Q3 readme

a. Data Preparation and Loading

- **Data Reading:** By using the to_categorical function, we can convert the labels into a one-hot encoded format, which is crucial for multi-class classification tasks. So train_tag DataFrame is created from tarin_tag.txt, containing video file names and their associated tags.
- **Video Data Processing:** Define a function process_video to process each video file
  - Open the video file using cv2.VideoCapture.
  - Read images frame by frame from the video and resize each frame to 60x60 pixels
  - To extract a specific number of frames, we operate differently based on the total number of frames in the video. If the total number of frames is fewer than frame_count, the function repeatedly adds the last frame until the length of frames equals frame_count. If the number of frames exceeds frame_count, it utilizes NumPy's linspace and round functions to select evenly spaced frames, ensuring that exactly frame_count number of frames are extracted.
  - Finally, return an array containing the the processed frames, resized and formatted

```
def process_video(file_path, frame_size, frame_count):
    cap = cv2.VideoCapture(file_path)
    frames = []
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frame = cv2.resize(frame, frame_size, interpolation=cv2.INTER_CUBIC)
        frames.append(frame)
    cap.release()

    frames = np.array(frames)
    if len(frames) < frame_count:
        last_frame = frames[-1]
        while len(frames) < frame_count:
            frames = np.append(frames, [last_frame], axis=0)
    else:
        frame_indices = np.round(np.linspace(0, len(frames) - 1, frame_count)).astype(int)
        frames = frames[frame_indices]

    return frames
```

- **Dataset Creation:** Loop through all the MP4 files
  - Each video is processed using process_video.
  - The corresponding tags are fetched from train_tag and matched with each video.
  - The videos (x) and their tags (y) are stored in arrays.
- **Train-Test Split**

b. Model Building

- **Model Architecture:** Use a hybrid model combining Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) layers
  - The CNN component is a pre-trained VGG16 model used for feature extraction from

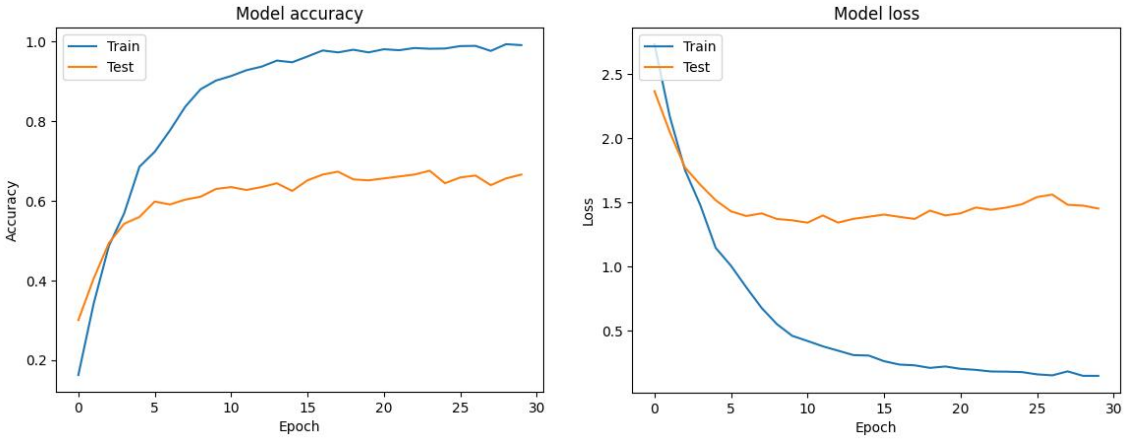
video frames.

- The LSTM layer is used to analyze sequences (video frames in this case), which is crucial for understanding temporal dynamics in videos.
- A fully connected layer with L2 regularization and dropout is added for classification.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDist  (None, 20, 1, 1, 512)     14714688
 ributed)

 time_distributed_1 (TimeDi  (None, 20, 512)           0
 stributed)

 lstm (LSTM)                 (None, 256)               787456

 dense (Dense)               (None, 64)                16448

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 15)                975

=================================================================
Total params: 15519567 (59.20 MB)
Trainable params: 804879 (3.07 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

- **Model Compilation:** The model is compiled with the Adam optimizer and categorical cross-entropy loss function, suitable for multi-class classification.

The accuracy is about 0.67 on val set after 30 epochs. Here are training and validation accuracy/loss plot.



In addition, I found that there is a pre-trained model available for UCF101 - Action Recognition Data Set. It uses VGG16 to extract features from individual frame of the video, the sequence of frame features are then taken into bidirectional LSTM recurrent networks for classifier. The performance is very very well.https://github.com/chen0040/keras-video-classifier

keras-video-classifier-web-api

Keras implementation of video classifiers serving as web

The training data is UCF101 - Action Recognition Data Set. Codes are included that will download the UCF101 if they do not exist (due to their large size) in the demo/very_large_data folder. The download utility codes can be found in keras_video_classifier/library/utility/ucf directory

The video classifiers are defined and implemented in the keras_video_classifier/library directory.

By default the classifiers are trained using video files inside the dataset "UCF-101" located in demo/very_large_data (the videos files will be downloaded if not exist during training). However, the classifiers are generic and can be used to train on any other datasets (just change the data_set_name parameter in its fit() method to other dataset name instead of UCF-101 will allow it to be trained on other video datasets)

The opencv-python is used to extract frames from the videos.

Deep Learning Models

The following deep learning models have been implemented and studied:

- VGG16+LSTM: this approach uses VGG16 to extract features from individual frame of the video, the sequ frame features are then taken into LSTM recurrent networks for classifier.
  - training: demo/vgg16_lstm_train.py
  - predictor: demo/vgg16_lstm_predict.py
  - training: demo/vgg16_lstm_hi_dim_train.py (VGG16 top not included)
  - predictor: demo/vgg16_lstm_hi_dim_predict.py (VGG16 top not included)
- VGG16+Bidirectional LSTM: this approach uses VGG16 to extract features from individual frame of the vi sequence of frame features are then taken into bidirectional LSTM recurrent networks for classifier.
  - training: demo/vgg16_bidirectional_lstm_train.py
  - predictor: demo/vgg16_bidirectional_lstm_predict.py
  - training: demo/vgg16_bidirectional_lstm_hi_dim_train.py (VGG16 top not included)
  - predictor: demo/vgg16_bidirectional_lstm_hi_dim_predict.py (VGG16 top not included)
- Convolutional Network: this approach uses stores frames into the "channels" of input of the CNN which t classify the "image" (video frames stacked in the channels)
  - training: demo/cnn_train.py
  - predictor: demo/cnn_predict.py

The trained models are available in the demo/models/UCF-101 folder (Weight files of two of the trained mod not included as they are too big to upload, they are

- demo/models/UCF-101/vgg16-lstm-hi-dim-weights.h5
- demo/models/UCF-101/vgg16-bidirectional-lstm-hi-dim-weights.h5)

References：

[1] cv2 视频操作：https://blog.csdn.net/weixin_63676550/article/details/128013573