Modelling a Spatial-Motion Deep Learning Framework to Classify Dynamic Patterns of Videos

Sandeli Priyanwada Kasthuri Arachchi *, Timothy K. Shih * and Noorkholis Luthfil Hakim

Department of Computer Science and Information Engineering, National Central University, Taoyuan 32001, Taiwan; koliskol@gmail.com
* Correspondence: sandelik@g.ncu.edu.tw (S.P.K.A), tshih@g.ncu.edu.tw (T.K.S.)

Received: date; Accepted: date; Published: 21 February 2020

Abstract: Video classification is an essential process for analyzing the pervasive semantic information of video content in computer vision. Traditional hand-crafted features are insufficient when classifying complex video information due to the similarity of visual contents with different illumination conditions. Prior studies of video classifications focused on the relationship between the standalone streams themselves. In this paper, by leveraging the effects of deep learning methodologies, we propose a two-stream neural network concept, named state-exchanging long short-term memory (SE-LSTM). With the model of spatial motion state-exchanging, the SE-LSTM can classify dynamic patterns of videos using appearance and motion features. The SE-LSTM extends the general purpose of LSTM by exchanging the information with previous cell states of both appearance and motion stream. We propose a novel two-stream model Dual-CNNSELSTM utilizing the SE-LSTM concept combined with a Convolutional Neural Network, and use various video datasets to validate the proposed architecture. The experimental results demonstrate that the performance of the proposed two-stream Dual-CNNSELSTM architecture significantly outperforms other datasets, achieving accuracies of 81.62%, 79.87%, and 69.86% with hand gestures, fireworks displays, and HMDB51 datasets, respectively. Furthermore, the overall results signify that the proposed model is most suited to static background dynamic patterns classifications.

Keywords: dynamic pattern classification; deep learning; spatiotemporal data; convolution neural network; recurrent neural network

1. Introduction

Dynamic pattern classification of videos serves as an essential step in the process of analyzing video contents [1]. Automatic classification of videos is crucial for monitoring, indexing, and retrieval purposes. Recently, there is growing interest in developing deep learning architectures for dynamic pattern classification. However, to adopt deep learning perfectly is still considered as a unique challenge in computer vision due to the appearance and motion features of video data. A combination of appearance and motion features attempts should produce a better-performing model. Therefore, a proper deep learning model should be able to integrate both static appearances in single frames and temporal relation between consecutive frames in order to achieve excellent classification accuracy. Among the extensive video classification methods in the literature, the majority deal with the same pattern or physical action with different contexts or backgrounds. More specifically, video frames of a particular video consist of the same object, human activity, face, gesture, or scene. Beyond these general classifications, this study intends to classify the displays of artificial fireworks using existing fireworks videos to validate the proposed Dual-CNNSELSTM model in addition to the hand gestures and human actions videos. The proposed two-stream deep learning architecture interacts...
through the long-term temporal feature extraction phase to input the information additionally learned from the consecutive time steps.

Currently, both single and two-stream neural net architectures achieve remarkable success on video pattern classification and prediction applications. In practice, the convolutional neural networks (CNN) are very successful at image analysis tasks [2,3]. Because of the high complexity of video data, most video classification based on deep learning models [4–6] demonstrated the same or similar performance to handcrafted features [7]. Apart from the static appearance features of images, video data additionally contain temporal motion features. Some recent studies [8,9] used recurrent neural networks (RNN) to model long-term temporal information, and thus achieved acceptable performance. Motivated by such models, currently, researchers adopt the long short-term memory (LSTM) model to solve the problem of long and complicated sequences of dynamic gesture problems. Moreover, LSTM has become an essential part of deep learning models for image sequence modeling [10,11].

Furthermore, enhanced methods were proven to recognize human actions and gestures [12–14]. Practically, spatial and temporal features are needed to classify video data. For human action recognition, previous studies always considered similar ideas to those of image recognition. Unlike static images, dynamic patterns of videos consist of ever-changing motions with different target objects that have various appearances against different backgrounds. Thus, it is crucial to explore distinct spatiotemporal features to classify such dynamic patterns. Moreover, the work of Molchanov et al. [15] proposed a combination of 3D CNN and RNN, with the fully connected spatiotemporal features transferred into RNN. Inspired by this study, [16] proposed a combination of a 3D CNN and LSTM model to classify types of hand gestures.

However, prior studies did not address well the spatial relationship between consecutive time steps. Although some previous works based on video classifications presented good models [17,18], we doubt whether they could satisfy the specific requirements, specifically for fireworks videos. In particular, even with the same firework type, the sample videos involve different object patterns in most time steps, as compared to previous models of other video classification tasks, such as human actions, scenes, objects, and faces. Therefore, to overcome the drawbacks of traditional deep learning methods, first, it is necessary to select a suitable neural network method to design an accurate two-stream architecture, which can utilize the spatial and motion information. Here, such a design should consider the spatiotemporal relationship to reach a higher performance than that of the existing two-stream models [17,18].

In this study, as our first contribution, we introduce a novel concept state-exchanging long short-term memory (SE-LSTM). The proposed SE-LSTM concept considers the advantages of maintaining the spatial–motion relationship and Section 3.1 presents several functional parts of this concept in detail. It thus modifies the general internal structure of the LSTM unit. Explicitly, the cell state contains information learned from the previous time steps of a particular stream. However, the proposed SE-LSTM is designed by adding some additional information on cell states previously learned from both streams to the consecutive time steps of each stream. During this process, we encode previously learned information in memory cells, which are regulated with nonlinear gates to identify temporal dependencies. Hereinafter, this spatiotemporal information-sharing process is known as state-exchanging. Utilizing this SE-LSTM, as a second contribution, our study proposes a promising design: a deep neural network based on a two-stream architecture, named Dual-CNNSELSTM.

The proposed Dual-CNNSELSTM architecture applies the SE-LSTM to overcome the limitations of CNN and RNN and integrates short-term motion, spatial, and long-term temporal to classify the dynamic patterns of videos. Two CNN streams were trained to model the spatial information of RGB videos and short-term motion features of stacked optical flows, and the dense optical flow method was used to create optical flow videos for the motion stream. The two LSTM networks followed by CNNs are employed to model the long-term temporal information, over the frame-level spatial and temporal information extracted by the top CNN phases. Afterwards, the proposed SE-LSTM concept interacts with this extracted information handling the cell states, which keeps the learned history of
both streams’ previous time step. Then we averaged the extracted cell information to be an input to recent time steps as additional information. Finally, we concatenate both spatial and motion features to evaluate the class probabilities to classify the dynamic patterns of input data. Since our study will classify the dynamic patterns of videos, this study considers fireworks [19] hand gestures [16], and human action videos from the HMDB51 dataset [20] to verify our proposed model. Furthermore, to demonstrate the significance of our proposed architecture, we conduct the performance examination under three datasets and standard HMDB51 benchmarking, finding that the performance boosts dramatically with the use of several state-of-the-art methods. Based on experimental results with different benchmarks, it is demonstrated that the interaction of the spatial and motion streams within the training phase could achieve a better performance than that of the previously discussed standalone two-stream networks.

The rest of the paper is organized as follows. Sections 2 and 3 discuss the related work and methods. Next, datasets and experimental evaluations are discussed in Sections 4 and 5, respectively. Finally, Section 6 presents the conclusion and future work.

2. Related Works

Video classification is a fundamental challenge in multimedia and computer vision communities. Successful state-of-the-art video classification systems rely heavily on multiple discriminative feature representations extracted from videos and, hence, most works have focused on designing robust features [9]. In order to achieve better performance, many existing video classification problems have been considered in the light of advances in the image domain. For instance, two studies [21,22] extended 2D Harris corner detector and traditional SIFT features into 3D space to obtain the space–time interest points (STIP). The dense trajectory feature-based method reported that densely sample local patches at regular positions in time and space domains outperform the detected STIP points of video classification problems [23]. Furthermore, instead of working with more complex models, the work [24] focused on low-level features with their encoding, using the Fisher vector as an alternative to bag-of-words histograms to aggregate a small set of state-of-the-art, low-level descriptors, in combination with linear classifiers. In addition, several previous studies utilized different fusion techniques to combine the different information features.

Most of the abovementioned hand-crafted features have achieved good performance using both appearance and motion information on video classifications; the ability to improve such features may be limited. With the promising results of DL models on image recognition tasks [2,6,25], several extensions to process videos have been recently proposed. Among them, one study has extended CNN models into the spatial–temporal domain by training on stacked video frames [4]. Also, one work [26] has compared multiple neural network architectures for action recognition, while [27] proposed a method to learn and compute generic spatial–temporal features more efficiently. Another study [6] proposed an interesting two-stream approach using two CNN networks, which were trained to capture the spatial and motion information using frames and stacked optical flows as input. Its final predictions were obtained by linearly averaging the prediction scores. Another two-stream approach based on traditional dense trajectories has recently reported reliable results [28].

Many kinds of research have been conducted to examine the temporal dynamics of videos. Among them, [29] introduced a hidden Markov model to capture video state changes, considering variable durations, while [30] combined feature patterns with parts in a maximum margin hidden conditional random field framework. Moreover, one study [31] proposed a method to train a linear ranking machine with video frames and then the output parameters used to obtain video-level representations.

More recently, RNN has been used in many sequential modeling problems, such as image and video analysis [32–34], as well as speech recognition [35]. Naturally, videos can be decomposed into spatial and temporal components. Motivated by this factor, [9] proposed a two-stream approach, which breaks down the learning of video representation into separate feature learning of spatial and temporal clues. Apart from that, the study of [36] trained two-stream LSTM networks for action recognition. This study also tried to combine CNN models with LSTMs, but did not obtain a
remarkable performance when training only with the LSTM model. However, the work discussed in [9] concatenates the outputs of standalone LSTM models and CNN models using spatiotemporal clues. This video classification model has observed that CNNs and LSTMs are complementary to state-of-the-art methods. In addition, the Long-term Recurrent Convolutional Networks (LRCN) model is capable of extracting spatial and temporal features. Moreover, 3DCNN, which extracts spatiotemporal features, is superior in such tasks, since it uses the strong point of CNN to classify images and combine them with temporal features.

However, it has a limited ability to learn the long-term temporal information essential for hand gesture recognition. Among the various types of gestures, hand gestures are the most expressive way of interacting more naturally when communicating with computers. Therefore, recently, hand gesture recognition has motivated new technologies in the area of computer vision. Previous studies have been proposed to solve hand gesture recognition tasks such as the glove-based approach [37], device-related methods [38] and the data glove-based method to address the issue of external sensors that enable us to monitor a user’s hand motions more frequently [39]. Apart from such methods, nowadays, deep learning-based models are utilized to solve the hand gesture recognition and classification problems more efficiently and accurately [40,41]. Furthermore, they have been successfully implemented in CNN-based hand gesture recognition problems, too [42,43]. In addition, LSTM has become an important part of deep learning models for human action recognition [10,11]. Enhanced methods, such as Bidirectional RNN [12], hierarchical RNN [13], and Differential RNN (D-RNN) [14] were proven to recognize gestures. The work of Molchanov et al. [15] proposed a combination of 3DCNN and RNN, with fully connected spatiotemporal features transferred into RNN. Even though 3DCNN can model fine motion, it cannot support long variation because this depends on the size of the filter. A practical approach is to model a short temporal snapshot of videos by averaging the prediction from a single frame and a stack of several optical flow frames after passing two replicas of 2DCNN. In this study, we also adopt a two-stream network combining CNN layers on top of LSTM using optical flow and RGB videos as input.

LSTM layers consist of blocks, which, in turn, consist of cells. Each unit or cell within the layer has an internal cell state and outputs a hidden state. However, the cell state is the key to the LSTMs unit. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Each cell has its inputs, outputs, and memory. Cells that belong to the same block share input, output, and forget gates. However, each cell can hold a different value in its memory. Nevertheless, the memory within the block is written to, read from, and erased all at once. This denotes that the information inside a cell state is a necessary thing among the sequential classifications. The blocks sharing the layers that they receive inputs from and feed their output to, make up a layer.

3. Methods

The video data considered for this study consisted of two components, spatial and temporal. The spatial information, in the form of individual frames, tells us about the objects in the video. The temporal information, in the form of motion through the frames, gives the motion features of the objects. Therefore, when we address the video classification problems, considering both features will be an advantage. Then the recognition strategy could be divided into two streams: one for the spatial component and one for the temporal component. In this section, we present the architecture of the baseline model and the proposed model for classifying the dynamic patterns of the videos.

3.1. State-Exchanging Long Short-Term Memory (SE-LSTM)

As discussed in Section 2, LSTMs are excellent at processing sequential data. Unlike other RNNs, LSTMs consist of internal gates, which can adjust the flow of information. These gates can help manage which data in a sequence are saved or discarded. Because of these advantages, LSTM is special among training models and successfully uses backpropagation to solve the vanishing gradient problem. Moreover, each cell inside the layer contains a cell state, which controls and stores previous information.
In handling sequential data to make predictions, especially image data, the LSTM cell state is doing a vital job. Therefore, during this study, we focused on updating and resetting cell state information outside of the standard cell state update process. The first step of this process extracts the previous time step’s $(C_{t-1})$ cell state from both streams’ LSTM units. Then we averaged the extracted cell state information and updated the cell state of the next time step $(C_t)$. Figure 1 gives the general idea of our proposed state-exchanging process.

Since no such connection physically exists between cell states, we had to implement the proposed model; we define this process as “State-exchanging.” Only within two-stream networks, we can exchange information between the cell states of both LSTM streams during their adjacent time steps. This process was done by extracting the LSTM’s cell state of both streams and averaging and updating the cell states of the next time step. One of our major contributions, the state-exchanging process, is discussed in the following section.

**Figure 1.** Proposed two-stream, RGB and optical flow-based state-exchanging in Long Short-Term Memory (SE-LSTM).

### 3.1.1. SE-LSTM Process

The idea of SE-LSTM is to extend the standard LSTM model, which updates the current time step’s cell state, as formulated in Equation (1). Theoretically, the cell state contains information learned from the previous time steps. Beyond this general process, before we updated the new cell state $c_t$ with the previous cell state $c_{t-1}$, we extracted the previous time step’s cell state of both streams, taking advantages of Keras API [44], which allowed us to access LSTM states within the layer. Then, as explained in Section 3.1.2, we did element-wise averaging of both state values and modified and updated the $c_t$ using this additional value. That averaged value was called the State-exchanging Package, denoted by $P_t$. Finally, we have to decide what we are going to output as the next cell state. However, this output will be a filtered version based on our cell state. In this step a first sigmoid layer runs and decides what information is needed to be output; then it runs the cell state through $\tanh$ and multiplies it by the sigmoid gate’s output information. So, during this process, the layer only outputs the information we decided to.

Equations (1)–(6) express the operations performed by LSTM unit, where, $i_t$, $f_t$, $o_t$, $z_t$ represent the input gate, forget gate, output gate, and cell gate, respectively. $c_t$ and $h_t$ are memory and output activation at time $t$. Equations (2), (3), (5) and (6) are the formulas for the forget gate, cell gate, output gate, and hidden state, respectively, of a standalone stream. The modified cell state of the proposed method, $P_t$, is given by Equation (11).

$$i_t = \sigma(x_t w_{xi} + h_{t-1} w_{hi} + c_{t-1} w_{ci} + w_{bi}),$$  \hspace{1cm} (1)
\[ f_t = \sigma(x_t w_{xf} + h_{t-1} w_{hf} + c_{t-1} w_{cf} + w_{f/bais}), \]  
\[ z_t = \tanh(x_t w_{xz} + h_{t-1} w_{hz} + w_{z/bais}), \]  
\[ c_t = z_t \otimes i_t + c_{t-1} \otimes f_t, \]  
\[ o_t = \sigma(x_t w_{xo} + h_{t-1} w_{ho} + c_{t-1} w_{co} + w_{o/bais}), \]  
\[ h_t = o_t \otimes \tanh(c_t), \]

3.1.2. State-Exchanging Package

The LSTM memory cell usually refers to one of the two shares of an LSTM layer’s hidden state [45]. This portion of the hidden state has the ability to modify by addition, subtraction, and scaling. Therefore, it tends to preserve information for a relatively long time. In this study, we consider the cell as a variable, not as a function, and this variable takes a piece of additional information at each time step rather than information learned from the previous time steps. Assuming the advantage of updating the cell state with additional learned information, we exchange the extracted information within each time step of both RGB and optical flow streams by calculating the average value as in Equation (7) and sharing it between the next time steps of both streams. The resulting information from this cell information exchanging process we define as the “State-exchanging Package” \( P_t \), which is represented in Equation (8).

\[ P_t = \left( \text{Cell}_{rgb} + \text{Cell}_{opt} / 2 \right), \]  
\[ P_t = (C_{t-1} + C_{t-1}')/2, \]  
where \( C_{t-1} \) and \( C'_{t-1} \) are the previous cell states of RGB and optical flow streams, respectively.

For the RGB stream, the modified cell state using proposed SE-LSTM with the \( P_t \) is formulated using Equations (9)–(11). Here \( C_t, Z_t, I_t \), and \( F_t \) are the cell state, cell gate, input gate, and forget gate of the modified LSTM, respectively. According to the SE-LSTM process, the previous cell state \( c_{t-1} \) in Equation (9) needed to be updated with the averaged newly learned information of both streams’ cell states \( (c_{t-1} + c'_{t-1})/2 \). Therefore, the modified cell state can be formulated as in Equation (10). Based on Equation (8), the new cell state \( (C_t) \), represented in Equation (10), can be replaced with \( P_t \) as in Equation (11). Then we updated the general objective function \( (H_t) \) in Equation (12) using \( P_t \) as in Equation (13); this is one of our main contributions in this study, i.e., modifying the objective function of general LSTM unit using our proposed SE-LSTM process. By the same token, we defined and modified the cell state and the objective function of the optical flow stream with \( P_t \). The state-exchanging internal structure of both LSTM sequences are shown in Figure 2.

\[ c_t = z_t \otimes i_t + c_{t-1} \otimes f_t, \]  
\[ C_t = Z_t \otimes I_t + [c_{t-1} + (c_{t-1} + c'_{t-1})/2] \otimes F_t \]  
\[ C_t = Z_t \otimes I_t + [c_{t-1} + P_t] \otimes F_t \]  
\[ H_t = O_t \otimes \tanh(C_t), \]  
\[ H_t = O_t \otimes \tanh(Z_t \otimes I_t + [c_{t-1} + P_t] \otimes F_t), \]  
\[ H_t = O_t \otimes \tanh(Z_t \otimes I_t + [c_{t-1} + P_t] \otimes F_t), \]
Figure 2. Process of state-exchanging package ($P_t$) working with the sequence of LSTMs. First, get both previous cell state values $C_{t-1}$ and $C_{t-1}$ of RGB and optical flow stream, average the value as in Equation (14) and calculate the value of the state-exchanging $P_t$. Then, share this newly learned information between both streams’ next time step. Finally, update the next time step’s previous cell state via $[C_{t-1} + P_t]$. Likewise, update and reset previous cell states until the last time step.

3.2. Dual-CNNSELSTM Model

Based on some studies, standard LSTM is challenging to use directly on sequences where the input is spatial. Therefore, to perform tasks that need sequences of images to predict relevant classes, we need a more sophisticated model. Hence, the combination of CNN and LSTM network is an architecture designed explicitly for dynamic pattern classification problems with spatial inputs, like a sequence of images or videos. Considering the advantage of adding CNN, we propose a novel two-stream model Dual-CNNSELSTM, utilizing the SE-LSTM concept combined with CNN to improve the dynamic pattern classification task. When considering models with CNN layers on top of LSTMs, we have achieved better accuracy than models with only with a sequence of LSTMs [46]. Because of the ability to find patterns in images through looking for existing patterns, a CNN is good at identifying objects in images. Also, Max Pooling layers only select and keep significant features and ignore the rest. This means it can classify small and big objects into the same class accurately and helps to overcome the feature extraction issues while directly inputting to the LSTM. Therefore, CNN helps the learning models to attain better performance than LSTM models.

The Dual-CNNSELSTM architecture consists of five CNN layers followed by an LSTM layer, including a dropout layer. A fully connected layer follows a softmax layer, which computes the classification probabilities. Within the CNN layers, we reduced the input size at lower layers since it helps to decrease the computational cost. Therefore, for low layers such as layer 1 and layer 2, we wanted to cut the size of the input in half for each layer. To do this, we set the stride size to 2 when the input is passing through the CNN layer. The kernel size of the first CNN layer is 5 × 5 and for the other four layers it is 3 × 3. To avoid high information loss, our input feature planes were set to 32 in the first layer. The feature maps consisted of four filter sizes such as 32, 64, 128, and 256, and we doubled them up within each CNN layer to increase the depth. Each CNN layer was followed by a batch normalization layer, a ReLU activation layer, and a MaxPooling layer with a pool size of 2 × 2. The Batch Normalization layer addressed the vanishing and exploding gradients problem and made the training faster; it also removed the internal co variate shift problem. Spatial and motion features were extracted from the five layers of CNN architecture and fed into the SeqLSTM, which has 256 hidden layers.

3.3. Dual-3DCNNLSTM Model

For video representation, it is easy to extend 2D CNNs by using them to extract features from each frame, then pooling their predictions through the whole video. This direct application is convenient but ignores the temporal connections among frames. To overcome this issue, some researchers add a recurrent layer such as LSTM to encode the state and capture the temporal ordering
of frames [32]. However, since 2DCNN ignores the temporal information, it seems it is not a natural representation. Because of this drawback, the 3DCNN concept, which can capture both spatial and temporal features using 2D spatial dimensions and one temporal dimension, is proposed [27]. Even though 2D CNN, together with LSTM, can extract high variation level features, it is unable to extract fine low-level features. Furthermore, 3DCNN can extract fine motion features, but it does not support large variations since it depends on the filter [47]. Therefore, together with LSTM, a 3DCNN model can give better classification results for the dynamic patterns of the videos. Moreover, the practical way to model short temporal video data is by fusion of the appearance features of a single frame and the stack of optical flow after passing through the two streams.

Considering the advantages of combining CNN and LSTM networks, our baseline two-stream architecture Dual-3DCNNLSTM, which was also used to classify hand gestures in our IC4You project, consists of a 3DCNN network followed by a stack LSTM layer [16]. The filter size of each Conv3D layer is 3 × 3 × 3, and the stride and padding are 1 × 1 × 1. The feature maps consist of four filter sizes doubled within each layer (32, 64, 64, and 128), and each number increases the depth. We applied batch normalization, which allows much higher learning rates and is less reliant on initialization to accelerate the model training. Each 3DCNN layer is followed by a batch normalization layer, a ReLU layer, and a 3D max pooling layer with a pooling size of 3 × 3 × 3. Features are extracted from the 3DCNN architecture and then fed into the one stack of LSTM layer with 256 hidden states. Also, we have added several dropout layers in every section with a value of 0.5 and then computed the output probability result, using the softmax layer in the end. Using both optical flow and RGB as the input data might produce a better result than using only one stream input. Thus, the two-stream model of the 3DCNN + LSTM is considered as a baseline.

4. Datasets

Dataset size is one of the key factors that deep neural network models depend on. To evaluate the performance of the proposed model, we used three datasets. Among them, fireworks and hand gestures datasets are our own datasets and HMDB51 is a very popular dataset for human action recognition in the current research area [20]. While doing the experiments, we randomly divided the dataset into three sections: training (70%), validation (15%), and testing (15%). Sample frames of the three datasets are visualized in Figure 3. In this section, we describe the fireworks, hand gestures, and HMDB51 datasets that we used to validate our proposed Dual-CNNSELSTM model.

4.1. Fireworks Dataset

We downloaded different types of fireworks videos from the web and, to decrease the computational cost, split them into less than 6-s length clips. Our input video size is 240 × 320, and we applied a few augmentation techniques such as flip-up, flip-down, and rotate to increase the size of the dataset. However, fireworks videos have been selected under three conditions to maintain the fairness of our dataset and control the scope of our study as: each video should have one firework type, videos should not contain any other moving objects except fireworks, and the background should be static with invisible objects. The dataset was manually categorized into eight classes, Chrysanthemum, Crossette, Desi, Dot, Drop, Fish, Palm, and WaterFlower, and our current dataset consists of 1500 video clips. Sample videos of the fireworks dataset can be seen on our project web page [46].

4.2. Hand Gestures Dataset

A simple data collection tool is implemented to speed up the collecting process for data consisting of sequences of gestures performed by real human actors. We used the Real Sense SR300 depth camera as the primary tool for recording data since we wanted to input the modality data to our proposed model. The modality data that we recorded are the RGB and depth data. Since we consider the hands as the main part of the gesture, half the body of a user, including the face and hands, was recorded. Later, during the preprocessing step, we extracted the hand from the body to
input to the model. Gesture videos were recorded from 20 individuals using 11 dynamic gestures, click, grab, scroll-down, scroll-up, scroll-right, scroll-left, pinch, zoom out, zoom in, backward, and forward, that have been sampled and clearly visualized on the research web page [16]. The user needs to re-perform each gesture six times in a different manner. Each recorded sequence consists of a 3-s gesture with 120 frames. In total there are 2162 sequences (or videos) of gestures in RGB and depth format recorded under five different environments with lighting conditions and variable distances to the camera.

![Sample video frames of datasets. (a) Fireworks dataset; (b) Hand gestures dataset; (c) HMDB51 dataset.](image)

**Figure 3.** Sample video frames of datasets. (a) Fireworks dataset; (b) Hand gestures dataset; (c) HMDB51 dataset.

4.3. HMDB51 Dataset

HMDB51 is a popular action recognition benchmark containing 6766 videos divided into 51 categories, with five types of actions: general facial actions, facial actions, general body movements, body movements with object interaction, and body movements for human interaction [20]. The sample videos have a 320 × 240 spatial resolution and 30 fps frame rate. Most of the actions in this dataset have many viewpoints and camera motions. The experiments are conducted on the first split of these two datasets.

4.4. Optical Flow Datasets

Optical flow is a crucial component of video classification approaches because it represents the pattern of the apparent trajectory of image objects between two consecutive frames caused by the movement of an object in a visual scene. Since our models process one video frame per second, they do not use any apparent motion information. However, of the several methods used to compute optical flow, we used the dense optical flow based on Gunner Farneback’s algorithm [48].

Therefore, we trained both RGB and optical flow frames separately as two streams and perform fusion methods to finalize the output classes. To create our optical flow dataset, we used the dense optical flow method, which computes the optical flow for all the pixels in the frames. Figure 4a shows our sample RGB frames, while Figure 4b shows the relevant optical flow results after the color code for better visualization.
Figure 4. Sample RGB and optical flow frames. We colored the optical flow results with Hue value for better visualization. (a) RGB frames of three datasets; (b) Optical flow frames of relevant RGB frames.

5. Experimental Evaluation

5.1. Experimental Setup

To implement the proposed two-stream model, we used Keras [44], a high-level deep learning framework, which was built on top of TensorFlow. For building neural network models, Keras provides a scikit-learn type API, written in Python. Without worrying about the mathematical aspects of tensor algebra, optimization methods, and numerical techniques, researchers can easily use Keras to develop neural networks. Apart from that, in the area of deep learning, TensorFlow has gained much more momentum than other competitors: Theano, Torch, Caffe, and other well-known frameworks. TensorFlow can help to implement a multilayer model that is broadly used in voice recognition, image recognition, and text-based applications like Google Translate, and video classification. Furthermore, we executed our models with a Ge-Force GTX 1080-GPU machine with 32 GB RAM. Both two-stream networks were trained using Adam [49] optimizer with a momentum fixed to 0.9. To fine-tune the model, we gradually decreased the learning rate from $10^{-2}$ to $10^{-4}$ after 50 epochs and then to $10^{-5}$ after 100 epochs. Also, dropout is applied to the visible and hidden layers with a ratio of 0.5 and 0.25, respectively, to overcome the problem of overfitting. To train two-stream networks, we used the adaptive gradient descent algorithm Adam to update to the RMSProp [50] optimizer, which is like RMSprop with momentum as well. To evaluate the performance, we trained the networks by scheduling the learning rate rather than the adaptive learning rate.

To determine whether the model has converged or generalized, we analyzed graphs of its performance during the training with two common performance metrics of loss and accuracy. The loss metric gives a numerical estimate of how far the model is from producing the expected answers. The accuracy gives the percentage of the time that it chooses the correct prediction. Therefore, during this study, we used these similar evaluation methods to investigate the significance of the proposed method and compare the performance of previous works.

5.2. Evaluation of Fireworks Dataset

To the best of our knowledge, there is no other work that introduces the classification of firework types, so to monitor the robustness of our proposed model while validating the firework dataset, we conducted several control experiments using baseline models. As in our first work [19], we evaluated the performance of the fireworks dataset with different single-stream models. Furthermore, as in our project website [46], we implemented several two-stream baseline architectures using RGB and optical flow data and compared the results with our single-stream models. We noticed that our two-
stream architectures outperformed the single-stream architectures. In practice, to classify dynamic patterns, it is required that we consider spatial and motion aspects separately as well as evaluating their relationship. However, prior studies did not address well the spatial-motion relationship between consecutive time steps.

Inspired by this point, we implemented a deep neural network based on a two-stream architecture, using the SE-LSTM, to overcome the limitations of CNN and RNN. To the best of our knowledge, no previous studies assess the spatial–motion features while training to classify dynamic patterns. Performances comparisons of both single and two-stream models combining with proposed SE-LSTM concept is figuring out in Figure 5. Based on the results, it is demonstrated that the Dual-CNNSELSTM model achieved 81.76% accuracy while validating the fireworks dataset and, thus, we employ this result to make the dynamic patterns classification comparison in this work.

![Figure 5. Performance (%) comparison of single-stream network and two-stream network architectures with the proposed Dual-CNNSELSTM model.](image)

5.3. Performance of Dual-CNNSELSTM with Dual-3DCNNLSTM Model

In study, we focused on evaluating the performances of the baseline Dual-3DCNNLSTM model and the proposed Dual-CNNSELSTM model using the three experimental datasets discussed in Section 4. While training the models, we noticed the computational times for both models with three training datasets. Our proposed Dual-CNNSELSTM architecture took 4.8 h, 5.3 h, and 18.7 h to learn the parameters for fireworks, hand gestures, and HMDB51, respectively, while the Dual-3DCNNLSTM learned similar datasets within 4.2 h, 4.6 h, and 16.5 h, respectively. Both models’ average testing accuracies with the three datasets are illustrated in Table 1. The results demonstrated that the Dual-3DCNNLSTM model had the best classification accuracy of 77.31% with the fireworks dataset. However, when we trained the Dual-CNNSELSTM model, the fireworks dataset had the second-highest performance at 79.87%, but that was higher than with Dual-3DCNNLSTM. Apart from that, our Dual-CNNSELSTM model had the best classification results with all three datasets compared to the baseline model.
Table 1. Testing the accuracies of both models with the three experimental datasets.

| Model                  | Fireworks | Hand Gestures | HMDB51    |
|------------------------|-----------|---------------|-----------|
| Dual-3DCNNLSTM         | 77.31%    | 73.12%        | 67.84%    |
| Dual-CNNSELSTM         | 79.87%    | 81.62%        | 69.86%    |

The hand gesture dataset with 11 dynamic classes achieved a significant testing accuracy of 81.62% with the proposed Dual-CNNSELSTM model. The accuracy and loss of the model during training is shown in Figure 6. The model reached training and validation accuracies of 83.81% and 74.09%, while the loss was down to 0.38% and 0.59% for training and validation, respectively. For better understanding, the training accuracies of all three experimental datasets are presented in Figure 7. The time it takes to generalize the model also differs in different datasets with different classes and samples. Thus, the number of epochs may vary from dataset to dataset. Since the HMDB51 dataset generalizes around the 100th epoch to maintain the fairness of the experimental results, we limit the number of epochs in Figure 7 to 105. Moreover, we notice that the HMDB51 dataset obtained a considerably low accuracy with the two tested models. We believe one reason is the low quality of the videos compared with the other two datasets. In addition, both the fireworks and hand gesture datasets’ videos have static backgrounds rather than the dynamic backgrounds of HMDB51’s videos. Moreover, the dissimilar behavior in Figures 6 and 7 may be because of the different number of classes and different dataset sizes. Our fireworks dataset consists of eight classes with 1500 sample videos, while the hand gesture dataset contains 11 classes with 2100 videos. However, HMDB51 has 51 action classes with 6766 samples. When a dataset has a higher number of classes and samples, in some cases, it is hard to learn the model. Hence, it achieves a lower accuracy than the datasets with a small number of classes.

Furthermore, the firework and hand gesture datasets have static backgrounds and no other moving objects except the target objects. Hence, in most of the cases, the objects are clearly visualized, and CNN can extract the features well. However, the videos of HMDB51 contain many variations such as human actions, dynamic and complicated backgrounds, as well as a lot of unnecessary data or noise. These drawbacks make it difficult for the model to learn and extract features and may cause the lower accuracy in the HMDB51 dataset. Moreover, in this stage, it is hard for either the fireworks or the hand gesture dataset to achieve higher performance than this due to some similar movements, objects, and shapes within several classes. Consequently, the results imply that the proposed model is most appropriate with static background dynamic patterns classifications.

![Figure 6](image_url)

**Figure 6.** Model performances (%) of the proposed Dual-CNNSELSTM model with the hand gesture dataset. (a) Training and validation accuracy; (b) training and validation loss.
5.4. Evaluation of HMDB51 Dataset.

The fireworks and hand gestures datasets are our own datasets, while HMDB51 is an accessible benchmark dataset. Therefore, it is necessary to compare it with recent similar DL-based works in order to illustrate the robustness of our models. Since HMDB51 is a popular benchmark, this dataset provides three training–testing splits with 70% per category for training and 30% per category for testing. Hence, we also used the same ratio for all the experiments. Comparisons of the average accuracies of our experimental models and the state-of-the-art methods are given in Table 2. Both the Dual-3DCNNLSTM and Dual-CNNSELSTM architectures’ layer-wise weights and neurons are given in Appendix A. In addition, all of the compared architectures in Table 2 are summarized in Appendix A5. Compared to the original two-stream architecture [6], our proposed Dual-CNNSELSTM model and the baseline model improve by 10.46% and 8.44% with fusion by the SVM method respectively. Even though this two-stream network is able to do multi-task learning and SVM fusion, the novelty of our models is higher. Another interesting comparison is with Two-stream 3DCNN [48], which exploits both RGB and optical flow independently and then concatenates them with an early fusion method. This work achieved an average accuracy of 62.54%, while our baseline model, which also used 3DCNN on top of the LSTM layer, achieved 67.84%. Moreover, compared with the Two-stream Fusion ConvNet architecture [51], which used VGG16 for both spatial and temporal streams, our model achieved an accuracy increase of 4.46%—from 65.4% to 69.86%. Furthermore, the baseline model achieved a testing accuracy increase of 2.44%, from 65.45% to 67.84%.

Table 2. Results comparison of HMDB51 dataset with our models and the state-of-the-art methods.

| Method                                      | Accuracy   |
|---------------------------------------------|------------|
| Two-stream model (fusion by averaging) [6]  | 58.0%      |
| Two-stream SVM [6]                          | 59.4%      |
| Two-stream 3DCNN [48]                       | 62.54%     |
| Two-stream Fusion [51]                      | 65.4%      |
| Two-Stream Fusion + IDT [51]                | 69.2%      |
| ST-ResNet [52]                              | 66.41%     |
| ST-ResNet + IDT [52]                        | 70.03%     |
| Dual-3DCNNLSTM (ours)                       | 67.84%     |
| Dual-CNNSELSTM (ours)                       | 69.86%     |
Moreover, though ST-ResNet [52], a novel implementation that generalizes ResNets for the spatiotemporal domain by introducing residual connections, achieved slightly higher accuracy than Two-stream Fusion [51], it could not outperform our baseline and proposed model. However, both Two-stream Fusion [51] and ST-ResNet [52], when further tested to see what benefit results from a late fusion of hand-crafted IDT features, achieved higher accuracies of 69.62% and 70.03%, respectively. Overall, among all of the state-of-the-art models compared, except for the Two-Stream Fusion + IDT [51] model’s slightly increasing gap of 0.17%, our proposed Dual-CNNSELSTM model clearly shows a significant accuracy of 69.86% with the HMDB51 dataset.

Since the fireworks and hand gestures datasets are balanced datasets, all the classes consist of an equal number of videos. However, the HMDB51 dataset is imbalanced in that its classes include a different number of sample videos. Therefore, for further clarification, we computed the F1-score of the HMDB51 dataset using class probability scores in the softmax layer of the networks for each class. We used this measurement to find the most beneficial classes. Figure 8 shows the best 10 classes’ F1-score of the proposed Dual-CNNSELSTM model with identical class scores of the Dual-3DCNNLSTM model. When we compared the top 10 beneficial classes of both models, we found that only the eat, laugh and stand classes had slightly higher scores with the Dual-3DCNNLSTM model.

![Figure 8. Class-wise accuracy among the best 10 classes of HMDB51 dataset.](image)

6. Discussion

Our work has presented a novel two-stream architecture called State-exchanging LSTM (SE-LSTM), which is an extension of general LSTM to classify the dynamic patterns in videos more efficiently. To the best of our knowledge, no previous studies could distinguish the spatial and motion feature exchanging within long-term temporal phase while training. For this work, we selected videos under three conditions to maintain the fairness of our fireworks and hand gesture dataset and control the scope of our study. Each video should have one fireworks and gesture type, videos should not contain any other moving objects except the target objects, and the background should be static with invisible objects. Therefore, our model is not suitable for classifying multiple targeting objects—for instance, more than one type of firework or hand gesture within one video. The proposed model is the best fit with the video data, which have static backgrounds. Sample fireworks video types, which are not included in our dataset, are shown in Figure 9.

Although previous works based on video classifications, discussed in Section 2, presented several good models, these prior studies did not address well the spatiotemporal relationship between consecutive time steps. However, our proposed model integrates short-term motion, spatial, and long-term temporal to classify the dynamic patterns. Moreover, the proposed Dual-
CNNSELSTM architecture interact with the spatial and motion streams. The way that the cell states, which store the learned history of both streams’ previous time steps, were extracted and averaged to be an input to the next time steps as additional information is a novel contribution of this work. Consequently, based on experimental results with different benchmarks, it is demonstrated that interaction with the spatial and motion streams within the training phase could more significantly improve performance than standalone two-stream networks.

Figure 9. Sample fireworks videos that did not satisfy our data selection conditions. (a) Videos containing multiple types of fireworks; (b) moving objects like ships and airplanes; (c) visible and static objects in the backgrounds.

However, one crucial factor in this kind of classification problem is the mutual exclusiveness of the classes. For better classification results, it is necessary to have independent features among the classes but similar features inside a class. As we know, humans are error-prone. Thus, even though we categorize carefully while creating our datasets to maintain the uniqueness of each class, we had some mislabeled classification results, as shown in Figure 10b, because of the relatively similar types of classes.

Figure 10. Instances of misclassified samples from (a) the hand gestures dataset; (b) the fireworks dataset; and (c) the HMDB51 dataset.

In the fireworks dataset, some samples of the Crossette class were wrongly classified as WaterFlower class and some videos of WaterFlower class were misclassified as Palm class. Moreover, some samples were recognized as Dasi class even though they were originally in the Drop class. Apart from that, in the hand gesture dataset, there are also some mislabeled data. Some videos of the up and down and zoomIn and zoomOut classes are wrongly classified since these videos have late starting positions and early ending positions in the gesture performance. Not only the fireworks and hand gesture datasets but also the HMDB51 dataset had some mislabeled classes. We think this is because some classes consist of videos with a similar context or person present as in Figure 9c. For instance,
some videos of sword Exercise and draw sword classes are acted by the same person and the swing_baseball and throw classes generally contain video of the same baseball tournament. Moreover, flic_flac and cartwheel are two performances of the same gymnastics activity.

7. Conclusions

In this work, we propose a two-stream neural network concept, named SE-LSTM, considering the advantages of utilizing both spatial and motion information to classify dynamic patterns of videos. The SE-LSTM extends the general purpose of LSTM by exchanging information with previous cell states of both appearance and motion stream. The modified objective function of the LSTM unit is a new contribution of this work, presented in Section 3.1.2. Combining this SE-LSTM concept with CNNs, we investigated the advantages of communicating spatial and motion features to classify dynamic patterns by implementing the Dual-CNNSELSTM two-stream neural net architecture. This proposed architecture can model static visual features, short-term motion features, and long-term temporal features. In this architecture, first, we extracted spatial and motion features training over two CNNs on static RGB frames and stacked optical flows. Then, for long-term temporal modeling, the two extracted types of features were used separately as inputs of two LSTM networks. During the training, the time step cell states of both streams communicate with each other to share the spatiotemporal features.

To validate this proposed Dual-CNNSELSTM model, we used fireworks, hand gestures, and HMDB51 datasets during this study. As baseline models of this study, we implemented two single streams and two two-stream learning models. When we were comparing the results of the baseline and the proposed Dual-CNNSELSTM model, the Dual-CNNSELSTM achieved significant accuracy with all experimental datasets. Furthermore, the datasets with both the baseline model and the proposed model significantly outperformed similar state-of-the-art methods with the HMDB51. Based on the conditions in which we created our fireworks and hand gesture datasets as well as the experimental results, we can see some limitations of our model. The videos should not have any other moving objects except the target objects and should have static backgrounds, to attain high classification accuracy. In future work, we plan to remove all the misclassified data from our experimental datasets to reduce the misclassifications. Another promising future direction is to classify the different kinds of dynamic patterns in the video data to enhance the robustness of our proposed model.

Author Contributions: Conceptualization, S P Kasthuri Arachchi; Data curation, S P Kasthuri Arachchi and Noorkholis Luthfil Hakim; Funding acquisition, Timothy K. Shih; Methodology, S P Kasthuri Arachchi; Resources, Noorkholis Luthfil Hakim; Supervision, Timothy K. Shih; Validation, S P Kasthuri Arachchi and Noorkholis Luthfil Hakim; Writing – original draft, S P Kasthuri Arachchi; Writing – review & editing, S P Kasthuri Arachchi and Timothy K. Shih.

Funding: This research is funded by the Ministry of Science and Technology (MOST) under the grant number MOST 109-2634-F-008-008.

Acknowledgments: We acknowledge the support of the Ministry of Science and Technology, Taiwan.

Conflicts of Interest: The authors declare no conflict of interest.
Appendix A

Figure A1. Visualizing the Proposed Dual-CNNSELSTM Architecture.
Figure A2. Visualizing the Dual-3DCNNLSTM Architecture.
| Layer (type) | Output Shape | Param # |
|-------------|--------------|---------|
| InputEth (InputLayer) | (None, 50, 50, 3) | 0       |
| InputOpt (InputLayer) | (None, 50, 50, 3) | 0       |
| time_distributed_1 (TimeDistrib) | (None, 50, 50, 3) | 2432    |
| time_distributed_2 (TimeDistrib) | (None, 50, 50, 3) | 2432    |
| time_distributed_3 (TimeDistrib) | (None, 50, 50, 3) | 120     |
| time_distributed_4 (TimeDistrib) | (None, 50, 50, 3) | 120     |
| time_distributed_5 (TimeDistrib) | (None, 50, 25, 25) | 610496  |
| time_distributed_6 (TimeDistrib) | (None, 50, 25, 25) | 6256    |
| time_distributed_7 (TimeDistrib) | (None, 50, 25, 25) | 6256    |
| time_distributed_8 (TimeDistrib) | (None, 25, 25, 25) | 60      |
| time_distributed_9 (TimeDistrib) | (None, 25, 25, 25) | 60      |
| time_distributed_10 (TimeDistrib) | (None, 25, 25, 25) | 173856  |
| time_distributed_11 (TimeDistrib) | (None, 25, 25, 25) | 173856  |
| time_distributed_12 (TimeDistrib) | (None, 25, 25, 25) | 1512    |
| time_distributed_13 (TimeDistrib) | (None, 25, 25, 25) | 1512    |
| time_distributed_14 (TimeDistrib) | (None, 25, 25, 25) | 10      |
| time_distributed_15 (TimeDistrib) | (None, 25, 25, 25) | 10      |
| time_distributed_16 (TimeDistrib) | (None, 25, 25, 25) | 1024    |
| time_distributed_17 (TimeDistrib) | (None, 25, 25, 25) | 1024    |
| time_distributed_18 (TimeDistrib) | (None, 25, 25, 25) | 1024    |
| time_distributed_19 (TimeDistrib) | (None, 25, 25, 25) | 1024    |
| time_distributed_20 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_21 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_22 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_23 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_24 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_25 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_26 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_27 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_28 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_29 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_30 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_31 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_32 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_33 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_34 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_35 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_36 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_37 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_38 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_39 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_40 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_41 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_42 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_43 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| time_distributed_44 (TimeDistrib) | (None, 25, 25, 25) | 0       |
| lstm_1 (LSTM) | (None, 256) | 525312  |
| lstm_2 (LSTM) | (None, 256) | 525312  |
| dropout_2 (Dropout) | (None, 256) | 0       |
| dropout_4 (Dropout) | (None, 256) | 0       |
| dense_2 (Dense) | (None, 256) | 66790   |
| dense_4 (Dense) | (None, 256) | 66790   |
| concatenate_1 (Concatenate) | (None, 12) | 0       |
| dense_5 (Dense) | (None, 1) | 2565    |

Figure A3. Dual-CNNSELSTM Output shape and number of weights in each layer.
| Layer Type          | Output Shape     | Param #  |
|---------------------|------------------|----------|
| Input (InputLayer)  | (None, 50, 50, 50) | 0        |
| Inputdepth (InputLayer) | (None, 50, 50, 50) | 0        |
| conv3d_1 (Conv3D)   | (None, 50, 50, 50) | 3 2624   |
| conv3d_7 (Conv3D)   | (None, 50, 50, 50) | 3 856    |
| activation_1 (Activation) | (None, 50, 50, 50) | 3 0      |
| activation_7 (Activation) | (None, 50, 50, 50) | 3 0      |
| batch_normalization_1 (BatchNorm) | (None, 50, 50, 50) | 3 128    |
| batch_normalization_7 (BatchNorm) | (None, 50, 50, 50) | 3 128    |
| conv3d_2 (Conv3D)   | (None, 50, 50, 50) | 3 27600  |
| conv3d_8 (Conv3D)   | (None, 50, 50, 50) | 3 27600  |
| activation_2 (Activation) | (None, 50, 50, 50) | 3 0      |
| activation_8 (Activation) | (None, 50, 50, 50) | 3 0      |
| batch_normalization_2 (BatchNorm) | (None, 50, 50, 50) | 3 128    |
| batch_normalization_8 (BatchNorm) | (None, 50, 50, 50) | 3 128    |
| max_pooling3d_1 (MaxPooling3D) | (None, 17, 17, 17) | 3 0      |
| max_pooling3d_3 (MaxPooling3D) | (None, 17, 17, 17) | 3 0      |
| dropout_1 (Dropout) | (None, 17, 17, 17) | 3 0      |
| dropout_3 (Dropout) | (None, 17, 17, 17) | 3 0      |
| conv3d_9 (Conv3D)   | (None, 17, 17, 17) | 6 55360  |
| conv3d_10 (Conv3D)  | (None, 17, 17, 17) | 6 55360  |
| activation_9 (Activation) | (None, 17, 17, 17) | 6 0      |
| activation_9 (Activation) | (None, 17, 17, 17) | 6 0      |
| batch_normalization_3 (BatchNorm) | (None, 17, 17, 17) | 6 256    |
| batch_normalization_9 (BatchNorm) | (None, 17, 17, 17) | 6 256    |
| conv3d_4 (Conv3D)   | (None, 17, 17, 17) | 6 610656 |
| conv3d_11 (Conv3D)  | (None, 17, 17, 17) | 6 610656 |
| activation_4 (Activation) | (None, 17, 17, 17) | 6 0      |
| activation_10 (Activation) | (None, 17, 17, 17) | 6 0      |
| batch_normalization_4 (BatchNorm) | (None, 17, 17, 17) | 6 256    |
| batch_normalization_10 (BatchNorm) | (None, 17, 17, 17) | 6 256    |
| max_pooling3d_2 (MaxPooling3D) | (None, 6, 6, 6, 6) | 0        |
| max_pooling3d_4 (MaxPooling3D) | (None, 6, 6, 6, 6) | 0        |
| dropout_2 (Dropout) | (None, 6, 6, 6, 6) | 0        |
| dropout_4 (Dropout) | (None, 6, 6, 6, 6) | 0        |
| conv3d_5 (Conv3D)   | (None, 6, 6, 6, 6) | 110856   |
| conv3d_12 (Conv3D)  | (None, 6, 6, 6, 6) | 110856   |
| activation_5 (Activation) | (None, 6, 6, 6, 6) | 0        |
| activation_11 (Activation) | (None, 6, 6, 6, 6) | 0        |
| batch_normalization_5 (BatchNorm) | (None, 6, 6, 6, 6) | 256      |
| batch_normalization_11 (BatchNorm) | (None, 6, 6, 6, 6) | 256      |
| conv3d_6 (Conv3D)   | (None, 6, 6, 6, 50) | 86450    |
| conv3d_13 (Conv3D)  | (None, 6, 6, 6, 50) | 86450    |
| activation_6 (Activation) | (None, 6, 6, 6, 50) | 0        |
| activation_12 (Activation) | (None, 6, 6, 6, 50) | 0        |
| batch_normalization_6 (BatchNorm) | (None, 6, 6, 6, 50) | 200      |
| batch_normalization_12 (BatchNorm) | (None, 6, 6, 6, 50) | 200      |
| flatten_1 (Flatten) | (None, 100000) | 0        |
| flatten_2 (Flatten) | (None, 100000) | 0        |
| reshape_1 (Reshape) | (None, 50, 216) | 0        |
| reshape_2 (Reshape) | (None, 50, 216) | 0        |
| lstm_1 (LSTM)       | (None, 128) | 176640   |
| lstm_2 (LSTM)       | (None, 128) | 176640   |
| concatenate_1 (Concatenate) | (None, 256) | 0        |
| dense_1 (Dense)     | (None, 11) | 2056     |

Total params: 1,142,958
Trainable params: 1,141,694
Non-trainable params: 1,224

Figure A4. Dual-3DCNNLSTM Output shape and number of weights in each layer.
| Method                  | Accuracy | Architecture                                                                 |
|------------------------|----------|------------------------------------------------------------------------------|
| Two-stream model       | 58.00%   | Spatial stream ConvNet                                                        |
| Two-stream model       | 59.40%   | Motion stream ConvNet                                                         |
| Two-stream SVM [6]     |          | RGB stream(C3D)                                                              |
| Two-stream SVM [6]     |          | Optical Flow stream(C3D)                                                     |
| Two-stream Fusion [52] | 65.40%   | Spatial stream ConvNet                                                        |
| Two-stream Fusion + IDT [52] |       | Temporal stream ConvNet                                                       |
| ST-ResNet[53]         | 66.41%   | Motion stream ConvNet                                                         |
| ST-ResNet + IDT[53]    | 70.03%   | RGB stream(C3D)                                                              |
| Dual-3DCNNLSTM(ours)   | 67.84%   | Optical Flow stream(C3D)                                                     |
| Dual-CNN5LSTM(ours)    | 69.86%   | Spatial stream ConvNet                                                        |

Figure A5. Comparison of the Different Architectures reported in Table 2.
References

1. Farabet, C.; Couprie, C.; Najman L.; LeCun, Y. Learning Hierarchical Features for Scene Labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* 2013, 35, 1915–1929.

2. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv 2013*, arXiv:1311.2524 [cs].

3. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 2014*, arXiv:1409.1556 [cs].

4. Ji, S.; Xu, W.; Yang M.; Yu, K. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 2013, 35, 221–231.

5. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* 1994, 5, 157–166.

6. Simonyan, K.; Zisserman, A. Two-Stream Convolutional Networks for Action Recognition in Videos. *Adv. Neural Inf. Process. Syst.* 2014, 1,568–576.

7. Wang, H.; Schmid, C. Action Recognition with Improved Trajectories. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013.

8. Ng, J.Y.-H.; Hausknecht, M.; Vijayanarasimhan, S.; Vinyals, O.; Monga, R.; Toderici, G. Beyond Short Snippets: Deep Networks for Video Classification. *arXiv 2015*, arXiv:1503.08909 [cs].

9. Wu, Z.; Wang, X.; Jiang, Y.-G.; Ye, H.; Xue, X. Modeling Spatial-Temporal Dependencies in a Hybrid Deep Learning Framework for Video Classification. *arXiv 2015*, arXiv:1504.01561 [cs].

10. Shahroudy, A.; Liu, J.; Ng, T.-T.; Wang, G. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, June 27–30 2016; pp. 1010–1019.

11. Singh, B.; Marks, T.K.; Jones, M.; Tuzel, O.; Shao, M. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, June 27–30 2016; pp. 1961–1970.

12. Pigou, L.; Van Den Oord, A.; Dieleman, S.; Van Herreweghe, M.; Dambre, J. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. *Int. J. Comput. Vis.* 2018, 126, 430–439.

13. Du, Y.; Wang, W.; Wang, L. Hierarchical recurrent neural network for skeleton based action recognition. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1110–1118.

14. Veeriah, V.; Zhuang, N.; Qi, G.-J. Differential recurrent neural networks for action recognition. In Proceedings of the IEEE International Conference on Computer Vision, Convention Center, Santiago, Chile, 13–16 December 2015; pp. 4041–4049.

15. Molchanov, P.; Yang, X.; Gupta, S.; Kim, K.; Tyree, S.; Kautz, J. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4207–4215.

16. Hakim, N.L.; Shih, T.K.; Kasthuri Arachchi, S.P.; Aditya, W.; Chen, Y.-C.; Lin, C.-Y. Dynamic Hand Gesture Recognition Using 3DCNN and LSTM with FSM Context-Aware Model. *Sensors* 2019, 19, 5429.

17. Abu-El-Haija, S.; Kothari, N.; Lee, J.; Natsev, P.; Toderici, G.; Varadarajan, B.; Vijayanarasimhan, S. YouTube-8M: A Large-Scale Video Classification Benchmark. *arXiv 2016*, arXiv:1609.08675 [cs].

18. Jiang Y.; Wu, Z.; Wang, J.; Xue X.; Chang, S. Exploiting Feature and Class Relationships in Video Categorization with Regularized Deep Neural Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 2018, 40, 352–364.

19. Kasthuri Arachchi, S.P.; Shih, T.K.; Lin, C.Y.; Wijayarathna, G. Deep Learning-based Firework Video Pattern Classification. *J. Internet Technol.* 2019, 20, 2033–2042.

20. Available online: http://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/ (accessed on 19 January 2020).

21. Laptev, I. On space-time interest points. *Int. J. Comput. Vis.* 2005, 64, 107–123.

22. Laptev, M.; Marszalek Schmid, C.; Rozenfeld, B. Learning Realistic Human Actions from Movies. In Proceedings of the CVPR, Anchorage, AK, USA, 23–28 June 2008.
23. Wang, H.; Ullah, M.M.; Klaser, A.; Laptev, I.; Schmid, C. Evaluation of Local Spatio-Temporal Features for Action Recognition. In Proceedings of the British Machine Vision Conference (BMVC), London, United Kingdom, 7-10 September, 2009.

24. Oneata, D.; Verbeek, J.; Schmid, C. Action and event recognition with fisher vectors on a compact feature set. In Proceedings of the ICCV, Sydney, NSW, Australia, 1–8 December 2013.

25. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 8–10 June 2015.

26. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. Large-scale Video Classification with Convolutional Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014.

27. Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE international conference on computer vision, Washington, DC, USA, 7–13 December 2015.

28. Wang, L.; Qiao, Y.; Tang, X. Action Recognition with Trajectory-Pooled Deep-Convolutional Descriptors. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 8–10 June 2015.

29. Tang, K.; Fei-Fei, L.; Koller, D. Learning latent temporal structure for complex event detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 18–20 June 2012.

30. Wang Y.; Mori, G. Max-margin hidden conditional random fields for human action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.

31. Fernando, B.; Gavves, E.; Oramas, J.M.; Ghodrati, A.; Tuytelaars, T. Modeling Video Evolution for Action Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–10, June 2015.

32. Donahue, J.; Hendricks, L.A.; Rohrbach, M.; Venugopalan, S.; Guadarrama, S.; Saenko, K.; Darrell, T. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.

33. Yao, L.; Torabi, A.; Cho, K.; Ballas, N.; Pal, C.; Larochelle, H.; Courville, A. Describing Videos by Exploiting Temporal Structure. arXiv 2015, arXiv:1502.08029 [cs, stat].

34. Zhang, H.; Wang, M.; Hong R.; Chua, T.S. Play and Rewind: Optimizing Binary Representations of Videos by Self-Supervised Temporal Hashing. In Proceedings of the 24th ACM International Conference on Multimedia, Amsterdam, The Netherlands, 15–19 October 2016.

35. Graves, A.; Mohamed, A.; Hinton, G.; Speech Recognition with Deep Recurrent Neural Networks. arXiv 2013, arXiv:1303.5778 [cs].

36. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. Neural Comput. 1997, 9, 1735–1780.

37. Sturman, D.J.; Zeitzer, D. A survey of glove-based input. IEEE Comput. Graph. Appl. 1994, 14, 30–39.

38. Wang, R.Y.; Popović, J. Real-time hand-tracking with a color glove. ACM Trans. Graph. TOG 2009, 28, 63.

39. Mummid, C.; Leo, F.; Verma, K.; Kasireddy, S.; Scholl, P.; Kempfle, J.; Laerhoven, K. Real-Time and Embedded Detection of Hand Gestures with an IMU-Based Glove, Inform. Multidiscip. Digit. Publ. Inst. 2018, 5, 28.

40. Oyedotun, O.K.; Khashman, A. Deep learning in vision-based static hand gesture recognition. Neural Comput. Appl. 2017, 28, 3941–3951.

41. Molchanov, P.; Gupta, S.; Kim, K.; Kautz, J. Hand gesture recognition with 3D convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Boston, MA, USA, 7–12 June 2015; pp. 1–7.

42. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS 2012); NV, USA, 3-8 December 2012; pp. 1097–1105.

43. Pilou, L.; Dieleman, S.; Kindermans, P.-J.; Schrauwen, B. Sign Language Recognition Using Convolutional Neural Networks; Springer: Berlin/Heidelberg, Germany, 2014; pp. 572–578.

44. Home -Keras Documentation. Available online: https://keras.io/ (accessed on 26 April 2019).
45. Understanding LSTM Networks - Colah’s blog. Available online: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed on 26 April 2019).

46. Deep Learning Based Dynamic Patterns Classification of Videos Using Spatio-Temporal Virtual Communication. Available online: https://srilankanstudytaiwan.wixsite.com/twostreamvclstm/ (accessed on 18 February 2020).

47. Khong, V.M.; Tran, T.H. Improving human action recognition with two-stream 3d convolutional neural network. In Proceedings of the 2018 1st International Conference on Multimedia Analysis and Pattern Recognition (MAPR), Ho Chi Minh City, Vietnam, 5–6 April 2018; pp. 1–6.

48. Farnebäck, G. Two-Frame Motion Estimation Based on Polynomial Expansion. In Scandinavian Conference on Image Analysis; Springer: Halmstad, Sweden, June 29 – July 2, 2003, pp. 363–370.

49. Kingma, D.P.; Adam Ba, J. A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference for Learning Representations, San Diego, CA, USA, 7–9 May 2015; arxiv:1412.6980.

50. Igel, C.; Hüskens, M. Improving the Rprop Learning Algorithm. In Proceedings of the Second International Symposium on Neural Computation, NC’2000, ICSC; Academic Press: Cambridge, MA, USA, 23-26 May 2000; pp. 115–121.

51. Feichtenhofer, C.; Pinz, A.; Zisserman, A. Convolutional Two-Stream Network Fusion for Video Action Recognition, In Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1933–1941.

52. Feichtenhofer, C.; Pinz, A.; Wildes, R.P. Spatiotemporal Residual Networks for Video Action Recognition. Adv. Neural Inf. Process. Syst; Barcelona, Spain 5-10 December 2016; pp.3468–3476.