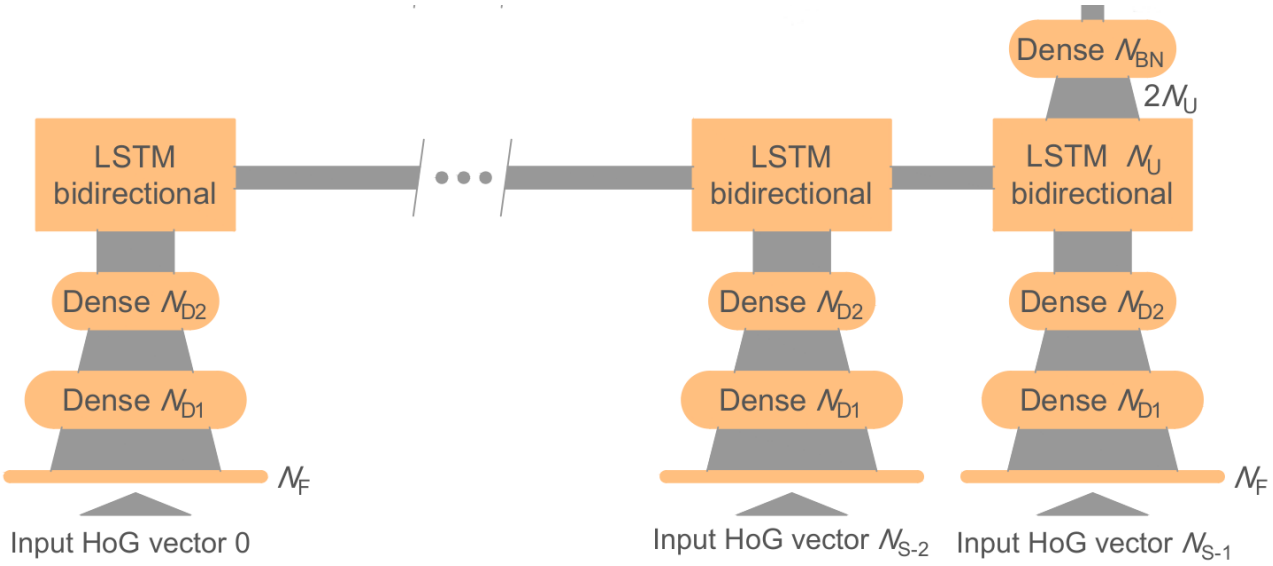
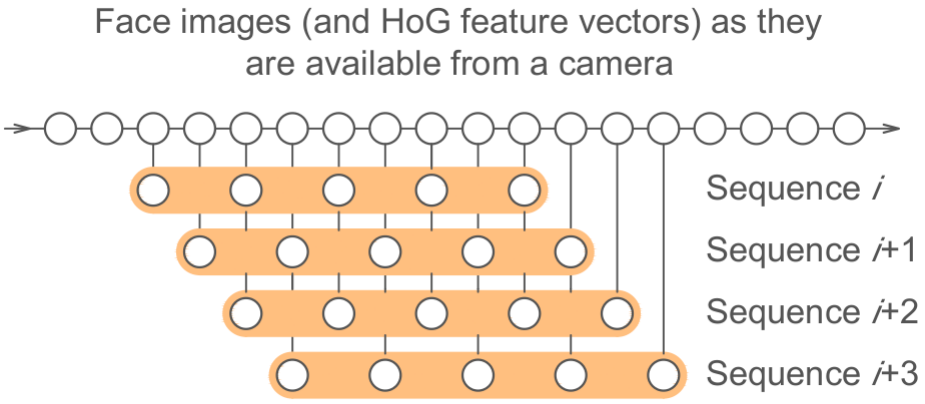
Detekce jednotlivých činností




```
self.inference_net = tf.keras.Sequential(
    [    #Encoder
         tf.keras.layers.InputLayer(input_shape=(numTimeSteps, hogSize)),
         tf.keras.layers.TimeDistributed(Dense(numFeaturesIntern0, activation='relu')),
         tf.keras.layers.TimeDistributed(Dense(numFeaturesIntern1)),
         tf.keras.layers.Bidirectional(LSTM(numVarsInLSTM, activation='relu')),
         tf.keras.layers.Dense(latent_dim), # No activation
    ]
)
```

Vodorovně jsou různé časy. Co tam ale posílat?
- celé obrazy (oblasti zájmu v obrazech)?
- univerzální příznaky (HoG nad oblastí zájmu) ?
- Příznaky odvozené z kostry nebo klíčových bodů (z hloubkových/RGB/IR obrazů)?

Jaká má být architektura sítě? (Mnoho možností.)
Docela náročné na trénovací data (délka trvání akce)



Příznaky odvozené z kostry a klíčových bodů.

Nástroje na detekci kostry často CNN z RGB.

OpenPose ale i jiné (promítnout real time ukázku).

Ale např. i naše z hloubkových senzorů (promítnout two_in_car, outdoor_ride3.avi).

Analýza muže být i podrobnější (result_wide_cam2_BFH.avi), OpenPose

Některé výsledky: result_skeleton_102.avi, vidFace1.avi

HoG: anomal_logi_face_michael_result.mp4, připomenout HoG

Z celých obrazů: dost problém

Detelce anomálií – inspirace segmentací obrazu

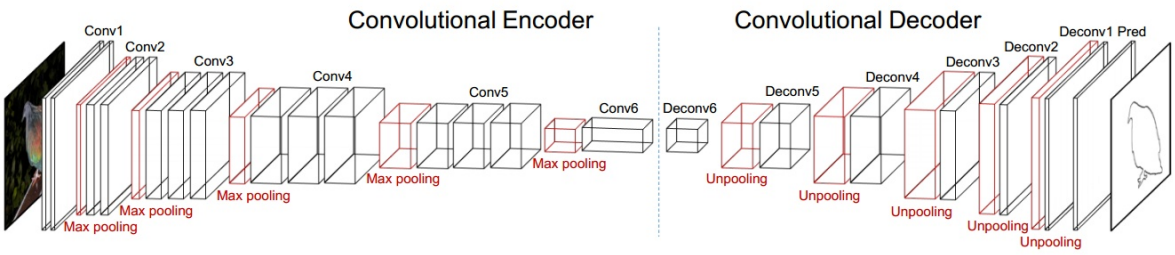
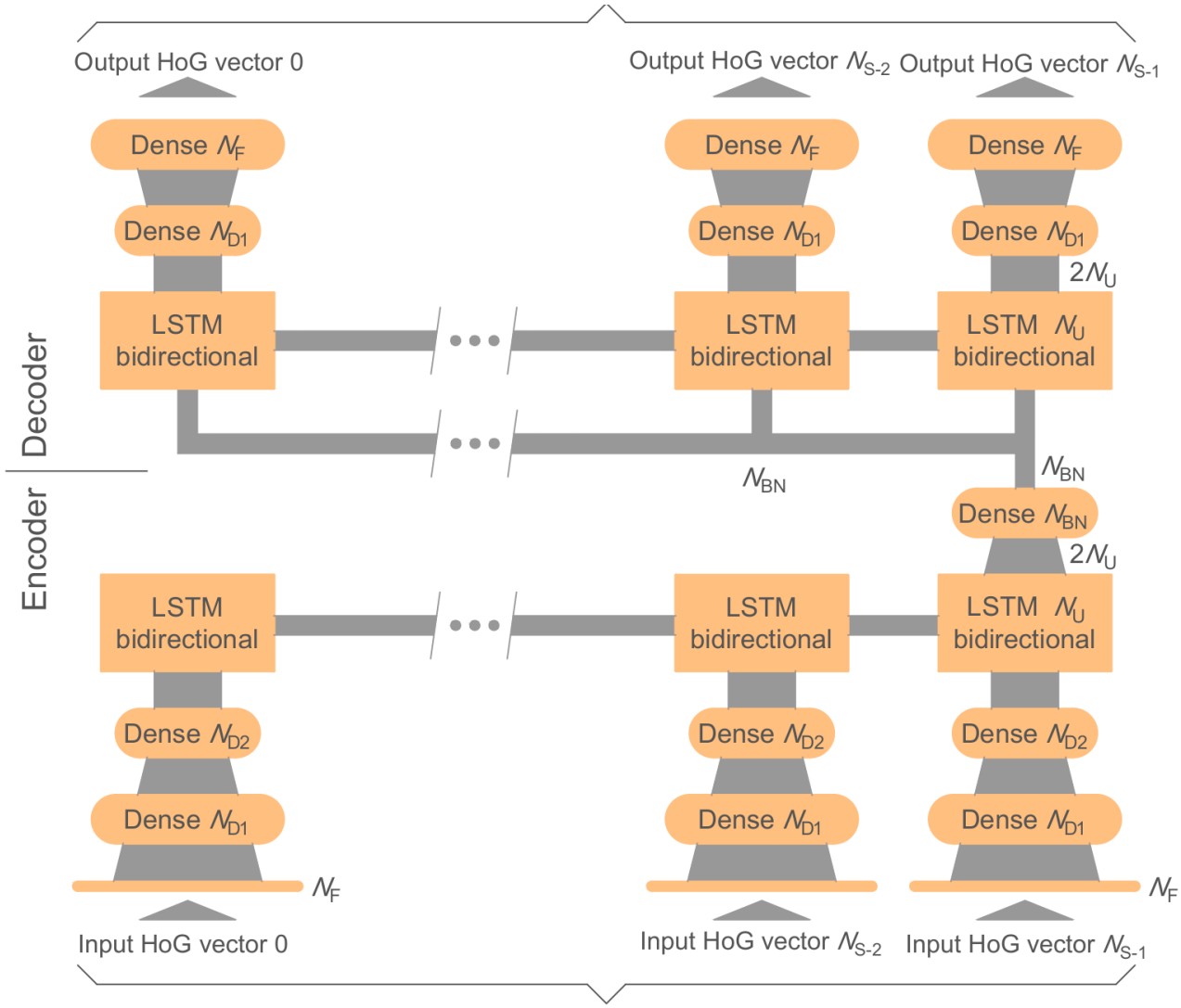


Figure 2. Architecture of the proposed fully convolutional encoder-decoder network.



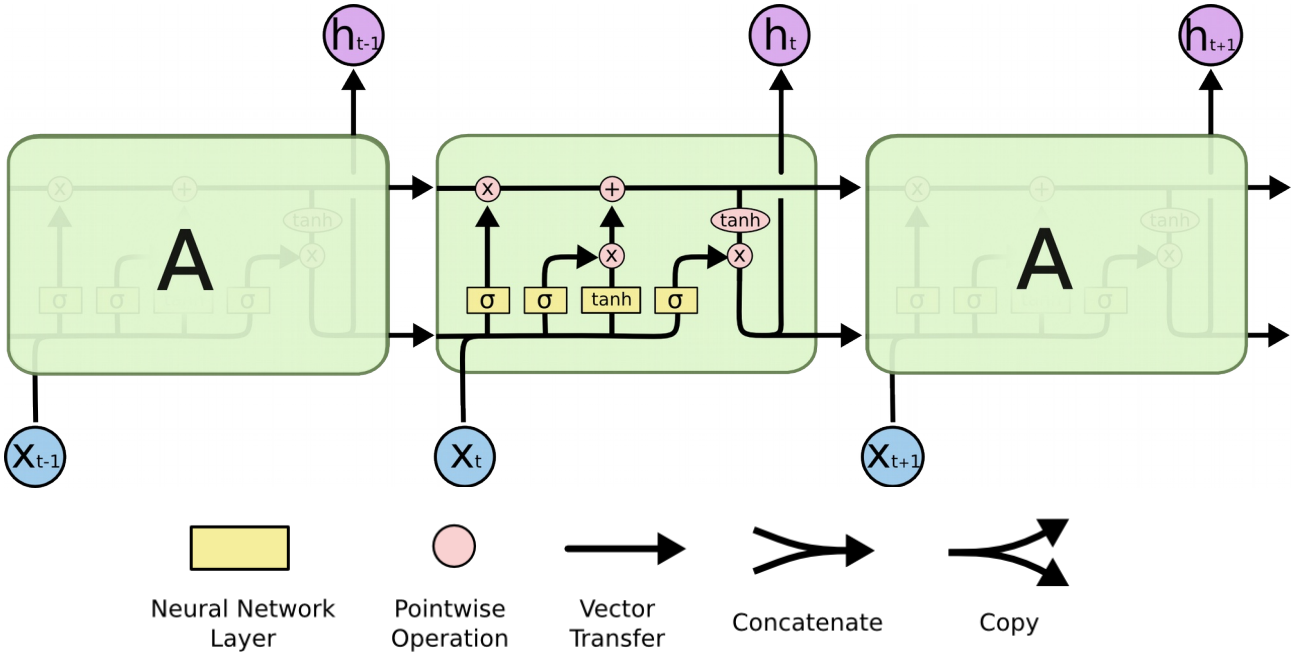

```
self.inference_net = tf.keras.Sequential(
  [   #Encoder
      tf.keras.layers.InputLayer(input_shape=(numTimeSteps, hogSize)),
      tf.keras.layers.TimeDistributed(Dense(numFeaturesIntern0, activation='relu')),
      tf.keras.layers.TimeDistributed(Dense(numFeaturesIntern1)),
      tf.keras.layers.Bidirectional(LSTM(numVarsInLSTM, activation='relu')),
      tf.keras.layers.Dense(latent_dim), # No activation
  ]
)

self.generative_net = tf.keras.Sequential(
  [   #Decoder
      tf.keras.layers.InputLayer(input_shape=(latentDim)),
      tf.keras.layers.RepeatVector(numTimeSteps),
      tf.keras.layers.Bidirectional(LSTM(numVarsInLSTM, activation='relu', return_sequences=True)),
      tf.keras.layers.TimeDistributed(Dense(numFeaturesIntern0, activation='relu')),
      tf.keras.layers.TimeDistributed(Dense(hogSize)),
  ]
)
```

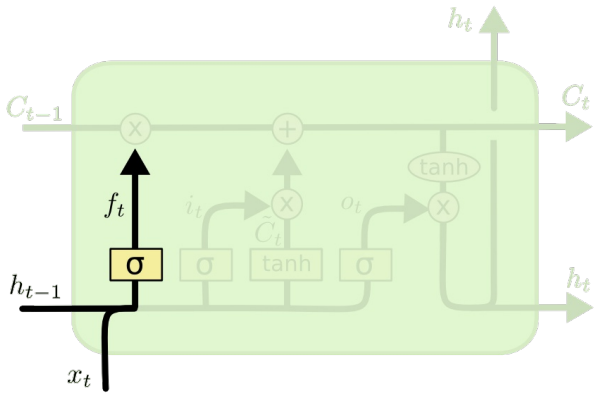
Ne vždy je žádoucí a možné rozpoznávat všechny typy činností (nelze vytvořit odpovídající trénovací množiny). Někdy stačí říct, že něco je obvyklé a něco neobvyklé.

Myšlenka: Co síť (autoenkodér) už někdy viděla a co se naučila rekonstruovat je normální, ostatní je nenormální (co neumí rekonstruovat). Míra nenormálnost je tedy dána velikostí chyby rekonstrukce.
driving_anomalies_eda_01.mp4

Long Short-Term Memory Recurrent Neural Network

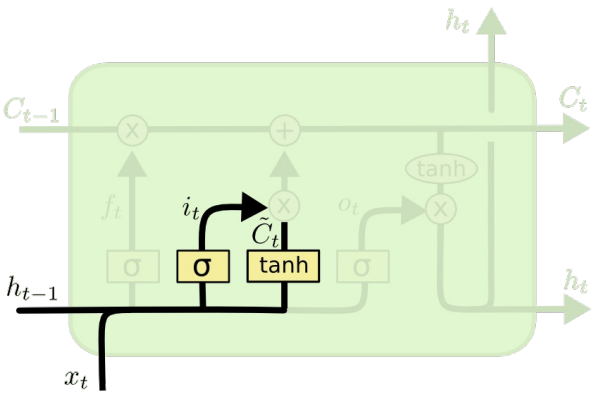
http://colah.github.io/posts/2015-08-Understanding-LSTMs/





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

The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at $ht-1$ and xt, and outputs a number between 0 and 1 for each number in the cell state $Ct-1$. A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

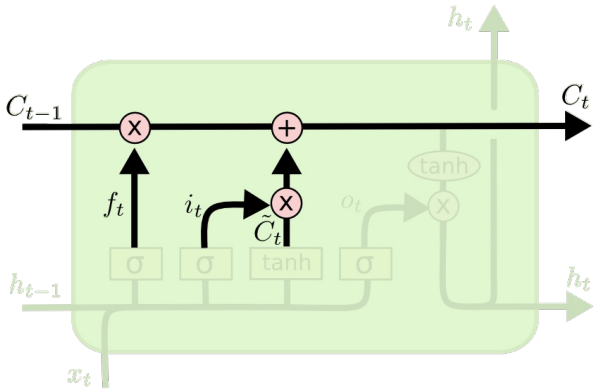


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

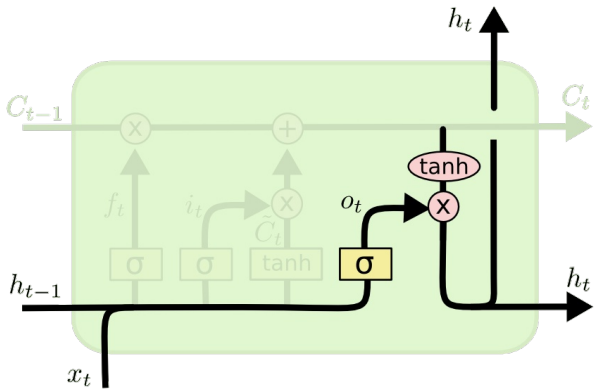
The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a

vector of new candidate values, $C\sim t$, that could be added to the state. In the next step, we'll combine these two to create an update to the state.



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

Updating the old cell state, *Ct*−1, into the new cell state *Ct.* We multiply the old state by *ft* forgetting the things we decided to forget earlier. Then we add *it*∗*C*~*t*. This is the new candidate values, scaled by how much we decided to update each state value.



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

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. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through `tanh` (to push the values to be between `-1` and `1`) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.