DSAA 5002 - Data Mining and Knowledge Discovery in Data Science

Final Exam Report – Q3 Short Video Classification

50015940 Jiaxiang Gao

1. Algorithm Overview

The algorithm employs a 3D Convolutional Neural Network (CNN) to classify videos. The process is divided into several key steps:

I. Video Frame Extraction:

Utilizes FFmpeg to extract key frames from videos. The extract_key_frames_ffmpeg function takes a video file and saves its key frames at a specified rate (default is one frame per second). The process_videos_with_ffmpeg function processes all videos in each folder using FFmpeg.

II. Label Loading and Preprocessing:

Reads a tag file that contains labels for each video. The load_labels function creates a DataFrame mapping video names to their corresponding labels.

III. Frame Loading and Preprocessing:

Loads and preprocesses video frames. The load_video_frames function reads the extracted frames, resizes them, and ensures uniformity in the number of frames per video. Frames are normalized, and labels (if available) are one-hot encoded.

IV. 3D CNN Model Building:

A Sequential model is built using Keras. The model contains layers of 3D convolutions, max pooling, flattening, dense layers, and dropout for regularization. The final layer uses a SoftMax activation function for classification.

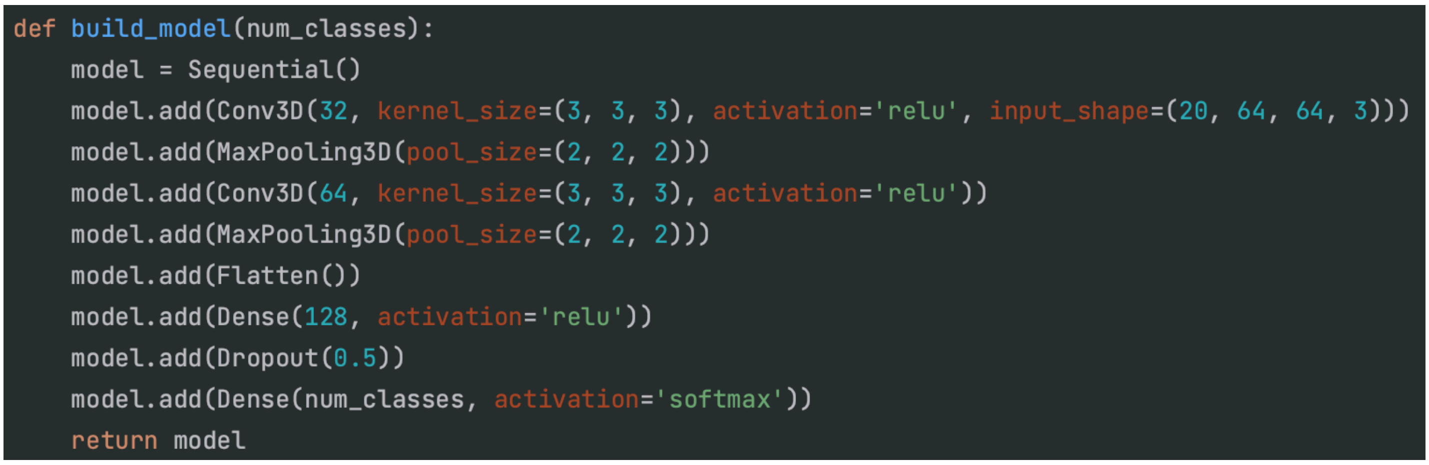
V. Model Training and Evaluation:

The dataset is split into training and testing sets. The model is compiled with categorical cross-entropy loss and the Adam optimizer. Training is performed over 50 epochs with batch size 32.

2. Model Architecture:

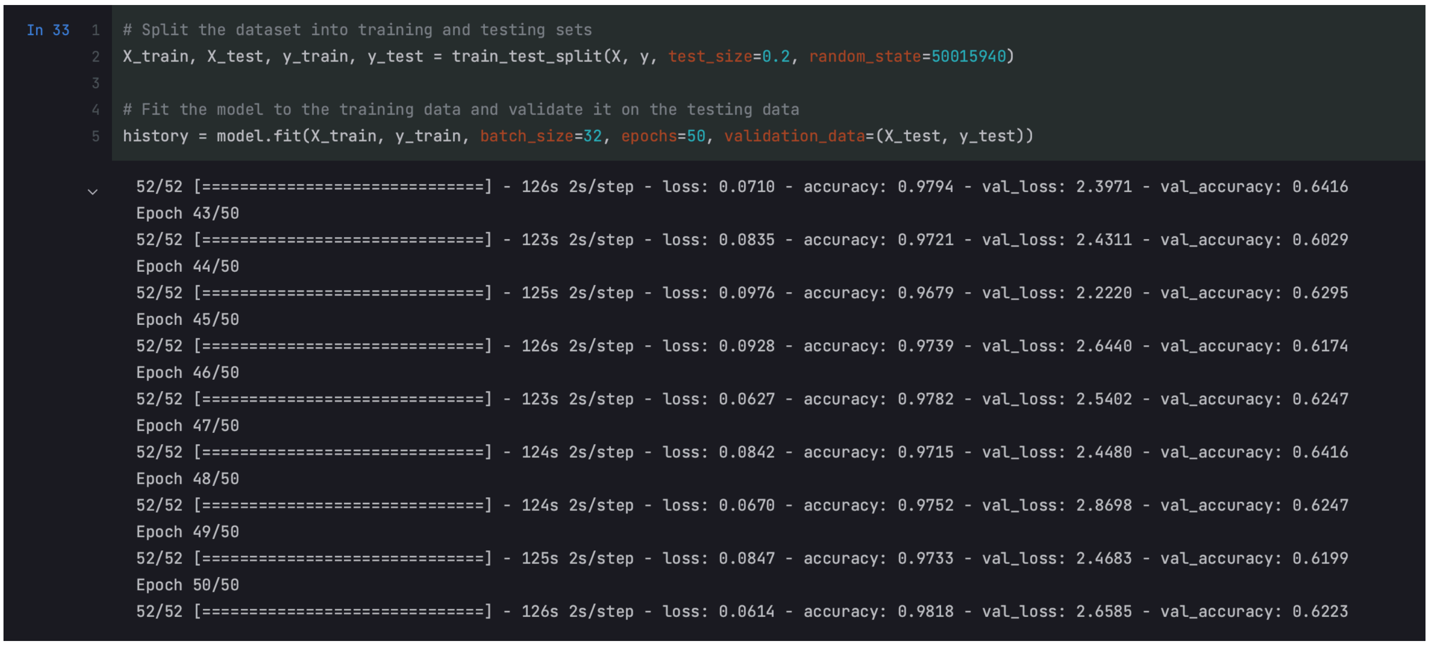
- Input Shape: (20, 64, 64, 3) for 20 frames, each 64x64 in size, with 3 color channels.
- Convolutional Layers: Multiple layers with varying number of filters (32, 64).
- Max Pooling Layers: Reduce spatial dimensions.
- Flatten Layer: Converts 3D feature maps to 1D feature vectors.
- Dense Layers: Fully connected layers for classification.
- Dropout: Regularization to prevent overfitting.

- Output Layer: Softmax activation for multi-class classification



```
def build_model(num_classes):
    model = Sequential()
    model.add(Conv3D(32, kernel_size=(3, 3, 3), activation='relu', input_shape=(20, 64, 64, 3)))
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Conv3D(64, kernel_size=(3, 3, 3), activation='relu'))
    model.add(MaxPooling3D(pool_size=(2, 2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    return model
```

3. Model Training



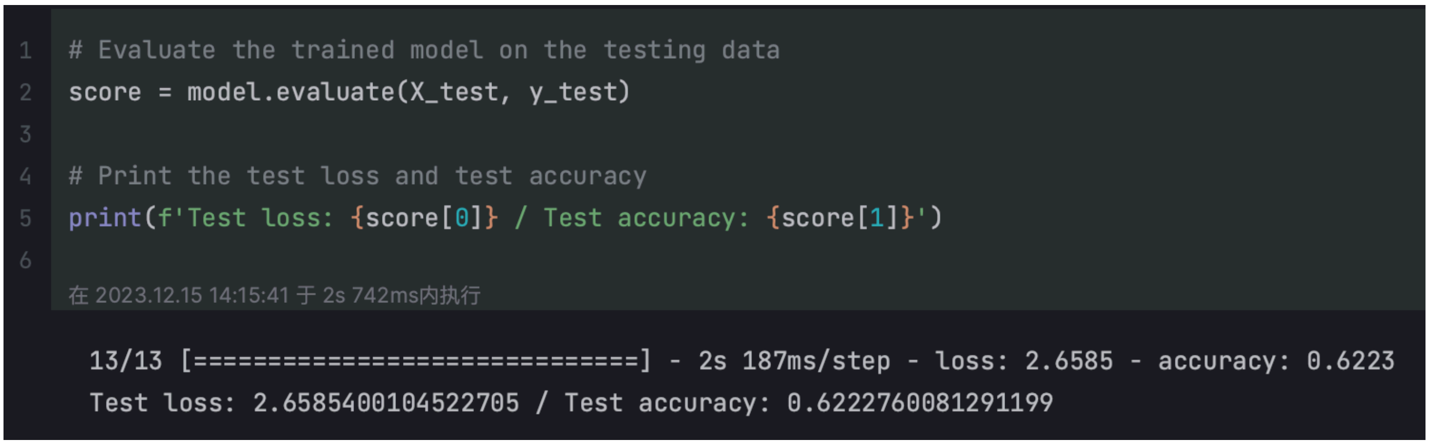
```
In 33
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=50015940)

# Fit the model to the training data and validate it on the testing data
history = model.fit(X_train, y_train, batch_size=32, epochs=50, validation_data=(X_test, y_test))
```

```
52/52 [==============================] - 126s 2s/step - loss: 0.0710 - accuracy: 0.9794 - val_loss: 2.3971 - val_accuracy: 0.6416
Epoch 43/50
52/52 [==============================] - 123s 2s/step - loss: 0.0835 - accuracy: 0.9721 - val_loss: 2.4311 - val_accuracy: 0.6029
Epoch 44/50
52/52 [==============================] - 125s 2s/step - loss: 0.0976 - accuracy: 0.9679 - val_loss: 2.2220 - val_accuracy: 0.6295
Epoch 45/50
52/52 [==============================] - 126s 2s/step - loss: 0.0928 - accuracy: 0.9739 - val_loss: 2.6440 - val_accuracy: 0.6174
Epoch 46/50
52/52 [==============================] - 123s 2s/step - loss: 0.0627 - accuracy: 0.9782 - val_loss: 2.5402 - val_accuracy: 0.6247
Epoch 47/50
52/52 [==============================] - 124s 2s/step - loss: 0.0842 - accuracy: 0.9715 - val_loss: 2.4480 - val_accuracy: 0.6416
Epoch 48/50
52/52 [==============================] - 124s 2s/step - loss: 0.0670 - accuracy: 0.9752 - val_loss: 2.8698 - val_accuracy: 0.6247
Epoch 49/50
52/52 [==============================] - 125s 2s/step - loss: 0.0847 - accuracy: 0.9733 - val_loss: 2.4683 - val_accuracy: 0.6199
Epoch 50/50
52/52 [==============================] - 126s 2s/step - loss: 0.0614 - accuracy: 0.9818 - val_loss: 2.6585 - val_accuracy: 0.6223
```

4. Model Evaluate

Accuracy in spilt training set: 0.62



```
# Evaluate the trained model on the testing data
score = model.evaluate(X_test, y_test)

# Print the test loss and test accuracy
print(f'Test loss: {score[0]} / Test accuracy: {score[1]}')

```

在 2023.12.15 14:15:41 于 2s 742ms内执行

```
13/13 [==============================] - 2s 187ms/step - loss: 2.6585 - accuracy: 0.6223
Test loss: 2.6585400104522705 / Test accuracy: 0.6222760081291199
```