Practical Deep learning for Computer Vision

Dr Kien Nguyen
Email: k.nguyenthanh@qut.edu.au
Queensland University of Technology, Australia

Warning

Deep learning is new SEXY (for a reason!)

Week 3: Object Recognition

- Resnet from scratch
- Transfer Learning
- Google Colab

Week 4: Object Detection

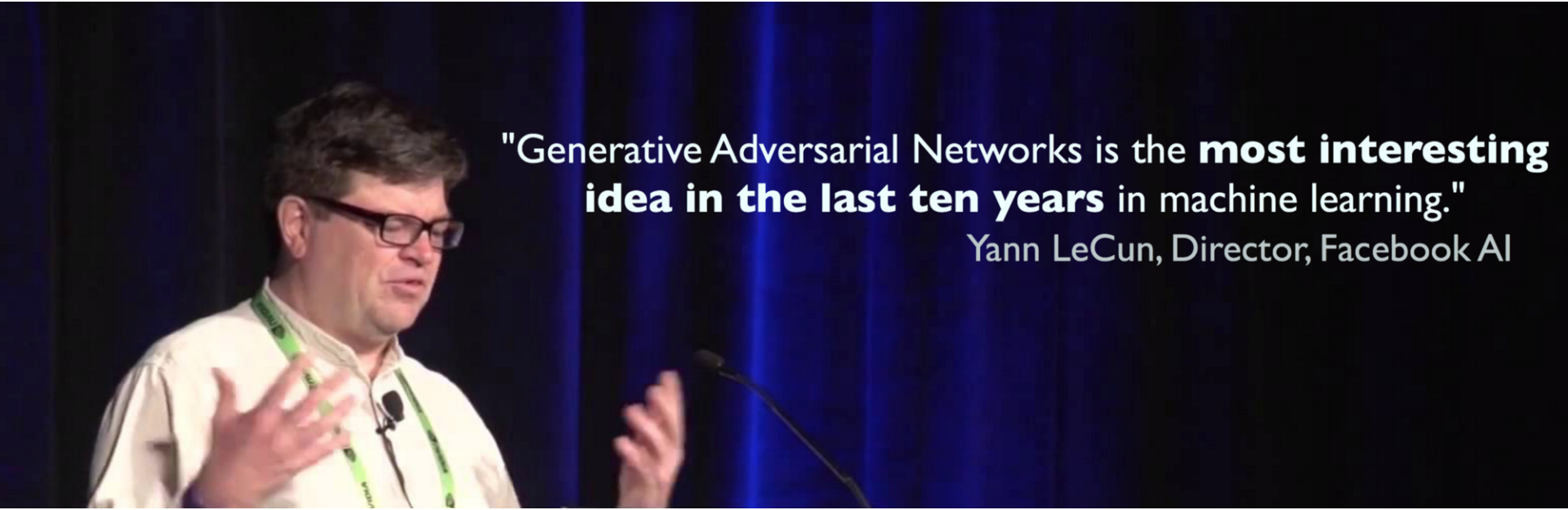
- DarkNet & YOLO
- Transfer Learning
- AWS

Week 5: GAN

- GANs

Week 5. Generative Adversarial Models - GANs

Why?



- Very cool applications

Create Anime characters

Towards the automatic Anime characters creation with Generative Adversarial Networks

Pose-guided Person Image Generation

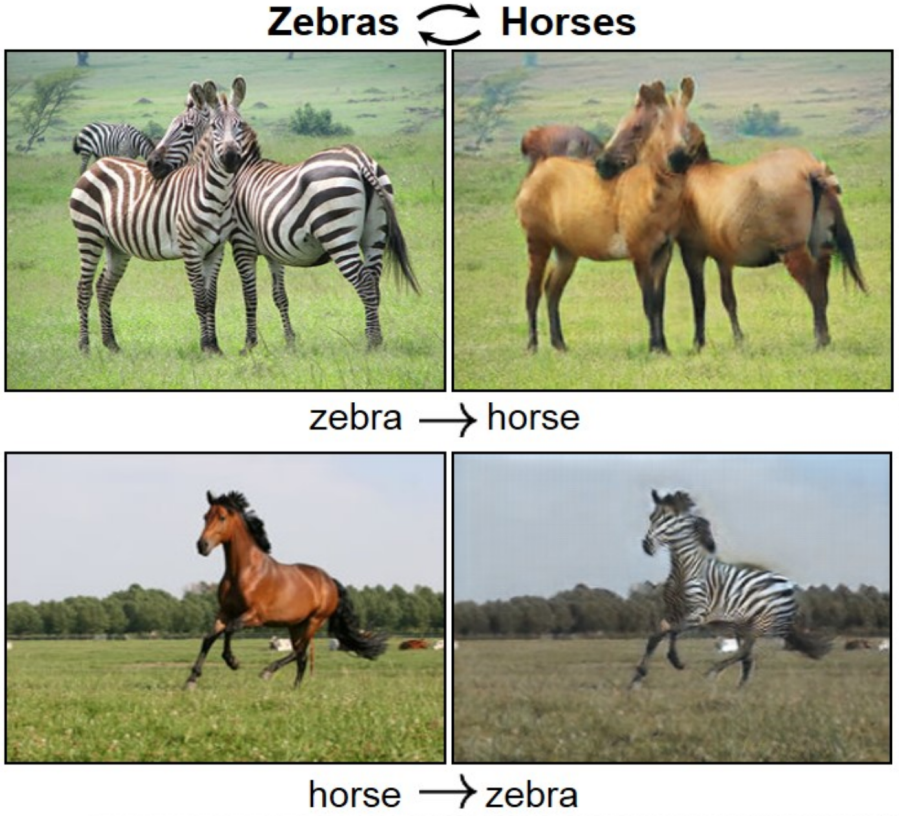
Ground truth

Generated

https://papers.nips.cc/paper/6644-pose-guided-person-image-generation.pdf

CycleGAN

CycleGAN



PixelDTGAN








Super resolution

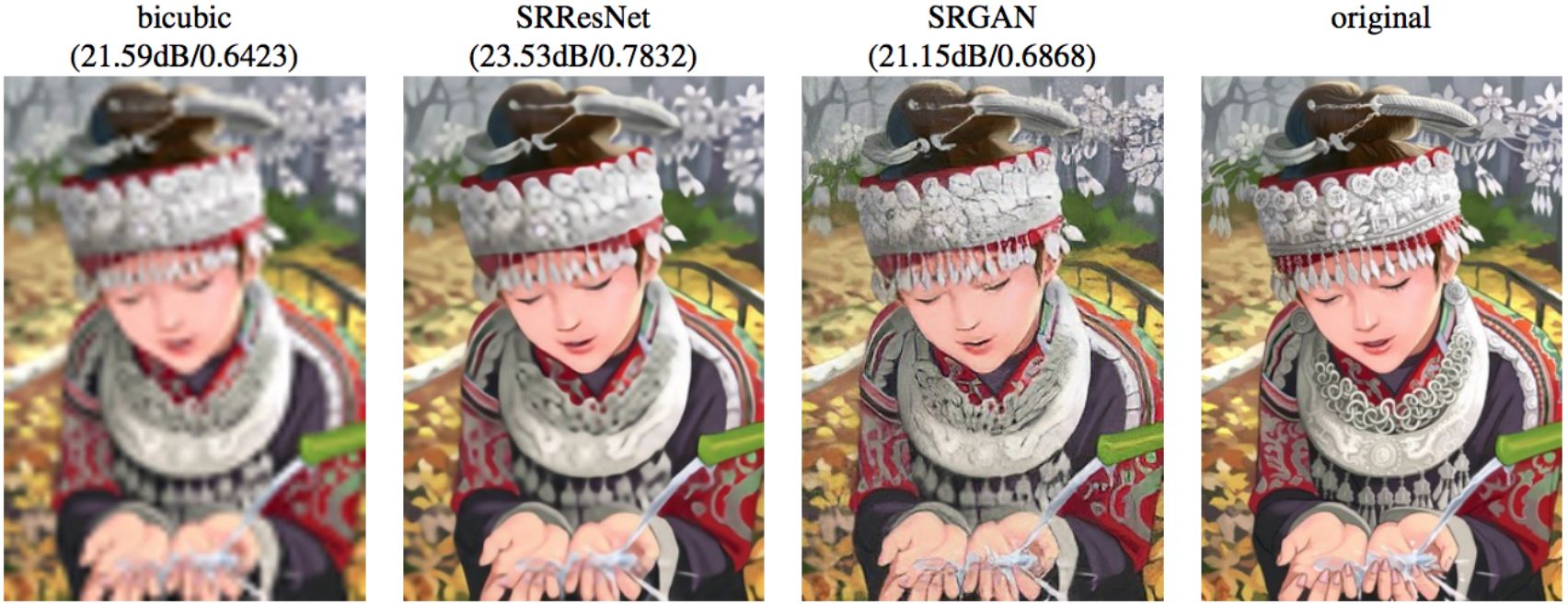


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Progressive GAN

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

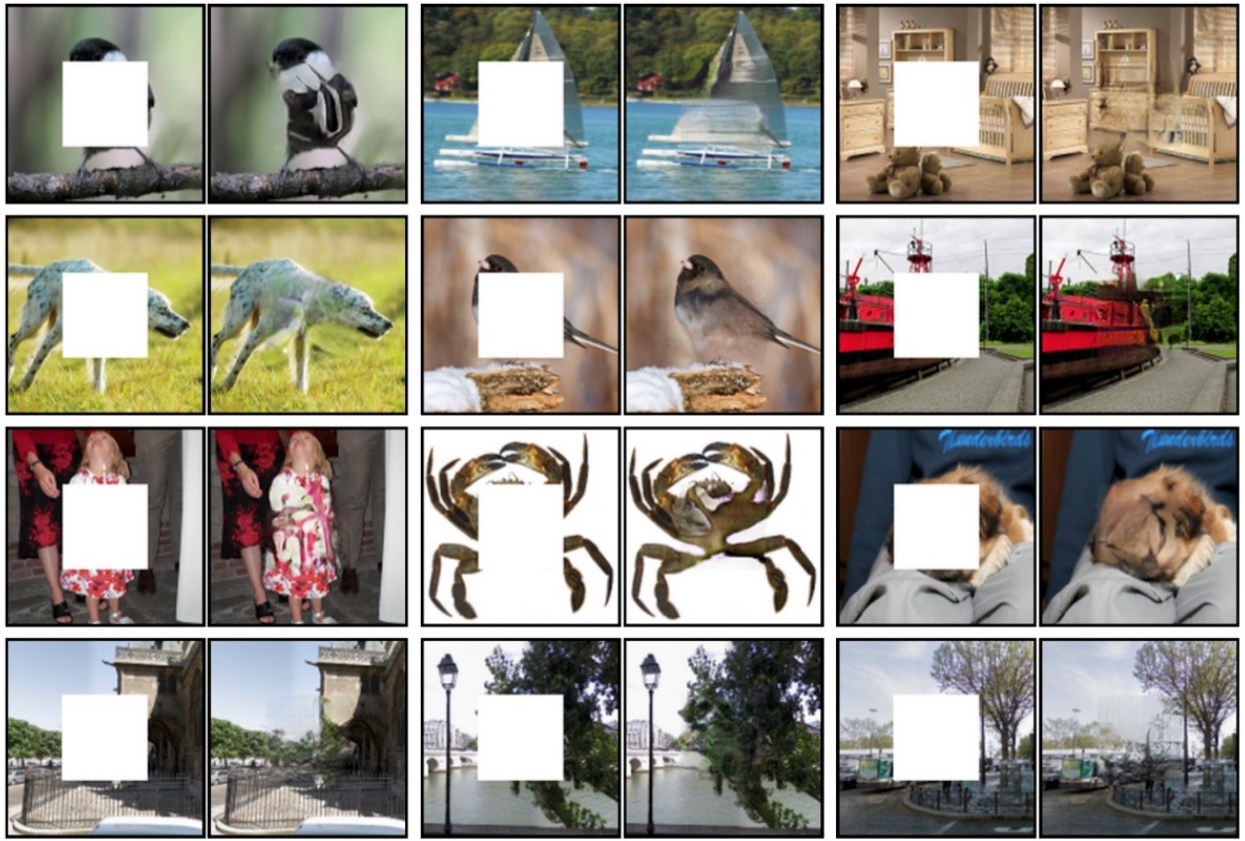
High-resolution image synthesis

Text to image (StackGAN)

Face synthesis

Synthesis results under various illuminations. The first row is the synthesized image, the second r

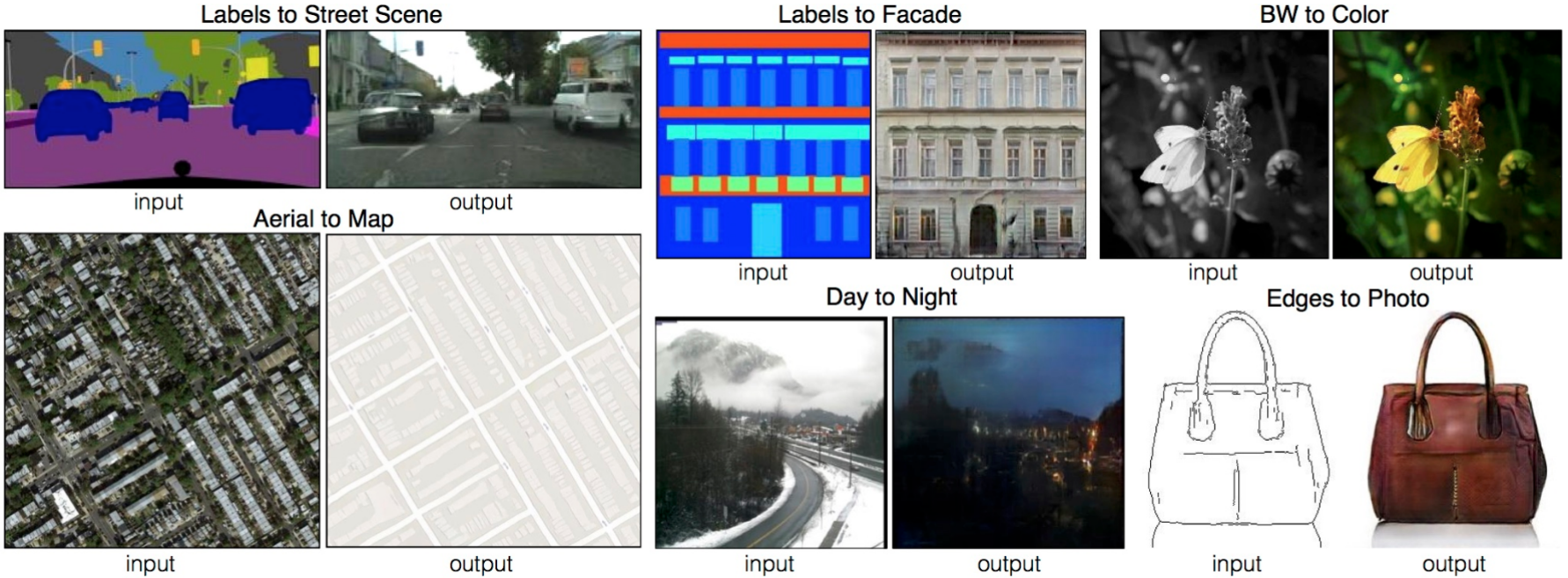
Image inpainting



Matching style - DiscoGAN

(b) Handbag images (input) & **Generated** shoe images (output)

Pix2Pix

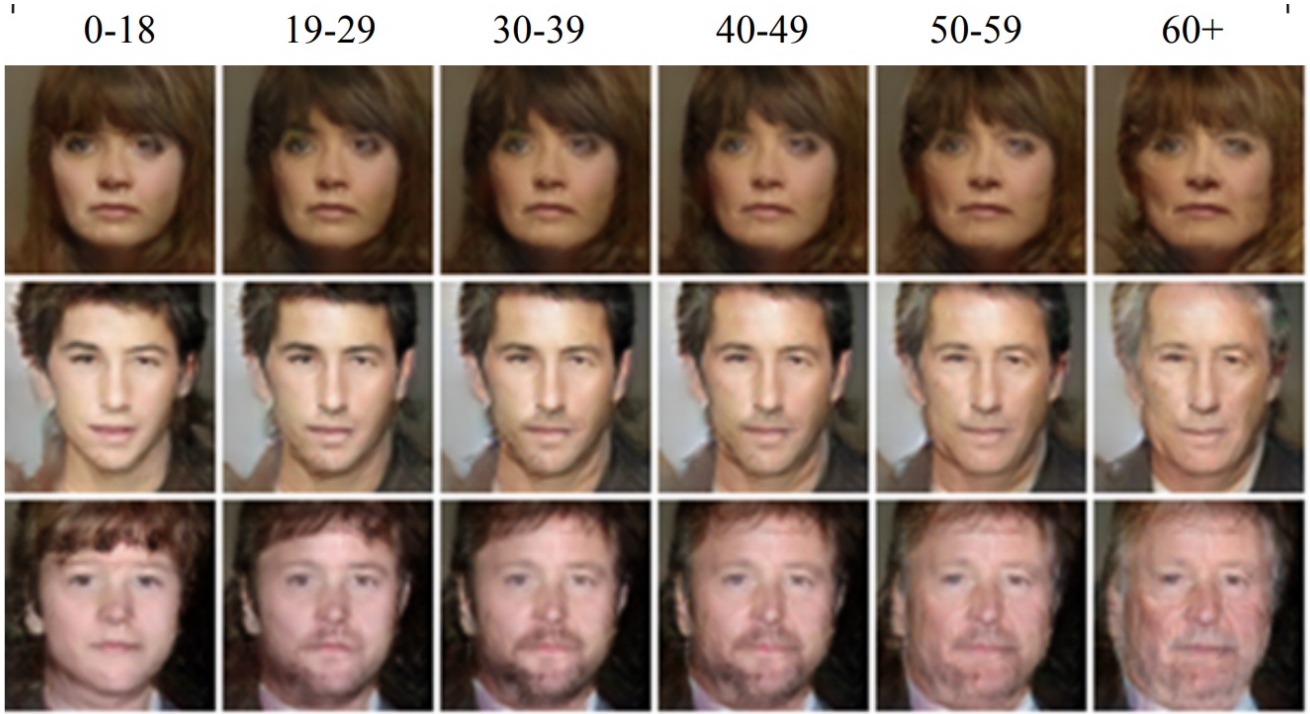

Labels to Street Scene
input
output
Labels to Facade
input
output
BW to Color
input
output
Aerial to Map
input
output
Day to Night
input
output
Edges to Photo
input
output

Emoji from pictures - DTN

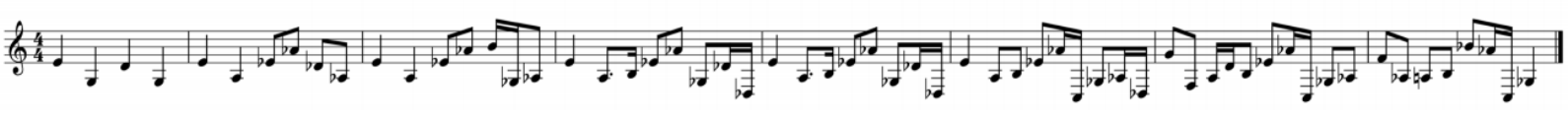
Figure 4: Shown, side by side are sample images from the CelebA dataset, the emoji images created manually using a web interface (for validation only), and the result of the unsupervised DTN. See Tab. 4 for retrieval performance.

Image editing (IcGAN)

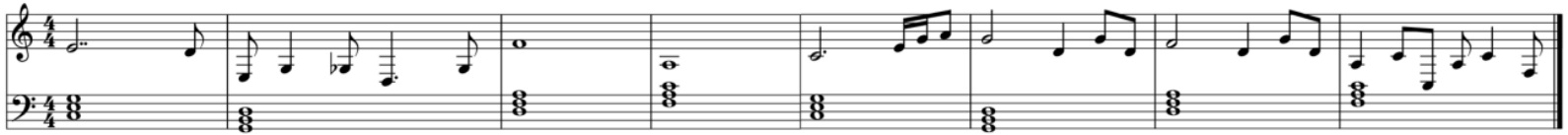
Face aging (Age-cGAN)



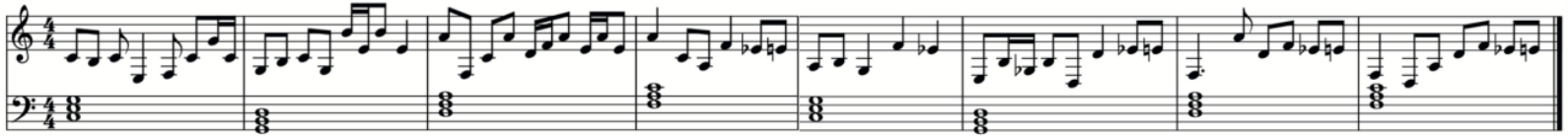
Music generation



(a) MidiNet model 1



(b) MidiNet model 2

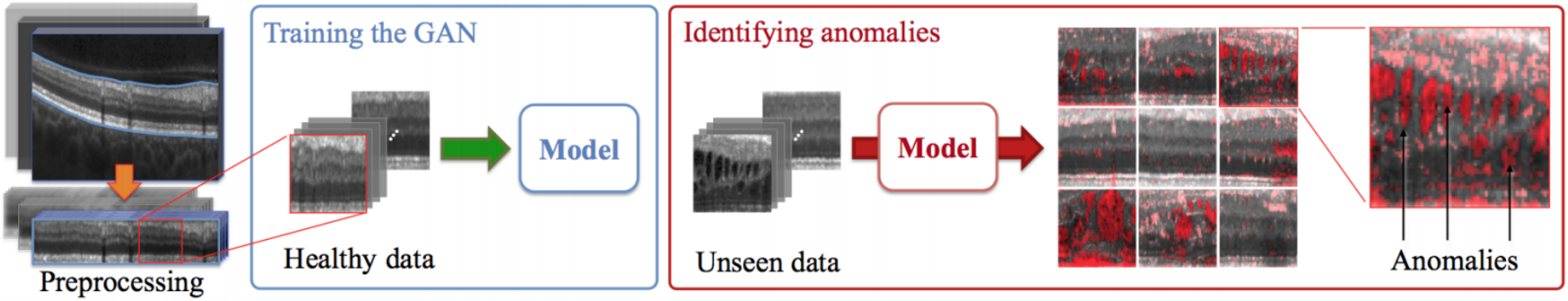


(c) MidiNet model 3

Figure 3. Example result of the melodies (of 8 bars) generated by different implementations of MidiNet.

https://salu133445.github.io/musegan/results

Medical (Anomaly Detection)

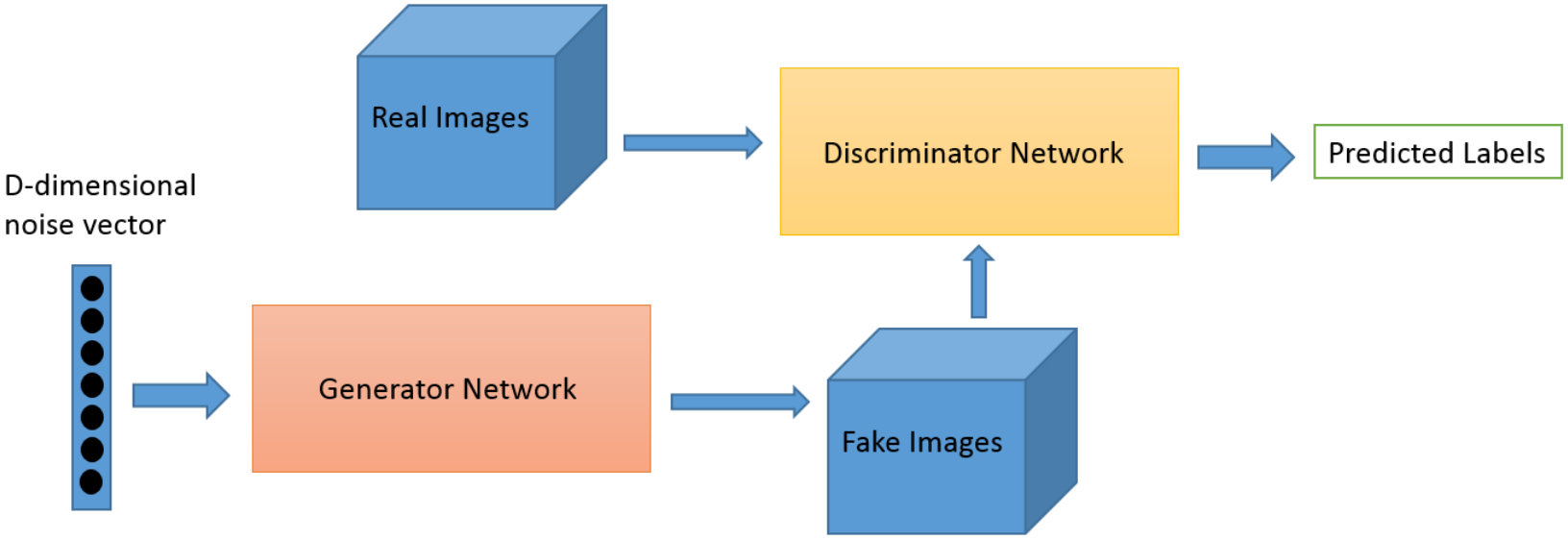


Generative Adversarial Models

Generative	vs.	Discriminative
• generate samples		• classify input data
class label y => sample x		sample x => class label y

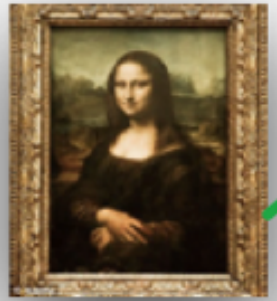
Examples

Generative Adversarial Models



Examples, akin to counterfeiter vs. cop

D: Detective



R: Real Data

G: Generator (Forger)

I: Input for Generator

Interactive playground

- https://affinelayer.com/pixsrv/

```
class GAN():
    def __init__(self):
        self.img_rows = 28
        self.img_cols = 28
        self.channels = 1
        self.img_shape = (self.img_rows, self.img_cols, self.channels)
        self.latent_dim = 100

        optimizer = Adam(0.0002, 0.5)

        # Build and compile the discriminator
        self.discriminator = self.build_discriminator()
        self.discriminator.compile(loss='binary_crossentropy',
            optimizer=optimizer,
            metrics=['accuracy'])

        # Build the generator
        self.generator = self.build_generator()

        # The generator takes noise as input and generates imgs
        z = Input(shape=(self.latent_dim,))
        img = self.generator(z)

        # For the combined model we will only train the generator
        self.discriminator.trainable = False

        # The discriminator takes generated images as input and
determines validity
        validity = self.discriminator(img)

        # The combined model  (stacked generator and discriminator)
        # Trains the generator to fool the discriminator
        self.combined = Model(z, validity)
        self.combined.compile(loss='binary_crossentropy',
optimizer=optimizer)
```

```
def build_generator(self):
        model = Sequential()
        model.add(Dense(256, input_dim=self.latent_dim))
        model.add(LeakyReLU(alpha=0.2))
        model.add(BatchNormalization(momentum=0.8))
        model.add(Dense(512))
        model.add(LeakyReLU(alpha=0.2))
        model.add(BatchNormalization(momentum=0.8))
        model.add(Dense(1024))
        model.add(LeakyReLU(alpha=0.2))
        model.add(BatchNormalization(momentum=0.8))
        model.add(Dense(np.prod(self.img_shape),
activation='tanh'))
        model.add(Reshape(self.img_shape))

        model.summary()
        noise = Input(shape=(self.latent_dim,))
        img = model(noise)

        return Model(noise, img)
```

```
def build_discriminator(self):

        model = Sequential()
        model.add(Flatten(input_shape=self.img_shape))
        model.add(Dense(512))
        model.add(LeakyReLU(alpha=0.2))
        model.add(Dense(256))
        model.add(LeakyReLU(alpha=0.2))
        model.add(Dense(1, activation='sigmoid'))
        model.summary()

        img = Input(shape=self.img_shape)
        validity = model(img)

        return Model(img, validity)
```

```
def train(self, epochs, batch_size=128,
sample_interval=50):
        # Load the dataset
        (X_train, _), (_, _) = mnist.load_data()

        # Rescale -1 to 1
        X_train = X_train / 127.5 - 1.
        X_train = np.expand_dims(X_train, axis=3)

        # Adversarial ground truths
        valid = np.ones((batch_size, 1))
        fake = np.zeros((batch_size, 1))
```

```
if __name__ == '__main__':
    gan = GAN()
    gan.train(epochs=30000, batch_size=32,
sample_interval=200)
```

```
for epoch in range(epochs):

    # ---------------------
    #  Train Discriminator
    # ---------------------

    # Select a random batch of images
    idx = np.random.randint(0, X_train.shape[0], batch_size)
    imgs = X_train[idx]

    noise = np.random.normal(0, 1, (batch_size, self.latent_dim))

    # Generate a batch of new images
    gen_imgs = self.generator.predict(noise)

    # Train the discriminator
    d_loss_real = self.discriminator.train_on_batch(imgs, valid)
    d_loss_fake = self.discriminator.train_on_batch(gen_imgs, fake)
    d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

    # ---------------------
    #  Train Generator
    # ---------------------

    noise = np.random.normal(0, 1, (batch_size, self.latent_dim))

    # Train the generator (to have the discriminator label samples as valid)
    g_loss = self.combined.train_on_batch(noise, valid)

    # Plot the progress
    print ("%d [D loss: %f, acc.: %.2f%%] [G loss: %f]" % (epoch, d_loss[0],
    100*d_loss[1], g_loss))

    # If at save interval => save generated image samples
    if epoch % sample_interval == 0:
        self.sample_images(epoch)
```

- All codes are on this Colab link:
  - simpleGAN: https://colab.research.google.com/drive/1aZeqH2ssCaEF9OMG8eP8jh93BT0EzWHo
  - discoGAN: https://colab.research.google.com/drive/1hd_eShZaNt5FTVdOrVXFDLCm2zI_9svx
  - cycleGAN: https://colab.research.google.com/drive/1qw2Qr-rOoclY7c9UEdP_4wrLlhsJnBpy

More GAN codes on Keras & Pytorch

- https://github.com/eriklindernoren/Keras-GAN
- https://github.com/eriklindernoren/PyTorch-GAN

Useful resources

- http://speech.ee.ntu.edu.tw/~tlkagk/talk.html
- http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture12.pdf

- https://medium.com/@jonathan_hui/gan-some-cool-applications-of-gans-4c9ecca35900