作业一：基于深度神经网络的猫狗图像分类

一、　实验目的

1. 掌握卷积神经网络、循环神经网络等深度学习的各项基本技术。
2. 加强对pytorch、tensorflow等深度学习框架的使用能力。

二、　实验要求

1. 任选一个深度学习框架，实现在给定数据集上的基于DNN、CNN、RNN的图片分类模型。

三、　数据集介绍

猫狗分类，即对给定的图片用模型来判断其是猫还是狗，本质上是一个二分类的图像分类任务。本次实验所用的数据集提供在了附件的data.rar中，train和val文件夹下分别是2000张和500张猫狗图片用于训练和测试模型。这些图片都来自于kaggle竞赛数据集，**如果设备充足，有兴趣做完整数据集的**，也可以从下面的链接下载数据集来使用。

四、　任务说明

1. **由于自然语言处理任务普遍要使用词向量来完成，但课程进度还未达到。**因此，我们第一次作业以一个计算机视觉领域里的典型任务来让大家更好地掌握使用深度学习几个基本网络和框架的能力。
2. 请分别使用DNN、CNN、RNN**三种模型**来实现猫狗分类任务，具体网络结构不进行要求，可以根据自己上课的理解来进行设计。
3. 由于猫狗图片数目完全一致，因此测评指标采用各个类别的准确率即可。实验完 毕后，请提交自己的代码以及一份实验报告（不要求具体长度），报告中可以说明自己使用的网络结构、所得到的结果，以及自己的一些问题和思考（如果有）等。

一、基础设置

包导入

```
from tensorflow.keras.models import Sequential,Model
from tensorflow.keras.layers import Dense, Flatten, Dropout,
Activation,Bidirectional,SimpleRNN
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
img_to_array, load_img
from livelossplot.keras import PlotLossesCallback
import tensorflow as tf
from keras import regularizers
import numpy as np
import matplotlib.image as mpimg
from keras.optimizers import adam_v2
import matplotlib.pyplot as plt
from keras.layers import Activation, Dropout, Flatten, Dense,Input,LSTM,
GlobalMaxPooling2D, Conv2D, MaxPooling2D,ConvLSTM1D
```

基础配置

```
training_data_dir = '../data/train_set'
validation_data_dir = "../data/valid_set"
test_data_dir = '../data/test_set'
RNN_WEIGHTS_DOGS_VS_CATS = r'output/RNN/cats_vs_dogs_rnn.h5'

IMAGE_SIZE = 256
IMAGE_WIDTH, IMAGE_HEIGHT = IMAGE_SIZE, IMAGE_SIZE
nb_train_samples = 1500
nb_test_samples = 500
EPOCHS = 20
BATCH_SIZE = 16
LR = 0.001
TRAINING_LOGS_FILE = "output/RNN/training_logs_rnn.csv"
MODEL_SUMMARY_FILE = "output/RNN/model_summary_rnn.txt"
```

二、数据预处理

```
# 数据增强
training_data_generator = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True)
validation_data_generator = ImageDataGenerator(rescale=1. / 255)
test_data_generator = ImageDataGenerator(rescale=1. / 255)

# 数据准备
training_generator = training_data_generator.flow_from_directory(
    training_data_dir,
    target_size=(IMAGE_WIDTH, IMAGE_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode="binary")
validation_generator = validation_data_generator.flow_from_directory(
    validation_data_dir,
    target_size=(IMAGE_WIDTH, IMAGE_HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode="binary")
test_generator = test_data_generator.flow_from_directory(
    test_data_dir,
    target_size=(IMAGE_WIDTH, IMAGE_HEIGHT),
    batch_size=1,
    class_mode="binary",
    shuffle=False)
```

二、网络结构

DNN

```
input_shape = (IMAGE_WIDTH, IMAGE_HEIGHT, 3)
model = Sequential()
model.add(Flatten(input_shape=input_shape))
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=
['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten (Flatten)            (None, 196608)            0         
_________________________________________________________________
dense (Dense)                (None, 256)               50331904  
_________________________________________________________________
activation (Activation)      (None, 256)               0         
_________________________________________________________________
dropout (Dropout)            (None, 256)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896     
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256      
_________________________________________________________________
activation_2 (Activation)    (None, 64)                0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65        
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0         
=================================================================
Total params: 50,373,121
Trainable params: 50,373,121
Non-trainable params: 0
```

CNN

```
input_shape = (IMAGE_WIDTH, IMAGE_HEIGHT, 3)
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='rmsprop', metrics=
['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 254, 254, 32)      896       
_________________________________________________________________
activation (Activation)      (None, 254, 254, 32)      0         
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 127, 127, 32)      0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 125, 125, 64)      18496     
_________________________________________________________________
activation_1 (Activation)    (None, 125, 125, 64)      0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 62, 62, 64)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 128)       73856     
_________________________________________________________________
activation_2 (Activation)    (None, 60, 60, 128)       0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 128)       0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 256)       295168    
_________________________________________________________________
activation_3 (Activation)    (None, 28, 28, 256)       0         
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 256)       0         
_________________________________________________________________
flatten (Flatten)            (None, 50176)             0         
_________________________________________________________________
dense (Dense)                (None, 256)               12845312  
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0         
_________________________________________________________________
dropout (Dropout)            (None, 256)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257       
_________________________________________________________________
```

```
activation_5 (Activation)    (None, 1)                 0
=================================================================
Total params: 13,233,985
Trainable params: 13,233,985
Non-trainable params: 0
```

RNN（在CNN层中间添加LSTM层）

```
input_shape = (IMAGE_WIDTH, IMAGE_HEIGHT, 3)
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(ConvLSTM1D(256,3,activation='relu',return_sequences=True))
model.add(ConvLSTM1D(128,3,activation='relu',return_sequences=True))
model.add(ConvLSTM1D(64,3,activation='relu',return_sequences=True))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
adam = adam_v2.Adam(learning_rate=LR)
model.compile(optimizer=adam,loss='categorical_crossentropy', metrics=
['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 254, 254, 32)      896
_________________________________________________________________
activation (Activation)      (None, 254, 254, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 127, 127, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 125, 125, 64)      18496
_________________________________________________________________
activation_1 (Activation)    (None, 125, 125, 64)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 62, 62, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 128)       73856
_________________________________________________________________
activation_2 (Activation)    (None, 60, 60, 128)       0
_________________________________________________________________
```

```
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 128)       0

_________________________________________________________________
conv_lst_m1d (ConvLSTM1D)    (None, 30, 28, 256)       1180672

_________________________________________________________________
conv_lst_m1d_1 (ConvLSTM1D)  (None, 30, 26, 128)       590336

_________________________________________________________________
conv_lst_m1d_2 (ConvLSTM1D)  (None, 30, 24, 64)        147712

_________________________________________________________________
flatten (Flatten)            (None, 46080)             0

_________________________________________________________________
dense (Dense)                (None, 256)               11796736

_________________________________________________________________
activation_3 (Activation)    (None, 256)               0

_________________________________________________________________
dropout (Dropout)            (None, 256)               0

_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257

_________________________________________________________________
activation_4 (Activation)    (None, 1)                 0
=================================================================
Total params: 13,808,961
Trainable params: 13,808,961
Non-trainable params: 0
```

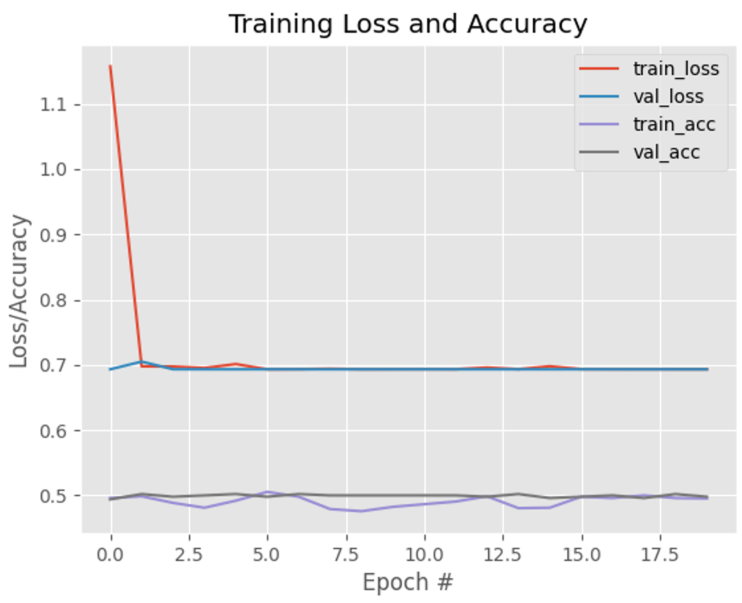
三、模型训练

```
H = model.fit_generator(
    training_generator,
    steps_per_epoch=len(training_generator.filenames) // BATCH_SIZE,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=len(validation_generator.filenames) // BATCH_SIZE,
    verbose=1)
model.save_weights(RNN_WEIGHTS_DOGS_VS_CATS)
```

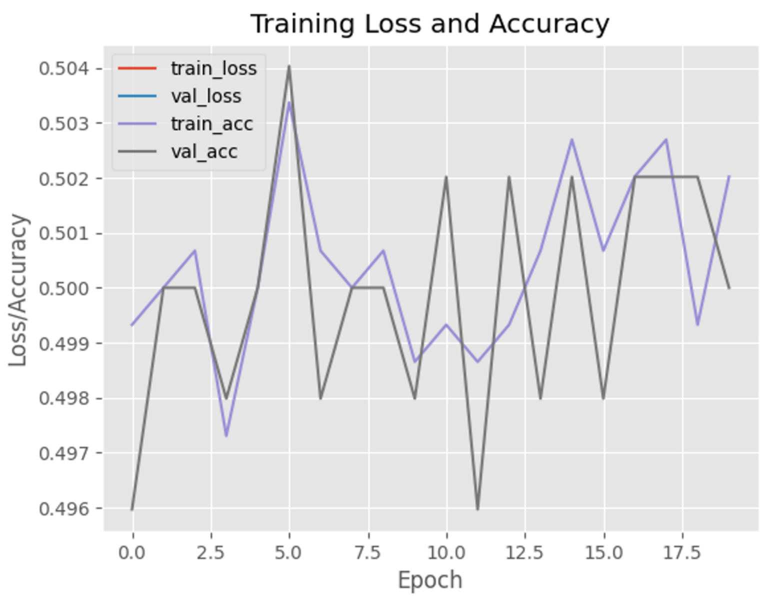
四、效果可视化

```
N = np.arange(0, EPOCHS)
plt.style.use("ggplot")
plt.figure()
plt.plot(N, H.history["loss"], label="train_loss")
plt.plot(N, H.history["val_loss"], label="val_loss")
plt.plot(N, H.history["accuracy"], label="train_acc")
plt.plot(N, H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig('output/(D\C)RNN/(c/d)rnn_plot.png')
plt.show()
```

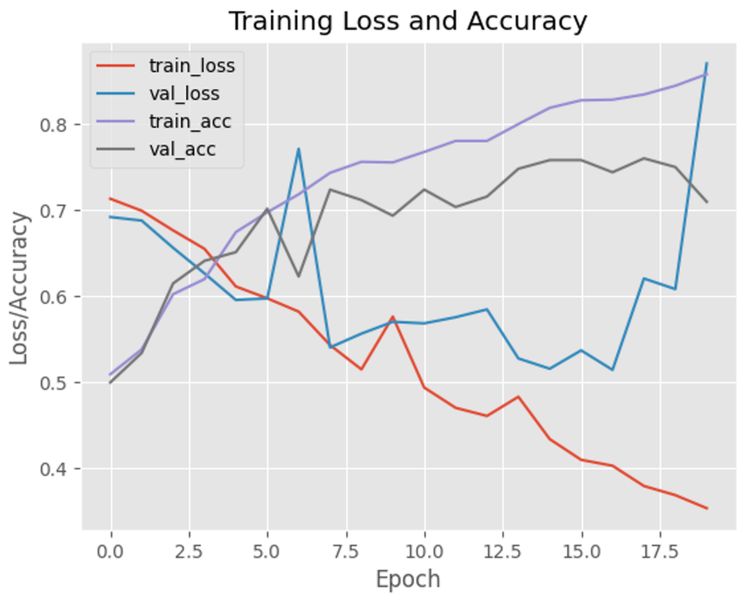
DNN：损失率在0.7左右，准确率在0.5左右，效果较差。



RNN：准确率大约在0.5左右，效果跟DNN差不多。



CNN：损失率在0.5上下波动，准确率在0.7上下波动，较DNN和RNN有明显提升。



Training Loss and Accuracy
train_loss
val_loss
train_acc
val_acc
Loss/Accuracy
Epoch
0.4
0.5
0.6
0.7
0.8
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5