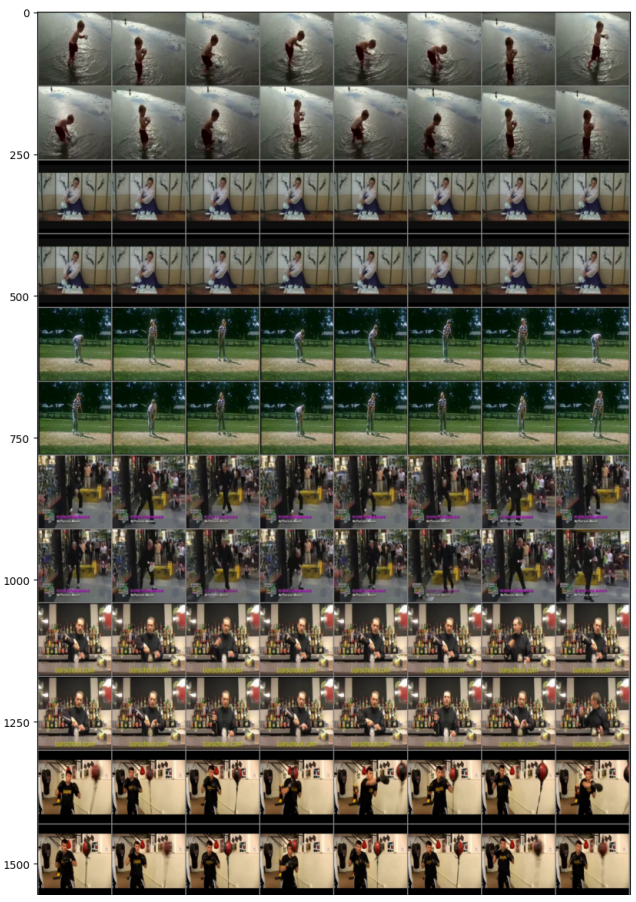
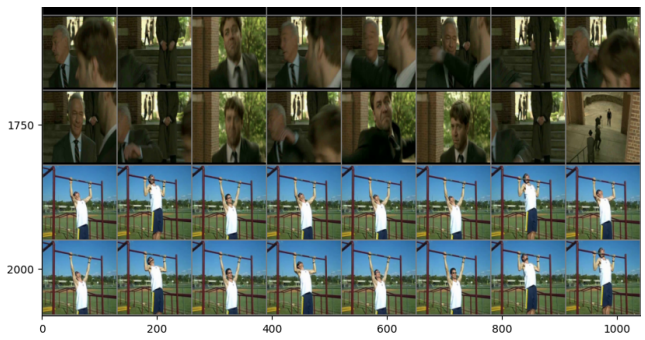
VCL Mini Project - 2

Problem Statement

The goal is to create and deploy a Video Activity Classifier with 5 classes and evaluate its performance under particular conditions.

Dataset Description





Classes/Number	Training - 319	Testing - 80
laugh	63	17
pullup	67	12
punch	65	15
pick	64	16
pour	60	20

The Images Sequences Corresponding to all classes are shown.

RNN based Video Activity Classification

1. Introduction

This report outlines the code and methodology for creating a video activity classification model using Recurrent Neural Networks (RNN). The primary steps in this approach include extracting frames from videos, creating a dataset from these frames, defining an RNN model, and then training and validating the model.

---

2. Extracting Frames from Videos

Video Dataset

```
data/
│
├── laugh/                  #Category of videos.
│   ├── video1.mp4
│   ├── video2.mp4
│   ...
│
└── punch/
├── videoA.mp4
├── videoB.mp4
...
```

Corresponding Image Dataset

```
image_data/
├── laugh/              #Category of image data corresponding to videos.
│   ├── video1/         #Images (16) extracted from video1.mp4
│   │   ├── 0.jpg
│   │   ├── 1.jpg
│   │   ...
│   ├── video2/
│   │   ├── 0.jpg
│   │   ├── 1.jpg
│   │   ...
│   ...
│
└── punch/
├── videoA/
│   ├── 0.jpg
│   ├── 1.jpg
│   ...
├── videoB/
...
```

- **Path Setup:** Set up paths for storing the original video data and the extracted image frames.
- **Video-to-Frame Conversion:** For each video, the code extracts and saves 16 equidistant frames. This ensures a uniform number of frames for each video sample.

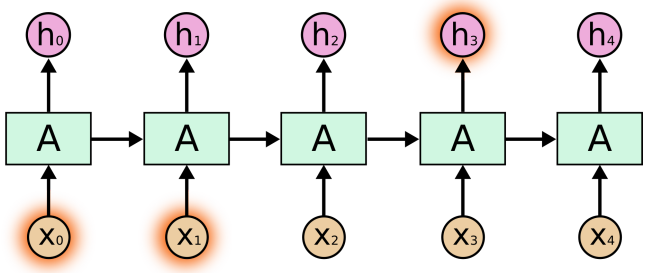
---

3. Creating the Dataset

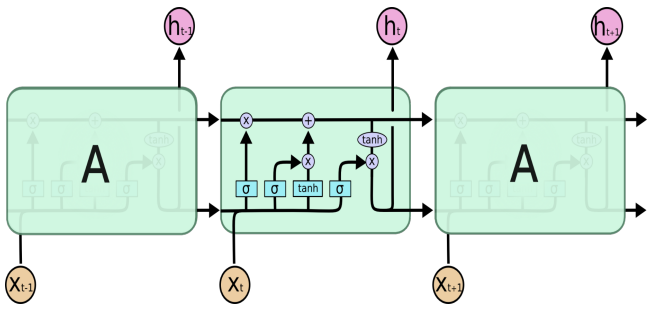
- **Dataset Structure:** The `VideoDataset` class is implemented to manage video data regarding individual frames.
- **Transformations:** All images are resized to 128x128 pixels and normalized.
- **Data Sanity Check**
  - A basic check is performed to confirm that the shape of the images and labels in the train loader is as expected.

---

5. RNN Model Definition



RNN



LSTM

- **Feature Extraction:** CNN layers extract features from the video frames.
- **Sequence Modeling:** LSTM layers analyze the sequence of frame features.
- **Classification:** Fully connected layers produce the final classification results.

Model Architecture: NetRnn

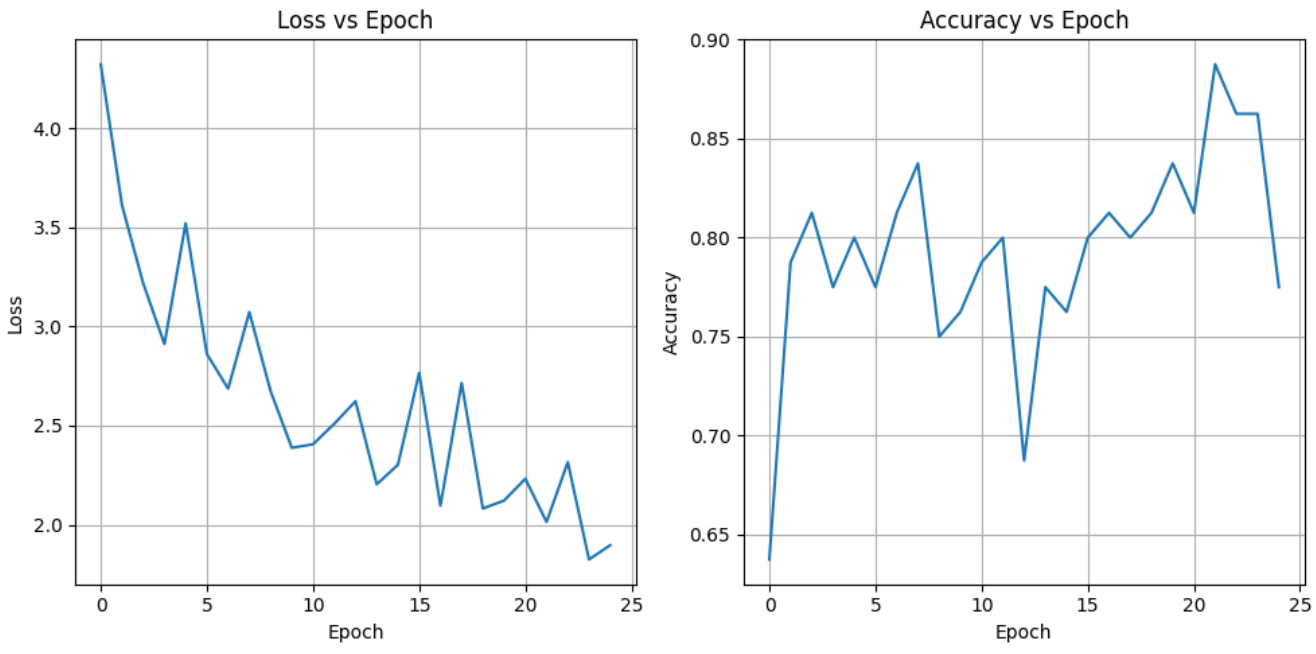
1. **Feature Extraction (CNN part)**:
   - **Convolutional Layer**: 3 input channels, 32 output channels, kernel size of 3, stride of 2.
   - **ReLU Activation**
   - **MaxPooling**: Kernel size of 3, stride of 2.
   - **Convolutional Layer**: 32 input channels, 64 output channels, kernel size of 3, stride of 2.
   - **ReLU Activation**
   - **MaxPooling**: Kernel size of 3, stride of 2.
   - **Batch Normalization**: 64 output channels.
2. **Recurrent Neural Network (RNN part)**:
   - **LSTM Layers**: Input size of 64x7x7, hidden size of 128, 2 layers, with batch-first configuration.
3. **Fully Connected Layer**:
   - **Linear Layer**: 128 input features, 5 output features (assuming 5 is the number of classes or activities to classify).

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
NetRnn                                   [8, 5]                    --
├─Sequential: 1-1                        [128, 64, 7, 7]           --
│    └─Conv2d: 2-1                       [128, 32, 63, 63]         896
│    └─ReLU: 2-2                         [128, 32, 63, 63]         --
│    └─MaxPool2d: 2-3                    [128, 32, 31, 31]         --
│    └─Conv2d: 2-4                       [128, 64, 15, 15]         18,496
│    └─ReLU: 2-5                         [128, 64, 15, 15]         --
│    └─MaxPool2d: 2-6                    [128, 64, 7, 7]           --
├─BatchNorm2d: 1-2                       [128, 64, 7, 7]           128
├─LSTM: 1-3                              [8, 16, 128]              1,804,288
├─Linear: 1-4                            [8, 5]                    645
==========================================================================
Total params: 1,824,453
Trainable params: 1,824,453
Non-trainable params: 0
Total mult-adds (G): 1.22
==========================================================================
Input size (MB): 25.17
Forward/backward pass size (MB): 148.14
Params size (MB): 7.30
Estimated Total Size (MB): 180.61
==========================================================================
```

6. Training the Model with Full Frame Validation

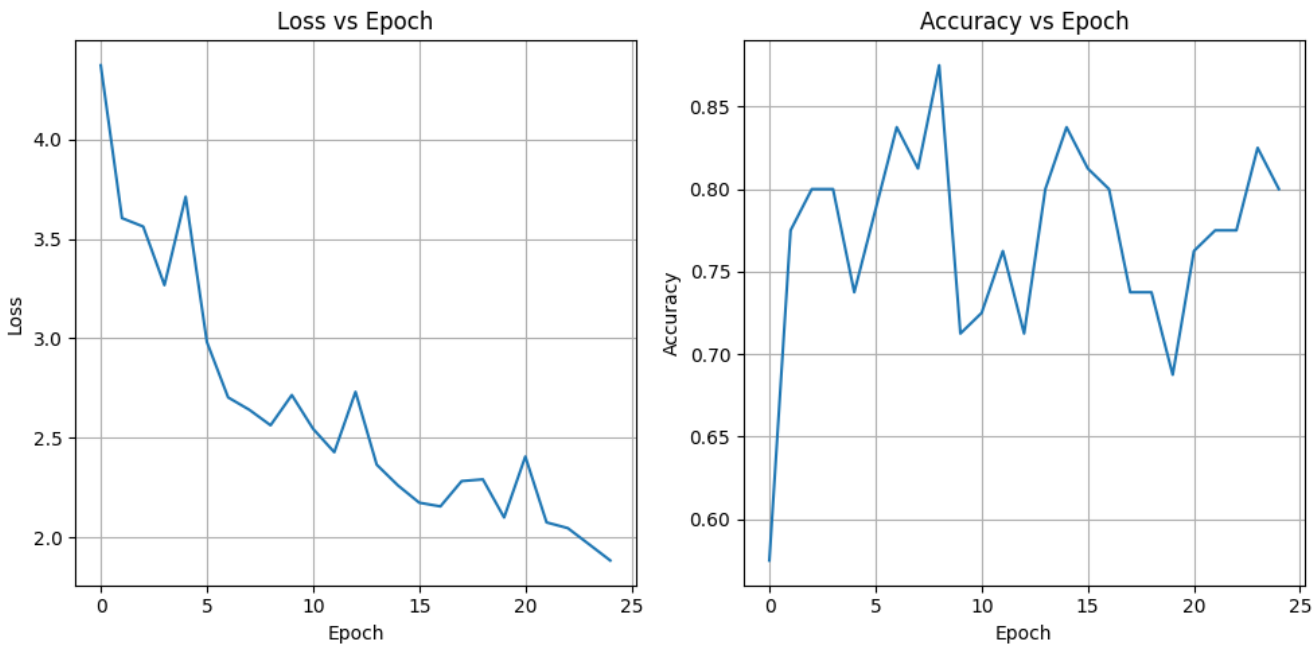
- **Loss Function:** Cross-entropy loss is used, with L1 regularization on model parameters.
- **Optimization:** Adam optimizer is employed with a learning rate of 0.001.

- **Training Loop:** The model is trained for 25 epochs. The loss and accuracy on the validation dataset are tracked after each epoch.



7. Training the Model with Limited Frame Validation

The model is re-trained, but only a subset (2 to 8) of the 16 frames is used during validation. This simulates scenarios where not all video frames might be available or used for prediction.



8. Conclusion

This approach recognizes the content in individual frames and captures the temporal relationships between frames, which is crucial for accurate video activity recognition. Future work might involve integrating more advanced models or fine-tuning for specific video datasets.

9. Intuition and Insights on Limited Frame Validation vs. Full Frame Validation

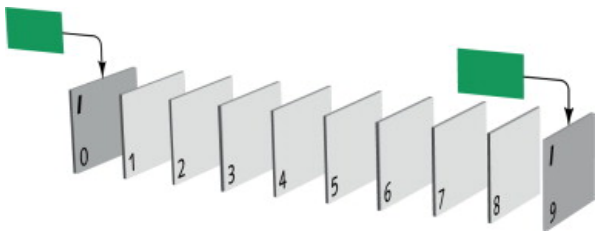
9.1. The Nature of Video Data

Videos are inherently sequential data where each frame continues the previous one. This sequence provides

9.2. The Role of RNNs

Recurrent Neural Networks (RNNs) like LSTMs are designed to handle sequences, capturing temporal

context and helps in understanding the activity over time. When we remove specific frames, we essentially take away pieces of this context, which can lead to losing important information.

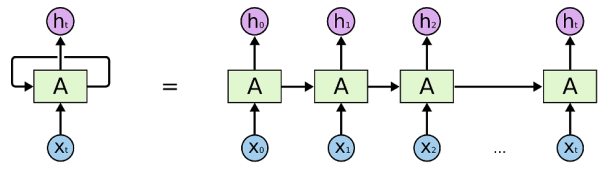


9.3. Temporal Resolution

Video activities can be subtle and may require a complete sequence of frames to be accurately identified. For instance, a hand movement only makes sense when seen in continuation.

If frames are randomly picked from this sequence, the movement might appear disjointed, leading to potential misclassifications.

dependencies in the data. When we reduce the number of frames in the sequence, the RNN receives less information, affecting its ability to capture these dependencies and thus potentially reducing its accuracy.



9.4. Model Expectation vs. Reality

The model was trained on a complete set of frames (16 frames per video). When validated with a limited set of frames, there's a mismatch between what the model expects (sequence length and continuity) and what it receives.

This disparity can affect the model's performance.

> 💡 In the limited frame validation, frames are selected randomly. Depending on the randomness, crucial frames that capture the essence of an activity might be left out. This can be compared to trying to understand a story by reading random pages from a book; the narrative may not always make sense.

---

10. Conclusion on Frame Validation Insights

The difference in performance between full frame and limited frame validation can be attributed to the nature of video data and the way RNNs process sequential data. While it's beneficial to see how the model performs under limited data scenarios (like in real-world applications where not all frames might be available), it's essential to understand that a reduced sequence might not always capture the entirety of the activity, leading to reduced performance.

Deliverables Required are Provided in the Sections Below

Codebase:

- Training Code
- Deployment Code
- Saved Model

https://github.com/eternal-f1ame/MP-2-VCL

WebApp:

https://video-activity-classification.streamlit.app/