Mutual information regularized identity-aware facial expression recognition in compressed video

Xiaofeng Liu , Linghao Jin , Xu Han , Jane You

摘要

如何提取有效的提取面部表情识别，并且能对人脸身份变化具有鲁棒性是人脸表情识别任务一个长期的难题，特别是在视频人脸表情识别中，之前的方法都是讲视频分割成图片然后抽帧进行识别，文章中认为与表情相关的运动已经可以在压缩视频中体现。其目标是去探索在压缩视频领域中的面部表情识别，并且对身份变化能够具有一定的鲁棒性。在最多两个数量级的压缩视频领域中，我只需要利用压缩中的互补模式，也就是其残差帧，并且可以通过预先训练好的面部识别网络从 I 帧提取身份因子出来，通过互信息最小化，将表情中的身份信息解耦出来，从而提高表情识别的性能，在测试阶段，只需要压缩残差帧就可以实现表情预测，在典型的 FER 任务基准上实现了很好的性能。

关键词：面部表情识别；互信息，解耦表征，压缩视频

1 引言

近年来，计算视频模态在识别中也应用得越来越广泛，计算机视觉、情感计算和人机交互领域的研究人员一直在尝试自动识别和解释面部表情 (Facial Expression Recognition,FER)[1]、[2]，许多的研究都利用视频中的时许信息来探索人脸表情识别的属性。而伴随着深度学习的不断发展[3]，基于 RGB 的人脸表情识别在各个方面都取得了不错的进展，但针对视频的表情识别仍然具有一定的困难。

虽然在视频领域之中，有很多帧之间的关系可以使用，这也为表情识别带来了很不错的变化特性，利用这种面部的变化特性，可以识别出不同类别的表情特定的变化，然后同样由于视频帧与帧之间存在很大的重复性，所以也为识别任务引来的很多噪声[4]。经典的一些探索时序信息的方法，是通过深度学习的方法提取连续帧之间的面部表情特征，最后进行融合识别。最常见的网络骨干例如递归神经网络 RNN，3D 卷积网络等[5]，然而这些方法应用在每一帧上的计算量很大，并且对于 RNN 来说，建模长时间的时空特征依赖是很困难的，因为它容易照成梯度消失，特别是对于许多仅仅使用静态图像进行识别，然后利用决策级融合的方法来说就已经是完全失去了时间的特征依赖性，所以在视频领域中的 FER 任务中，如果只使用传统的方法是很难利用到时间线索的。

文章中则认为压缩视频领域可以适合于 FER 任务，原因主要有四个方面：第一，视频模态之中的连续帧有很多信息不全和重复的模式，这可能会淹没真实的信号[6]。使用标准的视频压缩算法，压缩比通常可以达到数百倍。因此直接对压缩域进行操作可以明显的降低成本和内存。第二，一些经典压缩视频的方法是将视频第一个图片编码为了 I 帧，然后跟随着若干个 P 帧，然后这些 P 帧被编码为了“变化”或者“运动”[7]，而在表情识别的基础就是面部肌肉的运动，所以也有许多基于动作框架进行识别的方法。所以在 P 帧便包含了与表情相关的因素，而且他们比原始的 RGB 图像更加的简单。第三，我们在压缩域探索表情识别也是有效的，因为他会帮我们剔除一些冗余的信息，呈现出最真实的

信息。第四，由于输入的数据是以压缩视频的格式进行输入的，也因此在实际的任务当中它是不需要进行解码的。

此外，由于表情的识别当中会存在许多主体之间的身份差异，这会导致学习到的特征不仅仅是表情特征，也包含了很多跟主体身份相关的特征，这会导致纯粹的 FER 任务性能下降。度量学习是一种从表情识别中提取身份的标准方法[8]，受到对抗性解纠缠的启发,[8]提出了消除了主体身份使用一个 GAN 来完成. 在一些基于图像的面部表情识别当中,**ali2019al**扩展了度量学习的视频数据，没有考虑到视频的特点，但是这些方法需要人工去标注他的身份标签，这极大增加了 FER 任务的成本。

因此，在文章中，使用一个已经训练好的网络 Facenet 来提取主题之间的身份信息，他的是预训练于数百万个身份上的数据集，对身份具有很高的准确率，同时面对姿态、遮挡、光照等变化都具有很强的鲁棒性，使用提取的身份信息作为一个锚点，我们可以去加强身份信息与表情信息之间的边缘独立性，将身份信息从表情信息中解耦出来，提高表情信息的纯度，解耦所使用的方法是采用互信息（Mutual Information,MI) 最小化的方法[9]，而不需要进行复杂的对抗解缠训练. 在高维空间中，两个随机变化的互信息难以预测，近年来有一些新的方法是将互信息转换成了可微的形式[10]，所以文章遵循同样的工作，将互信息转换成了可利用神经网络求最小值的形式来加强身份与表情的边缘独立性。因此通过在压缩视频领域上进行视频表情识别的探索，仅仅利用一个残差帧（Residual Frames,R) 从而降低内存和计算成本，同时利用互信息来提高面部表情识别的准确性，在视频表情识别的数据集都验证了该方法的有效性, 并且推断速度更快，其良好的性能证明了它的通用性和可扩展性。

2 相关工作

2.1 视频表情识别

基于视频表情识别已经得到了广泛的研究，因为面部表情是人类交流的一种自然和普遍的手段[4]，实际上，面部表情是一个运动的变化，他应该考虑到面部上每一时间的变化。在传统上，是使用人工自主定制的时间特征来完成 FER 任务，使用单独的 RGB 图像的方法正在与深度学习同样的发展，典型的方法就在决策级[11]和特征级[12]进行帧聚合。另一个方面，时间的相关性还没有被确认，时空的 FER 网络通过获取空间或者纹理信息来提取序列帧的时间信息，级联网络建议将 cnn 学习的感知视觉表示与 rnn 集成到可变长度视频[13]. 还可以利用 3D 卷积神经网络来获取时空特征，它具备有时间轴上共享参数的特点，因此也被广泛应用于基于视频的 FER。

然而，近些年来研究员都把大部分的精力致力于基于图像上，对于 FER 时空网络似乎并没有优于帧聚合的方法，所以文章第一次将 FER 时空网络扩展到了压缩视频领域，它于压缩视频领域的各个优点例如内存小，冗余信息减少等相结合。传统上的压缩视频格式例如 MPEG-4 是压缩视频的高级算法之一, 通过会将一个视频编码器将它编码为若干份，每一份都是由一个 I 帧和多个 P 帧组成的。在形式上来说的化，I 帧其实就是一个完整的 RGB 图像，而 P 帧是包含了一个运动矢量和前一帧的残差帧，运动矢量可以理解为一个光流。最新的动作识别的方法[14]提出了一种不需要解码 RGB 图像的方法，只需要对 I 帧，残差帧和运动矢量分别进行识别，最终将他们融合的方法。虽然 FER 任务跟动作识别有一些相似之处，但是对于对于 I 帧的运动范围和使用有很大的不同，FER 中的 I 帧通常是一个中性面部表情，这是不同于视频标签的，此外运动矢量也无法很好的编码表情识别，因为表情的变化

是非常细微的，低分辨率的运动矢量可能无法捕获。

2.2 互信息

基于互信息的方法已经被广发的应用，特别是在无监督学习的领域中，使用神经网络估计互信息的最大化是 ICA 算法的基础原理[15]，但是这些算法对于深度学习来说都是很难实用的，因为他们都是非线性的。最近一些研究则利用互信息来实现一个无监督的学习任务,[16]提出了一种生成对抗网络，以最小化 ICA 算法中的正负样本的互信息，此外，他还介绍了一种从联合分布和边缘分布中抽取样本的策略，并提出利用编码器和判别器来减少 JS 发散.GAN 框架除了应用于 MI 的最大化或者最小化，还可以应用于互信息的估计器中。受这些工作的启发，文章的目标是利用 MI 作为独立的合理衡量标准，并将其直接最小化作为我们的解纠缠目标。

2.3 身份消除

身份的消除是提高面部表情识别的一种手段，传统的身份消除方法是使用一个三元组进行采样，并且还要对身份进行标注才可构建一个三元组，而身份标签在 FER 任务中是无法提供的，除非你额外的进行标注，所以相比于需要标签的方法，文章是不需要以依赖身份标签的，而是在人脸识别数据集上进行预训练来获得身份信息。

FER 的典型解决方法就是度量学习，而文章中的互信息也可以于三重损失[17]相结合，它使两个恒等式之间的欧氏距离或余弦距离最大化。伴随着 GAN 的发展，对抗训练也可以用于解纠缠，与这个不同的是，文章是利用互信息来捕获身份与表情表示之间的关系，此外这些方法都是需要标签的，大大提高了训练的成本，在文章框架中实现身份消除。我们注意到对抗博弈以其不稳定的模型崩溃而臭名昭著，而我们的解决方案遵循协作的方式。

3 本文方法

3.1 本文方法概述

文章提出了一种有效的基于视频的 FER 框架，该框架是直接应用于压缩视频领域的。总体框架如图 1所示：由三个核心模块组成。预训练分支和 FER 分支分别提取了 I 帧的身份信息和解编码的 P 帧中的表情信息，然后利用可微的互信息正则化模块来测量身份和表情的相关性，然后用一个度量技巧来加速其收敛。

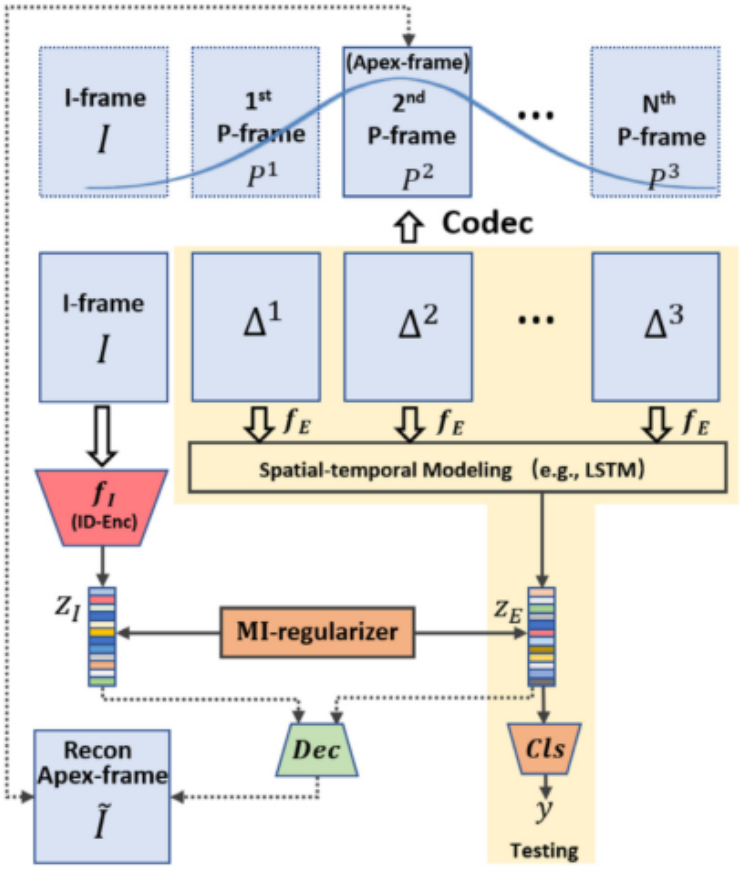


图 1: 论文总体框架

3.2 压缩建模模块

为了说明输入视频的格式，选择 MPEG-4 作为示例。压缩域有两个典型的帧，即 I 帧和 P 帧。P^t 可以用存储的偏移量重构，它是有残差 Δ^t 和运动矢量 T^t 构成的. 运动矢量 T^t 的分辨率要比原始的图片要低得多，然而面部表情识别是一个面部的微细变化，因此对于低分辨率的运动矢量来说，是对 FER 任务没有帮助的，对于 P 帧而言，$P_i^t = P_{i-\mathcal{T}_i^t}^{t-1} + \Delta_i^t$, 其中下标是所有像素的索引并且 $P^0 = I$，T^t 和 Δ^t 是采用离散余弦变换和熵编码处理。

考虑到残余图像的结构比解码后的图像简单得多，可以使用更简单、更快的 cnn 提取每一帧的特征 $f_E : \mathbb{R}^{h\times w\times 3} \rightarrow \mathbb{R}^{512}$ 实际上，我们遵循典型的 CNN-LST 吗 FER 结构的 CNN，但使用更少的层来探索其中的信息。此外，现有的大多数动作识别方法都是使用压缩视频独立地将配对的 T^t 和 Δ^t 在每个时间步长进行串联，并预测每个 p 帧的动作得分。时间线索及其发展模式对于 FER 任务非常重要。简单地选择 LSTM 来建模剩余帧的顺序发展，并将信息总结为一个表达式特征 z_E。由于的 LSTM 应用于 512 维特征，所以计算负担大大小于对原始图像的工作。对于原始图像格式的 I 帧，我们简单地使用预先训练了数百万个身份的 FaceNet 作为身份特征提取器 $f_I : \mathbb{R}^{h\times w\times 3} \rightarrow Z_I$，其中 $z_I \in \mathbb{R}^{1024}$ 表示身份特征。我们注意到 z_E 和 z_I 对于 MI 正则化不需要具有相同的维数, 一些数据集的 FER 视频以一个中性表达式开始，这可能有助于身份识别。

3.3 互信息正则化

为了消除 FER 表示中与身份相关的因素，文章提出利用预先训练的人脸识别器的身份特征 z_I 作为锚点，并显式检查 z_E 中的信息。实现不同因素的解纠缠需要两个主要目标，即每个因素都有其特定的信息，并且不包含其他因素的信息。例如，z_E 可以用常规的 ce 实现交叉熵损失最小化表情标签

y, z_I 来自预先训练的身份提取器，它固有的身份信息。然而，如何显式地测量这些因素之间的依赖关系，并最小化该指标以实现后一个目标可能是具有挑战性的。

实际上，互信息 (MI) 是通过观察另一个随机变量获得的关于一个随机变量的信息量的精确度量。

$$\mathrm{MI}\left(z_E; z_I\right) = \int_{\mathcal{E}\times\mathcal{I}} \log \frac{d\mathbb{P}_{z_E z_I}}{d\mathbb{P}_{z_E} \otimes \mathbb{P}_{z_l}} d\mathbb{P}_{z_E z_l}$$

其中 z_E 和 z_I 是随机变量分别服从分布 和 T。$\mathbb{P}_{z_E z_I}$ 指的是 (z_E, z_I) 的联合概率分布，$\mathbb{P}_{z_E} = \int_{\mathcal{I}} d\mathbb{P}_{z_E z_I}$ 和 $\mathbb{P}_{z_I} = \int_{\mathcal{E}} d\mathbb{P}_{z_E} z_I$ 是边缘分布，$\mathbb{P}_{z_E} \otimes \mathbb{P}_{z_I}$ 是边值的乘积.

MI 最小化显式地强制联合分布等于边缘的乘积，这导致了两个向量的统计独立性。相反，MI 最大化可以导致两个向量具有相同的信息，并且 MI 几乎等于一个变量的熵。我们提出利用互信息神经估计量（Mutual Information Neural Estimator ,MINE) 对 n 个独立同分布样本提供 MI 的无偏估计，利用梯度下降的神经网络 $T_\theta : \mathcal{E} \times \mathcal{I} \to \mathbb{R}$，MINE 是在 Kullback-Leibler 潜数的 Donsker-Varadhan 表示的基础上，利用一个下界逼近 MI。因此，神经信息测度可表示为：

$$\mathrm{MI}\left(\widehat{z_E; z_I}\right)_n = \sup_{\theta\in\Theta} \left\{ \mathbb{E}_{\mathbb{P}^n_{z_E z_l}}\left[T_\theta\right] - \log\left(\mathbb{E}_{\mathbb{P}^n_E} \otimes \hat{\mathbb{P}}^n_{z_E}\left[e^{T_\theta}\right]\right)\right\}$$

给定分布 P, $\widehat{\mathbb{P}}^n_{z_E}$ 表示与 n 个样本相关的经验分布。因为 T 的所有函数都取了最大值，所以这两个期望是有限的。MI 可估计为:

$$\begin{aligned}\mathrm{MI}\left(\widehat{z_E; z_I}\right)_n = &\iint \mathbb{P}^n_{z_E z_l}\left(z_E, z_I\right) T\left(z_E, z_I, \theta\right) \\ &- \log\left(\iint \mathbb{P}^n_{z_E}\left(z_E\right) \mathbb{P}^n_{z_I}\left(z_I\right)\left[e^{T\left(z_E, z_1, \theta\right)}\right]\right)\end{aligned}$$

此外，我们利用蒙特卡罗积分来避免计算 MI 的积分:

$$\frac{1}{n}\sum_{i=1}^{n} T\left(z_E, z_I, \theta\right) - \log\left(\frac{1}{n}\sum_{i=1}^{n} e^{T\left(z_E, \hat{z}_I, \theta\right)}\right)$$

估计 MI 被用作监督来更新 FER 分支。通过利用 MI 正则化，文章的新框架不再需要对抗判别器，这使得各个模块的平衡更容易。注意，我们需要额外的神经网络 T 来测量 MI，但它与 FER 分支协同训练，以最大化两个特征之间的离散性。本质上是最小-最小博弈而不是最小-最大博弈。因此，它更容易稳定的训练 (相对于对抗训练)。

许多 FER 数据集遵循定义良好的收集协议，该协议通常从中性脸开始，然后发展到一个表达式。具体地说，CK+ 中的视频由一个从中性表达转移到顶点面部表达的序列组成。最后一帧通常是顶点帧，表达强度最强。实际上，基于图像的 FER 方法选择最后三帧来构建训练和测试数据集。类似地，在 MMI 中，视频帧通常从中性面开始发展到视频中间的顶点，并在视频结束时返回到中性。注意到顶点帧 (即 CK+ 中的最后一帧或 MMI 中的中间帧) 可以清楚地包含身份信息和表达信息。因此，我们可以利用顶点坐标系作为重建的参考，简单地应用 L_1 损失。

$$\mathcal{L}_1 = \left\| I_{\text{Apex}} - \hat{I}_{\text{Apex}} \right\|_2^2$$

其中 $\hat{I}_{\text{Apex}} = \mathrm{Dec}\left(z_I, z_E\right)$, 互补限制对于我们的系统来说是不必要的，因为 FER 损耗在 FER 分支中是非常重要的。它要求保持充分的信息表达，不容易没有任何意义。然而，互补约束在早期确实有助于收敛。

3.4 总体目标

我们有三个待最小化目标，即交叉熵损失、互信息和 L_1 损失，协同工作来更新各个模块。表达式分类是 FER 的主要任务。我们选择典型的交叉熵损失，以保证 Z_E 包含足够的表达式信息，最终在 C 类表达式分类中具有良好的性能。我们用 y_c 和 Cls_c 分别表示标签和分类器软最大预测的第 c 类概率。由于 FER 分支可以用所有的损失进行更新，我们将平衡参数 α□[0,1] 和 β□[0,1] 分别指定为相互信息和 L_1 损失最小化目标：

$$\mathcal{L}_{CE} + \alpha MI\left(\widehat{z_E; z_I}\right)_n + \beta\mathcal{L}_1$$

对于 MI 估计量 T，我们用 MI $\left(\widehat{z_E; z_I}\right)_n$ 来更新由于我们使用神经网络 T 进行互信息估计，它是可扩展的，灵活的，并完全可以通过反向传播训练。

4 复现细节

4.1 与已有开源代码对比

本文章在网上没有开源的代码。

4.2 实验环境的搭建

我们对视频帧进行预处理对数据进行增强，以便进行公平的比较。对于三个数据集，视频只有一个 GOP，不需要分割视频。我们的框架使用 Pytorch 以及拥有 C++ 环境深度学习平台。在训练阶段，在所有数据集上，我们设置批大小为 48。所有模块都使用动量为 0.9 的 Adam 优化器，在 10 个 0 训练周期中权重衰减为 1e-5。在 CK+ 和 MMI 数据集上，学习率初始化为 1e-1，并在第 30 个 epoch 修改为 1e-2。对于 AFEW 数据集，我们将学习率初始化为 1e- 4，并在第 30 个 epoch 将其修改为 8e-6，在第 60 个 epoch 将其修改为 1e-7。

4.3 复现细节

首先，利用 ffmpeg 的视频转换功能，由于使用 ffmep 的功能需要 C++ 环境，因此我们将 C++ 编码后的代码导入到 python 环境中进行调用，该代码是将视频都转换成了压缩视频格式，这里采取的压缩视频格式是 MPEG 的格式具体的转换代码为图 2所示：

```
#!/usr/bin/env bash

if [ "$#" -ne 2 ]; then
    echo "Usage: ./reencode.sh [input dir] [output dir]"
fi

indir=$1
outdir=$2

mkdir outdir
if [[ ! -d "${outdir}" ]]; then
  echo "${outdir} doesn't exist. Creating it.";
  mkdir -p ${outdir}
fi

for c in $(ls ${indir})
do
  for inname in $(ls ${indir}/${c}/*avi)
  do
    class_path="$(dirname "$inname")"
    class_name="${class_path##*/}"

    outname="${outdir}/${class_name}/${inname##*/}"
    outname="${outname%.*}.mp4"

    mkdir -p "$(dirname "$outname")"
    ffmpeg -i ${inname} -vf setsar=1:1 -q:v 1 -g 300 -c:v mpeg4 -f rawvideo ${outname}

  done
done
```

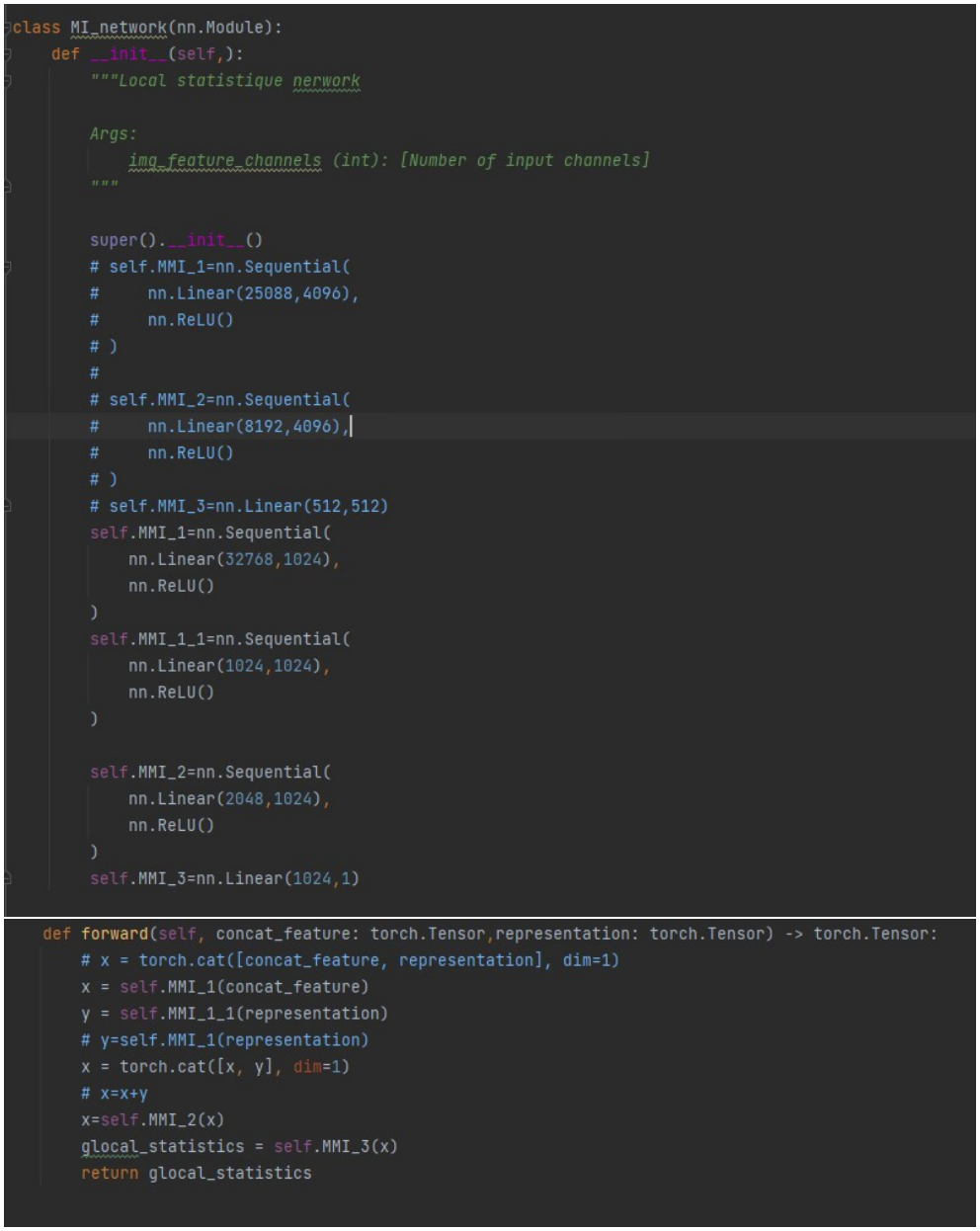
图 2: 视频转换成压缩格式

接着我们定义了一个可以获取到压缩视频残差帧类为图 3所示：

```
void create_and_load_mv_residual(
    AVFrameSideData *sd,
    PyArrayObject * bgr_arr,
    PyArrayObject * mv_arr,
    PyArrayObject * res_arr,
    int cur_pos,
    int accumulate,
    int representation,
    int *accu_src,
    int *accu_src_old,
    int width,
    int height,
    int pos_target) {

    int p_dst_x, p_dst_y, p_src_x, p_src_y, val_x, val_y;
    const AVMotionVector *mvs = (const AVMotionVector *)sd->data;

    for (int i = 0; i < sd->size / sizeof(*mvs); i++) {
        const AVMotionVector *mv = &mvs[i];
        assert(mv->source == -1);

        if (mv->dst_x - mv->src_x != 0 || mv->dst_y - mv->src_y != 0) {

            val_x = mv->dst_x - mv->src_x;
            val_y = mv->dst_y - mv->src_y;

            for (int x_start = (-1 * mv->w / 2); x_start < mv->w / 2; ++x_start) {
                for (int y_start = (-1 * mv->h / 2); y_start < mv->h / 2; ++y_start) {
                    p_dst_x = mv->dst_x + x_start;
                    p_dst_y = mv->dst_y + y_start;

                    p_src_x = mv->src_x + x_start;
                    p_src_y = mv->src_y + y_start;

                    if (p_dst_y >= 0 && p_dst_y < height &&
                        p_dst_x >= 0 && p_dst_x < width &&
                        p_src_y >= 0 && p_src_y < height &&
                        p_src_x >= 0 && p_src_x < width) {

                        // Write MV.
                        if (accumulate) {
                            for (int c = 0; c < 2; ++c) {
                                accu_src       [p_dst_x * height * 2 + p_dst_y * 2 + c]
                                 = accu_src_old[p_src_x * height * 2 + p_src_y * 2 + c];
                            }
                        } else {
                            *((int32_t*)PyArray_GETPTR3(mv_arr, p_dst_y, p_dst_x, 0)) = val_x;
                            *((int32_t*)PyArray_GETPTR3(mv_arr, p_dst_y, p_dst_x, 1)) = val_y;
                        }
                    }
                }
            }
        }
    }
    if (accumulate) {
        memcpy(accu_src_old, accu_src, width * height * 2 * sizeof(int));
    }
    if (cur_pos > 0){
        if (accumulate) {
            if (representation == MV && cur_pos == pos_target) {
                for (int x = 0; x < width; ++x) {
                    for (int y = 0; y < height; ++y) {
                        *((int32_t*)PyArray_GETPTR3(mv_arr, y, x, 0))
                         = x - accu_src[x * height * 2 + y * 2];
                        *((int32_t*)PyArray_GETPTR3(mv_arr, y, x, 1))
                         = y - accu_src[x * height * 2 + y * 2 + 1];
                    }
                }
            }
        }
        if (representation == RESIDUAL && cur_pos == pos_target) {

            uint8_t *bgr_data = (uint8_t*) bgr_arr->data;
            int32_t *res_data = (int32_t*) res_arr->data;

            int stride_0 = height * width * 3;
            int stride_1 = width * 3;
            int stride_2 = 3;

            int y;

            for (y = 0; y < height; ++y) {
                int c, x, src_x, src_y, location, location2, location_src;
                int32_t tmp;
                for (x = 0; x < width; ++x) {
                    tmp = x * height * 2 + y * 2;
                    if (accumulate) {
                        src_x = accu_src[tmp];
                        src_y = accu_src[tmp + 1];
                    } else {
                        src_x = x - (*((int32_t*)PyArray_GETPTR3(mv_arr, y, x, 0)));
                        src_y = y - (*((int32_t*)PyArray_GETPTR3(mv_arr, y, x, 1)));
                    }
                    location_src = src_y * stride_1 + src_x * stride_2;

                    location = y * stride_1 + x * stride_2;
                    for (c = 0; c < 3; ++c) {
                        location2 = stride_0 + location;
                        res_data[location] =  (int32_t) bgr_data[location2]
                                            - (int32_t) bgr_data[location_src + c];
                        location += 1;
                    }
                }
            }
        }
    }
}
```

图 3: 定义获取 R 帧的类

对于互信息的估计器，代码使用的是利用神经网络来评估身份信息和表情信息的互信息最小化，因此定义了一个 MI-network, 如图 4所示：



```python
class MI_network(nn.Module):
    def __init__(self,):
        """Local statistique nerwork

        Args:
            img_feature_channels (int): [Number of input channels]
        """

        super().__init__()
        # self.MMI_1=nn.Sequential(
        #     nn.Linear(25088,4096),
        #     nn.ReLU()
        # )
        #
        # self.MMI_2=nn.Sequential(
        #     nn.Linear(8192,4096),
        #     nn.ReLU()
        # )
        # self.MMI_3=nn.Linear(512,512)
        self.MMI_1=nn.Sequential(
            nn.Linear(32768,1024),
            nn.ReLU()
        )
        self.MMI_1_1=nn.Sequential(
            nn.Linear(1024,1024),
            nn.ReLU()
        )

        self.MMI_2=nn.Sequential(
            nn.Linear(2048,1024),
            nn.ReLU()
        )
        self.MMI_3=nn.Linear(1024,1)

    def forward(self, concat_feature: torch.Tensor,representation: torch.Tensor) -> torch.Tensor:
        # x = torch.cat([concat_feature, representation], dim=1)
        x = self.MMI_1(concat_feature)
        y = self.MMI_1_1(representation)
        # y=self.MMI_1(representation)
        x = torch.cat([x, y], dim=1)
        # x=x+y
        x=self.MMI_2(x)
        glocal_statistics = self.MMI_3(x)
        return glocal_statistics
```

图 4: 互信息评估网络

为了去获取视频中身份信息，文章使用的是在身份数据集上预训练好的 facenet 进行提取身份信息 f_I, 并且使用一个时空网络 CNN+LSTM 的格式来获取视频中的表情信息 f_E，实现过程，facenet 的网络骨架使用的是 mobilenet，时空网络则是使用一个 resnet50+LSTM 来实现的，具体代码如图 5所示：

```python
def _prepare_base_model(self, base_model):

    if 'resnet' in base_model:
        # self.base_model = getattr(torchvision.models, base_model)(pretrained=True)
        self.backbone = "mobilenet"
        self.model_path = "/data/zjl/192-torch/facenet-pytorch-main/model_data/facenet_mobilenet.pth"
        base_model='resnet50'
        self.resnet = getattr(torchvision.models, base_model)(pretrained=True)
        self.base_model=nn.Sequential(*list(self.resnet.children())[:-1])

        self.mobilenet= Facenet(backbone=self.backbone, num_classes=self.num_classes, pretrained=False)

        model_dict = self.mobilenet.state_dict()
        pretrained_dict = torch.load(self.model_path)
        load_key, no_load_key, temp_dict = [], [], {}
        for k, v in pretrained_dict.items():
            if k in model_dict.keys() and np.shape(model_dict[k]) == np.shape(v):
                temp_dict[k] = v
                load_key.append(k)
            else:
                no_load_key.append(k)
        model_dict.update(temp_dict)
        self.mobilenet.load_state_dict(model_dict)

        self.facenet=nn.Sequential(*list(self.mobilenet.children())[:-4])



        self.lstm=nn.LSTM(input_size=self.input_size,hidden_size=self.hidden_size,num_layers=self.num_layers,batch_first=True)
```

图 5: 网络骨架

整个网络的前向传播如图 6所示：

```python
def forward(self, input,den_input):
    input = input.view((-1, ) + input.size()[-3:])
    den_input = den_input.view((-1,) + den_input.size()[-3:])
    if self._representation in ['mv', 'residual']:
        input = self.data_bn(input)

    input= self.base_model(input)

    input=input.flatten(1)
    den_input = self.facenet(den_input)
    den_input = den_input.flatten(1)
    #resnet50:cat
    input=input.view(-1,self.num_segments,input.size()[1])

    input,_=self.lstm(input)

    input = input.reshape(-1, input.size()[2]*self.num_segments)

    MI_1=self._MMI(input,den_input)

    mv_input2=den_input[1:]
    mv_input3=den_input[0:1]
    mv_input4=torch.cat([mv_input2, mv_input3], dim=0)

    MI_2 = self._MMI(input, mv_input4)


    base_out=self.fc(input)

    return base_out,MI_1,MI_2
```

图 6: 整个网络的前向传播

5 实验结果分析

5.1 数据集描述

CK+ 数据集[18]指的是 Cohn-Kanade AU-Coded Expression 数据集，它是一个被广泛接受的 FER 基准。视频采集在受限的环境中，参与者面对录影者，背景为空。CK+ 中的视频由一个从中性表情转换到顶点面部表情的序列组成。最后一帧通常是顶点帧，它具有最强的表达强度。这个数据集包含的表情是愤怒、蔑视、厌恶、恐惧、快乐、悲伤和惊讶。共收集了 118 名受试者的 327 个面部表情视频。

MMI 数据集[19]由 32 位参与者的 326 个面部表情视频组成。有 213 个标有表情标签的视频，包括愤怒、厌恶、恐惧、高兴、悲伤和惊讶。视频帧从中性脸开始。然后在视频中间，表情发展到顶点，在视频结束时回到中性。

AFEW 数据集[20]更接近不受控制的真实世界环境。它是由电影 [56] 的视频剪辑组成的。AFEW 的视频有一个自发的面部表情。AFEW 有七种表达: 愤怒、厌恶、恐惧、快乐、悲伤、惊讶和中性。按照 EmotiW[57] 中的评估协议，有训练集、验证集和测试集。由于它的测试标签不可用，我们按照前面的工作使用验证集进行比较。

5.2 实验结果分析

文中实现了从压缩视频中抽取出残差帧的表示，因此我们也可视化了抽取残差帧的效果，下面为 MMI 数据集上抽取愤怒类别视频的残差帧表示：

图 7: MMI 数据集的残差帧可视化

文中对三个视频人脸表情数据集进行了实验，实验的结果图下所示：

Table 1
Experimental results on the CK+ dataset. Note that in order to make the comparison fair, we do not consider image-based and 3D geometry based experiment setting and models [4,8,16].*Additional FER+ dataset is used. †Additional body language dataset is used.

Method	Accuracy	Landmarks	Ave Test
PHRNN-MSCNN (2017) [50]	98.50	✓	-
C3D-GRU (2019) [51]	97.25	×	-
CTSLSTM (2019) [52]	93.9	✓	-
(N+M)-tuplet (2019) [3]	93.90	✓	12fps
SC (2019) [53]	97.60	✓	-
G2-VER (2019) [54]	97.40	×	-
LBVCNN (2019) [45]	97.38	×	-
NST (2020) [55]†	**99.69**	×	-
Mode VLSTM (2019) [44]	97.42	×	11fps
MIC	**98.95**	×	**35fps**
MIC-MI	97.84	×	**35fps**
MIC-MI+Adv[23]	98.78	×	**35fps**
MIC-$\hat{I}$	98.72	×	**35fps**
MIC+$\mathcal{T}^t$	98.93	×	29fps
FAN+ResNet18* (2019) [7]	99.69	×	10fps
MIC+ResNet6	**99.71**	×	**31fps**

图 8: CK+ 实验结果

Table 3
Experimental results on MMI dataset. Note that in order to make the comparison fair, we do not consider image-based and 3D geometry based experiment setting and models [4,8,16].

Method	Accuracy	Landmarks	Ave Test
3D CNN-DAP (2014) [59]	63.4	✓	
CNN+LSTM (2017) [6]	78.61	×	
CTSLSTM (2019) [52]	78.40	✓	8fps
Mode VLSTM (2019) [44]	79.33	×	10fps
MIC	**81.29**	×	**32fps**
MIC-MI	80.25	×	**32fps**
MIC-MI+Adv[23]	80.98	×	**32fps**
MIC-$\hat{I}$	80.94	×	**32fps**
MIC+$\mathcal{T}^t$	81.24	×	28fps

图 9: MMI 实验结果

Table 4
Experimental results on AFEW dataset. *Additional FER+ dataset is used. † Optical flow is used. ††Additional body language dataset is used.

Method	Accuracy	Model type	Ave Test
Undirectional LSTM (2017) [62]	48.60	Dynamic	-
DenseNet-161 (2018) [64]	51.44	Static	-
CTSLSTM (2019) [52]	51.2	✓	-
C3D-GRU (2019) [51]	49.87	Dynamic	-
DSTA (2019)† [65]	42.98	Dynamic	-
E-ConvLSTM (2019)† [66]	45.29	Dynamic	4fps
NST (2020) [55]††	**99.69**	Dynamic	-
Mode VLSTM (2019) [44]	51.44	Dynamic	11fps
MIC	**53.18**	Dynamic	**34fps**
MIC-MI	52.62	Dynamic	**34fps**
MIC-MI+Adv[23]	53.01	Dynamic	**34fps**
MIC+$\mathcal{T}^r$	**53.18**	Dynamic	30fps
FAN+ResNet18* (2019) [7]	51.18	Static	9fps
MIC+ResNet6	**53.72**	Dynamic	**30fps**

图 10: AFEW 实验结果

复现后也对三个数据集进行了实验，实验结果如下：

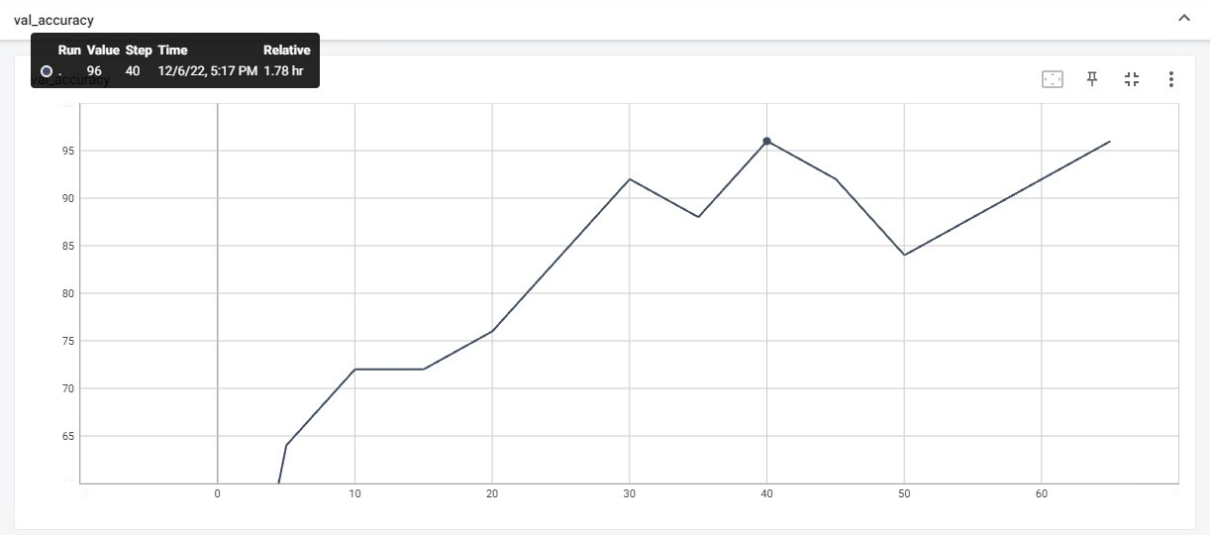


图 11: CK+ 复现结果

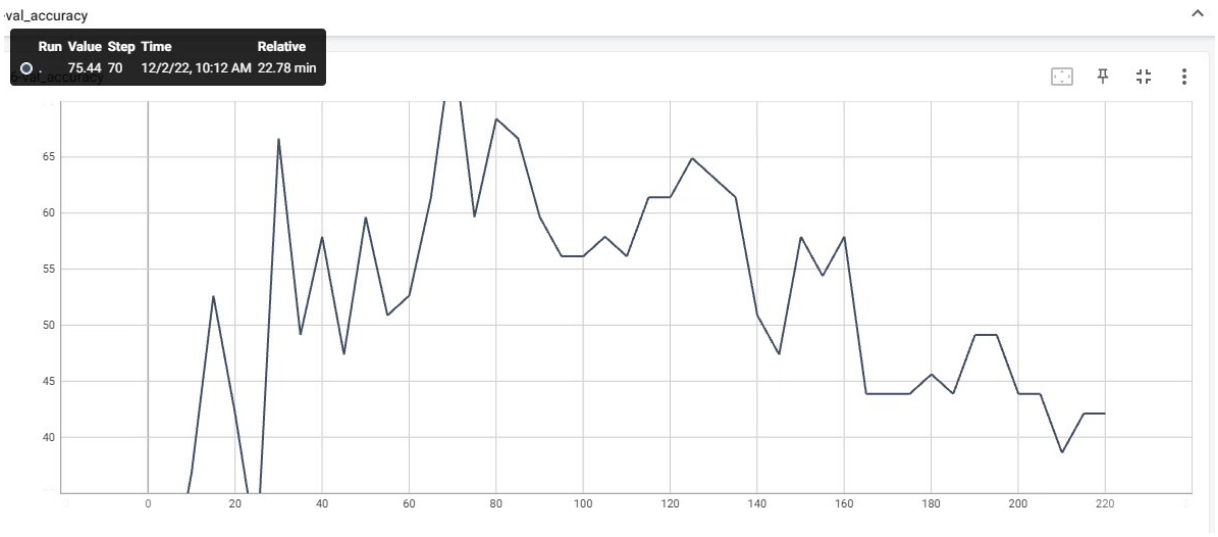


图 12: MMI 复现结果

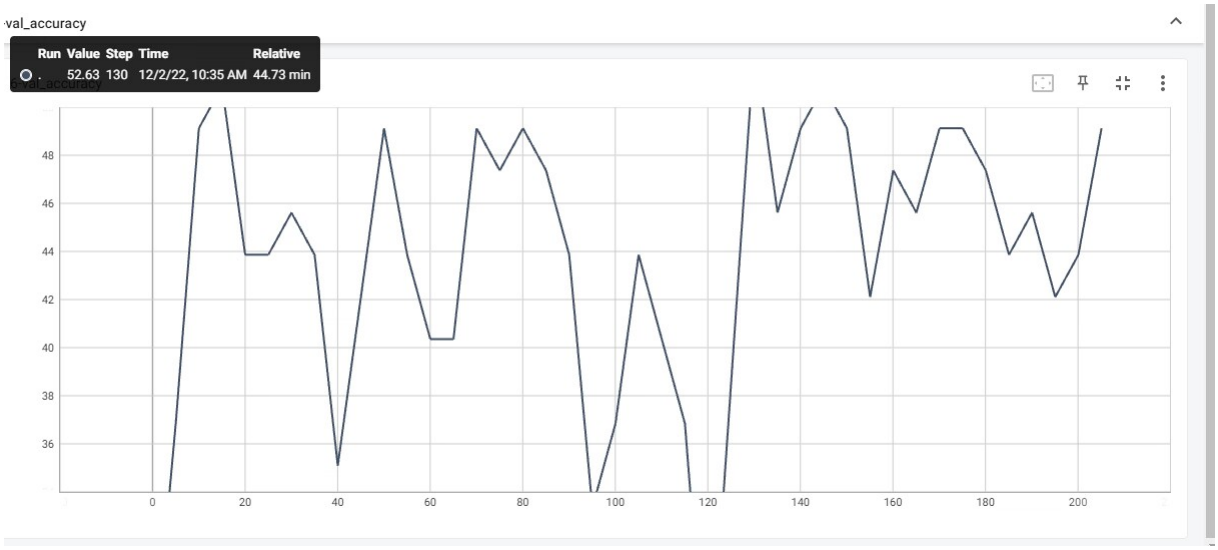


图 13: AFEW 复现结果

由上图可以看到，在 CK+、数据集上在第 40 个 eopch 的时候就到达了 96%, 在 MMI 数据集上，在第 70 个 epoch 的时候达到了 75.44%, 在 AFEW 数据集上，在第 130 个 epoch 的时候就达到了 52.63% 的准确率，相比于原文的效果来说，在 CK+，MMI，AFEW 三个数据集上分别还有 3%、5%、1% 的差距，MMI 上的效果之所以没有那么明显，主要是其数据集的数量相比于其它了两个来说太少了，所以容易导致过拟合，并且他的表情变化是从中性到峰值，再从峰值到中性，峰值帧是不确定的，而其他两个都是从中性到峰值，即最后便是峰值，因此选择帧上面比较容易。

AFEW 效果相对来说是最好的，因为在三个数据集中，只有 AFEW 的数据集是来自野外的，其中因为包含了电影片段，需要进行预处理才能带来效果提升，因此我们对数据进行了清洗，所以相对来说比较容易实现。

另外，由于本次复现过程中的互信息根据其原理搭建一个简单的神经网络，而原论文上没有说明其搭建过程，也没有相应的源代码，所以在性能上面可能不如原论文上的好，同时对于一些超参数比如学习率，帧数、优化器或者实验设备等客观因素综合影响，所以导致了在三个数据集上的效果都没有原论文上的效果好，这后续是本次复现需要进一步考虑和改进的地方。

6 总结与展望

本文旨在直接探讨压缩视频域的面部表情线索，其动机是我们的实际观察的面部肌肉运动可以很好地编码在残差帧。此外，视频压缩可以减少视频中的重复播放模式，使视频的表示具有鲁棒性。在 FER 视频中增加的相关性和减少的复杂性或冗余使计算更加有效。

在实现文章的过程中，由于论文中没有给出残差帧的数量，所以当识别一个视频时候无法确保其效果跟论文的效果一样，并且对于实现互信息评估器是存在许多种方法，而自己在实现过程中也仅是使用了神经网络来学习，这在一定程度上减弱了互信息的效果，最后也深刻认识到自己 code 能力的不足之处，因此后续会加强这方面的学习。对于未来，文章因为只考虑了一个 R 帧，在效果方面虽然有效，但仍然有提升空间，因此未来进一步计划是把压缩视频中的 I 帧和运动矢量一起考虑进来，尝试看看能否在压缩视频上做到全面的提升。

参考文献

[1] Littlewort, Whitehill, WU T, et al. The computer expression recognition toolbox (CERT)[C]//IEEE International Conference on Automatic Face & Gesture Recognition & Workshops. 2011.

[2] ANDERSON K, MCOWAN P W. A real-time automated system for the recognition of human facial expressions[J]. IEEE TRANSACTIONS ON CYBERNETICS, 2006, 36(1): 96-105.

[3] LIU X, GE Y, YANG C, et al. Adaptive metric learning with deep neural networks for video-based facial expression recognition[J]. Journal of Electronic Imaging, 2018, 27(1): 013022.

[4] LIU X, KUMAR B V, JIA P, et al. Hard negative generation for identity-disentangled facial expression recognition[J]. Pattern Recognition, 2019, 88: 1-12.

[5] KIM D H, BADDAR W J, JANG J, et al. Multi-objective based spatio-temporal feature representation learning robust to expression intensity variations for facial expression recognition[J]. IEEE Transactions on Affective Computing, 2017, 10(2): 223-236.

[6] YEO C, AHAMMAD P, RAMCHANDRAN K, et al. Compressed domain real-time action recognition [C]//2006 IEEE Workshop on Multimedia Signal Processing. 2006: 33-36.

[7] LE GALL D. MPEG: A video compression standard for multimedia applications[J]. Communications of the ACM, 1991, 34(4): 46-58.

[8] MENG Z, LIU P, CAI J, et al. Identity-aware convolutional neural network for facial expression recognition[C]//2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017). 2017: 558-565.

[9] LIU X, YANG C, YOU J, et al. Mutual information regularized feature-level frankenstein for discriminative recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021, 44(9): 5243-5260.

[10] BELGHAZI M I, BARATIN A, RAJESWAR S, et al. Mine: mutual information neural estimation[J]. arXiv preprint arXiv:1801.04062, 2018.

[11] KAHOU S E, BOUTHILLIER X, LAMBLIN P, et al. Emonets: Multimodal deep learning approaches for emotion recognition in video[J]. Journal on Multimodal User Interfaces, 2016, 10(2): 99-111.

[12] XU B, FU Y, JIANG Y G, et al. Video emotion recognition with transferred deep feature encodings[C]//proceedings of the 2016 ACM on international conference on multimedia retrieval. 2016: 15-22.

[13] DONAHUE J, ANNE HENDRICKS L, GUADARRAMA S, et al. Long-term recurrent convolutional networks for visual recognition and description[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 2625-2634.

[14] WU C Y, ZAHEER M, HU H, et al. Compressed video action recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6026-6035.

[15] LINSKER R. Self-organization in a perceptual network[J]. Computer, 1988, 21(3): 105-117.

[16] BRAKEL P, BENGIO Y. Learning independent features with adversarial nets for non-linear ica[J]. arXiv preprint arXiv:1710.05050, 2017.

[17] ALI K, HUGHES C E. All-in-one: Facial expression transfer, editing and recognition using a single network[J]. arXiv preprint arXiv:1911.07050, 2019.

[18] KANADE T, COHN J F, TIAN Y. Comprehensive database for facial expression analysis[C]// Proceedings fourth IEEE international conference on automatic face and gesture recognition (cat. No. PR00580). 2000: 46-53.

[19] PANTIC M, VALSTAR M, RADEMAKER R, et al. Web-based database for facial expression analysis [C]//2005 IEEE international conference on multimedia and Expo. 2005: 5-pp.

[20] DHALL A, GOECKE R, JOSHI J, et al. Emotion recognition in the wild challenge 2014: Baseline, data and protocol[C]//Proceedings of the 16th international conference on multimodal interaction. 2014: 461-466.