Lecture 3.3
Posters and
Intro to Diffusion Models

Dimitrios Papadopoulos
Associate Professor, DTU Compute

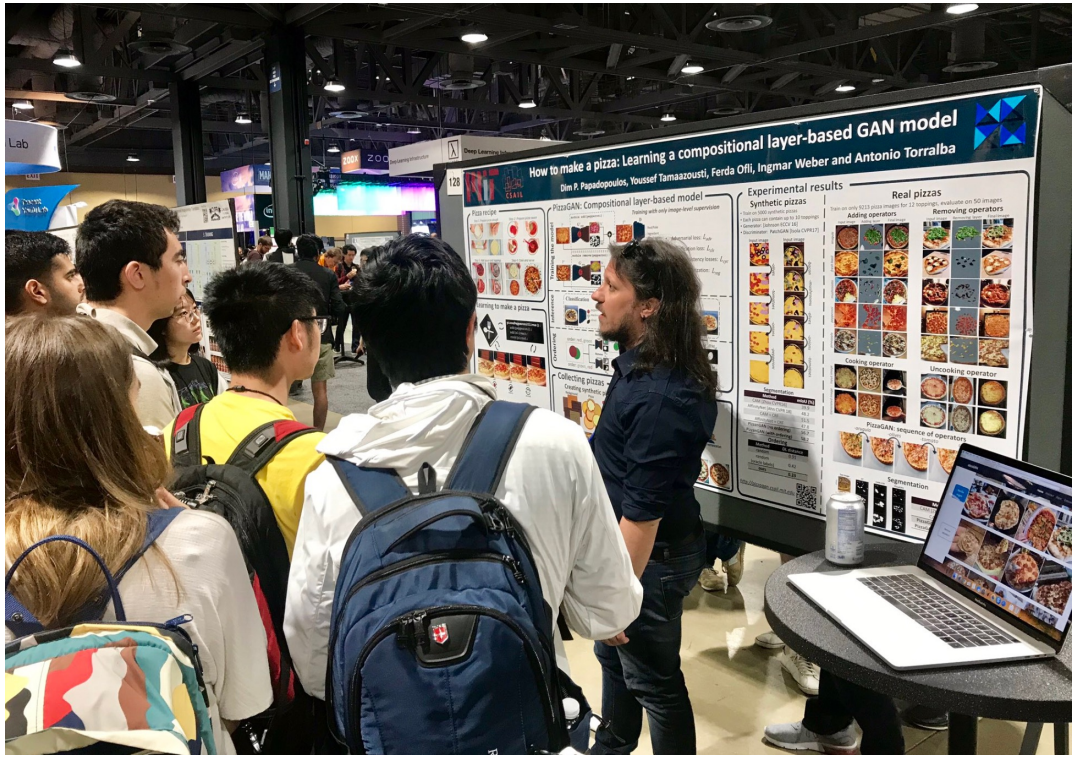
Yesterday

- Progressive GANs, StyleGAN, StyleGAN2, StyleGAN2-ADA
- Interpolation in latent space
- Reconstructing real images
- Learn and apply latent directions
- Generate images from text prompts (DALLE, DALLE2, Imagen)
- CLIP + StyleGAN
- Project 2

Today

- How to make a poster (Tips/ Feedback)
- Quick introduction to diffusion models
- Poster session and feedback to other groups

Poster sessions



School Logo here

Your title here: Maybe add some pictures and/or school logo on the left

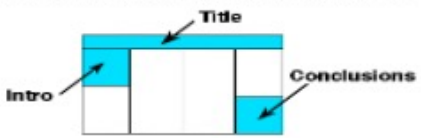
authors and affiliation (First names of authors increase interaction potential)

Section 1 (sizes):

- Posters boards are 48" tall and 96" wide, but we recommend you leave a little border since you may not be able to pin at the vertical edge. Since PowerPoint does not let one define such a large paper size, this template is designed to be printed at 200%, yielding a 46" x94" poster. You can scale it up or down a bit (e.g. 42" is a common paper size at FexEd). Note there is no direct international A0.. A1 equivalent. The poster size is approximately three A0 boards next to each other, i.e., each column in this example is about one A0 board.

- Ideally you want to keep it very readable: this is not your paper, it is a poster. 32pt here (64 final printing) is good for most text:
  - Sub-bullets are 28 here (56 final)
  - Don't use smaller than 24pt in this template (which is 48pt in final printing at 200%)
  - Insert plenty of graphics and any math you need

- When inserting graphics or equations, keep the resolution high (remember this will be printed at 200%). If you can see blocking artifacts at 400% magnification in PowerPoint, consider finding better graphics. This is an example of BAD/LOW RES GRAPHICS



- Leave enough margin for pushpin and remember many big plotters cannot get within .5" of the actual paper edge.

- You are free to use colored backgrounds and such but they generally reduce readability.

- You are free to use what ever fonts you like.
  - San Serif fonts like Arial are more readable from a distance,
  - Serif fonts like times may look more consistent with your mathematics

Section 2 (layout):

- Remember the poster session will be crowded so design the poster to be read in columns so people can read what is in front of them and move left to right to get the whole story.

- The poster should use photos, figures, and tables to tell the story of the study. For clarity, present the information in a sequence that is easy to follow.

- There is often way too much text in a poster - there definitely is in this template! Posters primarily are visual presentations; the text should support the graphics. Look critically at the layout. Some poster 'experts' suggest that if there is about 20-25% text, 40-45% graphics and 30-40% empty space, you are doing well.

Section 3:

- Include more figures than are in the paper so you can talk to them. Include things that are not in the paper and then encourage them to read the paper. Don't try to just put all the paper here.

- If it looks like a cut/paste of the paper, people skip that poster since they can read the papers after the conference. Many people find it better to spend time talking with poster presenters that have more to offer than just redoing the paper content paper in big fonts.

- People will likely have already seen your posted video, so the poster can serve as a talking point for you.

- Remember Poster boards look like this.. This is your canvas. Paint us a picture of your work.

Summary/Conclusion

- Summarize your contributions

- Summarize your results (if applicable)

- **You an add in links to additional videos, code, or project website (or QR code)**

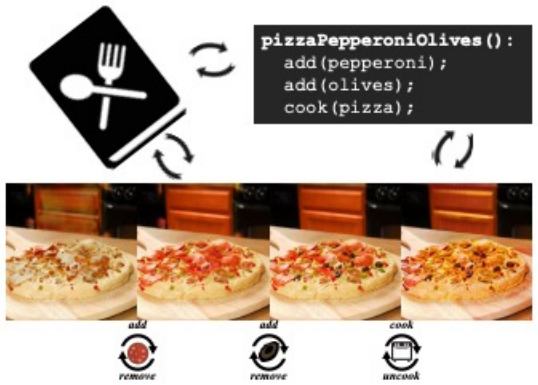
References

How to make a pizza: Learning a compositional layer-based GAN model

Dim P. Papadopoulos, Youssef Tamaazousti, Ferda Ofli, Ingmar Weber and Antonio Torralba

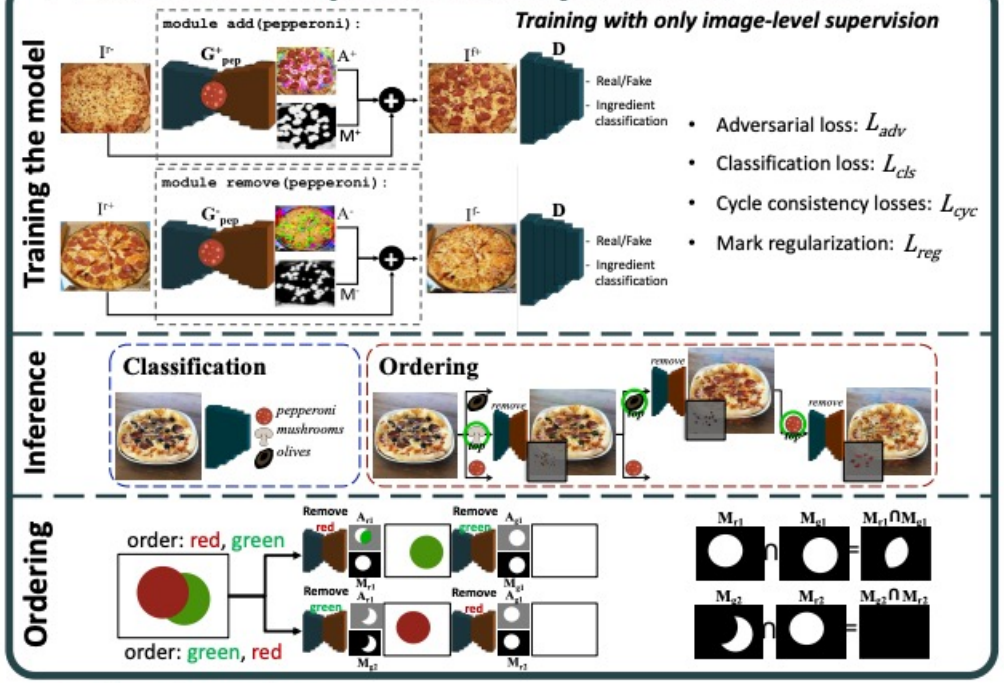
Pizza recipe

Learning to make a pizza



- Build a generative model to mirror the recipe procedure.
- Learn composable modules that carry on different cooking operations (e.g. add/remove ingredient).
- Decompose an image into an ordered sequence of layers.

PizzaGAN: Compositional layer-based model



Collecting pizzas

Creating synthetic pizzas

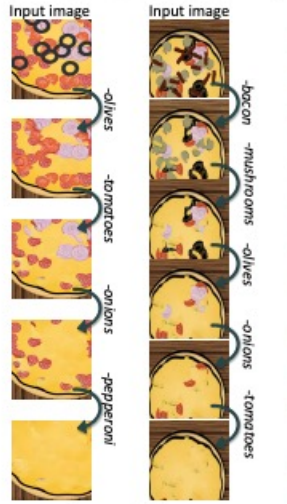
Annotating real pizzas

#pizza #pizzatime #pizzaparty #ilovepizza

Experimental results

Synthetic pizzas

- Train on 5000 synthetic pizzas
- Each pizza can contain up to 10 toppings
- Generator: [Johnson ECCV 16]
- Discriminator: PatchGAN [Isola CVPR17]



Segmentation

Method	mIoU (%)
CAM [Zhou CVPR16]	39.9
AffinityNet [Ahn CVPR 18]	48.2
CAM + CRF	51.5
AffinityNet + CRF	47.8
PizzaGAN (no ordering)	**56.7**
PizzaGAN (with ordering)	**58.2**

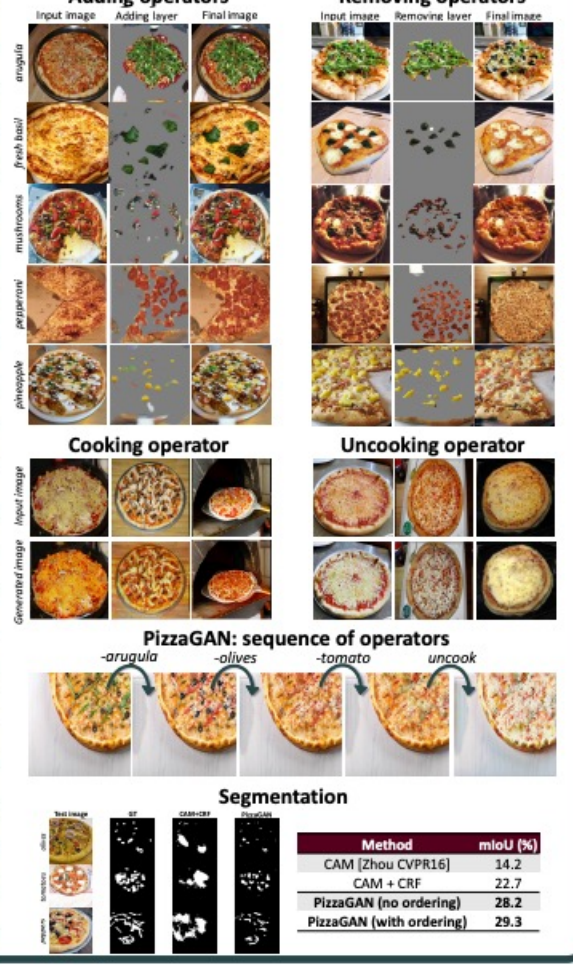
Ordering

Method	DL distance
random	0.91
random (oracle labels)	0.42
ours	**0.33**

http://pizzagan.csail.mit.edu

Real pizzas

Train on only 9213 pizza images for 12 toppings, evaluate on 50 images



Segmentation

Method	mIoU (%)
CAM [Zhou CVPR16]	14.2
CAM + CRF	22.7
PizzaGAN (no ordering)	**28.2**
PizzaGAN (with ordering)	**29.3**

Learning Program Representations for Food Images and Cooking Recipes

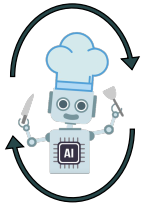
Dim P. Papadopoulos, Enrique Mora, Nadiia Chepurko, Kuan Wei Huang, Ferda Ofli, Antonio Torralba

Introduction

Food understanding

look & cook

New Orleans style Gumbo



Instructions

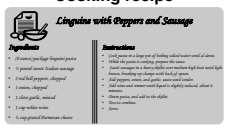
Mix the butter and the flour in a large pot and cook for 20 minutes on medium high heat. Stir in the green pepper, onion and celery and cook for 5 minutes. Add garlic and Cajun seasoning. Slowly add stock and simmer gently for 30 minutes. Stir in the sausages Add the shrimp and simmer for another 5 minutes. Add black pepper and salt to taste. Serve!

read & cook

Program representation

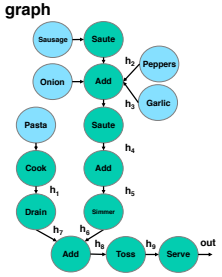
Food image

Cooking recipe



Cooking program and graph

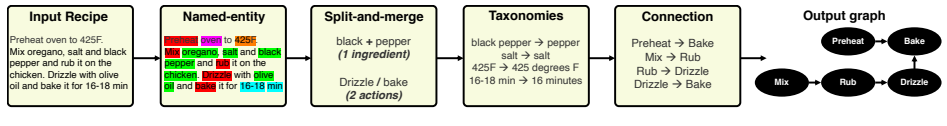
```
h1 = Cook(pasta, tool=pot, time=until al dente)
h2 = Saute(sausage, tool=skillet, temp=medium
heat, time=until light brown, how=breaking
clumps)
h3 = Add(h2, peppers, onions, garlic)
h4 = Saute(h3, time=until tender)
h5 = Add(h4, wine)
h6 = Simmer(h5, time=6 minutes)
h7 = Drain(h1)
h8 = Add(h6, h7)
h9 = Toss(h8, why=to combine)
out = Serve(h9)
return out
```



Cooking Programs:

- ✓ provide a non-ambiguous and structured presentation
- ✓ capture cooking semantics and action relationships
- ✓ can be easily manipulated by users
- ✓ can be potentially executed by agents

Cooking programs



Valid program sequences

Preheat()	Mix()	Mix()	Mix()
Mix()	Preheat()	Rub()	Rub()
Rub()	Rub()	Preheat()	Drizzle()
Drizzle()	Drizzle()	Drizzle()	Preheat()
Bake()	Bake()	Bake()	Bake()

Invalid examples

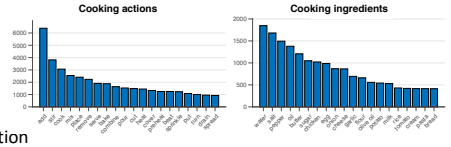
Preheat()	Mix()
Rub()	Rub()
Mix()	Drizzle()
Drizzle()	Bake()
Bake()	Preheat()

Final program

```
h1= Preheat(tool=oven, temp=425 degrees F)
h2= Mix(oregano, salt, pepper)
h3= Rub(h2, chicken)
h4= Drizzle(h3, olive oil)
out= Bake(h1,h4, time=16 minutes)
```

Program collection

- **3,708 programs** from Recipe1M dataset [Salvador CVPR17]
- **42,473 annotated sentences** with **478,285 tagged words**
- **Total annotation time:** 4 minutes per recipe
- **Human agreement:** 97.9% Named-entity and 92.7% Connection

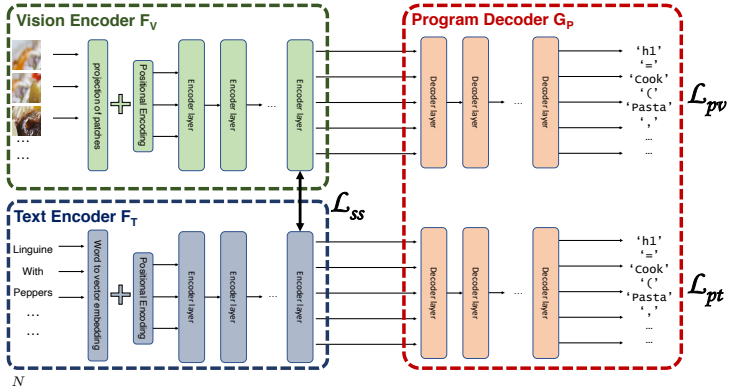


From food images and recipes to programs

Image

Text recipe
(Title, Ingredients, Instructions)

Linguine with Peppers and Sausage
(8 ounce) package linguini pasta
½ pound sweet Italian sausage
...
...
¼ cup grated Parmesan cheese
Cook pasta in a large pot.
Saute sausages in a heavy skillet over medium high heat.
...
...
Serve.



$$\mathcal{L}_{ss} = \frac{1}{N}\sum_{i=1,j\neq i}^{N} max(0, s(i,j) - s(i,i) + m) + max(0, s(j,i) - s(i,i) + m)$$

$$\mathcal{L}_{pv} = \frac{1}{N}\sum_{i=1}^{N} \min_{j\in[1,\beta]} \mathcal{L}_{ce}(\Pi_i^I, P_i^j)$$

Experiments

- **Recipe1M** [Salvador CVPR17], 340k recipes with images
- **Vision encoder**: ViT-B/16 [Dosovitskiy ICLR21]
- **Text encoder** and **Program decoder**: Transformer [Vaswani NeurIPS17]
- **50 epochs**
- **Adam** optimizer
- **0.1 step decay** every 20 epochs

Image-to-recipe retrieval

Query Image | **Retrieved Recipe**

Recipe Component	Program Loss	Encoder Layers	image-to-recipe medR	R@1	R@5	R@10	recipe-to-image medR	R@1	R@5	R@10
Title		8 (small)	4.6	24.0	54.4	67.8	4.5	23.8	54.2	67.5
Ingredients		8 (small)	2.8	39.1	68.1	80.3	3.0	38.9	68.4	80.5
Instructions		8 (small)	3.1	36.6	65.2	76.8	3.0	36.5	65.8	76.9
Title+Ingr		8 (small)	2.0	43.6	75.6	85.0	2.0	44.9	76.2	85.4
Title+Ingr+Inst		8 (small)	1.0	53.5	81.8	89.2	1.0	53.1	82.0	89.6
Title+Ingr+Inst	✓	8 (small)	1.0	58.6	85.7	91.7	1.0	58.2	85.5	92.0
Title+Ingr+Inst	✓	12 (base)	**1.0**	**66.9**	**90.9**	**95.1**	**1.0**	**66.8**	**89.8**	**94.6**

State-of-the-art cross-modal retrieval	Img Encoder	image-to-recipe medR	R@1	R@5	R@10	recipe-to-image medR	R@1	R@5	R@10
Salvador CVPR 17 [49]	ResNet-50	5.2	24.0	51.0	65.0	5.1	25.0	52.0	65.0
Chen ACM MM 18 [8]	ResNet-50	4.6	25.6	53.7	66.9	4.6	25.7	53.9	67.1
Carvalho SIGIR 18 [5]	ResNet-50	2.0	39.8	69.0	77.4	1.0	40.2	68.1	78.7
Zhu CVPR 19 [67]	ResNet-50	2.0	39.1	71.0	81.7	2.0	40.6	72.6	83.3
Fu CVPR 20 [14]	ResNet-50	2.0	48.2	75.8	83.6	1.9	48.4	76.1	83.7
Pham AAAI 21 [42]	ResNet-50	1.6	49.7	79.3	86.3	1.6	50.1	79.0	86.4
Wang CVPR 19 [57]	ResNet-50	1.0	51.8	80.2	87.5	1.0	52.8	80.2	87.6
Wang TMM 21 [58]	ResNet-50	1.0	54.0	81.7	88.8	1.0	54.9	81.9	89.0
Fain arXiv 19 [13]	ResNeXt-101	1.0	60.2	84.0	89.7	–	–	–	–
Salvador CVPR 21 [48]	ResNet-50	1.0	60.0	87.6	92.9	1.0	60.3	87.6	93.2
Salvador CVPR 21 [48]	ViT-B/16	1.0	63.2	88.3	93.1	–	–	–	–
Ours	**ViT-B/16**	**1.0**	**66.9**	**90.9**	**95.1**	**1.0**	**66.8**	**89.8**	**94.6**

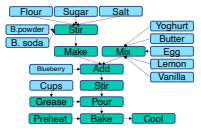
Program prediction

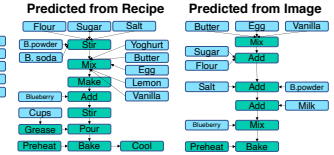
Image | **Recipe** | **GT** | **Predicted from Recipe** | **Predicted from Image**

Title: lemon blueberry muffins
Ingredients: 2 cups flour, 2/3 cup sugar, 1 teaspoon baking powder, 1 teaspoon baking soda, 1/2 teaspoon salt, ...
Instructions: Preheat oven to 400f (200c). grease muffin cups. stir together flour, 2/3 cup sugar, baking powder, baking soda and salt. separately mix yoghurt, butter, egg, lemon zest and vanilla extract until blended. make a well in the centre of the dry ingredients, add yoghurt mixture and blueberries and stir to combine. pour into muffin cups. bake 20-25 minutes. cool for 5 minutes before eating.

GT

olive oil	tuna	chocolate	sugar
squash	celery	milk	milk
radish	mayonnaise	sugar	salt
salt	cheddar cheese	cream cheese	vanilla extract
pepper	tortilla	whipped topping	egg
pasta	lettuce	graham cracker	
parmesan cheese			

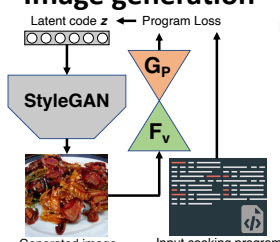


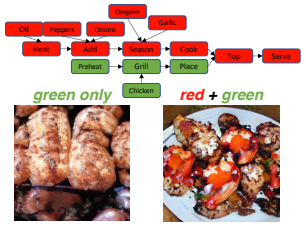


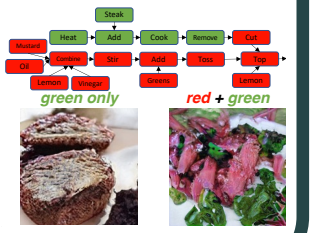
Input: Cooking recipes	Ingredients (F1 ↑)	Actions (F1 ↑)	Tools (F1 ↑)	Full graph* (GED ↓)
Random recipe	12.4	14.5	14.2	101.5
Retrieved recipe	43.4	55.2	74.2	67.1
NN (oracle)	53.5	66.5	81.1	57.2
Instructions	41.6	49.3	66.6	–
Programs (CE)	75.4	83.1	83.8	19.1
Programs (minCE)	75.5	83.1	84.1	16.8

Input: Food images	Ingredients (F1 ↑)	Actions (F1 ↑)	Tools (F1 ↑)	Full graph* (GED ↓)
Random image	12.6	14.6	14.2	102.1
Retrieved image	39.4	51.6	66.9	79.1
NN (oracle)	53.5	66.5	81.1	62.2
Instructions	28.5	38.3	50.5	–
Programs (CE)	52.8	64.5	78.1	72.1
Programs (minCE)	53.5	64.7	78.1	67.2

Image generation





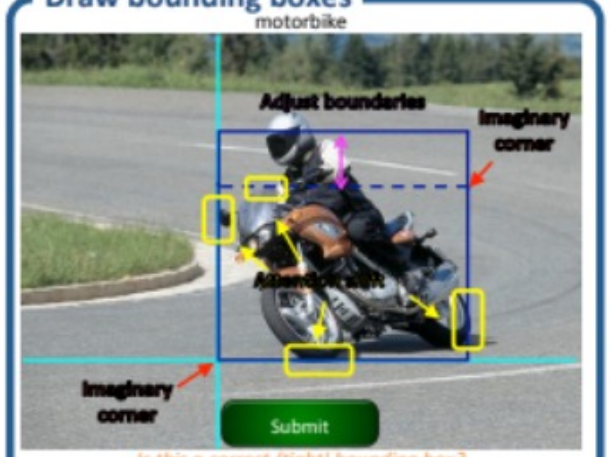


http://cookingprograms.csail.mit.edu/

Extreme clicking for efficient object annotation

Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller and Vittorio Ferrari

Draw bounding boxes

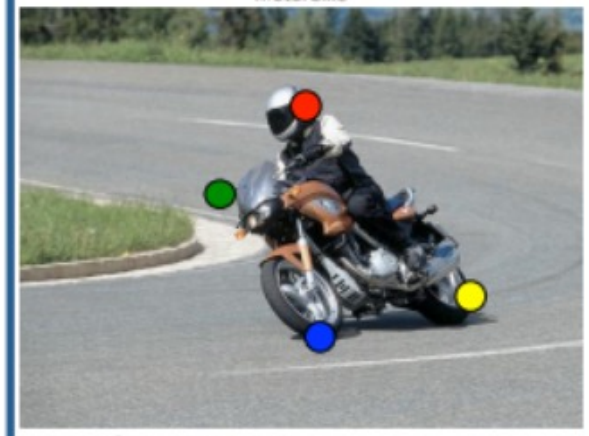


Is this a correct (tight) bounding box?

35 s for 1 bounding box!

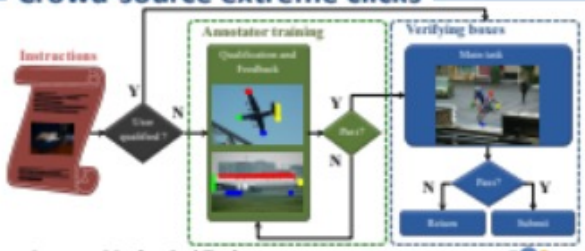
(optimistic estimation of official ILSVRC annotation protocol [Su12AAAI])

Extreme clicking



- Click on four extreme points:
  left-most, right-most, top, bottom
- Attention is on single point at a time
- Physical points on object, not imaginary
- No rectangle is involved, neither real nor imaginary
- Single task, no task-switching

Crowd-source extreme clicks

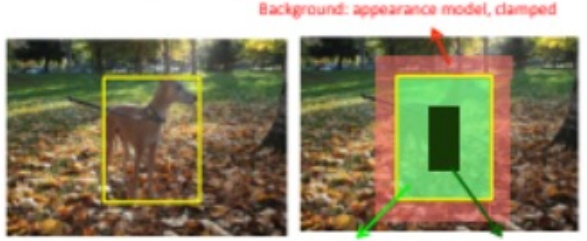


- **Amazon Mechanical Turk**
  - PASCAL VOC 2007 and 2012 (one object per class per image)
- **Annotation time:** 7s per object

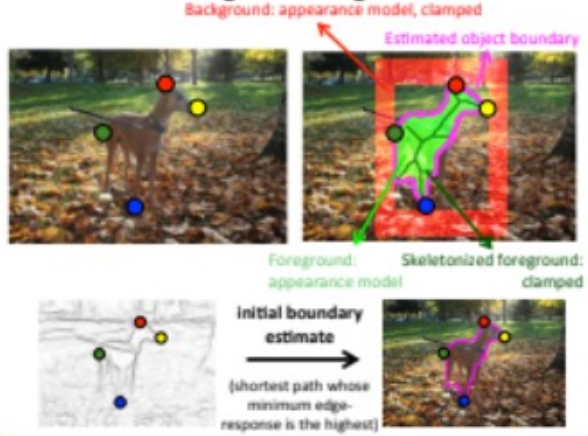
Create object segmentation

Boxes to segments using GrabCut

$$E(L) = \sum_p U(l_p) + \sum_{p,q} V(l_p, l_q)$$ [Rother SIGGRAPH 04]



Extreme clicks to segments using GrabCut



Extreme clicking results

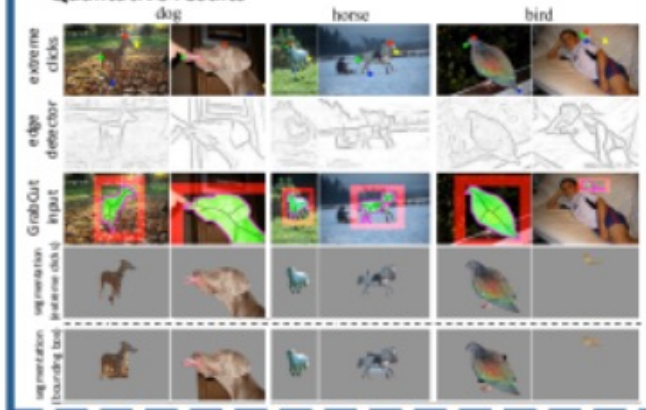
- PASCAL VOC 2007 and 2012 (1 instance per class per image)
- Quality drawing by using expert agreement: boxes vs boxed segment
- **Detector:** Fast-RCNN [Girschick ICCV 15]

		Annotation quality	Detection performance (mAP)		Annotation time
Dataset	Approach	mIoU	AlexNet	VGG16	Instance (s)
VOC 2007	Extreme clicks	88	56	66	7.0
	PASCAL GT boxes	88	56	66	34.5
VOC 2012	Extreme clicks	87	52	62	7.2
	PASCAL GT boxes	87	52	62	34.5

5 times faster without any compromise on quality

Results on object segmentation

Qualitative results



Quantitative results

Approach		mIoU VOC 2007	mIoU VOC 2012
	original GrabCut	37.3	35.8
	optimized GrabCut	74.4	71.0
	GrabCut and extreme clicks	**78.1**	**72.7**

extreme clicks into GrabCut improve object segmentation by 2-4%

[Hariharan ICCV 2011]

DeepLab on PASCAL VOC 2012

	Full supervision	Segments from GT Boxes	Segments from extreme clicks
mIoU	59.9	55.8	58.4

extreme clicks strong for weakly supervised semantic segmentation

Training Object Class Detectors from Eye Tracking Data

Dim P. Papadopoulos, Alasdair D. F. Clarke, Frank Keller and Vittorio Ferrari

THE UNIVERSITY of EDINBURGH informatics

VALVIN erc

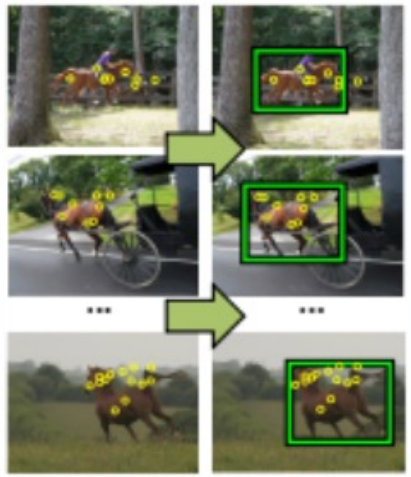
Introduction

Training object detectors

Current way: draw bounding-boxes

- time consuming (26s-42s per box) [Su AAAI 2012]
- need detailed annotation guidelines

New way: bounding-boxes from eye-tracking data



+ annotation time (1s per image)
+ reduce annotation time by 6.8x
+ simple annotation guidelines
+ correct localizations in half images

Eye tracking dataset

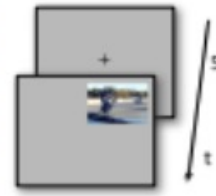
Data

- Large scale (**6270 images**)
- Pascal VOC 12 : *train+val* images of 10 classes
- 5 distinct viewers (28 in total) for each image

- 178,000 fixations (5.7 per viewer per image)
- **Mean response time** = 889 ms/image
- **Fixations on target objects** = 75.2%

Experiment

- **Visual search paradigm**
  - more fixations on target objects
  - faster than free-viewing
- **Pairs of classes**
  - two-alternative forced choice object discrimination
  - pair classes with similar background (e.g. cat, dog)



- Add random offset (central bias)
- Random image order

From fixations to bounding-boxes

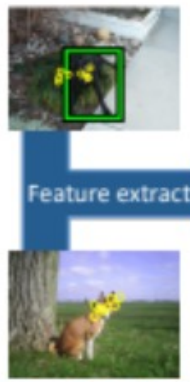
Bounding-box estimation as figure-ground superpixel labeling

$\mathcal{R}_{bb+fix}$

- small subset (7%)
- learn to predict bounding-boxes from fixations

$\mathcal{R}_{fix}$

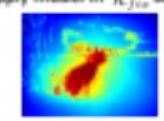
- derive new bounding-boxes from fixations



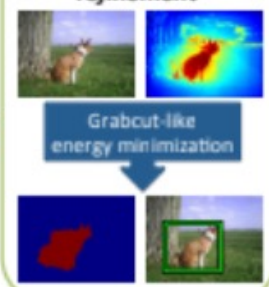
Feature extraction

Initial object segmentation

- Train a superpixel classifier in $\mathcal{R}_{bb+fix}$ (linear SVM + Platt scaling)
- Apply model in $\mathcal{R}_{fix}$ set



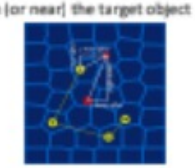
Segmentation refinement



Features

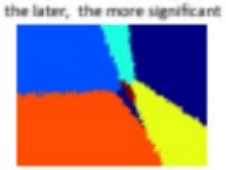
Fixation position

Visual search increases fixations on (or near) the target object



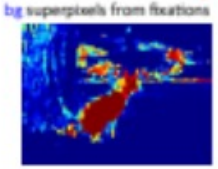
Fixation timing

Timing matters: the longer or the later, the more significant



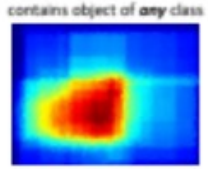
Fixation appearance

Learn color distribution of fg and bg superpixels from fixations

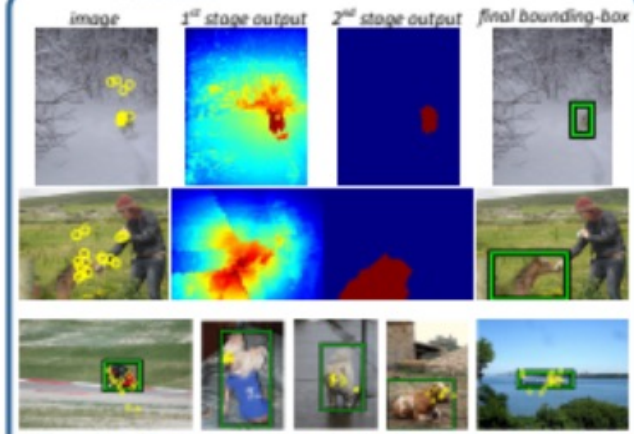


Objectness

Probability that a window contains object of ***any*** class

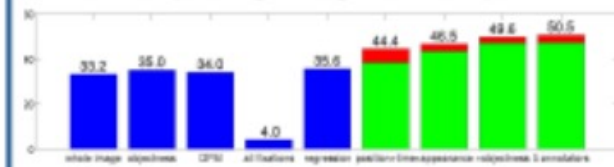


Results



Quantitative results

- 10 Pascal VOC 12 classes, 6270 images in trainval
- Evaluate predicted bounding-boxes in $\mathcal{R}_{fix}$
- Performance: percentage of images with correct predictions



+ all feature types contribute
+ full model outperforms all baselines
+ segmentation refinement always helps (+ 3-5%)

Train DPM detector from fixations [Felzenszwalb PAMI10]

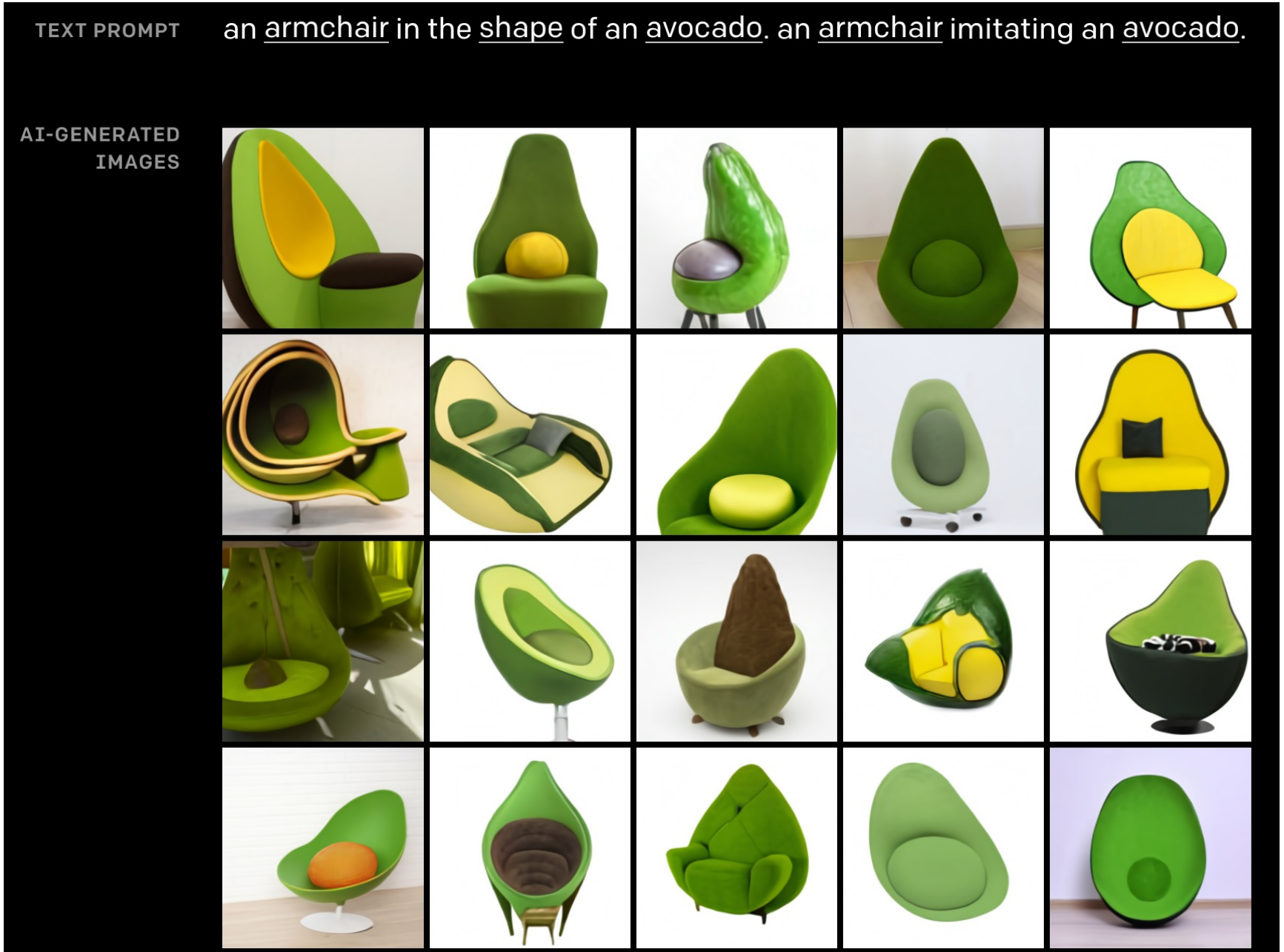
- Pascal VOC 12: train on trainval, test on test set (10991 images)

mAP = 25.5% mAP = 12.5% mAP = 13.7%

Given same annotation time, get comparable mAP performance

A quick introduction to diffusion models

DALLE



https://openai.com/blog/dall-e/

DALLE

TEXT PROMPT

a soap dispenser in the style of a pikachu. a soap dispenser imitating a pikachu.

AI-GENERATED IMAGES

https://openai.com/blog/dall-e/

DALLE 2

TEXT DESCRIPTION

An astronaut Teddy bears A bowl of soup

mixing sparkling chemicals as mad scientists shopping for groceries working on new AI research

as kids' crayon art on the moon in the 1980s underwater with 1990s technology

→

DALL·E 2

https://openai.com/dall-e-2/

Imagen by Google AI

A photo of a Corgi dog riding a bike in Times Square. It is wearing sunglasses and a beach hat.

https://imagen.research.google/

Make-A-Video by Meta

September 2022

https://makeavideo.studio/

Dreambooth

Input images

in the Acropolis

swimming

sleeping

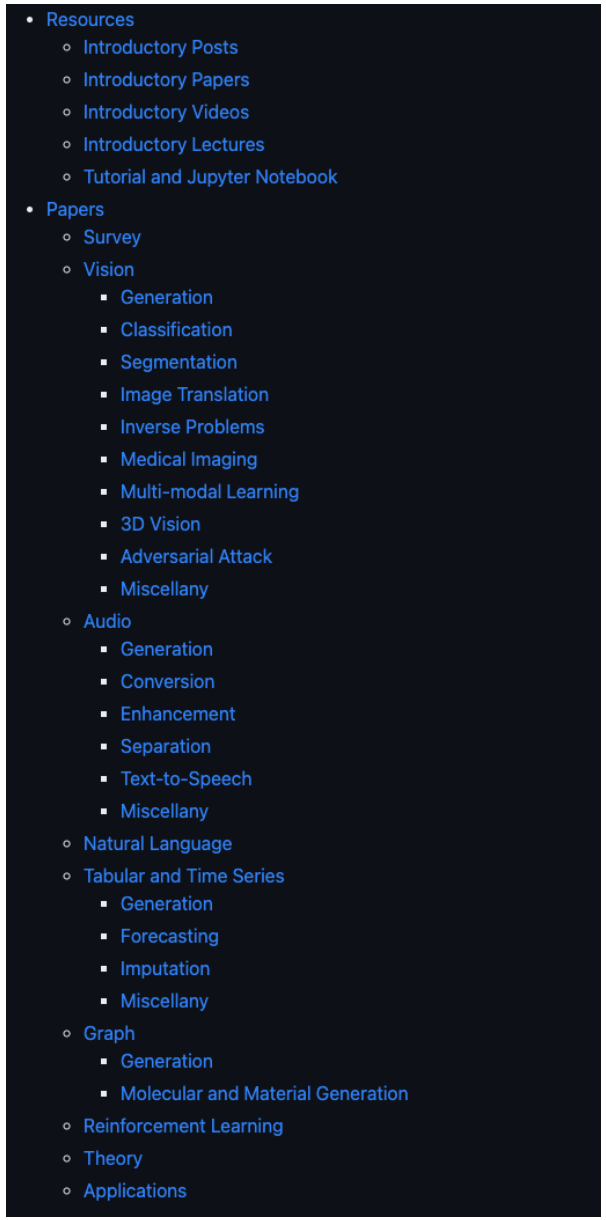
in a doghouse

in a bucket

getting a haircut

Dreamfusion

More diffusion models

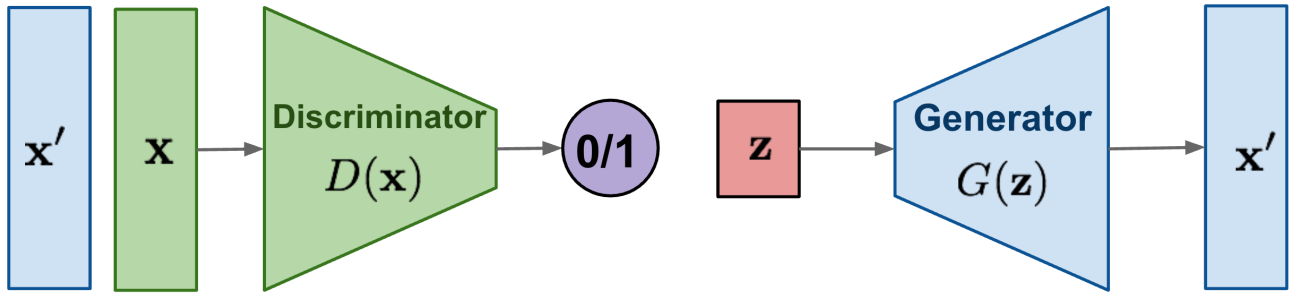


- Resources
  - Introductory Posts
  - Introductory Papers
  - Introductory Videos
  - Introductory Lectures
  - Tutorial and Jupyter Notebook
- Papers
  - Survey
  - Vision
    - Generation
    - Classification
    - Segmentation
    - Image Translation
    - Inverse Problems
    - Medical Imaging
    - Multi-modal Learning
    - 3D Vision
    - Adversarial Attack
    - Miscellany
  - Audio
    - Generation
    - Conversion
    - Enhancement
    - Separation
    - Text-to-Speech
    - Miscellany
  - Natural Language
  - Tabular and Time Series
    - Generation
    - Forecasting
    - Imputation
    - Miscellany
  - Graph
    - Generation
    - Molecular and Material Generation
  - Reinforcement Learning
  - Theory
  - Applications

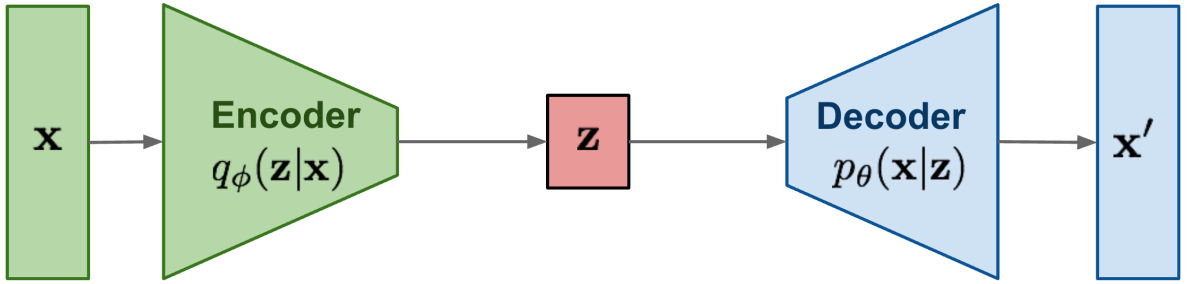
https://github.com/heejkoo/Awesome-Diffusion-Models

Generative models

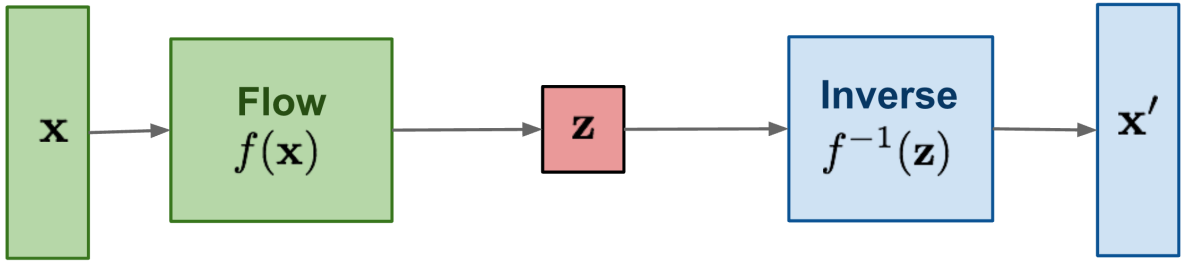
GAN: Adversarial training



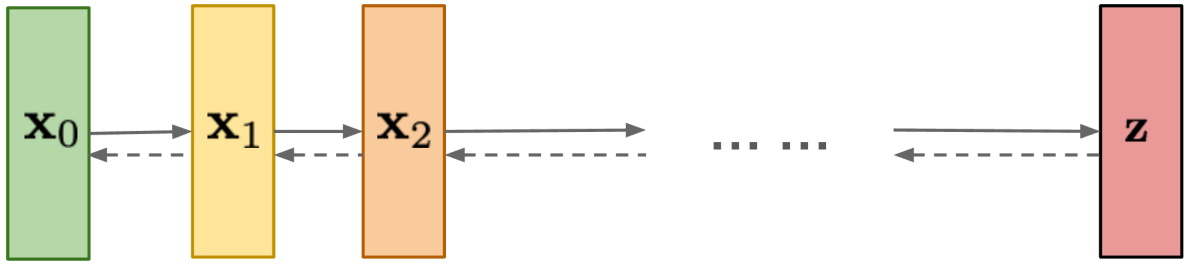
VAE: maximize variational lower bound



Flow-based models: Invertible transform of distributions



Diffusion models: Gradually add Gaussian noise and then reverse



Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

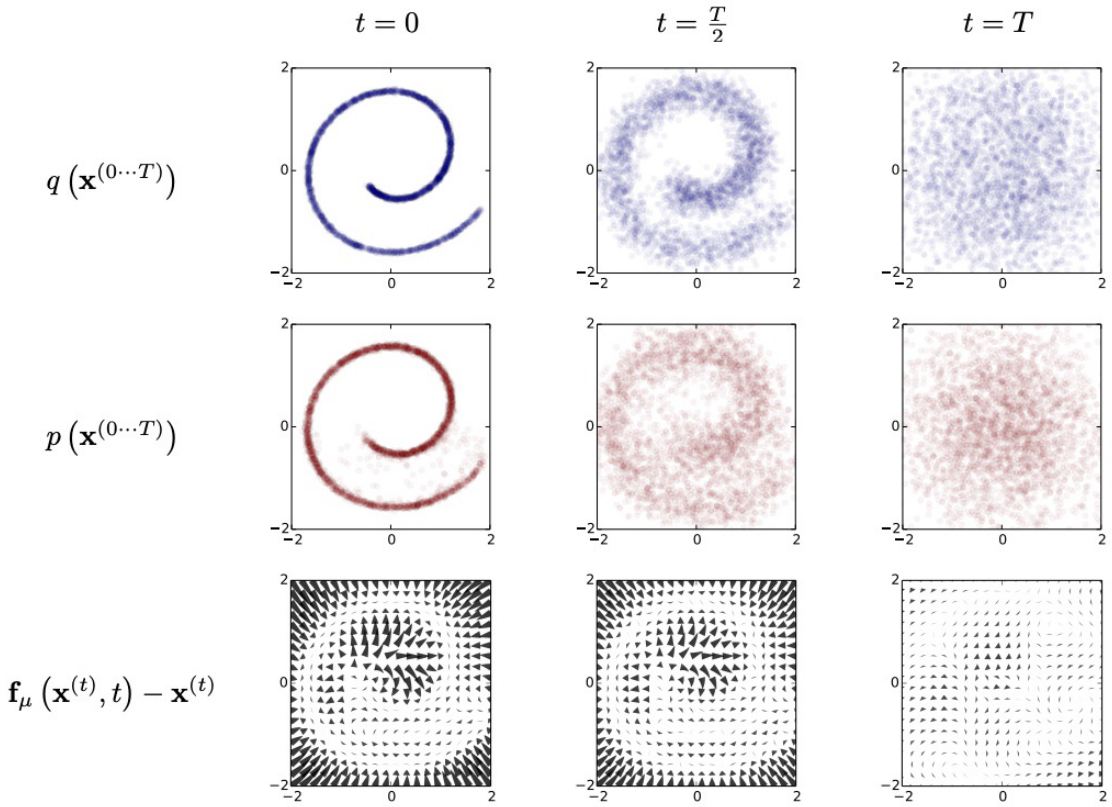
Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves

these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we


$t = 0$
$t = \frac{T}{2}$
$t = T$
$q\left(\mathbf{x}^{(0\cdots T)}\right)$
$p\left(\mathbf{x}^{(0\cdots T)}\right)$
$\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right) - \mathbf{x}^{(t)}$


(a)
(b)
(c)
(d)

Denoising Diffusion Probabilistic Models

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].

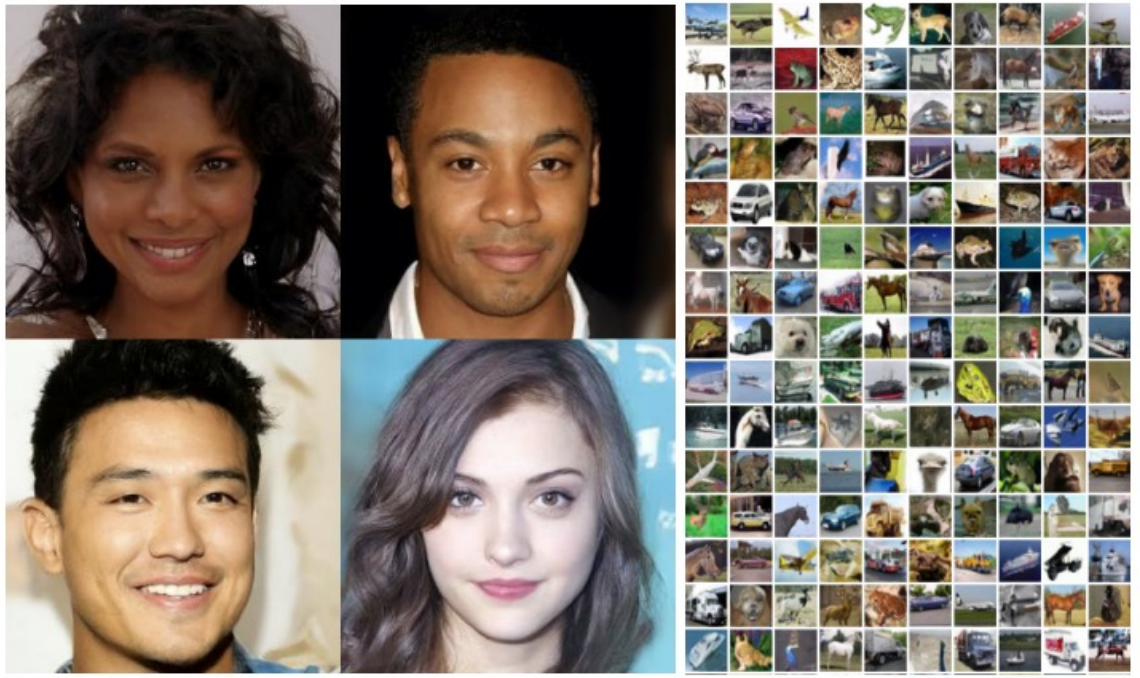
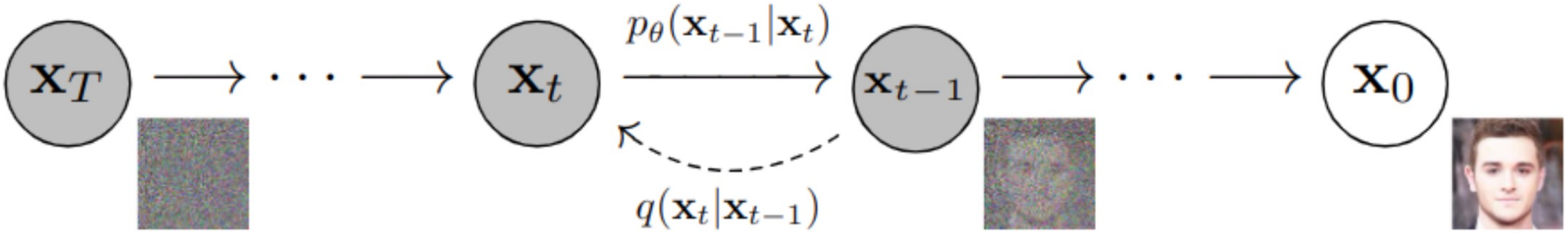


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

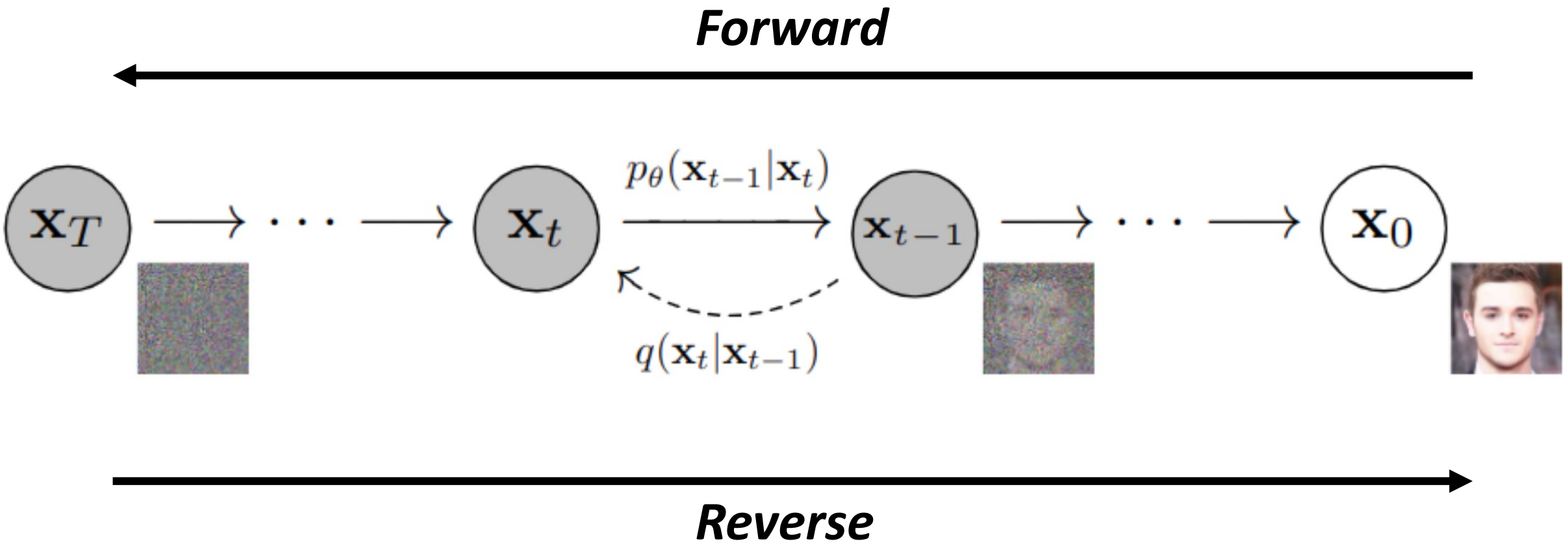
Denoising Diffusion Probabilistic Models

https://huggingface.co/blog/annotated-diffusion

Algorithm


$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$

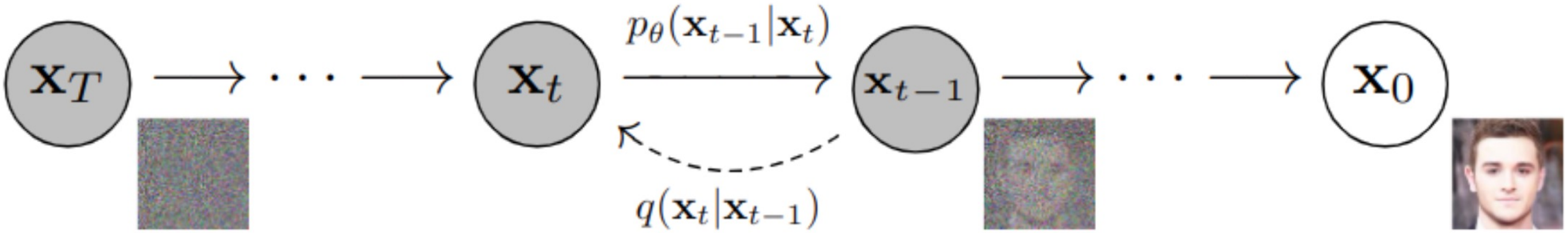
Forward and reverse process



Forward and reverse process

Forward

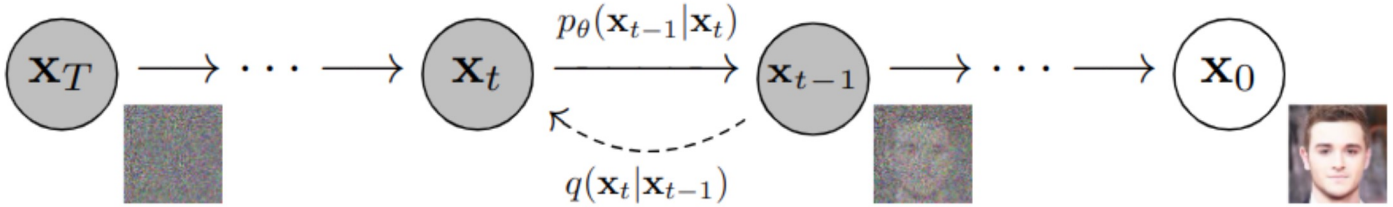
Fixed process that gradually adds Gaussian noise to an image, until you end up with pure noise.



Reverse

Learned reverse denoising diffusion process where a neural network is trained to gradually denoise an image starting from pure noise, until you end up with an actual image

Forward process



Forward diffusion process

Given a schedule $\beta_1 < \beta_2 < \cdots < \beta_T$,

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \quad (1)$$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

We define $\alpha_t = 1 - \beta_t$ and $\overline{\alpha_t} = \prod_{i=1}^{t} \alpha_i$, then we have

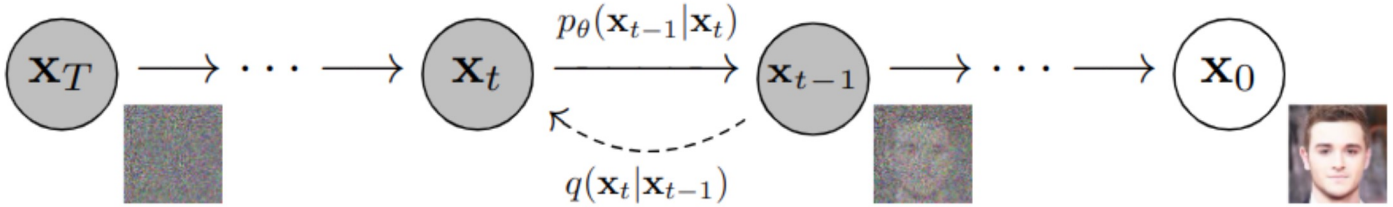
$$\begin{aligned} x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}, \text{ with } \epsilon_{t-1} \sim \mathcal{N}(0, I) \quad (2) \\ &= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\tilde{\epsilon}_t \end{aligned}$$

Hence, we have

$$x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon \quad (3)$$

- Diffusion schedule (beta values)
- DDPM T=1000

Approximate the reverse diffusion



Training: to approximate **the reversed diffusion** $q(x_{t-1}|x_t)$ by a neural network given by $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \beta_t I)$ and $p(x_T) \sim \mathcal{N}(0, I)$, we maximize the usual Variational bound:

$$\mathbb{E}_{q(x_0)} \ln p_\theta(x_0) \geq L_T + \sum_{t=2}^{T} L_{t-1} + L_0 \text{ with, } L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\mu_\theta(x_t, t) - \mu(x_t, \epsilon, t)\|^2 \right] \tag{17}$$

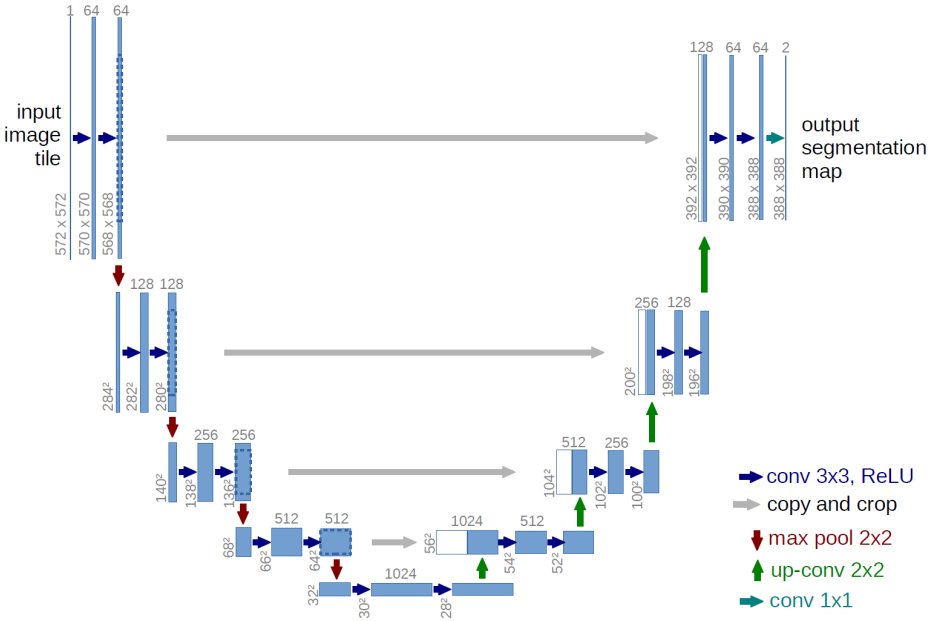
With the change of variable:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \overline{\alpha}_t}} \epsilon_\theta(x_t, t) \right), \tag{18}$$

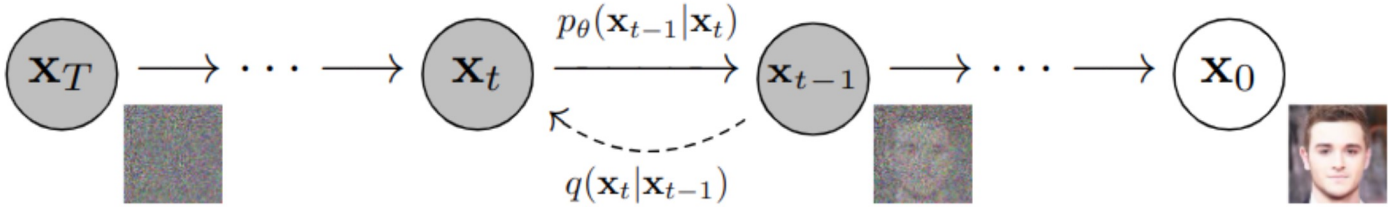
ignoring the prefactor and sampling τ instead of summing over all t, the loss is finally:

$$\ell(\theta) = \mathbb{E}_\tau \mathbb{E}_\epsilon \left[\|\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha}_\tau} x_0 + \sqrt{1 - \overline{\alpha}_\tau} \epsilon, \tau)\|^2 \right] \tag{19}$$

- Train denoising U-Nets
- MSE Loss
- Predict noise at each step



Training and Sampling



Algorithm 1 Training

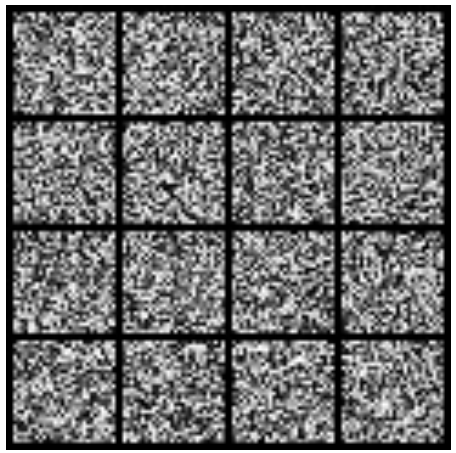
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

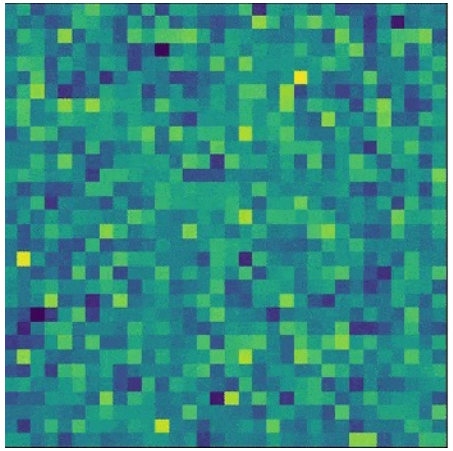
Algorithm 2 Sampling

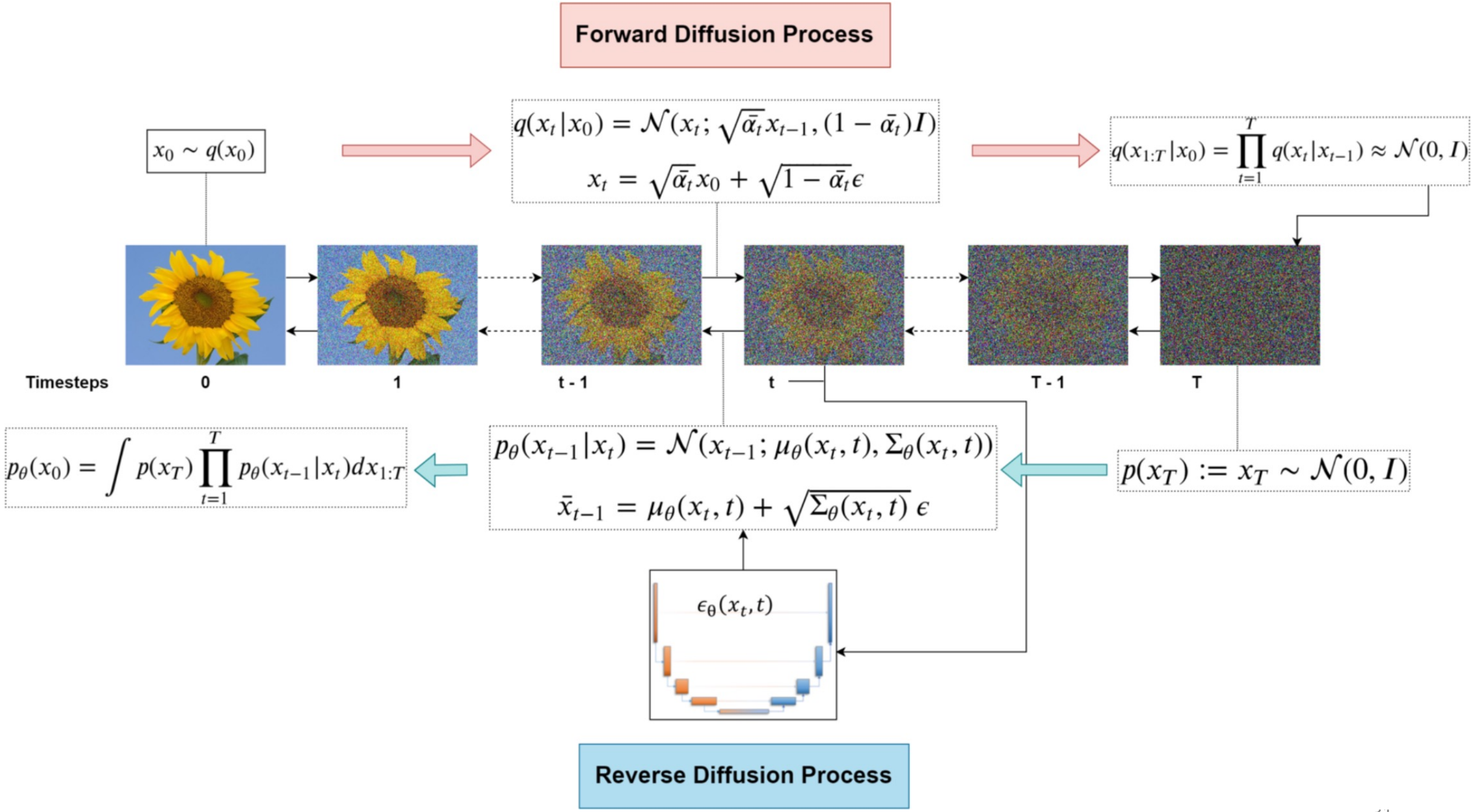
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}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

- we take a random sample $\mathbf{x}_0$ from the real unknown and possibily complex data distribution $q(\mathbf{x}_0)$
- we sample a noise level t uniformally between 1 and T (i.e., a random time step)
- we sample some noise from a Gaussian distribution and corrupt the input by this noise at level t (using the nice property defined above)
- the neural network is trained to predict this noise based on the corrupted image $\mathbf{x}_t$ (i.e. noise applied on $\mathbf{x}_0$ based on known schedule β_t)

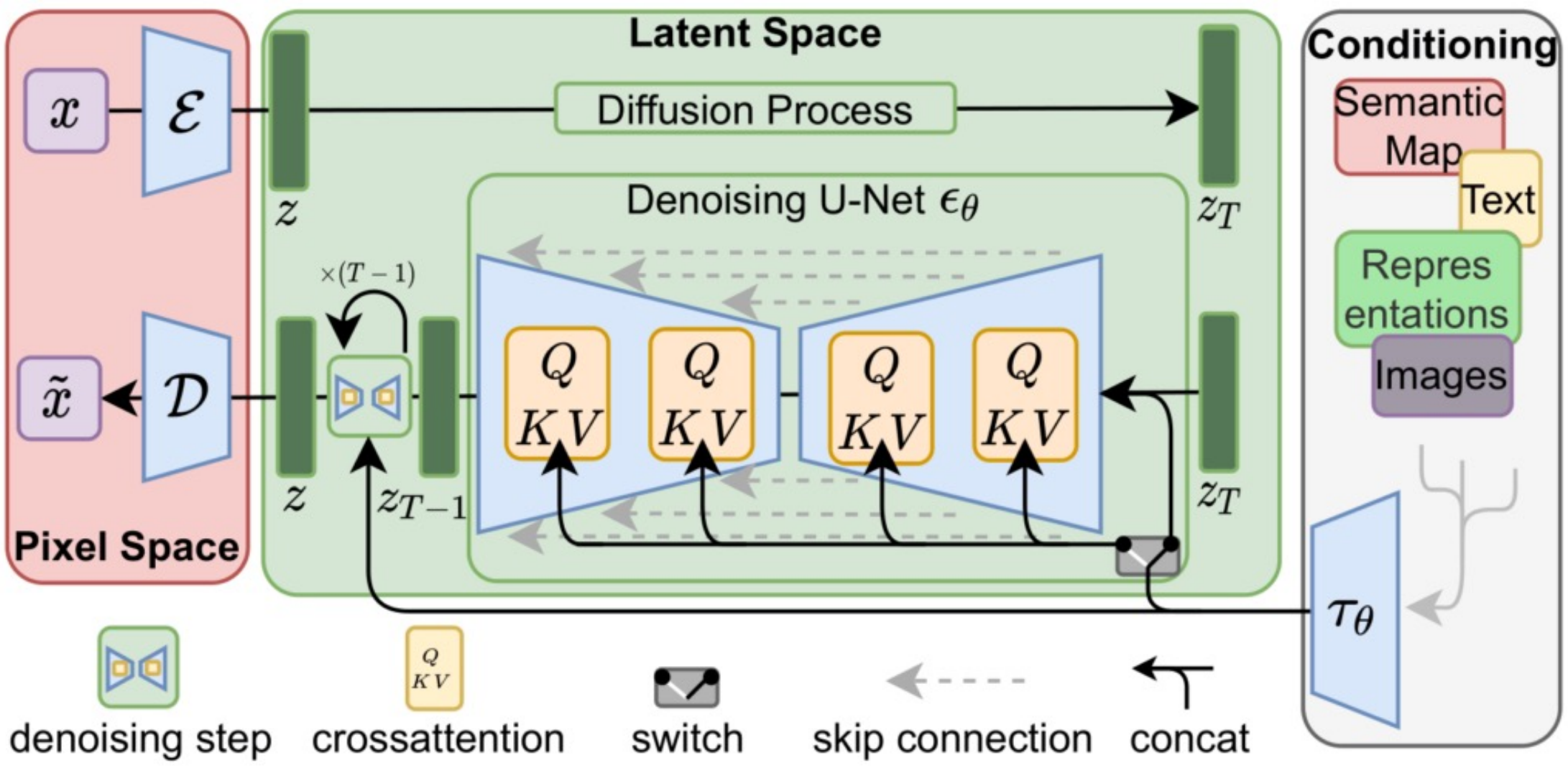
In reality, all of this is done on batches of data, as one uses stochastic gradient descent to optimize neural networks.






Forward Diffusion Process
$x_0 \sim q(x_0)$
$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_{t-1}, (1-\bar{\alpha}_t)I)$
$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$
$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \approx \mathcal{N}(0, I)$
Timesteps
0
1
t - 1
t
T - 1
T
$p_\theta(x_0) = \int p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) dx_{1:T}$
$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$
$\bar{x}_{t-1} = \mu_\theta(x_t, t) + \sqrt{\Sigma_\theta(x_t, t)}\,\epsilon$
$p(x_T) := x_T \sim \mathcal{N}(0, I)$
$\epsilon_\theta(x_t, t)$
Reverse Diffusion Process

Latent diffusion models (pixel vs feature space)



Latent diffusion models (pixel vs feature space)

ControlNet

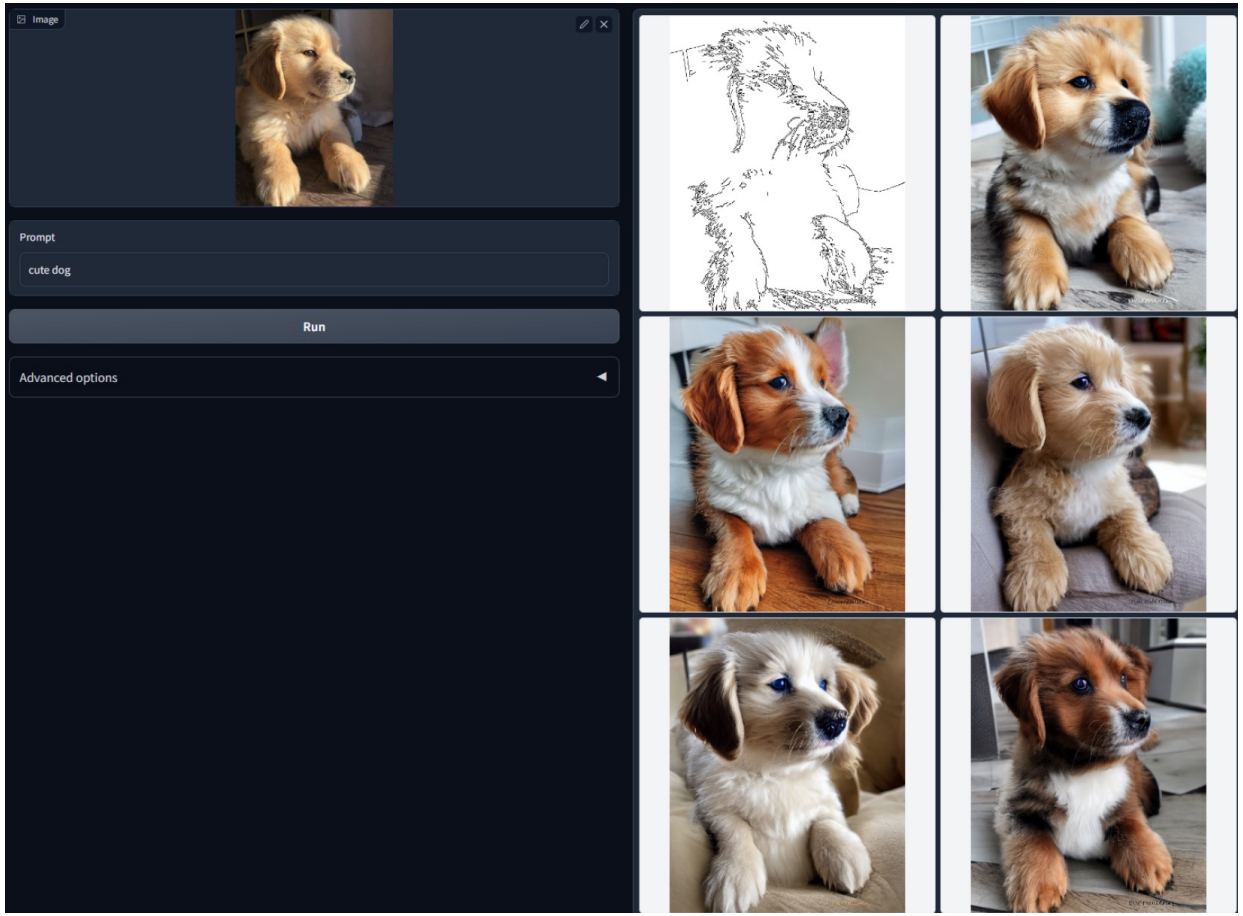


Image
Prompt
cute dog
Run
Advanced options

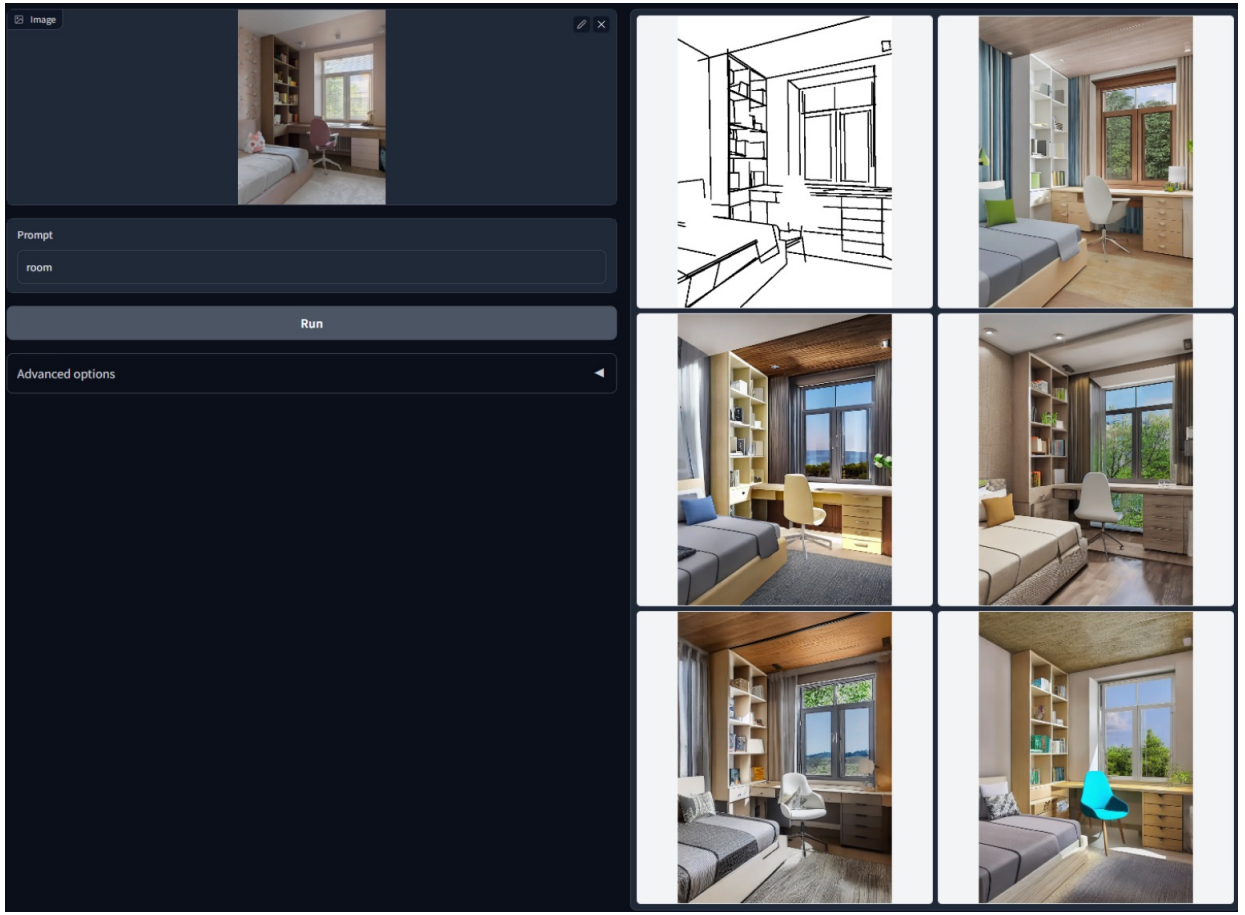


Image
Prompt
room
Run
Advanced options

ControlNet

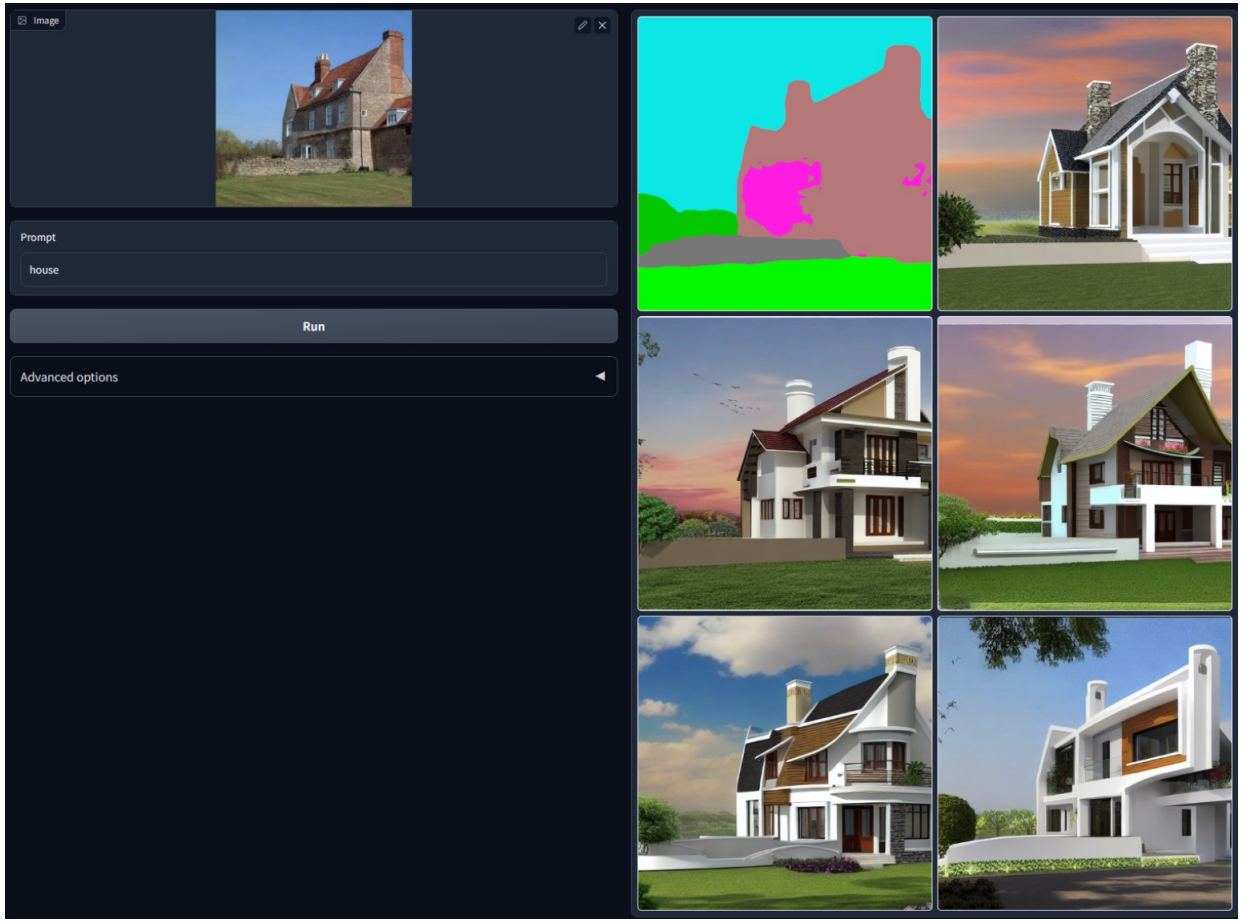


Image
Prompt
house
Run
Advanced options

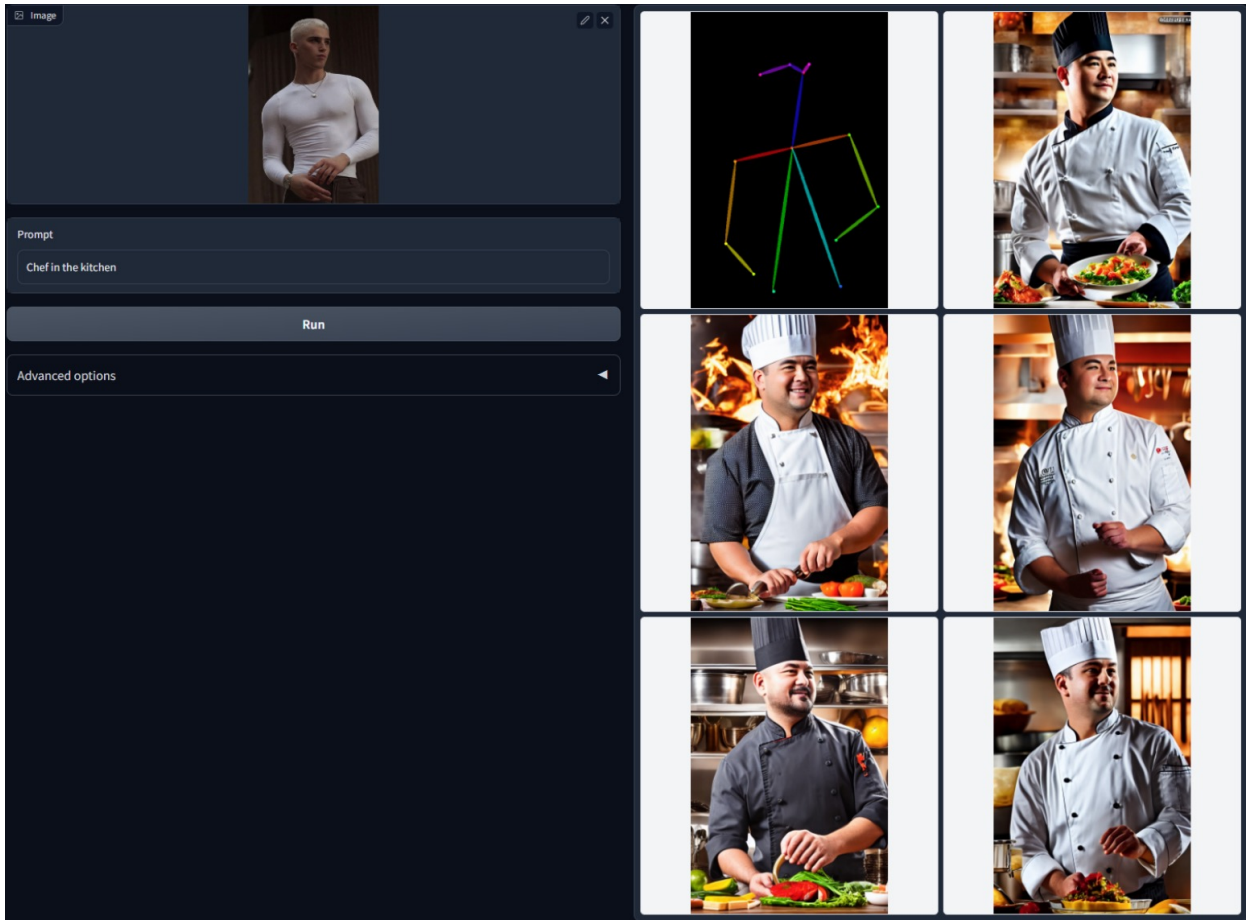


Image
Prompt
Chef in the kitchen
Run
Advanced options