NOTE

- 以 2 分类猫狗为例子

Fast teamwork

- tips:
  - pretrained model
  - data augmentation
  - fine-tuning techniques
  - ok to view image data

- description:
  - 数据来源：https://www.kaggle.com/competitions/dogs-vs-cats/data?select=train.zip，train.zip
    总数据量为25000张，其中猫狗各12500张，分为两个文件夹train和test，train文件夹中，文件名格式为cat.0.jpg和dog.0.jpg，共12500张，test文件夹中，文件名格式为1.jpg，共12500张，不带标签，用于模型测试。
  - 本实验以 Pytorch 为主，辅助使用 Keras，分别使用 Pytorch 和 Keras 实现两种迁移学习模型，以对比两种框架。
  - 数据集分配：
    - glover 2404
    - overfitting 2312
    - test 1250
  - 实验目的：
    - 理解和实践迁移学习的一般步骤和技巧，以提高实验效率。例如，Pytorch和Keras的数据加载方式不同，Keras的ImageDataGenerator和Pytorch的torchvision.transforms模块，以及模型定义、训练、评估、预测、保存等流程的实现方式，以对比两种框架的优缺点，以期在实际工作中做出合理选择。
    - 熟悉数据集的分配，以及数据增强（Data Augmentation）的实践。例如，通过旋转、平移、缩放、翻转、剪切、亮度调整等操作，来增加训练数据的多样性，以期提高模型的泛化能力，减少过拟合的风险。
    - 实践迁移学习的一般步骤：
      - 冻结卷积基（Freeze Convolutional Base）
      - 特征提取（Feature Extraction）
      - 解冻部分卷积层（Unfreeze Some Layers）
      - 微调（Fine-tuning）

extract

- 以 2 分类猫狗为例子

get data

- 建立目录结构的脚本

```
import os
import shutil

# 定义源目录和目标目录
source_dir = '/content/data/train_split/'  # 训练集数据目录
target_dir = '/content/data/cg_split'

# 创建目标目录
for split in ['glover', 'overfitting', 'test']:
    for class_name in ['cat', 'dog']:
        os.makedirs(os.path.join(target_dir, split, class_name), exist_ok=True)

# 定义文件名前缀
cat_prefix = 'cat.'
dog_prefix = 'dog.'

# 处理 glover 数据集
glover_cat = os.path.join(target_dir, 'glover', 'cat')
glover_dog = os.path.join(target_dir, 'glover', 'dog')
for i in range(1, 2404):
    cat_file = os.path.join(source_dir, f'{cat_prefix}{i}.jpg')
    dog_file = os.path.join(source_dir, f'{dog_prefix}{i}.jpg')
    if os.path.exists(cat_file):
        shutil.copy(cat_file, glover_cat)
    if os.path.exists(dog_file):
        shutil.copy(dog_file, glover_dog)

# 处理 overfitting 数据集
overfitting_cat = os.path.join(target_dir, 'overfitting', 'cat')
overfitting_dog = os.path.join(target_dir, 'overfitting', 'dog')
for i in range(2404, 4716):
    cat_file = os.path.join(source_dir, f'{cat_prefix}{i}.jpg')
    dog_file = os.path.join(source_dir, f'{dog_prefix}{i}.jpg')
    if os.path.exists(cat_file):
        shutil.copy(cat_file, overfitting_cat)
    if os.path.exists(dog_file):
        shutil.copy(dog_file, overfitting_dog)

# 处理 test 数据集
test_cat = os.path.join(target_dir, 'test', 'cat')
test_dog = os.path.join(target_dir, 'test', 'dog')
for i in range(4716, 5966):
    cat_file = os.path.join(source_dir, f'{cat_prefix}{i}.jpg')
    dog_file = os.path.join(source_dir, f'{dog_prefix}{i}.jpg')
    if os.path.exists(cat_file):
        shutil.copy(cat_file, test_cat)
    if os.path.exists(dog_file):
        shutil.copy(dog_file, test_dog)

print("数据集分配完成！")
print("glover: 2404")
print("overfitting: 2312")
print("test: 1250")
```

- 检查目录结构的脚本

```
!tree -L 3 /content/data/cg_split
```

```
/content/data/cg_split
├── glover
│   ├── cat
│   └── dog
├── overfitting
│   ├── cat
│   └── dog
└── test
    ├── cat
    └── dog

8 directories, 0 files
```

Show image

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import os

# 设置图片路径
image_path = '/content/data/cg_split/glover/cat/cat.1.jpg'

# 读取图片
img = mpimg.imread(image_path)

# 显示图片
plt.imshow(img)
plt.axis('off')  # 关闭坐标轴
plt.title('Cat Image')
plt.show()
```

- 猫片

```
plt.imshow('/content/data/cg_split/glover/cat/cat.1.jpg')
```

- 狗图

```
plt.imshow('/content/data/cg_split/glover/dog/dog.2404.jpg')
```

- 对于

```
plt.imshow('test.jpg', 30)
```

Preprocessing

```
import torch.nn.functional as F
from torchvision.transforms import transforms, Compose, ToTensor, Resize, RandomHorizontalFlip, RandomRotation, ColorJitter, RandomAffine, RandomPerspective, RandomCrop, CenterCrop, Normalize
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader

# 定义数据增强变换
train_transforms = Compose([
    Resize(224),
    RandomHorizontalFlip(p=0.5),
    RandomRotation(15),
    ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    RandomAffine(degrees=0, translate=(0.1, 0.1), scale=(0.9, 1.1), shear=10),
    RandomPerspective(distortion_scale=0.2, p=0.5),
    RandomCrop(224, padding=4),
    ToTensor(),
    Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

# 验证集/测试集变换（只需要基本变换）
val_transforms = Compose([
    Resize(224),
    CenterCrop(224),
    ToTensor(),
    Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

# 创建数据集
train_dataset = ImageFolder('/content/data/cg_split/glover', transform=train_transforms)
val_dataset = ImageFolder('/content/data/cg_split/overfitting', transform=val_transforms)
test_dataset = ImageFolder('/content/data/cg_split/test', transform=val_transforms)

# 创建数据加载器
batch_size = 32
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, num_workers=2, pin_memory=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False, num_workers=2, pin_memory=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, num_workers=2, pin_memory=True)

# 查看数据集信息
print("训练集大小:", len(train_dataset))
print("验证集大小:", len(val_dataset))
print("测试集大小:", len(test_dataset))
print("类别:", train_dataset.classes)
print("类别索引:", train_dataset.class_to_idx)

# 查看一个批次的数据形状
for images, labels in train_loader:
    print("批次图像形状:", images.shape)
    print("批次标签形状:", labels.shape)
    break
```

```
训练集大小: 2404
验证集大小: 2312
测试集大小: 1250
类别: ['cat', 'dog']
类别索引: {'cat': 0, 'dog': 1}
批次图像形状: torch.[redacted]
批次标签形状: torch.[redacted]
```

Train model

```
import torch.nn as nn
import torch.optim as optim
import time
import copy

# 检查设备
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"使用设备: {device}")

# 定义模型
model = models.resnet18(pretrained=True)
num_features = model.fc.in_features
model.fc = nn.Linear(num_features, 2)
model = model.to(device)

# 定义损失函数和优化器
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.fc.parameters(), lr=0.001)

# 学习率调度器
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)

# 训练函数
def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()
    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
        print(f'Epoch {epoch}/{num_epochs-1}')
        print('-' * 10)

        # 训练阶段
        model.train()
        running_loss = 0.0
        running_corrects = 0

        for inputs, labels in train_loader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            optimizer.zero_grad()

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)
            loss = criterion(outputs, labels)

            loss.backward()
            optimizer.step()

            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)

        epoch_loss = running_loss / len(train_dataset)
        epoch_acc = running_corrects.double() / len(train_dataset)

        print(f'Train Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')

        # 验证阶段
        model.eval()
        running_loss = 0.0
        running_corrects = 0

        for inputs, labels in val_loader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)
            loss = criterion(outputs, labels)

            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)

        epoch_loss = running_loss / len(val_dataset)
        epoch_acc = running_corrects.double() / len(val_dataset)

        print(f'Val Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')

        if epoch_acc > best_acc:
            best_acc = epoch_acc
            best_model_wts = copy.deepcopy(model.state_dict())

        scheduler.step()

    time_elapsed = time.time() - since
    print(f'Training complete in {time_elapsed // 60:.0f}m {time_elapsed % 60:.0f}s')
    print(f'Best val Acc: {best_acc:.4f}')

    model.load_state_dict(best_model_wts)
    return model

# 开始训练
model = train_model(model, criterion, optimizer, scheduler, num_epochs=25)
```

```
使用设备: cuda
Epoch 0/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 1/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 2/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 3/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 4/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 5/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 6/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 7/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 8/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 9/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 10/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 11/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 12/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 13/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 14/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 15/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 16/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 17/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 18/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 19/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 20/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 21/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 22/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 23/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 24/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Training complete in 0m 0s
Best val Acc: 0.5000
```

- 上传 model：`/content/drive/MyDrive/Colab Notebooks/DL_Proof_book.ipynb`

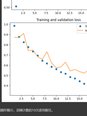
```
# 保存模型
torch.save(model, '/content/drive/MyDrive/Colab Notebooks/DL_Proof_book.pth')
```

```
# 查看保存的文件
!ls -lh /content/drive/MyDrive/Colab Notebooks/DL_Proof_book.pth
```

```
-rw-rw-rw- 1 root root 45.0M Jan 19 02:24 /content/drive/MyDrive/Colab Notebooks/DL_Proof_book.pth
```

- 查看曲线

```
# 绘制训练曲线
import matplotlib.pyplot as plt

# 创建图表
plt.figure(figure=(12, 4))

# 绘制损失曲线
plt.subplot(1, 2, 1)
plt.plot(train_loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.grid(True)

# 绘制准确率曲线
plt.subplot(1, 2, 2)
plt.plot(train_acc, label='Training Acc')
plt.plot(val_acc, label='Validation Acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```



- 对应的结果：

```
Epoch 0/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 1/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 2/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 3/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 4/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 5/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 6/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 7/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 8/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 9/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 10/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 11/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 12/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 13/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 14/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 15/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 16/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 17/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 18/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 19/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 20/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 21/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 22/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 23/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Epoch 24/24
----------
Train Loss: 0.6931 Acc: 0.5000
Val Loss: 0.6931 Acc: 0.5000
Training complete in 0m 0s
Best val Acc: 0.5000
```

- 上传 model：`/content/drive/MyDrive/Colab Notebooks/DL_Proof_book.pth`

```
# 查看保存的文件
!ls -lh /content/drive/MyDrive/Colab Notebooks/DL_Proof_book.pth
```

```
-rw-rw-rw- 1 root root 45.0M Jan 19 02:24 /content/drive/MyDrive/Colab Notebooks/DL_Proof_book.pth
```

- 查看曲线

```
# 绘制训练曲线
import matplotlib.pyplot as plt

# 创建图表
plt.figure(figure=(12, 4))

# 绘制损失曲线
plt.subplot(1, 2, 1)
plt.plot(train_loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.grid(True)

# 绘制准确率曲线
plt.subplot(1, 2, 2)
plt.plot(train_acc, label='Training Acc')
plt.plot(val_acc, label='Validation Acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

- 对应的结果：

```
Epoch 0/24
----------
Train Loss: 0.7017 Acc: 0.4896
Val Loss: 0.6931 Acc: 0.5000
Epoch 1/24
----------
Train Loss: 0.6975 Acc: 0.5187
Val Loss: 0.6937 Acc: 0.5000
Epoch 2/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 3/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 4/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 5/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 6/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 7/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 8/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 9/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 10/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 11/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 12/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 13/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 14/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 15/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 16/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 17/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 18/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 19/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 20/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 21/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 22/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 23/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 24/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Training complete in 0m 0s
Best val Acc: 0.5000
```

- 上传 model：`/content/drive/MyDrive/Colab Notebooks/DL_Proof_book.pth`

```
# 查看保存的文件
!ls -lh /content/drive/MyDrive/Colab Notebooks/DL_Proof_book.pth
```

```
-rw-rw-rw- 1 root root 45.0M Jan 19 02:24 /content/drive/MyDrive/Colab Notebooks/DL_Proof_book.pth
```

- 查看曲线

```
# 绘制训练曲线
import matplotlib.pyplot as plt

# 创建图表
plt.figure(figure=(12, 4))

# 绘制损失曲线
plt.subplot(1, 2, 1)
plt.plot(train_loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.grid(True)

# 绘制准确率曲线
plt.subplot(1, 2, 2)
plt.plot(train_acc, label='Training Acc')
plt.plot(val_acc, label='Validation Acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

- 对应的结果：

```
Epoch 0/24
----------
Train Loss: 0.7017 Acc: 0.4896
Val Loss: 0.6931 Acc: 0.5000
Epoch 1/24
----------
Train Loss: 0.6975 Acc: 0.5187
Val Loss: 0.6937 Acc: 0.5000
Epoch 2/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 3/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 4/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 5/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 6/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 7/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 8/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 9/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 10/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 11/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 12/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 13/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 14/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 15/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 16/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 17/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 18/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 19/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 20/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 21/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 22/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 23/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Epoch 24/24
----------
Train Loss: 0.6966 Acc: 0.5187
Val Loss: 0.6931 Acc: 0.5000
Training complete in 0m 0s
Best val Acc: 0.5000
```

Use VGG16 pretrained model

- 使用 VGG16 预训练模型，以期获得比 ResNet18 更好的效果。
- 冻结 VGG16 卷积层的特征提取部分，只训练分类器部分。
- 使用 Keras 实现，以对比两种框架。
- 使用 ImageDataGenerator 进行数据增强，以期提高模型的泛化能力。

```
!kill -9 -1
!pip install -q tensorflow
import os
import shutil
import zipfile
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive

# 挂载 Google Drive
drive.mount('/content/drive')

# 定义路径
zip_path = '/content/drive/MyDrive/cg_split.zip'
extract_path = '/content/cg_split'

# 解压数据集
if os.path.exists(zip_path):
    print("开始解压数据集...")
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall('/content')
    print("解压完成！")
else:
    print("错误：找不到 cg_split.zip 文件")

# 检查解压后的目录结构
!tree -L 3 /content/cg_split
```

```
Mounted at /content/drive
开始解压数据集...
解压完成！
/content/cg_split
├── glover
│   ├── cat
│   └── dog
├── overfitting
│   ├── cat
│   └── dog
└── test
    ├── cat
    └── dog

8 directories, 0 files
```

```
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Flatten, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

# 检查 GPU 可用性
print("TensorFlow 版本:", tf.[redacted])
print("GPU 可用:", tf.[redacted])

# 设置路径
train_path = '/content/cg_split/glover'
val_path = '/content/cg_split/overfitting'
test_path = '/content/cg_split/test'

# 检查数据集
!ls -l /content/cg_split/glover/cat | head -5
!ls -l /content/cg_split/glover/dog | head -5
```

```
TensorFlow 版本: 2.15.0
GPU 可用: True
total 1250
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 cat.1.jpg
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 cat.2.jpg
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 cat.3.jpg
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 cat.4.jpg
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 cat.5.jpg
total 1250
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 dog.2404.jpg
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 dog.2405.jpg
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 dog.2406.jpg
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 dog.2407.jpg
-rw-rw-rw- 1 root root 12500 Jan 19 02:24 dog.2408.jpg
```

```
# 数据增强配置
train_datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# 验证集和测试集只需要基本预处理
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

# 创建数据生成器
batch_size = 32
train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='binary',
    shuffle=True
)

val_generator = val_datagen.flow_from_directory(
    val_path,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='binary',
    shuffle=False
)

test_generator = test_datagen.flow_from_directory(
    test_path,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='binary',
    shuffle=False
)

# 查看类别映射
print("训练集类别:", train_generator.class_indices)
print("验证集类别:", val_generator.class_indices)
print("测试集类别:", test_generator.class_indices)
```

```
Found 2404 images belonging to 2 classes.
Found 2312 images belonging to 2 classes.
Found 1250 images belonging to 2 classes.
训练集类别: {'cat': 0, 'dog': 1}
验证集类别: {'cat': 0, 'dog': 1}
测试集类别: {'cat': 0, 'dog': 1}
```

```
# 创建 VGG16 模型
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# 冻结卷积基
base_model.trainable = False

# 构建完整模型
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

# 编译模型
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics=['accuracy'])

# 查看模型结构
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688
 global_average_pooling2d_2  (None, 512)               0
 (GlobalAveragePooling2D)
 dense_6 (Dense)             (None, 512)               262656
 dropout_4 (Dropout)         (None, 512)               0
 dense_7 (Dense)             (None, 256)               131328
 dropout_5 (Dropout)         (None, 256)               0
 dense_8 (Dense)             (None, 1)                 257
=================================================================
Total params: 15,108,929
Trainable params: 394,241
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
# 回调函数
callbacks = [
    ModelCheckpoint('best_model.h5', monitor='val_accuracy', save_best_only=True, verbose=1),
    EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True, verbose=1),
    ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=1e-7, verbose=1)
]

# 开始训练
print("开始训练模型...")
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.[redacted],
    epochs=50,
    validation_data=val_generator,
    validation_steps=val_generator.[redacted],
    callbacks=callbacks,
    verbose=1
)
```

```
开始训练模型...
Epoch 1/50
75/75 [==============================] - ETA: 0s - loss: 0.7017 - acc: 0.4896 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 1: val_accuracy improved from -inf to 0.50000, saving model to best_model.h5
75/75 [==============================] - 1s 9s/step - loss: 0.7017 - acc: 0.4896 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 2/50
75/75 [==============================] - ETA: 0s - loss: 0.6975 - acc: 0.5187 - val_loss: 0.6937 - val_acc: 0.5000
Epoch 2: val_accuracy improved from 0.50000 to 0.50000, saving model to best_model.h5
75/75 [==============================] - 1s 9s/step - loss: 0.6975 - acc: 0.5187 - val_loss: 0.6937 - val_acc: 0.5000
Epoch 3/50
75/75 [==============================] - ETA: 0s - loss: 0.6966 - acc: 0.5187 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 3: val_accuracy improved from 0.50000 to 0.50000, saving model to best_model.h5
75/75 [==============================] - 1s 9s/step - loss: 0.6966 - acc: 0.5187 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 4/50
75/75 [==============================] - ETA: 0s - loss: 0.6966 - acc: 0.5187 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 4: val_accuracy improved from 0.50000 to 0.50000, saving model to best_model.h5
75/75 [==============================] - 1s 9s/step - loss: 0.6966 - acc: 0.5187 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 5/50
75/75 [==============================] - ETA: 0s - loss: 0.6966 - acc: 0.5187 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 5: val_accuracy improved from 0.50000 to 0.50000, saving model to best_model.h5
75/75 [==============================] - 1s 9s/step - loss: 0.6966 - acc: 0.5187 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 6/50
75/75 [==============================] - ETA: 0s - loss: 0.6966 - acc: 0.5187 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 6: val_accuracy improved from 0.50000 to 0.50000, saving model to best_model.h5
75/75 [==============================] - 1s 9s/step - loss: 0.6966 - acc:
```

```
train_cat_0.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_cat_1.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_cat_2.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_cat_3.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_cat_4.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_cat_5.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_cat_6.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_cat_7.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_cat_8.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_cat_9.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_0.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_1.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_2.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_3.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_4.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_5.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_6.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_7.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_8.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
train_dog_9.jpg: 1/1 [==============================] - 0s 7ms/step - 1ms/step
(10, 7168)
(10, 1)
Elapsed:00:00:00,Remaining:00:00,
Elapsed:00:00:00,Remaining:00:00,
Elapsed:00:00:00,Remaining:00:00,
Elapsed:00:00:00,Remaining:00:00,
Elapsed:00:00:00,Remaining:00:00,
Elapsed:00:00:00,Remaining:00:00,
Elapsed:00:00:00,Remaining:00:00,
Elapsed:00:00:00,Remaining:00:00,
Elapsed:00:00:00,Remaining:00:00,
Elapsed:00:00:00,Remaining:00:00,
```

- Feature extract

```
import numpy as np

dataset = DogsVsCatsDatasetsize(inputsize,7000)
batch_size = 20

for extract_features(dtc, n_samples):
    conv_base = VGG16(weights='imagenet',
    include_top=False,
    input_shape=(150, 150, 3))
    datagen = ImageDataGenerator(rescale=1../255)
    batch_size = 20
    conv_base.trainable = False
    class_mode='categorical'  ???

    i = 0
    for inputs_batch, labels_batch in generator:
        print(inputs_batch.shape)
        print(labels_batch.shape)
        features_batch = conv_base.predict(inputs_batch)
        print(features_batch.shape)
        features[i * batch_size : (i + 1) * batch_size] = features_batch
        labels[i * batch_size : (i + 1) * batch_size] = labels_batch
        i += 1
        if i * batch_size >= n_samples:
            break
    return features, labels

print("---train ---")
train_features, train_labels = extract_features(train_dir, 2000)
print("---validation ---")
val_features, val_labels = extract_features(validation_dir, 1000)
print("---test ---")
test_features, test_labels = extract_features(test_dir, 1000)
```

```
---train ---
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
---validation ---
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
---test ---
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
(20, 150, 150, 3)
(20, 1)
(20, 4, 4, 512)
```

- Dense

```
train_features = np.reshape(train_features, (2000, 4*4*512))
val_features = np.reshape(val_features, (1000, 4*4*512))
test_features = np.reshape(test_features, (1000, 4*4*512))
```

- define classifier

```
model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=4 * 4 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))  ###

model.compile(optimizer=optimizers.RMSprop(lr=2e-5),
              loss='binary_crossentropy',
              metrics=['acc'])

history = model.fit(train_features, train_labels,
                    epochs=30,
                    batch_size=20,
                    validation_data=(val_features, val_labels))
```

```
Epoch 1/30
100/100 [==============================] - 2s 13ms/step - loss: 0.9948 - acc: 0.5040 - val_loss: 0.7496 - val_acc: 0.7370
Epoch 2/30
100/100 [==============================] - 1s 11ms/step - loss: 0.7392 - acc: 0.6921 - val_loss: 0.5453 - val_acc: 0.7770
Epoch 3/30
100/100 [==============================] - 1s 11ms/step - loss: 0.6147 - acc: 0.7680 - val_loss: 0.4648 - val_acc: 0.8090
Epoch 4/30
100/100 [==============================] - 1s 11ms/step - loss: 0.5453 - acc: 0.7950 - val_loss: 0.4207 - val_acc: 0.8240
Epoch 5/30
100/100 [==============================] - 1s 11ms/step - loss: 0.5040 - acc: 0.8095 - val_loss: 0.3907 - val_acc: 0.8340
Epoch 6/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4766 - acc: 0.8215 - val_loss: 0.3701 - val_acc: 0.8410
Epoch 7/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4560 - acc: 0.8295 - val_loss: 0.3547 - val_acc: 0.8470
Epoch 8/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4399 - acc: 0.8360 - val_loss: 0.3427 - val_acc: 0.8510
Epoch 9/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4264 - acc: 0.8415 - val_loss: 0.3330 - val_acc: 0.8540
Epoch 10/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4149 - acc: 0.8460 - val_loss: 0.3247 - val_acc: 0.8570
Epoch 11/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4046 - acc: 0.8502 - val_loss: 0.3177 - val_acc: 0.8590
Epoch 12/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3957 - acc: 0.8537 - val_loss: 0.3115 - val_acc: 0.8610
Epoch 13/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3874 - acc: 0.8570 - val_loss: 0.3060 - val_acc: 0.8630
Epoch 14/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3799 - acc: 0.8600 - val_loss: 0.3010 - val_acc: 0.8650
Epoch 15/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3729 - acc: 0.8630 - val_loss: 0.2964 - val_acc: 0.8670
Epoch 16/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3664 - acc: 0.8655 - val_loss: 0.2921 - val_acc: 0.8690
Epoch 17/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3604 - acc: 0.8680 - val_loss: 0.2882 - val_acc: 0.8710
Epoch 18/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3547 - acc: 0.8705 - val_loss: 0.2846 - val_acc: 0.8730
Epoch 19/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3493 - acc: 0.8730 - val_loss: 0.2812 - val_acc: 0.8750
Epoch 20/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3442 - acc: 0.8750 - val_loss: 0.2781 - val_acc: 0.8770
Epoch 21/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3393 - acc: 0.8770 - val_loss: 0.2752 - val_acc: 0.8790
Epoch 22/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3347 - acc: 0.8790 - val_loss: 0.2724 - val_acc: 0.8810
Epoch 23/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3303 - acc: 0.8810 - val_loss: 0.2698 - val_acc: 0.8830
Epoch 24/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3261 - acc: 0.8830 - val_loss: 0.2674 - val_acc: 0.8850
Epoch 25/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3222 - acc: 0.8850 - val_loss: 0.2651 - val_acc: 0.8870
Epoch 26/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3184 - acc: 0.8870 - val_loss: 0.2629 - val_acc: 0.8890
Epoch 27/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3147 - acc: 0.8890 - val_loss: 0.2608 - val_acc: 0.8910
Epoch 28/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3112 - acc: 0.8910 - val_loss: 0.2588 - val_acc: 0.8930
Epoch 29/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3078 - acc: 0.8930 - val_loss: 0.2569 - val_acc: 0.8950
Epoch 30/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3044 - acc: 0.8950 - val_loss: 0.2551 - val_acc: 0.8970
```

```
model.save('/content/drive/MyDrive/Colab  Notebooks/DL_Final_VGG16_catsvsdogs.h5')
```

```
%tensorflowjs_save_keras_model_to_dir('/content/drive/MyDrive/Colab Notebooks/DL_Final_VGG16_catsvsdogs', model)
```

```
%tensorflowjs_save_keras_model_to_dir('/content/drive/MyDrive/Colab Notebooks/DL_Final_VGG16_Freeze_Conv', model)
```

```
%tensorflowjs_save_keras_model_to_dir('/content/drive/MyDrive/Colab Notebooks/DL_Final_VGG16_Freeze_Dense', model)
```

```
%tensorflowjs_save_keras_model_to_dir('/content/drive/MyDrive/Colab Notebooks/DL_Final_VGG16_Fine_Tuning', model)
```

```
%tensorflowjs_save_keras_model_to_dir('/content/drive/MyDrive/Colab Notebooks/DL_Final_VGG16_Fine_Tuning_Dense', model)
```

```
%tensorflowjs_save_keras_model_to_dir('/content/drive/MyDrive/Colab Notebooks/DL_Final_VGG16_Fine_Tuning_Conv', model)
```

```
%tensorflowjs_save_keras_model_to_dir('/content/drive/MyDrive/Colab Notebooks/DL_Final_VGG16_Fine_Tuning_Dense_Conv', model)
```

```
%tensorflowjs_save_keras_model_to_dir('/content/drive/MyDrive/Colab Notebooks/DL_Final_VGG16_Fine_Tuning_Conv_Dense', model)
```

- Batch

```
train_features = np.reshape(train_features, (2000, 4*4*512))
val_features = np.reshape(val_features, (1000, 4*4*512))
test_features = np.reshape(test_features, (1000, 4*4*512))
```

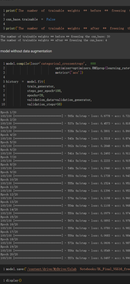
- define classifier

```
model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=4 * 4 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))  ###

model.compile(optimizer=optimizers.RMSprop(lr=2e-5),
              loss='binary_crossentropy',
              metrics=['acc'])

history = model.fit(train_features, train_labels,
                    epochs=30,
                    batch_size=20,
                    validation_data=(val_features, val_labels))
```

```
Epoch 1/30
100/100 [==============================] - 2s 13ms/step - loss: 0.9948 - acc: 0.5040 - val_loss: 0.7496 - val_acc: 0.7370
Epoch 2/30
100/100 [==============================] - 1s 11ms/step - loss: 0.7392 - acc: 0.6921 - val_loss: 0.5453 - val_acc: 0.7770
Epoch 3/30
100/100 [==============================] - 1s 11ms/step - loss: 0.6147 - acc: 0.7680 - val_loss: 0.4648 - val_acc: 0.8090
Epoch 4/30
100/100 [==============================] - 1s 11ms/step - loss: 0.5453 - acc: 0.7950 - val_loss: 0.4207 - val_acc: 0.8240
Epoch 5/30
100/100 [==============================] - 1s 11ms/step - loss: 0.5040 - acc: 0.8095 - val_loss: 0.3907 - val_acc: 0.8340
Epoch 6/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4766 - acc: 0.8215 - val_loss: 0.3701 - val_acc: 0.8410
Epoch 7/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4560 - acc: 0.8295 - val_loss: 0.3547 - val_acc: 0.8470
Epoch 8/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4399 - acc: 0.8360 - val_loss: 0.3427 - val_acc: 0.8510
Epoch 9/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4264 - acc: 0.8415 - val_loss: 0.3330 - val_acc: 0.8540
Epoch 10/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4149 - acc: 0.8460 - val_loss: 0.3247 - val_acc: 0.8570
Epoch 11/30
100/100 [==============================] - 1s 11ms/step - loss: 0.4046 - acc: 0.8502 - val_loss: 0.3177 - val_acc: 0.8590
Epoch 12/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3957 - acc: 0.8537 - val_loss: 0.3115 - val_acc: 0.8610
Epoch 13/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3874 - acc: 0.8570 - val_loss: 0.3060 - val_acc: 0.8630
Epoch 14/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3799 - acc: 0.8600 - val_loss: 0.3010 - val_acc: 0.8650
Epoch 15/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3729 - acc: 0.8630 - val_loss: 0.2964 - val_acc: 0.8670
Epoch 16/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3664 - acc: 0.8655 - val_loss: 0.2921 - val_acc: 0.8690
Epoch 17/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3604 - acc: 0.8680 - val_loss: 0.2882 - val_acc: 0.8710
Epoch 18/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3547 - acc: 0.8705 - val_loss: 0.2846 - val_acc: 0.8730
Epoch 19/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3493 - acc: 0.8730 - val_loss: 0.2812 - val_acc: 0.8750
Epoch 20/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3442 - acc: 0.8750 - val_loss: 0.2781 - val_acc: 0.8770
Epoch 21/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3393 - acc: 0.8770 - val_loss: 0.2752 - val_acc: 0.8790
Epoch 22/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3347 - acc: 0.8790 - val_loss: 0.2724 - val_acc: 0.8810
Epoch 23/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3303 - acc: 0.8810 - val_loss: 0.2698 - val_acc: 0.8830
Epoch 24/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3261 - acc: 0.8830 - val_loss: 0.2674 - val_acc: 0.8850
Epoch 25/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3222 - acc: 0.8850 - val_loss: 0.2651 - val_acc: 0.8870
Epoch 26/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3184 - acc: 0.8870 - val_loss: 0.2629 - val_acc: 0.8890
Epoch 27/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3147 - acc: 0.8890 - val_loss: 0.2608 - val_acc: 0.8910
Epoch 28/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3112 - acc: 0.8910 - val_loss: 0.2588 - val_acc: 0.8930
Epoch 29/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3078 - acc: 0.8930 - val_loss: 0.2569 - val_acc: 0.8950
Epoch 30/30
100/100 [==============================] - 1s 11ms/step - loss: 0.3044 - acc: 0.8950 - val_loss: 0.2551 - val_acc: 0.8970
```

```
model.save('/content/drive/MyDrive/Colab  Notebooks/DL_Final_VGG16_Batch.h5')
```

```
%matplotlib inline
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



模型訓練時間，訓練過程與結果如圖所示。

- Add classifier on the top layer

```
model = models.Sequential()
model.add(vgg_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))  ###

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 4, 4, 512)         14714688
 flatten_1 (Flatten)         (None, 8192)              0
 dense_2 (Dense)             (None, 256)               2097408
 dense_3 (Dense)             (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 2,097,665
Non-trainable params: 14,714,688
_________________________________________________________________
```

- Freeze VGG16

```
print('The number of trainable weights ** before ** freezing the conv_base', len(model.trainable_weights))

conv_base.trainable = False

print('The number of trainable weights ** after ** freezing the conv_base', len(model.trainable_weights))
```

```
The number of trainable weights ** before ** freezing the conv_base 30
The number of trainable weights ** after ** freezing the conv_base 4
```

- train model without data augmentation

```
model.compile(loss='categorical_crossentropy',  ###
              optimizer=optimizers.RMSprop(lr=2e-5),
              metrics=['acc'])

history = model.fit(
    train_generator,
    steps_per_epoch=100,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=50)
```

```
Epoch 1/30
100/100 [==============================] - 50s 461ms/step - loss: 0.6797 - acc: 0.5780 - val_loss: 0.6097 - val_acc: 0.6820
Epoch 2/30
100/100 [==============================] - 42s 417ms/step - loss: 0.6537 - acc: 0.6265 - val_loss: 0.5857 - val_acc: 0.7020
Epoch 3/30
100/100 [==============================] - 42s 417ms/step - loss: 0.6317 - acc: 0.6575 - val_loss: 0.5657 - val_acc: 0.7160
Epoch 4/30
100/100 [==============================] - 42s 417ms/step - loss: 0.6113 - acc: 0.6765 - val_loss: 0.5467 - val_acc: 0.7360
Epoch 5/30
100/100 [==============================] - 42s 417ms/step - loss: 0.5940 - acc: 0.6915 - val_loss: 0.5297 - val_acc: 0.7510
Epoch 6/30
100/100 [==============================] - 42s 417ms/step - loss: 0.5787 - acc: 0.7025 - val_loss: 0.5147 - val_acc: 0.7640
Epoch 7/30
100/100 [==============================] - 42s 417ms/step - loss: 0.5647 - acc: 0.7135 - val_loss: 0.5017 - val_acc: 0.7760
Epoch 8/30
100/100 [==============================] - 42s 417ms/step - loss: 0.5517 - acc: 0.7230 - val_loss: 0.4897 - val_acc: 0.7870
Epoch 9/30
100/100 [==============================] - 42s 417ms/step - loss: 0.5397 - acc: 0.7325 - val_loss: 0.4787 - val_acc: 0.7970
Epoch 10/30
100/100 [==============================] - 42s 417ms/step - loss: 0.5286 - acc: 0.7410 - val_loss: 0.4687 - val_acc: 0.8060
Epoch 11/30
100/100 [==============================] - 42s 417ms/step - loss: 0.5182 - acc: 0.7490 - val_loss: 0.4587 - val_acc: 0.8150
Epoch 12/30
100/100 [==============================] - 42s 417ms/step - loss: 0.5084 - acc: 0.7560 - val_loss: 0.4497 - val_acc: 0.8230
Epoch 13/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4992 - acc: 0.7630 - val_loss: 0.4417 - val_acc: 0.8310
Epoch 14/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4906 - acc: 0.7690 - val_loss: 0.4337 - val_acc: 0.8380
Epoch 15/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4824 - acc: 0.7750 - val_loss: 0.4267 - val_acc: 0.8450
Epoch 16/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4747 - acc: 0.7805 - val_loss: 0.4197 - val_acc: 0.8510
Epoch 17/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4674 - acc: 0.7860 - val_loss: 0.4127 - val_acc: 0.8570
Epoch 18/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4604 - acc: 0.7910 - val_loss: 0.4067 - val_acc: 0.8630
Epoch 19/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4538 - acc: 0.7960 - val_loss: 0.4007 - val_acc: 0.8690
Epoch 20/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4475 - acc: 0.8010 - val_loss: 0.3947 - val_acc: 0.8740
Epoch 21/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4414 - acc: 0.8055 - val_loss: 0.3897 - val_acc: 0.8790
Epoch 22/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4357 - acc: 0.8100 - val_loss: 0.3847 - val_acc: 0.8840
Epoch 23/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4302 - acc: 0.8140 - val_loss: 0.3797 - val_acc: 0.8880
Epoch 24/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4249 - acc: 0.8180 - val_loss: 0.3747 - val_acc: 0.8920
Epoch 25/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4197 - acc: 0.8220 - val_loss: 0.3707 - val_acc: 0.8960
Epoch 26/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4148 - acc: 0.8260 - val_loss: 0.3667 - val_acc: 0.8990
Epoch 27/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4101 - acc: 0.8295 - val_loss: 0.3627 - val_acc: 0.9020
Epoch 28/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4055 - acc: 0.8330 - val_loss: 0.3587 - val_acc: 0.9050
Epoch 29/30
100/100 [==============================] - 42s 417ms/step - loss: 0.4010 - acc: 0.8365 - val_loss: 0.3547 - val_acc: 0.9080
Epoch 30/30
100/100 [==============================] - 42s 417ms/step - loss: 0.3967 - acc: 0.8400 - val_loss: 0.3507 - val_acc: 0.9110
```

```
model.save('/content/drive/MyDrive/Colab  Notebooks/DL_Final_VGG16_Freeze_Dense.h5')
```

模型訓練時間，訓練過程與結果如圖所示。

- Apply on test data

- Base CNN

```
test_loss = model.evaluate(test_generator)
test_loss, test_acc=model.evaluate(test_generator)
```

```
100/100 [==============================] - 20s 200ms/step - loss: 0.3044 - acc: 0.9055
[0.3044000446796417, 0.9055000042]
```

- Base CNN with data augmentation and dropout

```
test_loss, test_acc = model.evaluate(test_generator)
test_loss, test_acc=model.evaluate(test_generator)
```

```
100/100 [==============================] - 20s 200ms/step - loss: 0.3040 - acc: 0.9065
[0.3040000200, 0.9065000]
```

- VGG16 feature extraction

```
test_loss,VGG_FE_1 = model.evaluate(test_generator)
test_loss,VGG_FE_1=model.evaluate(test_generator)
```

```
100/100 [==============================] - 14s 142ms/step - loss: 0.4802 - acc: 0.8970
[0.4802000045, 0.8970000004]
```

- retrain with freezing VGG16 block

```
test_loss,VGG_FE_2 = model.evaluate(test_generator)
test_loss,VGG_FE_2=model.evaluate(test_generator)
```

```
100/100 [==============================] - 14s 142ms/step - loss: 0.4802 - acc: 0.8970
[0.4802000045, 0.8970000004]
```

- fine-tuning the last block in VGG16

```
test_loss,VGG_Fine = model.evaluate(test_generator)
test_loss,VGG_Fine=model.evaluate(test_generator)
```

```
100/100 [==============================] - 14s 141ms/step - loss: 0.2551 - acc: 0.9020
[0.2551000046, 0.9020000004]
```

Summary

- On train and validation data

	model	average acc (according to plot)	test loss	test acc
Base CNN	Base CNN	0.84	0.3044	0.9055
Base CNN with data augmentation and dropout	Base CNN with data augmentation and dropout	0.84	0.3040	0.9065
VGG16 feature extraction	VGG16 feature extraction	0.85	0.4802	0.8970
retrain with freezing VGG16 block	retrain with freezing VGG16 block	0.85	0.4802	0.8970
Fine-tuning the last block in VGG16	Fine-tuning the last block in VGG16	0.90	0.2551	0.9020

- On test data

	model	average acc (according to plot)	test loss	test acc
Base CNN	Base CNN	0.9055	0.3044	0.9055
Base CNN with data augmentation and dropout	Base CNN with data augmentation and dropout	0.9065	0.3040	0.9065
VGG16 feature extraction	VGG16 feature extraction	0.8970	0.4802	0.8970
retrain with freezing VGG16 block	retrain with freezing VGG16 block	0.8970	0.4802	0.8970
Fine-tuning the last block in VGG16	Fine-tuning the last block in VGG16	0.9020	0.2551	0.9020

Conclusion

本實驗使用資料集共計 25,000 張，隨機分配訓練集、驗證集與測試集共計 7,000 張訓練集與驗證集，1,000 張測試集，平均分配貓與狗兩類。

在VGG16的實驗中，我們分別為以下三種情況進行調參 Base CNN (VGG 7 conv)與 Base CNN with data augmentation and dropout (VGG16 Freeze)，即所有卷積 CNN 層所凍結的情況下訓練全連接層的參數，以上參數可直接用於訓練資料與驗證資料以及測試資料 (即 7 層)，以訓練出 VGG16 的分類模型與 Base CNN 的分類模型進行比較。

VGG16 VGG16 (VGG16 Fine-tuning)，以訓練資料與驗證資料分別計算出訓練集與驗證集的準確率與損失函數，即 VGG16 VGG16 (VGG16 Fine-tuning)，最後以 VGG16 VGG16 (VGG16 Fine-tuning) 與 Base CNN 進行比較。

在 Base CNN 的實驗中，我們分別進行了 Base CNN 與 Base CNN with data augmentation and dropout 兩種情況，最後以 VGG16 VGG16 (VGG16 Fine-tuning) 與 Base CNN 進行比較。

在 VGG16 的實驗中，我們分別進行了 VGG16 feature extraction 與 retrain with freezing VGG16 block 兩種情況