Dance Categorization

Categorizing dance videos, using only the frames (no sound).

Eren | Yash | Carolyn | Anisha

Dataset and modifications

'Let's Dance', a ~1500 video dataset (and growing) comprised of 10 visually overlapping dance categories. (https://www.cc.gatech.edu/cpl/projects/dance/)

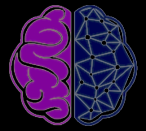
Transformations used:

RandomShortSideScale(min_size=256, max_size=320), RandomCrop(244), RandomHorizontalFlip(p=0.5)

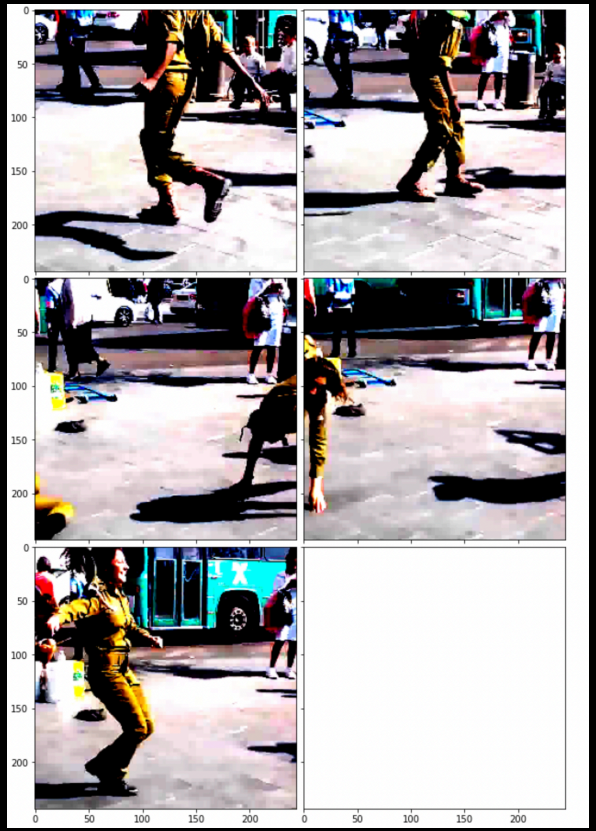
Data splitting:

train/val/test - 80/10/10% - Approx. 1170/143/150 videos

Sampling Technique



- Video training using the entire sequence of video frames (often several hundred) is too memory and compute intense.
- This implementation samples frames evenly from the video (sparse temporal sampling) so that the loaded frames represent every part of the video, with support for arbitrary and differing video lengths within the same dataset.
- Paper: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Farxiv.org%2Fpdf%2F1608.00859.pdf&clen=1853758&chunk=true

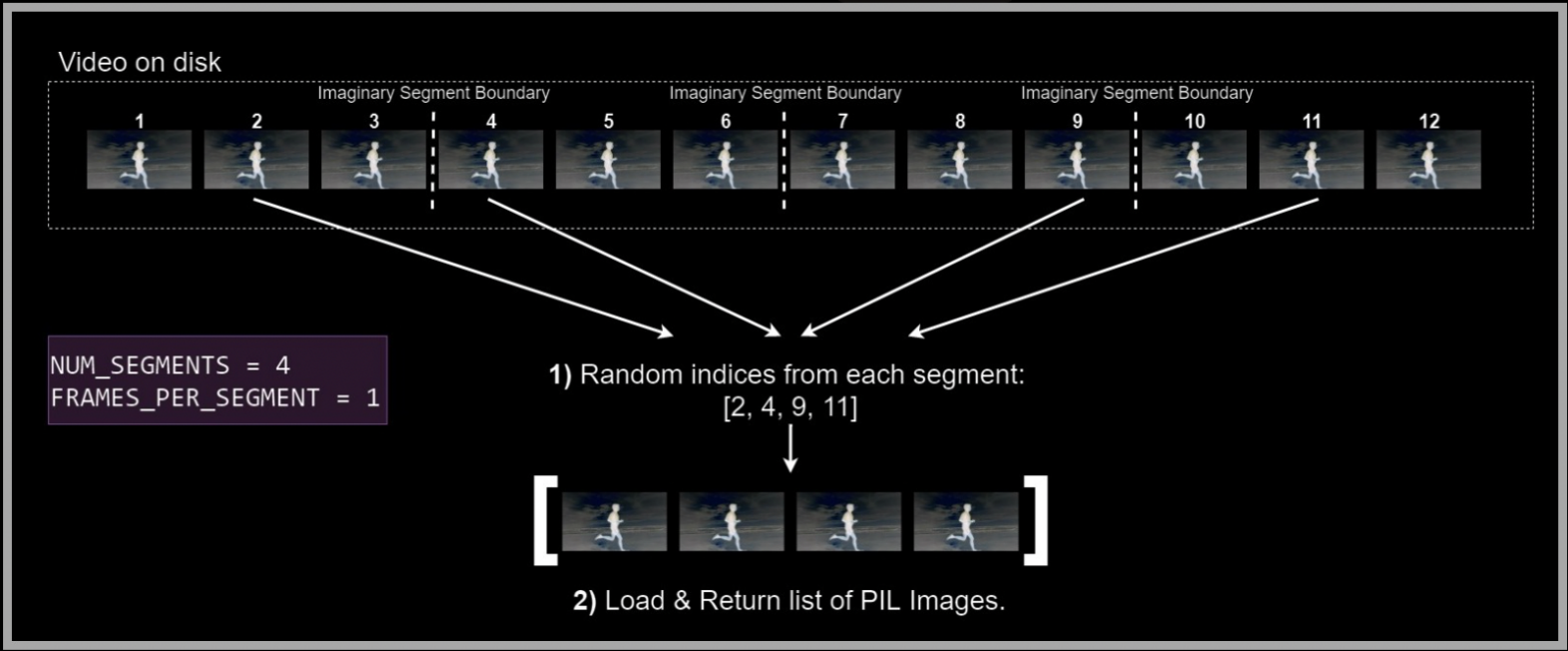

0
50
100
150
200
0
50
100
150
200

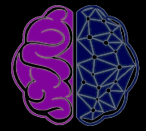

0
50
100
150
200
0
50
100
150
200

3. Video Frame Sampling Method

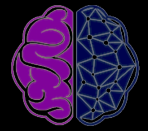
When loading a video, only a number of its frames are loaded. They are chosen in the following way:

1. The frame index range [START_FRAME, END_FRAME] is divided into NUM_SEGMENTS even segments. From each segment, a random start-index is sampled from which FRAMES_PER_SEGMENT consecutive indices are loaded. This results in NUM_SEGMENTS*FRAMES_PER_SEGMENT chosen indices, whose frames are loaded as PIL images and put into a list and returned when calling `dataset[i]`.

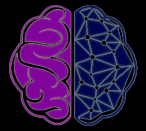




```
def _get(self, record, indices):
    """
    Loads the frames of a video at the corresponding
    indices.
    Args:
        record: VideoRecord denoting a video sample.
        indices: Indices at which to load video frames from.
    Returns:
        1) A list of PIL images or the result
        of applying self.transform on this list if
        self.transform is not None.
        2) An integer denoting the video label.
    """

    indices = indices + record.start_frame
    images = list()
    image_indices = list()
    for seg_ind in indices:
        frame_index = int(seg_ind)
        for i in range(self.frames_per_segment):
            seg_img = self._load_image(record.path, frame_index)
            images.extend(seg_img)
            image_indices.append(frame_index)
            if frame_index < record.end_frame:
                frame_index += 1

    # sort images by index in case of edge cases where segments overlap each other because the overall
    # video is too short for num_segments*frames_per_segment indices.
    #_, images = (list(sorted_list) for sorted_list in zip(*sorted(zip(image_indices, images))))

    if self.transform is not None:
        images = self.transform(images)

    return images, record.label
```

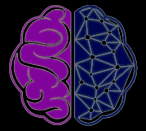


```
def __getitem__(self, index):
    """
    For video with id index, loads self.NUM_SEGMENTS * self.FRAMES_PER_SEGMENT
    frames from evenly chosen locations.
    Args:
        index: Video sample index.
    Returns:
        a list of PIL images or the result
        of applying self.transform on this list if
        self.transform is not None.
    """
    record = self.video_list[index]

    if not self.test_mode:
        segment_indices = self._sample_indices(record) if self.random_shift else self._get_val_indices(record)
    else:
        segment_indices = self._get_test_indices(record)

    return self._get(record, segment_indices)
```



```
def _sample_indices(self, record):
    """
    For each segment, chooses an index from where frames
    are to be loaded from.
    Args:
        record: VideoRecord denoting a video sample.
    Returns:
        List of indices of where the frames of each
        segment are to be loaded from.
    """

    segment_duration = (record.num_frames - self.frames_per_segment + 1) // self.num_segments
    if segment_duration > 0:
        offsets = np.multiply(list(range(self.num_segments)), segment_duration) + np.random.randint(segment_duration, size=self.num_segments)

    # edge cases for when a video has approximately less than (num_frames*frames_per_segment) frames.
    # random sampling in that case, which will lead to repeated frames.
    else:
        offsets = np.sort(np.random.randint(record.num_frames, size=self.num_segments))

    return offsets
```

Model: 3D ResNet 50

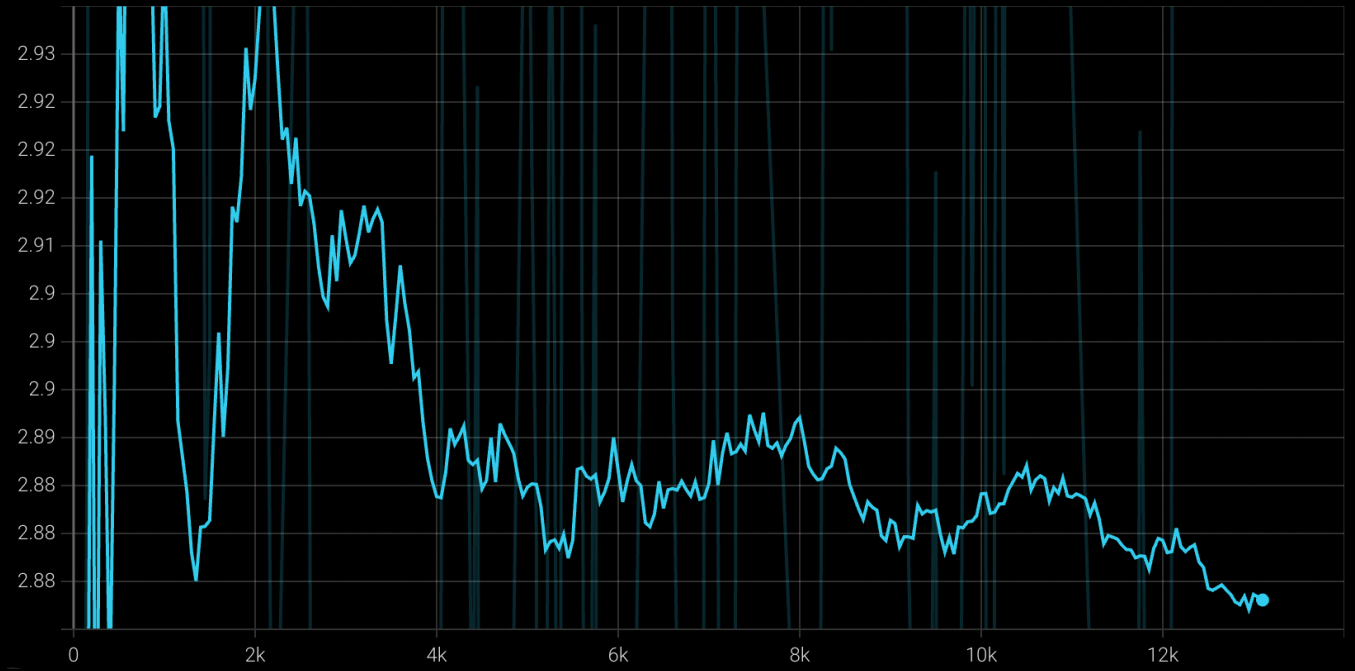


ResNet 3D is a type of model for video that employs 3D convolutions. This model collection consists of two main variants:

- Mixed convolution (MC) consists in employing 3D convolutions in the early layers of the network, with 2D convolutions in the top layers. The rationale behind this design is that motion modeling is a low/mid-level operation that can be implemented via 3D convolutions in the early layers of a network, and spatial reasoning over these mid-level motion features (implemented by 2D convolutions in the top layers) leads to accurate action recognition. We show that MC ResNets yield roughly a 3-4% gain in clip-level accuracy over 2D ResNets of comparable capacity and they match the performance of 3D ResNets, which have 3 times as many parameters.
- The “(2+1)D” convolutional block explicitly factorizes 3D convolution into two separate and successive operations, a 2D spatial convolution and a 1D temporal convolution. The first advantage is an additional nonlinear rectification between these two operations. This effectively doubles the nonlinearities compared to a network using full 3D convolutions for the same number of parameters, thus rendering the model capable of representing more complex functions. The second potential benefit is that the decomposition facilitates the optimization, yielding in practice both a lower training loss and a lower testing loss.

Paper: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Farxiv.org%2Fpdf%2F1711.11248v3.pdf&clen=2395882&chunk=true

Training



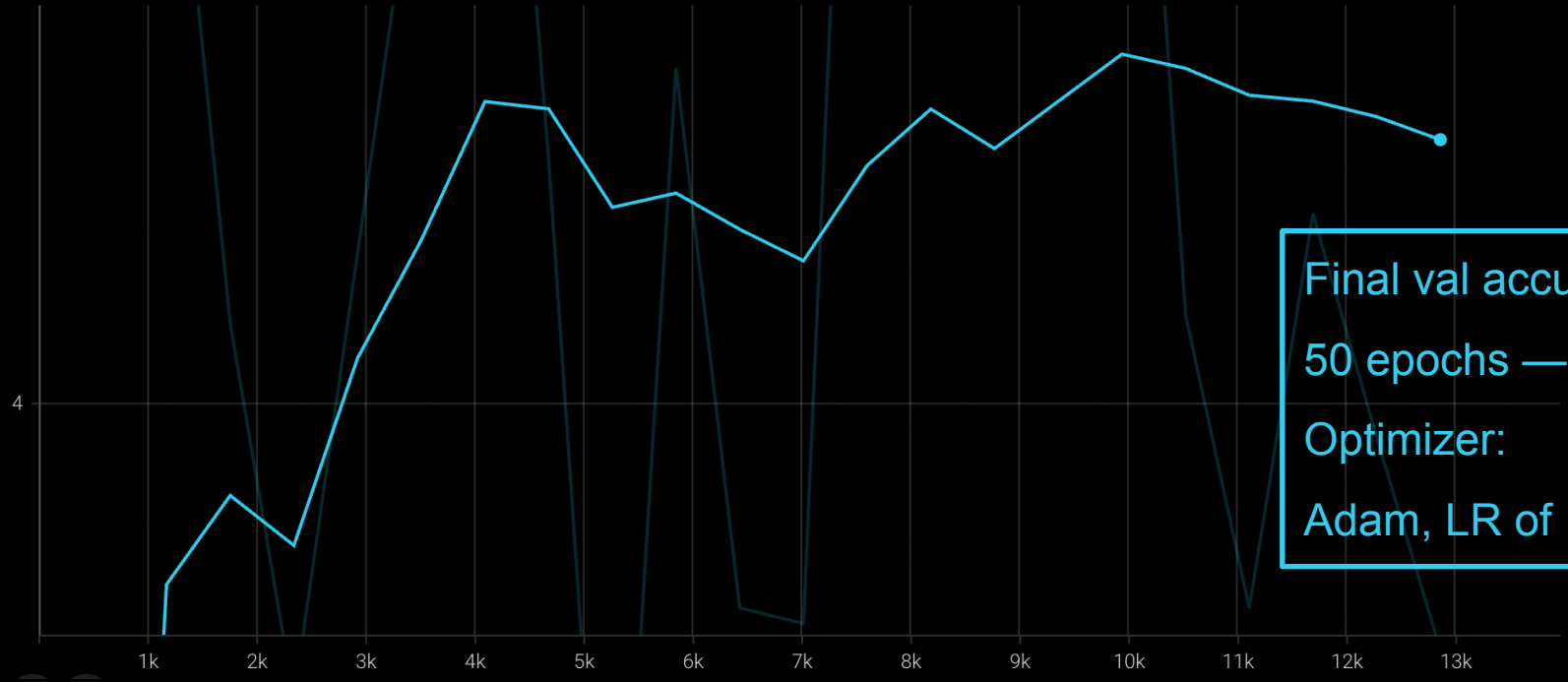
Loss:
Cross Entropy Loss;

Training:
Used 1 GPU;

Trained for 50 epochs

This is trash but then again, anything related to videos takes a lot of epochs. Well more than 50, maybe a 1000 LOL

Results



Final val accuracy:

50 epochs — 52.6%

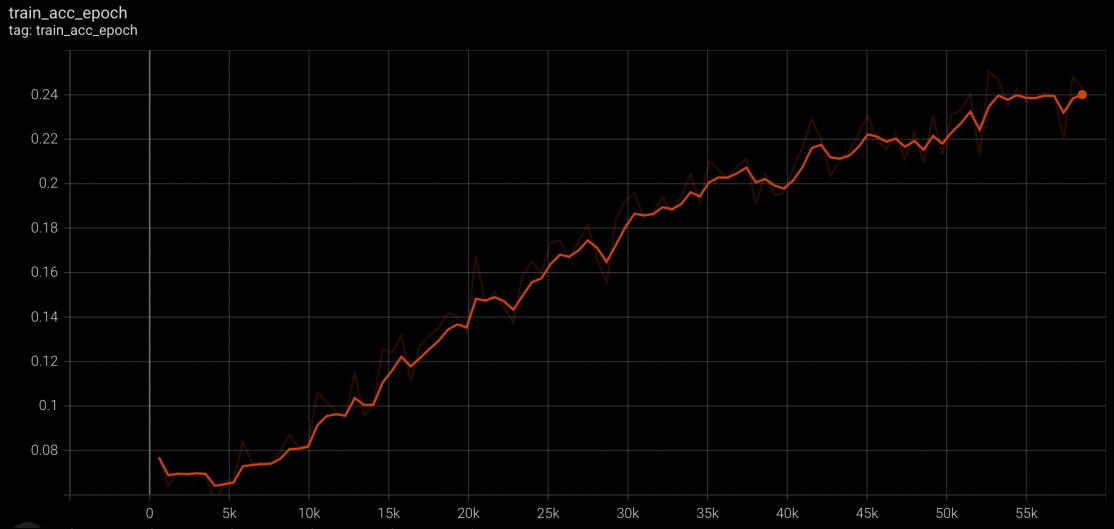
Optimizer:

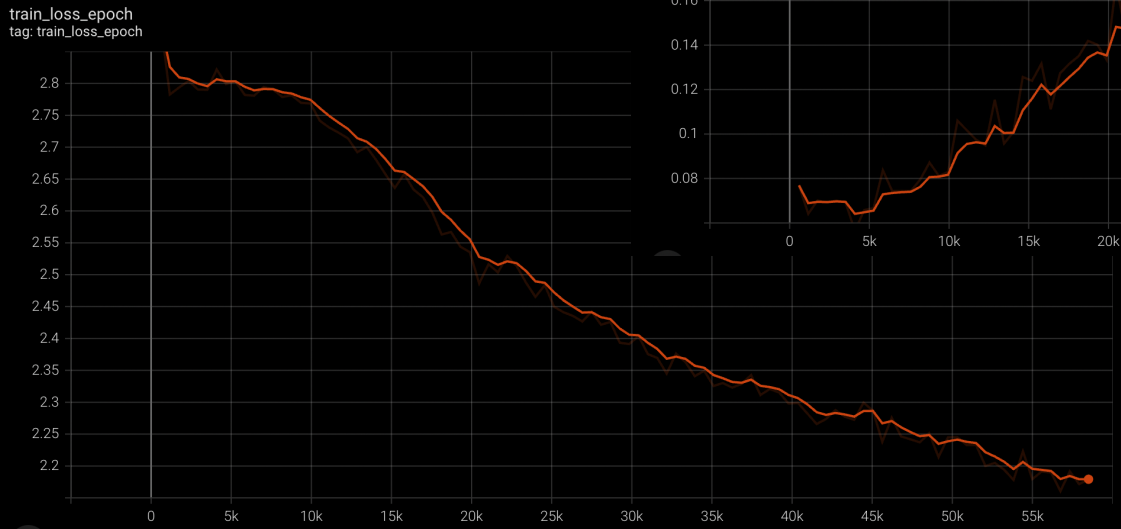
Adam, LR of 1e-1

Obviously not a good, conclusive accuracy but I think the curve shows that given adequate time and compute, the project could probably have succeeded

Results

Tried learning rate 1e-2 and trained for 100 epochs with val accuracy of about 24.48%.





Progressed a lot slower than before but fewer fluctuations.


Please don't kick us out