Deep Learning for Computer Vision

Farhood NEGIN

Research scientist

INRIA Sophia Antipolis

Outline: Video Classification

- Introduction to videos
- Traditional video processing using CNNS
- RNNs (specifically LSTMs)
- Implementing LSTMs

Why video analysis?

Data:

BBC Motion Gallery

~2.5 Billion new images / month

TV-channels recorded since 60's

~5K image uploads every min.

>34K hours of video upload every day

~30M surveillance cameras in US
=> ~700K video hours/day

And even more with future wearable devices

Why video analysis?

Applications:

First appearance of N. Sarkozy on TV

Sociology research: Influence of character smoking in movies

Education: How do I make a pizza?

Where is my cat?

Predicting crowd behavior
Counting people

Motion capture and animation

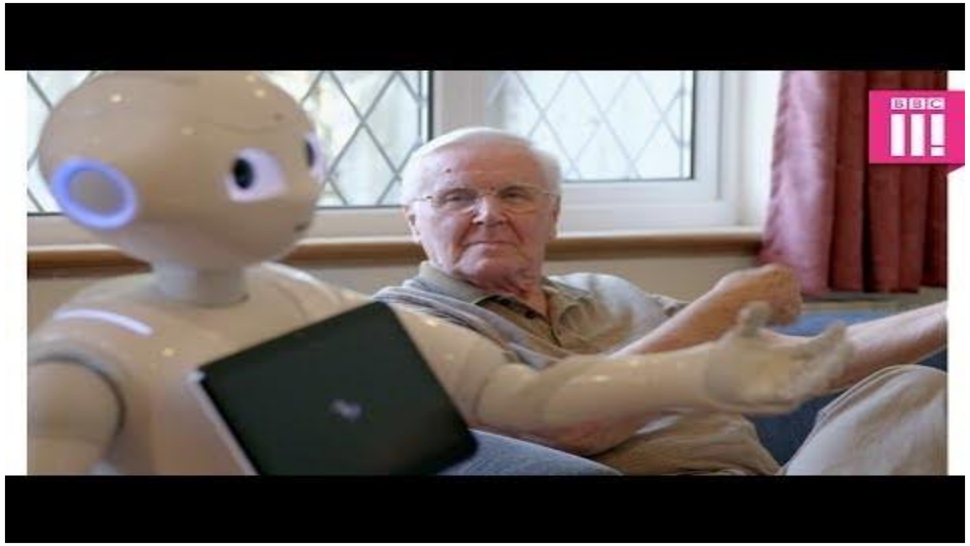
Why video analysis?

Applications:

Unconstrained video search

Why video analysis?

Amazon go



Assistive Robot

Waiter Robot!

Introduction to videos

- A video is a sequence of frames captured over time
- Now our image data is a function of space (x, y) and time (t)

Challenge is how to model time?

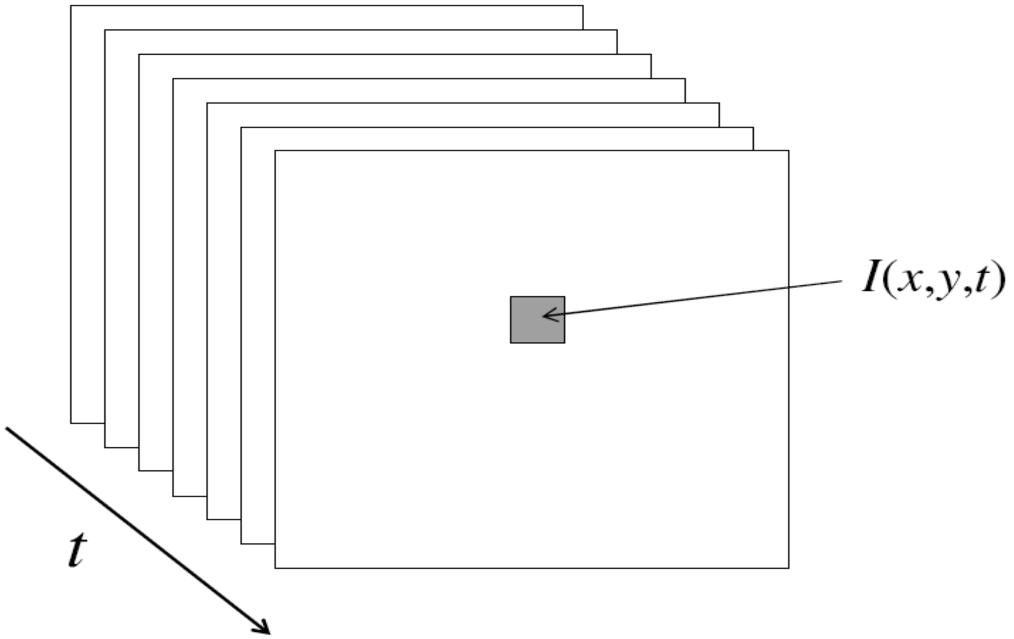
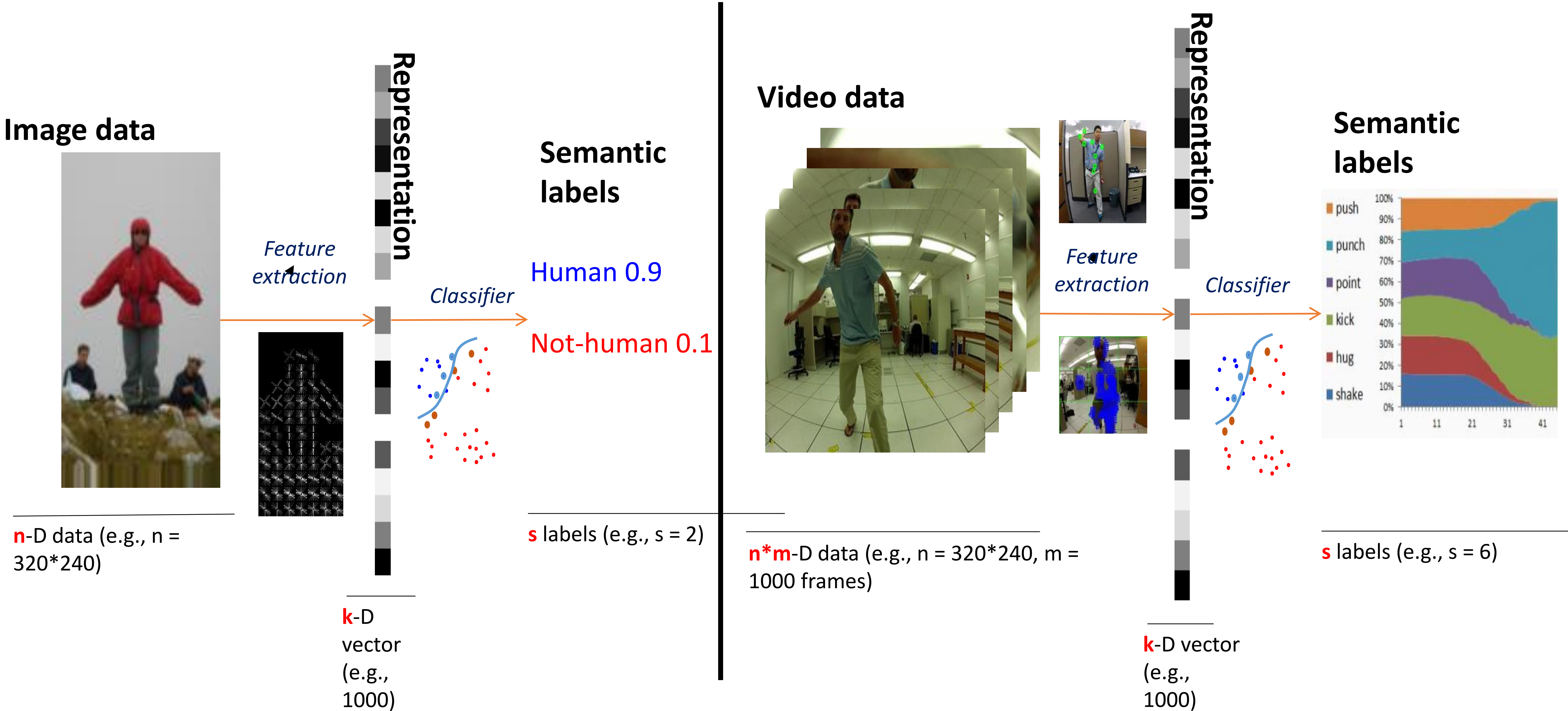


Image Vs Video Classification Networks

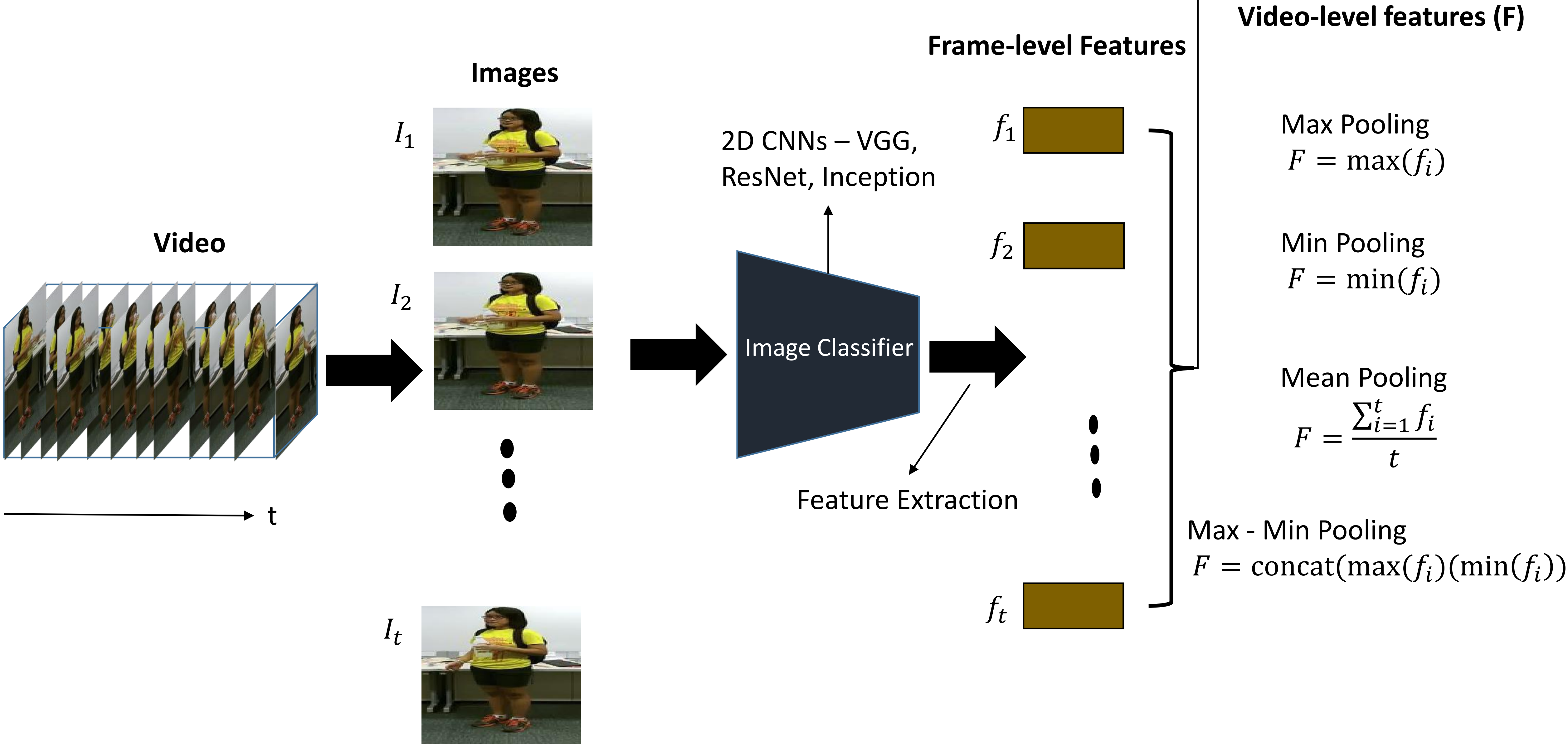


Video Classification Techniques

- Frame-level aggregation
  - Aggregating the frame-level information using pooling
  - Temporal information is lost
- Recurrent Neural Networks
  - Model the temporal evolution of the frames using gating functions
  - Does not handle space-time simultaneously
- 3D Convolutional Networks
  - Perform convolution across space-time simultaneously
  - Too rigid to capture subtle information

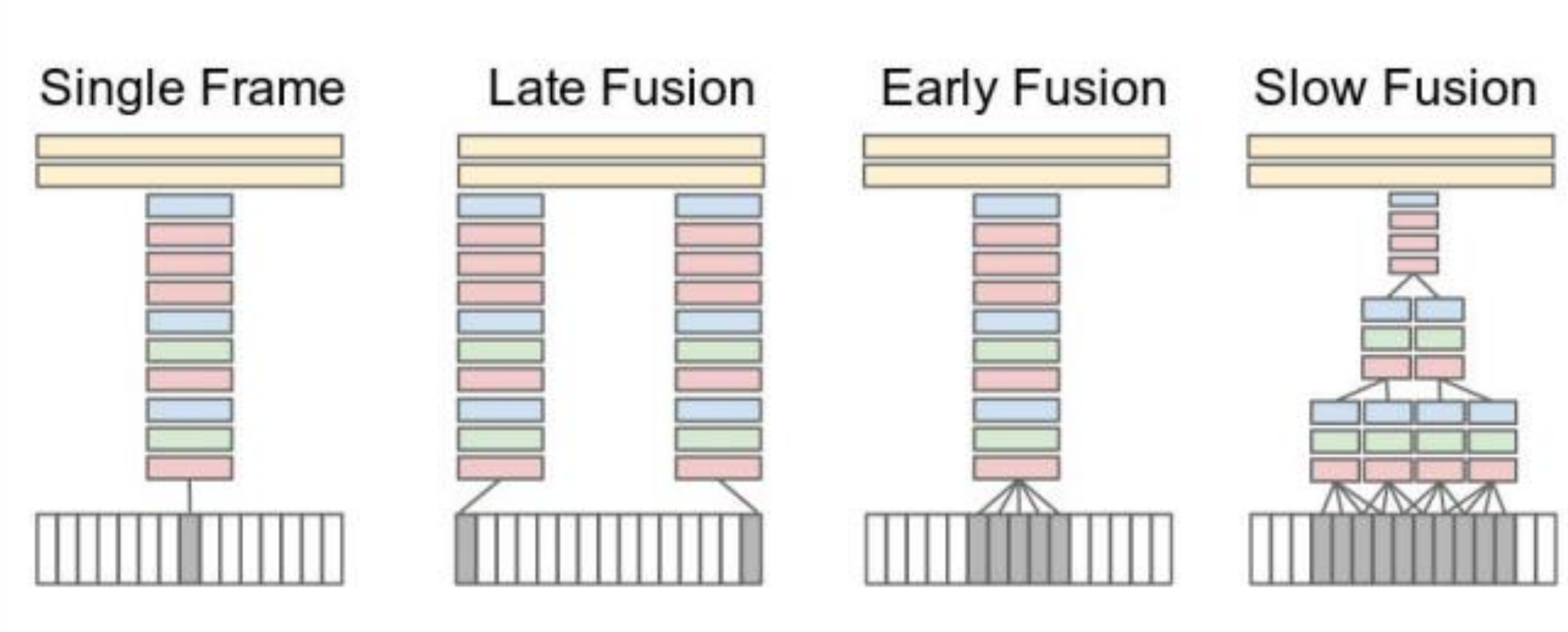
For handling time

Frame-level Aggregation



Frame-level Aggregation

- Temporal connectivity pattern?



Frame-level Aggregation

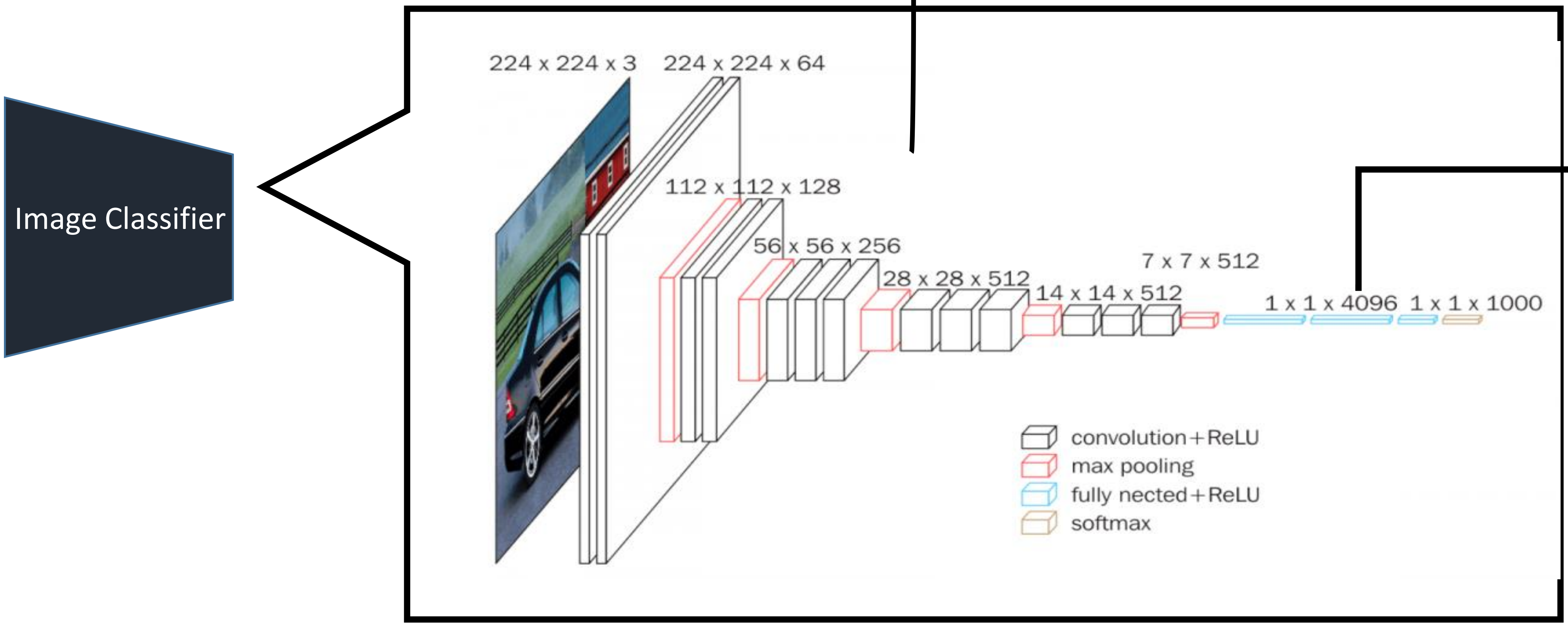
- These frame-level pooling mechanisms provide a video descriptor which focuses on the salient instances in the video.

- The video descriptors for each video are treated as data samples for a classifier (like SVM) for classifying the videos.

Frame-level Aggregation

Pre-trained on ImageNet

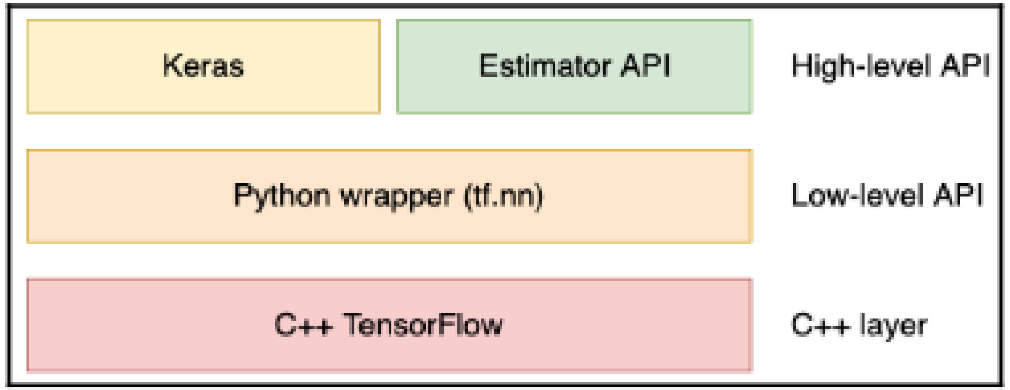
How do you extract the frame-level features?



Extract feature from
Fully-connected layer
(FC-2)

Implementation

All the practicals will be in Keras with Tensorflow in the back-end.



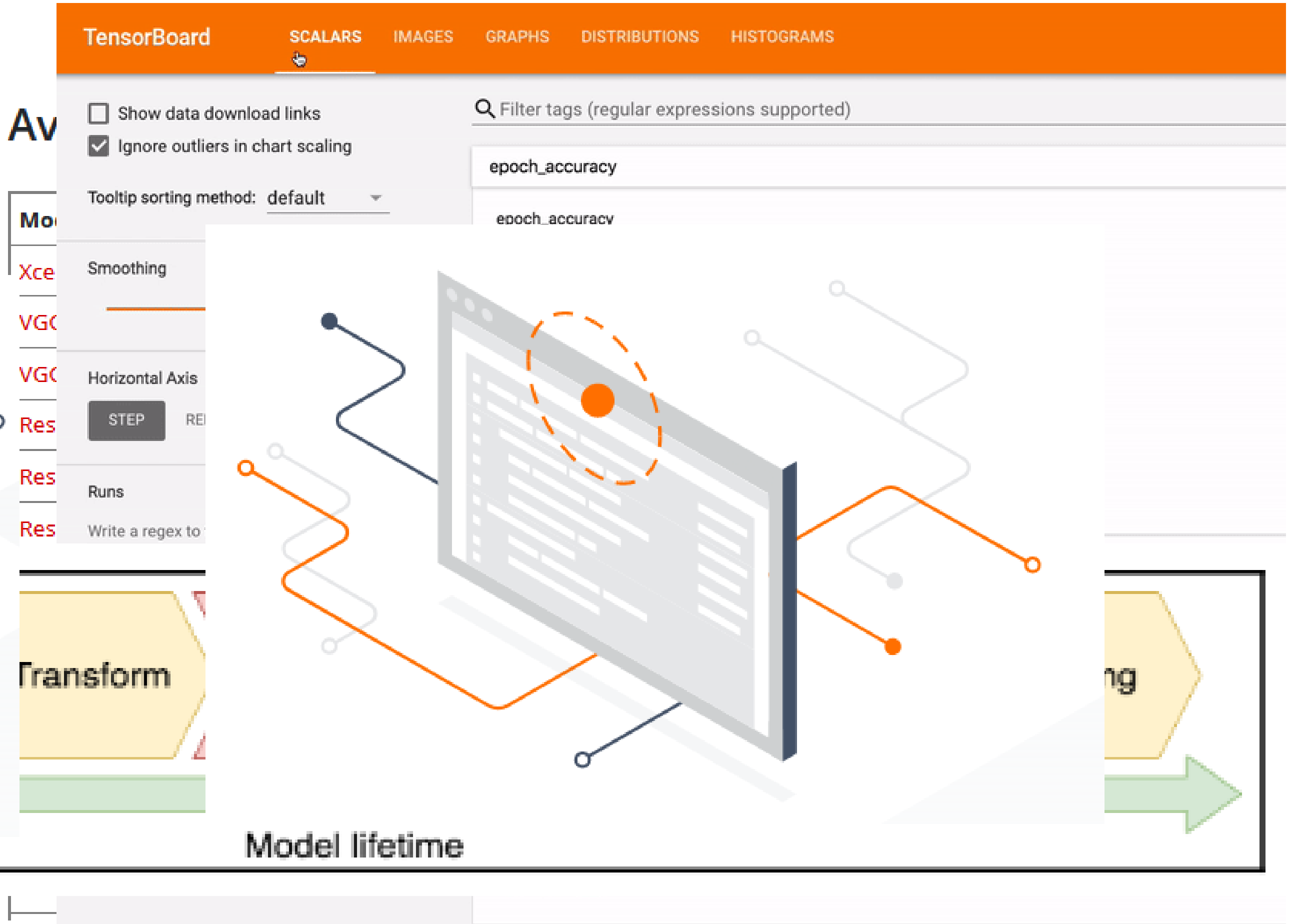
Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

TensorFlow Ecosystem: A Brief Introduction

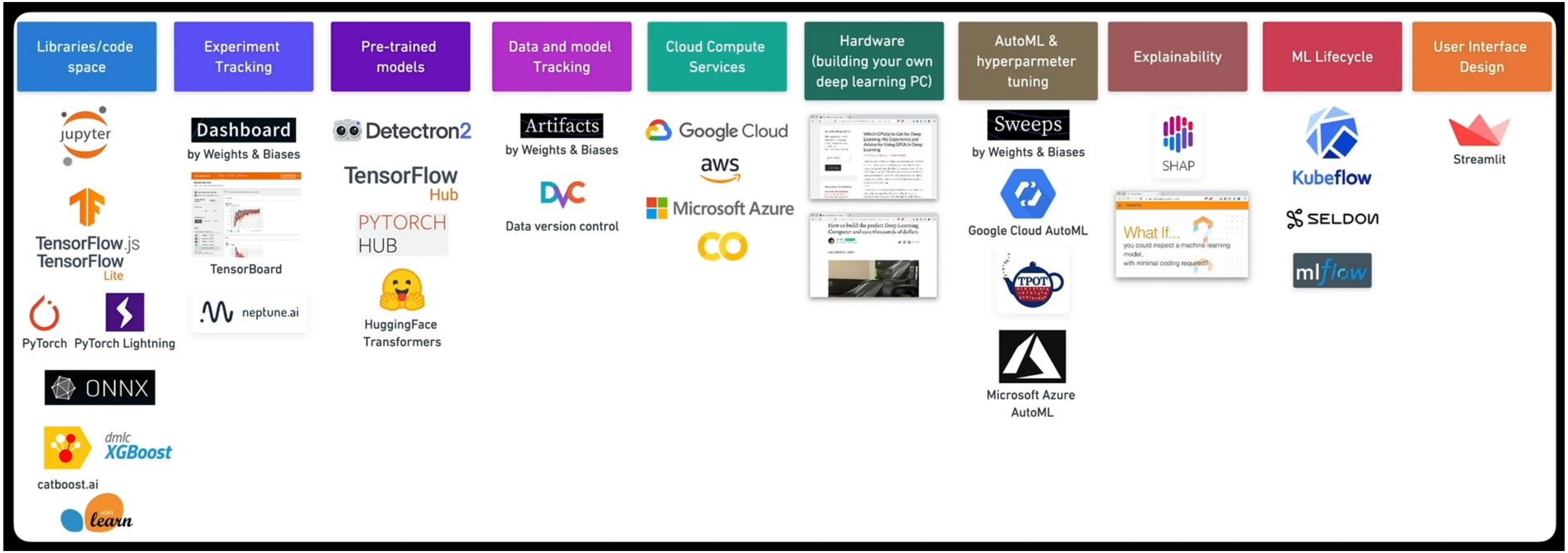
- Keras Applications
- TensorBoard
- TensorFlow Add-ons
    - Plot any metric (such as accuracy)
    - Display input and output images
    - Display the execution time
    - Draw your model's graph representation
    - TensorFlow Data validation
    -
    -
    -
- TensorFlow Extended (TFX)
- TensorFlow Lite and TensorFlow.j

```
callbacks = [tf.keras.callbacks.TensorBoard('./logs_keras')]
model.fit(x_train, y_train, epochs=5, verbose=1, validation_data=(x_test, y_test),
          callbacks=callbacks)
$ tensorboard --logdir ./logs_keras
```

```
from tensorflow.keras.applications.resnet50 import ResNet50
model = ResNet50(weights='imagenet')
```

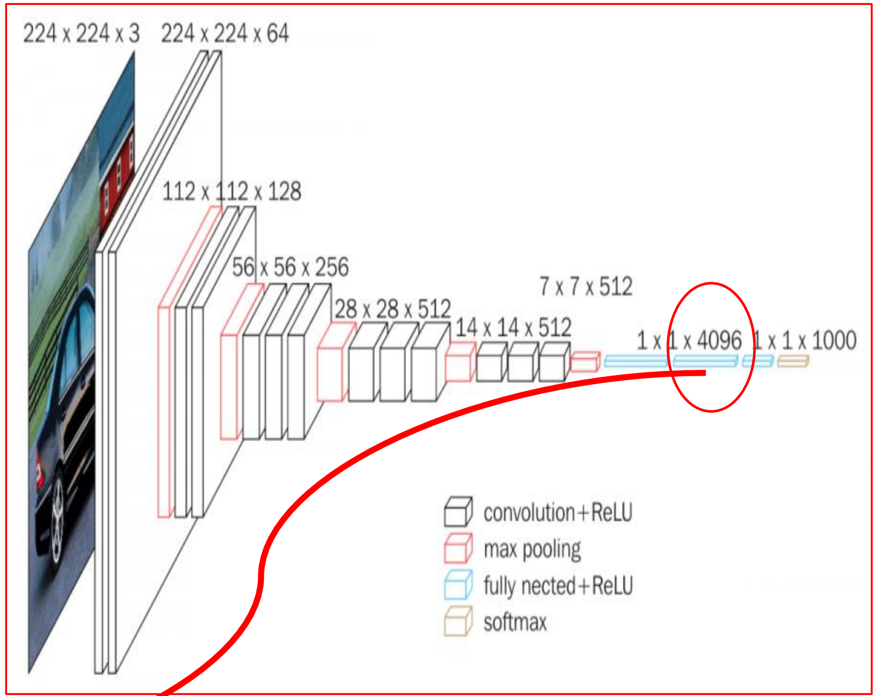


Toolboxes



Some demos at the end…

Implementation



Extracting 2D CNN features from a pre-trained model

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
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)))
```

Implementation

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

Disadvantages

- These video descriptors do not model temporal information and only relies on the salient frame-level features.

- Then how should we model temporal information???

Time for a short break may be …….

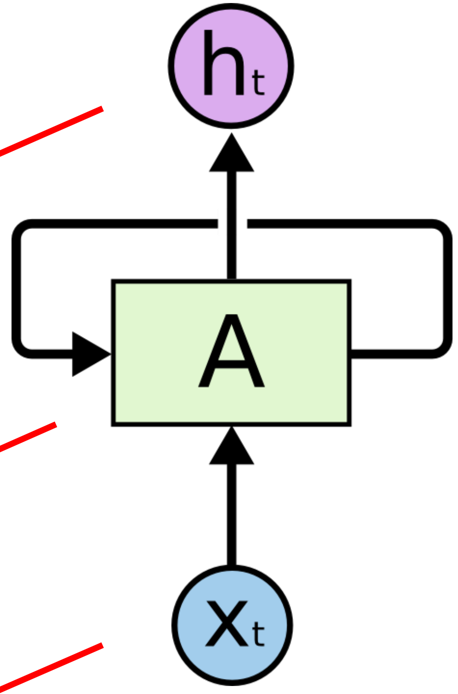
Recurrent Neural Networks (RNNs)

- Humans don’t start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don’t throw everything away and start thinking from scratch again.

- Traditional neural networks can’t do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It’s unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent Neural Networks (RNNs)

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.



Outputs a value at time *t*
(State)

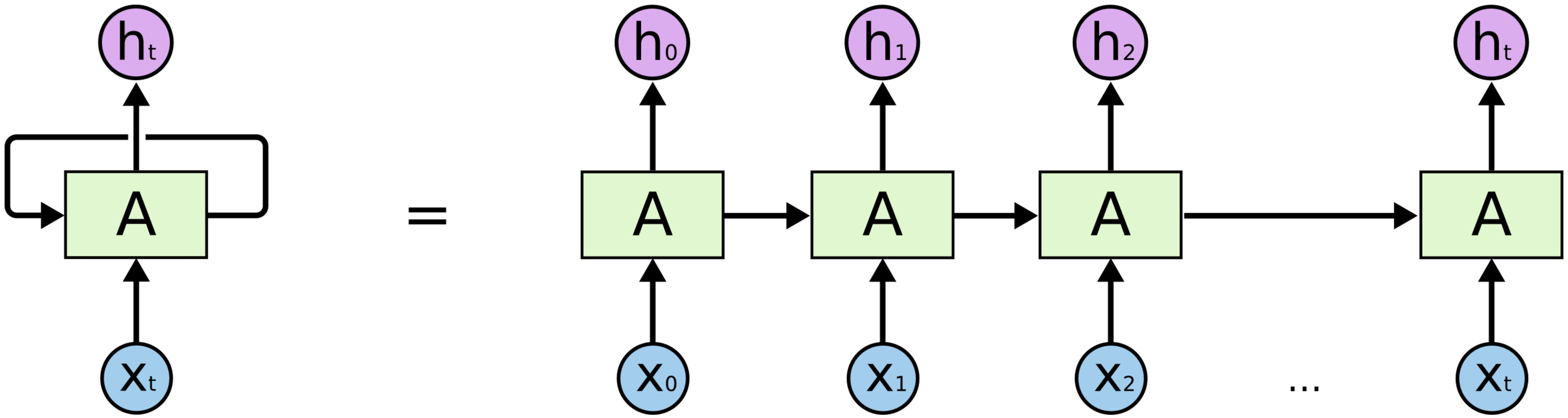
RNN Cell

Input at time *t*

RNN

A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.

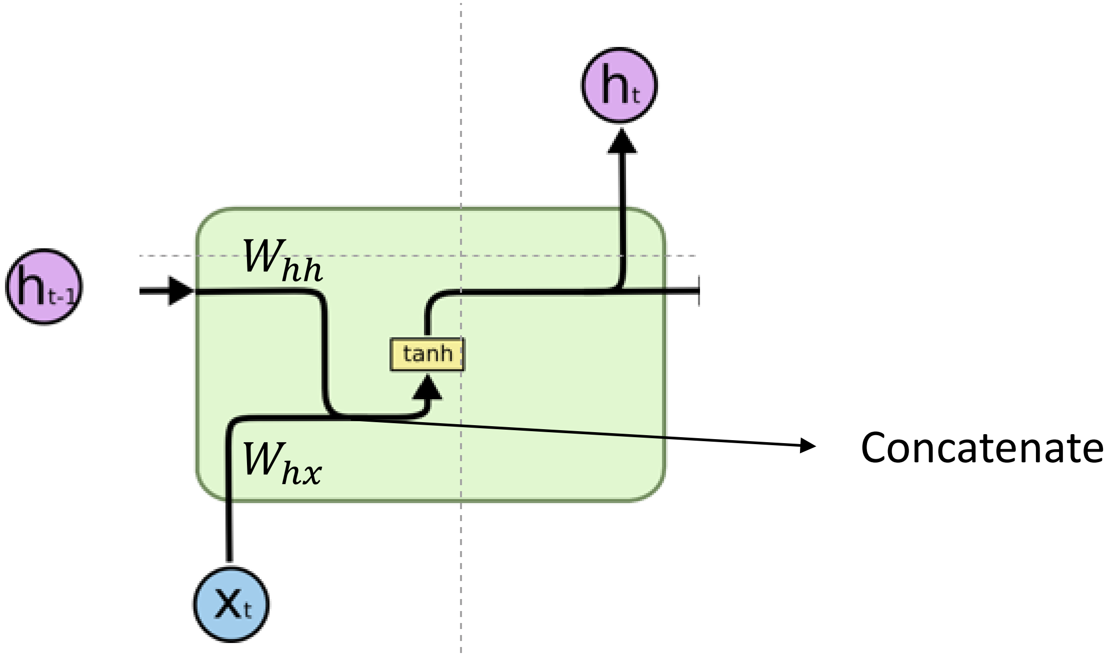
Recurrent Neural Networks (RNNs)



$\boxed{h_t = f_W(h_{t-1}, x_t)}$ $\xrightarrow{\text{A typical example}}$ $\boxed{h_t = tanh(W_{hh}h_{t-1} + W_{hx}x_t + b)}$

Some function with parameter W

Recurrent Neural Networks (RNNs)

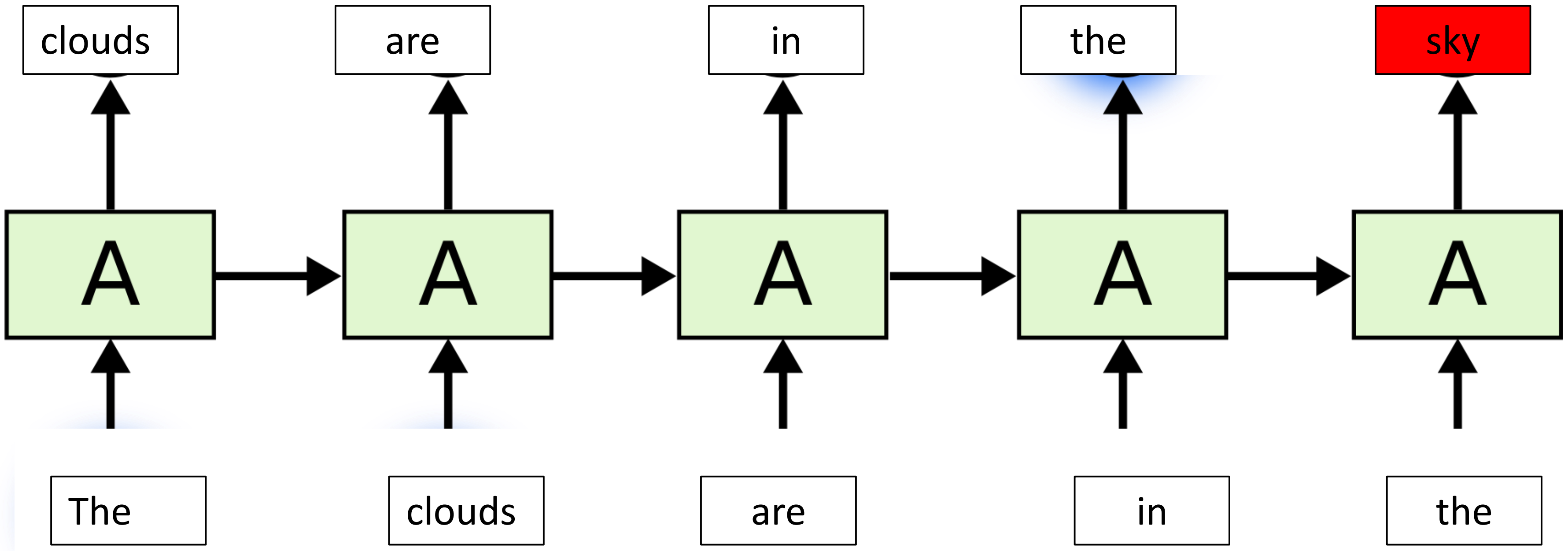


$$h_t = tanh(W_{hh}h_{t-1} + W_{hx}x_t + b)$$

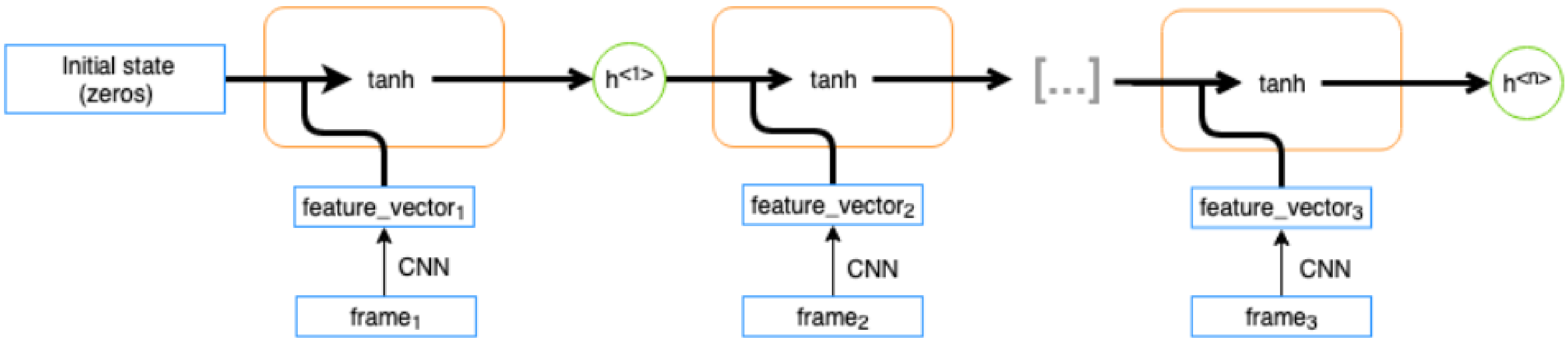
Recurrent Neural Networks (RNNs)

Task: Predict the next word

The clouds are in the *sky*



Recurrent Neural Networks (RNNs)

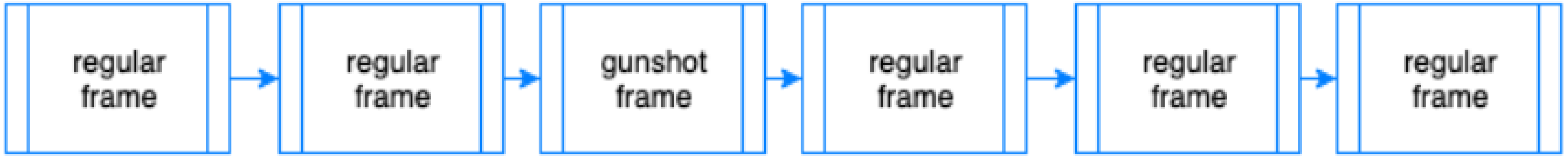


$$\hat{y}^{<t>} = \mathrm{softmax}(Vh^{<t>})$$

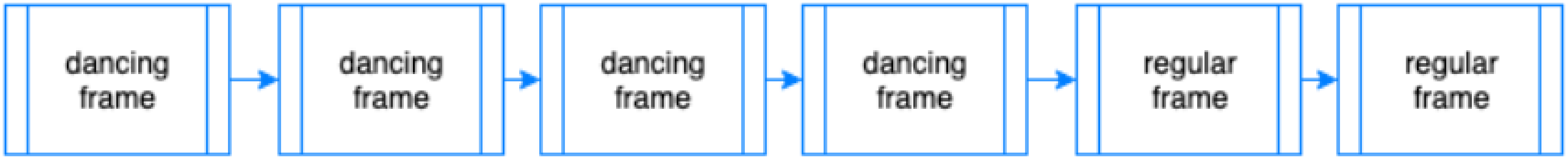
Recurrent Neural Networks (RNNs)

$$h^{<t>} = \tanh(W_{rec} h^{<t-1>} + W_{input} x^{<t>} + b)$$

Violent	0	0	0.9	0.9	0.9	0.9
Dance	0	0	0	0	0	0



Violent	0	0	0	0	0	0
Dance	0.1	0.3	0.5	0.7	0.6	0.5



Recurrent Neural Networks (RNNs)

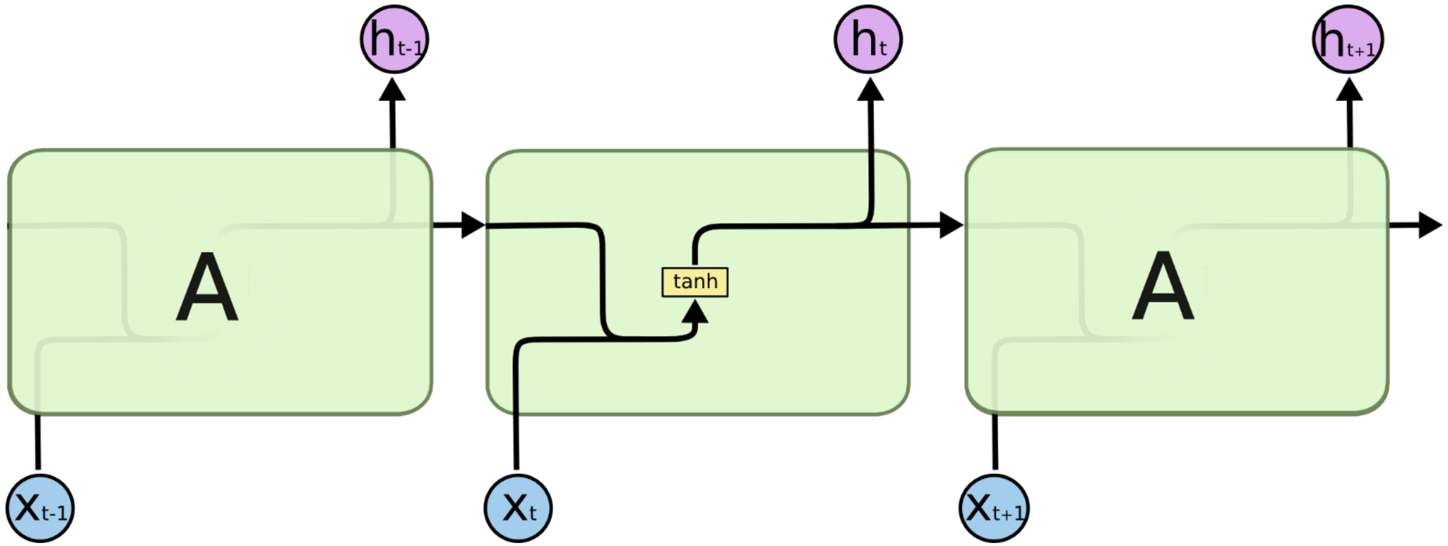
- Backpropagation through time (BPTT)

$$L^{<t>}(y,\hat{y}) = \sum_t L\left(y^{<t>},\hat{y}^{<t>}\right)$$

e.g. for t=4

$$\frac{\partial L^{<4>}}{\partial W_{rec}} = \frac{\partial L^{<4>}}{\partial \hat{y}^{<4>}} \frac{\partial \hat{y}^{<4>}}{\partial h^{<4>}} \frac{\boldsymbol{\partial h^{<4>}}}{\boldsymbol{\partial W_{rec}}}$$

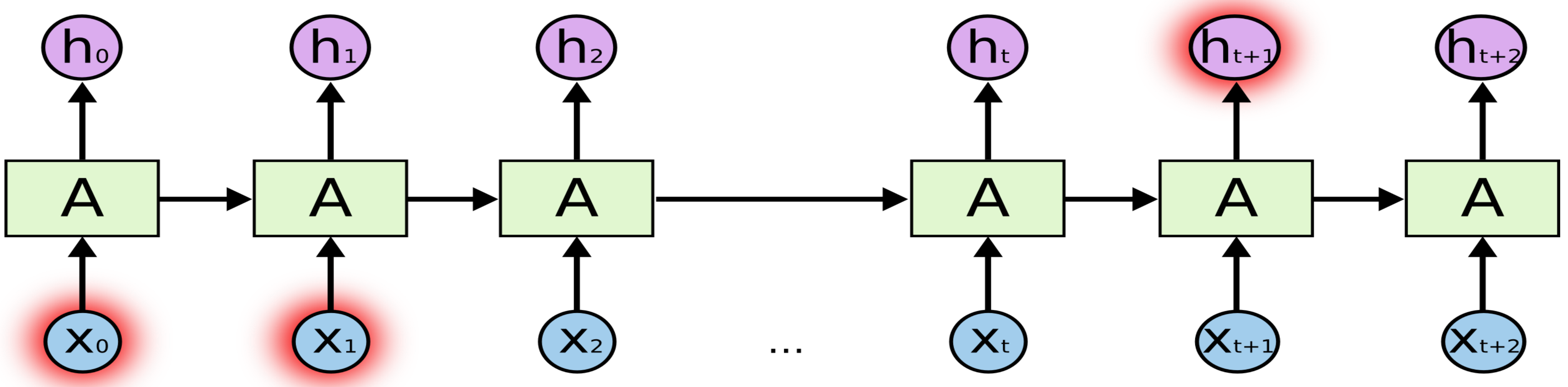
$$\frac{\partial h^{<4>}}{\partial W_{rec}} \rightarrow \frac{\partial h^{<4>}}{\partial W_{rec}} + \frac{\partial h^{<4>}}{\partial h^{<3>}} \frac{\partial h^{<3>}}{\partial W_{rec}} + \frac{\partial h^{<4>}}{\partial h^{<3>}} \frac{\partial h^{<3>}}{\partial h^{<2>}} \frac{\partial h^{<2>}}{\partial W_{rec}} \quad \ldots$$



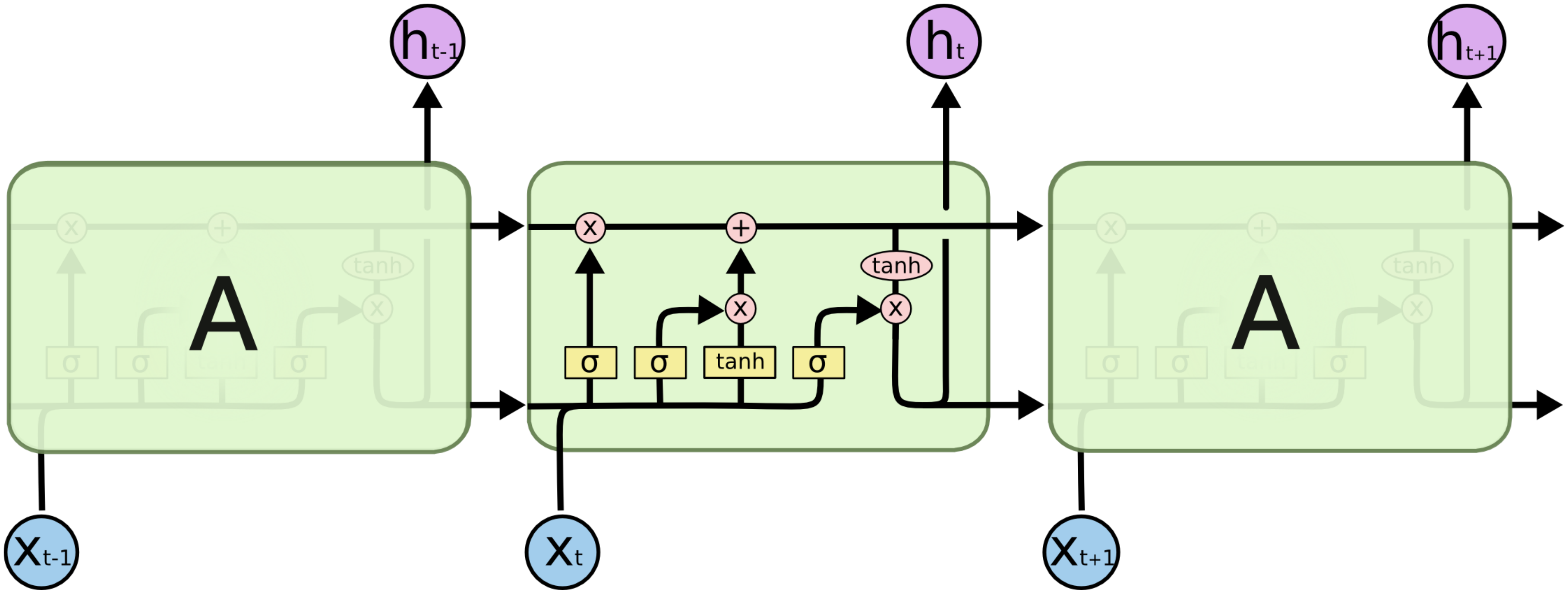
Recurrent Neural Networks (RNNs)

- Slow to train
- Gradient Vanishing/explosion

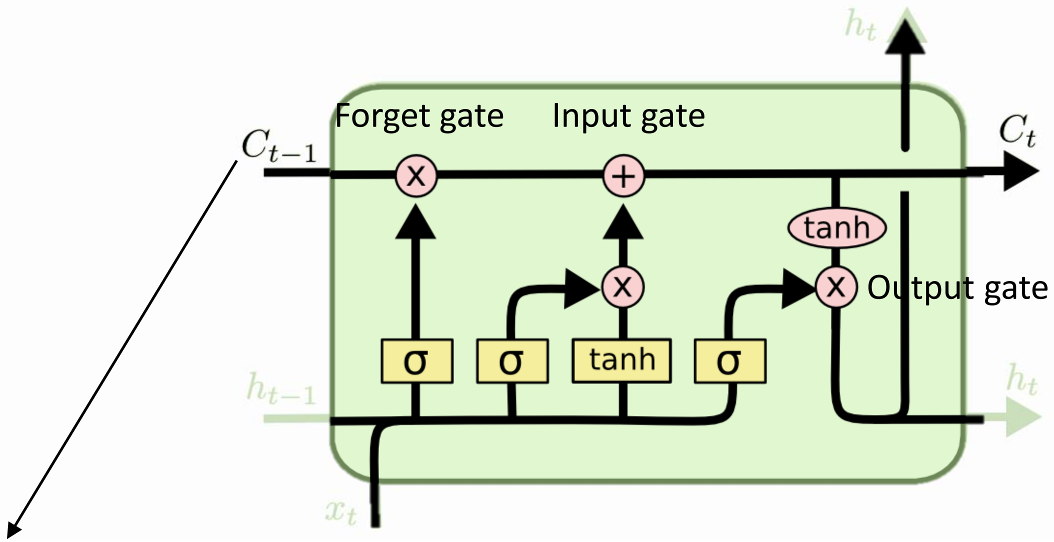
Not capable of learning long-term dependencies because of gradient vanishing factor.



Long Short term Memory (LSTM)

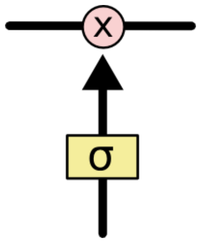


Long Short term Memory (LSTM)



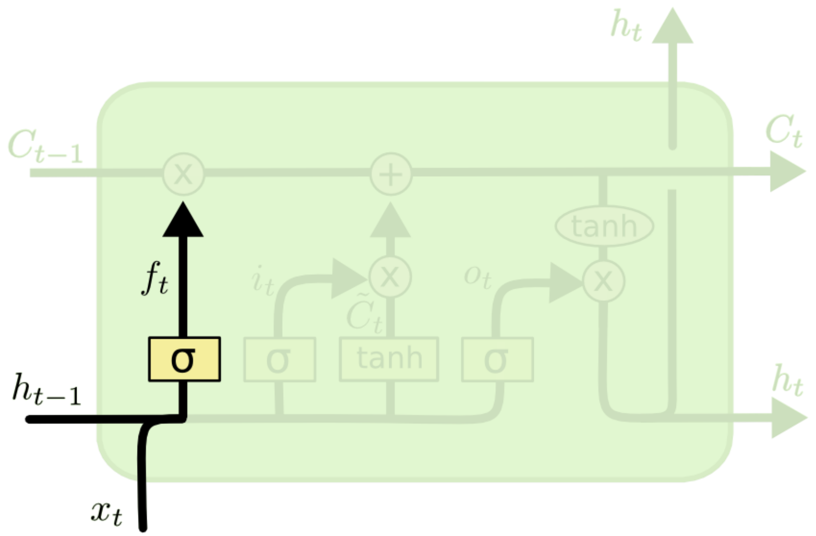
Key idea – The horizontal line
The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

cell state



Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

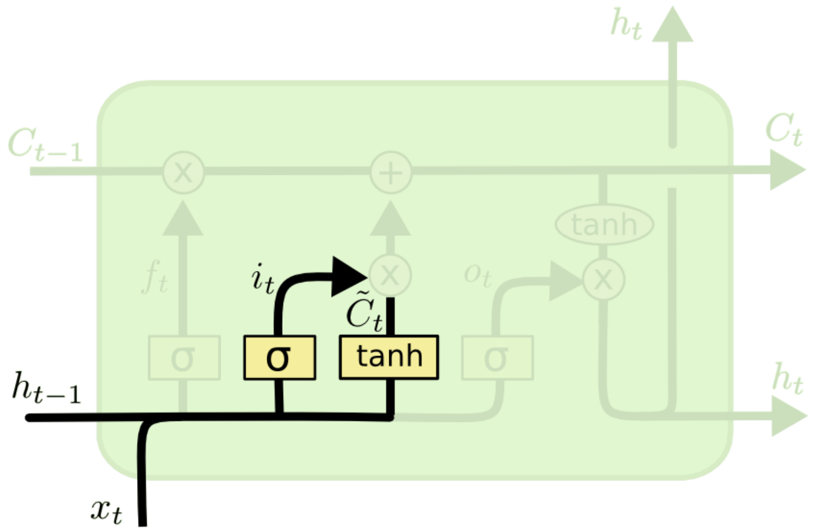
LSTM – How does it work?



Forget gate
The first step in our LSTM is to decide what information we're going to throw away from the cell state.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

LSTM – How does it work?



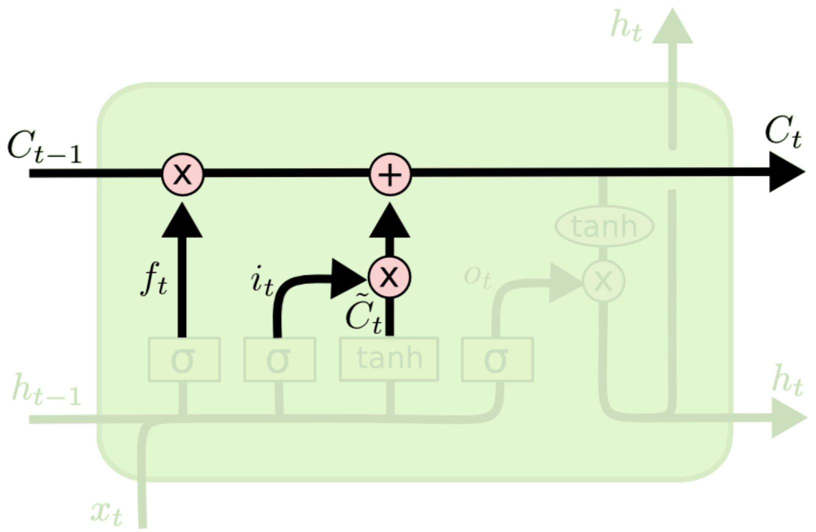
Input gate
The next step is to decide what new information we're going to store in the cell state.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Candidate cell state vector which can be added to the cell state

LSTM – How does it work?

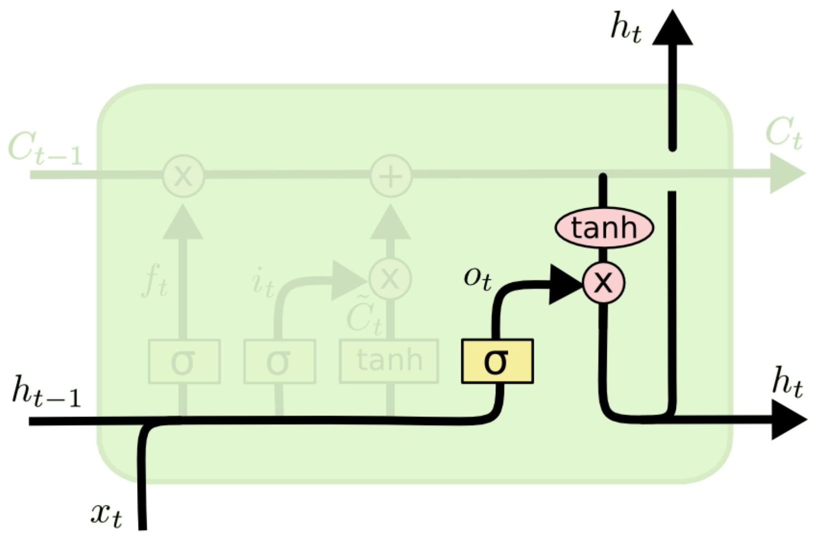


Cell state

It's now time to update the old cell state, C_{t-1}, into the new cell state C_t.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM – How does it work?

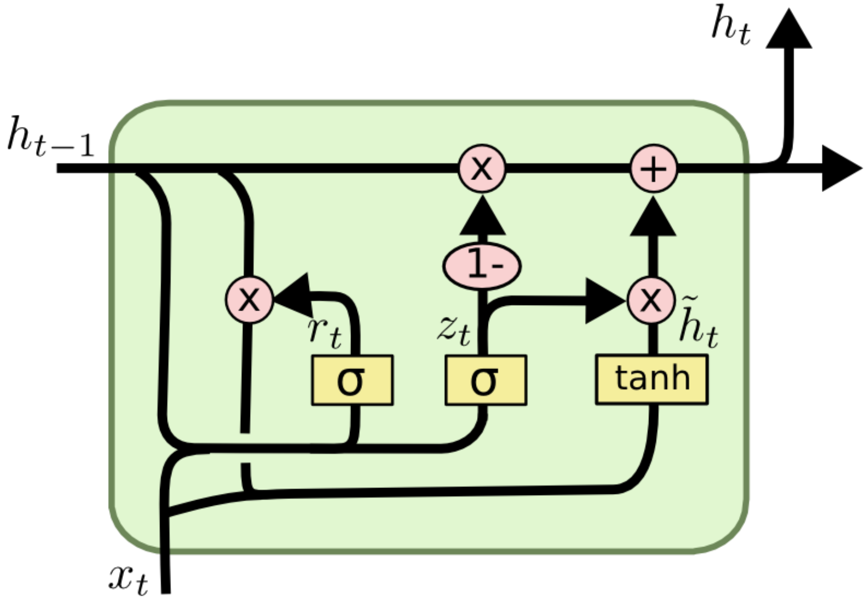


Output gate
Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version.

$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Gated Recurrent Units (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

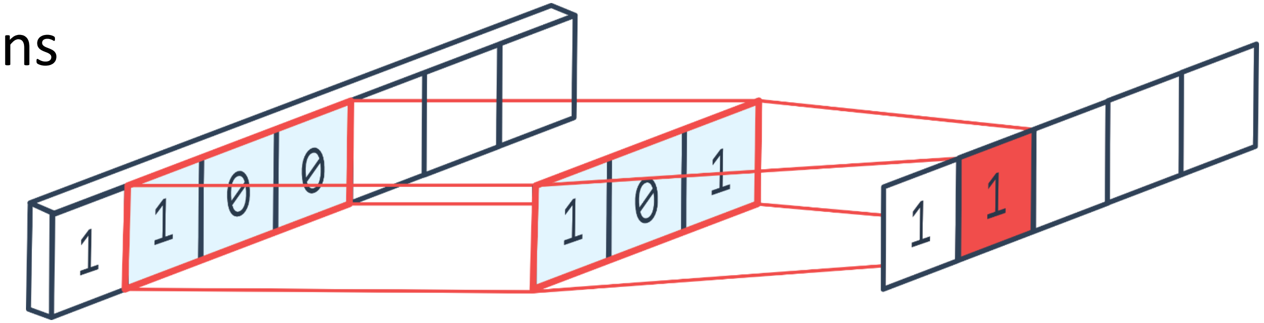
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

1D Convolutions

- LSTMs and GRUs problem with longer sequences
- 1 second of audio (at 22KHz) corresponds to ~22000 samples in
- Reduce dimensionality: 2D convolutions for images then 1D for sequences
- Recurrent layers with 1D convolutions



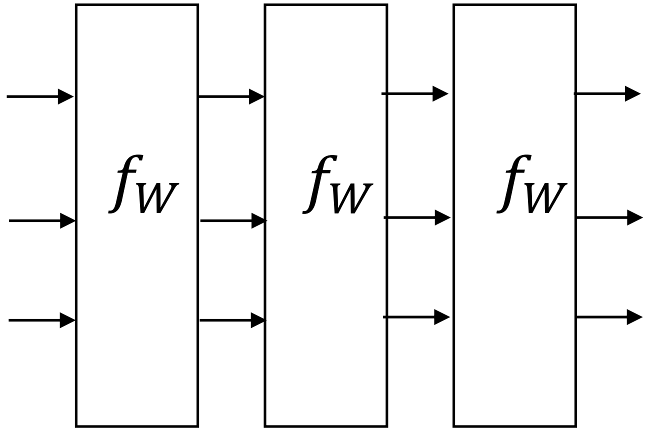
Implementations:

```
model = keras.models.Sequential([
    keras.layers.Conv1D(filters=20, kernel_size=4, strides=2, padding=
"valid",
                        input_shape=[None, 1]),
    keras.layers.GRU(20, return_sequences=True),
    keras.layers.GRU(20, return_sequences=True),
    keras.layers.TimeDistributed(keras.layers.Dense(10))
])
```

Next session: More on Temporal Convolutional Networks (TCNs)…

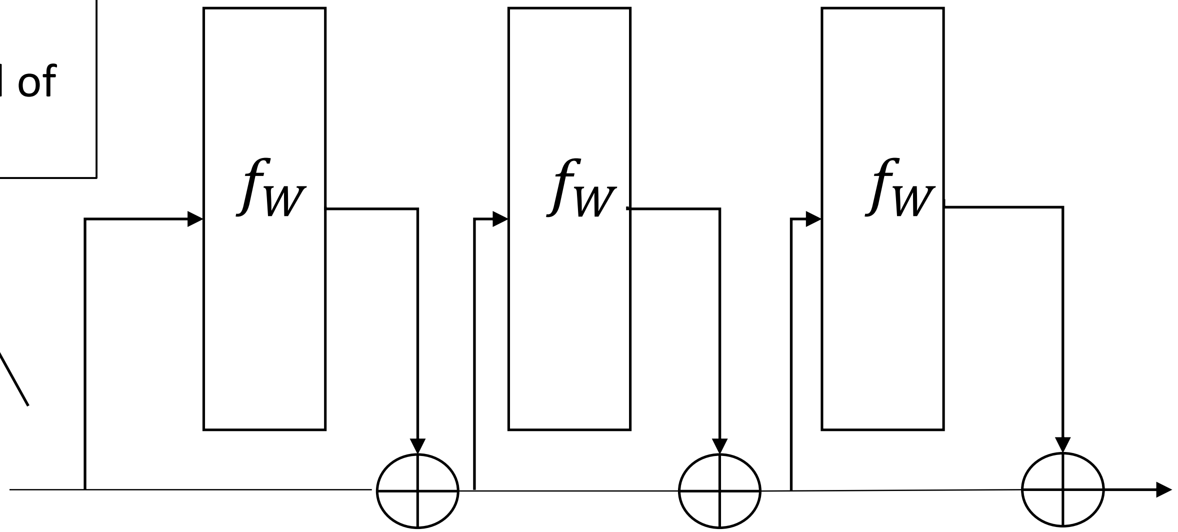
RNN vs LSTM

RNN



LSTM

Adding the updates instead of multiplying



At backprop, if we inject some gradients at the last time step, these $\oplus$ interaction are just gradient highways. They will flow till the first time step.

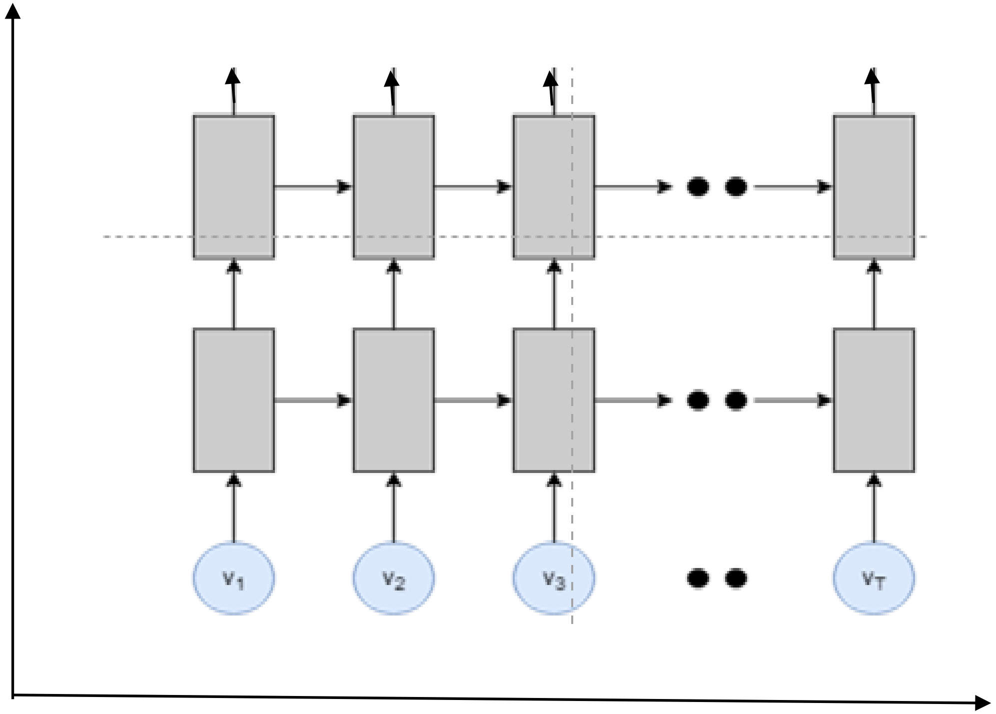
For RNN, there is the problem of vanishing gradients, where the gradients die off while backpropagating through.

RNNs or LSTMs

Output dimension of last time step – batch_size x n

Depth

Number of neurons (n) =2

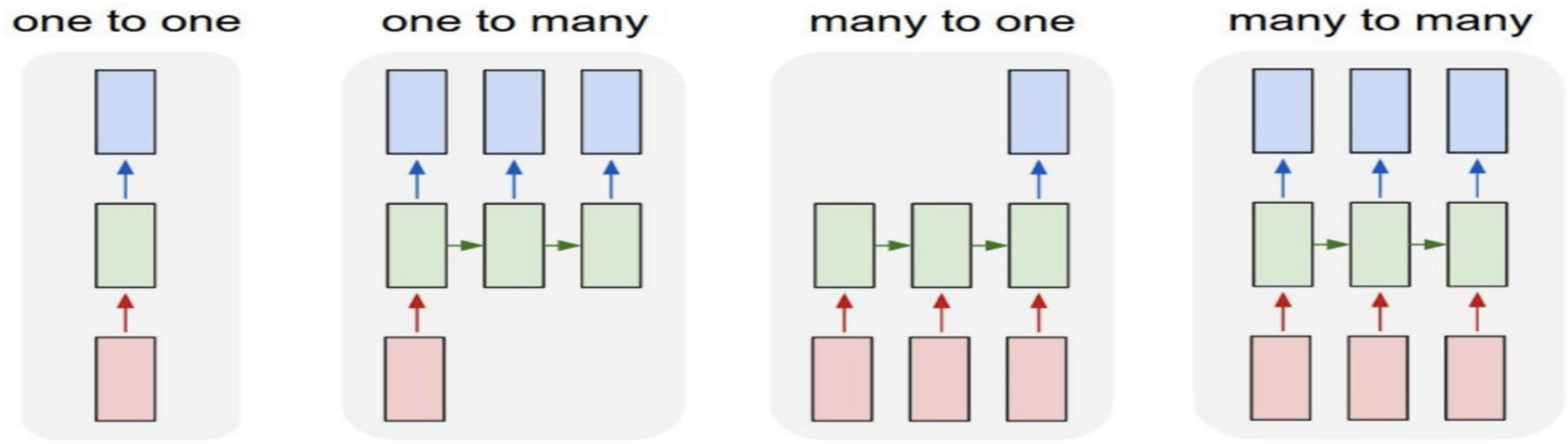


Input dimension – batch_size x T x #feature

time

Number of time steps = T

Types (Structural) of RNN



Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification)

Sequence output (e.g. image captioning takes an image and outputs a sentence of words).

Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).

Synced sequence input and output (e.g. video classification where we wish to label each frame of the video).

Implementing LSTMs

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

Data

X,	y
10, 20, 30	40
20, 30, 40	50
30, 40, 50	60

Predict

X,	y
70, 80, 90	??

https://colab.research.google.com/drive/1KsZsohKMPReksZAkDtLFK0p5mdF5RfdK?usp=sharing

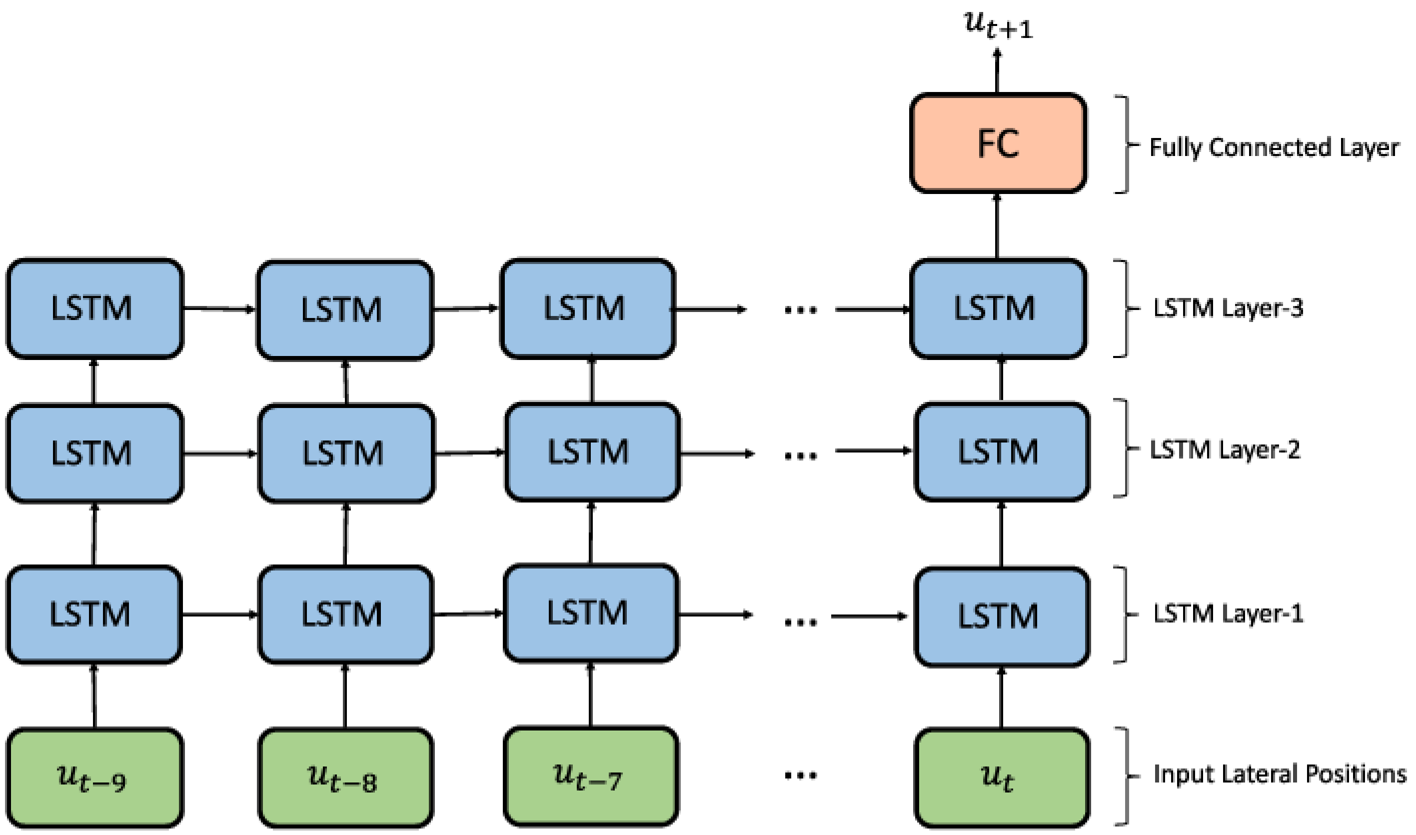
Time for a short break may be …….

Types (mechanism) of LSTMs

- Stacked LSTM – Stacking LSTMs layers
- Bi-directional LSTM – To model temporal information both forward and backward.
- CNN LSTM – To model temporal information on high level spatial features extracted from CNN
- ConvLSTM – The convolutional operation is embedded in each LSTM cell.

Stacked LSTM

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

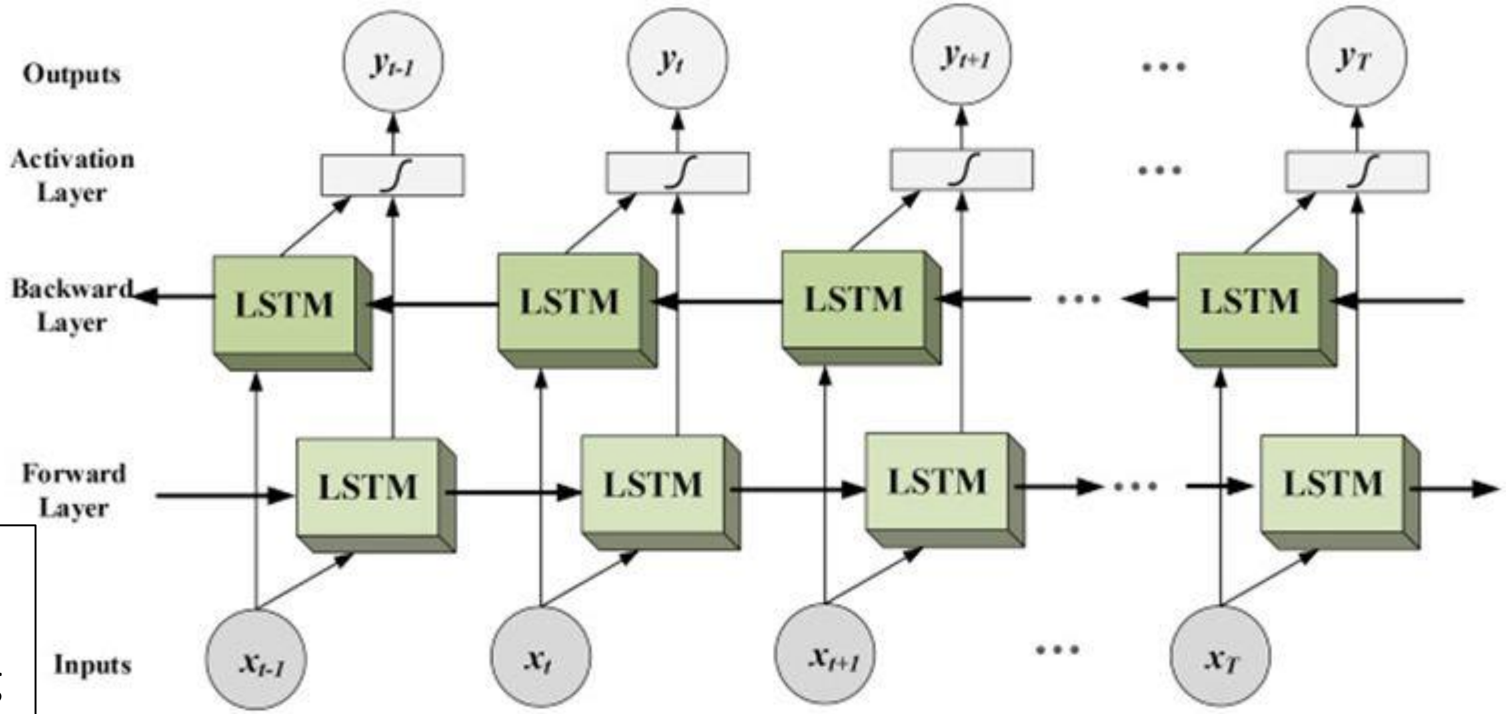


Let's try it!
https://colab.research.google.com/drive/1KsZsohKMPReksZAkDtLFK0p5mdF5RfdK?usp=sharing

Bi-directional 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.

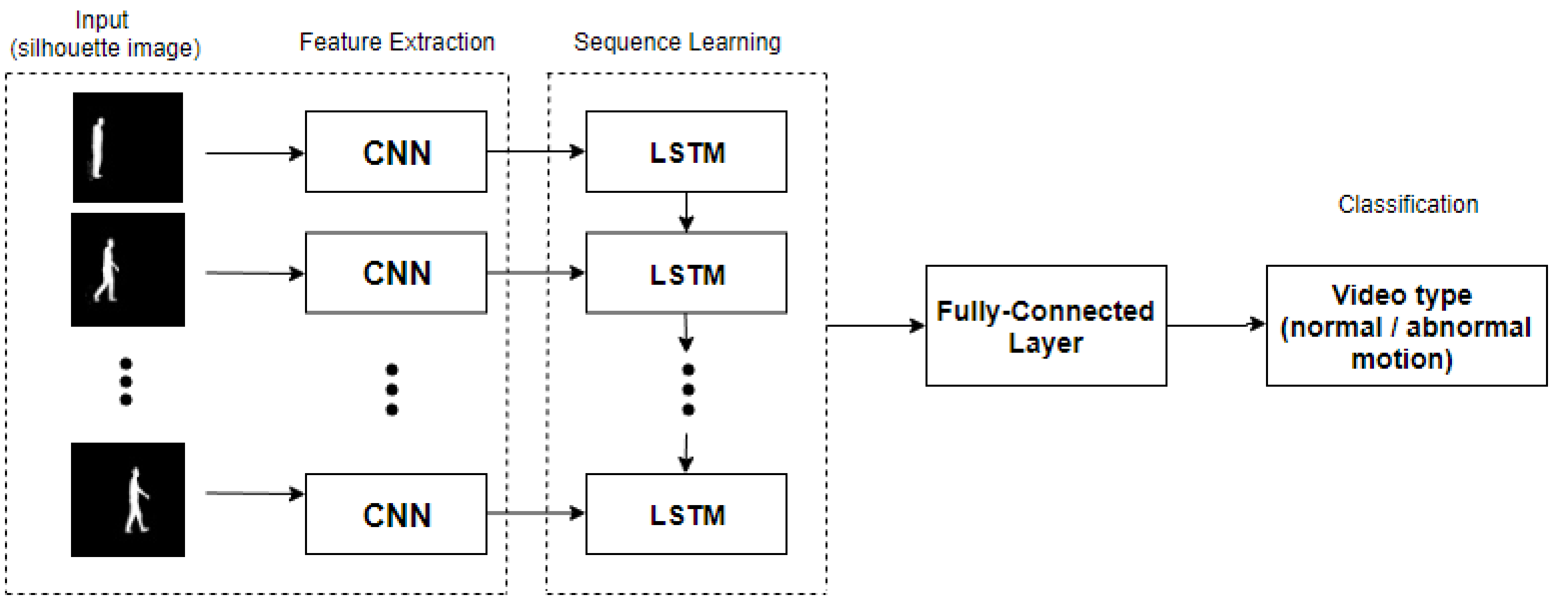


Implementation
We can implement a Bidirectional LSTM for univariate time series forecasting by wrapping the first hidden layer in a wrapper layer called Bidirectional.

Let's try it!
https://colab.research.google.com/drive/1KsZsohKMPReksZAkDtLFK0p5mdF5RfdK?usp=sharing

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.



<u>Data</u>

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

<u>Predict</u>

X,	y
60, 70, 80, 90	??

Let's try it!
https://colab.research.google.com/drive/1TRuHaLJbkqqLpbCn8E2J8Ky1qEdCTC64?usp=sharing

ConvLSTM

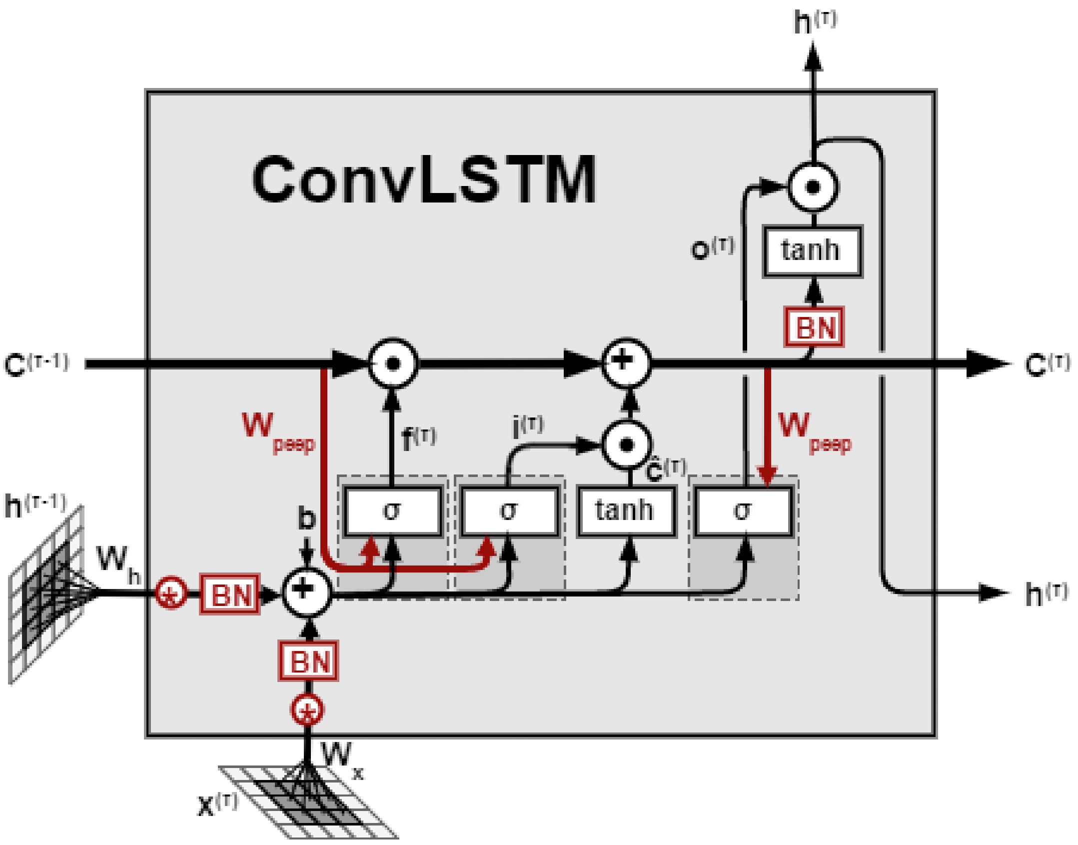
A type of LSTM related to the CNN-LSTM is the ConvLSTM, where the convolutional reading of input is built directly into each LSTM unit. The ConvLSTM was developed for reading two-dimensional spatial-temporal data.

<u>Data</u>

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

<u>Predict</u>

X,	y
60, 70, 80, 90	??



Let's try it!
https://colab.research.google.com/drive/1TRuHaLJbkqqLpbCn8E2J8Ky1qEdCTC64?usp=sharing

Disadvantages

- RNNs operate on spatial vectors fed to it. Hence, they do not capture spatio-temporal information. (will be discussed in detail later)

- Not much efficient on small datasets (pre-training LSTMs is not a good idea as they change the statistics learned by the gates).

- Works only when the data is highly informative in terms of temporal variation. (For example- fails to recognize low motion actions in a video)

Next Session ….

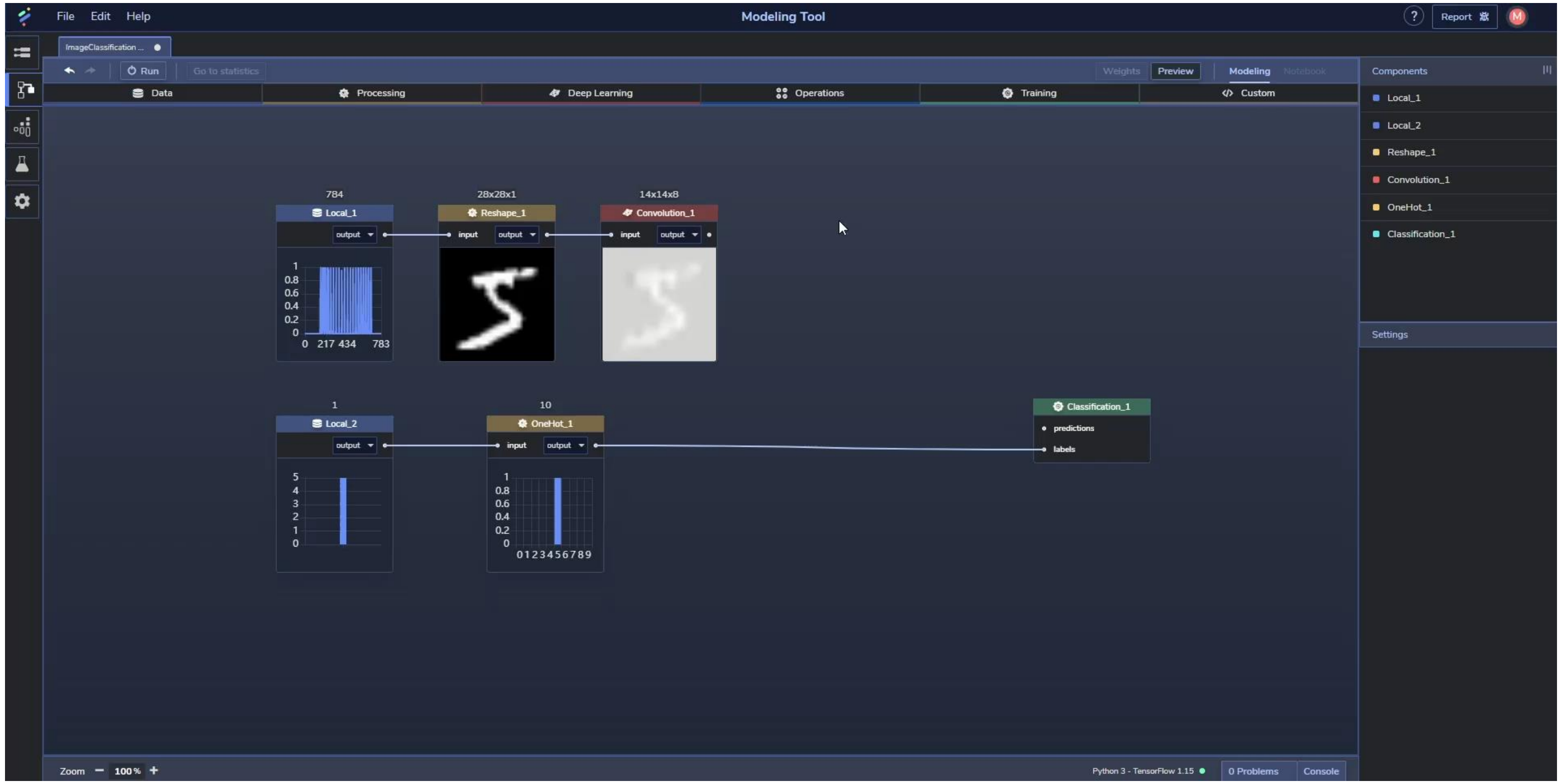
- Introduction to Action Recognition in videos
- 3D Convolutional Networks
- Action Detection
- Temporal Convolutional Networks (TCN)

e-mail: farhood.negin@inria.fr

PerceptiLabs (Bonus Content!)

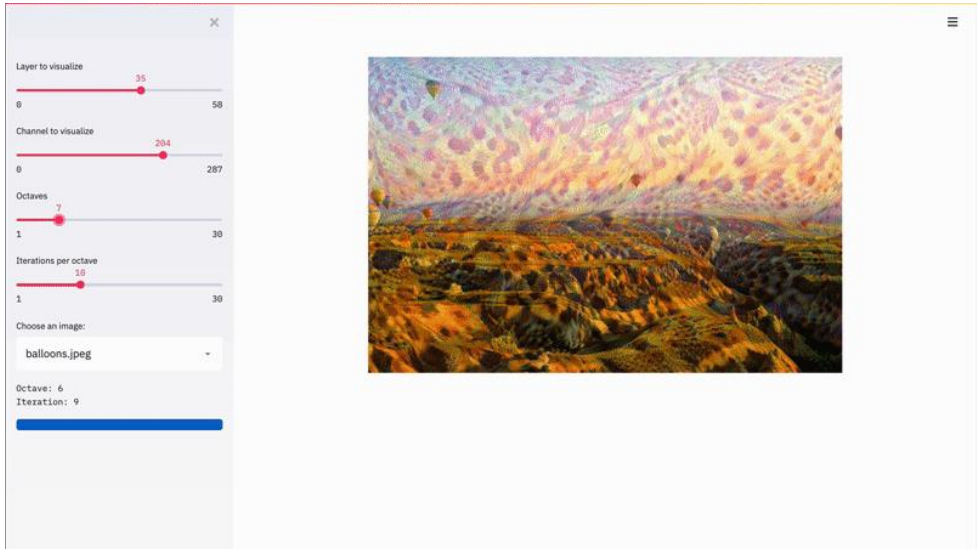
- Visual modeling tool for TensorFlow
  - Classification
  - Regression
  - Object detection
  - GANs
  - Reinforcement learning

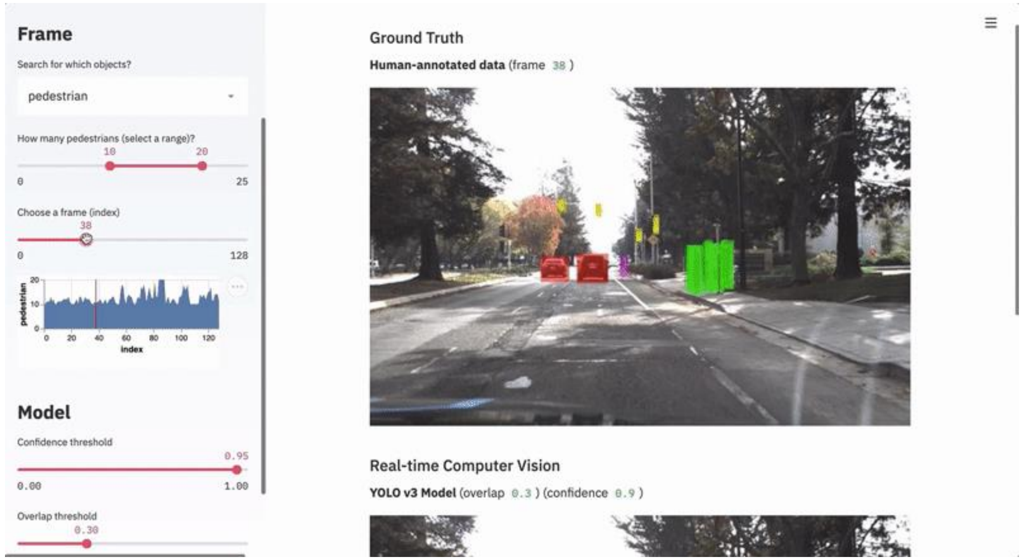
https://www.perceptilabs.com/home

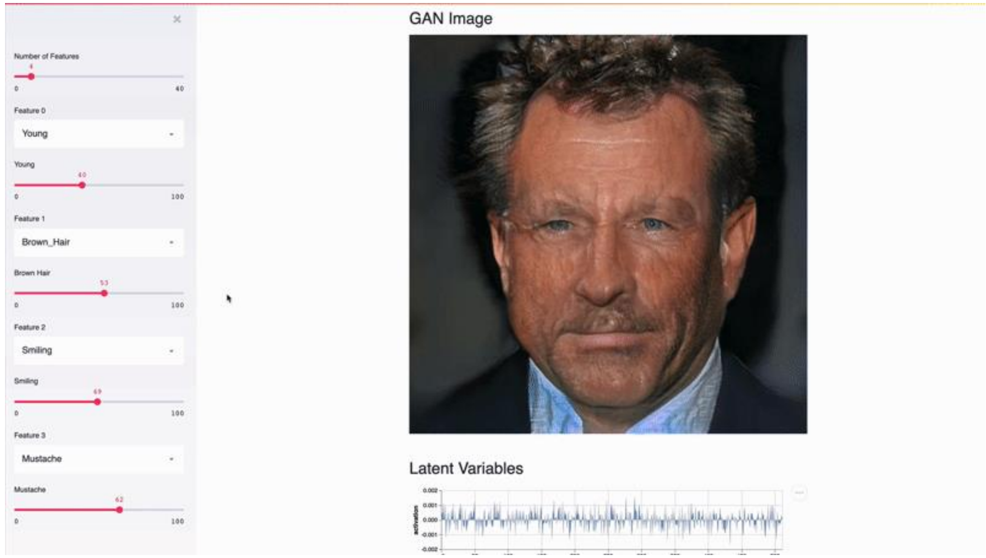


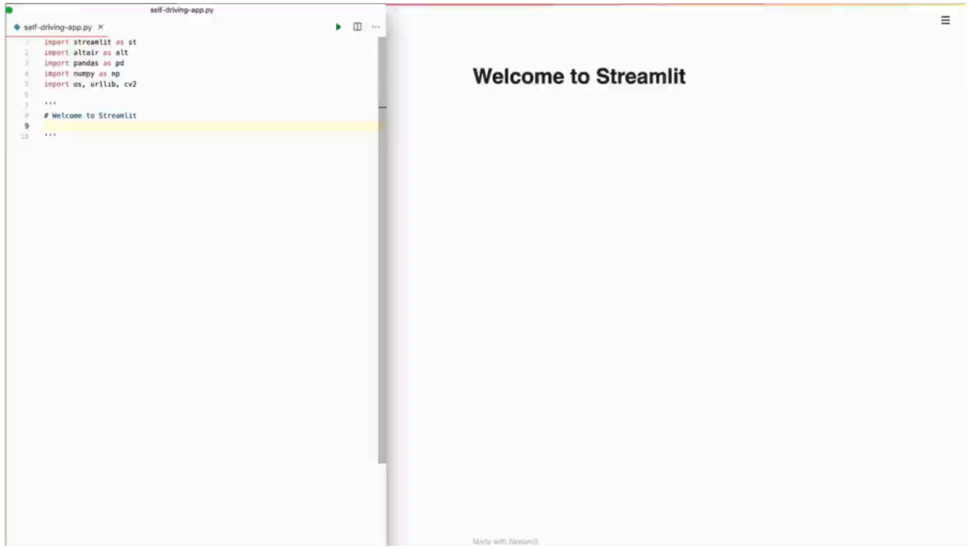
StreamLit (Bonus Content!)

- Design user interface for your models and create data apps









Microsoft Lobe (Bonus Content!)

https://lobe.ai/examples

Exercise!

- The link for the exercise: https://drive.google.com/file/d/1c4z9lAXdkqf1Ak7SHPtAWYwpg0hiVCQ8/view?usp=sharing

Deep Learning Winter School for Computer Vision 2020

Assignment 1

28 January 2020

Instructions- Answer the following questions in a pdf file. For question 3, include the code in the pdf file and the sharable link of Google Colab (with only view option).
Name of the pdf file should be your *Familyname_Firstname*.pdf. Submit the assignment before 2/Feb/2020, 23:59 PM at srijan.das@inria.fr with subject - **DLWSC - 2020 Assignment 1**.

1. What is the difference between statefll and stateless LSTM?
2. Differentiate between a single LSTM layer of 100 neurons and a stacked 2-layered LSTM each of 50 neurons?
3. The problem we are going to look at in this post is the International Airline Passengers prediction problem. This is a problem where, given a year and a month, the task is to predict the number of international airline passengers in units of 1,000. The data ranges from January 1949 to December 1960, or 12 years, with 144 observations. Download the data from airline-passengers.csv. Split the data into $(2/3)^{rd}$ for training and the rest for testing.

   We can phrase the problem as a regression problem. That is, given the number of passengers (in units of thousands) this month, what is the number of passengers next month? Implement the best possible Neural Network for this problem.
   Use the below code snippet to load the dataset.

```
# load the dataset
dataframe = pandas.read_csv('airline-passengers.csv', usecols=[1],
    engine='python')
dataset = dataframe.values
dataset = dataset.astype('float32')
```

References

- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
- B457/I400: Intro to Computer Vision (Spring 2018) (Michael Ryoo)
- CS231n Winter 2016: Lecture 10: Recurrent Neural Networks, Image Captioning, LSTM (Andrej Karpathy)