ML Lab3 Report

高健翔 521021910197

一．系统设计

i. 模型设计

采用 CNN-LSTM 模型：

CNN 部分：

对于输入的时空序列数据首先通过卷积神经网络进行特征提取，CNN 捕捉数据的空间局部特征，处理视频不同帧的空间结构数据，在我的 CNN 模型设计中，用到卷积层提取特征，池化层进行维度降低，激活函数加入非线性因素；使用的预训练模型是 reenet101。.

LSTM 部分：

对于输入的序列，CNN 已经将特征提取出来，LSTM 负责处理输入数据的时序特征，对于图像序列，LSTM 将序列中的帧是为时间步，并学习帧之间的时序关系，LSTM 具有门控结构，可以决定记忆或者遗忘的内容，以传递先前时间步的信息。

最终，将 LSTM 的输出作为全连接层的输入，输出维度为类别数量，哪一维度的值越大，表明在该类型的概率越高，取最大的下标作为最终的分类。

ii. 训练方法

对于分类任务，选择交叉熵函数作为 Loss_Function

优化器选择 adam 优化器

自行选择 LR 调整策略，本次实验使用了 ReduceLROnPlateau 和 CosineAnnealingWarmRestarts

重写数据集划分函数，在准备过程中便将数据集分割为 train，val，test 三个模块

自行构造 CNN-LSTM 模型，并参考他人代码构造基于预训练网络的模型

下面是对代码的重构：

```
def transform_data(num_class, batch_sz, num_frame, num_workers, data_dir, transform_fun):
    full_dataset = VideoDataset(data_dir=data_dir, num_frames=num_frame, num_classes=num_class, transform=transform_fun)
    train_val, test = train_test_split(full_dataset, test_size=0.2, random_state=42)
    train_, val_ = train_test_split(train_val, test_size=0.2, random_state=42)
    train_loader = dtl(train_, batch_size=batch_sz, shuffle=True, num_workers=num_workers)
    val_loader = dtl(val_, batch_size=batch_sz, shuffle=True, num_workers=num_workers)
    test_loader = dtl(test, batch_size=batch_sz, shuffle=False, num_workers=num_workers)
    return train_loader, val_loader, test_loader


2 用法  1393429978@qq.com <1393429978@qq.com> *
def split_dataloader(train_data, validation_split=0.2):
    train_ratio = 1 - validation_split
    train_size = int(train_ratio * len(train_data.dataset))
    val_size = len(train_data.dataset) - train_size

    train_dataset, val_dataset = rs(train_data, [train_size, val_size])
    batch_size = train_data.batch_size
    num_workers = train_data.num_workers

    train_data = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size, num_workers=num_workers, drop_last=True)
    val_data = torch.utils.data.DataLoader(val_dataset, batch_size=batch_size, num_workers=num_workers,
                                           drop_last=True)
```

数据分割部分：参考老师的代码，重新写出了将 dataset 分为 train，val，test 三个 dataset 并构造 dataloader 的函数；参考 kaggle 网站代码，使用 torch random_split 函数重新构造分割 dataloader 的代码，当 train 函数的验证集参数为空时，调用该函数从训练集分割出一部分

```python
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description="select model")
    parser.add_argument('--model_select', type=str, default="my_model",
                        help='model select from my_model, pre_trained')
    parser.add_argument('--lr_select', type=str, default="re",
                        help='lr method select from \'RE\' or \'COS\'')
    parser.add_argument('--pt_select', type=str, default="last",
                        help='pt last ? best ? acc')
    parser.add_argument('--train', type=str, default='yes',
                        help="yes or no / train or only test")
    args = parser.parse_args()
    train_, val_, test_, full_ = lab3_data_scratch.transform_data(num_classes, batch_size, num_frames, num_workers,
                                                                  './data', transform)
    model = my_model.MyModel(num_classes=num_classes, hidden_size=128, num_lstm=2) if args.model_select == "my_model" \
        else pretrained_model.ClassificationModel(num_classes=num_classes, hidden_size=128, num_lstm_layers=2)
    loss_fn = nn.CrossEntropyLoss()
    if args.train == 'yes':
        opt = torch.optim.Adam(model.parameters(), lr=1e-4)
        scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(opt, factor=0.1, mode='min', patience=3,
                                                               verbose=True) if args.lr_select == 'RE' \
            else torch.optim.lr_scheduler.CosineAnnealingWarmRestarts(opt, 50, 1, 1e-5)
        model, his = train_eva.train(model, train_, loss_fn, opt, weights=None, epochs=50, validation_data=val_,
                                     save_best_weights_path=best_weights, save_last_weights_path=last_weights,
                                     device=device, validation_split=None, steps_per_epoch=100, scheduler=scheduler)
        visual_history.visualize_history(his)
    else:
        test_ = dtl(full_, batch_size=batch_size, shuffle=True, num_workers=num_workers)
    test_loss, test_acc = train_eva.evaluate(model, weights=last_weights, val_data=test_, loss_fn=loss_fn,
                                             device='cuda', verbose=1)
    print(f'Loss: {test_loss : .3f}, Acc: {test_acc: .3f}')
```

增加 main.py 文件负责参数解析，配置相应的优化器，lr 调整策略和模型选择，然后调用训练，测试函数，最终输出结果

```python
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description="select model")
    parser.add_argument('--model_select', type=str, default="my_model",
                        help='model select from my_model, pre_trained')
    parser.add_argument('--pt_select', type=str, default="last",
                        help='pt last ? best ? acc')
    parser.add_argument('--video_path', type=str, default=None,
                        help="select path")

    args = parser.parse_args()
    assert not (args.video_path is None)
    data = lab3_data_scratch.read_radio(args.video_path, transform, num_frames)
    data = data.unsqueeze(0)
    data = data.to('cuda')
    model = my_model.MyModel(num_classes=num_classes, hidden_size=128, num_lstm=2) if args.model_select == "my_model" \
        else pretrained_model.ClassificationModel(num_classes=num_classes, hidden_size=128, num_lstm_layers=2)
    model.to('cuda')
    model.load_state_dict(torch.load('last_weights.pt'))
    model.eval()
    output = model(data)
    output = output.reshape(-1)
    _, idx = torch.max(output, dim=0)
    print(idx.item())
    print(output)
```

读取单一视频文件，使用模型进行分类

```python
if save_best_weights_path:
    if val_loss < best_loss:
        best_loss = val_loss
        torch.save(model.state_dict(), save_best_weights_path)
        print(f'Saved successfully best weights to:', save_best_weights_path)
if save_acc_path:
    if train_accuracy > best_train_acc and val_acc > best_val_acc:
        best_train_acc = train_accuracy
        best_val_acc = val_acc
        torch.save(model.state_dict(), save_acc_path)
        print(f'Saved successfully best acc weights to', save_acc_path)
history['val_loss'].append(float(val_loss))
history['val_acc'].append(float(val_acc))
```

增加了一个根据训练集，验证集准确率来保存模型参数的部分

```python
def __init__(self, num_classes, hidden_size, num_lstm=2):
    super().__init__()
    self.Conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1)
    self.Relu1 = nn.ReLU(inplace=True)
    self.MaxPooling1 = nn.MaxPool2d(kernel_size=2, stride=2)
    self.Conv2 = nn.Conv2d(in_channels=32, out_channels=128, kernel_size=3, padding=1)
    self.MaxPooling2 = nn.MaxPool2d(kernel_size=2, stride=2)
    self.Conv3 = nn.Conv2d(in_channels=128, out_channels=512, kernel_size=3, padding=1)
    self.Pooling3 = nn.AdaptiveMaxPool2d((2, 2))
    self.LSTM = nn.LSTM(input_size=2048, hidden_size=hidden_size, num_layers=num_lstm, batch_first=True)
    self.FC = nn.Linear(hidden_size, num_classes)
    self.SoftMax = nn.Softmax(dim=1)
    self.Conv = nn.Sequential(
        self.Conv1, self.Relu1, self.MaxPooling1, self.Conv2,
        self.Relu1, self.MaxPooling2, self.Conv3, self.Relu1, self.Pooling3
    )

1393429978@qq.com <1393429978@qq.com> *
def forward(self, x):
    x1, x2, x3, x4, x5 = x.shape
    x = torch.reshape(x, (-1, *x.shape[2:]))
    x = self.Conv(x)
    x = nn.Flatten()(x)
    x = torch.reshape(x, (x1, x2, -1))
    x, (h_n, c_n) = self.LSTM(x)
    x = h_n[-1, ...]
    x = self.FC(x)
    x = self.SoftMax(x)
    return x
```

自行设计模型

```python
def __init__(self, num_classes, hidden_size, num_lstm_layers=2, backbone_name='resnet101'):
    super().__init__()
    self.backbone = timm.create_model(backbone_name, pretrained=True, features_only=True)
    self.adap = nn.AdaptiveAvgPool2d((2, 2))

    self.lstm = nn.LSTM(2048, hidden_size, num_lstm_layers, batch_first=True)

    self.fc = nn.Linear(hidden_size, num_classes)

1393429978@qq.com <1393429978@qq.com>
def forward(self, x):
    'x: batch, num_frames, channels, height, width'
    batch, num_frames, channels, height, width = x.shape

    # x: batch * num_frames, channels, height, width
    x = torch.reshape(x, (-1, *x.shape[2:]))

    x1, x2, x3, x4, x5 = self.backbone(x)

    # x: batch * num_frames, 512, 2, 2
    x = self.adap(x3)

    # x: batch * num_frames, 2048
    x = nn.Flatten()(x)
```

参考代码给出的模型，观察到 backbone 函数返回的张量有多种，之后会尝试不同的张量形状

二．实验结果

i. 自定义 CNN-LSTM

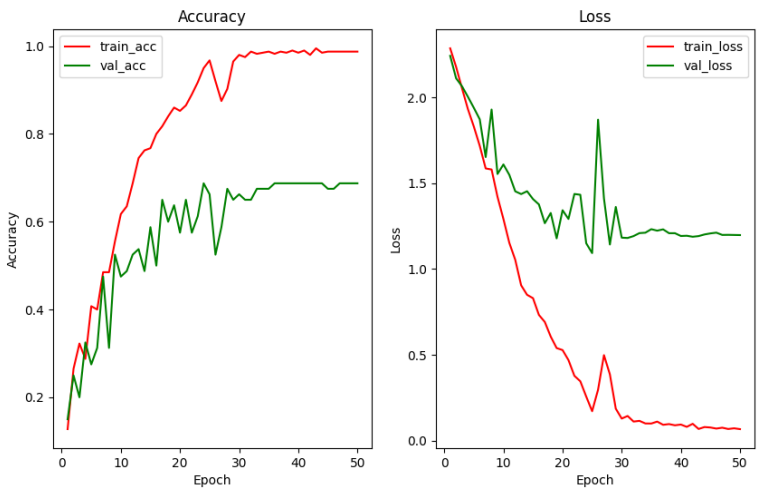
Size: 128 * 128 frame: 40 epoch: 50 batch_size: 4 lr_scheduler: ReduceLROnPlateau lr: 1e-4

模型为:

```python
class MyModel(torch.nn.Module):
    1393429978@qq.com <1393429978@qq.com> *
    def __init__(self, num_classes, hidden_size, num_lstm=2):
        super().__init__()
        self.Conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1)
        self.Relu1 = nn.ReLU(inplace=True)
        self.Convadd = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1)
        self.MaxPooling1 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.Conv2 = nn.Conv2d(in_channels=32, out_channels=128, kernel_size=3, padding=1)
        self.MaxPooling2 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.Conv3 = nn.Conv2d(in_channels=128, out_channels=512, kernel_size=3, padding=1)
        self.Pooling3 = nn.AdaptiveMaxPool2d((2, 2))
        self.LSTM = nn.LSTM(input_size=2048, hidden_size=hidden_size, num_layers=num_lstm, batch_first=True)
        self.FC = nn.Linear(hidden_size, num_classes)
        self.Conv = nn.Sequential(
            self.Conv1, self.Relu1, self.MaxPooling1, self.Conv2,
            self.Relu1, self.MaxPooling2, self.Conv3, self.Relu1, self.Pooling3
        )
        self.Conv_1 = nn.Sequential(
            self.Conv1, self.Relu1, self.Convadd, self.Relu1, self.MaxPooling1, self.Conv2,
            self.Relu1, self.MaxPooling2, self.Conv3, self.Relu1, self.Pooling3
        )
        self.Conv_2 = nn.Sequential(
            self.Conv1, self.Relu1, self.MaxPooling1, self.Conv3,
            self.Relu1, self.Pooling3
        )

    1393429978@qq.com <1393429978@qq.com> *
    def forward(self, x):
        x1, x2, x3, x4, x5 = x.shape
        x = torch.reshape(x, (-1, *x.shape[2:]))
        x = self.Conv(x)
        # x = self.Conv_1(x)
        # x = self.Conv_2(x)
        x = nn.Flatten()(x)
        x = torch.reshape(x, (x1, x2, -1))
        x, (h_n, c_n) = self.LSTM(x)
        x = h_n[-1, ...]
        x = self.FC(x)
        return x
```

■ Self.Conv self.Conv_1 self.Conv2 对应不同的 CNN 网络 本次采取 self.Conv

训练过程及结果:



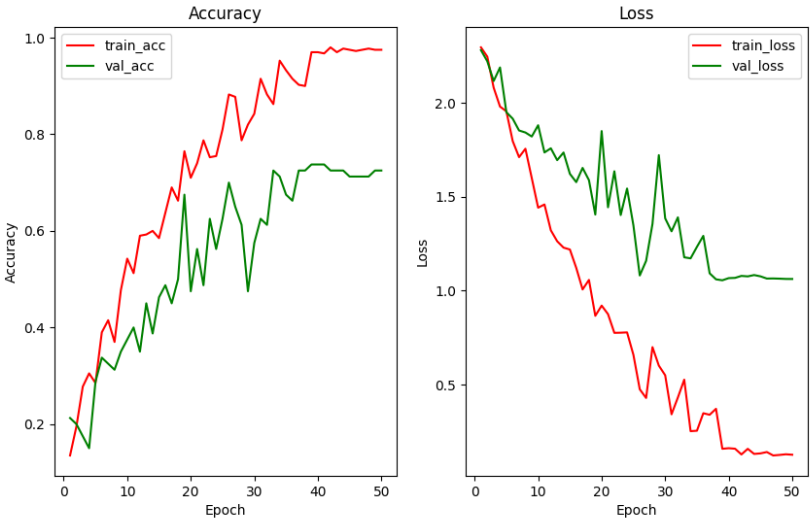
```
Epoch 00049: reducing learning rate of group 0 to 1.0000e-06.
epoch: 49, train_accuracy:  0.99, loss:  0.092, val_accuracy:  0.72, val_loss: 1.025
Saved successfully best weights to: best_weights.pt
epoch: 50/50: 100%|████████████████████████████████████████| 100/100 [00:12<00:00,  7.87it/s]
epoch: 50, train_accuracy:  0.99, loss:  0.081, val_accuracy:  0.72, val_loss: 1.025
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|████████████████████████████████████████| 25/25 [00:01<00:00, 13.23it/s]
Loss:  0.806, Acc:  0.760
```

测试集中 Accuracy 为 0.76 Loss 为 0.806 满足对自定义 CNN 的准确率要求

不同参数/CNN 形状下的结果：

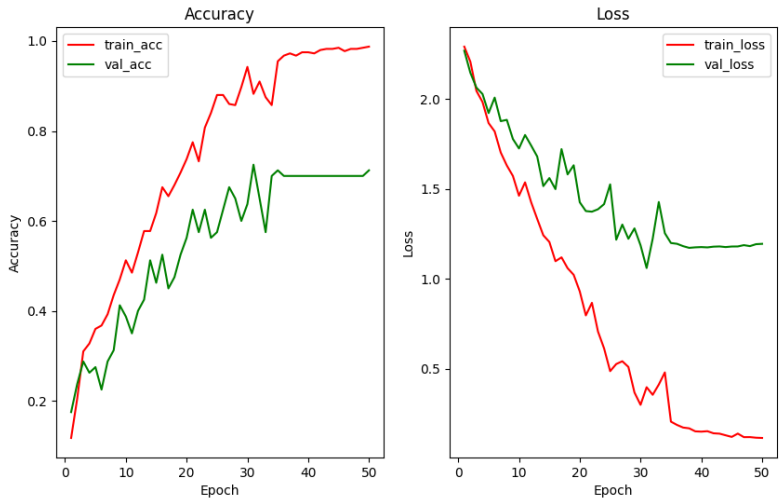
1. 调整网络结构(需要在自定义的模型下修改)

   a) 增加一层 LSTM：



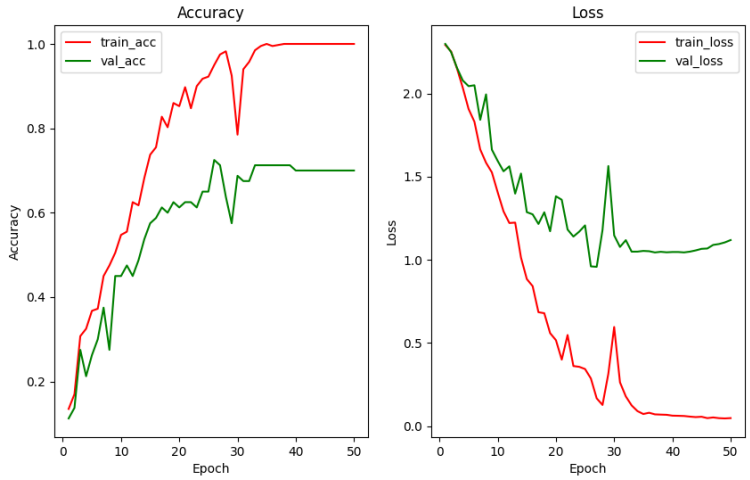

```
epoch: 48, train_accuracy:  0.98, loss:  0.126, val_accuracy:  0.71, val_loss: 1.064
epoch: 49/50: 100%|████████████████████████████████████████| 100/100 [00:12<00:00,  7.95it/s]
epoch: 49, train_accuracy:  0.97, loss:  0.129, val_accuracy:  0.72, val_loss: 1.063
epoch: 50/50: 100%|████████████████████████████████████████| 100/100 [00:12<00:00,  7.95it/s]
epoch: 50, train_accuracy:  0.97, loss:  0.127, val_accuracy:  0.72, val_loss: 1.063
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|████████████████████████████████████████| 25/25 [00:01<00:00, 13.85it/s]
Loss:  0.787, Acc:  0.750
```

   b) 增加一层 Conv2d(self.Conv_1)



```
epoch: 46, train_accuracy:  0.98, loss:  0.138, val_accuracy:  0.70, val_loss: 1.180
epoch: 47/50: 100%|████████████████████████████████████████| 100/100 [00:18<00:00,  5.28it/s]
epoch: 47, train_accuracy:  0.98, loss:  0.119, val_accuracy:  0.70, val_loss: 1.188
epoch: 48/50: 100%|████████████████████████████████████████| 100/100 [00:19<00:00,  5.15it/s]
epoch: 48, train_accuracy:  0.98, loss:  0.119, val_accuracy:  0.70, val_loss: 1.182
epoch: 49/50: 100%|████████████████████████████████████████| 100/100 [00:19<00:00,  5.25it/s]
epoch: 49, train_accuracy:  0.98, loss:  0.115, val_accuracy:  0.70, val_loss: 1.193
epoch: 50/50: 100%|████████████████████████████████████████| 100/100 [00:18<00:00,  5.29it/s]
epoch: 50, train_accuracy:  0.99, loss:  0.114, val_accuracy:  0.71, val_loss: 1.195
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|████████████████████████████████████████| 25/25 [00:02<00:00, 10.82it/s]
Loss:  0.964, Acc:  0.750
```

c) 减少一层卷积

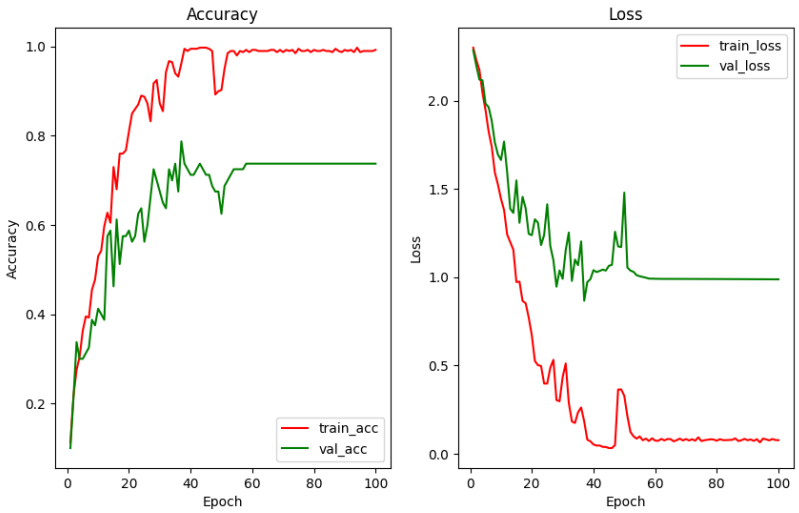



```
epoch: 48/50: 100%|████████████████████████████████████████| 100/100 [00:30<00:00,  3.25it/s]
epoch: 48, train_accuracy:  1.00, loss:  0.048, val_accuracy:  0.70, val_loss: 1.096
epoch: 49/50: 100%|████████████████████████████████████████| 100/100 [00:30<00:00,  3.30it/s]
epoch: 49, train_accuracy:  1.00, loss:  0.047, val_accuracy:  0.70, val_loss: 1.105
epoch: 50/50: 100%|████████████████████████████████████████| 100/100 [00:30<00:00,  3.30it/s]
epoch: 50, train_accuracy:  1.00, loss:  0.049, val_accuracy:  0.70, val_loss: 1.119
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|████████████████████████████████████████| 25/25 [00:03<00:00,  6.65it/s]
Loss:  0.874, Acc:  0.780
```

+ Code + Markdown

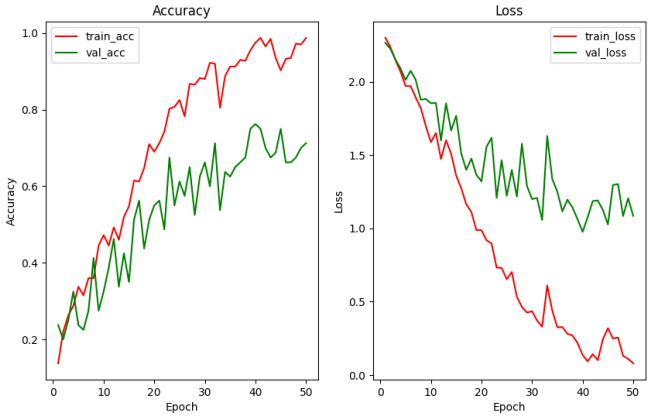
2. 调整训练次数(本实验将训练次数调整至 100 次)

a) 实验结果:



```
epoch: 96/100: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.31it/s]
epoch: 96, train_accuracy:  0.99, loss:  0.082, val_accuracy:  0.74, val_loss: 0.989
epoch: 97/100: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.61it/s]
epoch: 97, train_accuracy:  0.99, loss:  0.076, val_accuracy:  0.74, val_loss: 0.988
epoch: 98/100: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.63it/s]
epoch: 98, train_accuracy:  0.99, loss:  0.084, val_accuracy:  0.74, val_loss: 0.988
epoch: 99/100: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.59it/s]
epoch: 99, train_accuracy:  0.99, loss:  0.079, val_accuracy:  0.74, val_loss: 0.988
epoch: 100/100: 100%|███████████████████████████████████████| 100/100 [00:13<00:00,  7.36it/s]
epoch: 100, train_accuracy:  0.99, loss:  0.077, val_accuracy:  0.74, val_loss: 0.988
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|████████████████████████████████████████████| 25/25 [00:01<00:00, 14.06it/s]
Loss:  1.108, Acc:  0.680
```

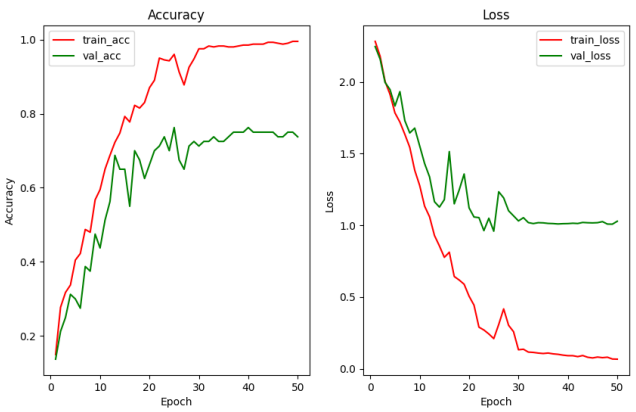
3. 调整 lr 优化方法(采取余弦退火的调整策略，但是参数好像没配置太好)
   a) 实验结果：




```
epoch: 43, train_accuracy:  0.98, loss:  0.101, val_accuracy:  0.68, val_loss: 1.191
epoch: 44/50: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.35it/s]
Epoch 0.24: adjusting learning rate of group 0 to 9.9964e-05.
epoch: 44, train_accuracy:  0.94, loss:  0.243, val_accuracy:  0.69, val_loss: 1.127
epoch: 45/50: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.65it/s]
Epoch 0.32: adjusting learning rate of group 0 to 9.9938e-05.
epoch: 45, train_accuracy:  0.90, loss:  0.320, val_accuracy:  0.75, val_loss: 1.027
epoch: 46/50: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.60it/s]
Epoch 0.25: adjusting learning rate of group 0 to 9.9962e-05.
epoch: 46, train_accuracy:  0.93, loss:  0.248, val_accuracy:  0.66, val_loss: 1.296
epoch: 47/50: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.58it/s]
Epoch 0.26: adjusting learning rate of group 0 to 9.9960e-05.
epoch: 47, train_accuracy:  0.94, loss:  0.256, val_accuracy:  0.66, val_loss: 1.303
epoch: 48/50: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.36it/s]
Epoch 0.13: adjusting learning rate of group 0 to 9.9990e-05.
epoch: 48, train_accuracy:  0.97, loss:  0.131, val_accuracy:  0.68, val_loss: 1.085
epoch: 49/50: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.58it/s]
Epoch 0.11: adjusting learning rate of group 0 to 9.9993e-05.
epoch: 49, train_accuracy:  0.97, loss:  0.110, val_accuracy:  0.70, val_loss: 1.206
epoch: 50/50: 100%|████████████████████████████████████████| 100/100 [00:13<00:00,  7.58it/s]
Epoch 0.08: adjusting learning rate of group 0 to 9.9996e-05.
epoch: 50, train_accuracy:  0.99, loss:  0.080, val_accuracy:  0.71, val_loss: 1.086
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|████████████████████████████████████████████| 25/25 [00:01<00:00, 13.21it/s]
Loss:  0.679, Acc:  0.810
```

   b) 观察到图像似乎并没有完全收敛，再增大 epoch 补做一组

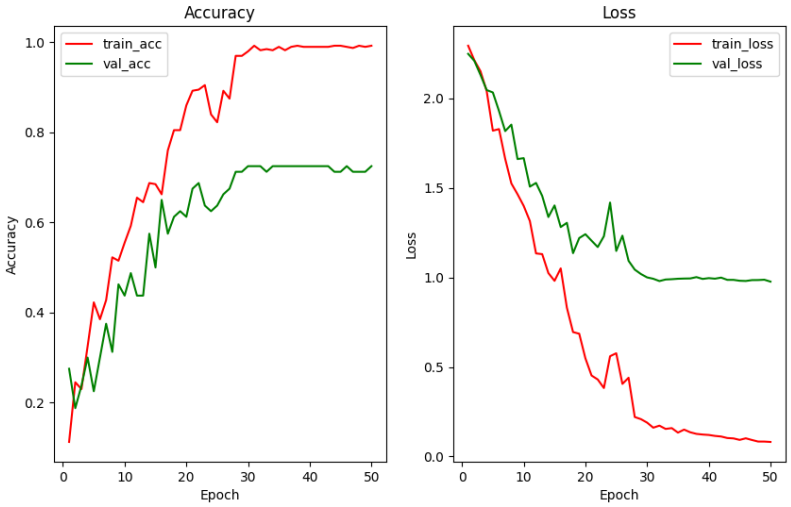
4. 调整视频 frame ， size
   a) 调整至 frame 30，size(64, 64)



```
epoch: 46, train_accuracy:  0.99, loss:  0.082, val_accuracy:  0.74, val_loss: 1.019
epoch: 47/50: 100%|| 100/100 [00:02<00:00, 33.59it/s]
epoch: 47, train_accuracy:  0.99, loss:  0.078, val_accuracy:  0.74, val_loss: 1.026
epoch: 48/50: 100%|| 100/100 [00:03<00:00, 30.91it/s]
epoch: 48, train_accuracy:  0.99, loss:  0.081, val_accuracy:  0.75, val_loss: 1.009
epoch: 49/50: 100%|| 100/100 [00:02<00:00, 33.60it/s]
epoch: 49, train_accuracy:  0.99, loss:  0.069, val_accuracy:  0.75, val_loss: 1.008
epoch: 50/50: 100%|| 100/100 [00:02<00:00, 33.46it/s]
epoch: 50, train_accuracy:  0.99, loss:  0.067, val_accuracy:  0.74, val_loss: 1.029
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|| 25/25 [00:00<00:00, 43.08it/s]
Loss:  1.047, Acc:  0.730
```

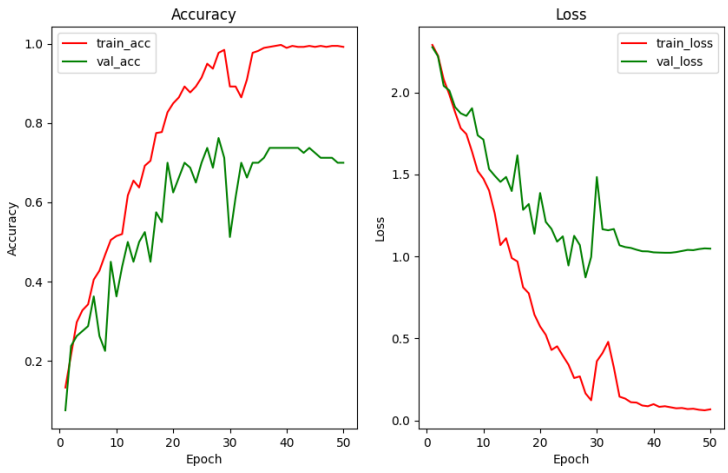
5. 调整学习率：

a) 减小学习率至 5e-5




```
epoch: 49/50: 100%|| 100/100 [00:12<00:00,  7.90it/s]
epoch: 49, train_accuracy:  0.99, loss:  0.083, val_accuracy:  0.71, val_loss: 0.987
acc_weight
epoch: 50/50: 100%|| 100/100 [00:12<00:00,  7.86it/s]
epoch: 50, train_accuracy:  0.99, loss:  0.081, val_accuracy:  0.72, val_loss: 0.977
Saved successfully best weights to: best_weights.pt
acc_weight
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|| 25/25 [00:01<00:00, 13.52it/s]
Loss:  0.888, Acc:  0.790
```

b) 增大学习率至 2e-4



```
epoch: 48, train_accuracy:  0.99, loss:  0.065, val_accuracy:  0.71, val_loss: 1.044
acc_weight
epoch: 49/50: 100%|██████████| 100/100 [00:12<00:00,  7.87it/s]
epoch: 49, train_accuracy:  0.99, loss:  0.062, val_accuracy:  0.70, val_loss: 1.049
acc_weight
epoch: 50/50: 100%|██████████| 100/100 [00:12<00:00,  7.85it/s]
epoch: 50, train_accuracy:  0.99, loss:  0.068, val_accuracy:  0.70, val_loss: 1.047
acc_weight
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
================================================
Evaluate: 100%|██████████| 25/25 [00:01<00:00, 13.41it/s]
Loss:  0.670, Acc:  0.810
```

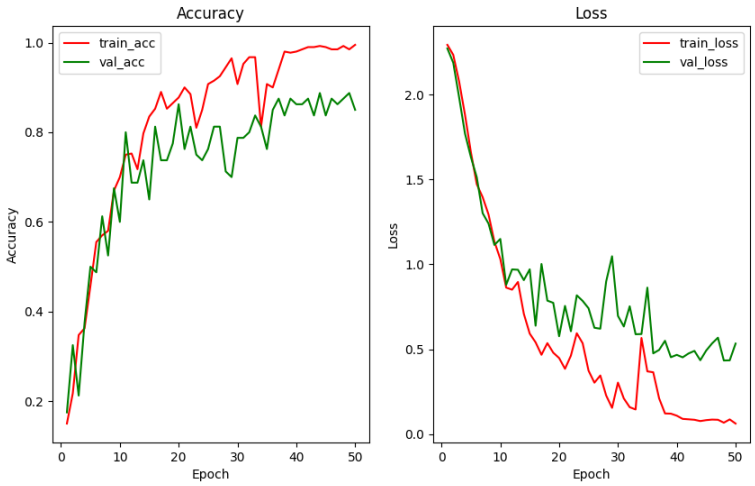
ii. 在预训练模型上增量训练

参数：resnet101，输出通道数为 512，2 层 lstm，size(128, 128), frame: 64, epoch: 50

模型：首先采用实例代码给出的模型

```python
class ClassificationModel(nn.Module):
    1393429978@qq.com <1393429978@qq.com>
    def __init__(self, num_classes, hidden_size, num_lstm_layers=2, backbone_name='resnet101'):
        super().__init__()
        self.backbone = timm.create_model(backbone_name, pretrained=True, features_only=True)
        self.adap = nn.AdaptiveAvgPool2d((2, 2))

        self.lstm = nn.LSTM(2048, hidden_size, num_lstm_layers, batch_first=True)

        self.fc = nn.Linear(hidden_size, num_classes)

    1393429978@qq.com <1393429978@qq.com> *
    def forward(self, x):
        'x: batch, num_frames, channels, height, width'
        batch, num_frames, channels, height, width = x.shape

        # x: batch * num_frames, channels, height, width
        x = torch.reshape(x, (-1, *x.shape[2:]))

        x1, x2, x3, x4, x5 = self.backbone(x)
        # print(x1.shape)
        # print(x2.shape)
        # print(x3.shape)
        # print(x4.shape)
        # print(x5.shape)

        x = self.adap(x3)

        x = nn.Flatten()(x)

        x = torch.reshape(x, (batch, num_frames, -1))

        x, (h_n, c_n) = self.lstm(x)

        x = h_n[-1, ...]

        x = self.fc(x)

        return x
```

训练过程及结果



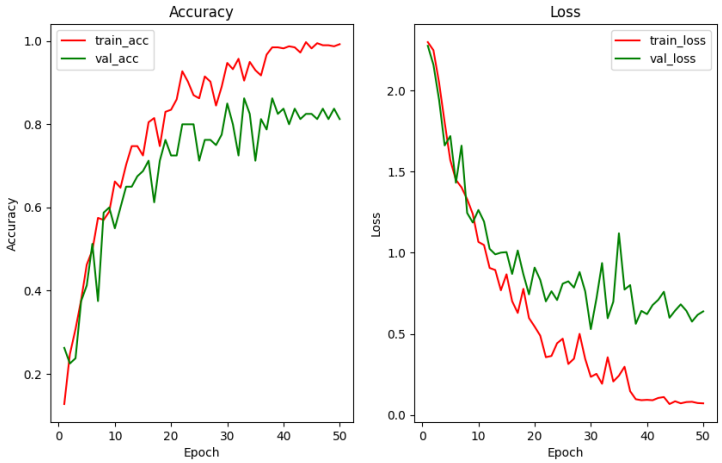

```
epoch: 46/50: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.67it/s]
epoch: 46, train_accuracy:  0.98, loss:  0.086, val_accuracy:  0.88, val_loss: 0.534
epoch: 47/50: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.69it/s]
epoch: 47, train_accuracy:  0.98, loss:  0.085, val_accuracy:  0.86, val_loss: 0.568
epoch: 48/50: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.61it/s]
epoch: 48, train_accuracy:  0.99, loss:  0.068, val_accuracy:  0.88, val_loss: 0.434
Saved successfully best weights to: best_weights.pt
epoch: 49/50: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.68it/s]
epoch: 49, train_accuracy:  0.98, loss:  0.087, val_accuracy:  0.89, val_loss: 0.435
epoch: 50/50: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.70it/s]
epoch: 50, train_accuracy:  0.99, loss:  0.062, val_accuracy:  0.85, val_loss: 0.533
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|████████████████████████████████████████| 25/25 [00:04<00:00,  5.66it/s]
Loss:  0.353, Acc:  0.890
```

测试集 loss 为 0.353 accuracy 为 0.89 符合实验对准确率的要求

不同参数下的结果：

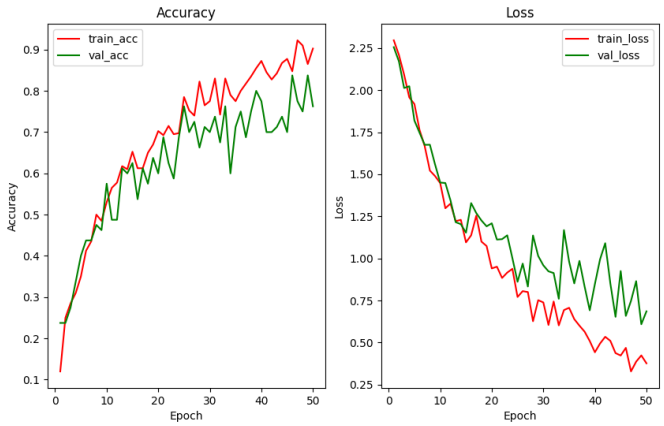
1. 调整网络结构

   a) 增加一层 lstm：




```
epoch: 47/50: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.67it/s]
epoch: 47, train_accuracy:  0.99, loss:  0.079, val_accuracy:  0.84, val_loss: 0.642
epoch: 48/50: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.61it/s]
Epoch 00048: reducing learning rate of group 0 to 1.0000e-06.
epoch: 48, train_accuracy:  0.99, loss:  0.081, val_accuracy:  0.81, val_loss: 0.575
epoch: 49/50: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.68it/s]
epoch: 49, train_accuracy:  0.99, loss:  0.073, val_accuracy:  0.84, val_loss: 0.617
epoch: 50/50: 100%|████████████████████████████████████████| 100/100 [00:27<00:00,  3.66it/s]
epoch: 50, train_accuracy:  0.99, loss:  0.071, val_accuracy:  0.81, val_loss: 0.638
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|████████████████████████████████████████| 25/25 [00:04<00:00,  5.62it/s]
Loss:  0.499, Acc:  0.850
```

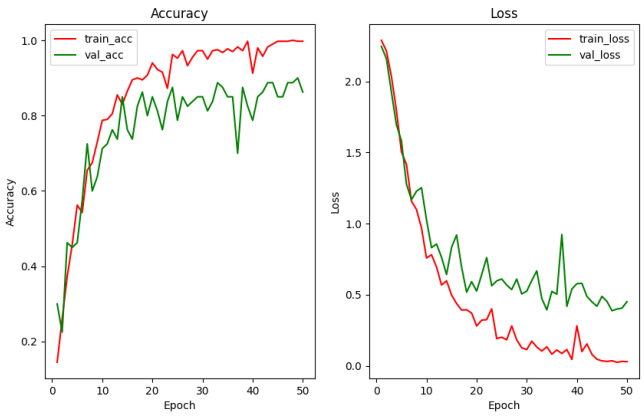
b) 使用输出为 256 通道的预训练网络:




```
epoch: 46, train_accuracy:  0.85, loss:  0.469, val_accuracy:  0.84, val_loss: 0.659
epoch: 47/50: 100%|| 100/100 [00:22<00:00,  4.52it/s]
epoch: 47, train_accuracy:  0.92, loss:  0.329, val_accuracy:  0.78, val_loss: 0.748
epoch: 48/50: 100%|| 100/100 [00:22<00:00,  4.41it/s]
epoch: 48, train_accuracy:  0.91, loss:  0.387, val_accuracy:  0.75, val_loss: 0.865
epoch: 49/50: 100%|| 100/100 [00:22<00:00,  4.52it/s]
epoch: 49, train_accuracy:  0.86, loss:  0.424, val_accuracy:  0.84, val_loss: 0.609
Saved successfully best weights to: best_weights.pt
epoch: 50/50: 100%|| 100/100 [00:22<00:00,  4.51it/s]
epoch: 50, train_accuracy:  0.90, loss:  0.377, val_accuracy:  0.76, val_loss: 0.686
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|| 25/25 [00:04<00:00,  5.68it/s]
Loss:  0.676, Acc:  0.820
```

2. 调整视频 frame ， size

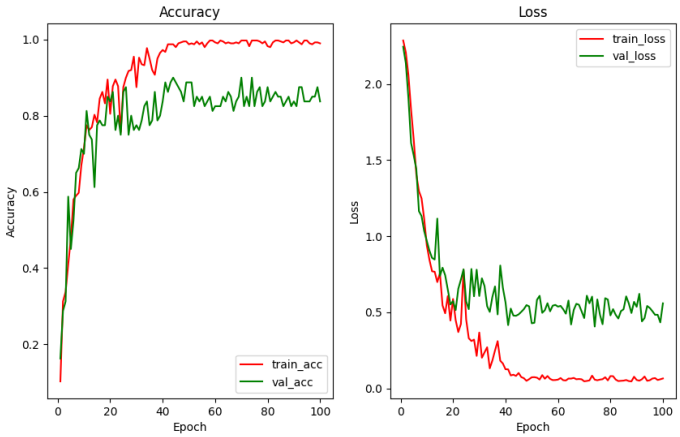
a) 调整为 frame: 30, size: (64, 64)




```
epoch: 48, train_accuracy:  1.00, loss:  0.024, val_accuracy:  0.89, val_loss: 0.399
acc_weight
epoch: 49/50: 100%|| 100/100 [00:07<00:00, 13.03it/s]
epoch: 49, train_accuracy:  1.00, loss:  0.030, val_accuracy:  0.90, val_loss: 0.405
acc_weight
Saved successfully best acc weights to acc_weight
epoch: 50/50: 100%|| 100/100 [00:07<00:00, 13.03it/s]
epoch: 50, train_accuracy:  1.00, loss:  0.029, val_accuracy:  0.86, val_loss: 0.451
acc_weight
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
==================================================
Evaluate: 100%|| 25/25 [00:01<00:00, 15.94it/s]
Loss:  0.290, Acc:  0.890
```

3. 调整训练次数

a) 调整至 100 次




```
epoch: 95, train_accuracy:  1.00, loss:  0.054, val_accuracy:  0.84, val_loss: 0.529
epoch: 96/100: 100%|████████████████████████████████████| 100/100 [00:27<00:00,  3.60it/s]
epoch: 96, train_accuracy:  0.99, loss:  0.065, val_accuracy:  0.84, val_loss: 0.508
epoch: 97/100: 100%|████████████████████████████████████| 100/100 [00:27<00:00,  3.68it/s]
epoch: 97, train_accuracy:  0.99, loss:  0.068, val_accuracy:  0.85, val_loss: 0.484
epoch: 98/100: 100%|████████████████████████████████████| 100/100 [00:27<00:00,  3.67it/s]
epoch: 98, train_accuracy:  0.99, loss:  0.056, val_accuracy:  0.85, val_loss: 0.484
epoch: 99/100: 100%|████████████████████████████████████| 100/100 [00:27<00:00,  3.66it/s]
epoch: 99, train_accuracy:  0.99, loss:  0.061, val_accuracy:  0.88, val_loss: 0.434
epoch: 100/100: 100%|███████████████████████████████████| 100/100 [00:27<00:00,  3.61it/s]
epoch: 100, train_accuracy:  0.99, loss:  0.066, val_accuracy:  0.84, val_loss: 0.559
Saved successfully last weights to: last_weights.pt
Weights loaded successfully from path: last_weights.pt
================================================
Evaluate: 100%|████████████████████████████████████████| 25/25 [00:04<00:00,  5.64it/s]
Loss:  0.553, Acc:  0.870
```

iii. 单独取出一些视频进行分类：(选择的是第一类 0，第二类的一个视频 1)

```python
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description="select model")
    parser.add_argument('--model_select', type=str, default="my_model",
                        help='model select from my_model, pre_trained')
    parser.add_argument('--pt_select', type=str, default="last",
                        help='pt last ? best ? acc')
    parser.add_argument('--video_path', type=str, default=None,
                        help="select path")

    args = parser.parse_args()
    assert not (args.video_path is None)
    data = lab3_data_scratch.read_radio(args.video_path, transform, num_frames)
    data = data.unsqueeze(0)
    data = data.to('cuda')
    model = my_model.MyModel(num_classes=num_classes, hidden_size=128, num_lstm=2) if args.model_select == "my_model" \
        else pretrained_model.ClassificationModel(num_classes=num_classes, hidden_size=128, num_lstm_layers=2)
    model.to('cuda')
    model.load_state_dict(torch.load('last_weights.pt'))
    model.eval()
    output = model(data)
    output = output.reshape(-1)
    _, idx = torch.max(output, dim=0)
    print(f'predict class is {idx.item()}')
    print(output)
```

自行编写的程序，用于测试单一视频文件(上面已经列出)

a) 原始自定义 CNN

```
!python test_one_file.py --model_select my_model --video_path ./data/Biking/v_Biking_g01_c01.avi

/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for thi
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
0
tensor([ 4.0730, -1.4649, -0.7287, -2.3461,  1.1372,  3.0510,  1.8551, -0.6510,
        -0.9874, -2.4777], device='cuda:0', grad_fn=<ReshapeAliasBackward0>)
```

分类结果

b) 调整一些参数的自定义 CNN(不是全部)

1. 减少一层 CNN

```
!python test_one_file.py --model_select my_model --video_path ./data/CliffDiving/v_CliffDiving_g01_c03.avi
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required fo
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
predict class is 1
tensor([-1.2916,  5.1620, -1.5068, -0.1738,  1.1701,  0.1434, -2.6146, -0.3765,
         1.3668, -0.9906], device='cuda:0', grad_fn=<ReshapeAliasBackward0>)
```

2. 增加一层 CNN

```
!python test_one_file.py --model_select my_model --video_path ./data/Biking/v_Biking_g01_c01.avi
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
predict class is 0
tensor([ 2.4723, -0.2442, -1.6847, -3.5207,  0.9730,  2.0940,  0.1658, -0.4836,
        -0.9610, -1.2447], device='cuda:0', grad_fn=<ReshapeAliasBackward0>)
```

c) 基于预训练模型的

1. 示例模型:

```
!python test_one_file.py --model_select pre_trained --video_path ./data/Biking/v_Biking_g01_c01.avi
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for thi
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
predict class is 0
tensor([ 4.5342, -0.8496, -1.1263, -3.3393, -0.2681,  1.7190, -0.4253, -0.7374,
         0.0206, -1.2710], device='cuda:0', grad_fn=<ReshapeAliasBackward0>)
```

```
!python test_one_file.py --model_select pre_trained --video_path ./data/CliffDiving/v_CliffDiving_g01_c03.avi
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for thi
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
predict class is 1
tensor([-1.4716,  5.1343, -1.8935, -0.2301,  1.1286, -2.4443, -1.3417,  0.4696,
         0.6211, -0.8791], device='cuda:0', grad_fn=<ReshapeAliasBackward0>)
```

2. 增加一层 lstm

```
!python test_one_file.py --model_select pre_trained --video_path ./data/Biking/v_Biking_g01_c01.avi
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
predict class is 0
tensor([ 5.4644,  0.1969, -0.6469, -2.7397,  0.3736,  1.6768,  0.7520, -2.0272,
        -0.5488, -3.1418], device='cuda:0', grad_fn=<ReshapeAliasBackward0>)
```

```
!python test_one_file.py --model_select pre_trained --video_path ./data/CliffDiving/v_CliffDiving_g01_c03.avi
```

出现错误预测

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
predict class is 3
tensor([-2.2831,  2.1788,  0.3038,  3.5242,  0.8067, -3.4744, -1.7631, -0.3582,
        -1.0687, -1.4047], device='cuda:0', grad_fn=<ReshapeAliasBackward0>)
```

3. 增加至 100 训练次数

```
!python test_one_file.py --model_select pre_trained --video_path ./data/Biking/v_Biking_g01_c01.avi
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for thi
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
predict class is 0
tensor([ 5.2577, -0.1465, -1.7167, -2.3386, -0.0721,  0.9942, -0.5253, -2.8383,
         0.2338, -0.8693], device='cuda:0', grad_fn=<ReshapeAliasBackward0>)
```

```
!python test_one_file.py --model_select pre_trained --video_path ./data/CliffDiving/v_CliffDiving_g12_c02.avi
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for thi
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
predict class is 1
tensor([-0.8885,  3.8674, -2.8125,  0.6432,  2.4527, -1.5041, -1.1924, -2.7691,
         0.6609, -1.7430], device='cuda:0', grad_fn=<ReshapeAliasBackward0>)
```

三．一些简单的总结

i. 自定义网络在验证集上的 loss 和准确率一直与训练集相差较大

在我自己的网络中，可能出现了过拟合的情况，导致验证集上虽然收敛，但是收敛点并不符合预期，可能是我在自定义的模型中学习率直接套用了在预训练模型上的学习率，导致陷入局部最优的情况，无法找到最优解(在最后对学习率略作调整的部分，发现效果比原始的好一些)

ii. 使用余弦退火调整策略在自定义网络的效果好一些

在对实验过程输出的观察中，我发现原始的 ReduceLROnPlateau 并没有及时调整学习率，因为很多时候验证集的 loss 都在下降，但是下降的幅度很低，这无法触发学习率调整，因此采用余弦退火的方式，按照余弦周期进行学习率调整，学习率在每个 epoch 都会稍微发生变化

iii. 增加网络层数，或者使用更大的 frame size 的效果并不一定更准确

本实验的数据集数量可能并不是很多，增大网络复杂性，或者视频的帧数尺寸，容易导致过拟合的情况，时空信息的表示也不是越准确越好，有时忽略一些不关键的视频信息对于提高模型的泛化性也有所帮助

总之，本次实验涉及到的参数过多，我没有办法在某个参数上做很多的测试，并且每次训练的结果都有一点不确定性，上面的数据也是多次测试后取出的比较中肯的结果，虽然本次实验不能得到一般性的结论，但还是完成了要求的任务，并且尝试了对不同参数的调整和对不同模型的构造与选择，也增进了的对机器学习实验中"微调"的理解。