# Deep Learning-Based Firework Video Pattern Classification

S. P. Kasthuri Arachchi[1], Timothy K. Shih[1], Chih-Yang Lin[2], Gamini Wijayarathna[3]

[1]Department of Computer Science and Information Engineering, National Central University, Taiwan
[2]Department of Electrical Engineering, Yuan-Ze University, Taiwan
[3]Department of Software Engineering, Faculty of Computing and Technology, University of Kelaniya, Sri Lanka
{sandelik, timothykshih}@gmail.com, andrewlin2011@gmail.com, gamini@kln.ac.lk

## Abstract

The video analytics technology has been a rapidly improving discipline in the past decade. With the recent developments in computer vision, we now have the ability to mine massive video data to obtain a clear understanding of what is happening in the world. Because of the remarkable successes of deep learning, currently, we are able to improve video analysis performance significantly than traditional statistical approaches. This study focuses on classifying the patterns of firework videos with various deep learning techniques using spatial and temporal features, beyond common types of pattern classifications. Among successful artificial neural networks, Convolutional Neural Networks (CNN) have demonstrated superiority on modeling high-level visual concepts, while Long Short-term Memory (LSTM) and Gated Recurrent Unit (GRU) units have shown great talent in modeling temporal dynamics in video-based pattern classification. Our basic models consist of CNN, LSTM, and GRU and, we did experiments by fine-tuning the parameters of layers and using different dropout values with sequence LSTM and GRU models. Our experimental results demonstrated that the model with a sequence of LSTM units and double dropout layers—one for input and another for hidden layers—outperforms the other experimental models with the training accuracy of 83.05%.

**Keywords:** Deep learning, Fireworks, Convolution Neural Network, Recurrent Neural Network

## 1 Introduction

Nowadays Firework shows are doing a major role in entertainment purposes. On the other hand, fireworks competitions are also frequently held at a number of famous places. People who are planning to arrange such kind of firework shows or competitions have to face a big challenge with firework types selection, because of the peoples' interest of their entertainment is mainly depend on the types of fireworks that they are decided to display. To address this challenge, our work proposed a few methods to classify types of fireworks using existing videos. Further improvement of our work will help such organizers to get the idea of widely used firework types among the most popular fireworks shows. Based on these resulting types they may able to decide which combination of fireworks that they can display in order to get the peoples' attraction to their event.

Because of the video data consists of temporal features in addition to the spatial features of 2D images; video classification has become an inimitable challenge. Among these classification methods, the majority is dealing with the same pattern or physical action with different contexts or backgrounds. More preciously, video frames of a particular video consist same object or design, human action, face, gesture, or scene. Beyond these common types of classifications, our work discusses a few models, which attempt to classify the patterns of firework using a different kind of Deep Learning (DL) techniques. According to our knowledge, this is the first study of identifying firework patterns using different DL models.

More recently, DL has been receiving increasing attention on video pattern classification and prediction applications. However, the challenge is to design a proper DL model to get the highest classification accuracy with the minimum loss using various deep learning techniques. Among successful DL techniques CNN and both LSTM [1] and GRU [2]—variants of Recurrent Neural Networks (RNN), are widely used to build complex video-based classification models.

Before designing models for the experiments, the principal problem that we had to face is a sufficient firework dataset for the training phase. Even there are many datasets for image, text, human action recognition, and sports classification, we did not find any dataset containing fireworks videos. Therefore, we focused to create a new dataset for our classification task. As we emphasized before our purpose is to find a best-fit model for firework pattern classification. Among different kinds of DL techniques, the next challenge was to select suitable techniques for designing the model to classify firework types. Even

*Corresponding Author: S. P. Kasthuri Arachchi; E-mail: sandelik@gmail.com
DOI: 10.3966/160792642019122007002

some previous works based on video classifications designed good models [3-5], we were doubtful whether those satisfy our requirements because of the firework videos consist with different patterns in most time steps comparison to such video classifications.

To address the above issues our first contribution was to create a dataset with different types of fireworks. To maintain the fairness of our dataset and control the scope of our study we selected firework videos under some conditions and restrictions as discussed under methodology. In addition, we applied some augmentation techniques to enhance the size of the dataset and our current dataset consists of 8 firework types with 1500 video clips. Our second and but the main contribution of this study is to recognize proper DL techniques for classifying firework types using existing video data. By considering the advantages of previous studies [1-2] our work focused on three main DL models: Long-term Recurrent Convolutional Networks (LRCNs) [6], which is a combination of CNN and LSTM which is dealing with long-range temporal recursion; sequence of LSTMs, which is one of the most effective structures to model sequential data and, sequence of GRUs—a simplified version of LSTM. The fundamental difference between LSTM and GRU is that an LSTM has three gates as input, output and forget gate while GRU has reset and update gates. Hence GRU is simpler, easy to modify and train faster than LSTMs.

The rest of the paper is organized as follows. Sections 2 and Section 3 discuss the related work and the proposed method. Next, experimental results and analysis are presented in Section 4. Finally, the conclusion and future works are concluded in Section 5.

## 2 Related Work

Video classification is a vital challenge in computer vision. In order to provide state-of-art results, traditional video classification research has been successful at obtaining overall video descriptors, which combine both appearance and motion information. However, because of the remarkable success of DL techniques, video classification performance has been improved significantly. Theatrical progress has been achieved by supervised convolutional models on image recognition tasks [7-9] and medical decision support images classification [10] as well. By taking such advantages, a number of extensions to process video have been recently proposed. Apart from that a proper video classification models should allow processing of variable length input sequences and provide variable length outputs, beyond conventional one versus all prediction tasks.

In order to analyze sequences, LRCN, which associate CNN top of RNN in one architecture has been proposed [6]. This work has become beneficial in several tasks: activity recognition; images description; and video description. Like conventional RNN, LRCN is able to handle "sequence input to single output," "single output to sequence output," and "sequence input to sequence output". In addition, the success of deep networks is also a result of the development of widely applicable simple learning techniques such as dropout [11], Rectified Linear Unit (*ReLUs*) [8]—a most common activation function and gradient clipping [12].

### 2.1 CNN and RNN

CNN is a special case of the neural network, widely used in pattern and image classification problems because of its advantages compared to other techniques. While CNN deals with models, which process on single input, RNN fits for sequential input. However, in most practical cases, RNN cannot memorize previous information that is far from current state. When input sequence is too long, training RNN with back-propagation through time (BPTT) may face to the problem of vanishing gradient (gradient becomes small when it flows back too many steps) [11]. Consequently, RNN forget long-term dependencies. To address this issue, a new neural net technique called LSTM, which has capability of remembering information for a long range of time steps has been proposed [1]. Replacing simple neural unit by LSTM unit in RNN showed significant improvement. Moreover, GRU has less computational steps than LSTM [2]. The idea of GRU makes LSTM simpler by summing up forget gate and input gate into single update gate.

### 2.2 Learning Rate

Neural networks are often trained by gradient descent on the weights. The learning rate determines how fast these weights are changing. Therefore, it is essential to adapt learning rate with suitable parameters. *Adagrad* described a way of adapting to learning rate with parameters, by adjusting learning rate process at the training stage [13]. Although *Adagrad* helps to adjust the learning rate, it can make trouble when training for long time. Because of many time steps, learning rate decreases and when learning rate approximates to zero, it is impossible to get any more parameters. Hence, the solution to overcome *Adagrad's* learning rate decreasing problem—a new thing that tries to adapt resilient prop (*rprop*) [14], widely used to train RNNs for batch training, to stochastic gradient descent. Therefore, in this work, we used *RMSprop* to optimize the learning rate.

## 3 Methodology

To classify the types of firework videos, we designed three main models using DL techniques namely, LRCN, LSTM, and GRU. LRCN, which is a combination of CNN and RNN (specifically LSTM)

proposed for sequence analyzing with Caffe [15]. CNN layers of LRCN the model naturally extract spatial features while following LSTM layers extract temporal features. In this study, our purpose is to investigate the ability to identify spatial-temporal features of fire-work video. Hence, we performed our first set of experiments based on LRCN architecture. As mentioned earlier, LSTM has the capability of remembering information for a long range of time steps of variable length inputs. Therefore, our second model designed using the sequence of recurrent units, *SeqLSTM*. GRU is related to LSTM as both are utilizing different ways of gating information to prevent vanishing gradient problem. The GRU unit controls the flow of information like the LSTM unit thus computationally more efficient [16]. Moreover, apart from *SeqLSTM*, we designed another model using a sequence of GRU units for performance comparison.

To find the best fit model to achieve the highest average accuracy we fine-tuned each model's necessary parameters such as dropout, learning rate, and gradient clipping—a popular technique to overcome the exploding gradients problem during backpropagation. Setting a good learning rate is tricky in DL models. For all our experiments, we used *RMSprop* with a staring learning rate of $10^{-3}$ and after 10 epochs decreased by 10 times ($10^{-4}$). According to the study [12], in practice, proper dropout values for real-valued inputs like video can be set to 0.8 and for hidden layers, it varies from 0.5 to 0.8.

## 3.1 Firework Dataset

Dataset size is one of the important factors that deep neural network architecture depends on. Due to the unavailability of a standard firework video dataset, first we had to create our own dataset. We downloaded different types of firework videos with low resolution, in order to decrease the computational cost and split them into less than 8-second length clips. Our first experimental dataset consists of 750 video clips and gradually increased the dataset up to 1500 for experimental purposes. Also, we applied a few augmentation techniques such as flip-up, flip-down, and rotate to increase the size of the dataset. However, firework videos have been selected under three conditions to maintain the fairness of our dataset and control the scope of our study as listed below.

- Each video should have one firework type.
- Videos should not contain any other moving objects except firework.
- The background should be static with invisible objects.

Sample firework video types, which are not included in our dataset are shown in Figure 1. We manually categorized the dataset into eight classes: *Chrysanhemum*, *Crosette*, *Desi*, *Dot*, *Drop*, *Fish*, *Palm*, and *WaterFlower* by looking at each video clip (Figure 2). We further randomly divided the dataset into 3 sections as training (70%), validation (15%), and testing (15%) to evaluate the model performances.

**Figure 1.** Sample firework video not satisfied with our data selection conditions. Such videos contain multiple types of fireworks, moving objects like ship and airplane while exploring the firework and, visible and statics objects thru the background

## 3.2 Experimental Architectures

Our experiments were based on three main models as we described earlier. The first model followed LRCN architecture and the other two were based on LSTMs and GRUs. LSTM and GRU units of these models consist of a single dropout layer inside hidden layers. In addition to that, we tested our dataset by adding another dropout layer in between visible layer or top of the first hidden layer (to input volume).

### 3.2.1 LRCN - with Single Dropout Layer (*Lrcn1Drop*) and Double Dropout Layers (*Lrcn2Drop*)

As appears in Table 1, the architecture of the *Lrcn1Drop* model designed using five layers including a single dropout layer. We limited our input size to 240×320 resolution because of the high computational cost of high- resolution input and since we are using RGB video with 3 channels. This model contains four CNN layers and followed by a sequence of LSTM units. Reducing input size at lower layers helps to decrease the computational cost. Hence, low layers such as layer 1 and layer 2, we wanted to reduce the size of input by 2 times for each layer. To do this we see the stride size to 2 when input passing through the CNN layer. To avoid high information loss, our input feature planes set to 32 in the first layer. Moreover, the Batch Normalization layer addresses the vanishing and exploding gradients problem and helps faster training as well as removes the internal covariate shift problem.

**Table 1.** Details of five layers of LRCN with 4 CNN layers and 1 LSTM layer

| | |
|---|---|
| **Input** | 320×240 size, 3 color channels RGB |
| **CNN Layer 1** | 32 planes, 5x5 kernel, stride 2, pad 3 |
| | Batch Normalization |
| | *ReLU* |
| | Max Pooling 3x3 kernel, stride 1, pad 1 |
| **CNN Layer 2** | 64 planes, 3x3 kernel, stride 1, pad 2 |
| | Batch Normalization |
| | *ReLU* |
| | Max Pooling 3x3 kernel, stride 2, pad 1 |
| **CNN Layer 3** | 128 planes, 3x3 kernel, stride 1, pad 1 |
| | *ReLU* |
| **CNN Layer 4** | 256 planes, 3x3 kernel, stride 1, pad 1 |
| | *ReLU* |
| | Max Pooling 3x3 kernel, stride 1, pad 1 |
| **Layer 5** | LSTM 256 hidden size with 0.5 Dropout |
| **Fully Connected** | 8 classes output |
| | Log *Softmax* |

LSTM layers we tested by adding different size of hidden layers. Since Dropout is necessary to prevent over-fitting, it is important to place a dropout layer between the last CNN layer and LSTM layers—a large number of parameters passing through CNN to LSTM is fully connected. For the *Lrcn2Drop* model, a modification of LRCN, we inserted an additional dropout layer to input volume of LSTMs to avoid over-fitting caused by real-valued inputs [12]. Because of inner CNN layers not consists of many parameters as fully connected and most probably dropout is not preventing co-adaption for CNN, we did not add dropout layers within CNN layers.

#### 3.2.2 RNN-*SeqLSTM* with Single Dropout Layer (*Lstm1Drop*) and *SeqLSTM* with Double Dropout Layers (*Lstm2Drop*)

Our second DL model, based on LSTMs, a variant of RNN, which perform much faster and detects long-term dependencies of the ordered sequence of frames. Besides the problem of long training time of RNN it also faces the problem of memory loss during long running. Subsequently, LSTM units use memory cells to store and output information thus allow to discover better long-range temporal relationships than RNN. The input volume of model *Lstm1Drop* also similar to our previous model and we randomly assigned 256 hidden layers with 0.6 dropout value within hidden layers—selected after fine-tuning. Similar to the previous model we added additional dropout layer to input volume of the *Lstm2Drop* model in order to compare the model accuracy.

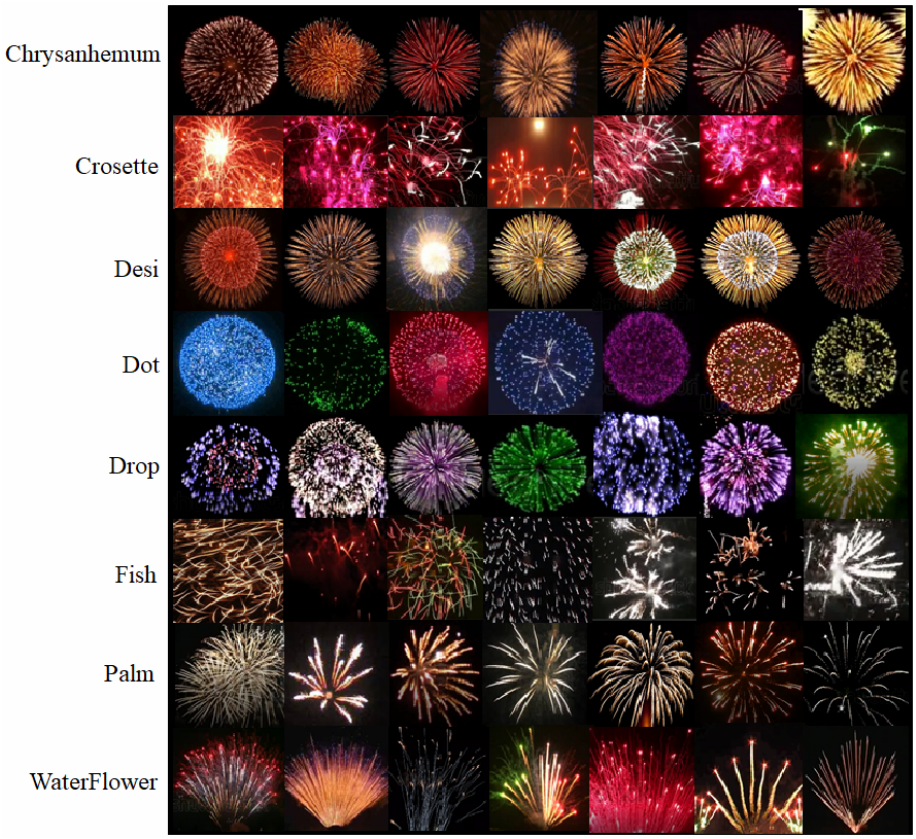



**Figure 2.** Manually categorized eight firework classes. All classes represent popular names of firework types

#### 3.2.3 RNN - *SeqGRU* with Single Dropout Layer (*Gru1Drop*) and *SeqGRU* with Double Dropout Layers (*Gru2Drop*)

GRU, another variant of RNN, as well as a simplified version of LSTM, makes each recurrent unit adaptively capture dependencies of different time scales. Even GRU is relatively new, we experienced that its performance as on a par with LSTM, but computationally more efficient because of less complex structure. Therefore, we designed *Gru1Drop* with one dropout layer and *Gru2Drop* with double dropout layers as our testing models to compare results of LSTMs and GRUs.

## 4 Implementation and Experimental Results Discussion

The next important factor after creating a dataset and designing models is to decide a proper DL framework to implement the system. According to the study [15], even LRCN was originally implemented in Caffe, in practice, Caffe is not the best for RNN and Torch7 is demonstrated as a reasonable framework because of its ability to script new modules, especially for RNN [17]. Moreover, our main contribution is to compare the model accuracy of LRCN, LSTM, and GRU. Hence, all models had to implement in the same framework. These reasons motivated us to implement the system on Torch7 and our supportive operating system was Ubuntu 16.04. For best practice, we implemented our system with 3 main modules: data loader for loading any type of video data; a model that contacts DL architecture; and a train/validation/test module to perform optimization tasks. Our dataset is about 2.25GB and we trained each model using a GPU (GeForce GTX 1080).

### 4.1 Model Performances with Different Size of Datasets

Big data is often discussed along with deep learning. In practice, a good amount of data needs for training a deep model, since it's necessary to make sure the model's capability of generalization. Under this experiment, we evaluated model skill over the size of the training dataset.

Our first experimental dataset consists of 750 firework video and we divided it as a similar ratio as mentioned in methodology. In addition, the experimental conditions also unique with the current dataset, which has 1250 video, as we discussed under the same section. For better comparison, we tested our models with different dataset sizes as 500, 750, 1000, 1250, and 1500 video clips. Table 2 to Table 6 illustrates the average model accuracies and losses of each dataset respectively.

**Table 2.** Comparison of model results with 500 video

| Model | Train Acc. | Val. Acc. | Test Acc. | Train Loss | Val. Loss | Test Loss |
|---|---|---|---|---|---|---|
| ***Lrcn 1Drop*** | 33.52% | 25.83% | 23.26% | 1.83% | 2.12% | 1.92% |
| ***Lrcn 2Drop*** | 64.24% | 35.10% | 40.42% | 1.09% | 2.40% | 2.01% |
| ***Lstm 1Drop*** | 63.12% | 44.93% | 38.48% | 1.17% | 2.17% | 1.89% |
| ***Lstm 2Drop*** | 67.31% | 47.74% | 57.36% | 1.14% | 1.89% | 1.12% |
| ***Gru 1Drop*** | 74.58% | 55.10% | 60.38% | 0.83% | 1.01% | 1.85% |
| ***Gru 2Drop*** | 58.10% | 30.46% | 41.01% | 1.47% | 1.58% | 2.21% |

**Table 3.** Comparison of model results with 750 video

| Model | Train Acc. | Val. Acc. | Test Acc. | Train Loss | Val. Loss | Test Loss |
|---|---|---|---|---|---|---|
| ***Lrcn 1Drop*** | 42.33% | 29.65% | 28.34% | 1.95% | 2.43% | 1.85% |
| ***Lrcn 2Drop*** | 68.86% | 42.78% | 41.53% | 0.97% | 2.02% | 1.01% |
| ***Lstm 1Drop*** | 76.24% | 45.00% | 43.48% | 0.75% | 2.11% | 1.89% |
| ***Lstm 2Drop*** | 73.27% | 47.22% | 45.46% | 0.85% | 1.89% | 0.96% |

**Table 4.** Comparison of model results with 1000 video

| Model | Train Acc. | Val. Acc. | Test Acc. | Train Loss | Val. Loss | Test Loss |
|---|---|---|---|---|---|---|
| ***Lrcn 1Drop*** | 24.29% | 23.33% | 22.36% | 1.94% | 2.04% | 1.96% |
| ***Lrcn 2Drop*** | 70.57% | 53.33% | 43.31% | 0.95% | 1.77% | 1.28% |
| ***Lstm 1Drop*** | 68.43% | 48.67% | 47.34% | 0.98% | 1.83% | 0.93% |
| ***Lstm 2Drop*** | 72.71% | 49.36% | 67.86% | 0.86% | 1.78% | 1.01% |

**Table 5.** Comparison of model results with 1250 video

| Model | Train Acc. | Val. Acc. | Test Acc. | Train Loss | Val. Loss | Test Loss |
|---|---|---|---|---|---|---|
| ***Lrcn 1Drop*** | 24.01% | 22.38% | 21.65% | 1.92% | 2.06% | 2.01% |
| ***Lrcn 2Drop*** | 76.56% | 42.27% | 42.13% | 0.70% | 1.86% | 1.01% |
| ***Lstm 1Drop*** | 64.38% | 43.09% | 40.31% | 1.08% | 1.73% | 1.87% |
| ***Lstm 2Drop*** | 82.91% | 45.30% | 78.54% | 0.51% | 1.89% | 1.02% |
| ***Gru 1Drop*** | 63.16% | 40.88% | 38.63% | 1.13% | 1.75% | 1.91% |
| ***Gru 2Drop*** | 54.47% | 37.82% | 35.62% | 1.47% | 2.23% | 2.21% |

**Table 6.** Comparison of model results with 1500 video

| Model | Train Acc. | Val Acc. | Test Acc. | Train Loss | Val. Loss | Test Loss |
|---|---|---|---|---|---|---|
| ***Lstm2 Drop*** | 83.05% | 51.42% | 81.78% | 0.54% | 1.99% | 0.68% |

However, since we noticed that the *Lstm2Drop* model outperforms other models when increasing the number of videos, as an additional experiment 1500 video dataset is only used for evaluating the *Lstm2Drop* model in order to save the computational cost. The *Lstm2Drop* model has shown the highest accuracy in all datasets except the 750 video dataset.

Even though Table 3 shows *Lstm1Drop* has the best accuracy, we noticed that this model addressed the problem of overfitting which we have discussed in a later section. Therefore, we can conclude that the *Lstm2Drop* model is the best model within all datasets. In addition, when going thru the average accuracy of the *Lstm2Drop* model over the dataset size, it enhanced that while increasing the size of dataset the average model accuracy also going up drastically.

## 4.2 Model Performance with Dropout Layers

Dropout deals with ignoring randomly selected neurons during model training. As a result, the model becomes less sensitive to the specific weights of neurons. This, in turn, results in a model that is capable of better generalization and is less likely to over-fit the training data. When comparing both LRCN and LSTM models, the models with two dropout layers were able to achieve the highest average accuracy and minimize the loss considerably. This implies the importance of a dropout between the visible layer and the first hidden layer of LSTMs when our input is real-valued.

Even Table 3 shows the model *Lstm1Drop* get the highest average accuracy of 76.24%, when we plot and see the model loss over a number of epochs, we can notice that this model shows the problem of overfitting. This issue is clearly visible in Figure 3—while train error decreases over the number of epochs, validation error increases. To overcome this issue, we added an additional dropout layer to the input stream of this model, which is defined as *Lstm2Drop*.

Apart from that when observing the results in Table 5, even we did not notice the overfitting issue of the *Lstm1Drop* model, we try to do experiments by adding additional dropout layer to input stream as previous—known as *Lstm2Drop*. After fitting the model, we noticed that accuracy has increased with the value of 82.91% than 64.38% of *Lstm1Drop* and decreases the loss as well.

In addition, by analyzing both average accuracy and loss of training and validation process, *Lstm2Drop* model achieved significant improvements than *Lrcn2Drop* with the average training accuracy of 83.05%, 51.42% of validation accuracy and, 0.54% and 1.99% of train and test loss respectively as

illustrated in Figure 4. These results together demonstrated that our firework dataset was able to achieve the highest average testing accuracy of 81.78% with the model *Lstm2Drop*.

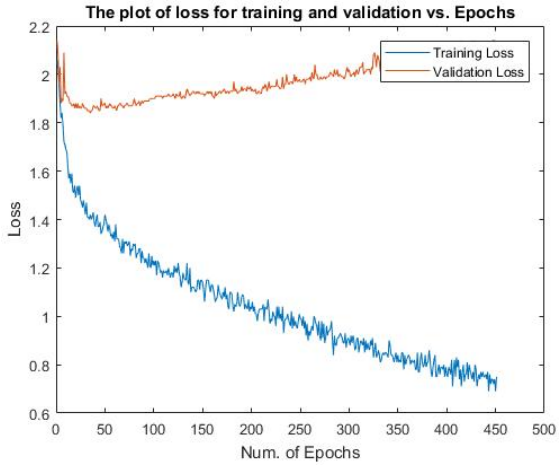



**Figure 3.** Illustration of loss on *Lstm1Drop* model. This clearly shows that this model is overfitting i.e. validation loss is increasing while training loss is decreasing after number of epochs


**Figure 4.** Illustration of loss and accuracy on *Lstm2Drop* model with the 1500 dataset. Model achieved average training accuracy of 83.05%, 81.78% of testing accuracy and, 0.54% and 0.68% of train and test loss

### 4.3 Fine-Tuning Parameters

Later on, we tested the model *Lstm2Drop*, by fine-tuning fixed dropouts with different gradient clipping values in order to identify the best fit. For this experiment, we set input and visible layer dropout as 0.8 and 0.6 for hidden layers while changing gradient clipping (Grad. Clip) values to 0, 5, 10, and 15 (Table 7). Even Grad. Clip is dealing with the exploding gradients, simply it is a very effective solution. We noticed that while increasing Grad. Clip the model accuracy decreased slightly. Hence, we limited experiments to Grad. Clip with the value of 15. Because of the experiment with zero Grad. Clip starting to converge at 420th epoch, we limited all experiments under this to epoch 420, in order to maintain fairness. That is the reason for accuracies ranging around 60%. However, according to Table 7, our *Lstm2Drop* model outputs the highest average accuracy with 5 Grad. Clip value by minimizing the loss.

**Table 7.** Model comparison results with different gradient clipping values

| Grad. Clip | Train Acc. | Test Acc. | Train Loss | Test Loss |
|---|---|---|---|---|
| 0 | 56.53% | 40.61% | 1.28% | 1.94% |
| 5 | 63.61% | 42.27% | 1.13% | 1.76% |
| 10 | 61.05% | 40.88% | 1.14% | 1.78% |
| 15 | 61.04% | 40.61% | 1.12% | 1.80% |

### 4.4 Performance of LSTM and GRU

GRUs have fewer parameters and thus may train a bit faster or need less training data to generalize. However, with large data, the LSTMs with higher expressiveness may lead to better results. To clarify this, we considered *Gru1Drop* and *Gru2Drop* modes with 500 and 1250 datasets. Both of these models have shown the highest accuracies with 500 video dataset as shown in Table 2. This implies that the GRU works better with less training data.

Since our best model is *Lstm2Drop*, in order to compare the performance, we used *Lstm2Drop* and *Gru2Drop* with the same two datasets as above. With both datasets, *Lstm2Drop* has shown the best average accuracy than *Gru2Drop* as in Table 8. This demonstrates LSTM is aware to remember longer sequences than GRUs and outperform them in tasks requiring modeling long-distance relations like firework video.

**Table 8.** Computation times of LSTM and GRU models with both single and double dropout layers

| Data sets | Model | Train Acc. | Test Acc. | Train Loss | Test Loss |
|---|---|---|---|---|---|
| 500 | ***Lstm1Drop*** | 63.12% | 38.48% | 1.17% | 1.89% |
| | ***Gru1Drop*** | 74.58% | 60.38% | 0.83% | 1.85% |
| 1250 | ***Lstm2Drop*** | 82.91% | 78.54% | 0.51% | 1.02% |
| | ***Gru2Drop*** | 54.47% | 35.62% | 1.47% | 1.87% |

While training, we noted all DL models' computational time. Among them, we only considered the time of LSTM and GRU models with both single

and double dropout layers to enhance the faster training of GRUs. Results in Table 9 show that both models with GRU required less time duration to generalize than LSTMs.

**Table 9.** Computation times of LSTM and GRU models with both single and double dropout layers

| Model | Time (hours) |
|---|---|
| ***Lstm1Drop*** | 17.03 |
| ***Gru1Drop*** | 16.06 |
| ***Lstm2Drop*** | 18.53 |
| ***Gru2Drop*** | 13.90 |

## 4.5 Model Complexity

Model complexity has characterized using many factors. In deep learning, the model complexity, especially with LSTM, often refers to the number of hidden layers, included in a given predictive model. In order to find the best number of hidden layers we did experiments by changing the number of hidden layers of the *Lstm2Drop* model.

Because of the computational time we make sure to stop all training at the 300th epoch. We randomly selected the starting number of hidden layers as 256 and increased and decreased hidden layers with a similar ratio. However, our current hardware facilities are not sufficient for 512 hidden layers and hence our upper bound was 448. To find the best-fit model's complexity we considered the loss difference of each running as shown in Table 10 and it verifies that model with 128 hidden layers has the minimum loss value.

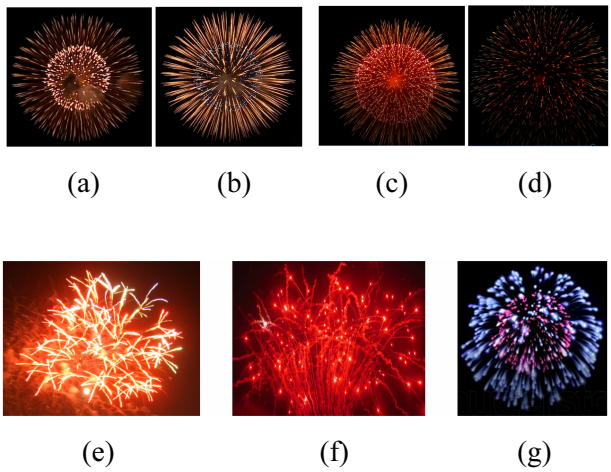
**Table 10.** Average train and test loss of *Lstm2Drop* model over number of hidden layers

| Hidden Layers | Train Loss | Val. Loss | Loss Diff. |
|---|---|---|---|
| 64 | 1.38% | 2.73% | 1.35% |
| 128 | 1.03% | 2.12% | 1.09% |
| 192 | 1.05% | 2.32% | 1.27% |
| 256 | 1.14% | 2.45% | 1.31% |
| 320 | 0.94% | 2.27% | 1.33% |
| 384 | 0.95% | 2.59% | 1.64% |
| 448 | 0.81% | 2.36% | 1.55% |

## 4.6 Classification Mismatches

One crucial factor of classification problem is the uniqueness of features. For better classification results, it is necessary to have independent features among the classes while features inside a class have similar features. To maintain the uniqueness of each class even we carefully categorized while creating our database, we had some mismatch classification results because of the relatively similar types of fireworks that exist among a few classes. Some of those misclassification results of *Lstm2Drop* model with 1500 dataset have illustrated in Figure 5.

Even Figure 5(a) and 5(b) are two frames of a particular video clip in the *Desi* class, it's classified as *Chrysanhemum* class. By the same token, Figure 5(c) and 5(d) represent two frames of a video in the *Desi* class and the model recognized it as in Dot class since part of that firework video contains similar features as such in *Dot* class. In addition, even though Figure 5(e) and 5(f) are samples of *Crosette* class, both classified as belongs to *Fish* and *WaterFlower* classes respectively. Besides, Figure 5(g) recognized as in the *Desi* class even it originally contains in *Dot* class.



(a) (b) (c) (d)

(e) (f) (g)

**Figure 5.** Misclassification results of few classes because of relatively similar features

In general, RNN and its variants: LSTM and GRU are means of learning from sequence using BPTT. In this case usually, the sequence is fed into the neural net at once. Thus if the input is a longer sequence or if it contains too many parameters, LSTM has a chance to forget the previously learned patterns or features, which may cause to do the misclassifications. Applying Attention mechanism—looking at a subsection or prediction of the inputs, and then doing multiple passes, hence it dynamically decided on each pass what it looks at—to the neural net model will help to overcome this classification mismatch [18]. In addition, the attention-based model helps to classify multiple objects within a single image because of its ability to subset wise learning [19-20].

# 5 Conclusion

At the present time, deep learning has greatly affected with video pattern classification in computer vision. During our work we trained and compared several neural net models based on CNN, LSTMs and GRUs to find a suitable model for classifying the patterns of fireworks in videos more efficiently,

beyond the traditional video pattern classification approaches. Our 1500 video dataset consists of 8 firework classes and we did experiments using different dataset sizes and different (single and double) dropout layers. Among all models that we implemented, the model with sequence LSTMs and double dropout layers—one for input and another for hidden layers—outperforms the other models with the testing accuracy of 81.78%.

Even our current dataset consists eight classes we plan to add more types of firework to our dataset including one additional class, which includes a mix (multiple types) firework video, to ensure the robustness of our model. In addition, our future works include combining the current approach with attention model, which automatically lets the neural net to focus on a particular sub-region of the video that helps to classify multiple patterns and prevent the model from being misclassified. Moreover, we plan to extend this work towards a two-stream approach by considering both spatial and temporal features together instead of single streams.

## Acknowledgement

We thank the Pervasive Artificial Intelligence Research (PAIR) Labs support. The Consortium is funded by the Ministry of Science and Technology (MOST) (MOST 108-2634-F-008-002).

## References

[1] S. Hochreiter, J. Schmidhuber J, Long Short-Term Memory, *Journal Neural Computation*, Vol. 9, No. 8, pp. 1735-1780, December, 1997.

[2] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, *NIPS 2014 Workshop on Deep Learning*, Montreal, Canada, 2014, pp. 1-9.

[3] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, S. Vijayanarasimhan, Youtube-8m: A Large-scale Video Classification Benchmark, *arXiv,* 1609.08675, September, 2016.

[4] Y. G. Jiang, Z. Wu, J. Wang, X. Xue, S. F. Chang, Exploiting Feature And Class Relationships in Video Categorization with Regularized Deep Neural Networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40, No. 2, pp. 352-364, February, 2018.

[5] J. Y. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, G. Toderici, Beyond Short Snippets: Deep Networks for Video Classification, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, 2015, pp. 4694-4702.

[6] J. Donahue, L. Hendricks, M. Rohrbach, S.Venugopalan, S. Guadarrama, K. Saenkom, T. Darrell, Long-Term Recurrent Convolutional Networks for Visual Recognition and Description, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 4, pp. 677-691, April, 2017.

[7] K. Wu, Y. Yu, Automatic Object Extraction from Images Using Deep Neural Networks and The Level-set Method, *IET Image Processing*, Vol. 12, No. 7, pp.1131-41, February, 2018.

[8] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet Classification with Deep Convolutional Neural Networks, *Advances in Neural Information Processing Systems*, Vol. 60, No. 6, pp. 84-90, June, 2017.

[9] Y. LeCun, Y. Bengio, G. Hinton, Deep Leearning, *Nature*, Vol. 521, No. 7553, pp. 436-444, May, 2015.

[10] Y. M. Hirimutugoda, G. Wijayarathna, Image Analysis System for Detection of Red Cell Disorders Using Artificial Neural Networks, *Sri Lanka Journal of Bio-Medical Informatics*, Vol. 1, No. 1, January, 2010.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929-1958, June, 2014.

[12] Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going Deeper with Convolutions, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014, pp. 1-9.

[13] J. Duchi, E. Hazan, Y. Singer, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research*, Vol. 12, No. 7, pp. 2121-2159, January, 2011.

[14] G. Hinton, N. Srivastava, K. Swersk, *Neural Networks for Machine Learning Lecture 6a Overview of Mini-Batch Gradient Descent*, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional Architecture for Fast Feature Embedding, *Proceedings of the 22nd ACM International Conference on Multimedia*, Orlando, FL, USA, 2014, pp. 675-678.

[16] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, *arXiv,* 1412.3555, December, 2014.

[17] N. Léonard, S. Waghmare, Y. Wang, J. H. Kim, RNN: Recurrent Library for Torch, *arXiv,* 1511.07889, November, 2015.

[18] V. Mnih, N. Heess, A. Graves, Recurrent Models of Visual Attention, *Advances in Neural Information Processing Systems*, pp. 2204-2212, 2014.

[19] J. Ba, V. Mnih, K. Kavukcuoglu, Multiple Object Recognition with Visual Attention, *arXiv,* 1412.7755, December, 2014.

[20] S. Sharma, R. Kiros, R. Salakhutdinov, Action recognition using visual attention, *arXiv*, 1511.04119, November, 2015.

## Biographies

**S. P. Kasthuri Arachchi** is a Ph.D. candidate in National Central University, Taiwan. Her research interest includes deep learning, computer vision, and image processing. She received her Master degree in Computer Science from University of Peradeniya, Sri Lanka and was a former assistant lecturer in University of Kelaniya, Sri Lanka.

**Timothy K. Shih** is a Distinguished Professor at the National Central University, Taiwan. He is a Fellow of the Institution of Engineering and Technology (IET). He has received many research awards, including IIAS research award from Germany, HSSS award from Greece, Brandon Hall award from USA, the 2015 Google MOOC Focused Research Award.

**Chih-Yang Lin** received the Ph.D. degree in computer science and information engineering from National Chung-Cheng University. He joined Asia University from 2010 to 2017, where he was a professor and department chair of Bioinformatics and Medical Engineering. Currently, he is an associate professor with the Department of Electrical Engineering, Yuan-Ze University, Taiwan.

**Gamini Wijayarathna** is the Dean and a Senior Lecturer at Faculty of Computing and Technology, University of Kelaniya, Sri Lanka. He is a Resource person for National Education Institute (NIE) and Ministry of Education. Also Dr. Wijayarathna is a Software development consultant for private and public institutes in Sri Lanka.