Deep Convolutional Generative Adversarial Network

Alireza Javaheri

OVERVIEW

Objectives

Architecture

Datasets

Training

Results

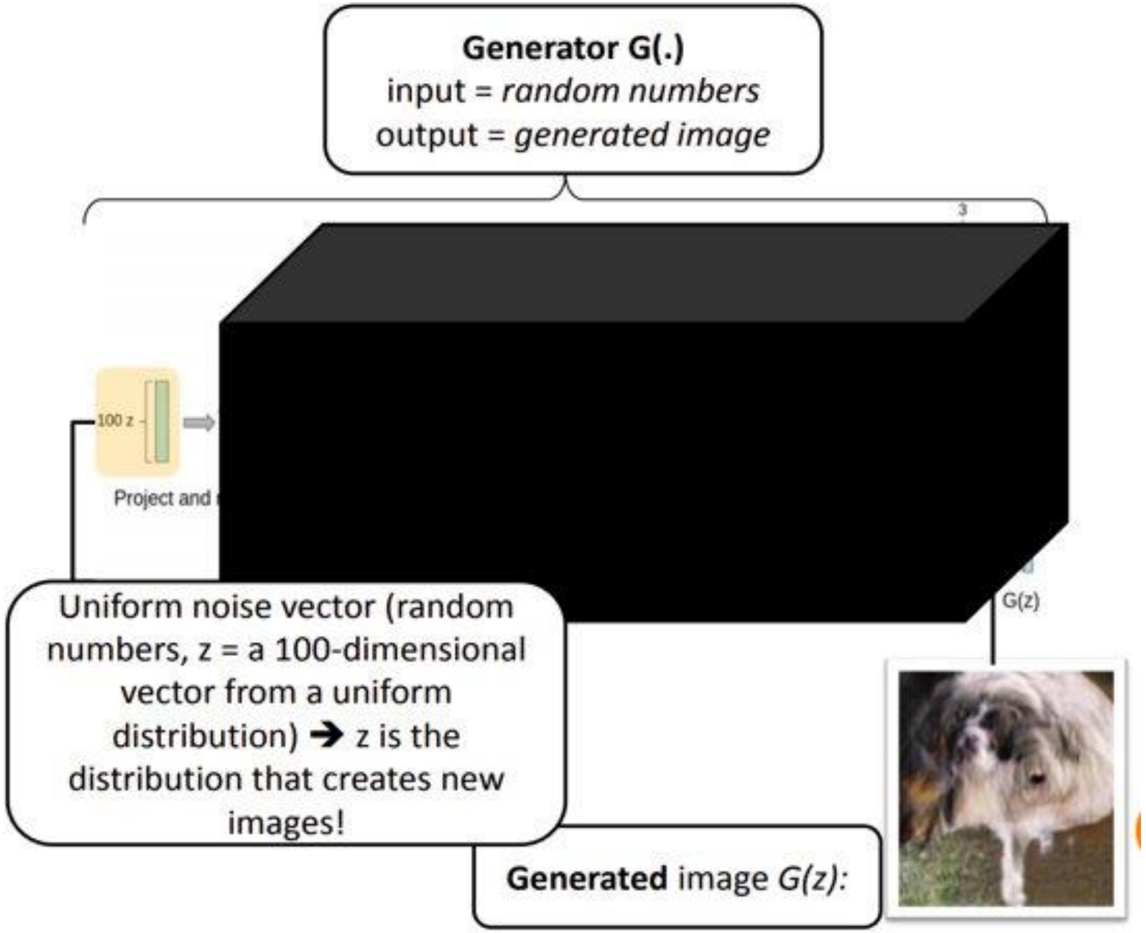
OBJECTIVES

Propose and evaluate a set of constraints on the architectural topology of Convolutional GANs that make them stable to train in most settings. We name this class of architectures Deep Convolutional GANs (DCGAN)

Use the trained discriminators for image classification tasks, showing competitive performance with other unsupervised algorithms

We visualize the filters learnt by GANs and empirically show that specific filters have learned to draw specific objects

ARCHITECTURE

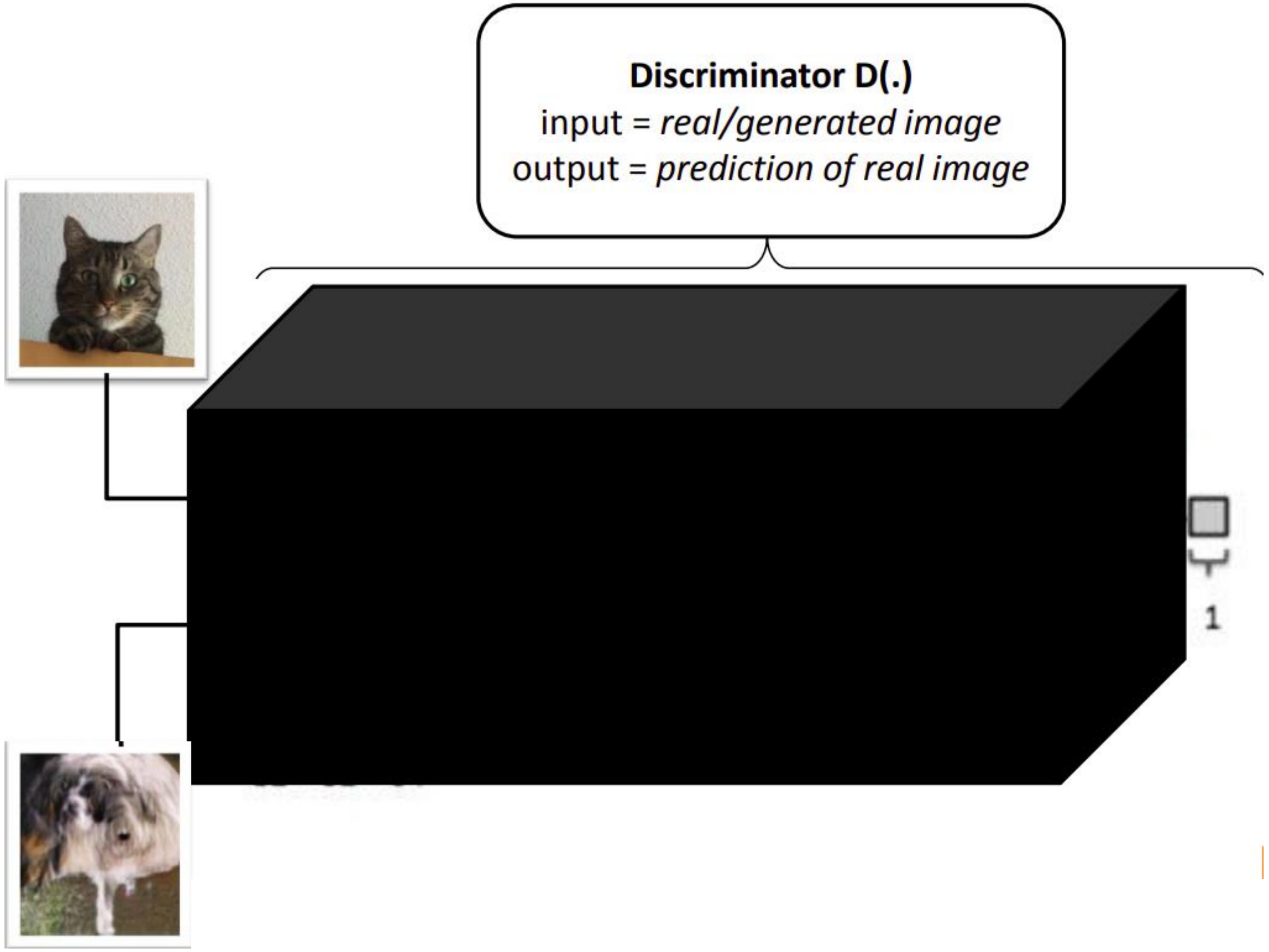


Generator G(.)
input = *random numbers*
output = *generated image*

Uniform noise vector (random numbers, z = a 100-dimensional vector from a uniform distribution) ➔ z is the distribution that creates new images!

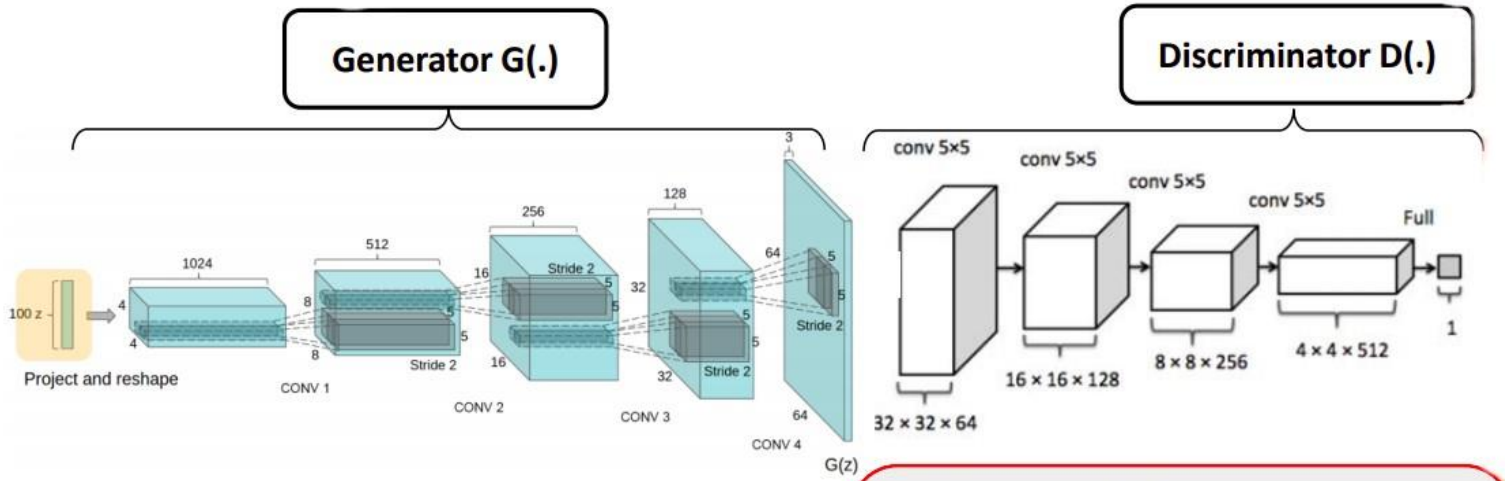
Generated image *G(z):*

ARCHITECTURE



Discriminator D(.)
input = *real/generated image*
output = *prediction of real image*

ARCHITECTURE



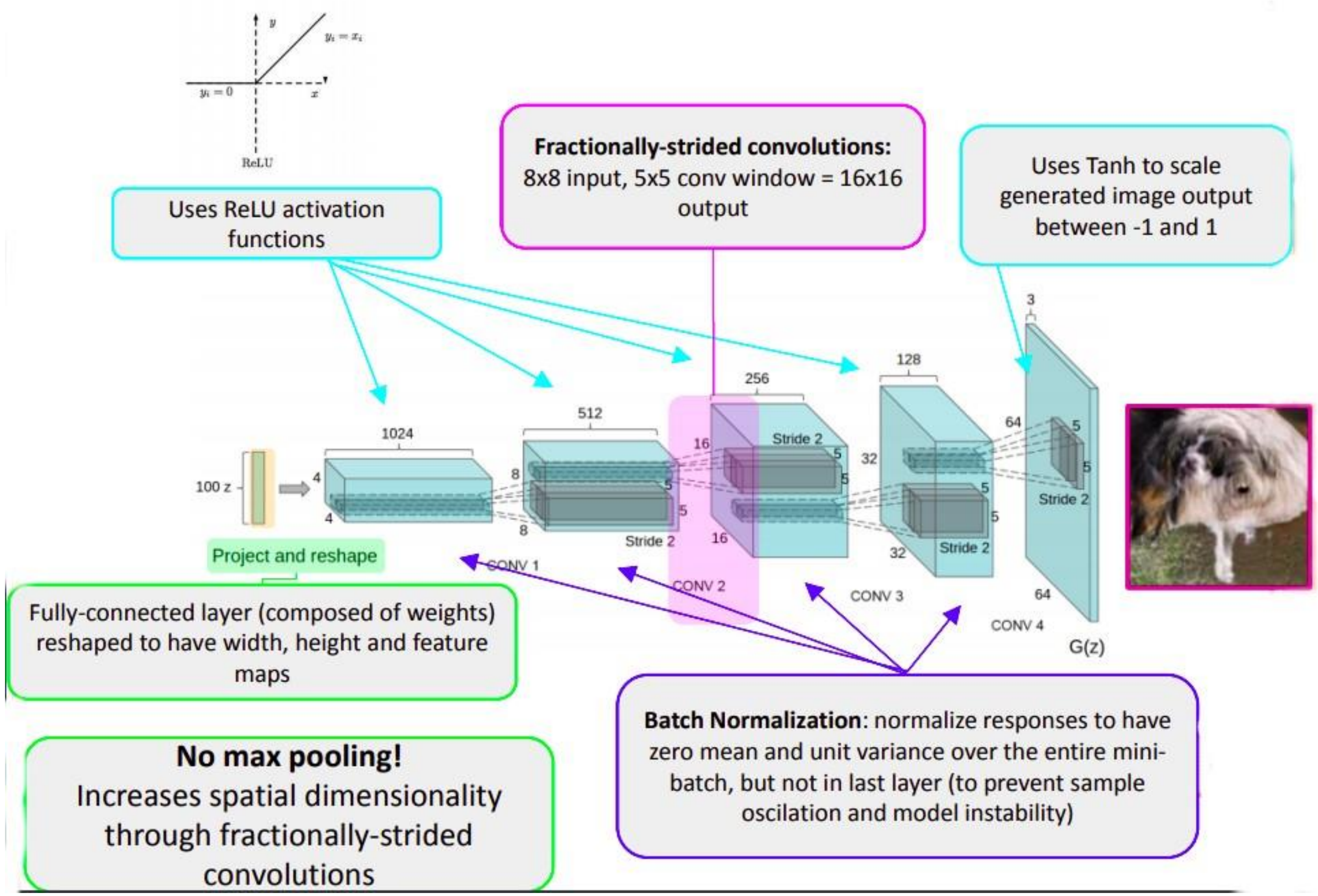
Generator Goal: Fool *D(G(z))* i.e., generate an image *G(z)* such that *D(G(z))* is wrong, i.e., *D(G(z)) = 1*

Discriminator Goal: discriminate between **real** and **generated** images *i.e., D(x)=1,* where *x* is a **real** image *D(G(z))=0,* where *G(z)* is a **generated** image.

- **Conflicting** goals.
- Both goals are **unsupervised.**
- Optimal when D(.)=0.5 (i.e., cannot tell the difference between real and generated images) and G(z)=learns the training images distribution.

ARCHITECTURE

Generator



Uses ReLU activation functions

Fractionally-strided convolutions: 8x8 input, 5x5 conv window = 16x16 output

Uses Tanh to scale generated image output between -1 and 1

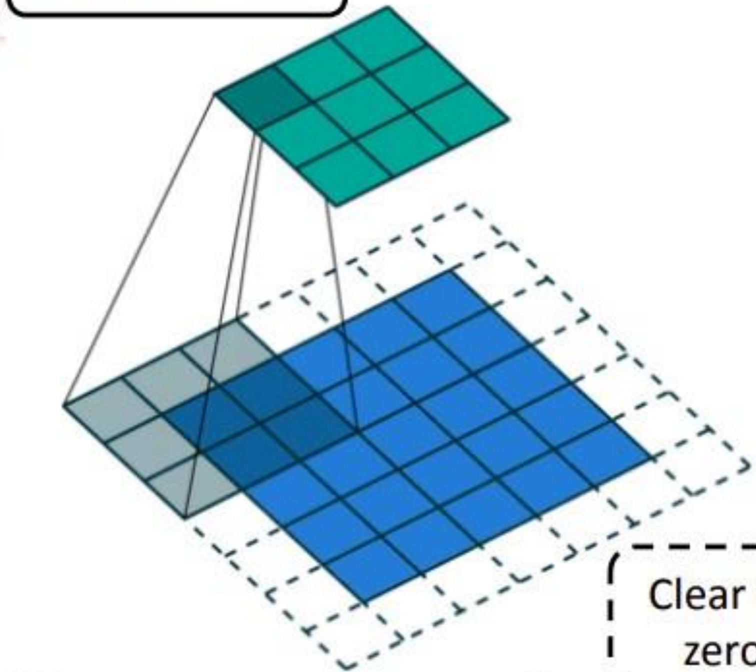
Fully-connected layer (composed of weights) reshaped to have width, height and feature maps

No max pooling! Increases spatial dimensionality through fractionally-strided convolutions

Batch Normalization: normalize responses to have zero mean and unit variance over the entire mini-batch, but not in last layer (to prevent sample oscilation and model instability)

Regular convolution

Output = 3x3



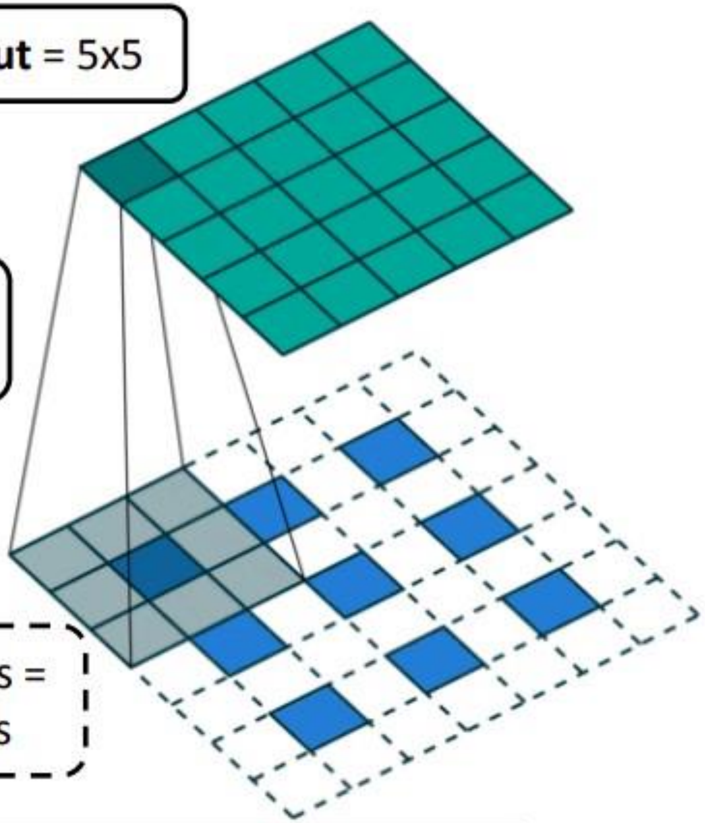
Input = 5x5
with zero-padding at
border = 6x6
(stride=2)

Filter
size=3x3

Clear dashed squares =
zero-padded inputs

Fractionally-strided convolution

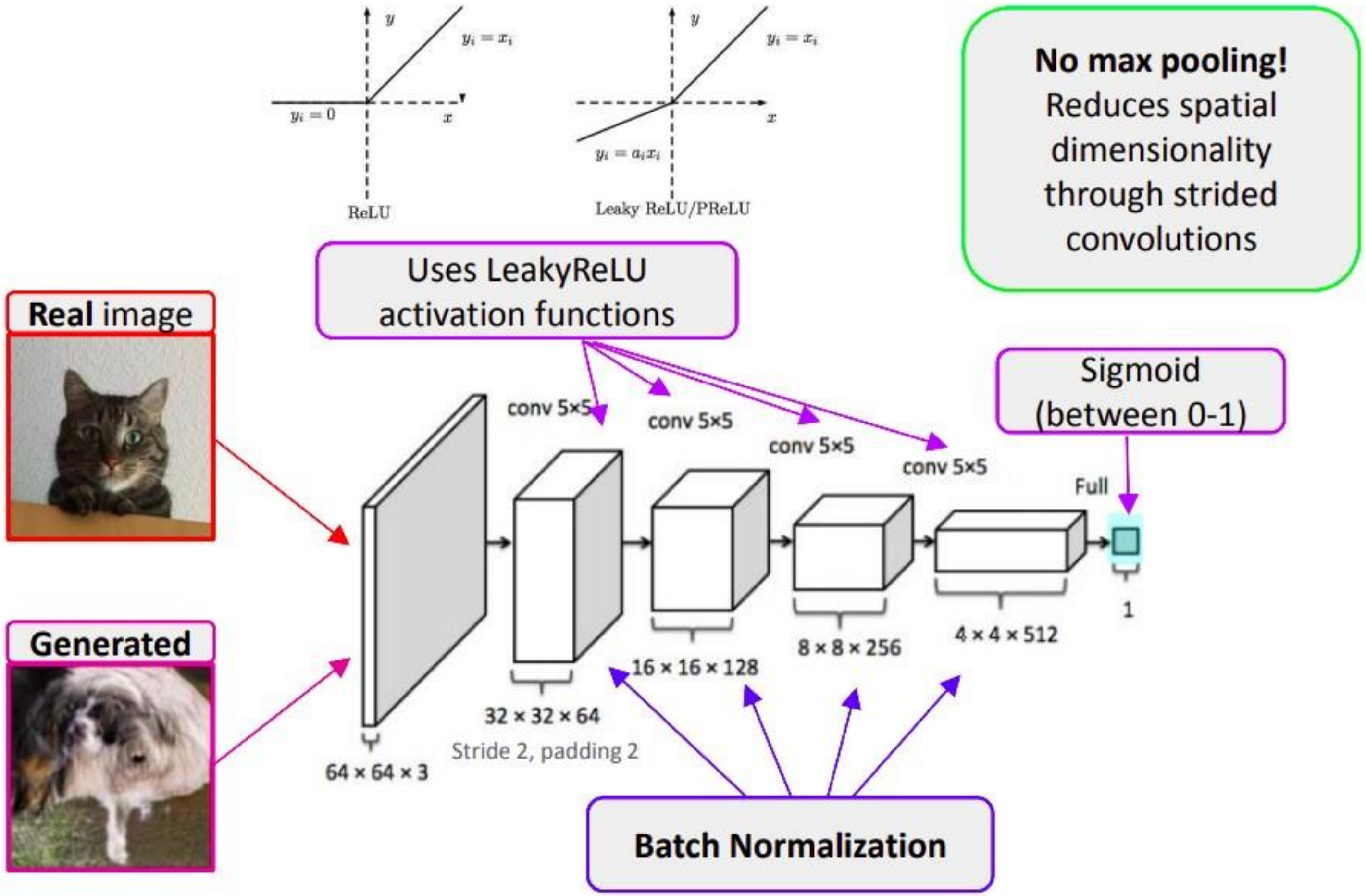
Output = 5x5



Input = 3x3
Interlace zero-padding with
inputs = 7x7
(stride=1)

ARCHITECTURE

Discriminator



Real image

Generated

Uses LeakyReLU activation functions

No max pooling! Reduces spatial dimensionality through strided convolutions

Sigmoid (between 0-1)

Batch Normalization

ARCHITECTURE GUIDELINES FOR STABLE DEEP CONVOLUTIONAL GANS

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Use ReLU activation in generator
- Use LeakyReLU activation in the discriminator.

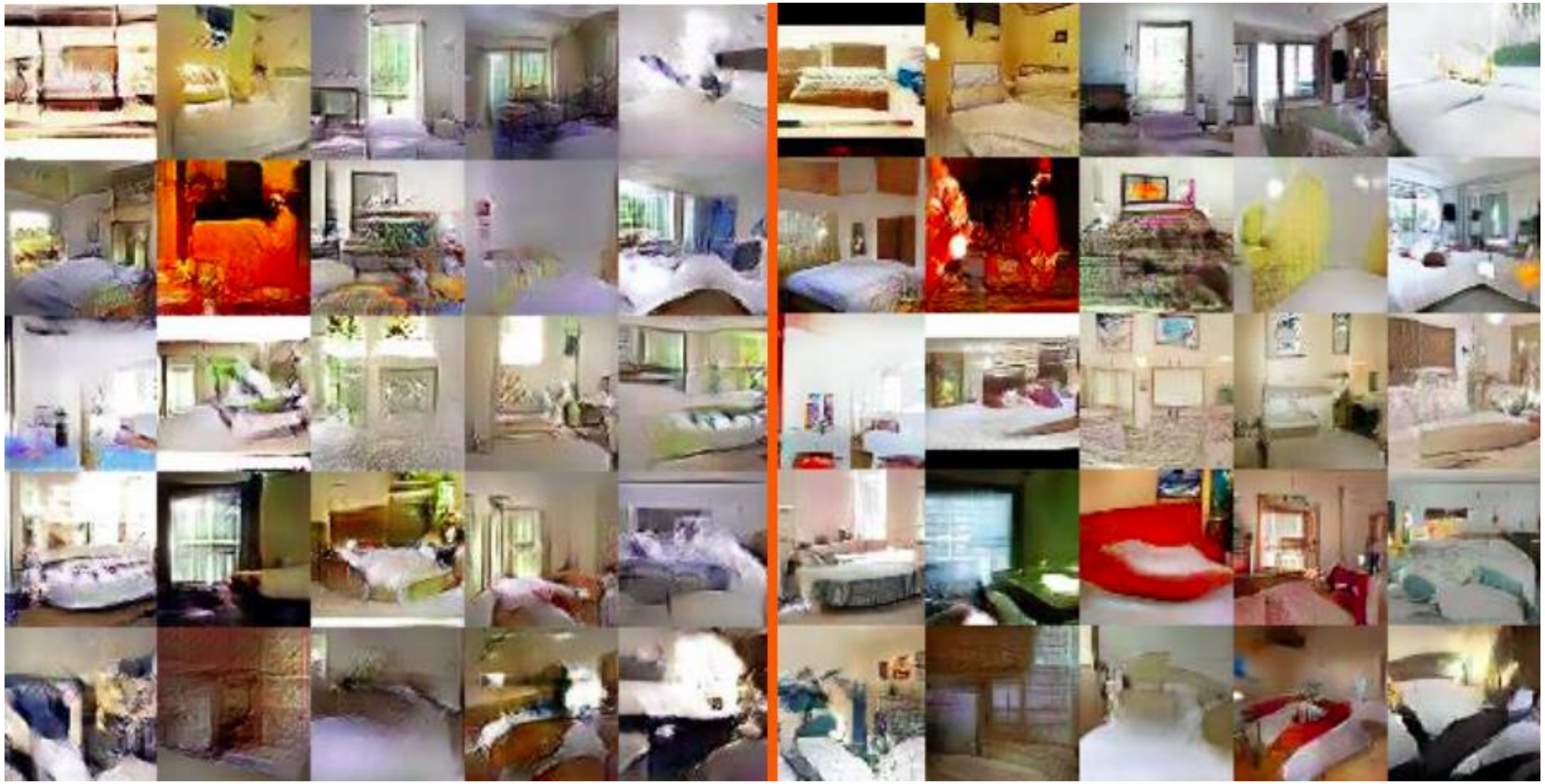
Datasets

- Large-scale Scene Understanding (LSUN)
- Faces
- ImageNet-1K

TRAINING

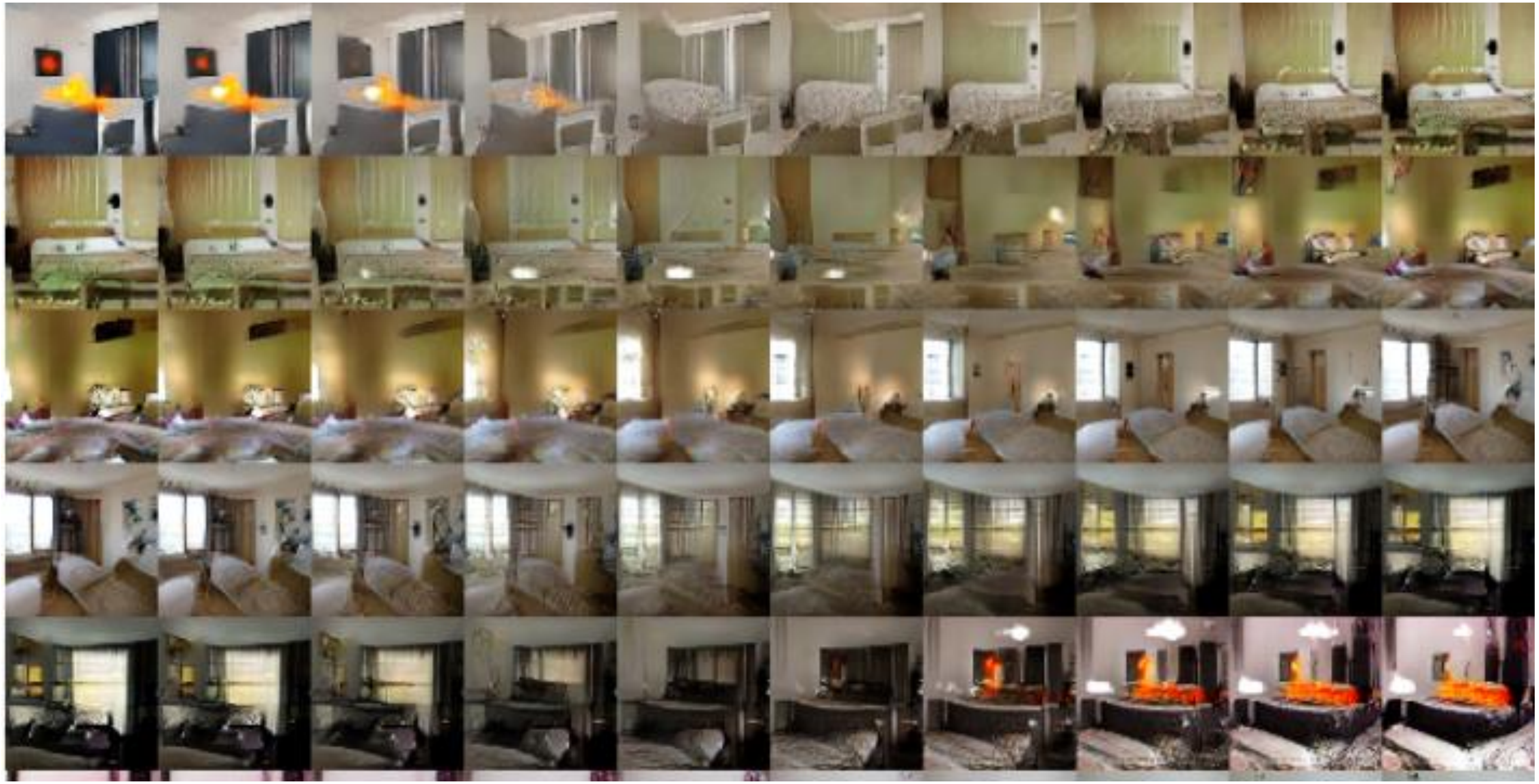
- Pre-processing: scale images between -1 and 1 (tanh range).
- Minibatch SGD (m = 128).
- Weight init.: zero-centered normal distribution (std. dev. = 0.02).
- Leaky ReLU slope = 0.2.
- Adam optimizer with tuned hyperparameters to accelerate training.
- Learning rate = 0.0002.
- Momentum term β1 = 0.5 to stabilize training

Results

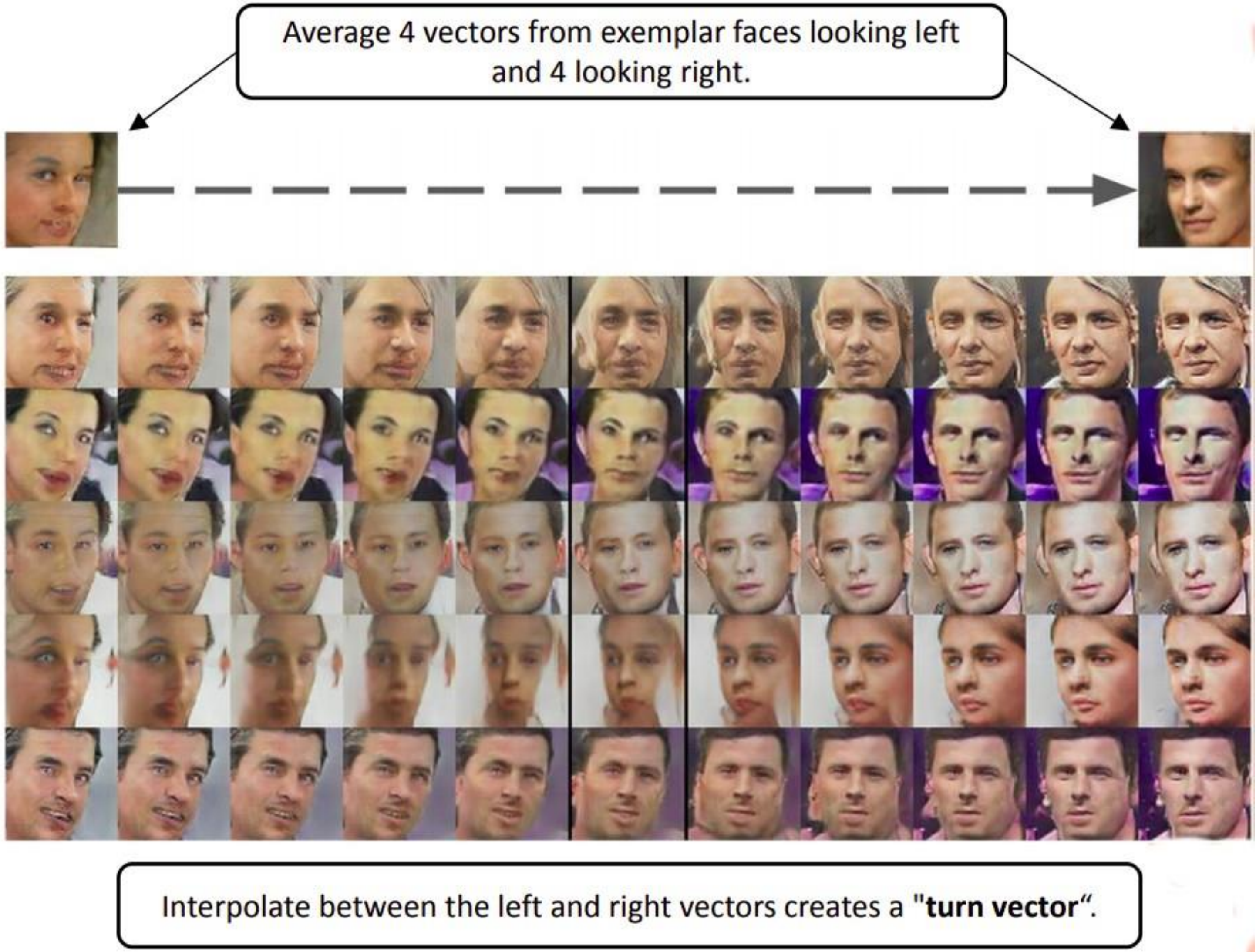


Results

INTERPOLATION BETWEEN A SERIES OF RANDOM POINTS IN Z



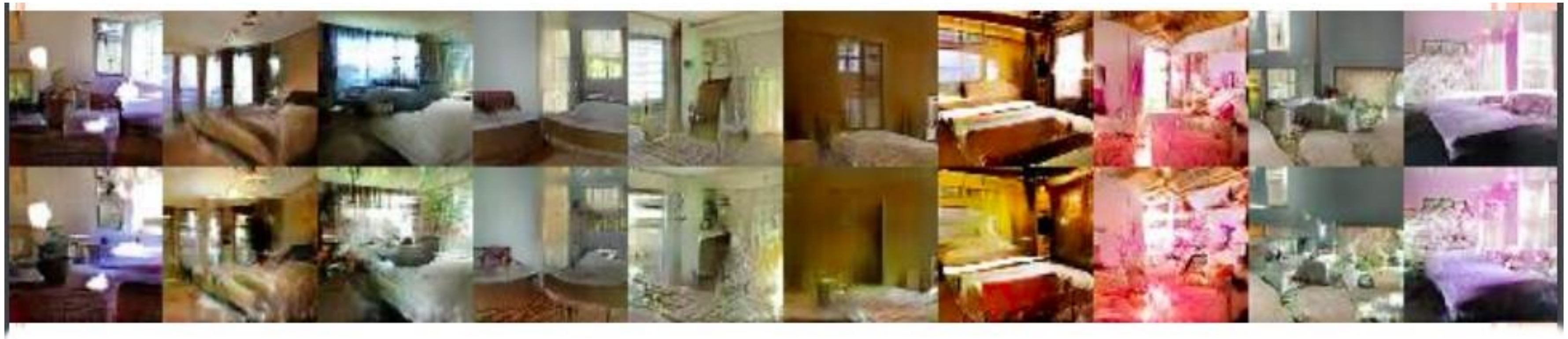
Results



Average 4 vectors from exemplar faces looking left and 4 looking right.

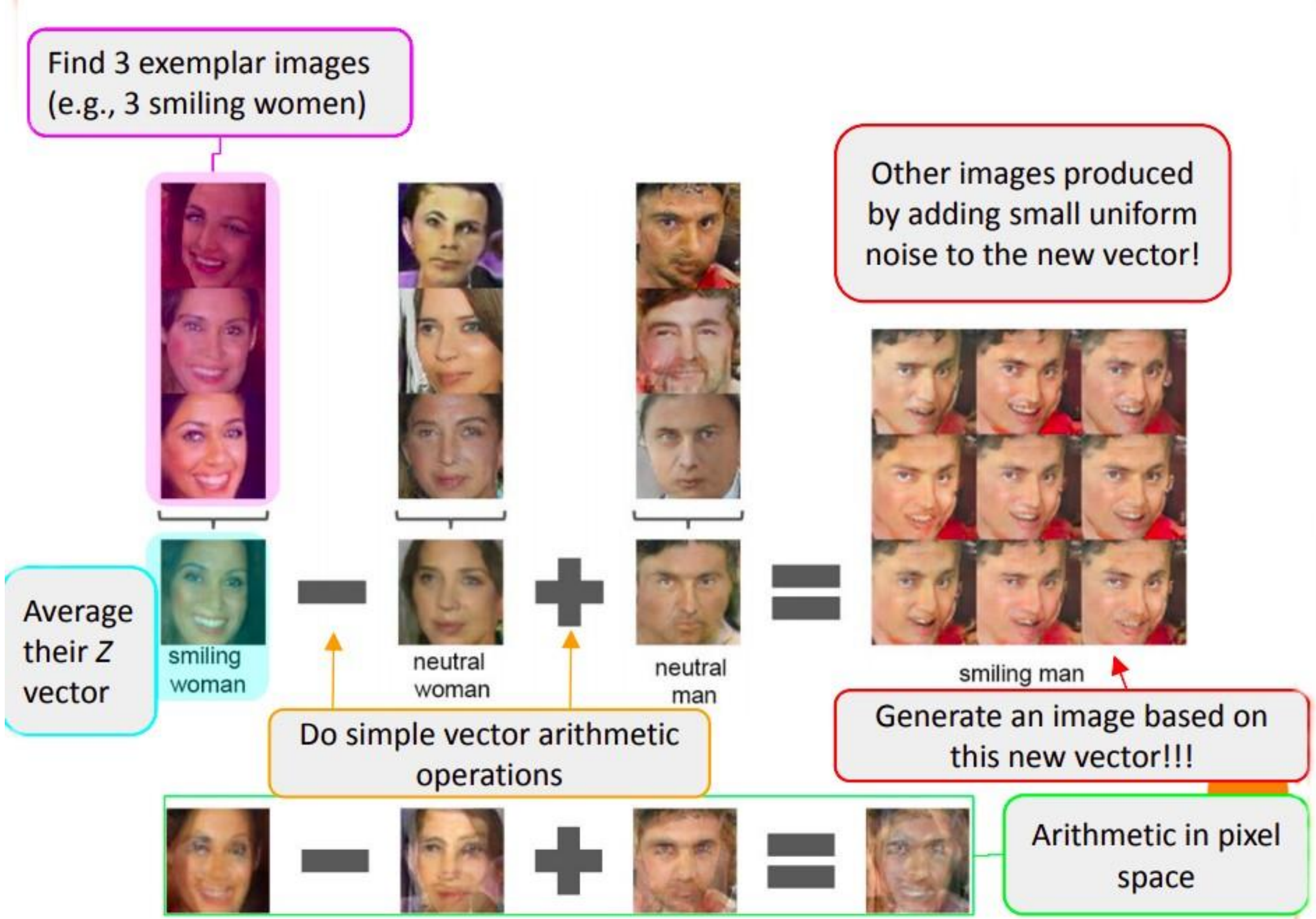
Interpolate between the left and right vectors creates a "**turn vector**“.

Results



(Bottom) Samples generated after dropping out "window" concept.
Some windows are removed or transformed.
The overall scene stays the same, indicating the generator has separated objects (windows) from the scene.

Results



Find 3 exemplar images (e.g., 3 smiling women)
Other images produced by adding small uniform noise to the new vector!
Average their Z vector
smiling woman
neutral woman
neutral man
smiling man
Do simple vector arithmetic operations
Generate an image based on this new vector!!!
Arithmetic in pixel space