UNIVERSITÉ CÔTE D'AZUR

Inria

VIDEO Classification

Hands ON

Snehashis MAJHI

Email: snehashis.majhi@inria.fr

Ph.D. Candidate @STARS Team INRIA

Collaboration with TOYOTA Motor Europe

TABLE OF CONTENT

- 2D CNN Feature Extraction from Pre-Trained Image Model
- LSTM and It’s Variants
- Video Classification with 2D CNN with LSTM/GRU
- Video Classification with Two-Stream 2D CNN
- Video Classification with I3D 3DCNN

Extracting 2D CNN Features::

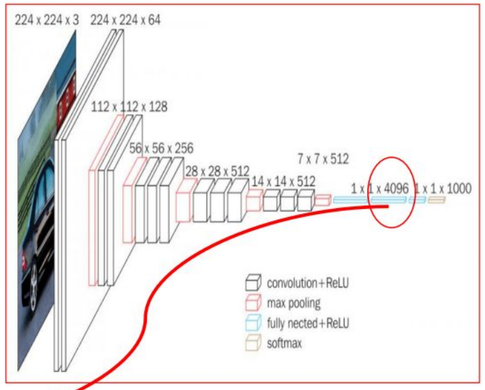
Implementation

Extracting 2D CNN features from a pre-trained model

```
from keras.applications.vgg16 import VGG16
from tensorflow.keras.utils import load_img, img_to_array
from keras.applications.vgg16 import preprocess_input
import numpy as np

model = VGG16(weights='imagenet', include_top=True)
model = Model(inputs=model.input, outputs=model.get_layer('fc2').output)

def feature_extraction(img_path):
        img = image.load_img(img_path, target_size=(224, 224))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        x = preprocess_input(x)
        return x
```



Processing a video

```
video_path = 'path to the video'
image_files = os.listdir(video_path)
features = []
for image in image_files:
        features.append(feature_extraction
        (os.path.join(video_path, image)))
```

Perform max-min pooling on the frame-level features

```
import numpy as np
import os
path = "../results/frame_features/"

def max_min_conv(video):
   frame_features  = np.loadtxt(video, delimiter=',')
   max_features = np.amax(frame_features, axis=0)
   min_features = np.amin(frame_features, axis=0)
   final_t1 = np.hstack([max_features, min_features])
   return final_t1

for video in os.listdir(path):
   desc = []
   video_descriptor = max_min_conv(os.path.join(path, video))
   desc = np.hstack([desc, video_descriptor.ravel()])
   np.savetxt('../results/video_descriptors/'+video, desc, delimiter=',')
```

Let's Try on Google Colab
https://colab.research.google.com/drive/1cmeK311FhfeEUHMQjO3poO1zHMQv_dGw?usp=sharing

LSTM and It's Variants ::

Let's implement a single layer LSTM of 3 time steps for time forecasting problem.

Data		Predict	
X,	y	X,	y
10, 20, 30	40	70, 80, 90	??
20, 30, 40	50		
30, 40, 50	60		

Let's Try on Google Colab
https://colab.research.google.com/drive/1KsZsohKMPReksZAkDtLFK0p5mdF5RfdK?usp=sharing

LSTM Variants :

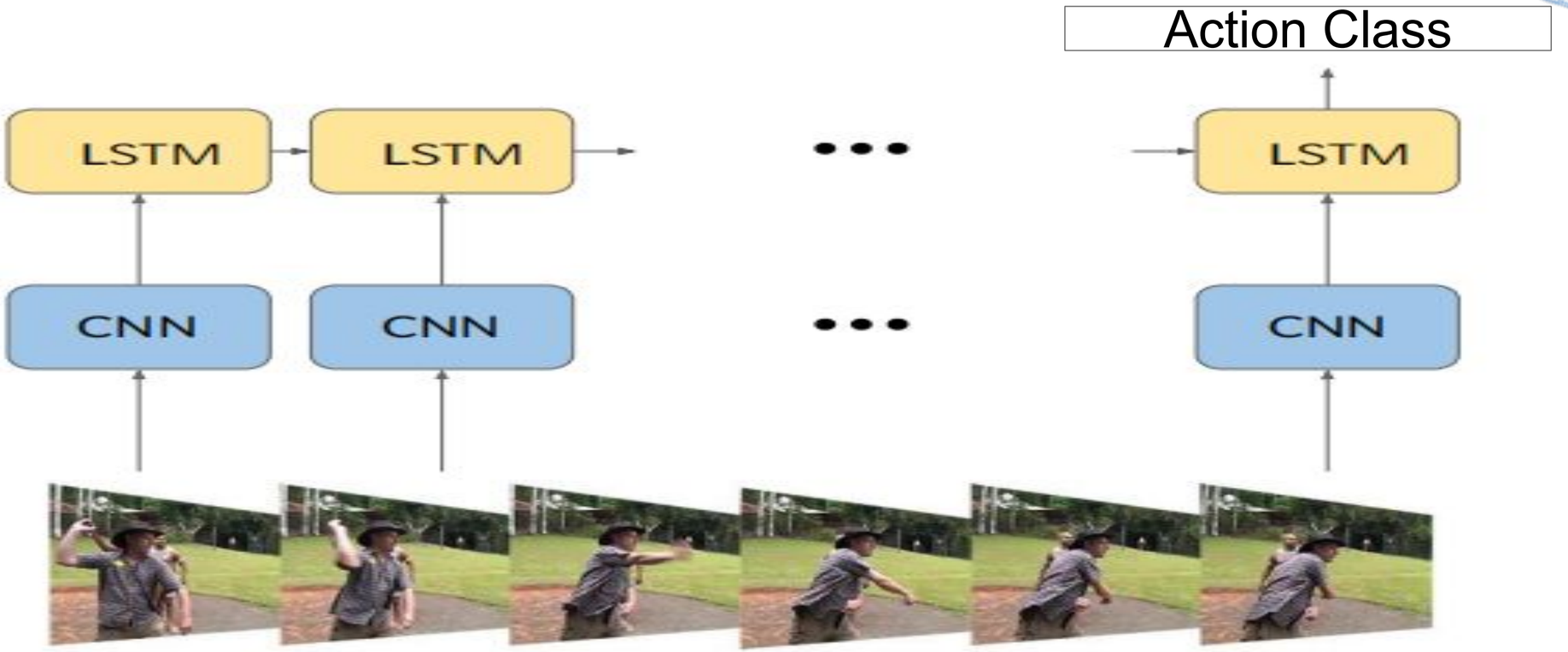
- **Stacked LSTM –** Stacking LSTMs layers
- **Bi-directional LSTM –** To model temporal information both forward and backward

Let's Try on Google Colab
https://colab.research.google.com/drive/1KsZsohKMPReksZAkDtLFK0p5mdF5RfdK?usp=sharing

- **CNN LSTM –** To model temporal information on high level spatial features extracted from CNN
- **ConvLSTM –** The convolutional operation is embedded in each LSTM cell

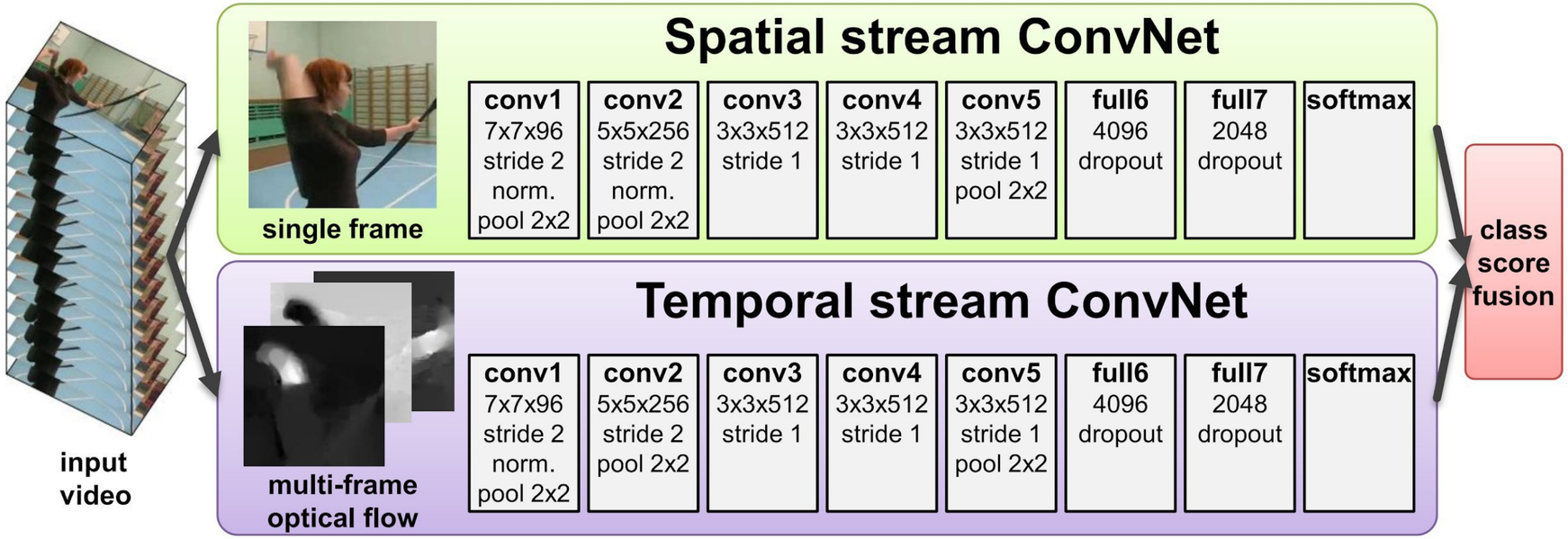
Let's Try on Google Colab
https://colab.research.google.com/drive/1TRuHaLJbkqqLpbCn8E2J8Ky1qEdCTC64?usp=sharing

Video Classification With Two-Stream 2D CNN ::



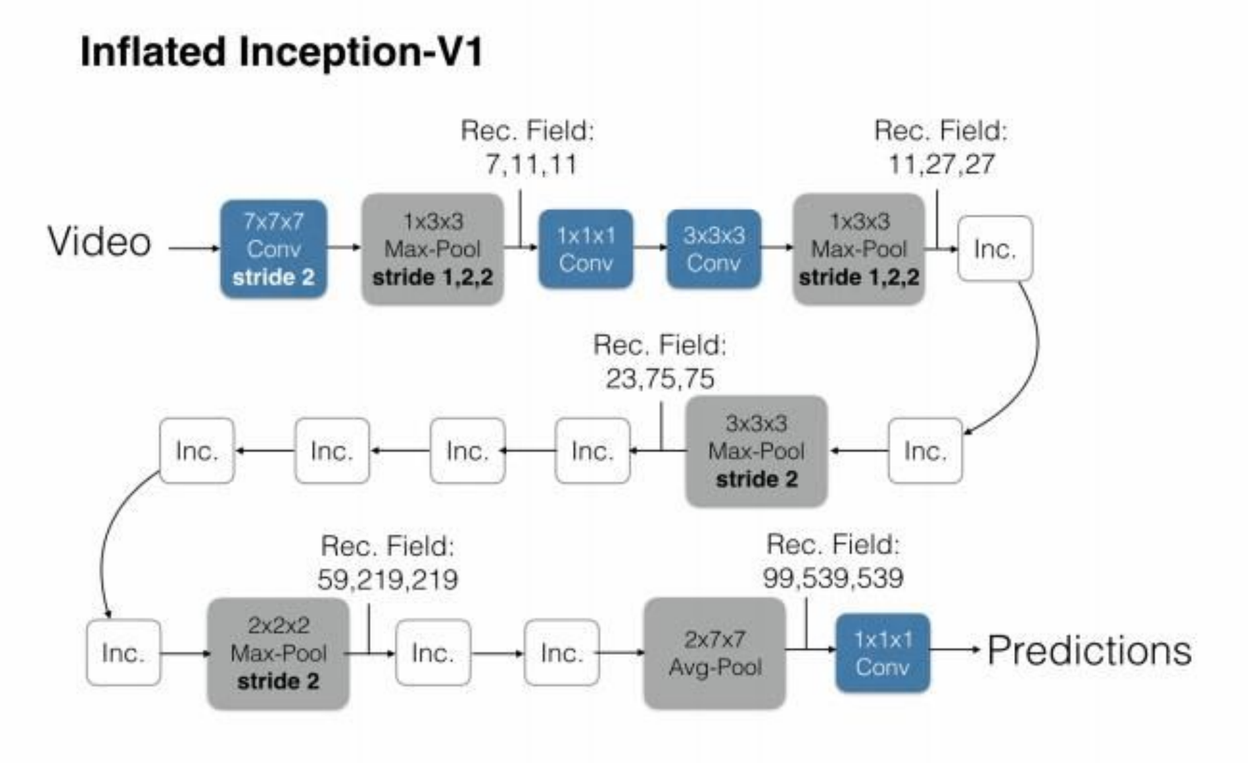
Let's Try on Google Colab
https://colab.research.google.com/github/sayakpaul/Action-Recognition-in-TensorFlow/blob/main/Video_Classification.ipynb#scrollTo=eIuAlLO-jQpy

Video Classification With 2D CNN +LSTM/GRU ::



Let's Try on Google Colab
https://colab.research.google.com/drive/1C8gPsD_sJlxNj1v4Z5kQDifhkTeTgEmY?usp=sharing#scrollTo=KzBVSjDiChk1

3. Video Classification with I3D Network ::

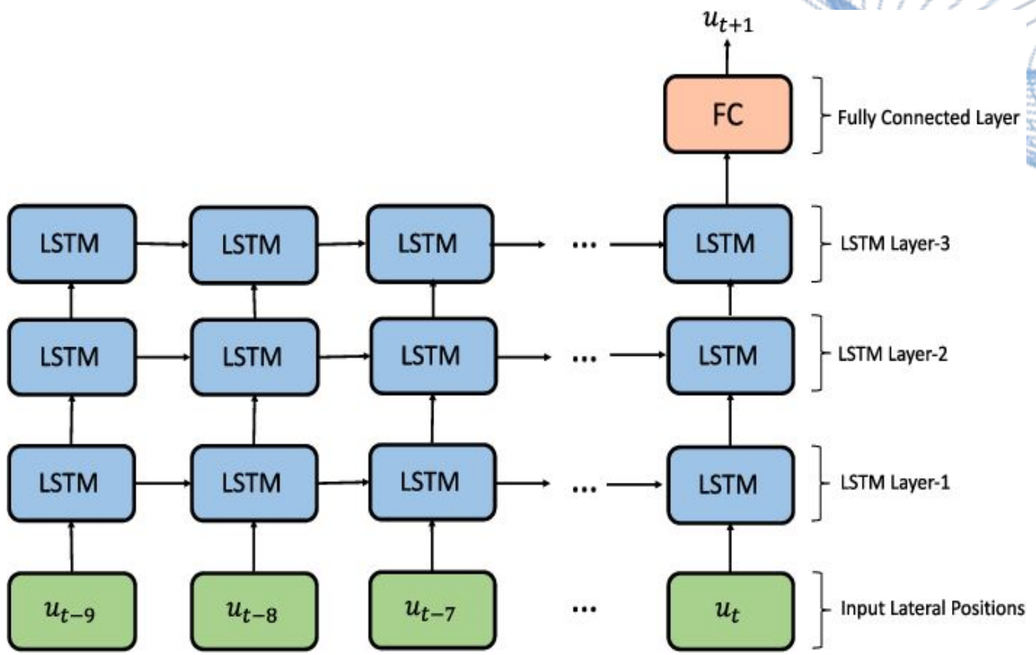


Let's Try on Google Colab
https://colab.research.google.com/drive/1M5Hj2tqBL0L2sDDzOPM_U0OzCiotqv29?usp=sharing

APPENDIX

1. Stacked LSTM ::

Multiple hidden LSTM layers can be stacked one on top of another in what is referred to as a Stacked LSTM model.

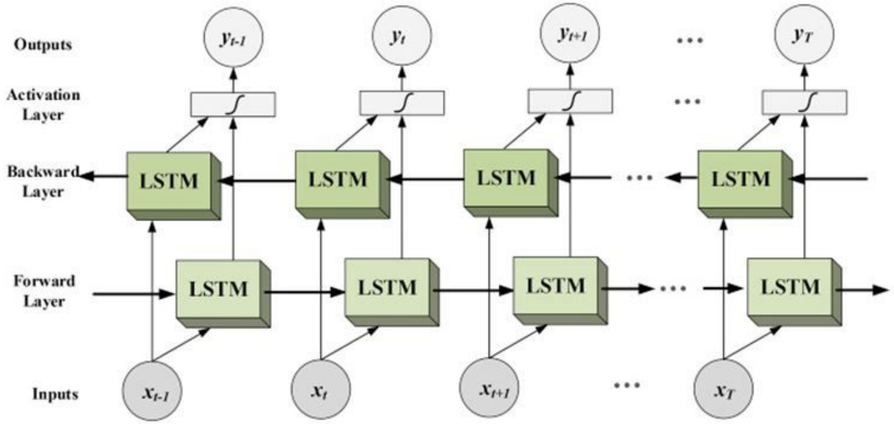


2. Bidirectional LSTM ::

On some sequence prediction problems, it can be beneficial to allow the LSTM model to learn the input sequence both forward and backwards and concatenate both interpretations.

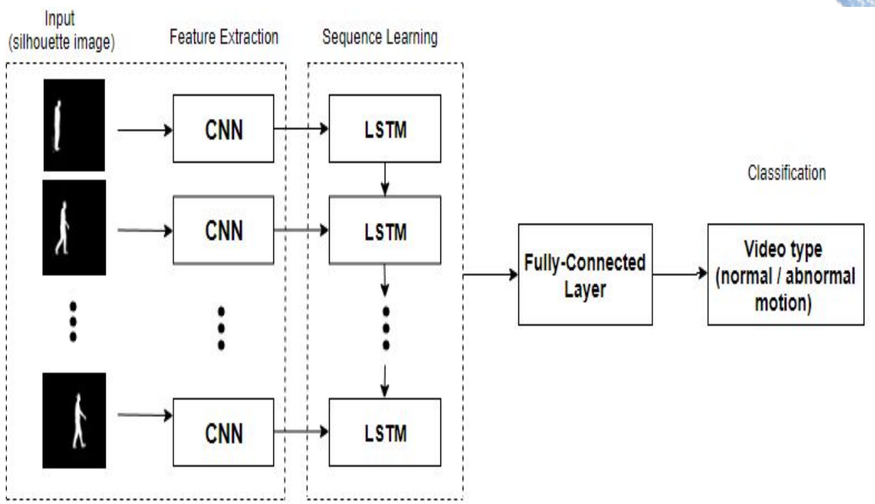
This is called a Bidirectional LSTM.

Implementation
We can implement a Bidirectional LSTM for univariate time series forecasting by wrapping the first hidden layer in a wrapper layer called Bidirectional.



3. CNN LSTM ::

A CNN model can be used in a hybrid model with an LSTM backend where the CNN is used to interpret subsequences of input that together are provided as a sequence to an LSTM model to interpret. This hybrid model is called a CNN-LSTM.



4. ConvLSTM ::

A type of LSTM related to the CNN-LSTM is the ConvLSTM, where the convolutional reading of input is built directly into each LSTM unit. The ConvLSTM was developed for reading two-dimensional spatial-temporal data.