FINAL PROJECT

Image Classifier and Generator

Group 8
108034044張柏清 108062129吳東曄
109062223郭彥廷 109090023甘倍菁

Methodology

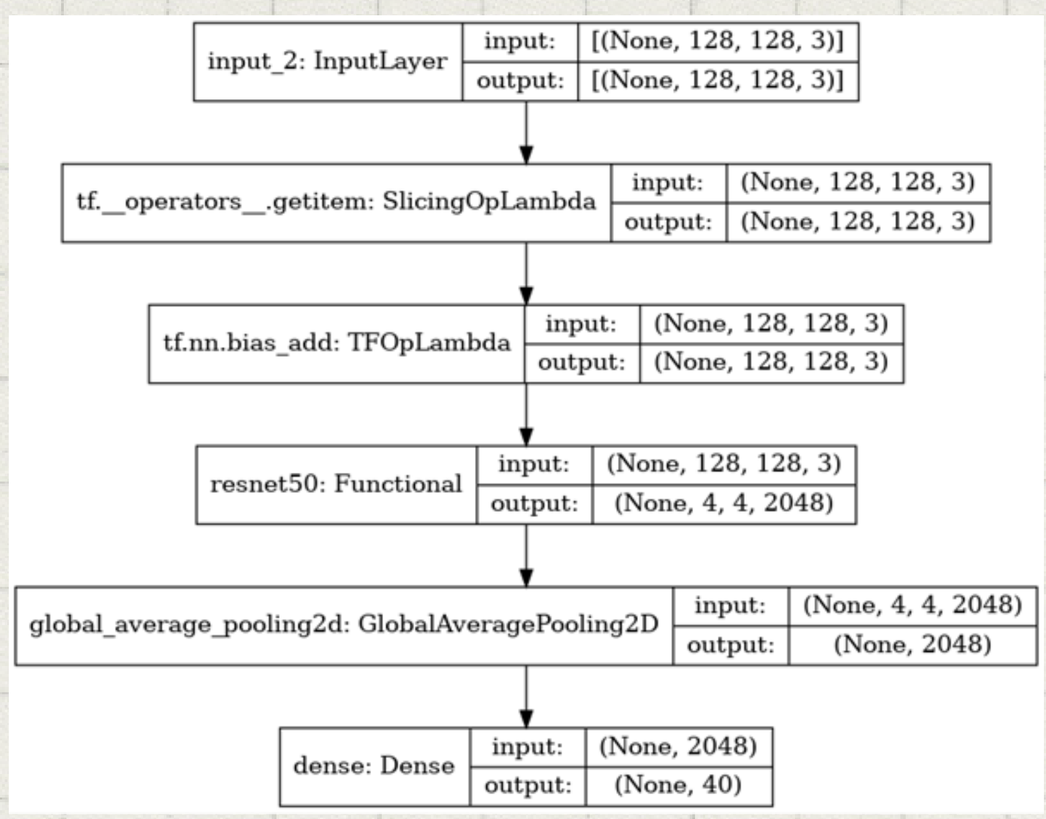
classifier

- method : CNN tensorflow to do our classifier
- resize the dataset to (128, 128)
- two layers for data augmentation, one for randomrotation, the other for randomFlip

Methodology

classifier

using model ResNet50



Methodology

classifier

Parameter

dataset : Military Aircraft Detection Dataset
batch size : 25
optimizer : optimizers.Adam
learning rate : 0.001
Cosine decay step: 500

Methodology

classifier

test result and demo result

Final accuracy : 0.7443 on the test set

demo result

filename	class
0001.jpg	A400M
0002.jpg	AV8B
0003.jpg	B2
0004.jpg	C130
0005.jpg	EF2000
0006.jpg	F16
0007.jpg	F117
0008.jpg	JAS39
0009.jpg	MQ9
0010.jpg	Rafale

Methodology

generator

Discriminator

- It is a binary output with "yes" or "no"
- Use binary cross entropy to be the loss function
- Separately give the true and the fake data.
- Calculate the score to adjust the weight of the discriminator.

```python
def train_discriminator(real_images, opt_d):
    # Clear discriminator gradients
    opt_d.zero_grad()

    # Pass real images through discriminator
    real_preds = discriminator(real_images)
    real_targets = torch.ones(real_images.size(0), 1, device=device)
    real_loss = F.binary_cross_entropy(real_preds, real_targets)
    real_score = torch.mean(real_preds).item()

    # Generate fake images
    latent = torch.randn(batch_size, latent_size, 1, 1, device=device)
    fake_images = generator(latent)

    # Pass fake images through discriminator
    fake_targets = torch.zeros(fake_images.size(0), 1, device=device)
    fake_preds = discriminator(fake_images)
    fake_loss = F.binary_cross_entropy(fake_preds, fake_targets)
    fake_score = torch.mean(fake_preds).item()

    # Update discriminator weights
    loss = real_loss + fake_loss
    loss.backward()
    opt_d.step()
    return loss.item(), real_score, fake_score
```

Methodology

generator

Generator

- Using method of DCGAN, so we use convolution transpose to have Conv2DTranspose.
- It can expand the message of the picture

Methodology

generator

Generator

- the purpose of generator is to trick the discriminator when training
- produce the fake image first
- update the weight of generator by the loss calculate from the discriminator

```python
def train_generator(opt_g):
    # Clear generator gradients
    opt_g.zero_grad()

    # Generate fake images
    latent = torch.randn(batch_size, latent_size, 1, 1, device=device)
    fake_images = generator(latent)

    # Try to fool the discriminator
    preds = discriminator(fake_images)
    targets = torch.ones(batch_size, 1, device=device)
    loss = F.binary_cross_entropy(preds, targets)

    # Update generator weights
    loss.backward()
    opt_g.step()

    return loss.item()
```

Methodology

generator

Parameter

dataset : FGVC dataset benchmark

batch size : 25

optimizer : optimizers.Adam

learning rate : 0.0002

Methodology

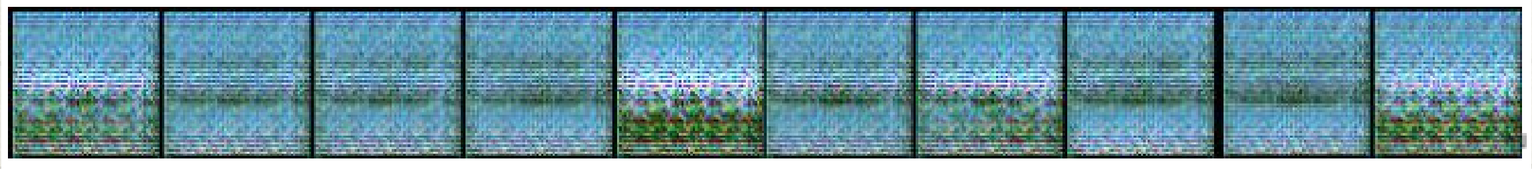
generator

test result and demo result

Aircraft

Airbus

Boeing



Discussion and Conclusion

Classifier
the accuracy of our result is good,
but we can still adjust to make our result better

Generator

- time-consuming
- the parameter is not good enough, leading to our result
- the bounding of the images have plenty of background, which interfere the result

Thanks you for listening