Generative Adversarial Networks

Andrew H. Fagg

Cycle GAN

Zhu et a., 2017

- Image to image translation
  - Translate an image in one domain into another domain
- Use discriminator for each domain to tell whether the translation was right

Monet ⇄ Photos

Monet → photo

photo → Monet

Zebras ⇄ Horses

zebra → horse

horse → zebra

Summer ⇄ Winter

summer → winter

winter → summer

Photograph

Monet

Van Gogh

Cezanne

Ukiyo-e

Cycle GAN Implementation

- Must train the image translators and the discriminators at the same time
- Convenient to use Model nesting to make this work

- One tool:
  - model.trainable property (a Boolean) controls whether the parameters in the model can be adjusted
  - Catch: this property is *only* read by model.compile()

Cycle GAN Implementation

```
// Create new models
dA = create_discriminator()
dB = create_discriminator()

// Compile these models with adjustable
//   parameters
dA.compile(loss='mse', …)
dB.compile(loss='mse', …)
```

Ref: David Foster (2019), “Generative Deep Learning”

Cycle GAN Implementation

```
// Create individual generator models
gAB = build_generator()
gBA = build_generator()

// Future uses of dA/dB will not be trainable
dA.trainable = False
dB.trainable = False
```

Cycle GAN Implementation

```
// Create the Meta-Model
inA = Input(shape=img_shape)
inB = Input(shape=img_shape)

// Create fake images
fakeA = gBA(inB)
fakeB = gAB(inA)

// Create duplicate images from the fakes
reconA = gBA(fakeB)
reconB = gAB(fakeA)

// Create image identities: don’t change an image if it is
// already the right type
idA = gBA(inA)
idB = gAB(inB)
```

Cycle GAN Implementation

```
// Evaluate the fake images
validA = dA(fakeA)
validB = dB(fakeB)

// Create the meta Model
model = Model(inputs=[inA, inB]
                     outputs=[validA, validB,
                                      idA, idB,
                                      reconA, reconB], …)

// Compile it
model.compile(loss=[‘mse’, ‘mse’,
                                ‘mae’, ‘mae’,
                                ‘mae’,‘mae’], …)
```

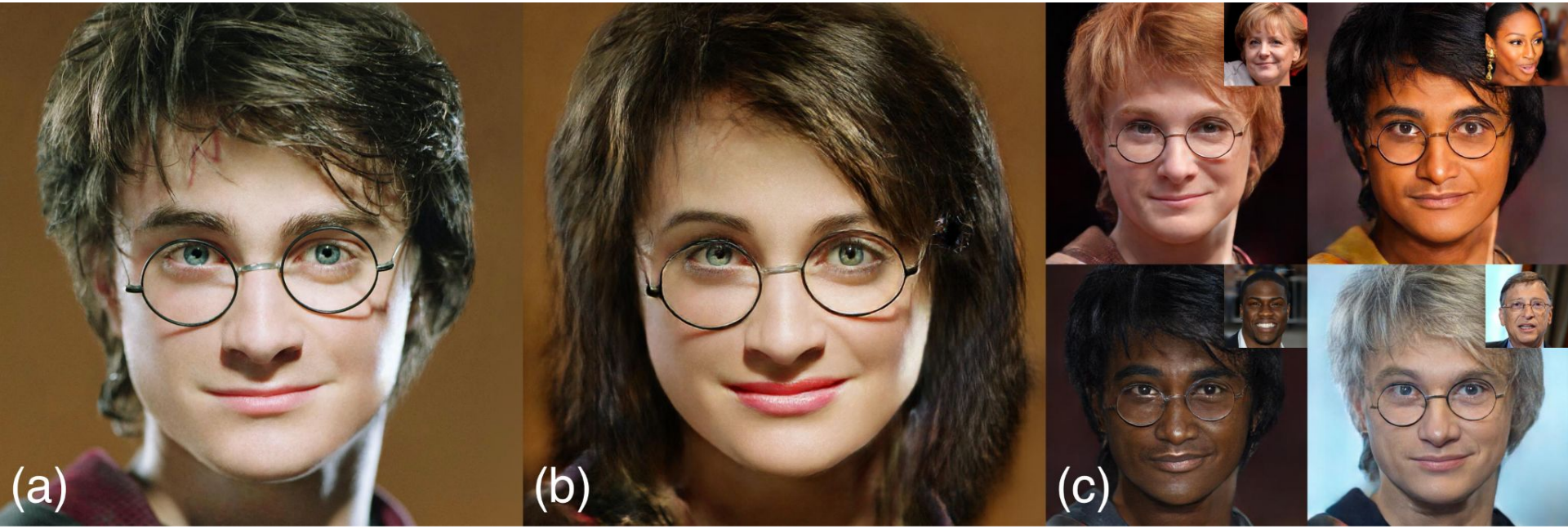
Cycle GAN Implementation

```
// Train one batch
imgsA, imgsB are the batch

fakeA = gBA(imgsB)
fakeB = gAB(imgsA)

dA.fit(epochs=1, inputs=np.concatenate([imgsA,fakeA]),
                 outputs=np.contatenate([1s,0s]))

dB.fit(epochs=1, inputs=np.concatenate([imgsB,fakeB]),
                 outputs=np.contatenate([1s,0s]))

model.fit(epochs=1, inputs=[imgsA, imgsB]
                    outputs=[1s, 1s,
                             imgsA, imgsB,
                             imgsA, imgsB], …)
```

Style GAN

Two source images:

- Content: Goal is to create an image that maintains the detailed structure of the input image (e.g., where are the edges and other texture? What shapes are there?)
- Style: Goal is to create an image that tries to capture “style” elements in the image (higher-level features)
  - Color
  - Larger shapes and their spatial relationships
  -


(a)
(b)
(c)