**Video Classification Using CNN and Transformer**

[[Abdulkader Helwan](https://abdulkaderhelwan.medium.com/?source=post_page-----798b84c345bd--------------------------------)](https://abdulkaderhelwan.medium.com/?source=post_page-----798b84c345bd--------------------------------)

[Abdulkader Helwan](https://abdulkaderhelwan.medium.com/?source=post_page-----798b84c345bd--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Fd1f2a9be01f6&operation=register&redirect=https%3A%2F%2Fabdulkaderhelwan.medium.com%2Fvideo-classification-using-cnn-and-transformer-798b84c345bd&user=Abdulkader+Helwan&userId=d1f2a9be01f6&source=post_page-d1f2a9be01f6----798b84c345bd---------------------post_header-----------)

6 min read

·

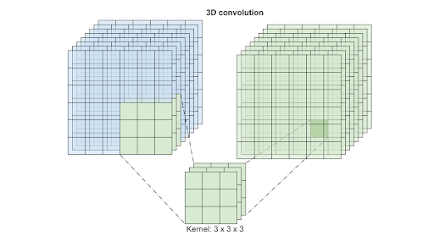
Feb 3

53

1

Video classification is an important task in computer vision, with many applications in areas such as surveillance, autonomous vehicles, and medical diagnostics. Until recently, most methods used 2D convolutional neural networks (CNNs) to classify videos. However, this approach has several limitations, including being unable to capture the temporal relationships between frames and being unable to capture 3D features like motion.

To address these challenges, 3D convolutional neural networks (3D CNNs) have been proposed. 3D CNNs are similar to 2D CNNs but are designed to capture the temporal relationships between video frames by operating on a sequence of frames instead of individual frames. Moreover, 3D CNNs have the ability to learn 3D features from video sequences, such as motion, which are not possible with 2D CNNs.



In this blog post, we will discuss how to classify videos using 3D convolutions in Tensorflow. We will first look at the architecture of 3D CNNs and then discuss how to build a 3D CNN for video classification using Tensorflow. Moreover, we will showcase how to use CNN as a Feature extractor of the frames of videos and use them as inputs for a Transformer that will work as a classification model.

**Classifying Videos Using 3D Convolutions in Tensorflow**

The architecture of 3D CNNs is similar to that of 2D CNNs but with two main differences. First, 3D CNNs use three-dimensional kernels, which allow them to capture temporal relationships between frames in a video. Second, 3D CNNs use three-dimensional feature maps, which allow them to capture 3D features such as motion.

Here is a snippet of the code for creating the 3D-CNN in Tensorflow:

import numpy as np

import h5pyfrom tensorflow.keras.utils import to\_categoricalfrom tensorflow.keras import layersfrom tensorflow.keras.models import Sequentialfrom tensorflow.keras.initializers import Constantfrom tensorflow.keras.optimizers import Adamfrom tensorflow.keras.callbacks import EarlyStopping

**Load the dataset**

,,,,

**Buil the model**

model = Sequential()

model.add(layers.Conv3D(32,(3,3,3),activation='relu',input\_shape=(16,16,16,1),bias\_initializer=Constant(0.01)))model.add(layers.Conv3D(32,(3,3,3),activation='relu',bias\_initializer=Constant(0.01)))model.add(layers.MaxPooling3D((2,2,2)))model.add(layers.Conv3D(64,(3,3,3),activation='relu'))model.add(layers.Conv3D(64,(2,2,2),activation='relu'))model.add(layers.MaxPooling3D((2,2,2)))model.add(layers.Dropout(0.6))model.add(layers.Flatten())model.add(layers.Dense(256,'relu'))model.add(layers.Dropout(0.7))model.add(layers.Dense(128,'relu'))model.add(layers.Dropout(0.5))model.add(layers.Dense(10,'softmax'))model.summary()

**Train the model**

In here, Adam will serve as our optimizer. The loss function for training the model, which is a multiclass classification, will be categorical cross-entropy. For training, the loss metric will be accuracy. As previously discussed, the model will be trained using both the dropout layers and the Earlystopping callback. The Earlystopping callback assists in stopping the training process after a parameter, such as loss or accuracy, does not improve over a predetermined period of epochs, hence preventing the model from becoming overfit. Dropouts force the model to learn rather than memorize, which helps prevent overfitting by randomly shutting off some neurons during training. The dropout rate shouldn’t be excessive because it may cause underfitting.

model.compile(Adam(0.001),'categorical\_crossentropy',['accuracy'])

model.fit(xtrain,ytrain,epochs=200,batch\_size=32,verbose=1,validation\_data=(xtest,ytest),callbacks=[EarlyStopping(patience=15)])Testing the 3D-CNN\_, acc = model.evaluate(xtrain, ytrain)  
print('training accuracy:', str(round(acc\*100, 2))+'%')  
\_, acc = model.evaluate(xtest, ytest)  
print('testing accuracy:', str(round(acc\*100, 2))+'%')

**Video Classification Using Transformer**

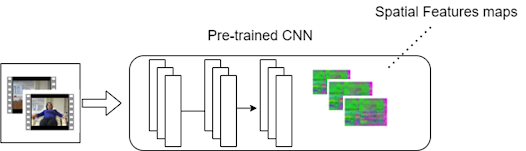
Video classification is the task of understanding videos based on their content. With the rapid growth of video data, it has become increasingly important to develop efficient Computer Vision methods for video classification. In this blog post, we will discuss how to classify videos using the Transformer architecture in TensorFlow.

The Transformer is a neural network architecture that was introduced in the paper “[Attention Is All You Need](https://www.blogger.com/blog/post/edit/6962154748903383574/7230382313564737391)” by Google researchers in 2017. It has since become a popular choice for a wide range of natural languages processing tasks, such as language translation and text summarization. The Transformer architecture is well-suited for video classification because it can handle sequential data, such as video frames, while also capturing global dependencies between the frames.

**Features Extraction Using CNN**

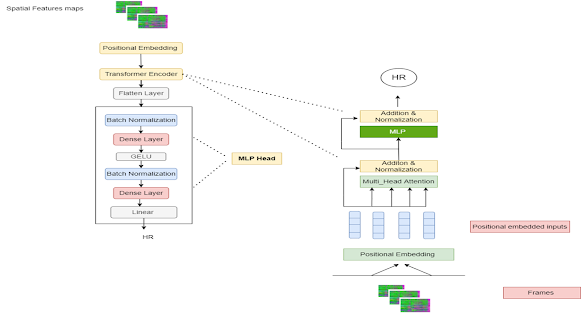
To classify videos using a Transformer in TensorFlow, we first need to extract features from the video frames. This can be done using pre-trained models such as [C3D,](https://www.blogger.com/blog/post/edit/6962154748903383574/7230382313564737391) [I3D](https://www.blogger.com/blog/post/edit/6962154748903383574/7230382313564737391), or [Two-Stream CNN](https://www.blogger.com/blog/post/edit/6962154748903383574/7230382313564737391). These models are trained on large datasets of videos and can extract features such as motion and appearance that are useful for video classification.

Once we have extracted the features from the video frames, we can feed them into the Transformer model. The [Transformer model](https://www.blogger.com/blog/post/edit/6962154748903383574/7230382313564737391) consists of an encoder and a decoder. The encoder takes in the video features and generates a fixed-length representation of the video. The decoder then takes in the encoded representation and generates a label for the video.



The Transformer model can be trained using the TensorFlow library. We first need to define the model architecture, which includes the number of layers, the number of heads, and the dimensions of the model. We then need to define the loss function, such as cross-entropy, and the optimizer, such as Adam. We can then train the model on a dataset of labeled videos and fine-tune it on a smaller dataset of videos.

Once the model is trained, we can use it to classify new videos. We first need to extract features from the video frames and feed them into the encoder. The encoder will then generate an encoded representation of the video, which we can feed into the decoder. The decoder will then generate a label for the video.



**Build The Transformer Model**

The process of video classification is a complex task that involves several steps, including feature extraction, model training, and prediction. Each of these steps requires several lines of code and it would be too complex to include in this format. However, in this post, I will provide you with an outline of the steps that you would need to take to classify videos using the Transformer architecture in TensorFlow.

* Import necessary libraries, such as TensorFlow, NumPy, and any libraries used for feature extraction (e.g. opencv, skvideo, etc)
* Extract features from the video frames using pre-trained models such as C3D, I3D, or Two-Stream CNN
* Define the Transformer model architecture, including the number of layers, the number of heads, and the dimensions of the model
* Define the loss function, such as cross-entropy, and the optimizer, such as Adam
* Train the model on a dataset of labeled videos
* Fine-tune the model on a smaller dataset of videos
* Extract features from new video frames and feed them into the encoder
* Use the encoder to generate an encoded representation of the video
* Feed the encoded representation into the decoder to generate a label for the video

As a first step, it is recommended to start with a basic example of TensorFlow and transformer and then incorporate feature extraction techniques as well as dataset-specific requirements. It is also recommended to check the TensorFlow’s documentation and other resources available online for more detailed information and examples.

Here is an example of how you can create and train a Transformer model in TensorFlow and Keras:

import tensorflow as tf

from tensorflow import keras# Define the number of layers, heads, and dimensions of the modelnum\_layers = 6num\_heads = 8d\_model = 512# Define the encoder and decoder layersencoder\_layers = [ keras.layers.MultiHeadAttention(num\_heads, d\_model), keras.layers.Dropout(0.1), keras.layers.Add(), keras.layers.LayerNormalization()]decoder\_layers = [ keras.layers.MultiHeadAttention(num\_heads, d\_model), keras.layers.Dropout(0.1), keras.layers.Add(), keras.layers.LayerNormalization()]# Create the encoder and decoderencoder = keras.layers.Encoder(encoder\_layers, num\_layers)decoder = keras.layers.Decoder(decoder\_layers, num\_layers)# Define the final modelmodel = keras.models.Transformer(encoder, decoder)# Define the optimizer and loss functionoptimizer = keras.optimizers.Adam()loss\_fn = keras.losses.CategoricalCrossentropy()# Compile the modelmodel.compile(optimizer=optimizer, loss=loss\_fn)# Train the model on your datasetmodel.fit(x\_train, y\_train, epochs=10, batch\_size=64)

Note that this is just a basic example and you will need to adjust it to suit your specific dataset and requirements. You need to pass the feature extracted from video frames as x\_train and the corresponding labels as y\_train.

Here is a small example of how to read video and extract its frames that would be input to the Transformer, using OpenCV:

import cv2

# Opens the Video filecap= cv2.VideoCapture('C:/Work/Videos/Dance.mp4')i=0while(cap.isOpened()): ret, frame = cap.read() if ret == False: break cv2.imwrite('kang'+str(i)+'.jpg',frame) i+=1cap.release()cv2.destroyAllWindows()

**Wrap-up**

In conclusion, video classification is an important task that can be achieved using the Transformer architecture in TensorFlow. By extracting features from video frames using pre-trained models and feeding them into a Transformer-based model, we can effectively classify videos based on their content. With the help of this architecture, we can classify videos with high accuracy and performance.

P.S. This article was originally published on [AI-ContentLab](https://www.ai-contentlab.com/):

<https://www.ai-contentlab.com/2023/01/video-classification-is-important-task.html>