Realistic Image Generation by Adversarial Training of Deep Networks

Alec Radford, Luke Metz, and Soumith Chintala
https://github.com/Newmu/dcgan_code

Presented by Matt Hardwick

Unsupervised Representation Learning

with Deep Convolutional Generative Adversarial Networks

Alec Radford, Luke Metz, and Soumith Chintala
https://github.com/Newmu/dcgan_code

Presented by Matt Hardwick

Computer Vision

In general, dealing with the extraction of high-dimensional information from real world imagery in order to produce symbolic knowledge (e.g., in the form of features/decisions)

Computer Vision & ML

Building models that are capable of being trained to disentangle high-dimensional information from real world imagery.

Building models that are capable of being trained to use that information as symbolic knowledge for decision making.

Computer Vision & ML 2

Building models that are capable of being trained to generate imagery from symbolic knowledge.

Building models that are capable of discriminating an input image as either real-world or generated based on features.

Unsupervised Representation Learning

with Deep Convolutional Generative Adversarial Networks

Disentangling a small amount of high-dimensional information from a large amount of low-dimensional data. Without Labels!

Feature Learning by Humans

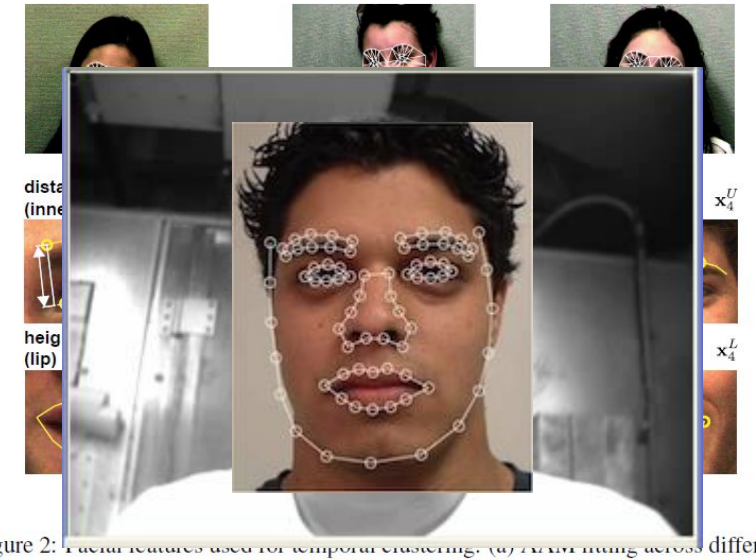
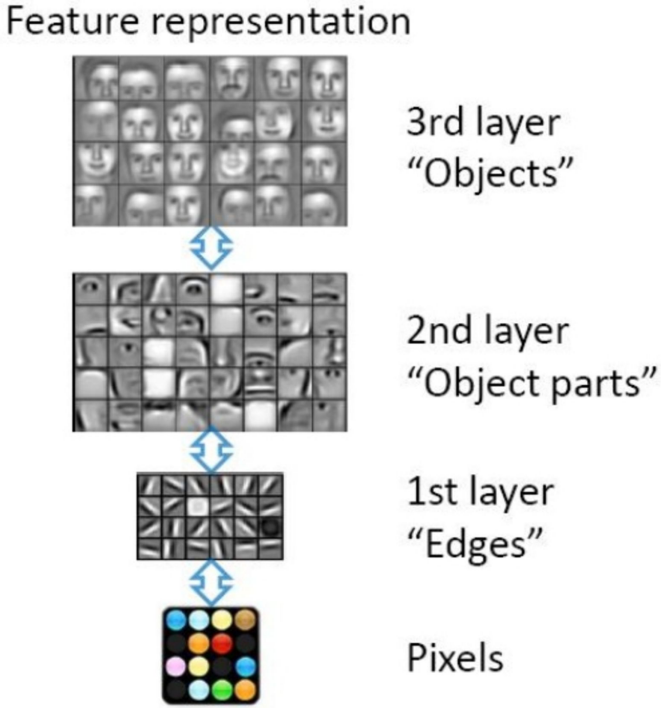


Figure 2: Facial features used for temporal clustering. (a) AAM fitting across different subjects. (b) Eight different features extracted from distance between tracked points, height of facial parts, angles for mouth corners, and appearance patches.

Humans intuitively extract high-dimensional features.

Humans can generate statistical models to extract certain features.

Unsupervised Repr. Learning

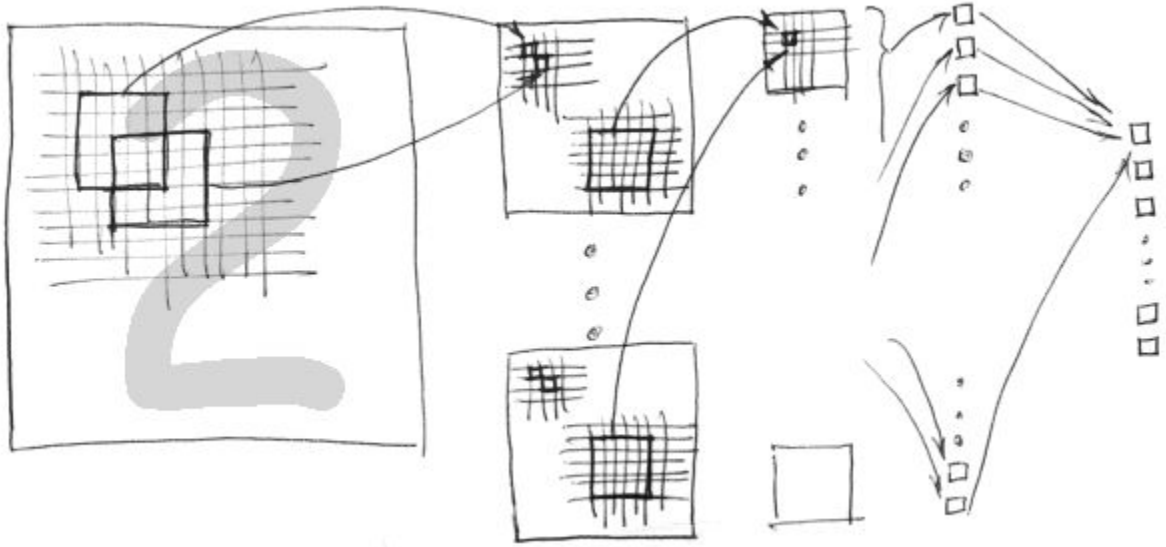


Feature Repr. as output

Pixel volume as input

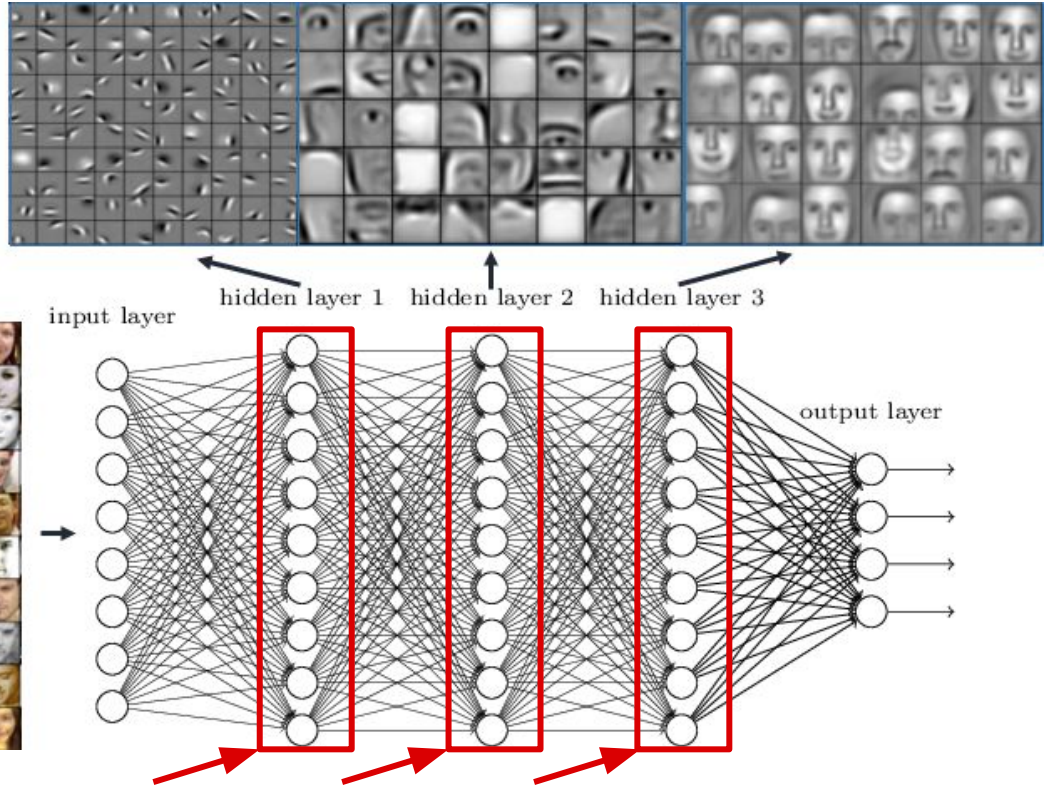
Width x Height x 3 Colors

Unsupervised Repr. Learning 2



Unsupervised Repr. Learning 3

Deep neural networks learn hierarchical feature representations



Unsupervised Representation Learning

Training with two learners that have competing objectives

with Deep Convolutional Generative Adversarial Networks

Architectures of Convolutional Networks that are stable to train in most settings. Main Contribution!

DCGANs

Deep - Many Processing Layers

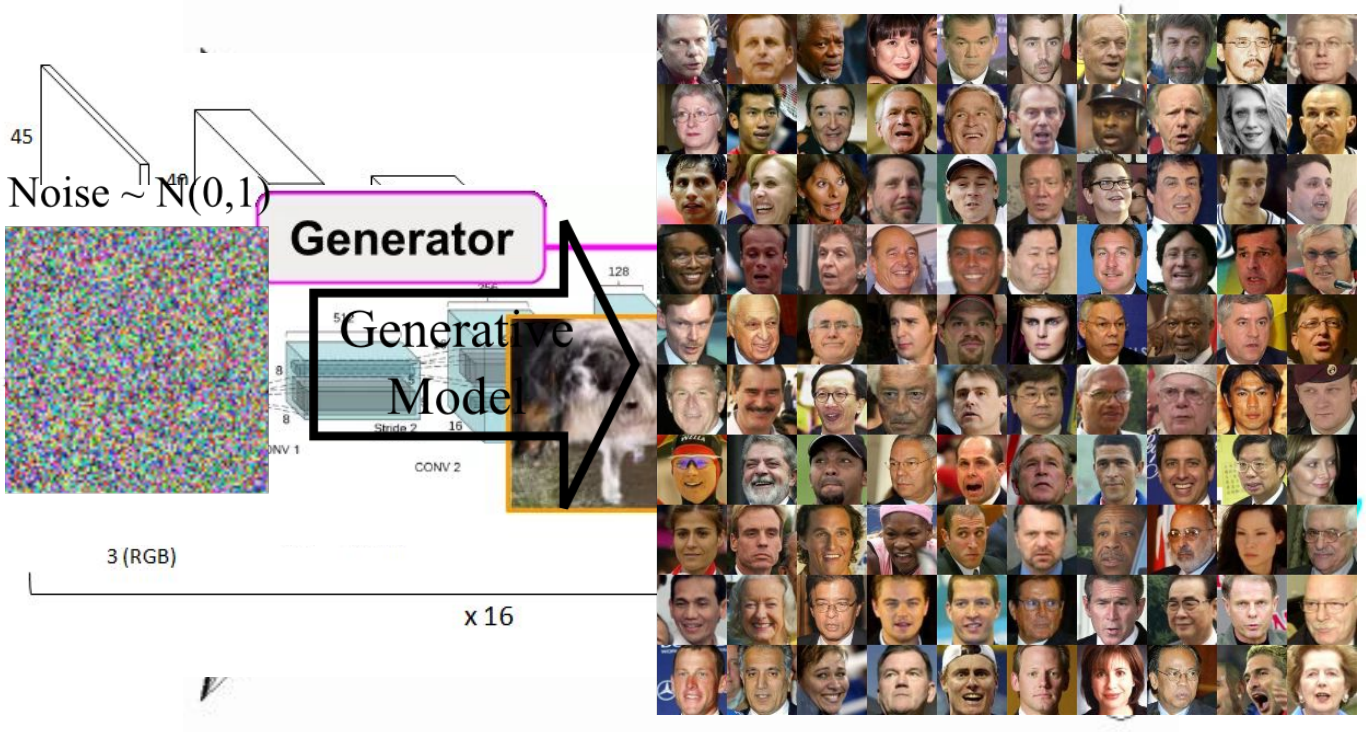
Convolutional - Inputs to each layer overlap

Generative - Output is human-readable

Adversarial - Trained by competition

Networks - It’s a neural network

DCGANs 2


45
Noise ~ N(0,1)
Generator
Generative Model
128
8
8
16
Stride 2
CONV 1
CONV 2
3 (RGB)
x 16

DCGAN Generator Model

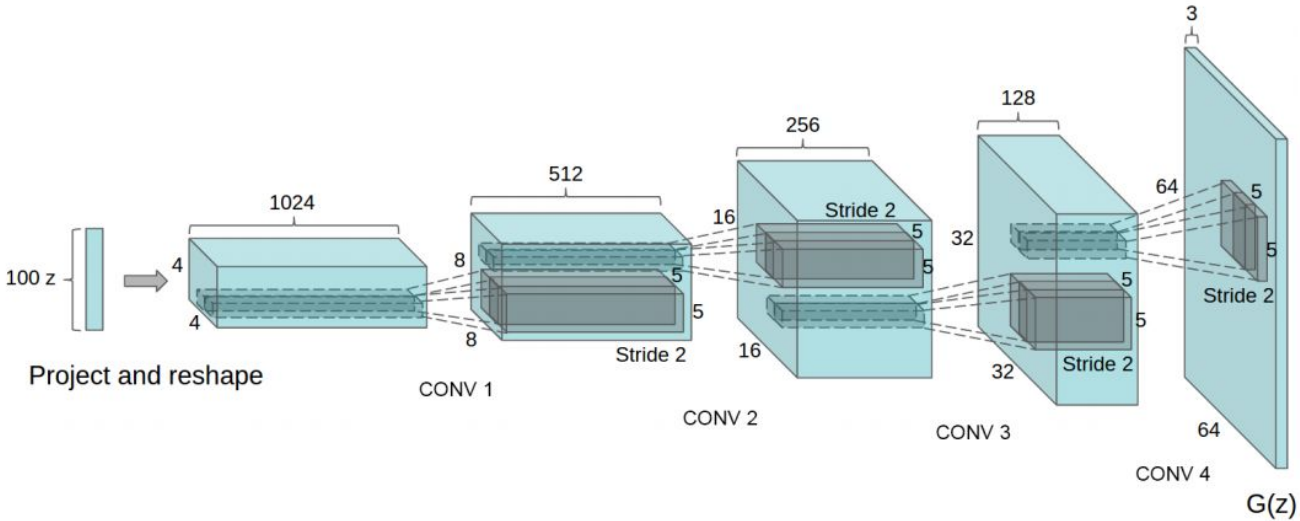


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

Architecture Guidelines for DCGANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

These things make them Stable

2 Models

Generator - A model that learns how to take an input vector and generate a realistic image (Really Cool)

Discriminator - A model that learns how to discern whether an input image is real or fake (Still “Cool”, but more theoretically interesting)

Figure 2: Generated bedrooms after one training pass through the dataset. Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization with SGD and a small learning rate.

Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

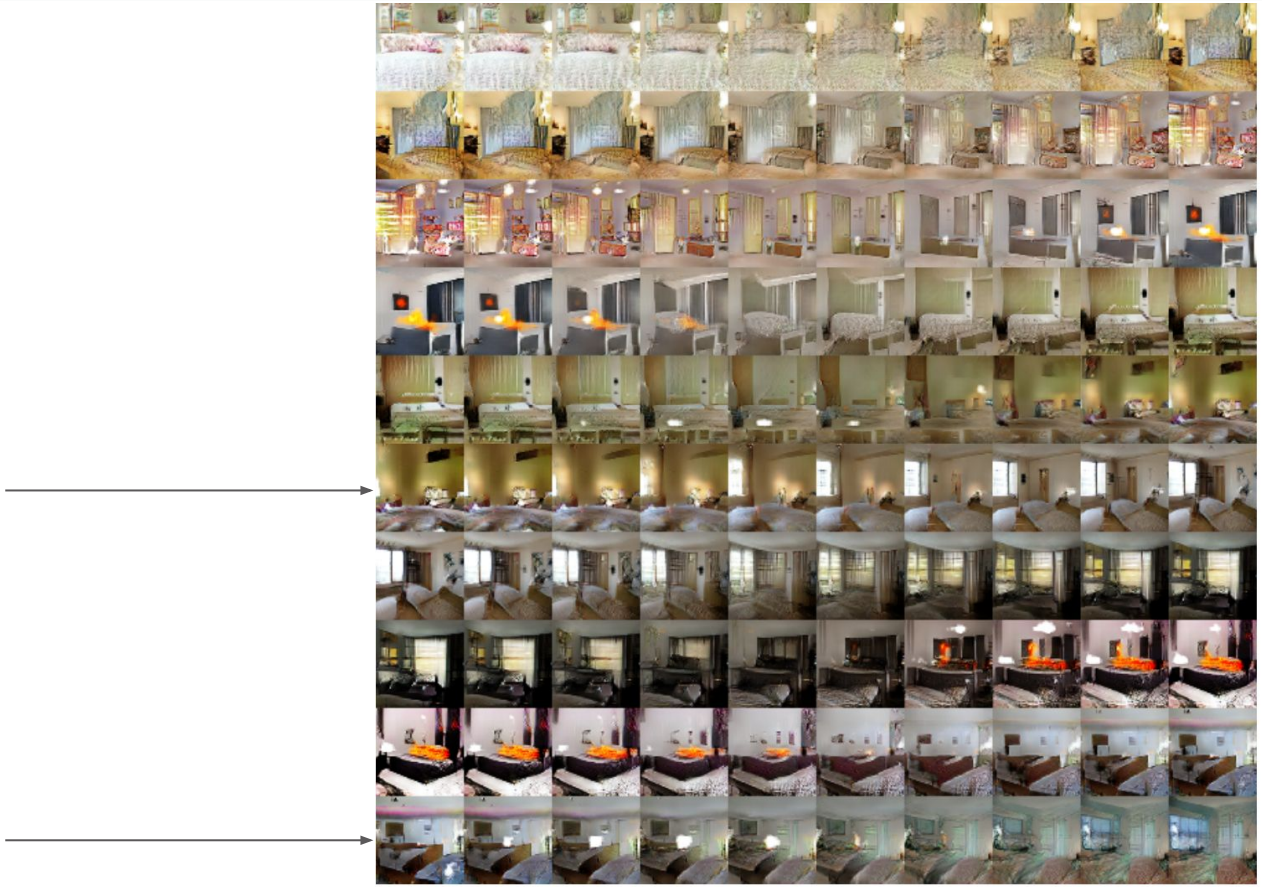


Figure 4: Top rows: Interpolation between a series of 9 random points in Z show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.

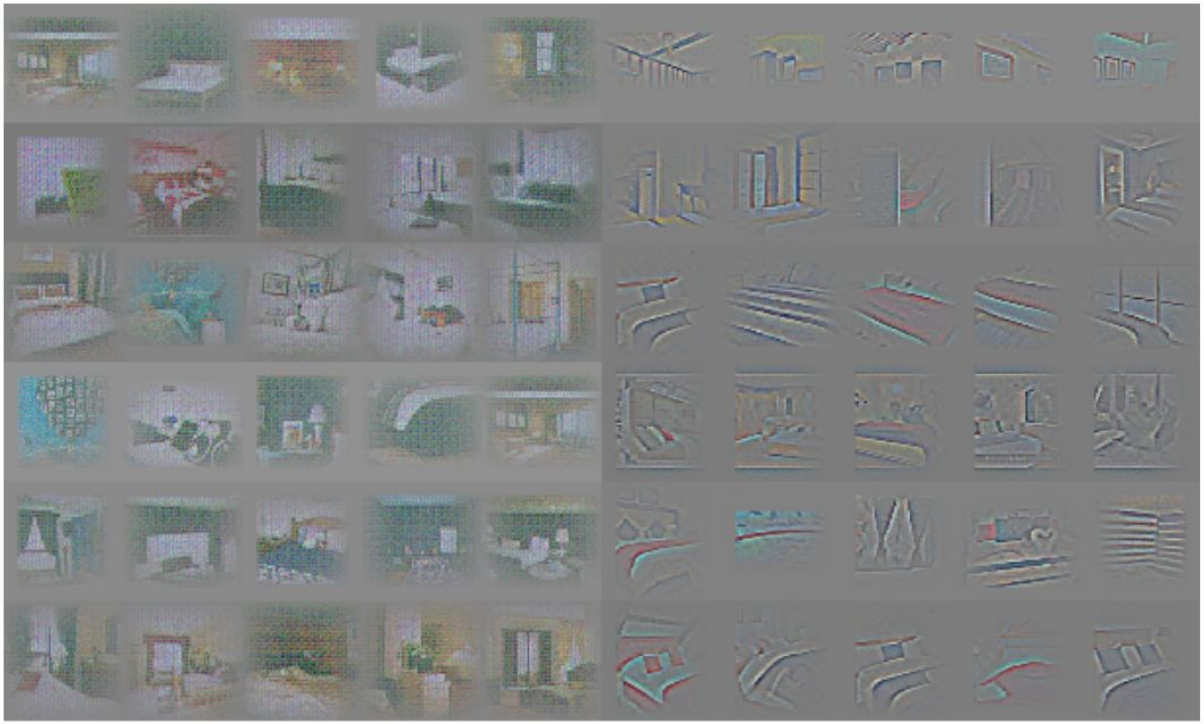


Figure 5: On the right, guided backpropagation visualizations of maximal axis-aligned responses for the first 6 learned convolutional features from the last convolution layer in the discriminator. Notice a significant minority of features respond to beds - the central object in the LSUN bedrooms dataset. On the left is a random filter baseline. Comparing to the previous responses there is little to no discrimination and random structure.

Figure 6: Top row: un-modified samples from model. Bottom row: the same samples generated with dropping out "window" filters. Some windows are removed, others are transformed into objects with similar visual appearance such as doors and mirrors. Although visual quality decreased, overall scene composition stayed similar, suggesting the generator has done a good job disentangling scene representation from object representation. Extended experiments could be done to remove other objects from the image and modify the objects the generator draws.

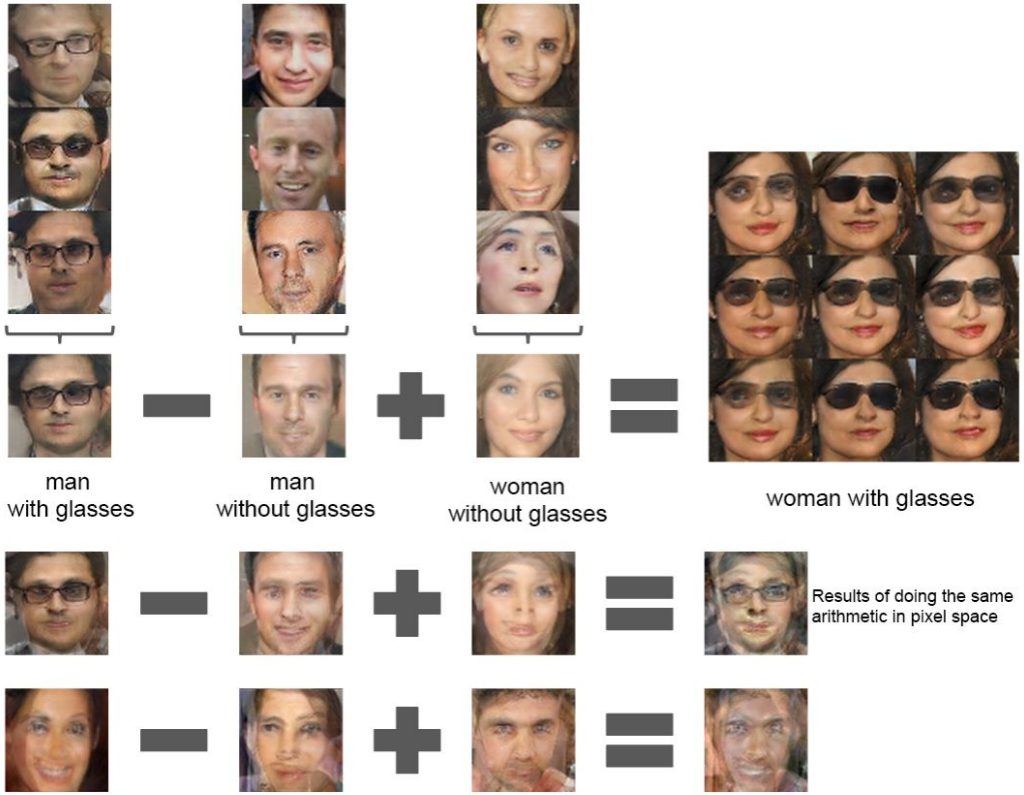
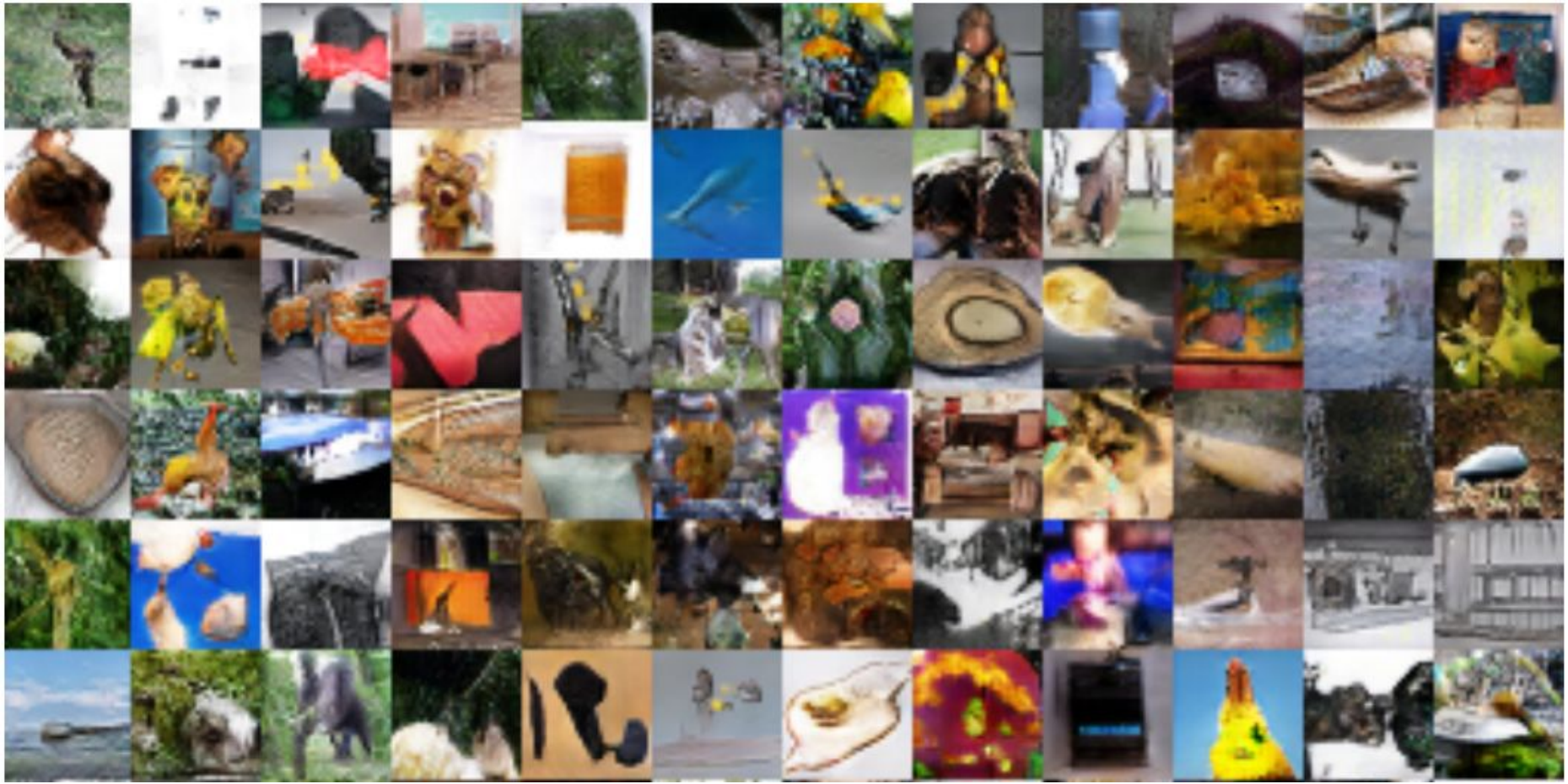


Figure 7: Vector arithmetic for visual concepts. For each column, the Z vectors of samples are averaged. Arithmetic was then performed on the mean vectors creating a new vector Y. The center sample on the right hand side is produce by feeding Y as input to the generator. To demonstrate the interpolation capabilities of the generator, uniform noise sampled with scale +-0.25 was added to Y to produce the 8 other samples. Applying arithmetic in the input space (bottom two examples) results in noisy overlap due to misalignment.


smiling woman
neutral woman
neutral man
smiling man

Figure 8: A ”turn” vector was created from four averaged samples of faces looking left vs looking right. By adding interpolations along this axis to random samples we were able to reliably transform their pose.

Figure 9: Side-by-side illustration of (from left-to-right) the MNIST dataset, generations from a baseline GAN, and generations from our DCGAN .



https://twitter.com/AlecRad/status/775268395348402176

Alec Radford Retweeted

The Practical Dev @ThePracticalDev · 5 Apr 2016

We love what we do, even if we spend half our time scattering print statements across our code and hitting refresh

26 1.3K 1.3K