HUMAN ACTION RECOGNITION USING 3D CNN

A mini project by
Sourabh Sathe
MT23MCS003
MTech CSE
VNIT

PROBLEM STATEMENT

Type: Video Classification Problem

The task of video classification for human action recognition within DL frameworks presents a challenge due to the temporal complexity inherent in video data. Unlike static images, videos contain rich temporal information, requiring models to effectively capture and analyze dynamic sequences of actions over time. This temporal complexity aggravates the limitations of simplistic CNNs, as they fail to account for the temporal dynamics. Simple CNN architectures lack mechanisms to capture time dependencies and often struggle with maintaining context over extended sequences, leading to suboptimal performance and limited generalization capabilities. Overcoming these challenges using advanced architectures is crucial for video related applications like surveillance and human-computer interaction.

APPROACH

Making use of 3D CNN

Unlike 2D CNN that operate on single image with dimensions (height, width), the 3D CNN operates on video volume (time, height, width) . The most obvious approach to this problem would be replace each 2D convolution (layers.Conv2D) with a 3D convolution (layers.Conv3D)

An optimization can be performed using concept of **Separable Kernels**. Instead of directly using a 3D conv kernel, we use a (2+1)D conv kernel. The (2 + 1)D conv allows for the decomposition of the spatial and temporal dimensions, therefore creating two separate steps. An advantage of this approach is that factorizing the convolutions into spatial and temporal dimensions saves parameters.

Using 3D CNN, the operation takes (time * height * width * channels) inputs and produces channels outputs (assuming the number of input and output channels are the same).

So a 3D convolution layer with a kernel size of (3 x 3 x 3) would need a weight-matrix with (**27 * channels ** 2)** entries. In the (2 + 1)D convolution the spatial convolution takes in data of the shape (1, width, height), while the temporal convolution takes in data of the shape (time, 1, 1). For example, a (2 + 1)D convolution with kernel size (3 x 3 x 3) would need weight matrices of size (9 * channels**2) + (3 * channels**2) = (**12 * channels ** 2)** , less than half as many as the full 3D convolution.

DATASET

UCF101

Link: https://www.crcv.ucf.edu/data/UCF101.php

Already preprocessed dataset of 13320 action videos, collected from YouTube for action recognition. Has 101 action categories. Each video has fixed frame rate of 25 FPS with the resolution of 320 × 240. Actions categorized as folders. Each folder contains 50-60 videos of the action. Some action categories are: Apply Lipstick, Archery, Baby Crawling, Balance Beam, Band Marching, Baseball Pitch, Basketball Dunk, Bench Press, etc.

IMPLEMENTATION

Input: 100 x 100 down sampled videos of human actions
Dataset: subset of UCF101 -> only 10 action categories
Model: custom ResNet -> each 2D conv layer replaced by a (2+1)D conv layer
Output: label of action category
Performance metrics: accuracy, loss, validation loss
Loss function: Sparse Categorical Cross Entropy
Optimizer: Adam

PARAMETERS

Total params: 443,322
Trainable params: 443,290
Non-trainable params: 32

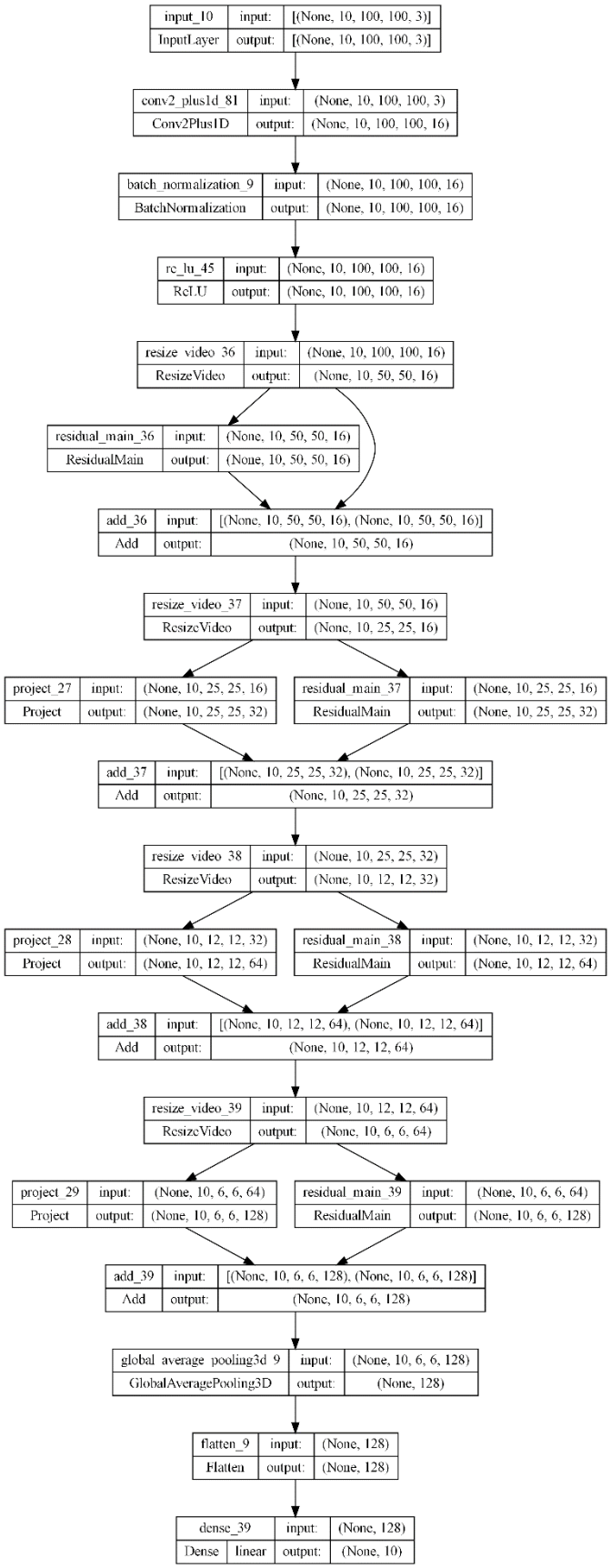
RESULTS

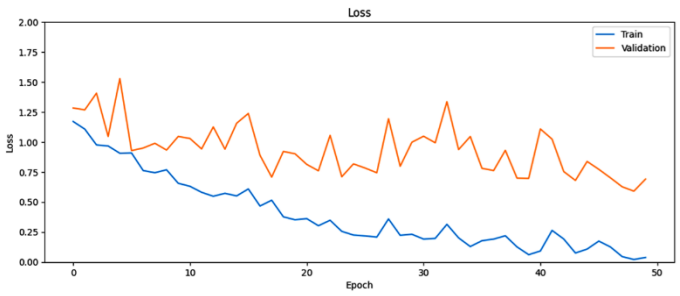
Training set

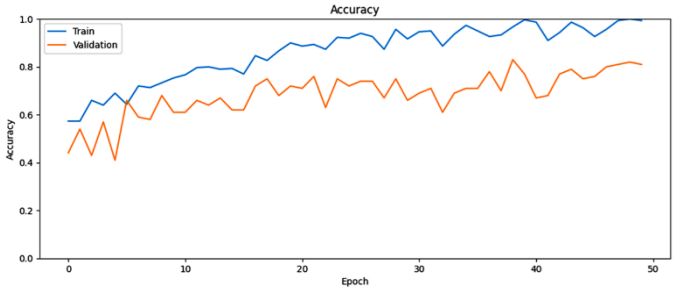
- loss: 0.0379
- accuracy: 0.9733
- val_loss: 0.6912
- val_accuracy: 0.8100

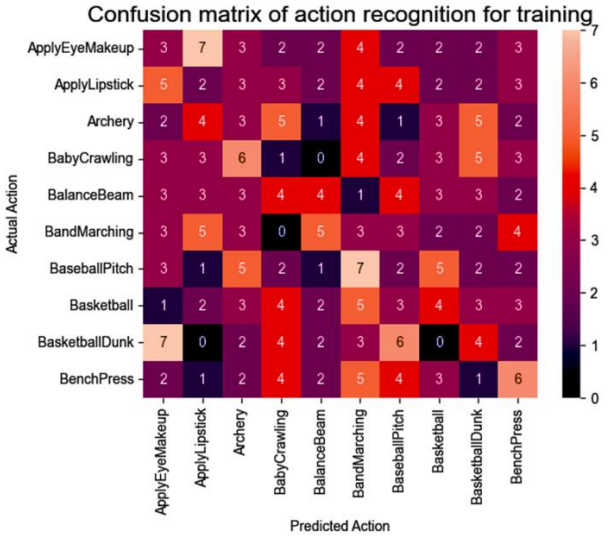
Testing set

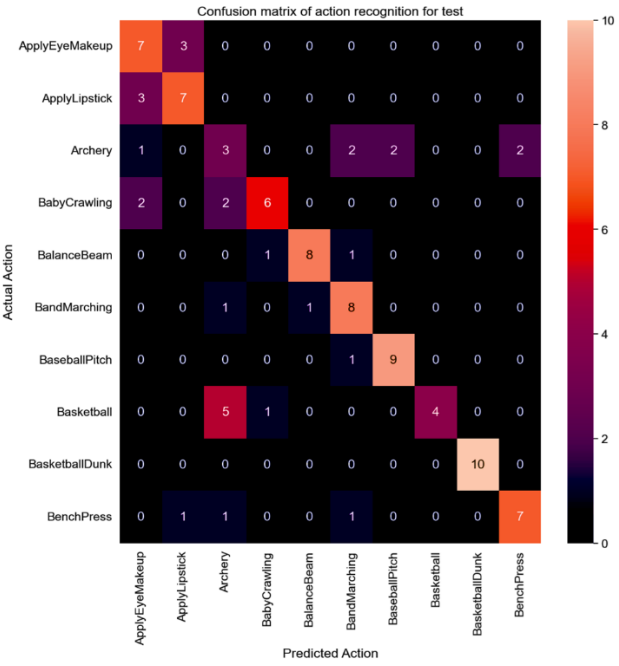
- loss: 1.0235
- accuracy: 0.6998











REFERENCES

[1] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, 'A Closer Look at Spatiotemporal Convolutions for Action Recognition', arXiv [cs.CV]. 2018.

[2] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, 'Learning Spatiotemporal Features with 3D Convolutional Networks', arXiv [cs.CV]. 2015.

[3] C. Feichtenhofer, A. Pinz, and R. P. Wildes, 'Spatiotemporal Residual Networks for Video Action Recognition', CoRR, vol. abs/1611.02155, 2016.

[4] 'Video classification with a 3D convolutional neural network', TensorFlow. [Online]. Available: https://www.tensorflow.org/tutorials/video/video_classification. [Accessed: 13-May-2024].