RNN Model to Detect Fire in Videos

BITS F312 Neural Networks and Fuzzy Logic - Assignment 2

Team 9
Aryan Gupta - 2019A7PS0017G
Tarush Agarwal - 2019A7PS0025G
Tanmay Patil - 2019A7PS0054G
Shubham Kalantry - 2019A7PS0141G

Dataset and Data Pre-processing

Observations about Dataset:

- 60 fire and 60 not-fire RGB videos
  - Each video of different length, framerate, resolution
- Min #frames = 45, Max #frames = 6103

Pre-processing applied:

- Each video should have **equal number of frames**, (so as to train on a many to one neural network)
- Selected a smaller number of frames, **45 frames** (= min #frames)**,** from each video
- Frames selected uniformly
- The video was downscaled to **128x128** resolution, and then normalized
- Each video can be represented as a numpy array of shape **(45, 128, 128, 3)**

Model Architecture

- Convolutional Layers: to extract features from video frames
- LSTM Layer: for classifying and predicting based on time series data

Snapshot of Architecture code snippet for details:

- Density of layers, Activation functions, Strides in MaxPool layers, Regulariser term, Dropout value

```
inp = Input(x_train[0].shape)
# conv layers
convOut = Conv2D(16, (3, 3), padding='valid', activation='relu', data_format="channels_last")(inp)
convOut = MaxPool3D((1, 2, 2))(convOut)
convOut = Conv2D(8, (3, 3), padding='valid', activation='relu', data_format="channels_last")(convOut)
convOut = MaxPool3D((1, 2, 2))(convOut)

# flatten
flatOut = Reshape((seq_len, -1))(convOut)

# RNN
rnnOut = LSTM(512, return_sequences = False)(flatOut)

# dense layers
denseOut = Dense(256, activation='tanh', kernel_regularizer=tf.keras.regularizers.L2(0.01))(rnnOut)
denseOut = Dropout(0.2)(denseOut)
denseOut = Dense(32, activation='tanh', kernel_regularizer=tf.keras.regularizers.L2(0.01))(denseOut)
out = Dense(1, activation='sigmoid')(denseOut)

model = Model(inputs = inp, outputs = out)
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.001), loss = 'binary_crossentropy', metrics = ['accuracy'])
print(model.summary())
```

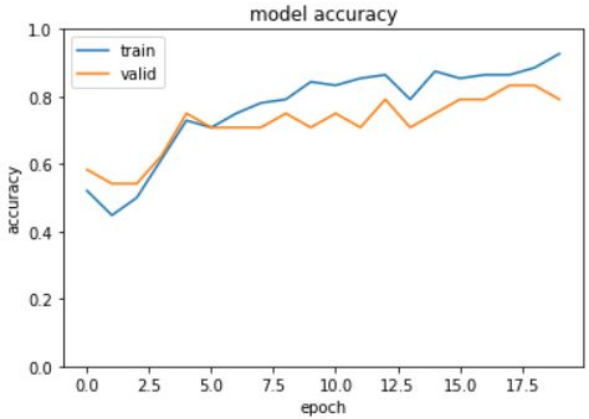
```
Model: "model_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_20 (InputLayer)        [(None, 45, 128, 128, 3)] 0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 45, 126, 126, 16)  448
_________________________________________________________________
max_pooling3d_30 (MaxPooling (None, 45, 63, 63, 16)    0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 45, 61, 61, 8)     1160
_________________________________________________________________
max_pooling3d_31 (MaxPooling (None, 45, 30, 30, 8)     0
_________________________________________________________________
reshape_19 (Reshape)         (None, 45, 7200)          0
_________________________________________________________________
lstm_19 (LSTM)               (None, 512)               15796224
_________________________________________________________________
dense_47 (Dense)             (None, 256)               131328
_________________________________________________________________
dropout_19 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_48 (Dense)             (None, 32)                8224
_________________________________________________________________
dense_49 (Dense)             (None, 1)                 33
=================================================================
Total params: 15,937,417
Trainable params: 15,937,417
Non-trainable params: 0
_________________________________________________________________
```

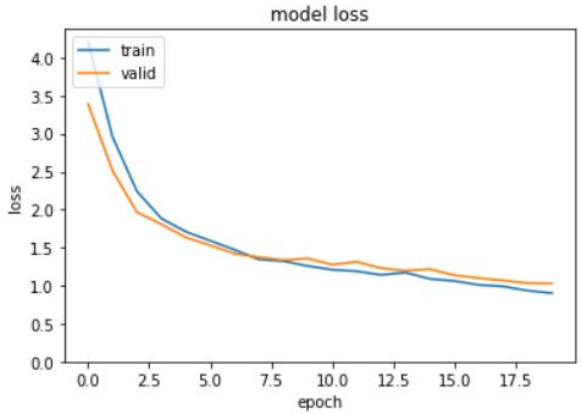
Training of the Model and Results

Optimiser: Adam optimiser
Callbacks: Reduce_lr_on_plateau and Early_stopping

```
history = model.fit(x_train, y_train, validation_data=(x_valid, y_valid), epochs=20, batch_size=2, callbacks=callbacks)
```

Plot of Accuracy and Loss curves:





Extra highlights

- Designed a CNN model similar to the one in the previous assignment
- Trained the model on individual frames as independent images
- Calculated the prediction over a few frames per video and classified the video according to the average over all predictions
- Performed very well with greater than 90% training, validation accuracy
- But, does not incorporate sequential analysis

Thank You!

Link to the NB: https://www.kaggle.com/code/aryangupta8501/grp-9-assign-2-lstm?scriptVersionId=81622678