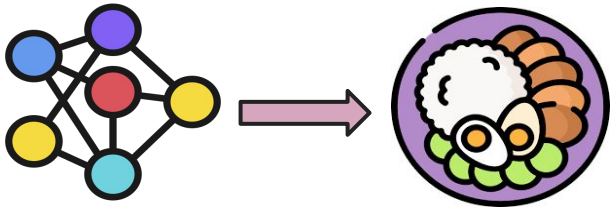
Synthetic Dish Images



Students:
Marco Cáceres, **RA:** m239281
Rafael Pedrosa, **RA:** r273034
Kevin Inofuente, **RA:** k272316

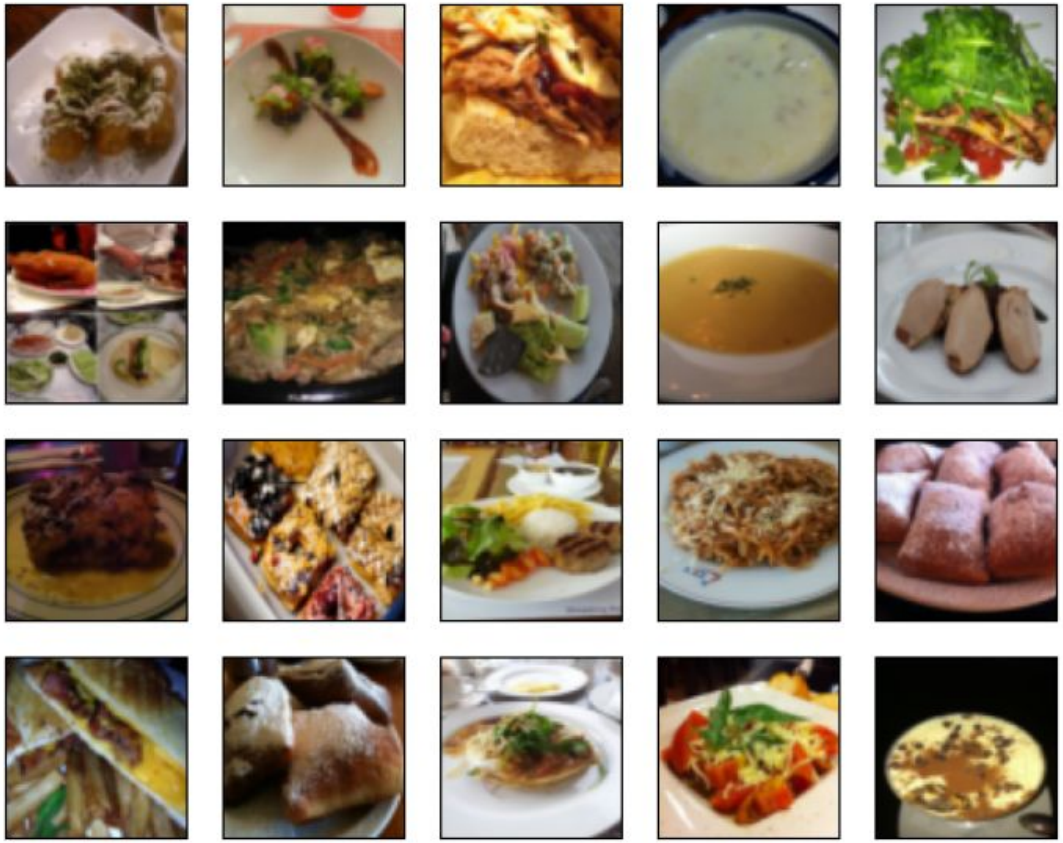
Professora: Paula Paro Costa

Datasets

UNICAMP

- Datasets were difficult to use, as most had few images of many different types of foods
- Food101 used as a standard
  - 100 images of 101 different types of food
- ChineseFoodNet was unavailable and Food11 had too few images

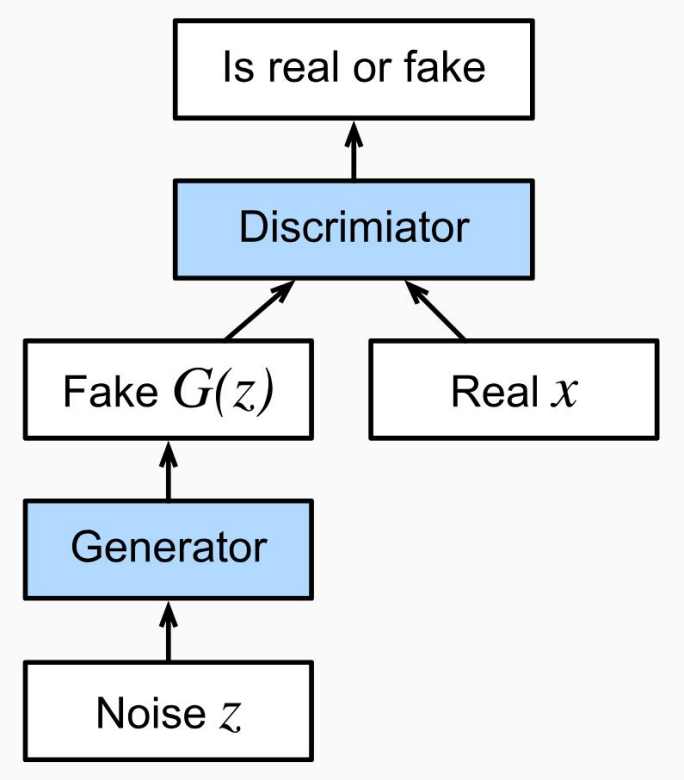
Examples of Food101 Dataset



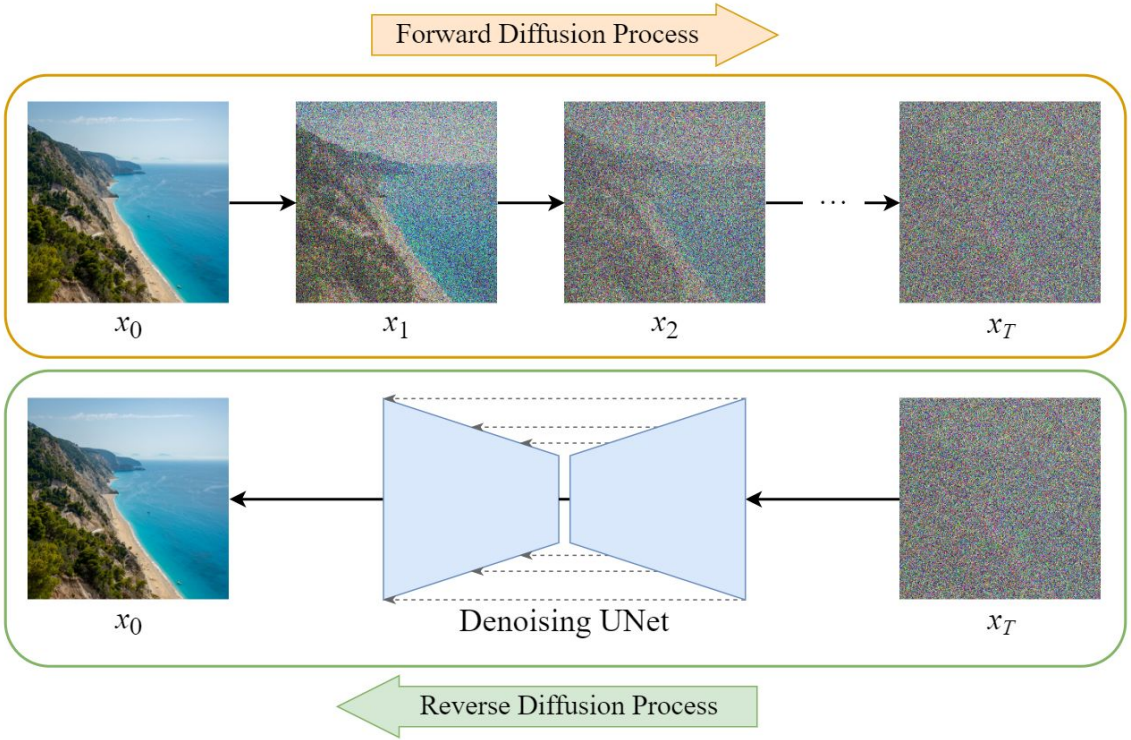
Networks

UNICAMP

- GAN
  - First attempts with a Convolutional GAN
  - Use of the Progressive GAN project
    - Based on https://github.com/odegeasslbc/Progressive-GAN-pytorch
- Diffusion
  - Use of diffusion searching for an alternative
    - Based on https://github.com/dome272/Diffusion-Models-pytorch/tree/main



Model of a Generative Adversarial Network
Source: Dive in to Deep Learning



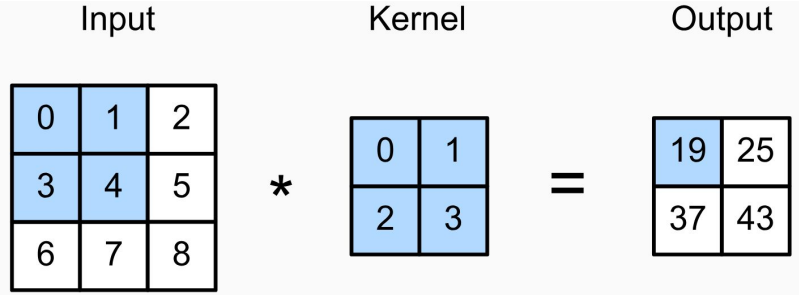
Source: https://medium.com/@jaskaranbhatia/summarizing-the-evolution-of-diffusion-models-insights-from-three-research-papers-6889339eba4

Model of a Diffusion network

First Network

- Convolutional GAN
- Inputs and Outputs 64x64 pixel images
- Generator with 4 convolutional layers plus one non linear layer
- Discriminator with 5 convolutional layers

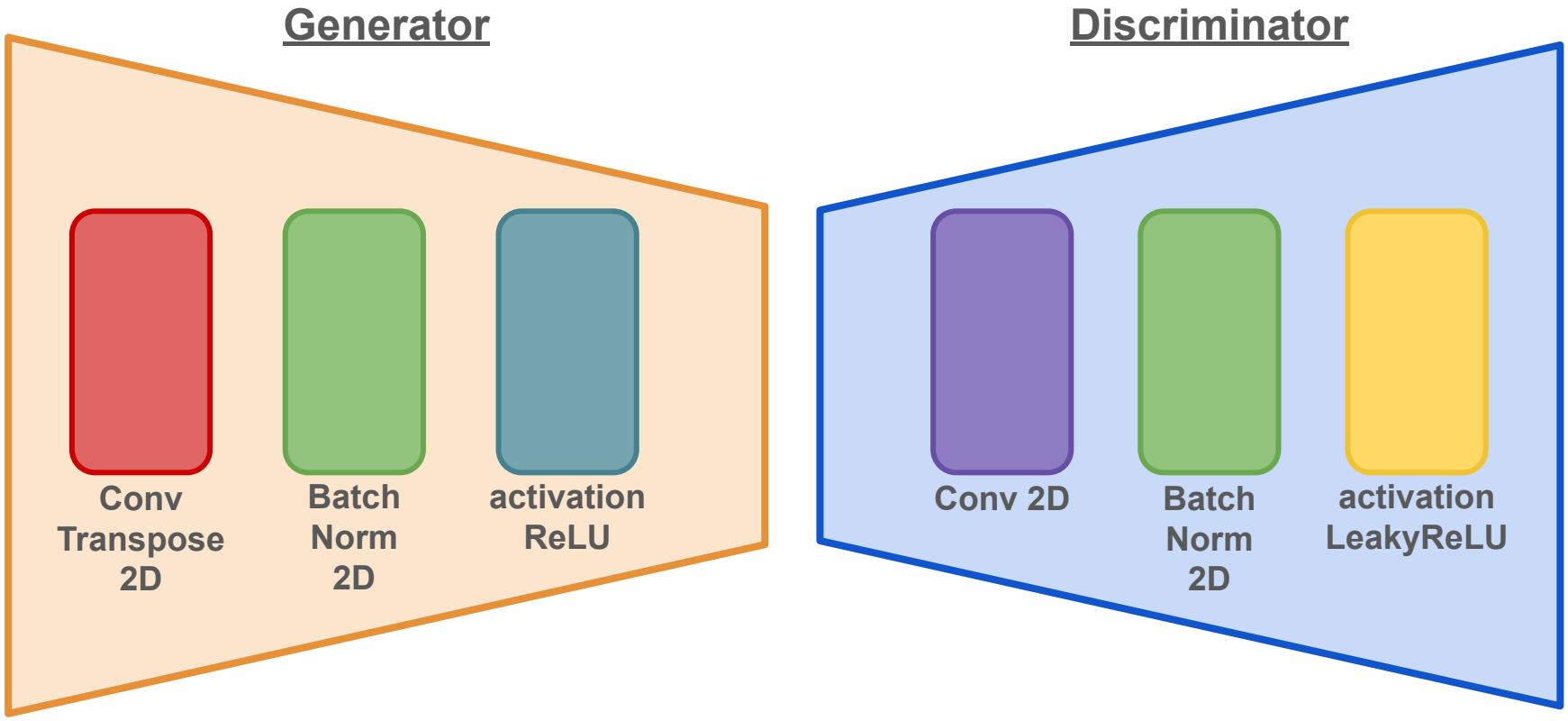
Convolutions for images:



$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43$$

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

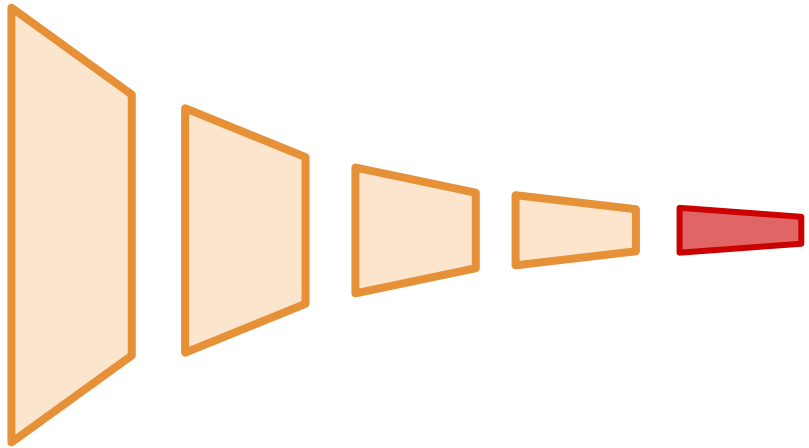
Architecture of Our G_ and D_



Architecture of Our GAN

UNICAMP

Net-Generator

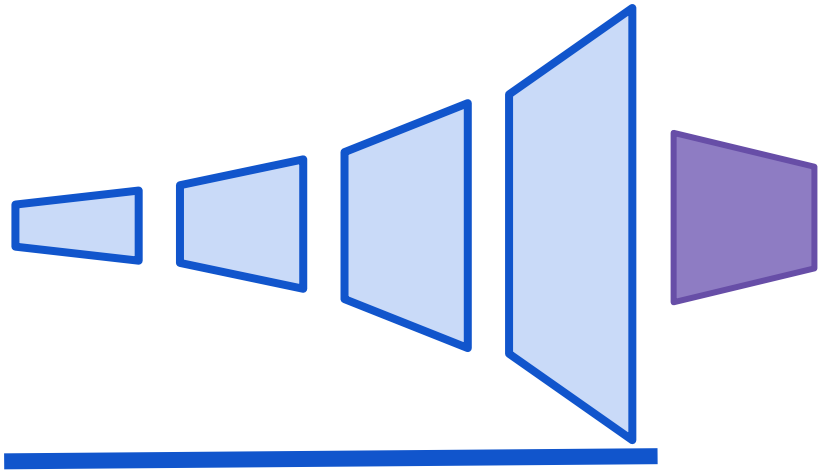


Kernel=4
Strides=1
Padding=0

Kernel=4
Strides=2
Padding=1

Kernel=4

Net-Discriminator



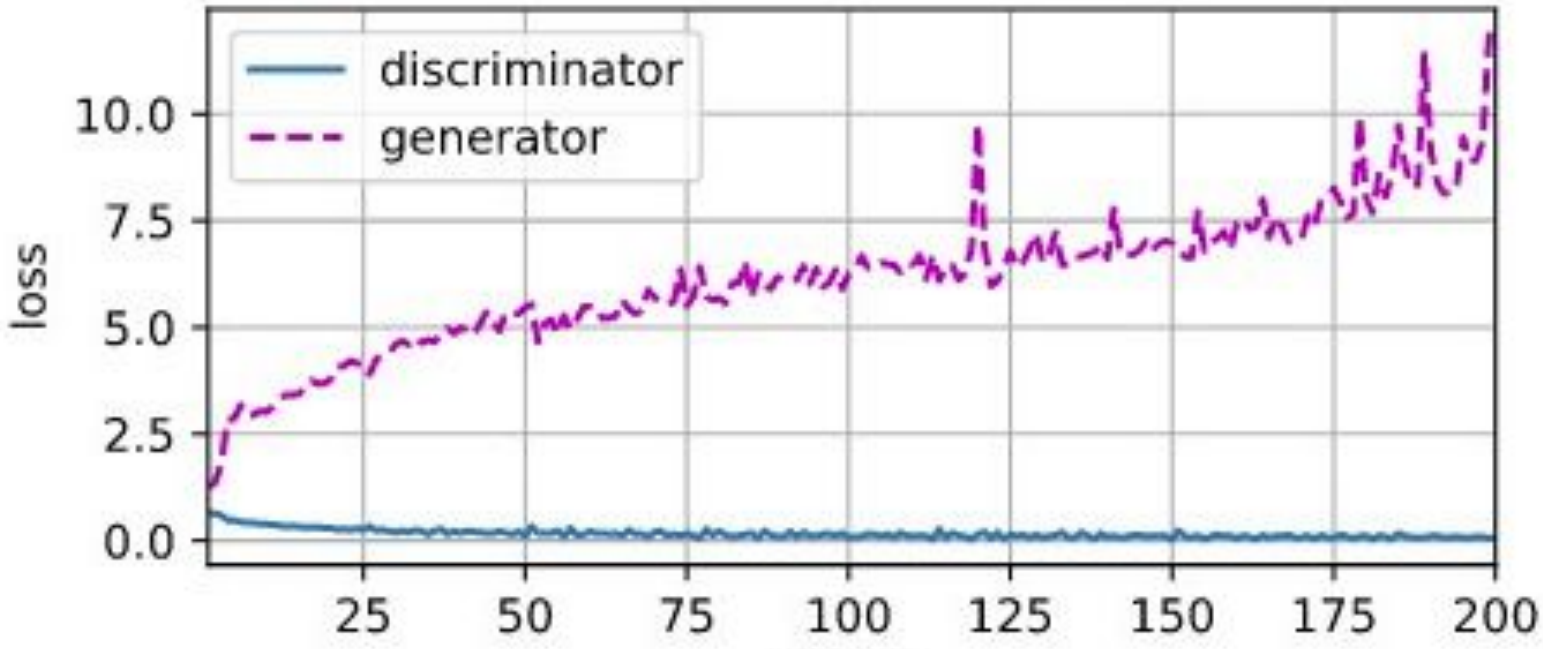
Kernel=4
Strides=2
Padding=1

Kernel=4

Result 1

Set:

Latent_dim: 100 **Lr:** 0.0005 **Epochs:** 150

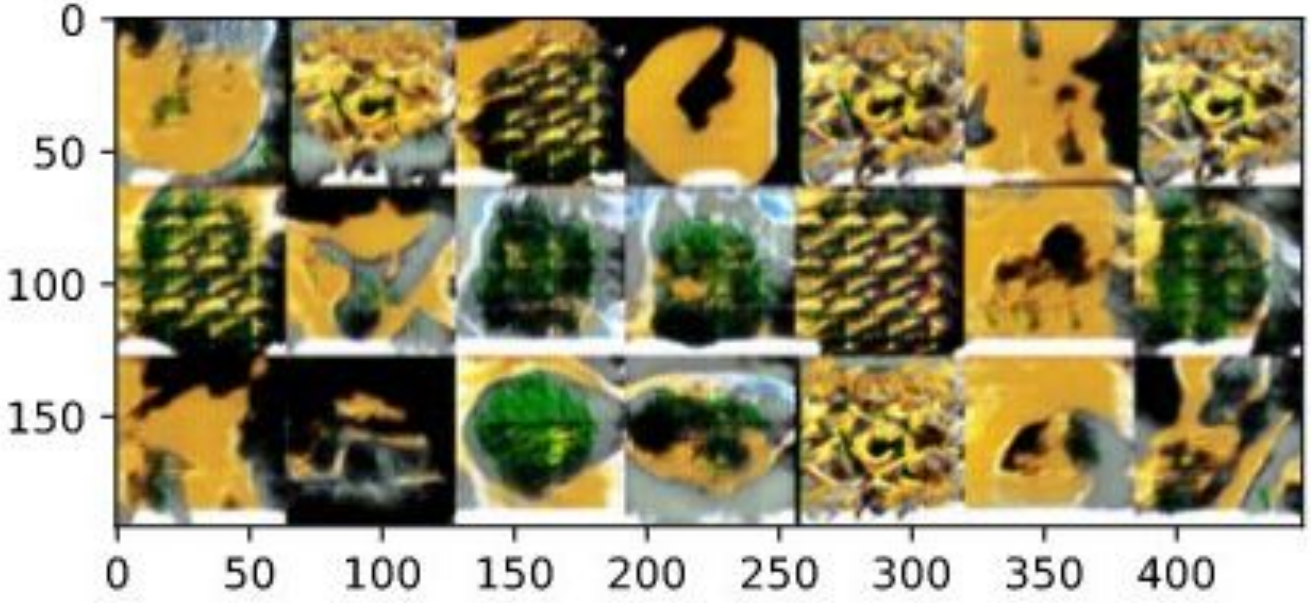


Failure to produce recognizable images

Result 1

Set:

Latent_dim: 100 **Lr:** 0.0005 **Epochs:** 150



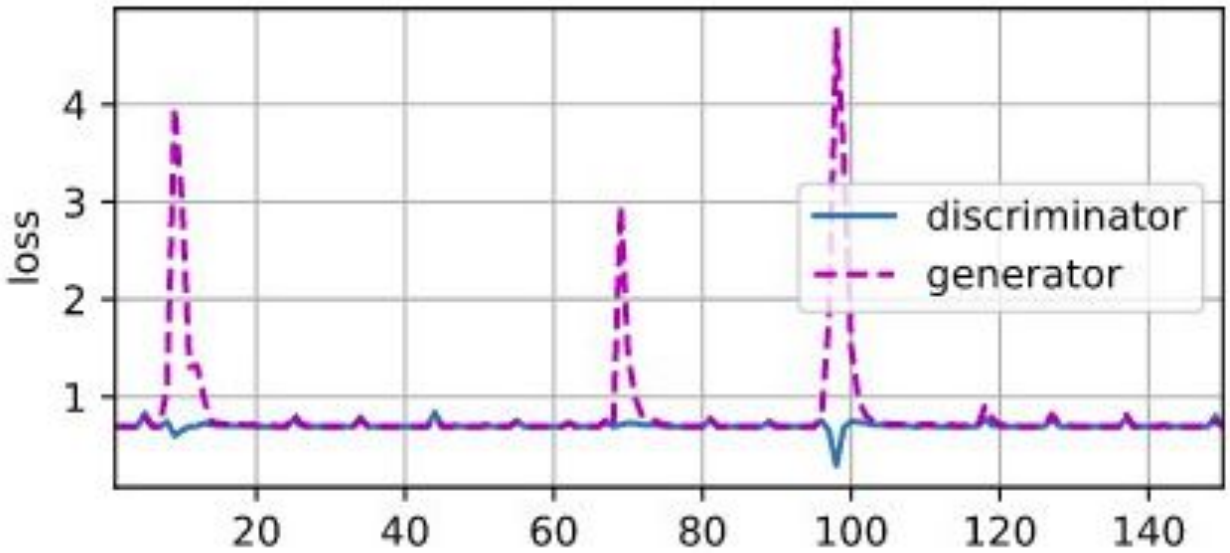
Failure to produce recognizable images

Result 2

Set:

Latent_dim: 100 **Lr:** 0.0005 **Epochs:** 150

Train generator 5 times as often as discriminator



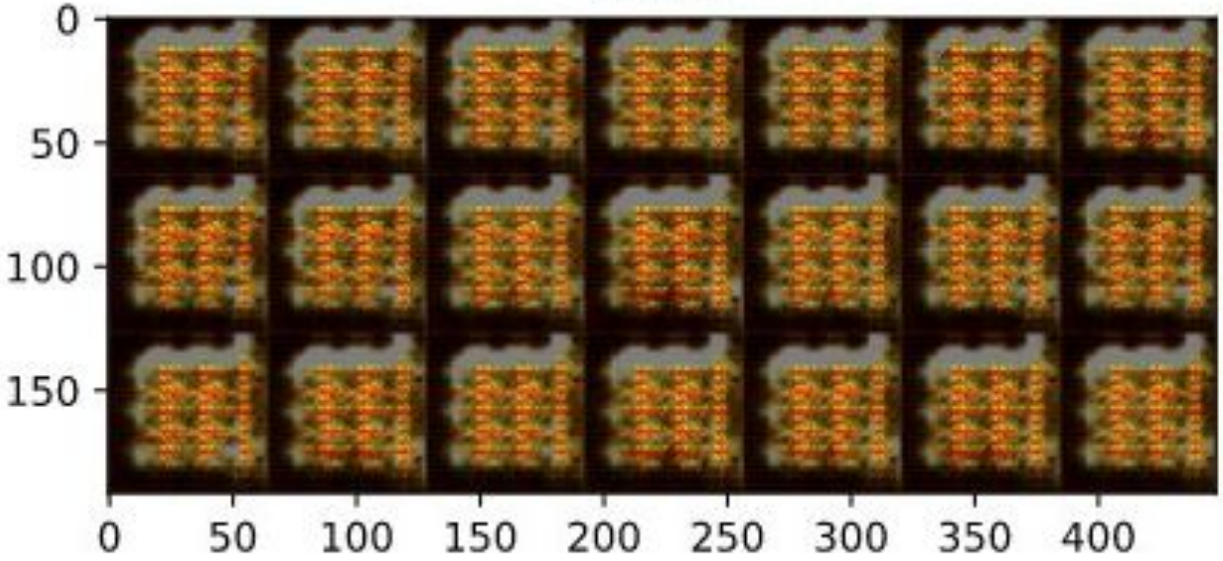
Failure to produce recognizable images

Result 2

Set:

Latent_dim: 100 **Lr:** 0.0005 **Epochs:** 150

Train generator 5 times as often as discriminator

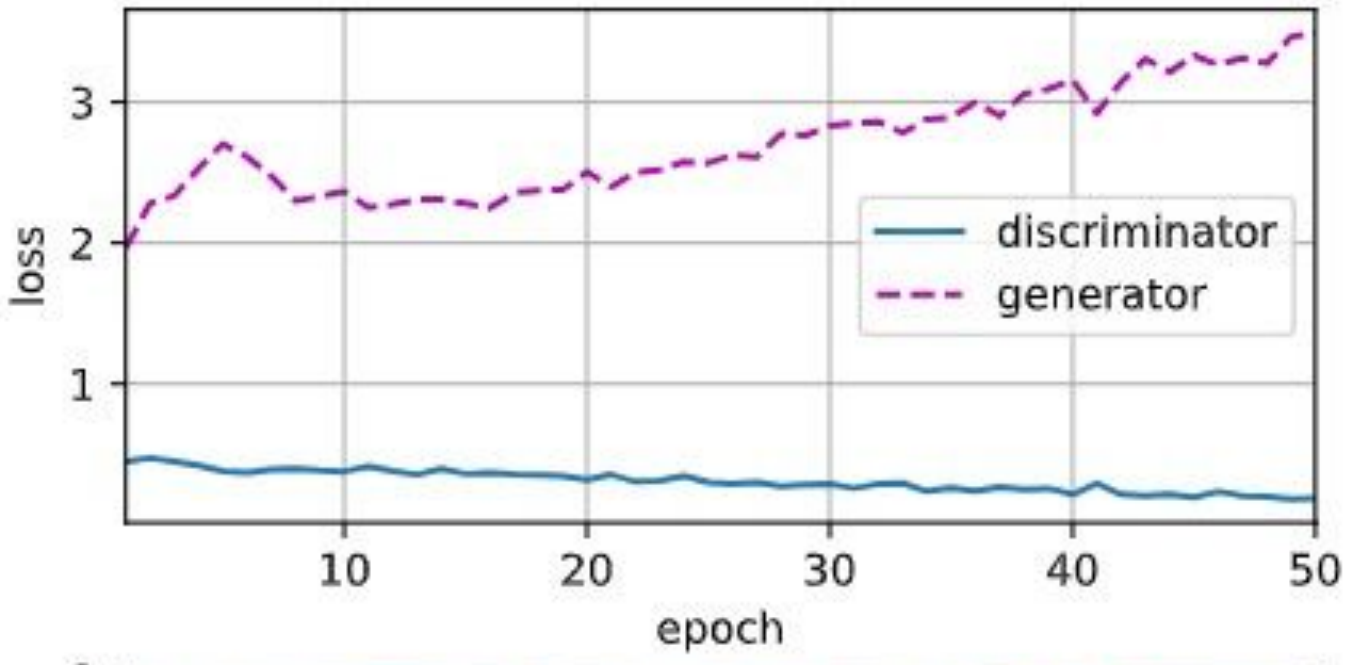


Failure to produce recognizable images

Result 3

Set:

Latent_dim: 200 **Lr:** 0.0003 **Epochs:** 50



Out:

Loss_D: 0.188 **Loss_G:** 3.492 **Examples/sec on CPU:** 64.7

Result 3

Set:

Latent_dim: 200 **Lr:** 0.0003 **Epochs:** 50

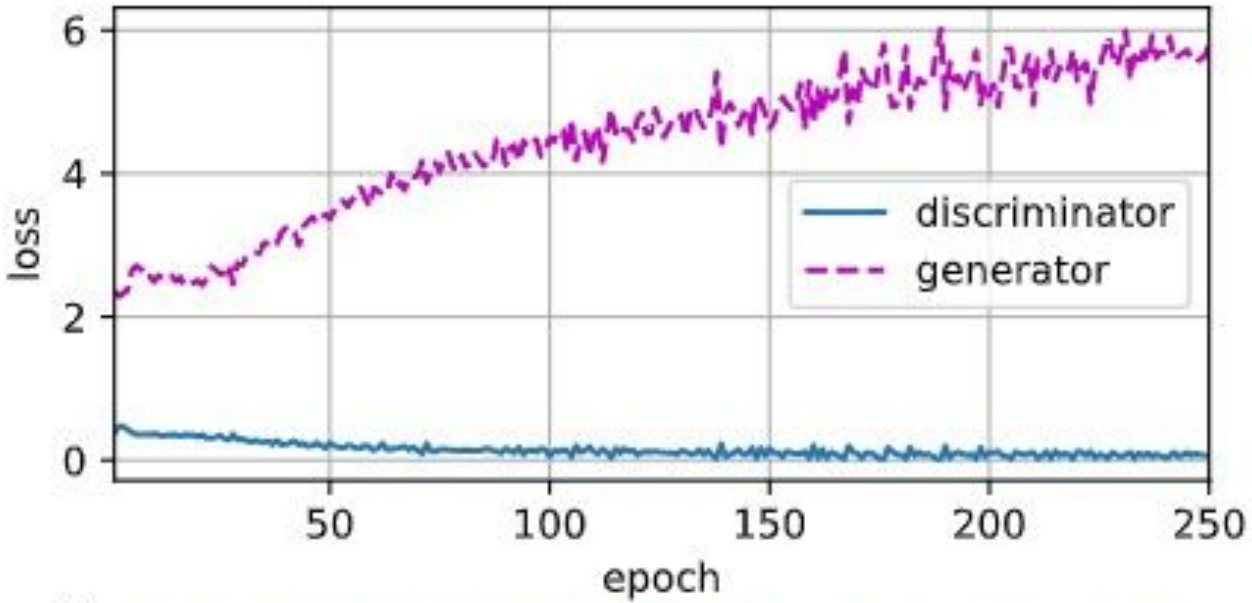
Out:

Loss_D: 0.188 **Loss_G:** 3.492 **Examples/sec on CPU:** 64.7

Result 4

Set:

Latent_dim: 200 **Lr:** 0.0003 **Epochs:** 250



Out:

Loss_D: 0.065 **Loss_G:** 5.816 **Examples/sec on CPU:** 63.9

Result 4

Set:

Latent_dim: 200 **Lr:** 0.0003 **Epochs:** 250

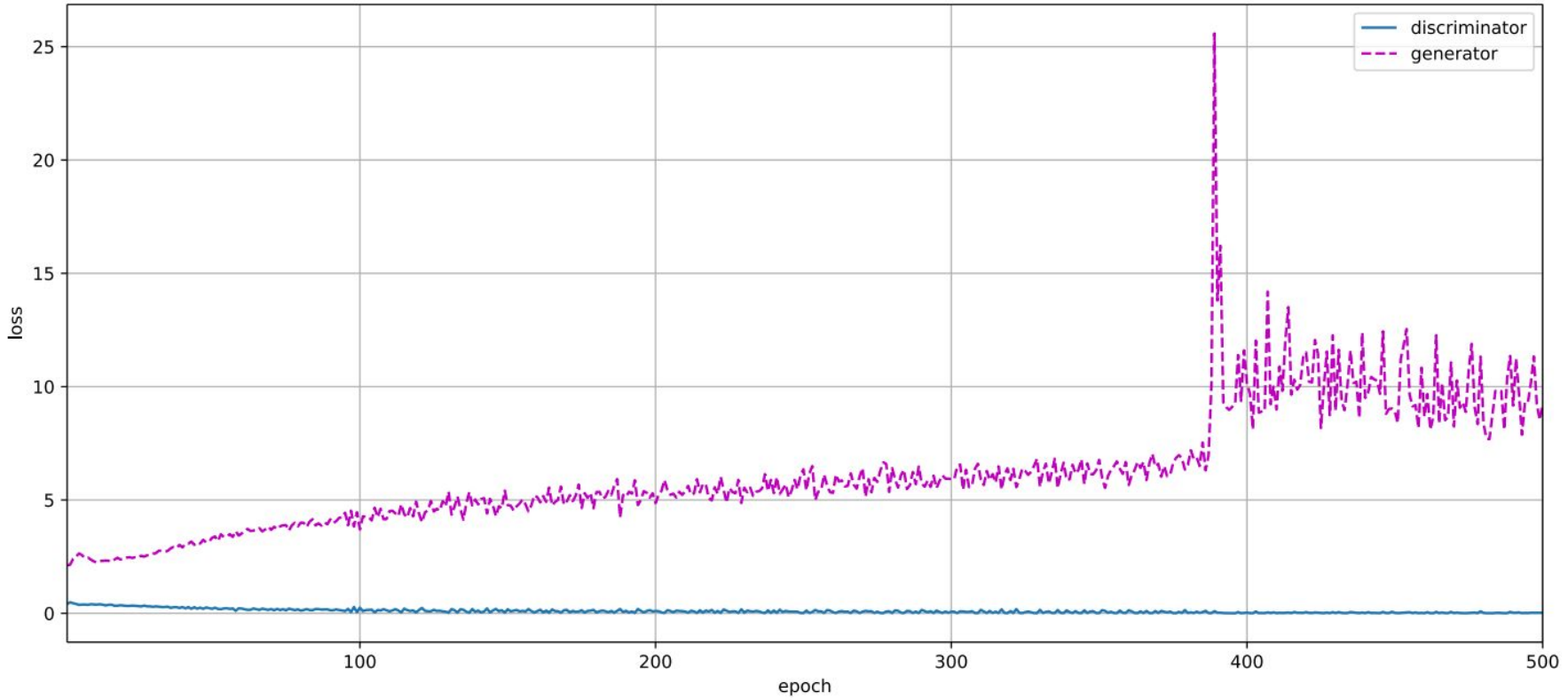
Out:

Loss_D: 0.065 **Loss_G:** 5.816 **Examples/sec on CPU:** 63.9

Result 5

Set:

Latent_dim: 200 **Lr:** 0.0003 **Epochs:** 500

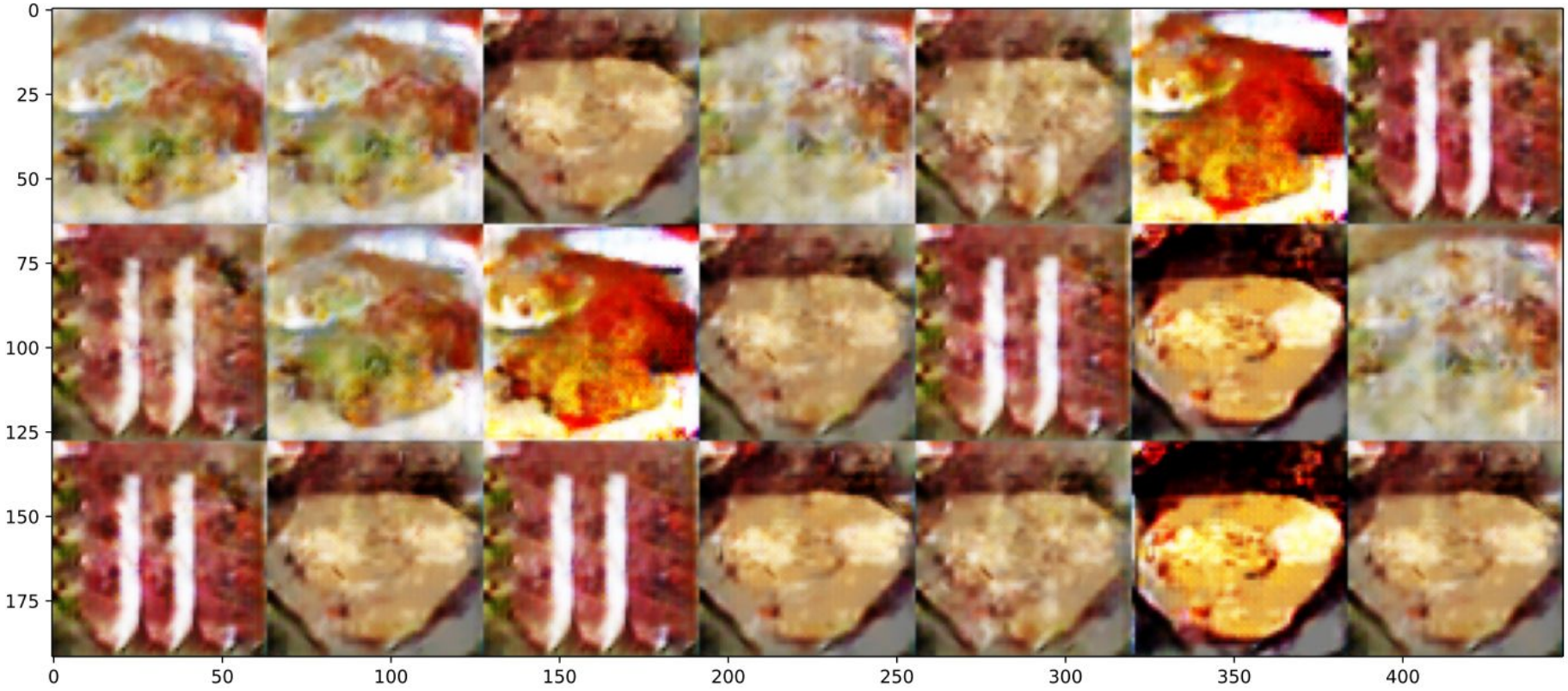


Result 5

UNICAMP

Set:

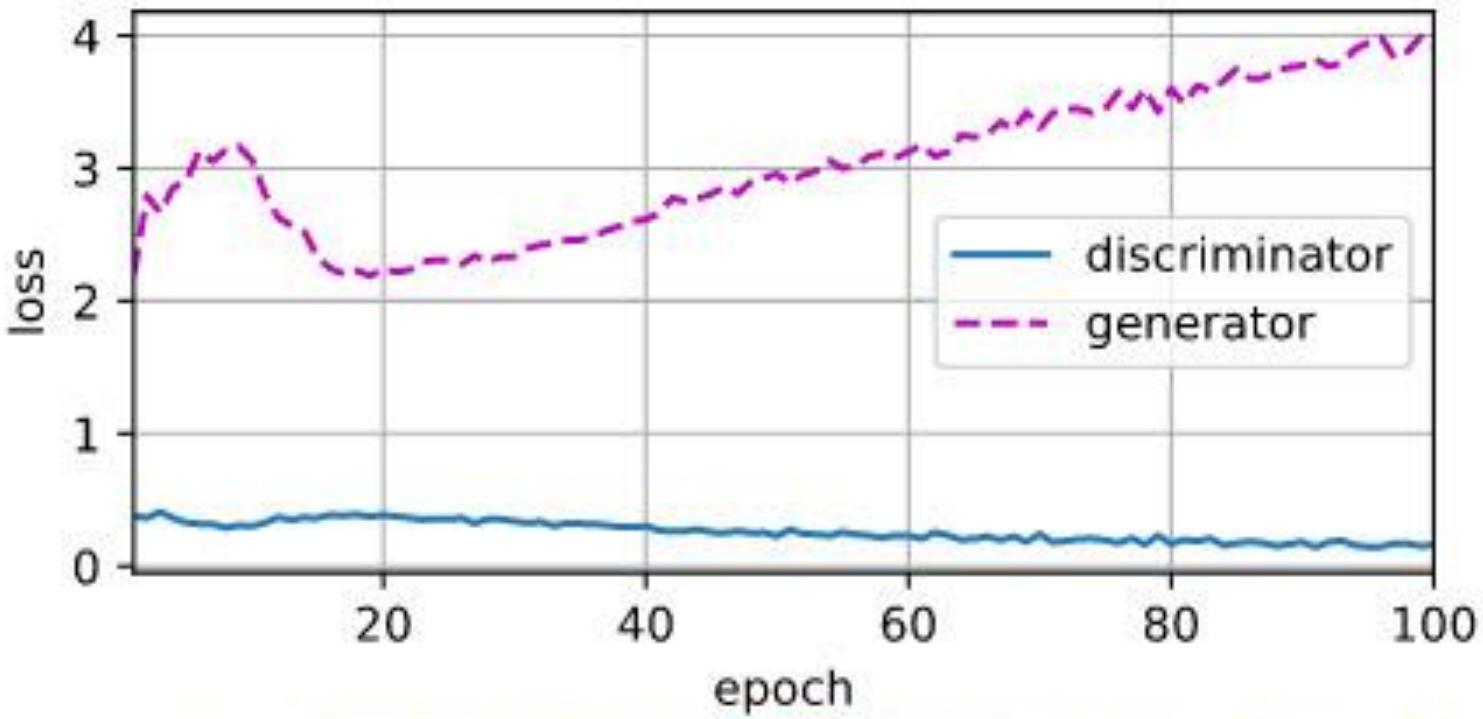
Latent_dim: 200 **Lr:** 0.0003 **Epochs:** 500



Result 6

Set:

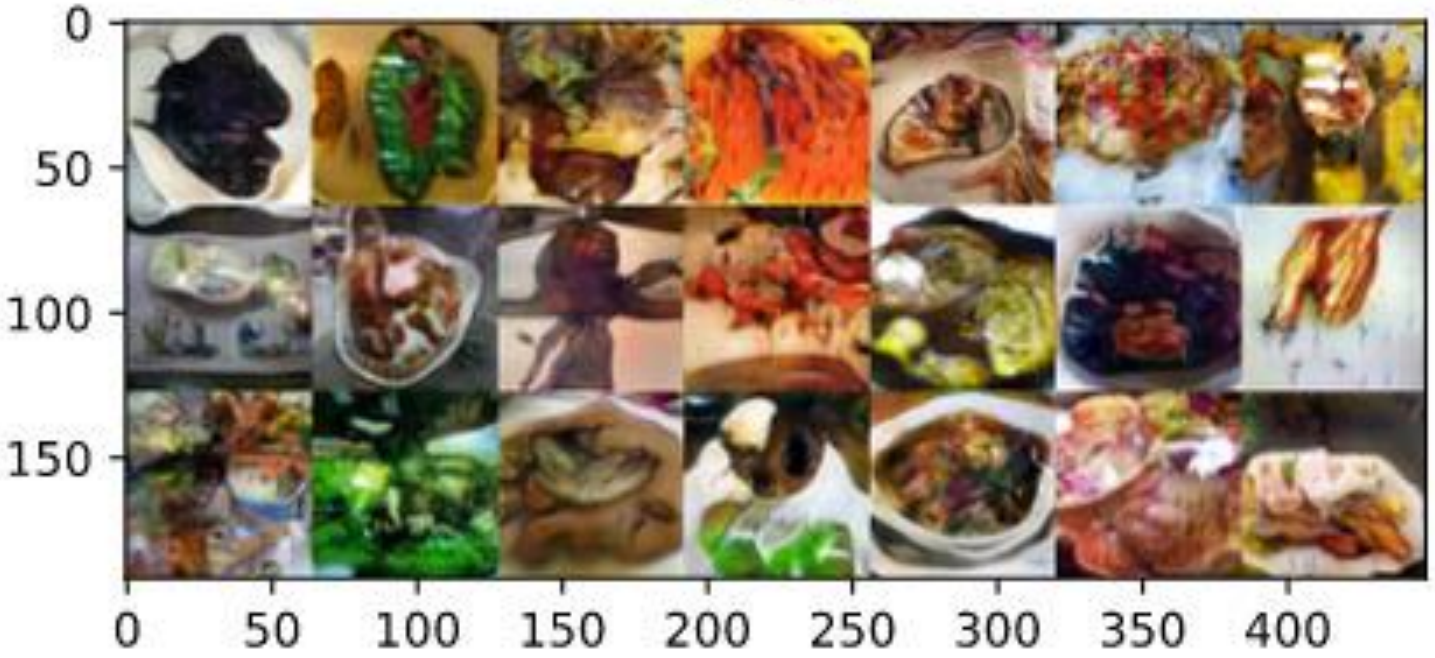
Latent_dim: 100 **Lr:** 0.0002 **Epochs:** 100



Result 6

Set:

Latent_dim: 100 **Lr:** 0.0002 **Epochs:** 100



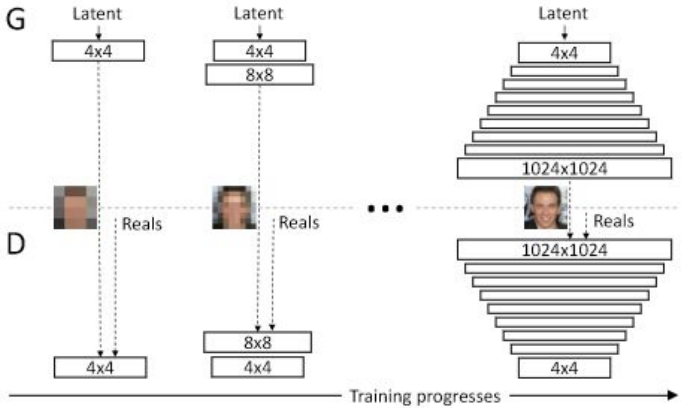
Results 1-6 Conclusions

UNICAMP

- Smaller learning rates were good for image quality
- Increases in latent dimensions also good
- Defects start appearing on high number of epochs
- Discriminator loss tends to zero, generator can’t properly catch-up
- Attempts at overtraining generator failed
    - Code issue?

Second Network

- Progressive GAN
  - Starts with smaller images and less layers, then progressively increases both image and network sizes
- Images begin at 8x8 sizes and progress up to 128x128
- Begins with 2 layers on generator and discriminator, grows to 6 layers on each



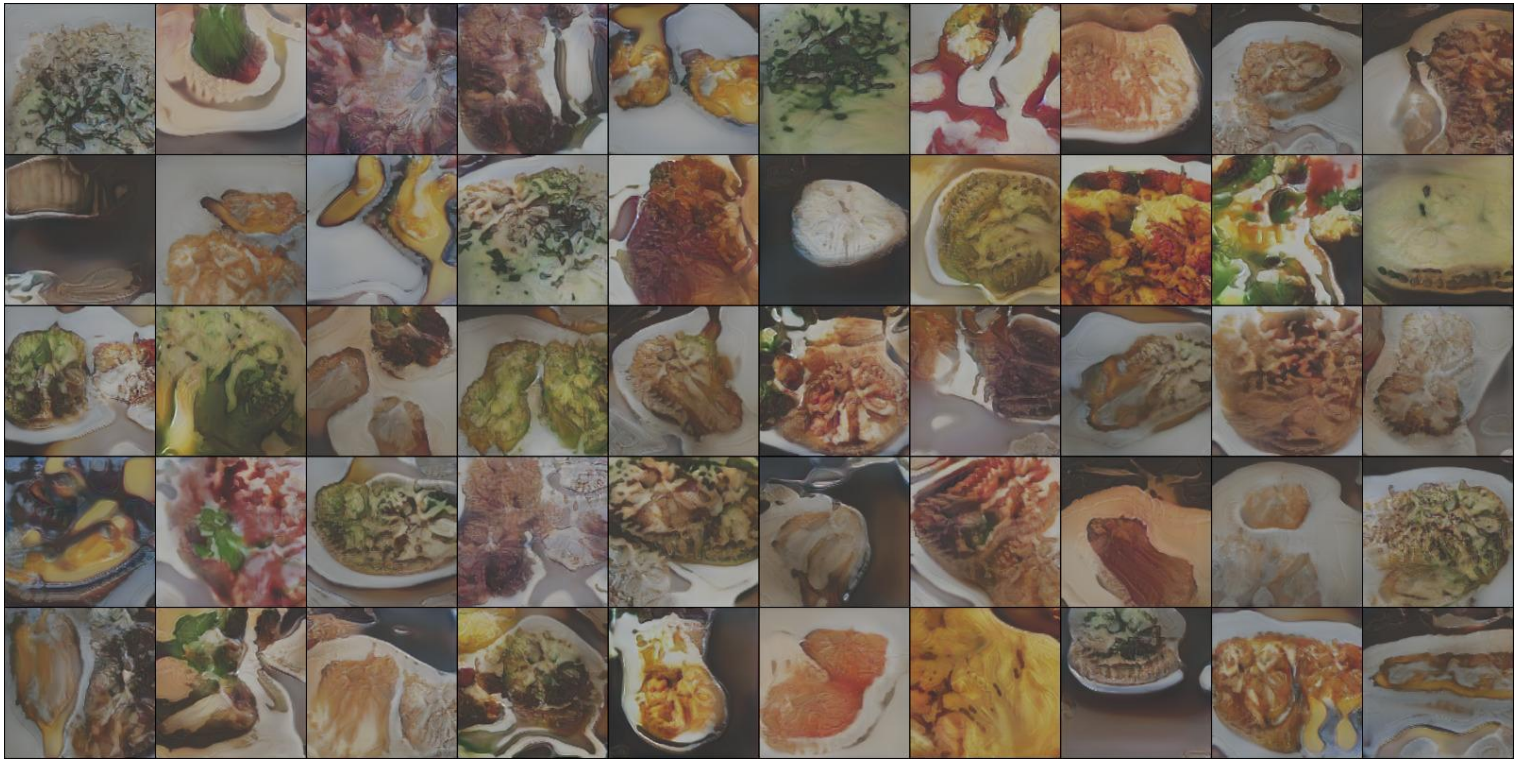
Source: PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION, Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen

Result 7

UNICAMP

Set:

Progressive GAN (8 - 128px) **Lr:** 0.001 **Iterations:** 300000

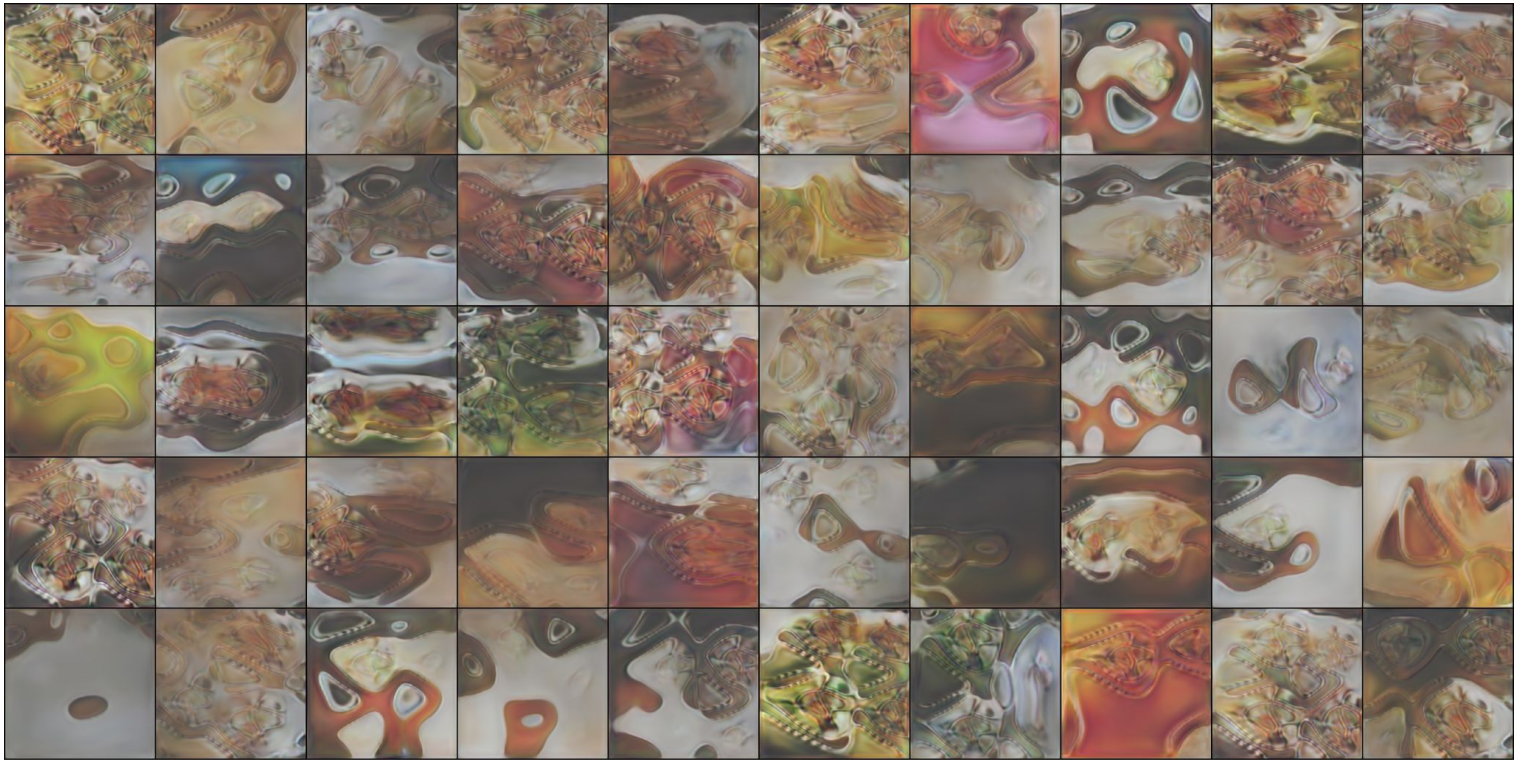


Result 8

UNICAMP

Set:

Progressive GAN (8 - 128px) **Lr:** 0.0003 **Iterations:** 30000



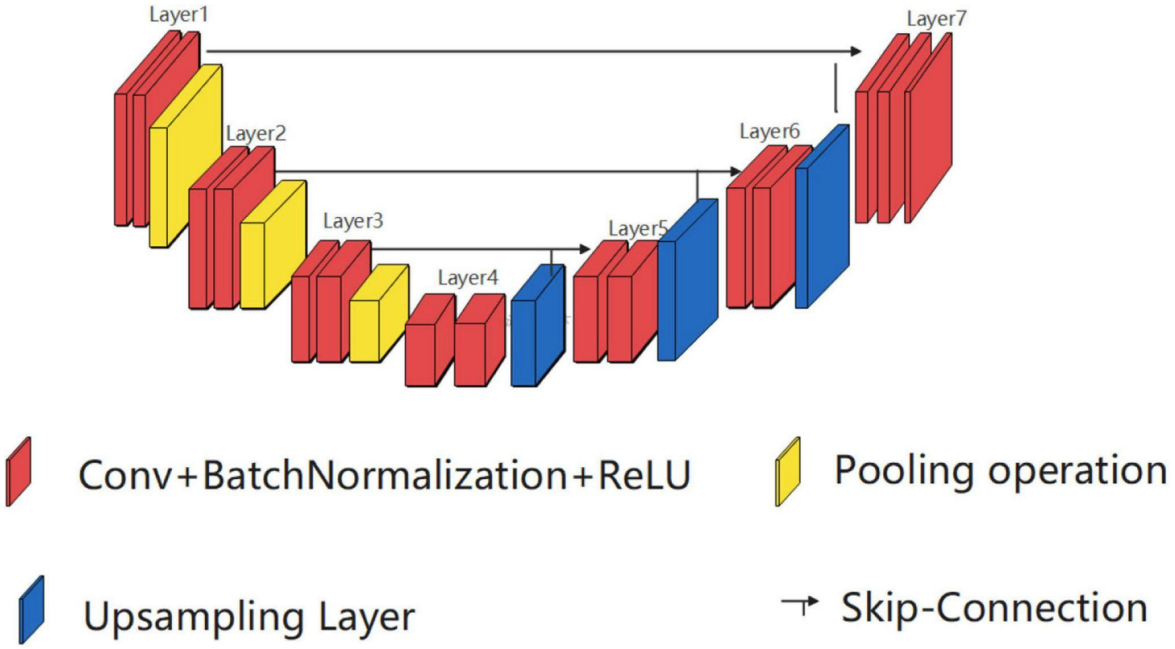
Results 7-8 Conclusions

UNICAMP

- Progressive GAN allows for faster training for bigger networks
- Results at higher image sizes
- Longer training times affected test run quantity

Third Network

- Diffusion model
  - UNet type
- Varying beta

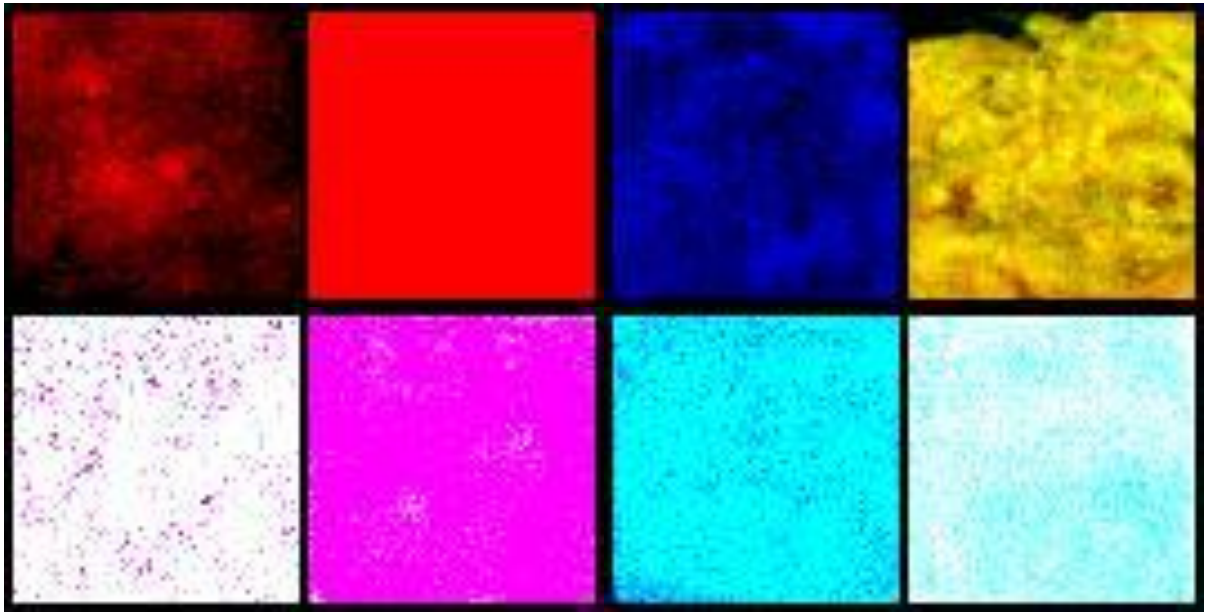


Source: https://www.frontiersin.org/articles/10.3389/fnagi.2022.841297/full

Result 9

Set:

Diffusion (64px) **Lr:** 0.0003 **Epochs:** 10



Result 9 Conclusions

UNICAMP

- Diffusion models often ran into memory issues
- Iterations/Epochs took longer than GANs
- Memory and time issues forced worse training parameters
    - Unsatisfactory results ensued

Overall Conclusions

UNICAMP

- GANs seem to be more economical than Diffusion models
- Even though Diffusion models produce very good results, we failed to reproduce that
- Progressive GANs are good solutions to the problem of increasing network resources demand
- However, training a network from scratch is always costly