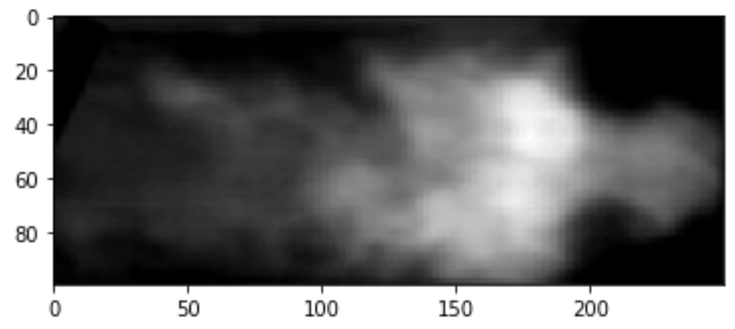
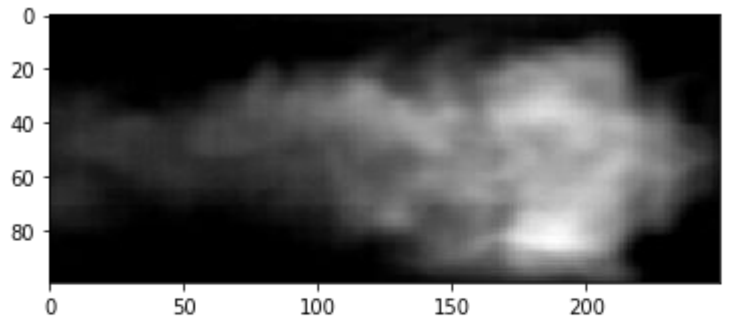
Time series Combustion image classification using Convolutional LSTM

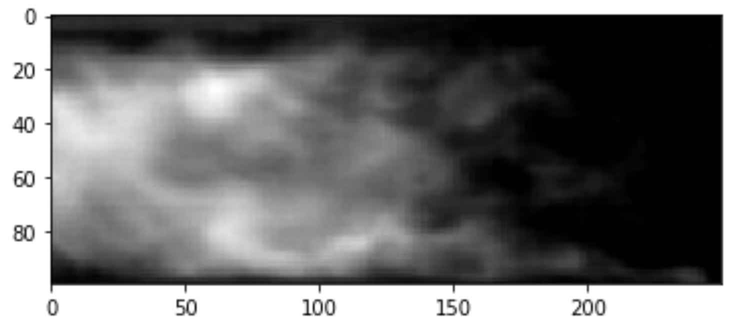
> Image Group 1

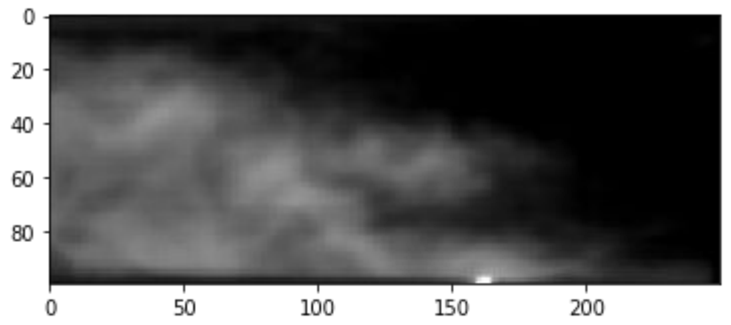
Jet propulsion Engine Combustion Image Dataset

Dataset is a collection of 81000 time series images of the combustion inside a Jet propulsion Engine. Labeled as **‘stable’** and **‘unstable’.**



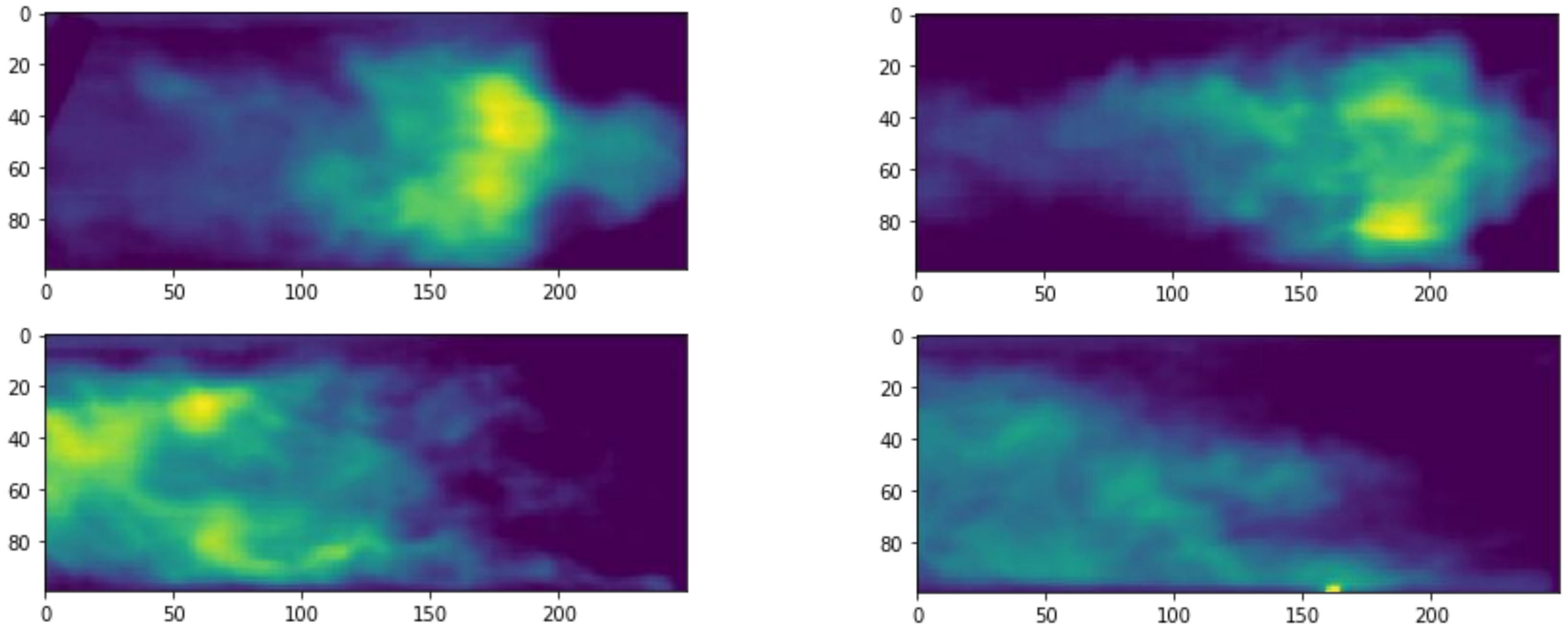






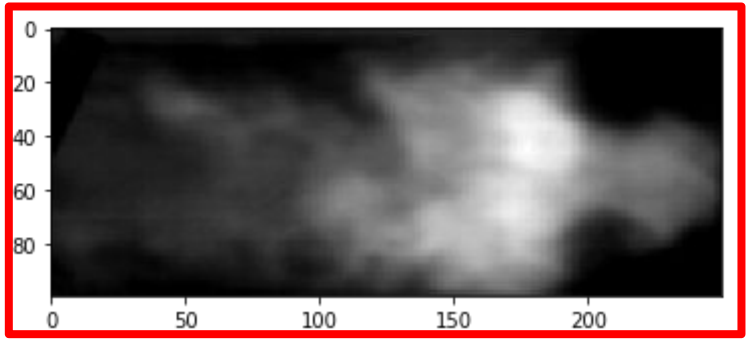
Jet propulsion Engine Combustion Image Dataset

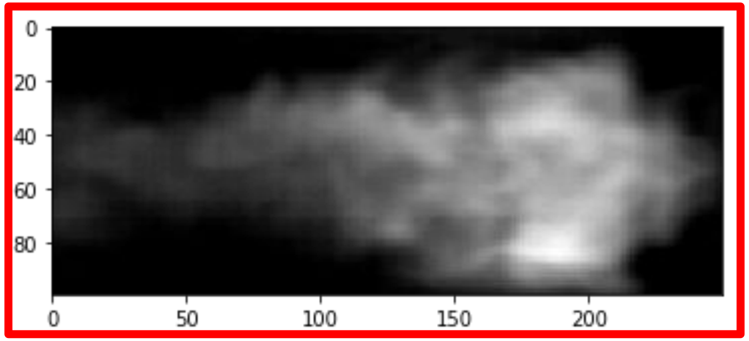
Dataset is a collection of 81000 time series images of the combustion inside a Jet propulsion Engine. Labeled as **‘stable’** and **‘unstable’.**

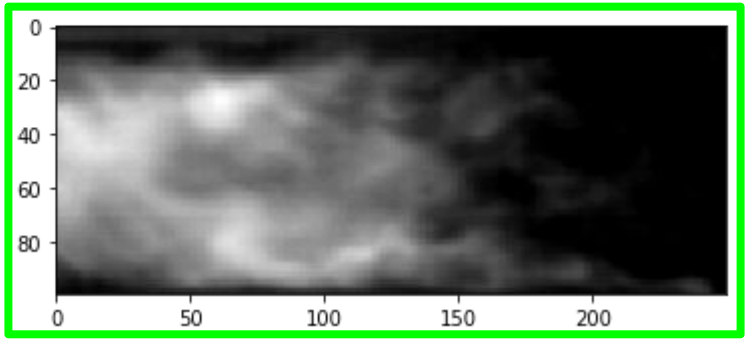


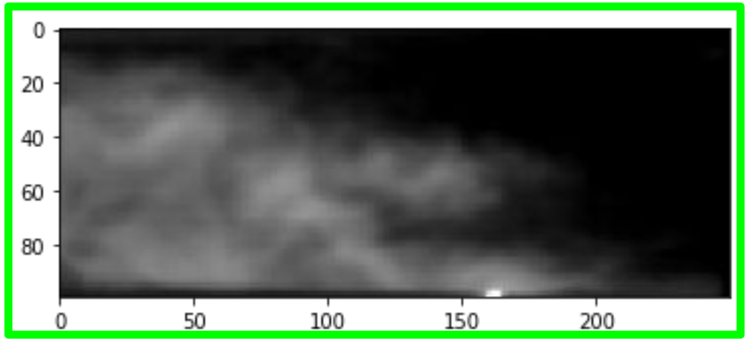
Jet propulsion Engine Combustion Image Dataset

Dataset is a collection of 81000 time series images of the combustion inside a Jet propulsion Engine. Labeled as **‘stable’** and **‘unstable’.**









Pre-processing

- Each original image is 100x250. We use 54000 of such images.
- We resize each image to the resolution of 32x64 (or similar) using bicubic interpolation in opencv.
- Further, we divide the dataset in Training (80% of the images) and Testing data (20% of the images).

For Convolutional LSTM model, we combine few images together to create one instance of the data. Number of images clubbed together is equal to `‘n_frames’`. It can also be called the strides for our LSTM network.

The challenge

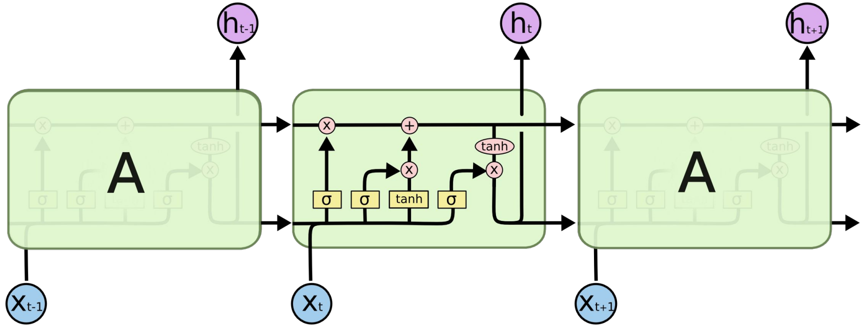
To classify each image as ‘stable’ or ‘unstable’.

- we need to use both the **temporal data (time series)** and the **spatial structure** (from 2D images) for our classification.

LSTM Network

- Long Short Term Memory (LSTM) networks are a special kind of Recurrent Neural Networks (RNN), capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem of RNNs.

- A typical LSTM network is comprised of different memory blocks called cells. There are two states that are being transferred to the next cell; the cell state and the hidden state. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called i) forget (ft), ii) input (it), iii) output (ot) gates.



$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f)\\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i)\\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o)\\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)\\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

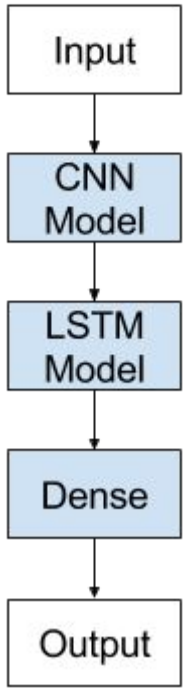
References:

1. *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*
2. *https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/*

Combination of Convolution and LSTM Network

This architecture is appropriate for problems that:

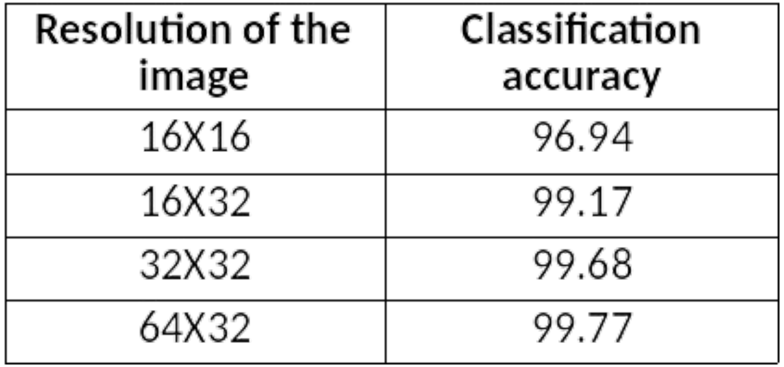
- Have spatial structure in their input such as the 2D structure or pixels in an image or the 1D structure of words in a sentence, paragraph, or document.
- Have a temporal structure in their input such as the order of images in a video or words in text, or require the generation of output with temporal structure such as words in a textual description.



References:
https://machinelearningmastery.com/cnn-long-short-term-memory-networks/

Keras implementation of Combination of Convolution and LSTM Network + Results (on test data)

```
model = Sequential()
#Convolution
model.add(Conv2D(input_shape= (num_features,1),filters=32, kernel_size=3,
padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
#LSTM
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())
```

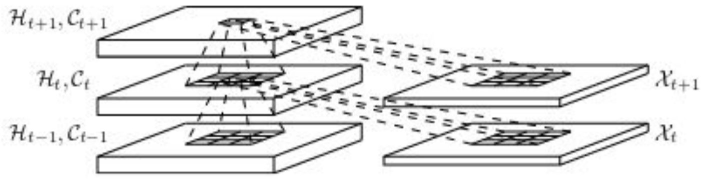


Resolution of the image	Classification accuracy
16X16	96.94
16X32	99.17
32X32	99.68
64X32	99.77

Convolutional LSTMs

- The major drawback of FC-LSTM in handling spatiotemporal data is its usage of full connections in input-to-state and state-to-state transitions in which no spatial information is encoded.

- A distinguishing feature of ConvLSTMs is that all the inputs X1,...,Xt, cell outputs C1,...,Ct, hidden states H1,...,Ht, and gates it,ft,ot of the ConvLSTM are 3D tensors whose last two dimensions are *spatial dimensions* (rows and columns). To get a better picture of the inputs and states, we may imagine them as vectors standing on a spatial grid. The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. This can easily be achieved by using a convolution operator in the state-to-state and input-to-state transitions (Fig below).


$\mathcal{H}_{t+1}, \mathcal{C}_{t+1}$
$\mathcal{H}_t, \mathcal{C}_t$
$\mathcal{H}_{t-1}, \mathcal{C}_{t-1}$
$\mathcal{X}_{t+1}$
$\mathcal{X}_t$

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned}
$$

Reference:
Xingjian, S. H. I., et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." Advances in neural information processing systems. 2015.

Keras implementation of Convolutional LSTM Network

```
seq = Sequential()
seq.add(ConvLSTM2D(filters=n_filters, kernel_size=(3, 3),
                   input_shape=(None,n_rows,n_cols,1),
                   padding='same', return_sequences=True))
seq.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))
seq.add(ConvLSTM2D(filters=n_filters, kernel_size=(3, 3),
                   input_shape=(None,n_rows,n_cols,1),
                   padding='same', return_sequences=True))

seq.add(TimeDistributed(MaxPooling2D((2, 2), strides=(2, 2))))

seq.add(TimeDistributed(Reshape((-1,))))
seq.add(TimeDistributed(Dense(1, activation='sigmoid')))
seq.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(seq.summary())
```

```
_________________________________________________________________
Layer (type)                          Output Shape              Param #
=================================================================
conv_lst_m2d_25 (ConvLSTM2D) (None, None, 128, 128, 6)          1536
_________________________________________________________________
time_distributed_29 (TimeDis         (None, None, 64, 64, 6)        0
_________________________________________________________________
conv_lst_m2d_26 (ConvLSTM2D)  (None, None, 64, 64, 6)           2616
_________________________________________________________________
time_distributed_30 (TimeDis          (None, None, 32, 32, 6)       0
_________________________________________________________________
time_distributed_31 (TimeDis          (None, None, 6144)             0
_________________________________________________________________
time_distributed_32 (TimeDis            (None, None, 1)            6145
=================================================================
Total params: 10,297
Trainable params: 10,297
Non-trainable params: 0
_________________________________________________________________
```

Training of dataset and hyper-parameter optimization in Convolutional LSTM:

Feature		Efficiency
Number of filters	4	81.16
	6	87.69
	8	89.33
Number of frames	5	98.7
	10	92.34
	20	86

Hyper parameters:

- Number of filters
- Number of frames

Resolution		Efficiency
Width	Height	
16	16	98.12
16	32	98.8
32	16	98.95
32	32	99.78

Fixed parameters:

- Number of channels = 1
- Number of rows = height of the resized image
- Number of columns = width of the resized image
- Number of train samples = 80% of train data/number of frames
- Number of test samples = 20% of train data/number of frames

Depthwise separable convolution

Depthwise separable convolutions reduce the number of parameters and computation used in convolutional operations while increasing representational efficiency.

Commonly called “separable convolution” in deep learning frameworks, it consists in a depthwise convolution, i.e. a spatial convolution performed independently over each channel of an input, followed by a pointwise convolution, i.e. a 1x1 convolution, projecting the channels output by the depthwise convolution onto a new channel space.

For example, if we have 3x3 convolutional layer on 16 input channels and 32 output channels, then we traverse the 16 channels with 1 3x3 kernel each, giving us 16 feature maps. Now, we traverse these 16 feature maps with 32 1x1 convolutions each and then start to add them together.

This results in 656 (16x3x3 + 16x32x1x1) parameters opposed to the 4608 (16x32x3x3) parameters from regular convolution.

References:

1. *Chollet, Francois. "Xception: Deep Learning With Depthwise Separable Convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.*
2. *https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d*

Keras implementation of Depthwise separable Convolutional Network

```
x = Conv2D(32, (3, 3), strides=(2, 2), use_bias=False)(img_input)

   x = Activation('relu')(x)
   x = MaxPooling2D((3, 3), strides=(2, 2))(x)
   x = SeparableConv2D(128, (3, 3), use_bias=False)(x)
   x = Activation('relu')(x)
   x = SeparableConv2D(128, (3, 3), use_bias=False)(x)
   x = MaxPooling2D((3, 3), strides=(2, 2))(x)
   x = SeparableConv2D(256, (3, 3), use_bias=False)(x)
   x = Activation('relu')(x)
   x = SeparableConv2D(256, (3, 3), use_bias=False)(x)
   x = MaxPooling2D((3, 3), strides=(2, 2))(x)
   x = GlobalAveragePooling2D(name='avg_pool')(x)
   x = Dense(classes, activation='softmax', name='predictions')(x)

   # Create model.
   model = Model(img_input, x)
```

Keras implementation of Depthwise separable Convolutional Network - Results

Testing data/Training data = 0.4 in this case.

Input image shape: (32400, 100, 100, 1)
Epoch 1/1
32400/32400 [==============================] - **54s** - loss: 0.0867 - acc: 0.9618
Accuracy: 99.75%

Time taken for training is less compared to ConvLSTM (~2 mins), even for very large network.