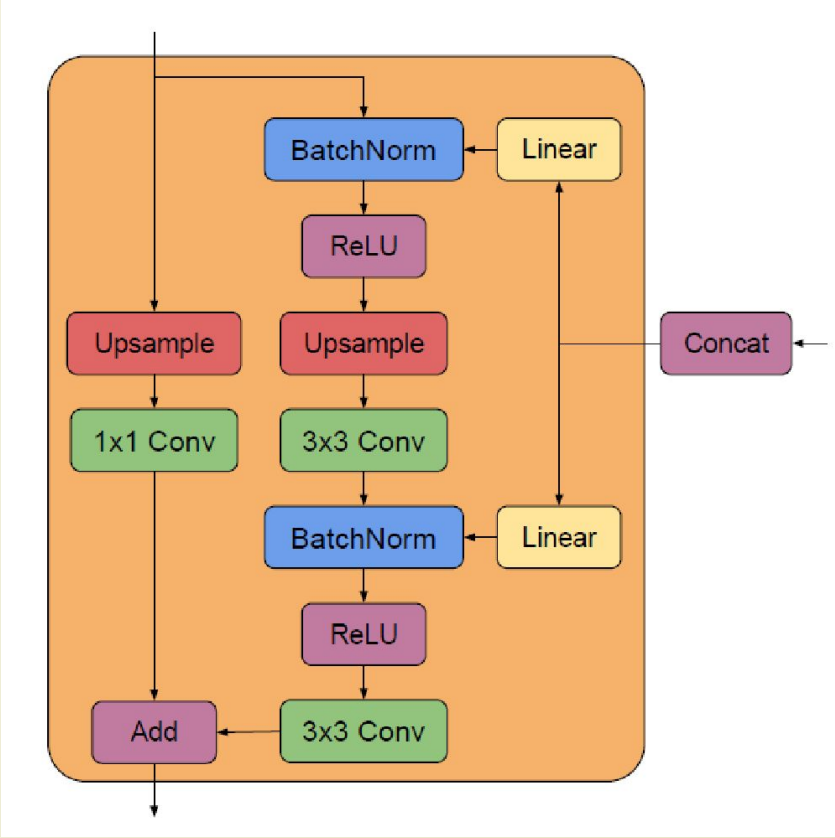


SR3 – G-Block

G-Blocks are advanced residual blocks derived from BigGAN, designed to enhance the feature extraction and refinement processes in the network.

Components:

- **Batch Normalization**: Normalizes the output of the previous layer.
- **ReLU Activation**: Applies a non-linear activation function.
- **Upsampling**: In the decoder, the G-Blocks include upsampling operations to increase the spatial resolution.
- **1x1 Convolutions**: Used for initial processing and dimensionality adjustment.
- **3x3 Convolutions**: Applied to capture spatial features.
- **Linear Layers**: Used for feature transformation.
- **Add Operation**: Combines the output of different layers, preserving residual connections.



SR3 U-Net

Task Specific U-Net Architecture Parameters

Task	Channel Dim	Depth Multipliers	# ResNet Blocks	# Parameters
$16 \times 16 \rightarrow 128 \times 128$	128	$\{1, 2, 4, 8, 8\}$	3	550M
$64 \times 64 \rightarrow 256 \times 256$	128	$\{1, 2, 4, 4, 8, 8\}$	3	625M
$64 \times 64 \rightarrow 512 \times 512$	64	$\{1, 2, 4, 8, 8, 16, 16\}$	3	625M
$256 \times 256 \rightarrow 1024 \times 1024$	16	$\{1, 2, 4, 8, 16, 32, 32, 32\}$	2	150M

Table A.1: Task specific architecture hyper-parameters for the U-Net model. Channel Dim is the dimension of the first U-Net layer, while the depth multipliers are the multipliers for subsequent resolutions.

SR3 U-Net

Modifications and Enhancements

1. **Residual Blocks from BigGAN:**
   - The architecture replaces original DDPM residual blocks with those from BigGAN. These residual blocks help in stabilizing training and improving performance by maintaining a better gradient flow through the network.
2. **Self-Attention Mechanism:**
   - Self-attention is applied to 16×16 feature maps. This mechanism helps the network capture long-range dependencies, which are crucial for high-quality super-resolution, by allowing the model to focus on different parts of the image simultaneously.
3. **Skip Connections Rescaling:**
   - Skip connections are rescaled by $\frac{1}{\sqrt{2}}$. This rescaling ensures that the contributions from the encoder and decoder paths are balanced, preventing dominance of features from either path.
4. **Channel Multipliers and Depth:**
   - The architecture uses specific channel multipliers and depth adjustments at different resolutions to optimize performance. For example, a task involving $64^2 \rightarrow 512^2$ uses depth multipliers {1, 2, 4, 8, 8, 16, 16}.

Noise Schedules

The noise schedule follows a piecewise distribution defined by:

$$p(\gamma) = \sum_{t=1}^{T} \frac{1}{T} U(\gamma_{t-1}, \gamma_t)$$

Training Process:

- During training, a time step t is uniformly sampled from the set $\{0,\ldots,T\}$.
- Subsequently, γ is sampled from the uniform distribution $U(\gamma t-1,\gamma t)$.

Parameters and Hyperparameters:

- For all experiments, the value of T is set to 2000.
- The γ_t values are uniformly spaced.

hyper-parameter search:

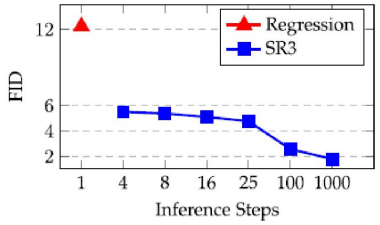


Fig. 12. FID score versus number of inference steps for 64×64 → 256×256 super-resolution. Regression (red) requires one step. All SR3 results (blue) are generated from the same denoising model but different inference noise schedules.

FID – Fréchet Inception Distance

EXPERIMENTS

Overview and Training Methodology

SR3 Model Overview

- Trained on face super-resolution using Flickr-Faces-HQ (FFHQ) and evaluated on CelebA-HQ.
- For natural images, trained on ImageNet 1K and evaluated using the development split.
- Used DDPM for training unconditional face models and class-conditional ImageNet models.
- Low-resolution images down-sampled with bicubic interpolation with anti-aliasing.
- High-resolution images created by cropping and resizing using area resampling.

Training Details

- Models trained for 1 million steps with a batch size of 256.
- Training typically takes about four days on 64 TPUv3 chips.
- Adam optimizer with linear warmup for 10,000 steps, followed by fixed learning rates (1e-4 for SR3, 1e-5 for regression models).
- Dropout rate of 0.2 used for specific models, otherwise no dropout.
- Regression models share the same architecture as SR3, providing a strong baseline for comparison.

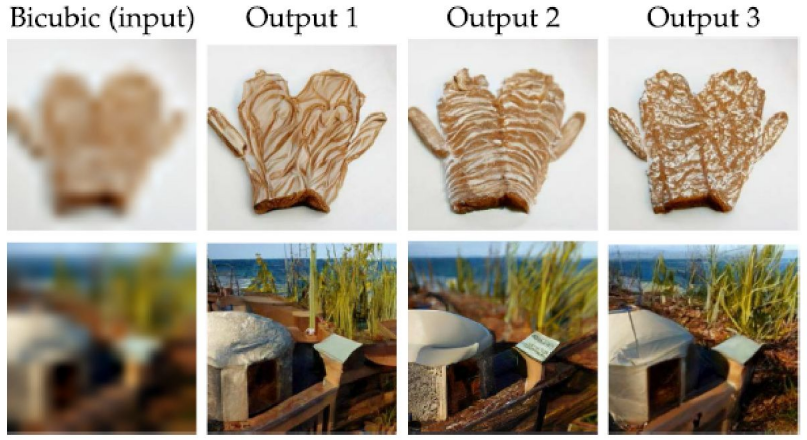


Fig. 5. Three samples from SR3 applied to ImageNet test images (16×16 → 256×256), demonstrating SR3 diversity.

Evaluation and Results

- **Evaluation Metrics**
- Face super-resolution evaluated at scales 16x16 to 128x128 and 64x64 to 512x512.
- Natural image super-resolution evaluated at scales 64x64 to 256x256 and 56x56 to 224x224.
- Unconditional 1024x1024 face generation and class-conditional 256x256 ImageNet image generation.
- **Comparative Analysis**
- Compared SR3 with EnhanceNet, ESRGAN, SRFlow, FSRGAN, and PULSE.
- Regression baseline used for direct comparison to assess iterative refinement advantages.
- Performance assessed qualitatively and quantitatively using human evaluation, FID scores, and classification accuracy of a pre-trained model.

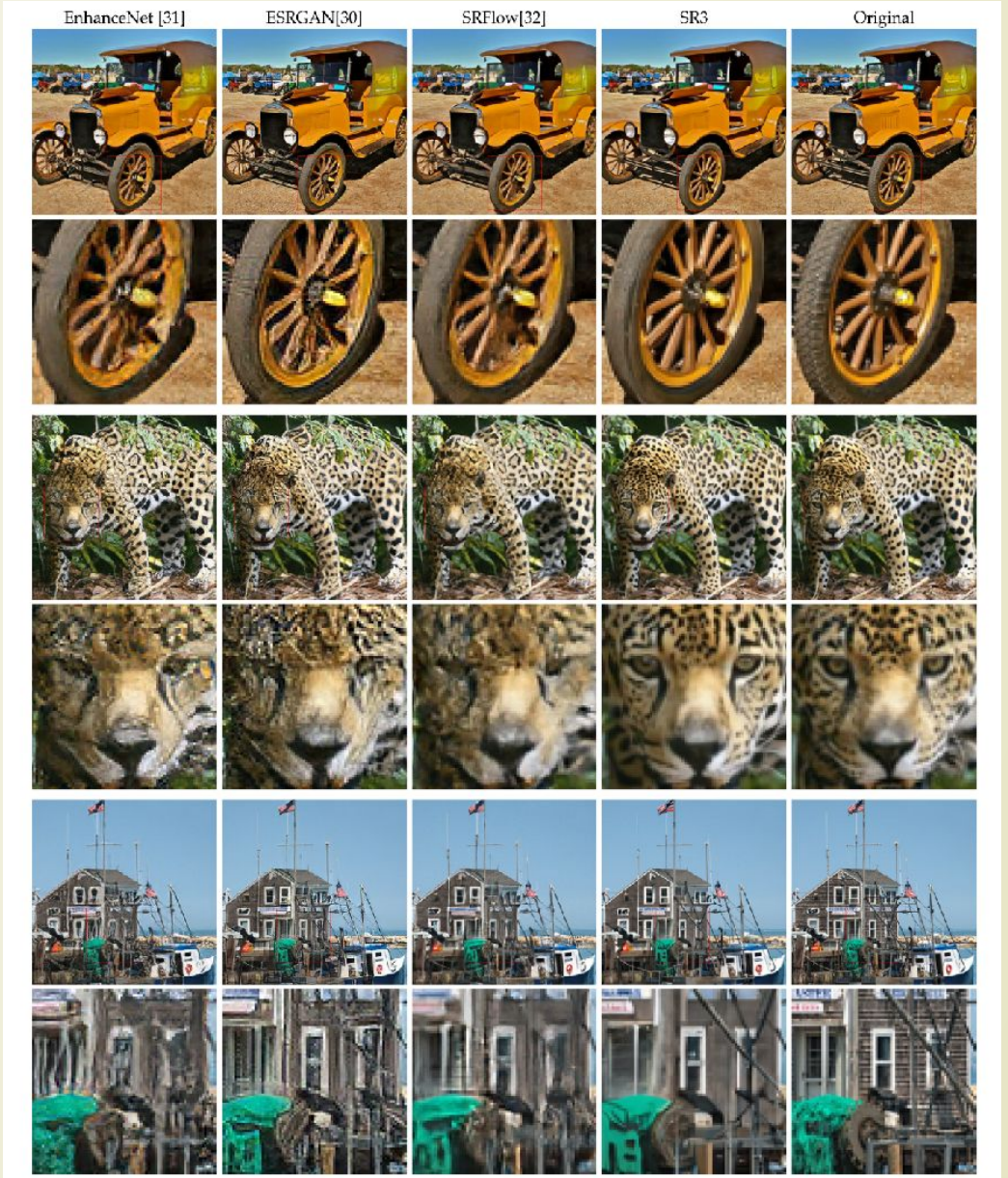


Fig. 7. SR3 and state-of-the-art methods on 4× super-resolution (64×64 → 256×256) applied to ImageNet test images. The outputs of EnhanceNet and ESRGAN are sharp, but include artifacts especially when inspecting enlarged patches. We found that ESRGAN trained on ImageNet-1 M produced similar artifacts. SR3 outputs seem to resemble the original images the most, but one can still find patches in the original images that contain more interesting texture than in SR3 outputs.

Qualitative Comparisons

SR3 vs. Regression Baseline:

- SR3 produces sharper images with fine-grained structure.
- Regression outputs are relatively blurry, especially noticeable in enlarged patches.

SR3 vs. SoTA GAN Models:

- GAN and Flow-based methods produce sharp details but also artifacts in fine-texture regions.
- SR3 produces sharp, realistic images with minimal artifacts.

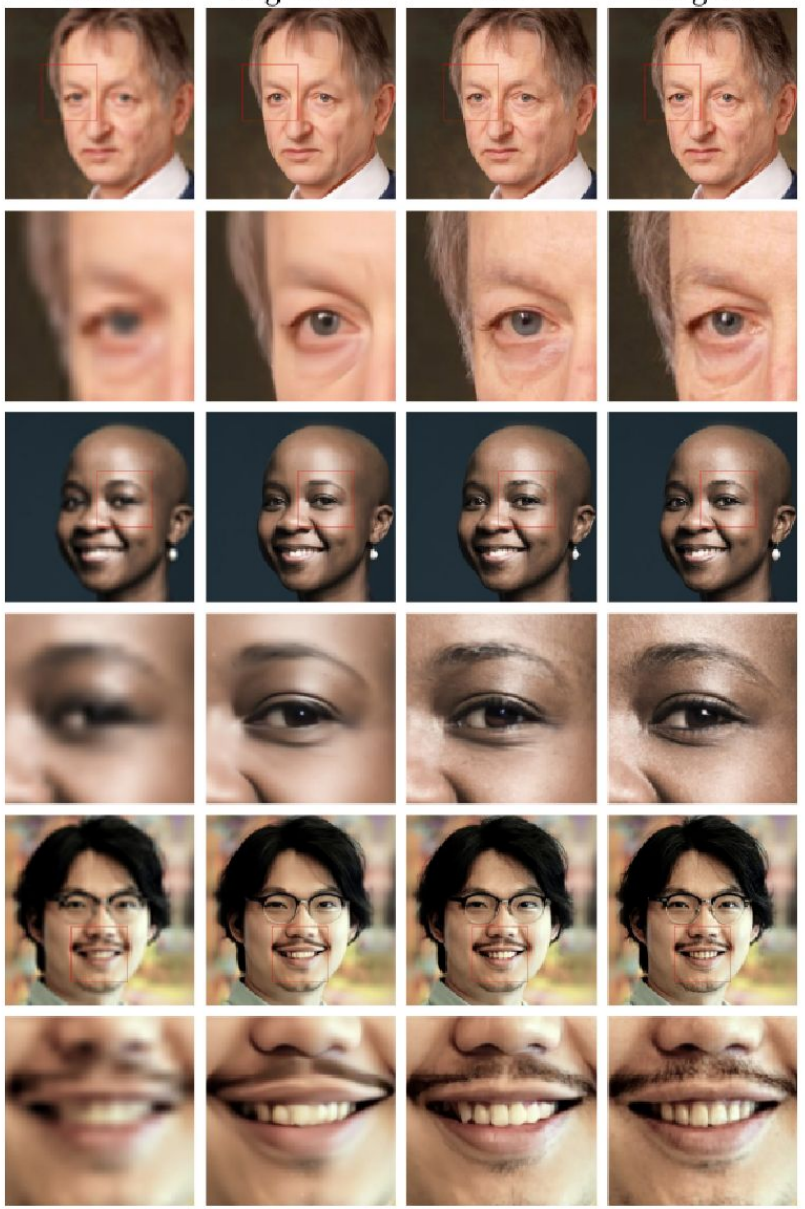


Fig. 6. Results of a SR3 model (64×64 → 512×512), trained on FFHQ, and applied to images outside of the training set.

Quantitative Evaluation

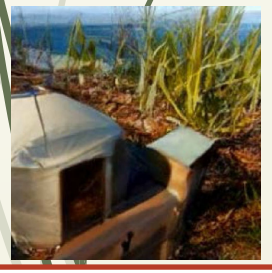
Face Super-Resolution

PSNR & SSIM on 16×16 → 128×128 Face Super-Resolution

Metric	PULSE [17]	FSRGAN [14]	Regression	SR3
PSNR ↑	16.88	23.01	**23.96**	23.04
SSIM ↑	0.44	0.62	**0.69**	0.65
Consistency ↓	161.1	33.8	2.71	**2.68**

Consistency measures MSE ($\times 10^{-5}$) between the low-resolution inputs and the down-sampled super-resolution outputs. SR3 outperforms GAN baselines in all metrics, especially improving consistency by a big margin.

- **PSNR – Peak Signal-to-Noise Ratio**
- **SSIM – Structural Similarity Index Measure**
- **Consistency – MSE (x10^{-5})**
- **Dataset – Flickr-Faces-HQ (FFHQ)**



Natural Image Super-Resolution

Performance Comparison Between SR3 and Regression Baseline on Natural Image Super-Resolution Using Standard Metrics Computed on the ImageNet Validation Set

Model	FID ↓	IS ↑	PSNR ↑	SSIM ↑
Reference	1.9	240.8	-	-
Regression	15.2	121.1	**27.9**	**0.801**
SR3	**5.2**	**180.1**	26.4	0.762

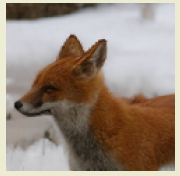
- **FID – Fréchet Inception Distance**
- **IS – Inception Score**
- **Dataset – ImageNet 1K**

Comparison of ResNet-50 Classification Accuracy on 4× Super-Resolution Outputs of the First 1 K Images From the ImageNet Validation Set

Method	Top-1 Accuracy	Top-5 Accuracy
Baseline	0.748	0.920
DRCN [50]	0.523	0.758
PsyCo [66]	0.546	0.776
ENet-E [31]	0.551	0.786
FSRCNN [67]	0.563	0.804
RCAN [64]	0.607	0.833
DRLN [68]	0.655	0.879
Regression	0.617	0.827
SR3	**0.683**	**0.880**

Note: These existing baselines have not been trained on ImageNet.

- **Top-1 Accuracy**: Measures the percentage of images for which the highest probability label predicted by the model matches the true label.
- **Top-5 Accuracy:** Measures the percentage of images for which the true label is among the top five highest probability predictions.


Human Evaluation (2AFC)

2AFC – 2-alternative forced-choice

"Which of the two images is a better high quality version of the low resolution image in the middle?"

"Which image would you guess is from a camera?"

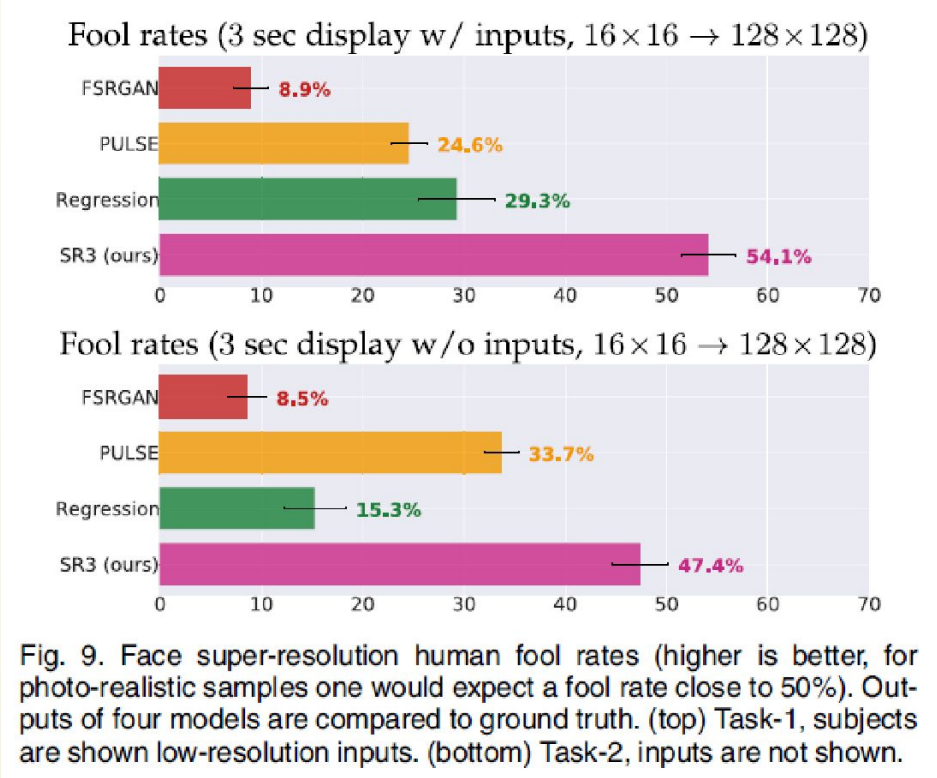


Fig. 9. Face super-resolution human fool rates (higher is better, for photo-realistic samples one would expect a fool rate close to 50%). Outputs of four models are compared to ground truth. (top) Task-1, subjects are shown low-resolution inputs. (bottom) Task-2, inputs are not shown.

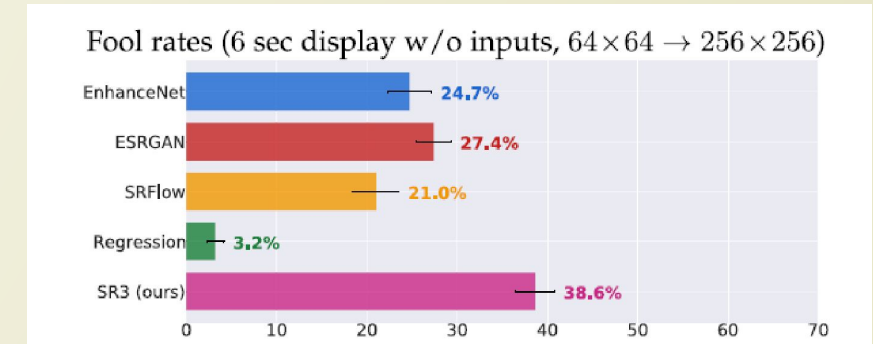


Fig. 10. ImageNet super-resolution fool rates. Model outputs are compared to ground truth with pair of images shown for 6 seconds.

Solution Design

Overview

- What is our problem?
- What is our solution?
- How our solution?
- How our solution? *Behind the scenes…*

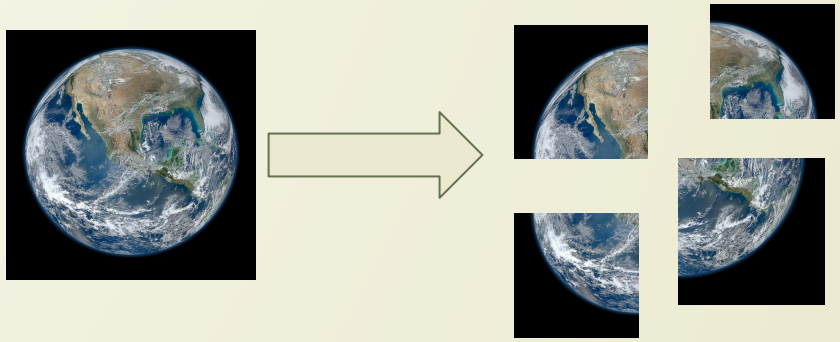
Solution Design

What is our problem?

- Flick-Faces-HQ
  - In total: 70000 images
  - Size: 2.56**TB**
  - Channels: RGB
- Flickr-Faces-HQ (small)
  - In total: 3143 images
  - Size: 4.28GB
  - Channel: RGB
- DIV2K Dataset
  - TR: 800 images
  - Validation: 100 images
  - TE: 100 images
  - Size: roughly 3 GB
  - Channels: RGB
- BSD500
  - TR: 200 images
  - TE: 100 images
  - Size: 22MB
  - Channels: RGB and Grayscale

Reshaping the world… of data

1 image is worth a 1000 images

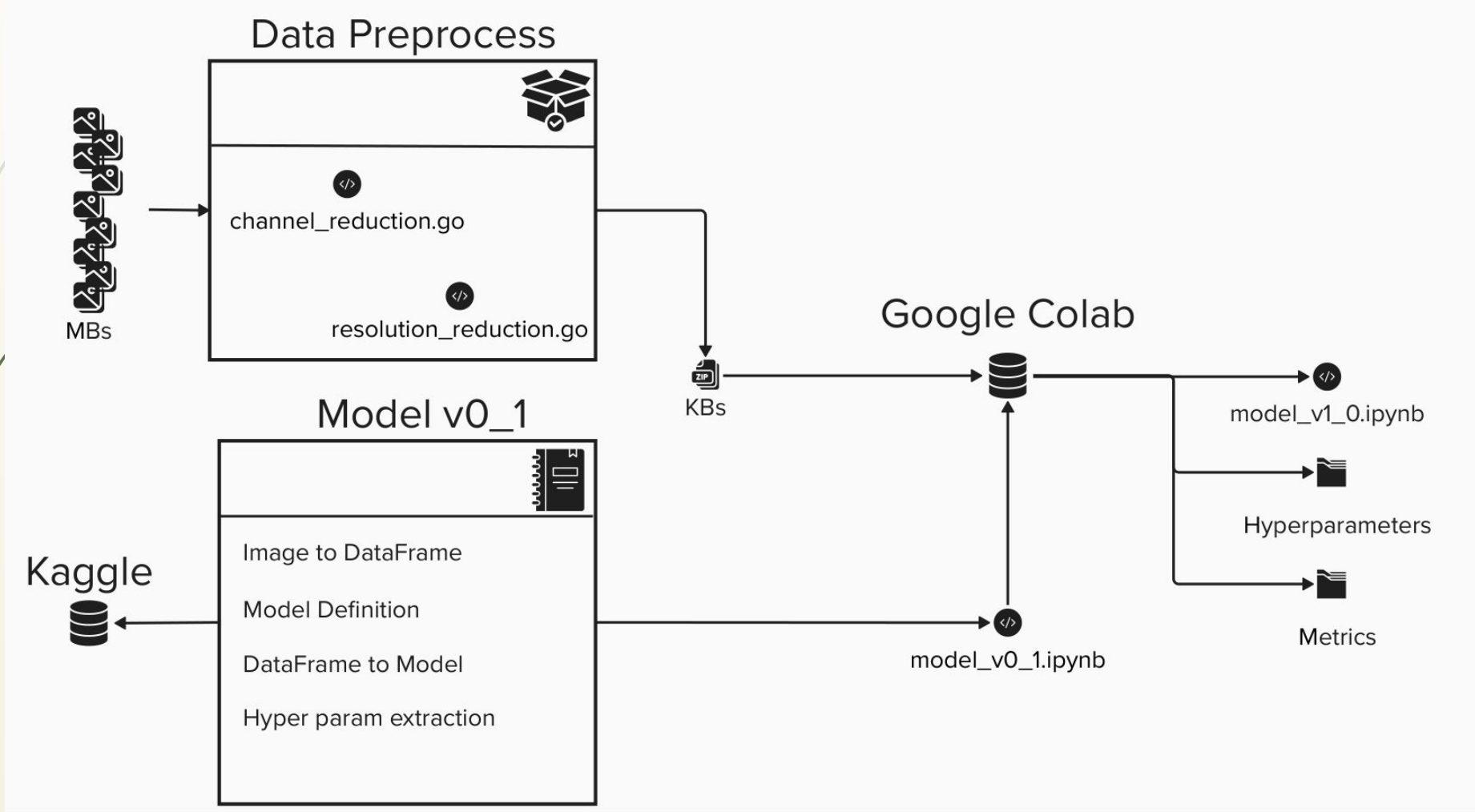


4 MB is worth a couple of KBs

- (1080/scale)x(1090/scale)
- RGB is 3 channels.
  What works for 1 channel, works for 3

Solution Design

What is our solution?



Solution Design

How is our solution?

Why .go, and .ipynb?
Python is great until parallel execution.
Go is not great, but brings parallel execution to your personal hardware.

Why Kaggle? Where's Google Colab?
Kaggle stands by free resources, as long as someone did not beat you to it.
Google Collab makes the usage more personal, through an affordable subscription.

What metrics?

- PSNR.
  Measures peak error "Actual VS Predicted", in Db. Depending on the super-resolution usage, may not be a precise metric. It depends on pixel-to-pixel comparison.
- MSE.
  Mean squared error. The fact that metric is fundamental doesn't make it obsolete. Suffers from the same drawback as PSNR.
- SSIM.
  Assess perceived image quality closer to human eye.
  It's focus is on the image structures (edges, textures, patterns).
  Given that PSNR and MSE, suffer from a similar drawback, it's beneficial to have SSIM.

Solution Design

How is our solution?

What good is hyperparameter records?
Carefully documented hyperparameter values provide an unprecedented opportunity: model reproduction.

How is reproduction different from model training?
There is no training.

What are we interested in?

- Model architecture parameters
  How many input and output channels? What's the number of filters in each layer (features)?
  What is kernel size? What about activation function? etc.
- Training hyperparameters
  - Optimizer's learning rate
  - Batch size
  - Number of epochs
  - Optimizer type
  - Loss function
    - variance noise in each iteration
    - marginalized noise over all iterations
    - etc.
- And more

The full list depends on the choice of architecture (U-Net) and the underlying tool that implements it.
An example is Pytorch package.

Solution Design

How is our solution? Behind the scenes…

U-net architecture is readily available, for example at Torch.nn module.
A couple of pieces of code divide SR3 from U-net:

- "Forward" - image downsampling path
- "Backward" - image upsampling path

The "gold nugget" here is the Loss Function.
It is constructed to minimize a difference between Gaussian and actual noise variation in-between iterations of image downsampling.
Learned hyper-parameters allow us to build a bridge between Gaussian and actual noise, so that image upsampling begins at pure Gaussian noise.

Algorithm 1 Training a denoising model f_θ

1: **repeat**
2: $(\boldsymbol{x}, \boldsymbol{y}_0) \sim p(\boldsymbol{x}, \boldsymbol{y})$
3: $\gamma \sim p(\gamma)$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take a gradient descent step on
$\nabla_\theta \left\| f_\theta(\boldsymbol{x}, \sqrt{\gamma}\boldsymbol{y}_0 + \sqrt{1-\gamma}\boldsymbol{\epsilon}, \gamma) - \boldsymbol{\epsilon} \right\|_p^p$
6: **until** converged

Algorithm 2 Inference in T iterative refinement steps

1: $\boldsymbol{y}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\boldsymbol{z} = \mathbf{0}$
4: $\boldsymbol{y}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{y}_t - \frac{1-\alpha_t}{\sqrt{1-\gamma_t}} f_\theta(\boldsymbol{x}, \boldsymbol{y}_t, \gamma_t) \right) + \sqrt{1-\alpha_t}\boldsymbol{z}$
5: **end for**
6: **return** $\boldsymbol{y}_0$

Thank you!