Lecture 12:

Long Short-Term Memory Networks (LSTMs) – Part II

Markus Hohle

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units

Spring 2025

https://www.analyticsvidhya.com

Outline

- LSTM for Classification
- Bidirectional LSTMs
- Stacked LSTMs
- LSTM + CNN

Machine Learning Algorithms:

https://www.analyticsvidhya.com

Outline

- **LSTM for Classification**
- Bidirectional LSTMs
- Stacked LSTMs
- LSTM + CNN

LSTM II:

minimal model:

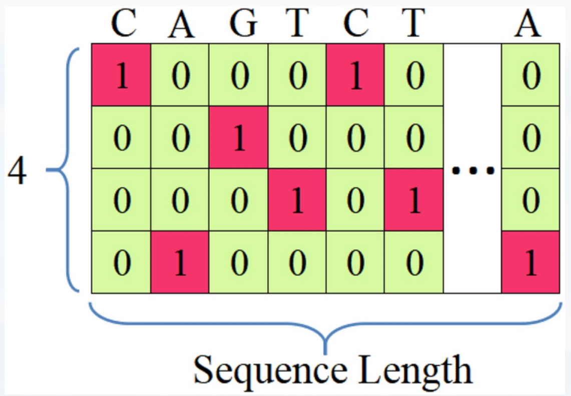
Y is one-hot encoded

```
[N_samples, LengthSeq, N_features] = X.shape
[N_samples,            N_classes ] = Y.shape


model = Sequential()

model.add(LSTM(n_neurons, activation = 'tanh',\
                       input_shape = (LengthSeq , N_features)))

model.add(Dense(N_classes, activation = 'softmax'))

opt = optimizers.Adam()
model.compile(loss = 'categorical_crossentropy', optimizer = opt,\
                      metrics = ['accuracy'])

model.summary()
```

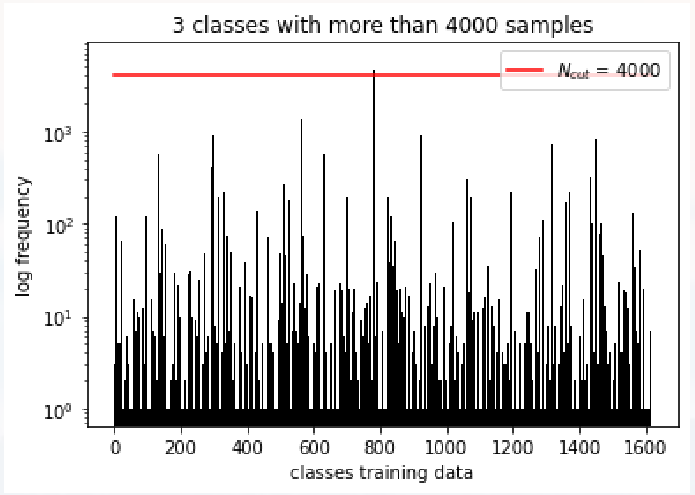
LSTM II:

LSTM for Classification

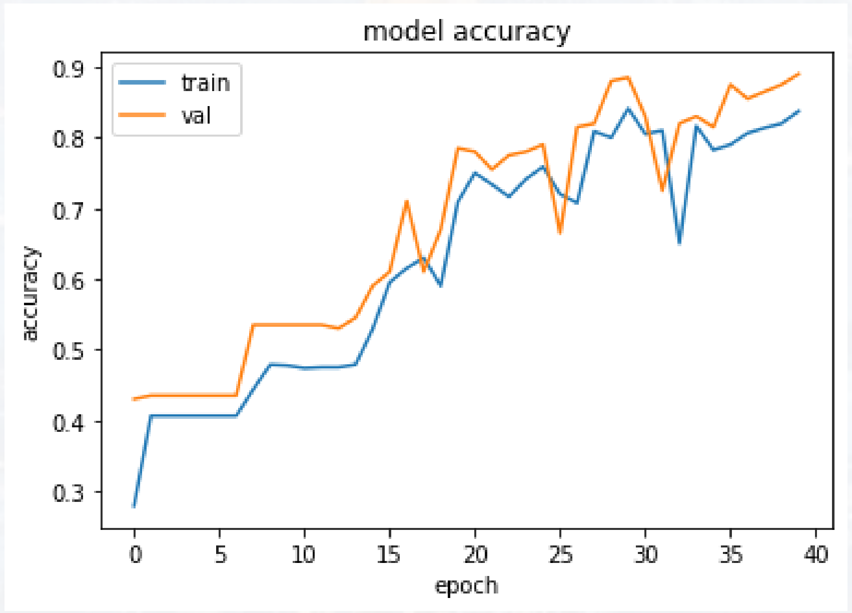
barcode example

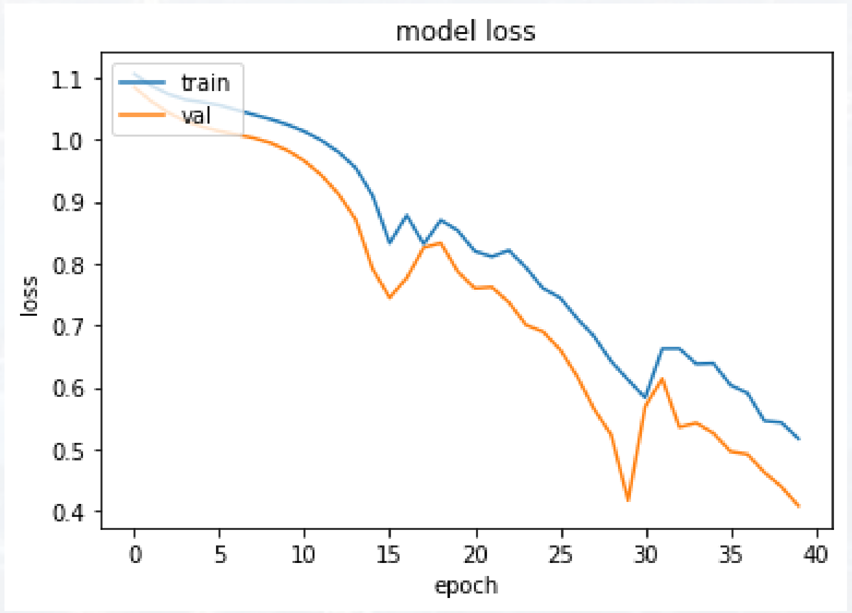


	C	A	G	T	C	T	…	A
	1	0	0	0	1	0		0
	0	0	1	0	0	0		0
4	0	0	0	1	0	1	…	0
	0	1	0	0	0	0		1



for computational reasons:
- **three** classes
- **1k** samples total
- sequences cut to length **500**

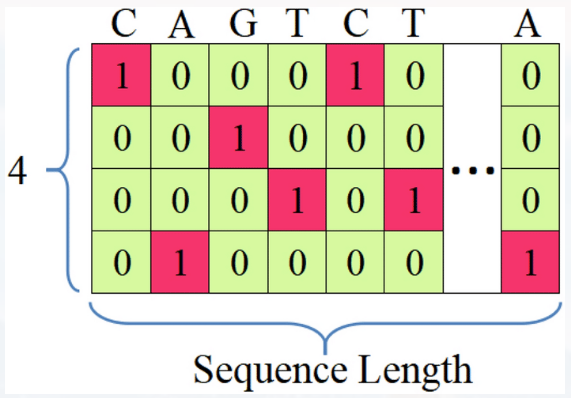


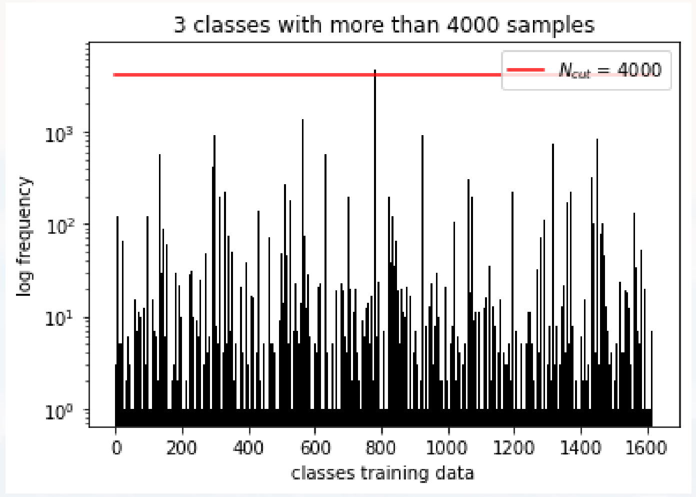


LSTM II:

LSTM for Classification

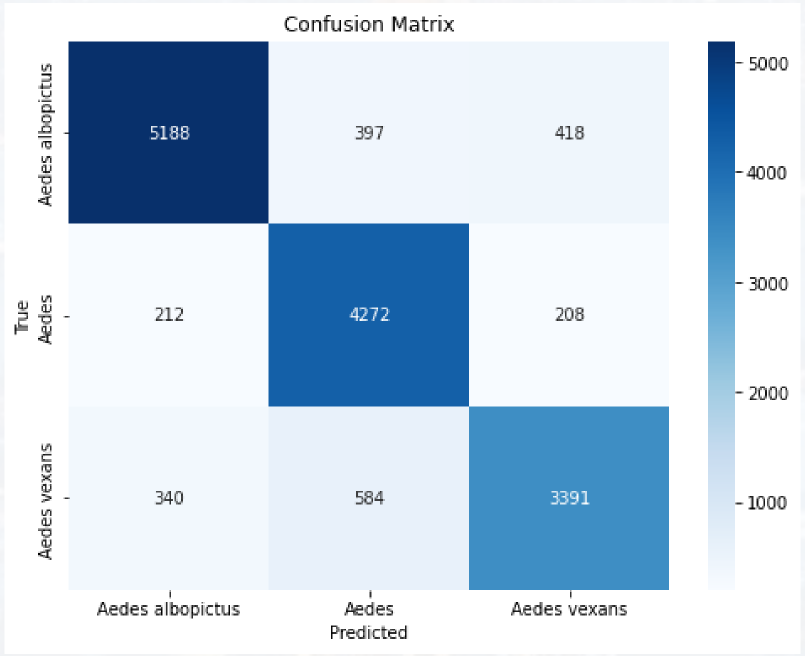
barcode example



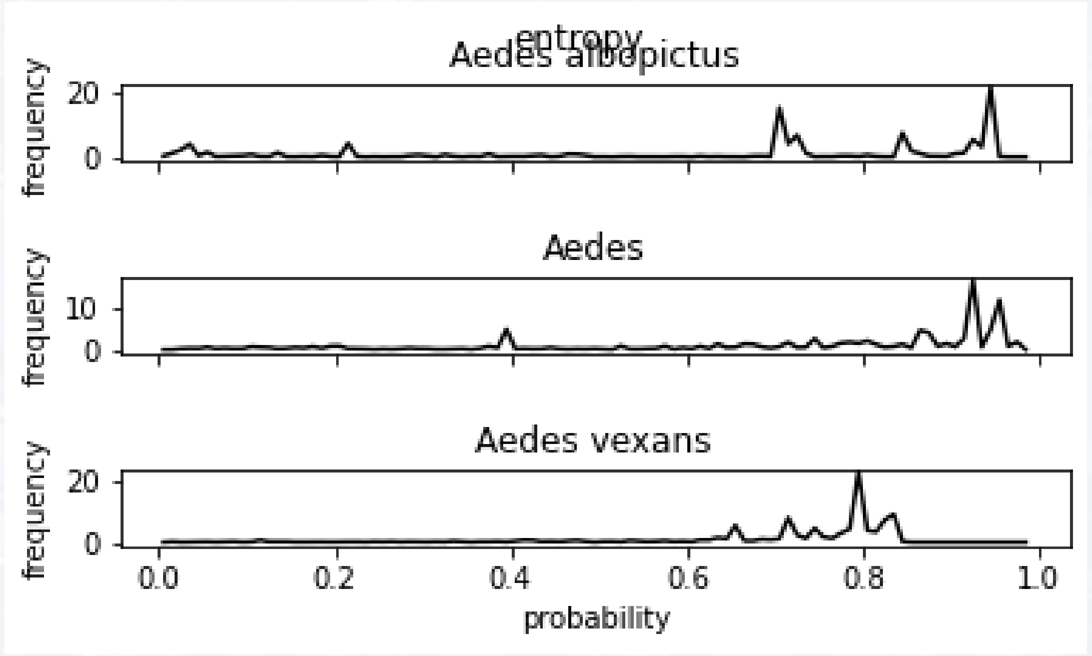


for computational reasons:

- **three** classes
- **1k** samples total
- sequences cut to length **500**



	Aedes albopictus	Aedes	Aedes vexans
Aedes albopictus	5188	397	418
Aedes	212	4272	208
Aedes vexans	340	584	3391



https://www.analyticsvidhya.com

<u>Outline</u>

- LSTM for Classification

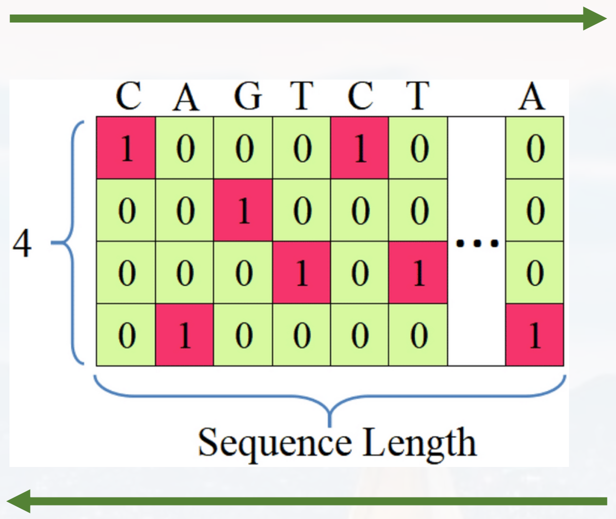
- Bidirectional LSTMs

- Stacked LSTMs

- LSTM + CNN

LSTM II: Bidirectional

sometimes, sequences can be read from two directions :



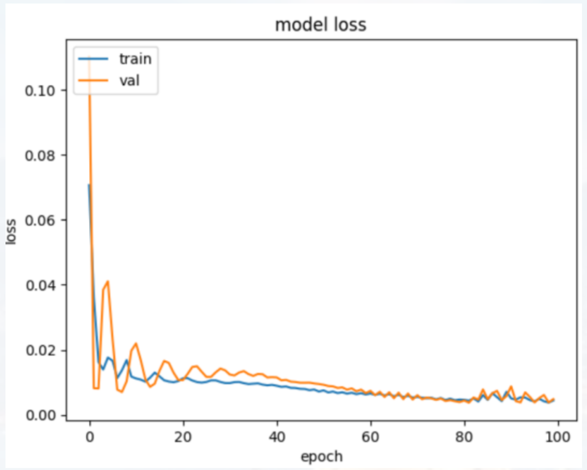
```
from keras.layers import Bidirectional
```

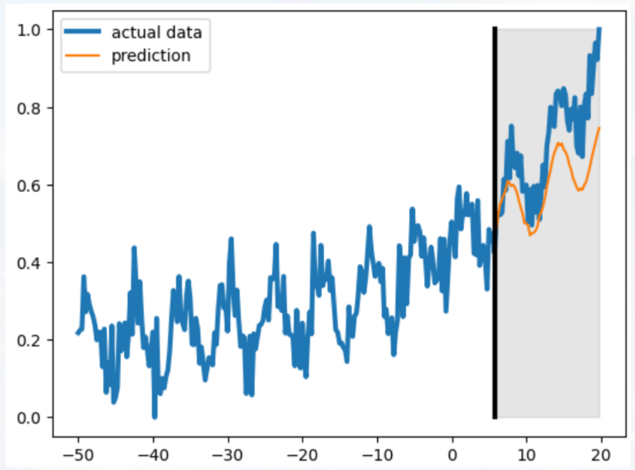
```
model = Sequential()
model.add(Bidirectional(LSTM(n_neurons, activation = 'tanh'),\
                        input_shape = (dt_past, n_features)))
model.add(Dense(dt_futu))

opt = optimizers.Adam()
model.compile(loss = 'mean_squared_error', optimizer = opt)

model.summary()
```

LSTM II: Bidirectional

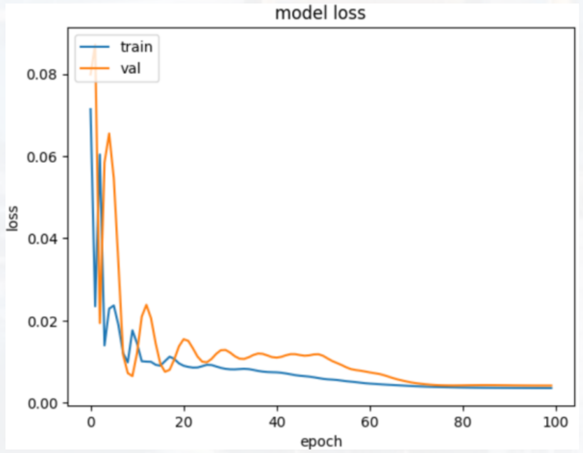
```
model = Sequential()
model.add(Bidirectional(LSTM(n_neurons, activation = 'tanh'), input_shape = (dt_past, n_features)))
model.add(Dense(dt_futu))
opt = optimizers.Adam()
model.compile(loss = 'mean_squared_error', optimizer = opt)
model.summary()
```

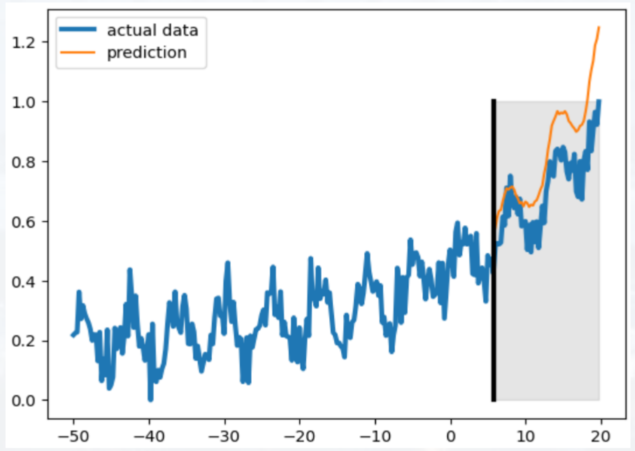
vanilla





bidirectional





```
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirection  (None, 800)               1286400
 al)

 dense_1 (Dense)             (None, 8)                 6408

=================================================================
Total params: 1292808 (4.93 MB)
Trainable params: 1292808 (4.93 MB)
Non-trainable params: 0 (0.00 Byte)
```

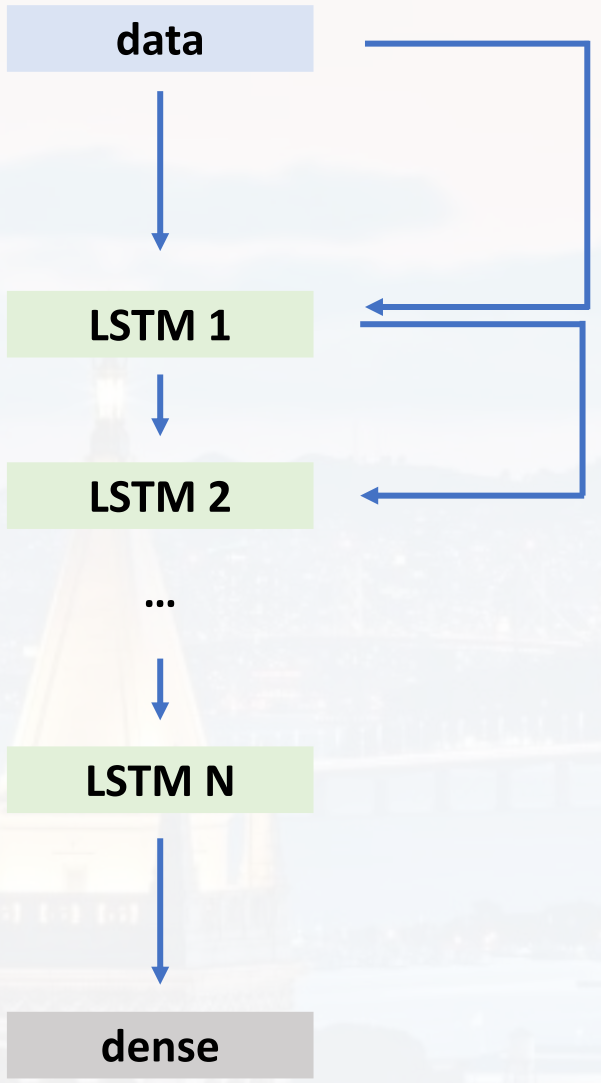
https://www.analyticsvidhya.com

Outline

- LSTM for Classification
- Bidirectional LSTMs
- **Stacked LSTMs**
- LSTM + CNN

LSTM II: Stacked

idea:

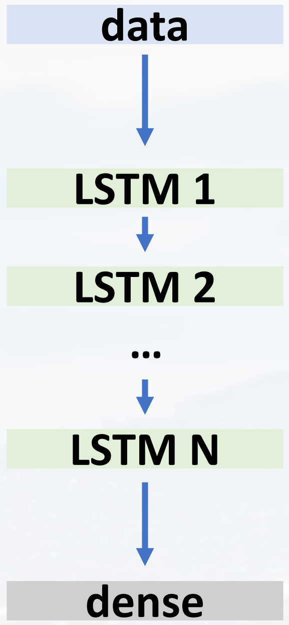


shape: (len(y(t)) - dt_past - dt_futu + 1) x dt_past x features

shape: (len(y(t)) - dt_past - dt_futu + 1) x dt_past x hidden state

```
return_sequences = True
```

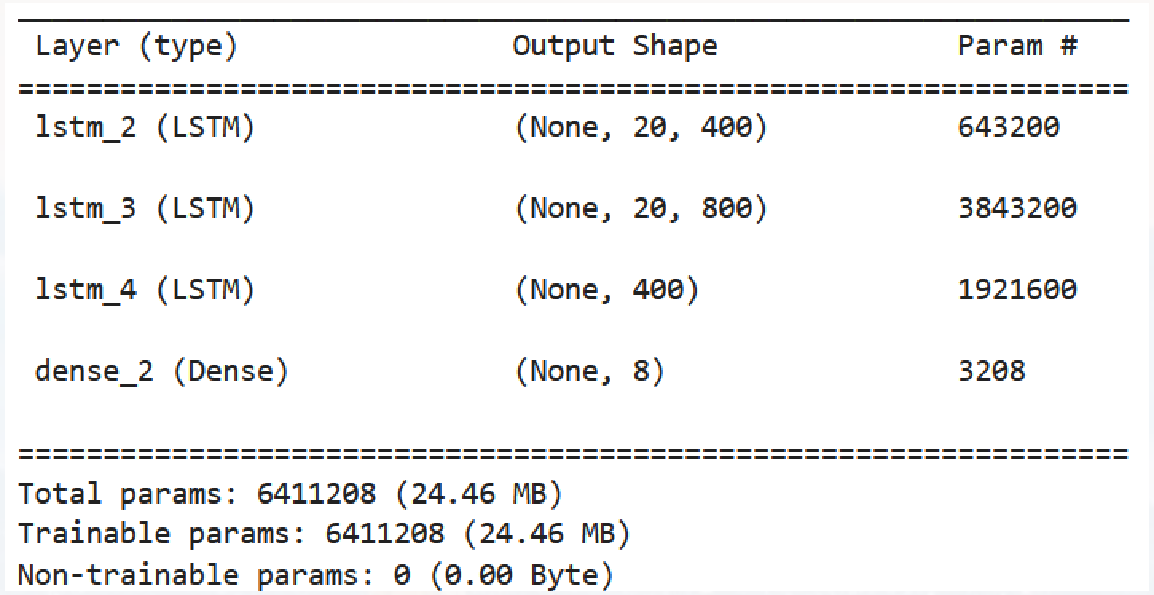
LSTM II: Stacked



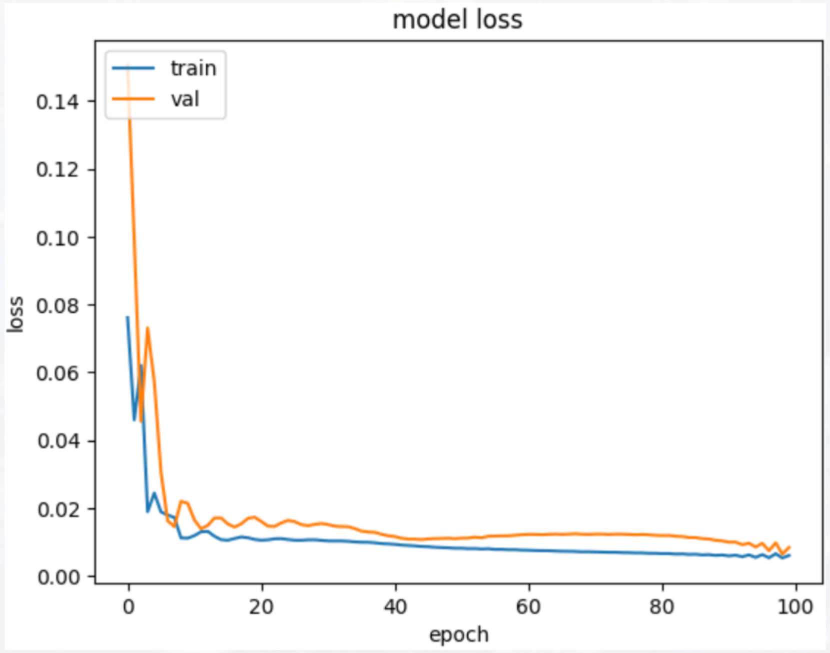
```
model = Sequential()

model.add(LSTM(n_neurons, activation = 'tanh',\
               return_sequences = True, input_shape = (dt_past, n_features)))

model.add(LSTM(2*n_neurons, activation = 'relu',\
               return_sequences = True))

model.add(LSTM(n_neurons, activation = 'relu'))

model.add(Dense(dt_futu))

opt = optimizers.Adam()
model.compile(loss = 'mean_squared_error', optimizer = opt)

model.summary()
```

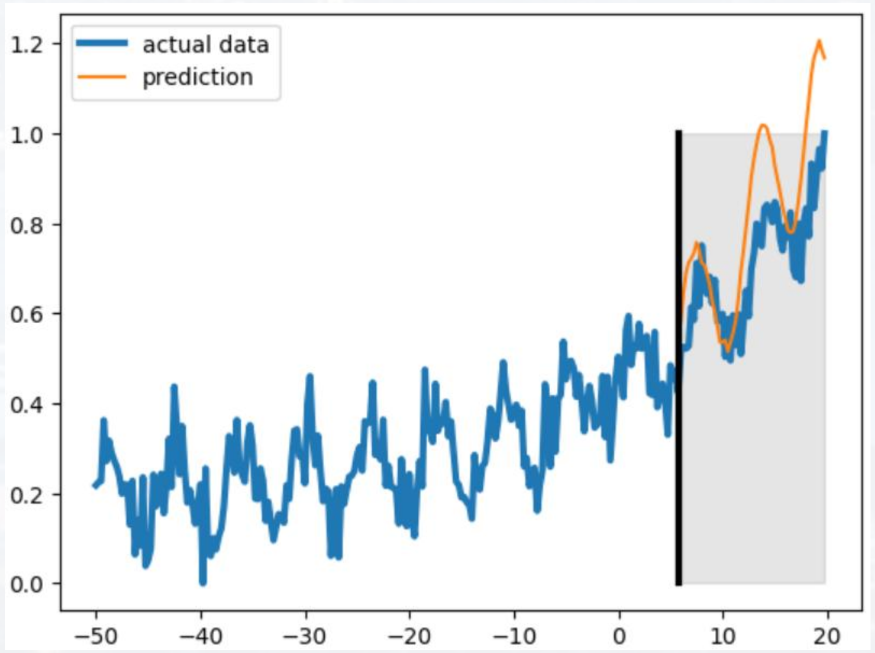
all LSTMs, **except the last** stack needs `return_sequences = True`

LSTM II: Stacked



```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 20, 400)           643200

lstm_3 (LSTM)                (None, 20, 800)           3843200

lstm_4 (LSTM)                (None, 400)               1921600

dense_2 (Dense)              (None, 8)                 3208

=================================================================
Total params: 6411208 (24.46 MB)
Trainable params: 6411208 (24.46 MB)
Non-trainable params: 0 (0.00 Byte)
```





Machine Learning Algorithms:

https://www.analyticsvidhya.com

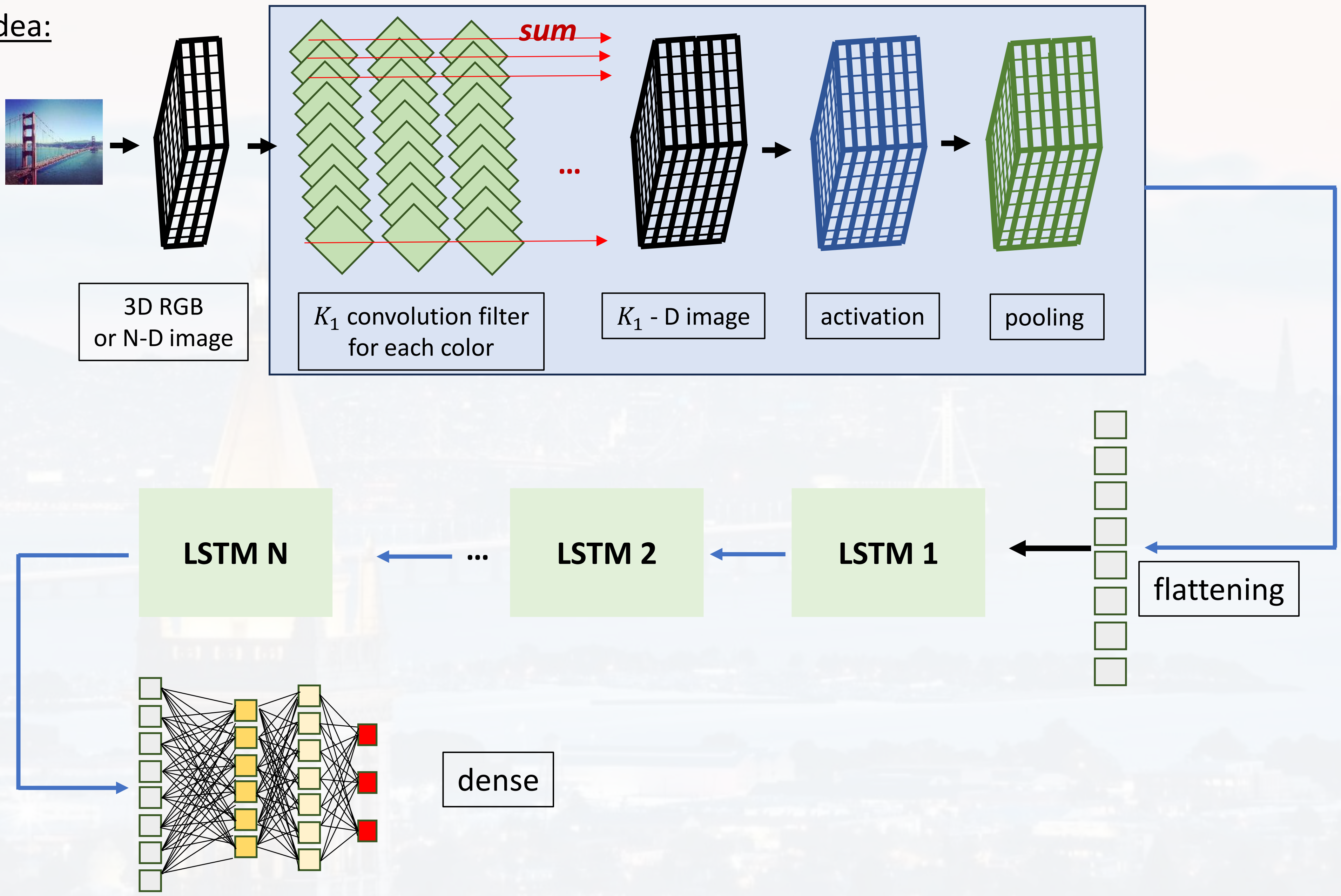
Outline

- LSTM for Classification
- Bidirectional LSTMs
- Stacked LSTMs
- **LSTM + CNN**

LSTM II:

LSTM + CNN

idea:



LSTM II:

idea:

input expected by CNN (images):

```
(N_images,    N_x, N_y, N_color)
```

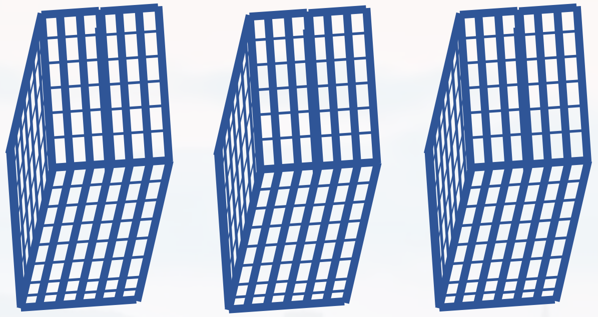
input expected by CNN (videos):

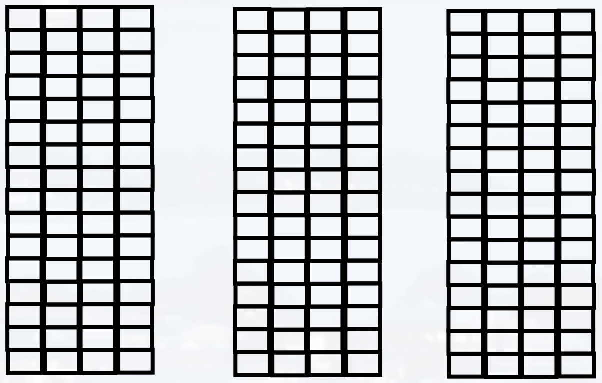
```
(N_videos,    N_t, N_x, N_y, N_color)
```

input expected by LSTM (sequences):

```
(N_sequences, N_t, N_feature)
```

None

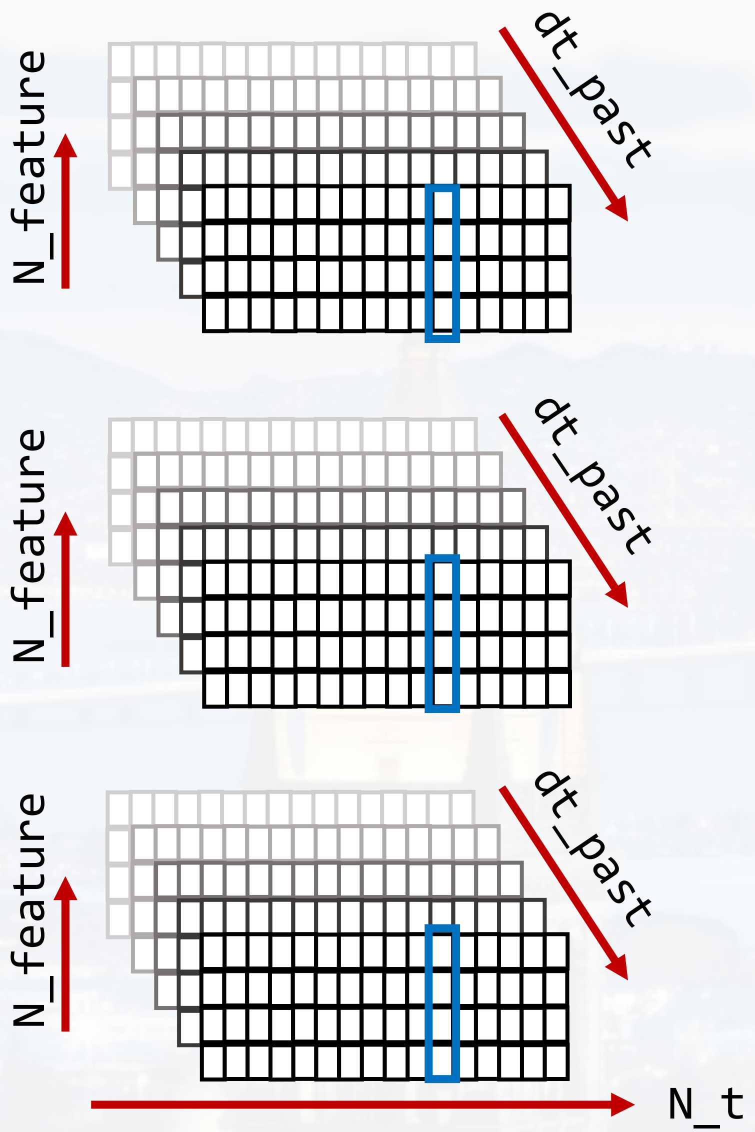




LSTM II:

LSTM + CNN

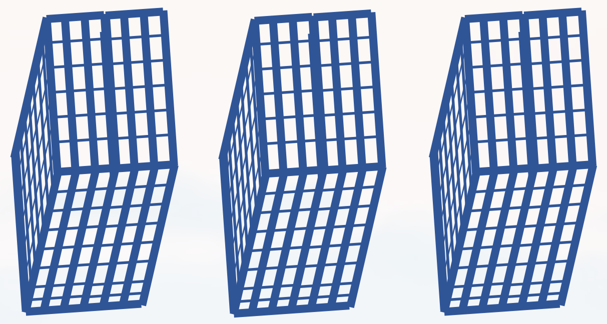
```
(N_images, N_x, N_y, N_color)
```

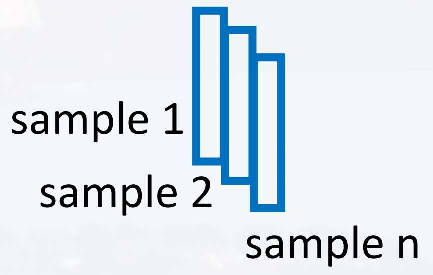


sample 1

sample 2

sample n

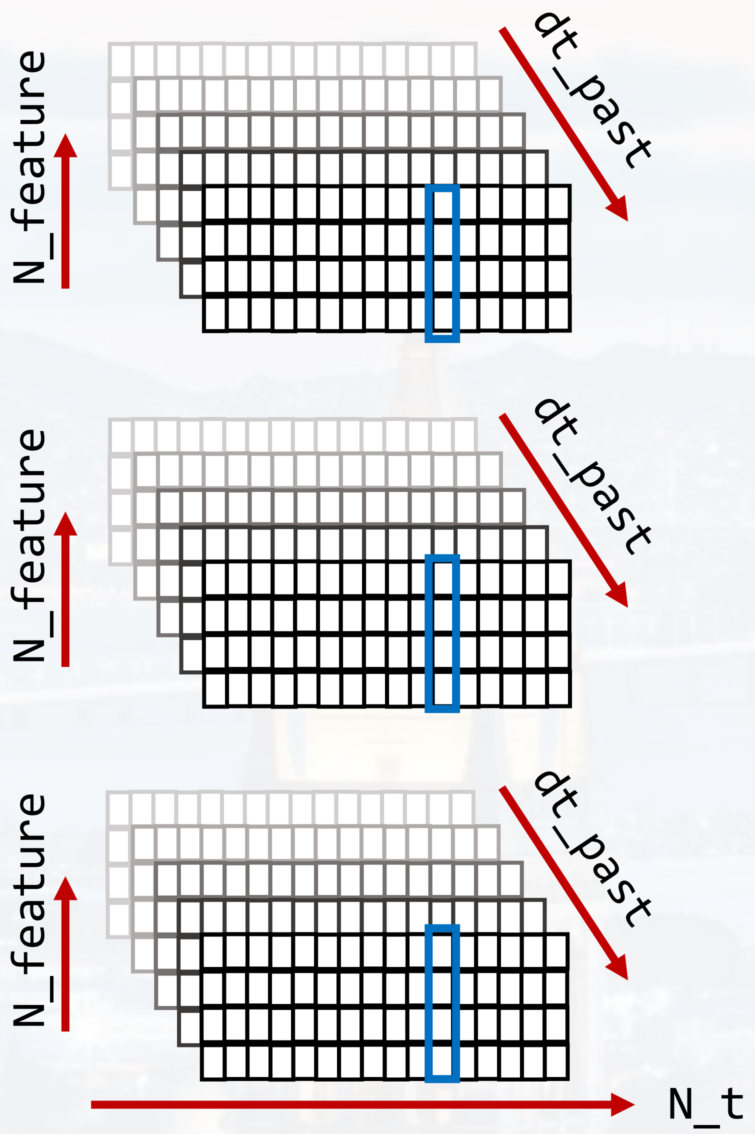




LSTM II:

LSTM + CNN

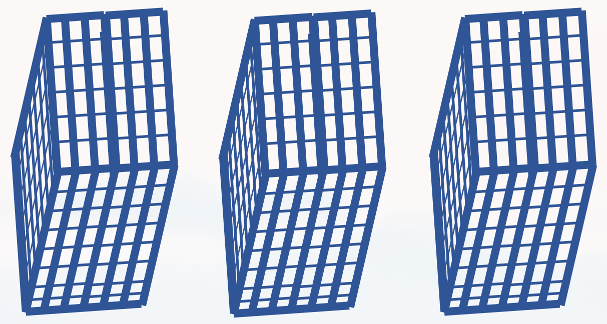
(N_images, N_x, N_y, N_color)

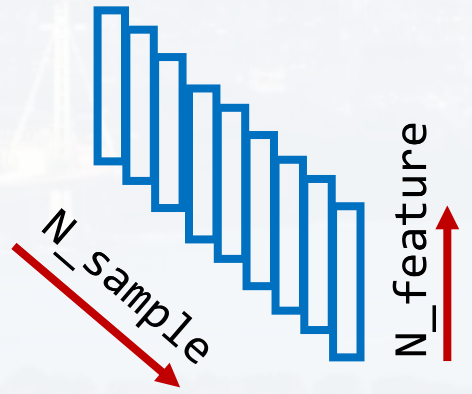


sample 1

sample 2

sample n



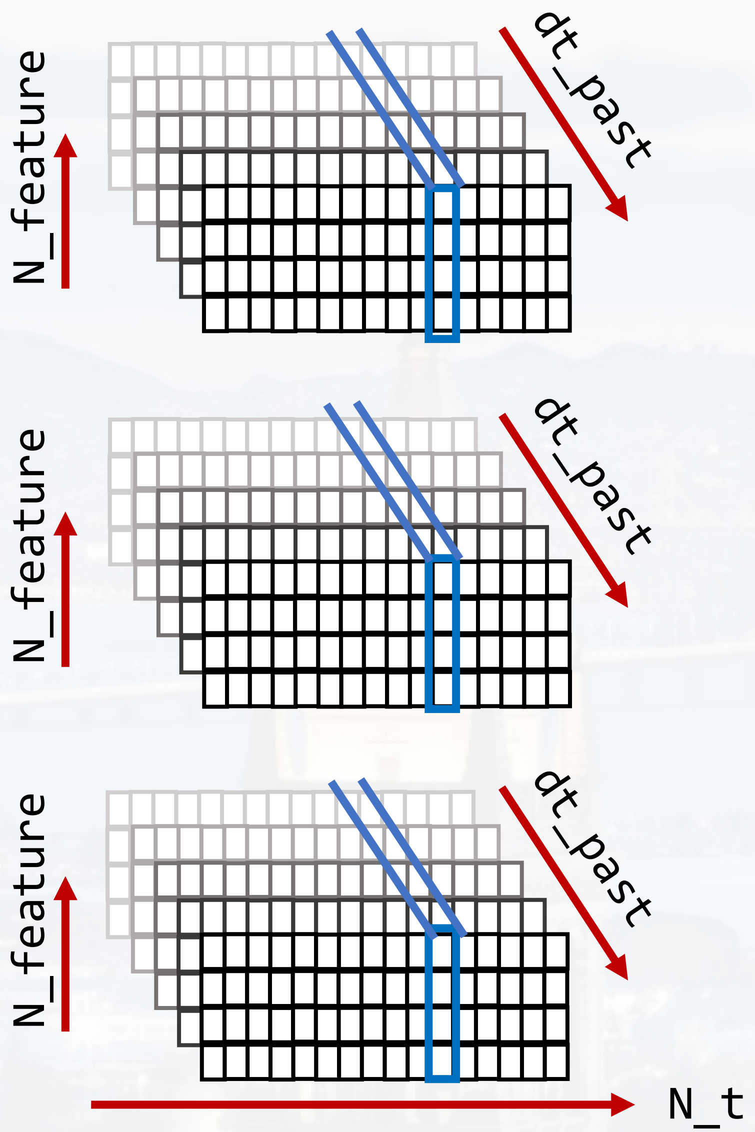


for **one** timepoint t

LSTM II:

LSTM + CNN

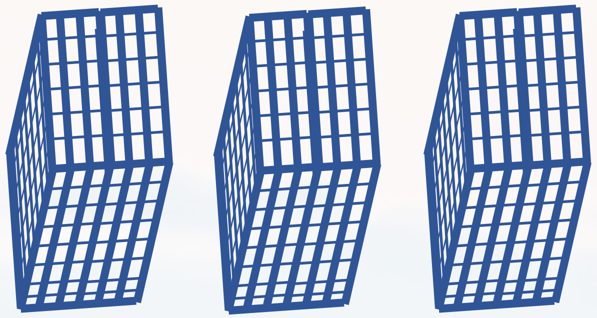
`(N_images, N_x, N_y, N_color)`

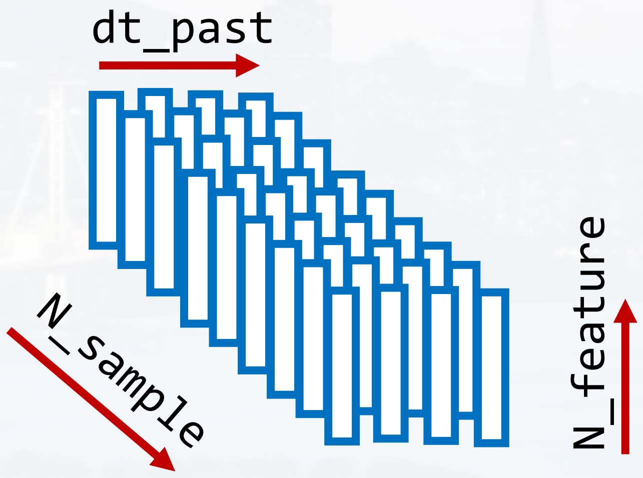


sample 1

sample 2

sample n



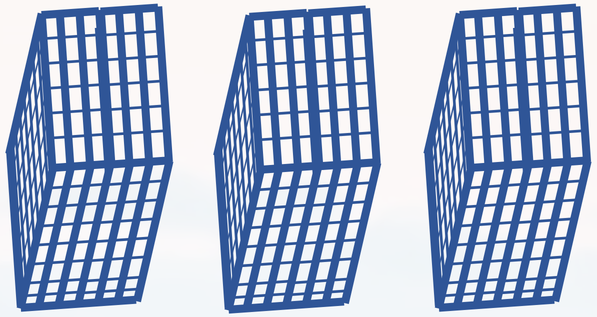


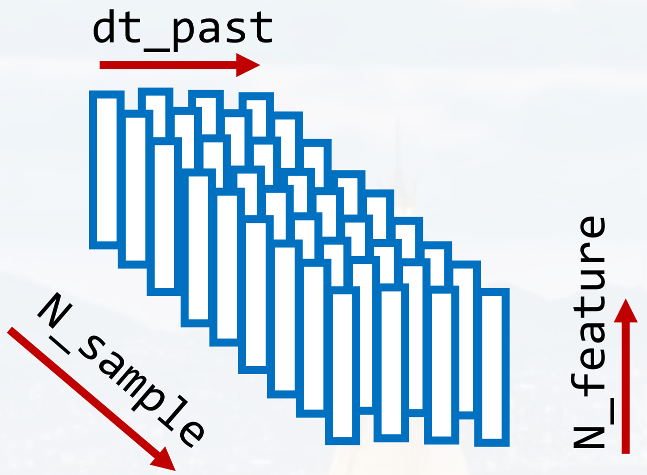
for **one** timepoint t

LSTM II:

LSTM + CNN

(N_images, N_x, N_y, N_color)





for **one** timepoint t

<u>regression:</u> **one** sample of N_features and dt_past

```
X = X.reshape((X.shape[0], N_samples, dt_past, n_features))
```

LSTM II:

LSTM + CNN

```
X = X.reshape((X.shape[0], N_samples, dt_past, n_features))

model = Sequential()
model.add(TimeDistributed(Conv1D(filters = 64, kernel_size = 3,\
                              activation = 'relu'),\
                              input_shape = (None, dt_past, n_features)))

model.add(TimeDistributed(MaxPooling1D(pool_size = 2)))

model.add(TimeDistributed(Flatten()))

model.add(LSTM(n_neurons, input_shape = (dt_past, n_features),\
                              activation = 'tanh'))
model.add(Dense(dt_futu))

opt = optimizers.Adam()
model.compile(loss = 'mean_squared_error', optimizer = opt)

model.summary()
```

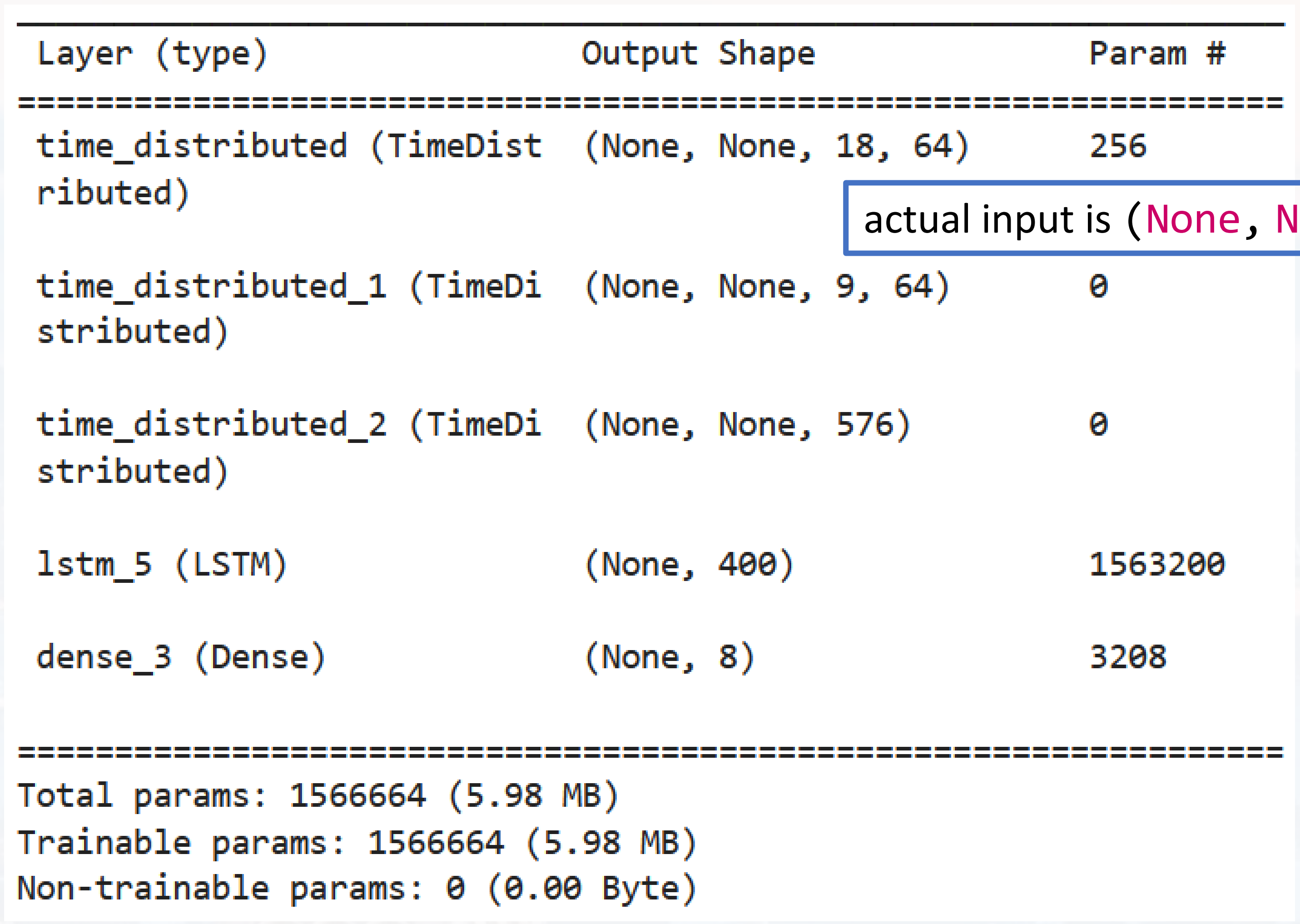
1D filter along time coordinate

takes care of maintaining matrix orientation

actual input is (None, None, dt_past, n_features)

LSTM II:

LSTM + CNN

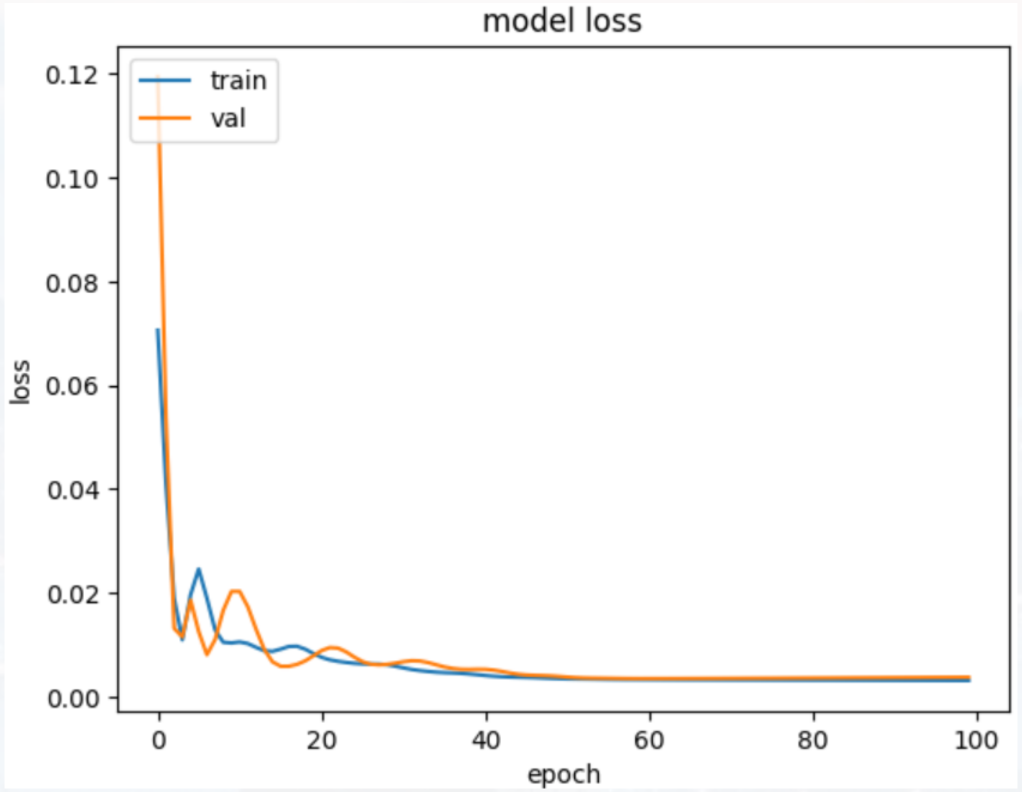


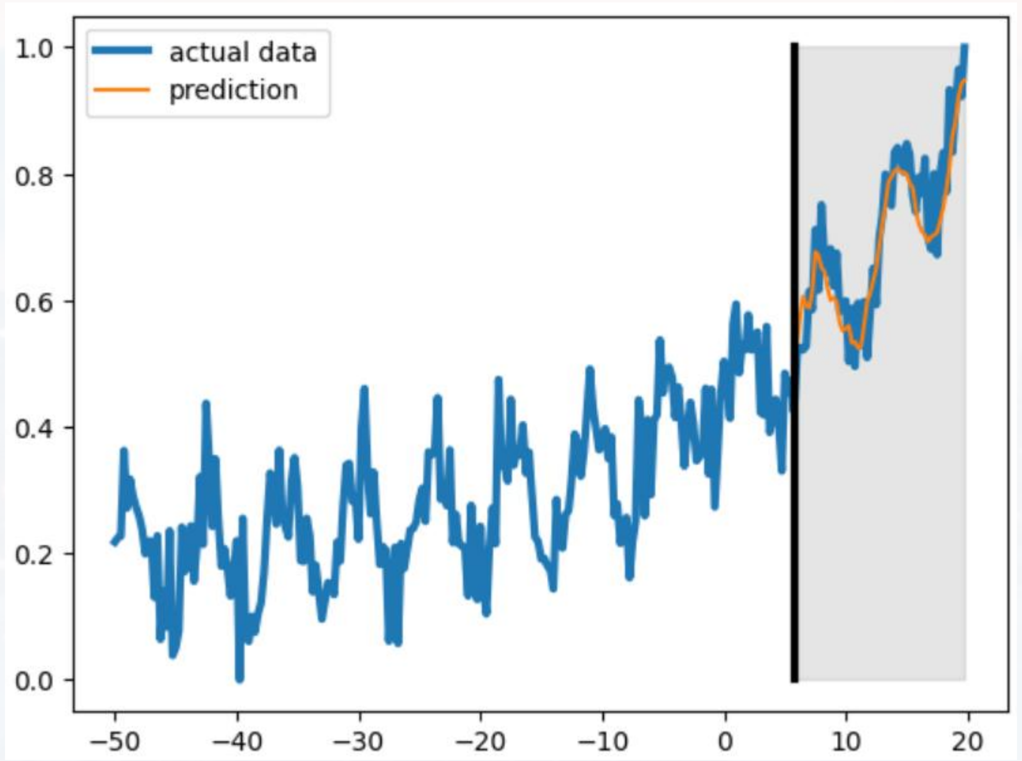
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDist  (None, None, 18, 64)      256
 ributed)

 time_distributed_1 (TimeDi  (None, None, 9, 64)       0
 stributed)

 time_distributed_2 (TimeDi  (None, None, 576)         0
 stributed)

 lstm_5 (LSTM)               (None, 400)               1563200

 dense_3 (Dense)             (None, 8)                 3208

=================================================================
Total params: 1566664 (5.98 MB)
Trainable params: 1566664 (5.98 MB)
Non-trainable params: 0 (0.00 Byte)
```

actual input is (None, None, dt_past, n_features)

LSTM II:

LSTM + CNN

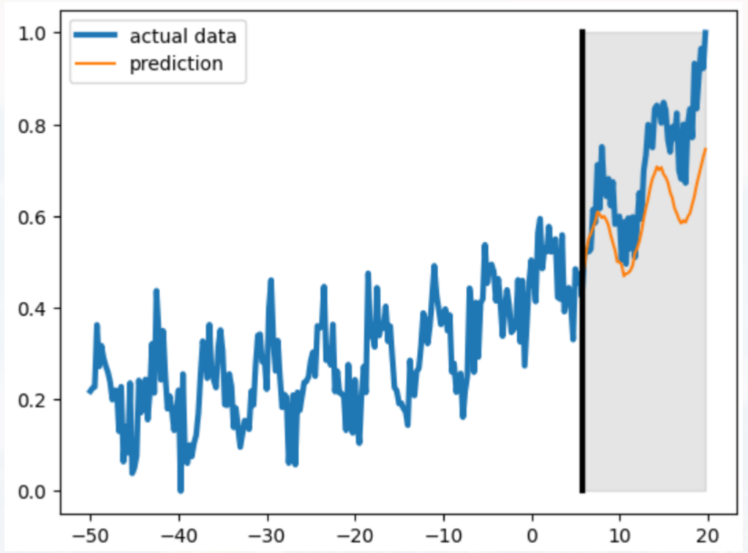




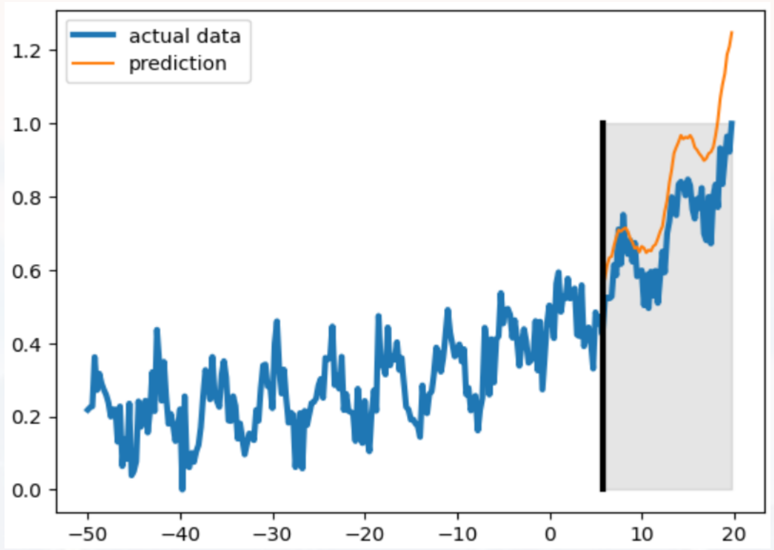
LSTM II:

LSTM + CNN

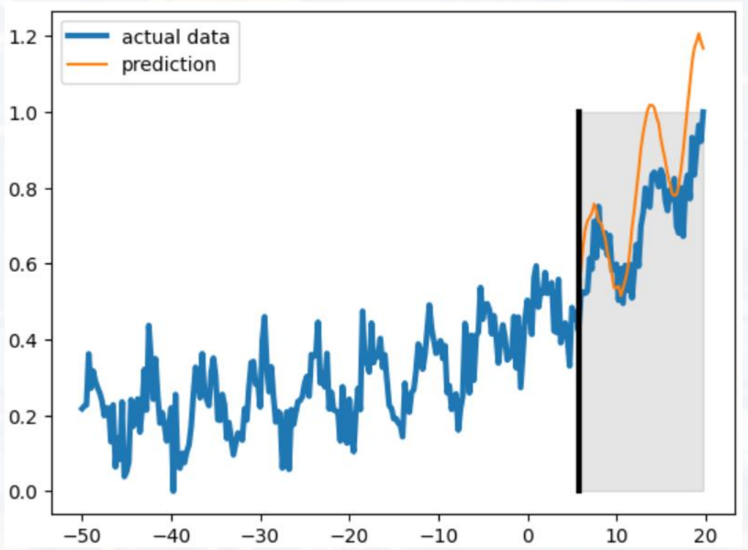
vanilla



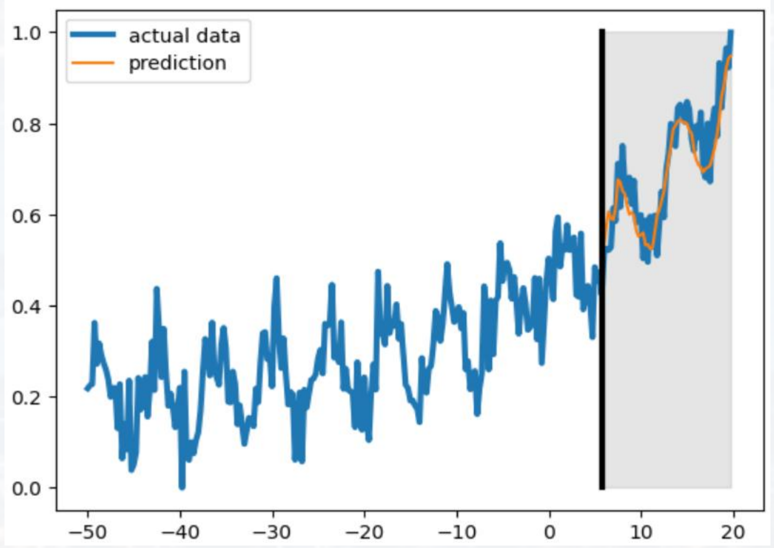
bidirectional



stacked



LSTM + CNN



LSTM II:

LSTM + CNN

<u>classification:</u> **N** samples of N_features and dt_past = Length_Seq

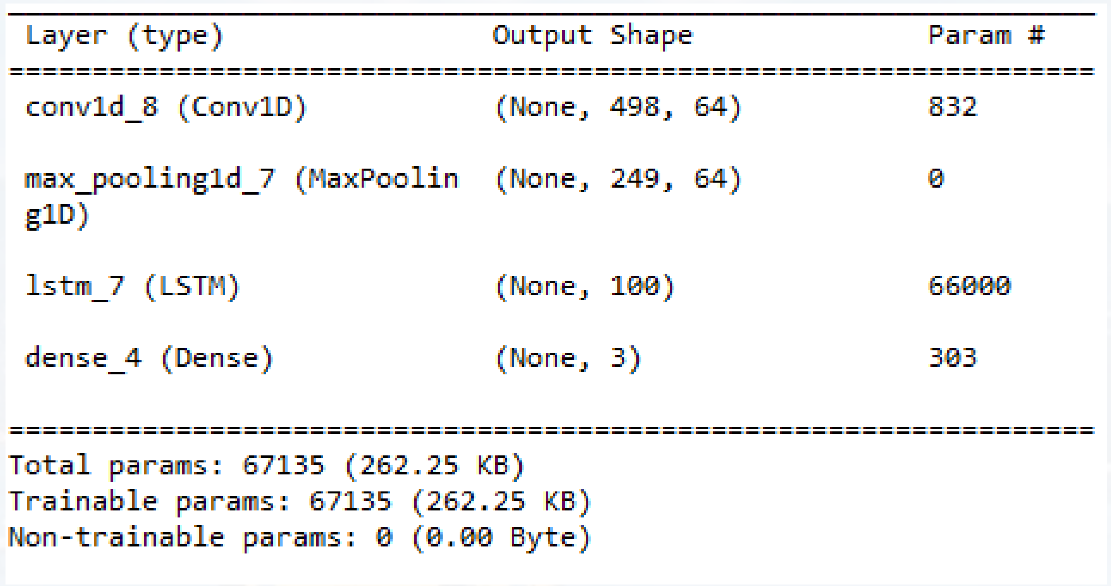
1D filter along time coordinate = LengthSeq

```
[N_sample, LengthSeq, N_features] = X.shape

model = Sequential()
model.add(Conv1D(filters = 64, kernel_size = 3, activation = 'relu',\
                                              input_shape = (LengthSeq, N_features)))

model.add(MaxPooling1D(pool_size = 2))


model.add(LSTM(n_neurons, activation = 'tanh'))

model.add(Dense(Nclass, activation = 'softmax'))

opt = optimizers.Adam()
model.compile(loss = 'categorical_crossentropy', optimizer = opt,\
                                              metrics = ['accuracy'])

model.summary()
```

LSTM II: LSTM + CNN

<u>classification:</u> **N** samples of `N_features` and `dt_past = Length_Seq`

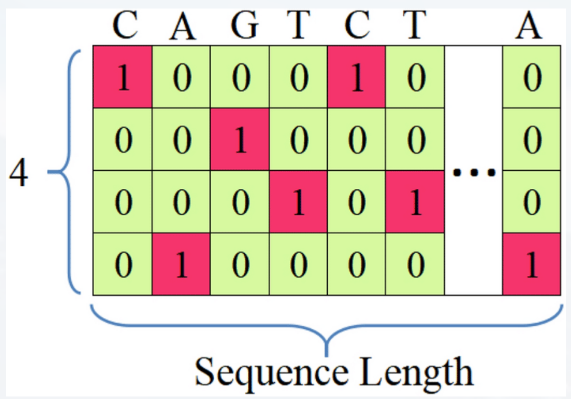
```
[N_sample, LengthSeq, N_features] = X.shape
```

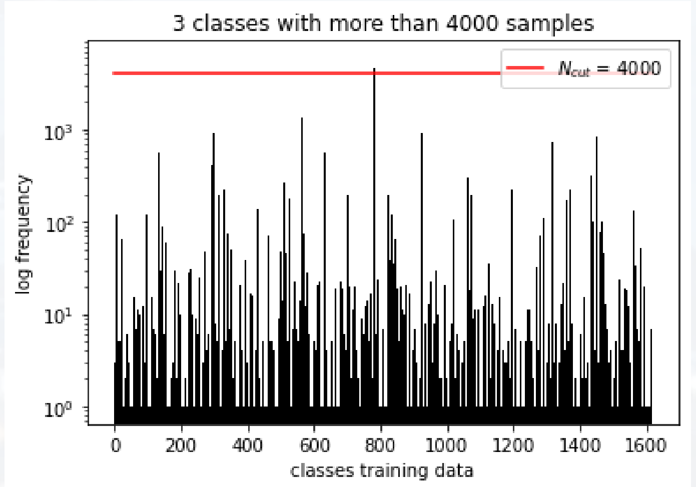


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_8 (Conv1D)           (None, 498, 64)           832

 max_pooling1d_7 (MaxPoolin  (None, 249, 64)           0
 g1D)

 lstm_7 (LSTM)               (None, 100)               66000

 dense_4 (Dense)             (None, 3)                 303

=================================================================
Total params: 67135 (262.25 KB)
Trainable params: 67135 (262.25 KB)
Non-trainable params: 0 (0.00 Byte)
```

for computational reasons:
- **three** classes
- **1k** samples total
- sequences cut to length **500**

barcode example

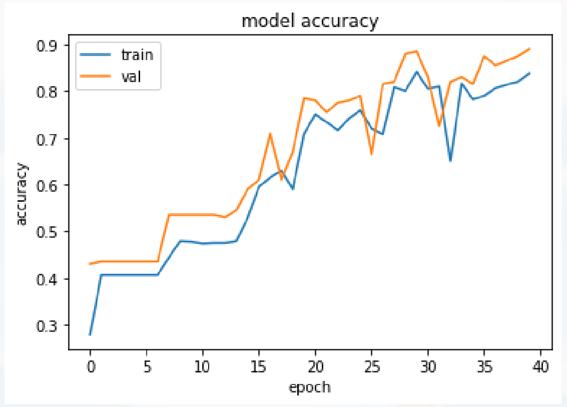


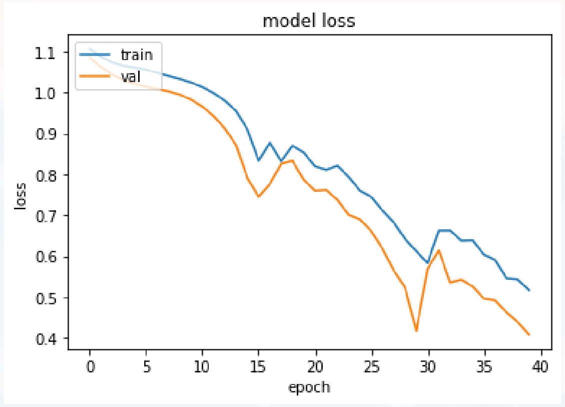


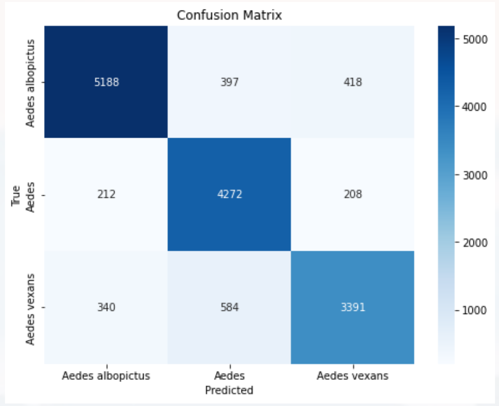
LSTM II:

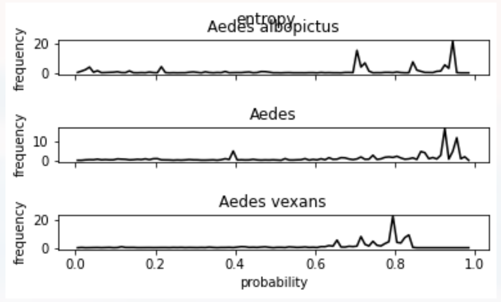
LSTM + CNN

LSTM

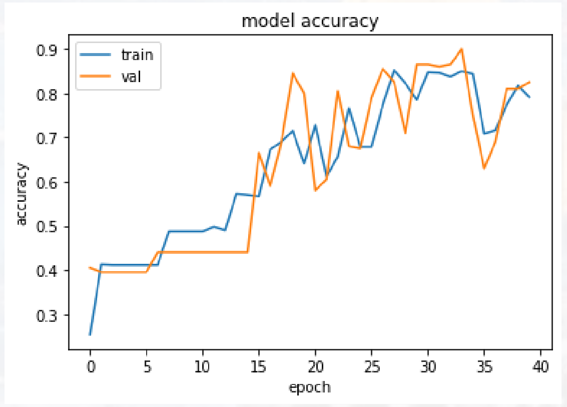


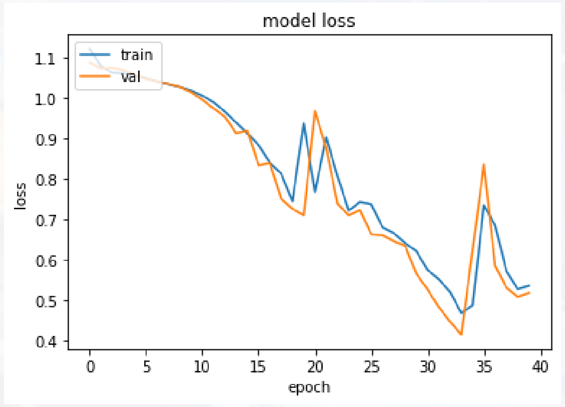


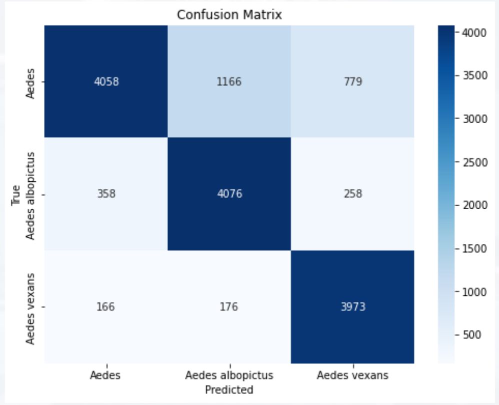


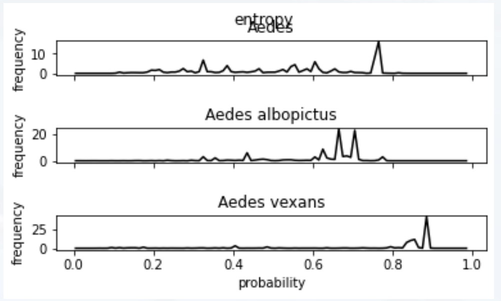


LSTM+CNN









Thank you for your attention!