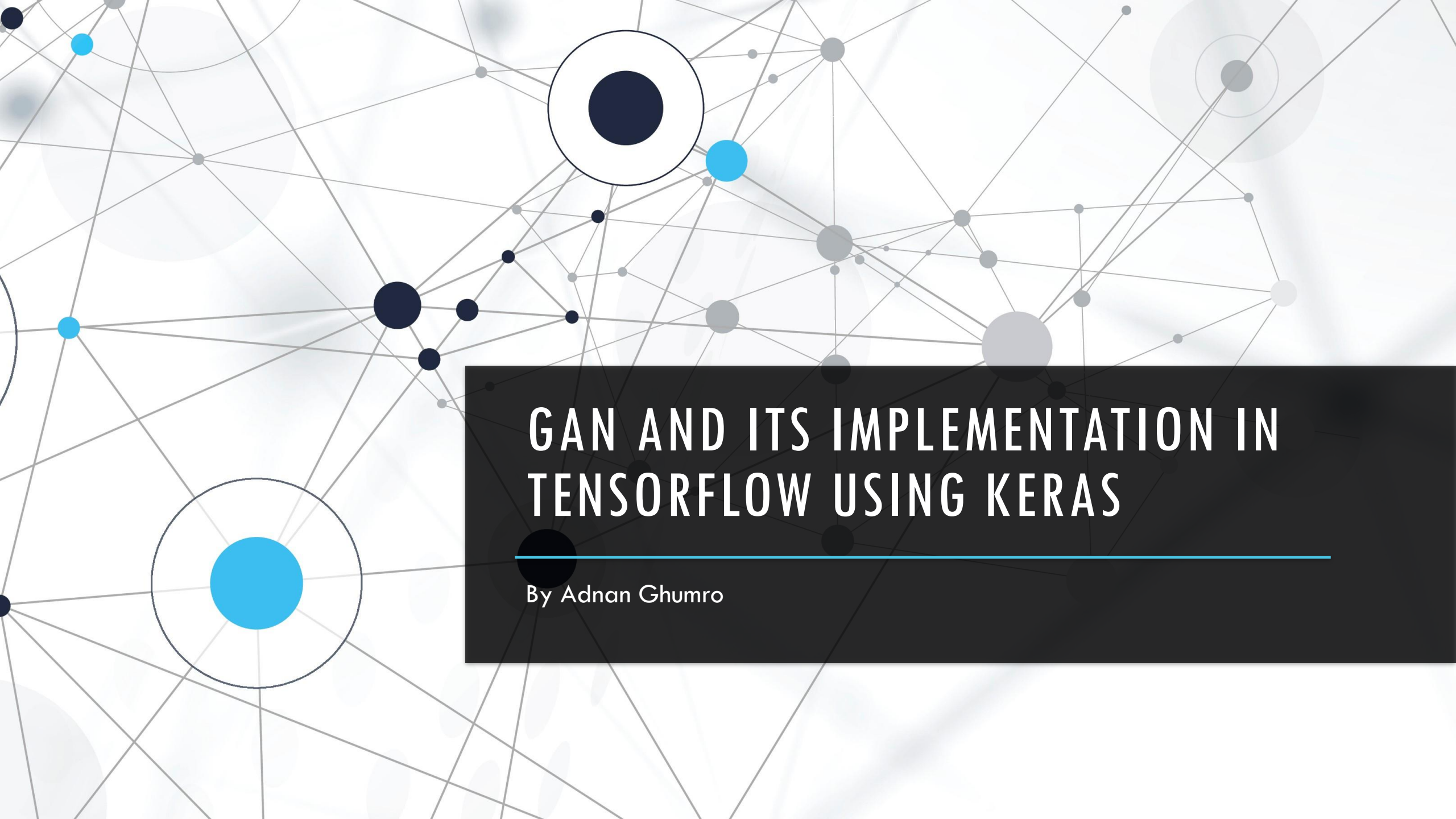


GAN AND ITS IMPLEMENTATION IN TENSORFLOW USING KERAS

By Adnan Ghumro

CONTENTS


Introduction to generative adversarial networks (GAN).


GAN Implementation.


Tricks for implementation


The generator


The discriminator

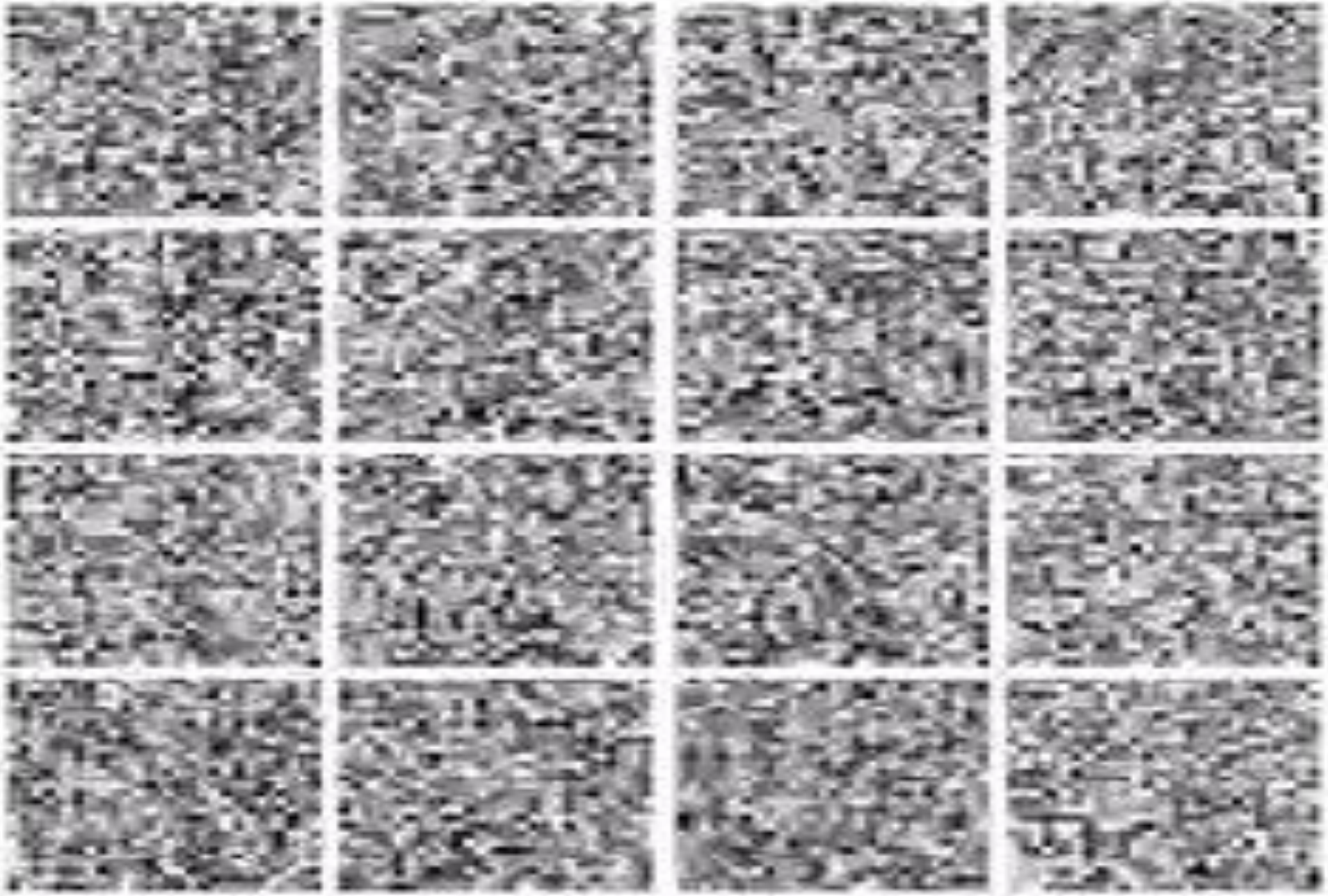

The adversarial network


How to train your DCGAN

INTRODUCTION TO GANs

- Generative adversarial networks (GANs), introduced in 2014 by Goodfellow et al.
- They enable the generation of fairly realistic synthetic images by forcing the generated images to be statistically almost indistinguishable from real ones.



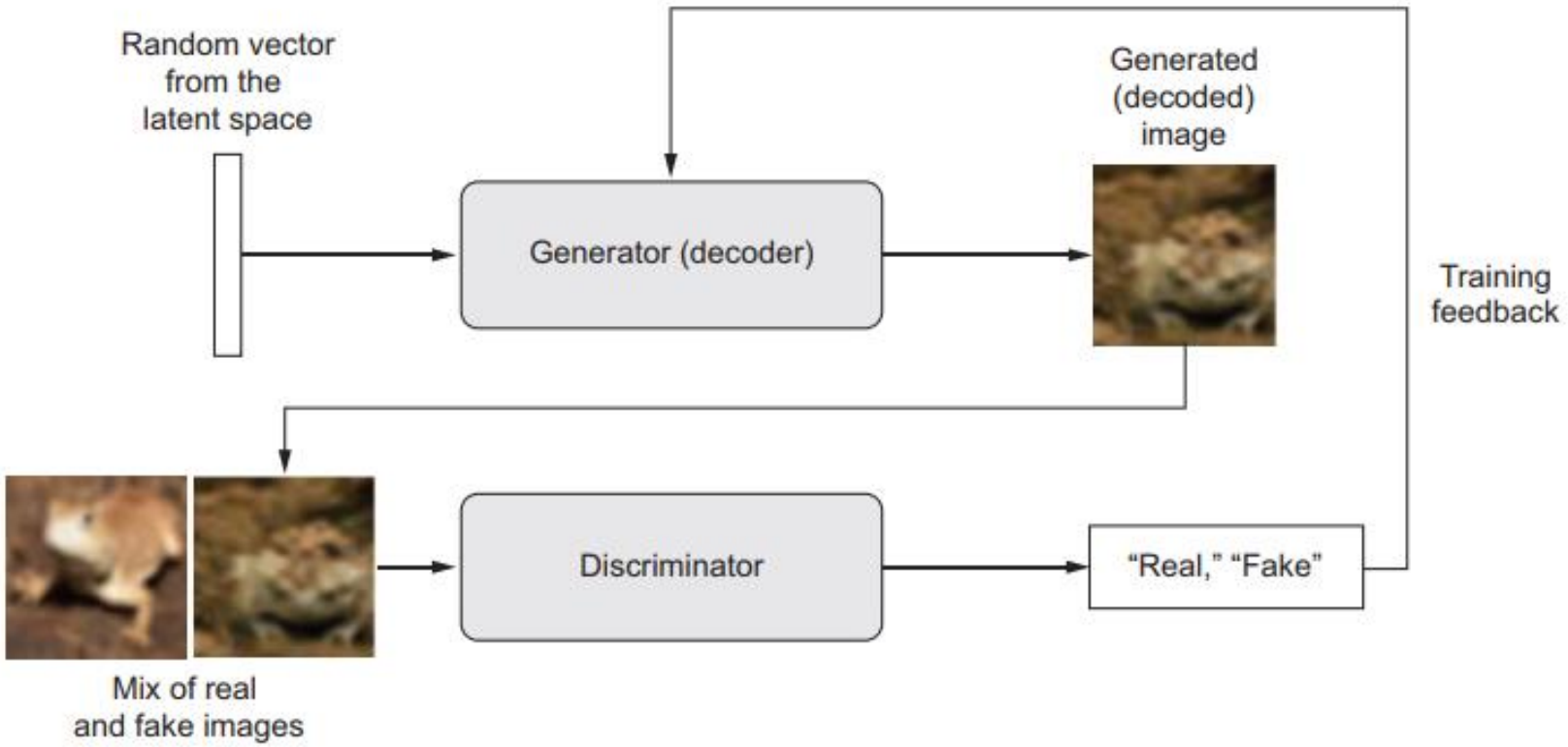


INTRODUCTION TO GANs (Cont.)

- GAN is a forger network and an expert network, each being trained to best the other. As such, a GAN is made of two parts:

  - *Generator network*—Takes as input a random vector (a random point in the latent space) and decodes it into a synthetic image.

  - *Discriminator network (or adversary)*—Takes as input an image (real or synthetic) and predicts whether the image came from the training set or was created by the generator network.



GAN ARCHITECTURE

GAN IMPLEMENTATION.

- I'll explain how to implement a *deep convolutional GAN* (DCGAN) in Keras.
- A GAN where the generator and discriminator are deep convnets. In particular, it uses a Conv2DTranspose layer for image up sampling in the generator.
- we'll train the GAN on images from CIFAR10, a dataset of 50,000 32 × 32 RGB images belonging to 10 classes (5,000 images per class). To make things easier, we'll only use images belonging to the class "frog."
- we'll train the GAN on images from Fashion-MNIST, a dataset of60,000 28 × 28 grayscale images belonging to 10 classes.

STEPS TO IMPLEMENT GAN

1. A generator network maps vectors of shape (latent dim,) to images of shape (32, 32, 3).
2. A discriminator network maps images of shape (32, 32, 3) to a binary score estimating the probability that the image is real.
3. A gan network chains the generator and the discriminator together: gan(x) =discriminator(generator(x)).
4. You train the discriminator using examples of real and fake images along with “real”/“fake” labels, just as you train any regular image-classification model.
5. To train the generator, you use the gradients of the generator’s weights with regard to the loss of the gan model.

“Train the generator to fool the discriminator”

TRICKS FOR IMPLEMENTATION

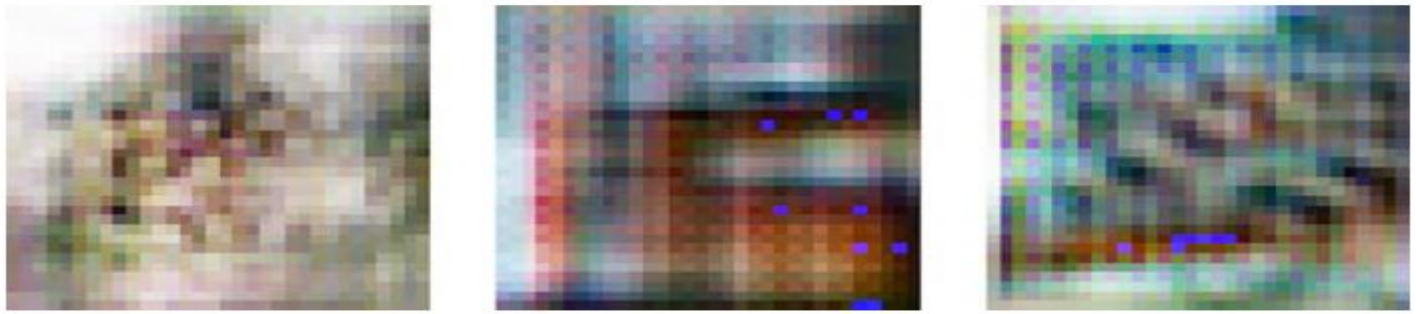
- Like most things in deep learning, it's more alchemy than science.
- These tricks are heuristics, not theory-backed guidelines.

1. We use tanh as the last activation in the generator, instead of sigmoid
2. We sample points from the latent space using a *normal distribution* (Gaussian distribution), not a uniform distribution.
3. GANs are likely to get stuck in all sorts of ways. Introducing randomness during training helps prevent this. We introduce randomness in two ways:
    1. By using dropout in the discriminator
    2. By adding random noise to the labels for the discriminator

TRICKS FOR IMPLEMENTATION (CONT.)

4. Sparse gradients can hinder GAN training. In deep learning, sparsity is often a desirable property, but not in GANs. Two things can induce gradient sparsity: **max pooling operations** and **ReLU activations**. Instead of max pooling, we recommend using **strided convolutions** for down sampling, and we recommend using a **LeakyReLU layer** instead of a ReLU activation. It's similar to ReLU, but it relaxes sparsity constraints by allowing small negative activation values.

5. In generated images, it's common to see checkerboard artifacts caused by unequal coverage of the pixel space in the generator as shown in figure. To fix this, we use a kernel size that's divisible by the stride size whenever we use a strided Conv2DTranpose or Conv2D in both the generator and the discriminator



IT'S TIME TO IMPLEMENT GAN USING ALL THESE TRICKS.

Available tools to train DNN

1. Juypter Notebook
2. Kaggle
3. Google Colab