Final Project
Classifier & Image Generation

Group 4

108062313 黃允暘 108062311杜方辰

108062122 顏訓哲 108062336陳柏元

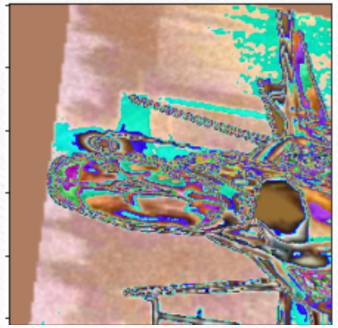
Classifier

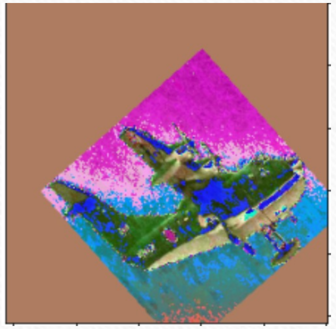
Methodology

- Model: Efficientnet-B6
- Data analysis: mean std
- Calculate each kind of plane number

Training method

- Optimizer: Adam
- Learning rate: 1e-5
- Loss function: categorical crossentropy
- Matric: Accuracy
- Image Augmentations: color jitter, rotation, invert…
- Adaptive learning rate



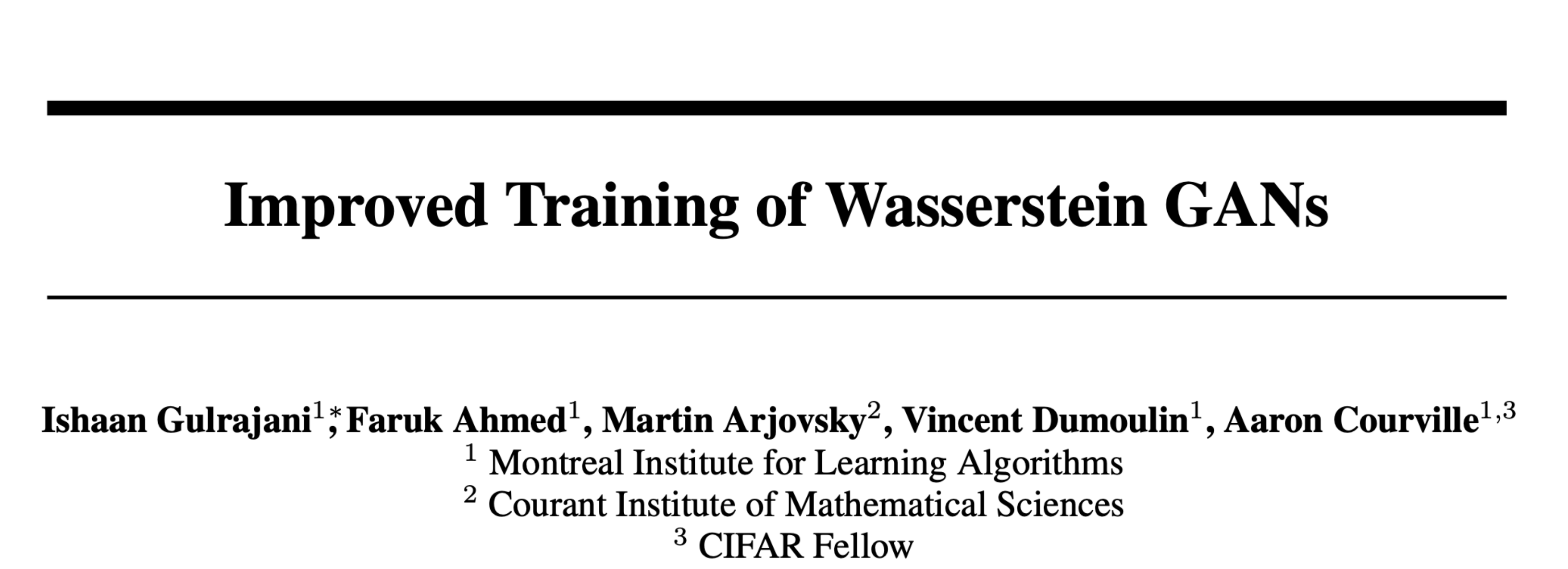


Result

- Validation set: 0.95
- Test set: 0.93

Generator

Improved WGAN



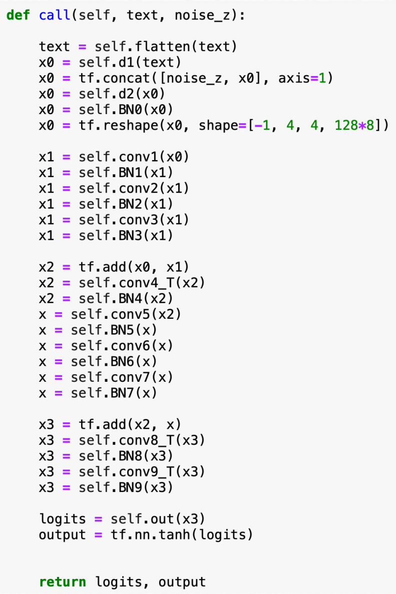
Improved Training of Wasserstein GANs

Ishaan Gulrajani[1]*, **Faruk Ahmed**[1], **Martin Arjovsky**[2], **Vincent Dumoulin**[1], **Aaron Courville**[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow

Model architecture

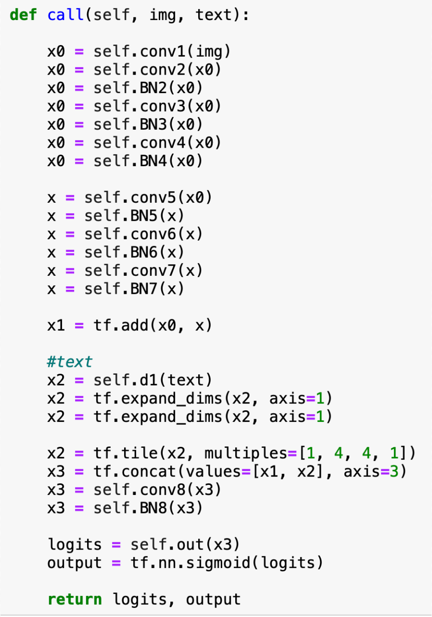
- Text Encoder
    - Pretrained Bert

- Generator



```
def call(self, text, noise_z):

    text = self.flatten(text)
    x0 = self.d1(text)
    x0 = tf.concat([noise_z, x0], axis=1)
    x0 = self.d2(x0)
    x0 = self.BN0(x0)
    x0 = tf.reshape(x0, shape=[-1, 4, 4, 128*8])

    x1 = self.conv1(x0)
    x1 = self.BN1(x1)
    x1 = self.conv2(x1)
    x1 = self.BN2(x1)
    x1 = self.conv3(x1)
    x1 = self.BN3(x1)

    x2 = tf.add(x0, x1)
    x2 = self.conv4_T(x2)
    x2 = self.BN4(x2)
    x = self.conv5(x2)
    x = self.BN5(x)
    x = self.conv6(x)
    x = self.BN6(x)
    x = self.conv7(x)
    x = self.BN7(x)

    x3 = tf.add(x2, x)
    x3 = self.conv8_T(x3)
    x3 = self.BN8(x3)
    x3 = self.conv9_T(x3)
    x3 = self.BN9(x3)

    logits = self.out(x3)
    output = tf.nn.tanh(logits)


    return logits, output
```

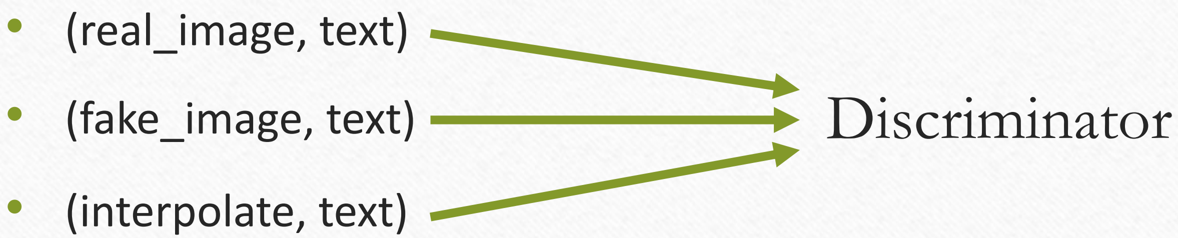
- Discriminator



```
def call(self, img, text):

    x0 = self.conv1(img)
    x0 = self.conv2(x0)
    x0 = self.BN2(x0)
    x0 = self.conv3(x0)
    x0 = self.BN3(x0)
    x0 = self.conv4(x0)
    x0 = self.BN4(x0)

    x = self.conv5(x0)
    x = self.BN5(x)
    x = self.conv6(x)
    x = self.BN6(x)
    x = self.conv7(x)
    x = self.BN7(x)

    x1 = tf.add(x0, x)

    #text
    x2 = self.d1(text)
    x2 = tf.expand_dims(x2, axis=1)
    x2 = tf.expand_dims(x2, axis=1)

    x2 = tf.tile(x2, multiples=[1, 4, 4, 1])
    x3 = tf.concat(values=[x1, x2], axis=3)
    x3 = self.conv8(x3)
    x3 = self.BN8(x3)

    logits = self.out(x3)
    output = tf.nn.sigmoid(logits)

    return logits, output
```

Loss function and optimization

- Optimizer: Adam
- Three pairs:

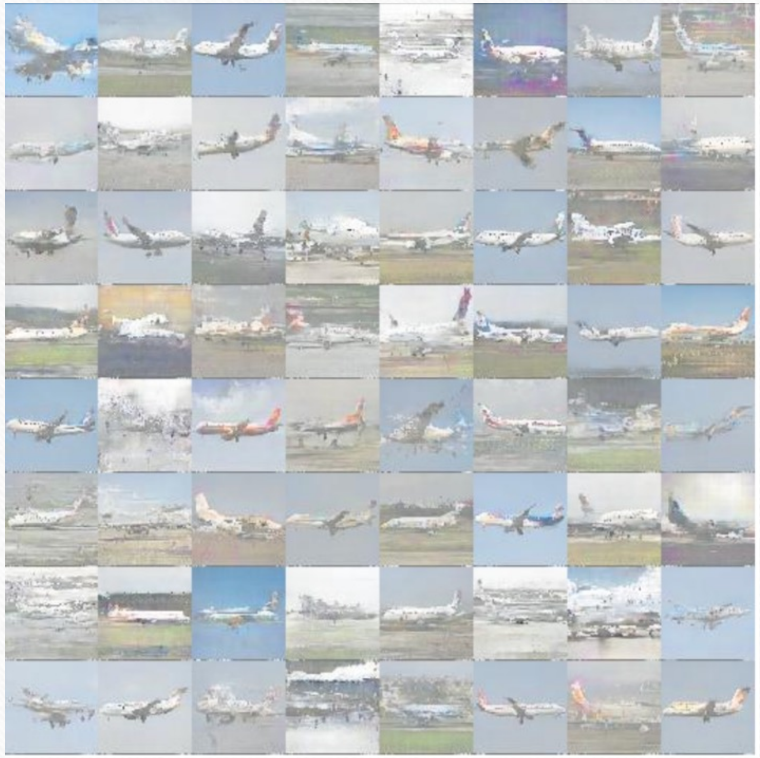


  - (real_image, text)
  - (fake_image, text)
  - (interpolate, text)

Discriminator

interpolate = a*real_image + b*fake_image, a+b=1

(Bonus) Test result / Demo result



Thank you for your listening