

PATTERN RECOGNITION

LSTM

Présenté par Hjaiej Mohamed



LSTMS

GENERALITE

Les réseaux LSTM sont une variation des réseaux de neurones récurrents (RNN). Ils ont été proposés par [Jürgen Schmidhuber](#) et [Sepp Hochreiter](#) dans l'article intitulé "LONG SHORT-TERM MEMORY", publié [en 1997](#) dans le journal [Neural Computation](#)

Les réseaux de neurones récurrents (RNN), et en particulier les Long Short-Term Memory (LSTM), sont souvent utilisés [en pattern recognition](#) en raison de leur capacité à traiter [des séquences](#) de données et à [conserver des informations](#) à long terme.

Les LSTMs ont été conçus pour résoudre les problèmes du [Vanishing Gradient](#) dans les réseaux récurrentes classiques([RNNs](#))

LSTMS

+ VANISHING AND EXPLODING GRADIENTS! C'EST QUOI?

LES RÉSEAUX LSTM ONT ÉTÉ INVENTÉS PRINCIPALEMENT POUR RÉSOUDRE LES PROBLÈMES DES RNN, QUI SE MANIFESTENT ESSENTIELLEMENT SOUS CE QUE L'ON APPELLE LES 'VANISHING AND EXPLODING GRADIENTS'.

Exploding: Le problème des gradients explosifs fait référence à **l'augmentation** significative de la norme du gradient pendant l'entraînement (Back Propagation Through Time / la phase de changements des poids(apprentissage)).

- Somme des gradients \rightarrow infinie lorsque $t \rightarrow$ infinie (à long terme = grand nombre de timesteps)

Vanishing: Le problème de disparition des gradients fait référence à **la diminution** significative de la norme du gradient pendant l'entraînement et donc le model sera incapable d'apprendre les anciennes informations.

- Somme des gradients $\rightarrow 0$ lorsque $t \rightarrow$ infinie (à long terme = grand nombre de timesteps)
- $w[t] = w[t-1] - 0 = w[t-1] = w[t-k]$ avec k l'instant de début de l'apparition de vanishing problem

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \leq t \leq T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \leq k \leq t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \geq i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

Explication:

Le gradient à un instant t est un produit des gradients des instants précédents, donc il existe un risque à certains moments si :

- Chacun des gradients est inférieur à 1 (0.9, 0.66, ...) alors le produit va tendre vers 0.
- Chacun des gradients est supérieur à 1 (3, 17.9, 9.4, ...) alors le produit va tendre vers l'infini.

Source: On the difficulty of training Recurrent Neural Networks : par Razvan Pascanu en 2013
DOI:[10.1142/S0218488598000094](https://doi.org/10.1142/S0218488598000094)
Cité 6847 fois

LSTMS

+ VANISHING AND EXPLODING GRADIENTS! DEMO:

DEMONSTRATION THEORIQUES:

On a la formule de DG classique: $W \leftarrow W - \alpha \frac{\partial E}{\partial W}$ Avec $\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W}$ W: Poids E: Erreur total:PBTT Et: Erreur à instant t

On applique the 'Chaine RULE' sur l'un des gradient : exemple instant K on trouve

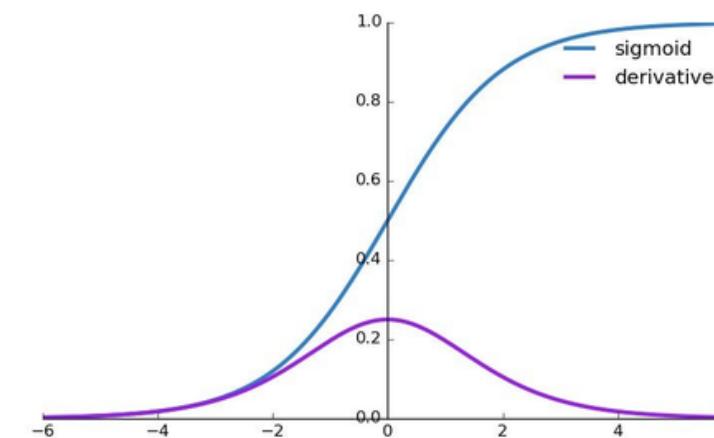
$$\begin{aligned}\frac{\partial E_k}{\partial W} &= \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \dots \frac{\partial c_2}{\partial c_1} \frac{\partial c_1}{\partial W} \\ &= \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^k \frac{\partial c_t}{\partial c_{t-1}} \right) \frac{\partial c_1}{\partial W}\end{aligned}$$

Nous sommes en train de résoudre le problème des produits qui cause le 'Vanishing problem' !!!

En RNN on a :cell state $c_t = \sigma(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t)$ Et donc $\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^k \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot W_{rec} \right) \frac{\partial c_1}{\partial W}$

Dans notre cas quelque soit la fonction d'activation f: $0 \leq f' \leq 1$

les courbes de la fonction Sigmoid et sa dérivée

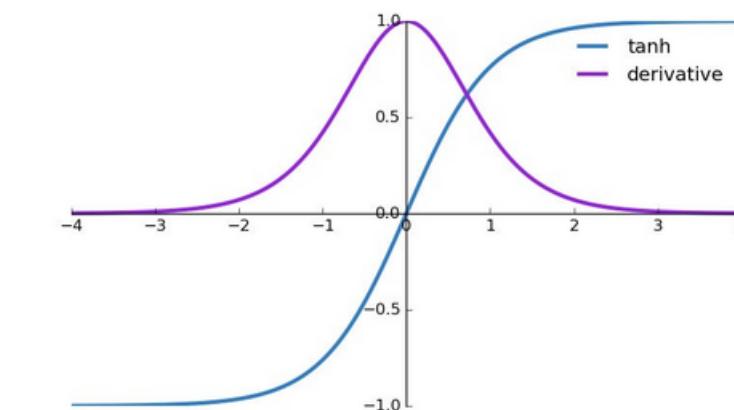


$$\frac{\partial E_k}{\partial W} \rightarrow 0 \quad \frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W} \rightarrow 0$$

Donc $W \leftarrow W - \alpha \frac{\partial E}{\partial W} \approx W$

=> Vanishing Gradient!!

les courbes de la fonction tanh et sa dérivée



LSTMS

+ VANISHING AND EXPLODING GRADIENTS! DEMO:

En RNN on a :cell state $c_t = \sigma(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t)$ Et donc $\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^k \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot W_{rec} \right) \frac{\partial c_1}{\partial W}$

En LSTM on a :cell state $c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t$

Et donc

$$\begin{aligned}\frac{\partial c_t}{\partial c_{t-1}} &= \frac{\partial}{\partial c_{t-1}} [c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t] \\ &= \frac{\partial}{\partial c_{t-1}} [c_{t-1} \otimes f_t] + \frac{\partial}{\partial c_{t-1}} [\tilde{c}_t \otimes i_t] \\ &= \frac{\partial f_t}{\partial c_{t-1}} \cdot c_{t-1} + \frac{\partial c_{t-1}}{\partial c_{t-1}} \cdot f_t + \frac{\partial i_t}{\partial c_{t-1}} \cdot \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial c_{t-1}} \cdot i_t\end{aligned}$$

donc

$$\frac{\partial c_t}{\partial c_{t-1}} = A_t + B_t + C_t + D_t$$

donc

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^k [A_t + B_t + C_t + D_t] \right) \frac{\partial c_1}{\partial W}$$

=> on a donc produit des sommes.

Avec le bon ajustement par les portes (gates) des LSTM, on peut résoudre le problème du vanishing gradient et éviter que dE ne tende vers 0. On remarque qu'il y a une opération d'ajout qui aide à choisir quelle information va être ajoutée à la mémoire à un instant t

LSTMS

+ VANISHING AND EXPLODING GRADIENTS! DEMO: ANNEXE

$$\frac{\partial c_t}{\partial c_{t-1}} = \sigma'(W_f \cdot [h_{t-1}, x_t]) \cdot W_f \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot c_{t-1}$$

$$+ f_t$$

$$+ \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$+ \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot i_t$$

$$A_t = \sigma'(W_f \cdot [h_{t-1}, x_t]) \cdot W_f \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot c_{t-1}$$

$$B_t = f_t$$

$$C_t = \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$D_t = \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot i_t$$

LSTMS + MOTIVATION

QUELQUES POINTS D'INTÉRÊT SPÉCIFIQUES À L'UTILISATION DES LSTM EN PATTERN RECOGNITION

Modélisation des **séquences temporelles et séquentielles** : Textes et Videos

Capaciter à **maintenir une mémoire à long terme** et à capter **de dépendances temporelles complexes**

Les portes présentes dans les unités LSTM (comme la **porte d'oubli** et la **porte d'entrée**) permettent au réseau d'apprendre **à ignorer** ou **à retenir** certaines informations en fonction du contexte.

Adaptabilité à **différentes longueurs de séquence** : Les LSTM sont conçus pour gérer **des séquences de longueurs variables**.

Applications variées : Les LSTM sont utilisés dans une variété de domaines, tels que la **reconnaissance de la parole**, la **traduction automatique**, la **prédiction de séries temporelles**, la **classification de texte** etc.

LSTMS + MOTIVATION

QUELQUES POINTS D'INTÉRÊT SPÉCIFIQUES À L'UTILISATION DES LSTMS AVEC CNNS

Grace à une architecture **hybride** CNN-LSTM on peut extraire des caractéristiques **spatiale** par les CNNs et caractéristiques **temporelles** avec LSTMs.(exemple : analyse et segmentation des séquences vidéos en temps réelle)

INCEPTION-TIME : ARCHITECTURE HYBRIDE (LSTM ET CNN) FOR TIME SERIES CLASSIFICATION

Proposé en 2020 par Hassan Ismail Fawaz.

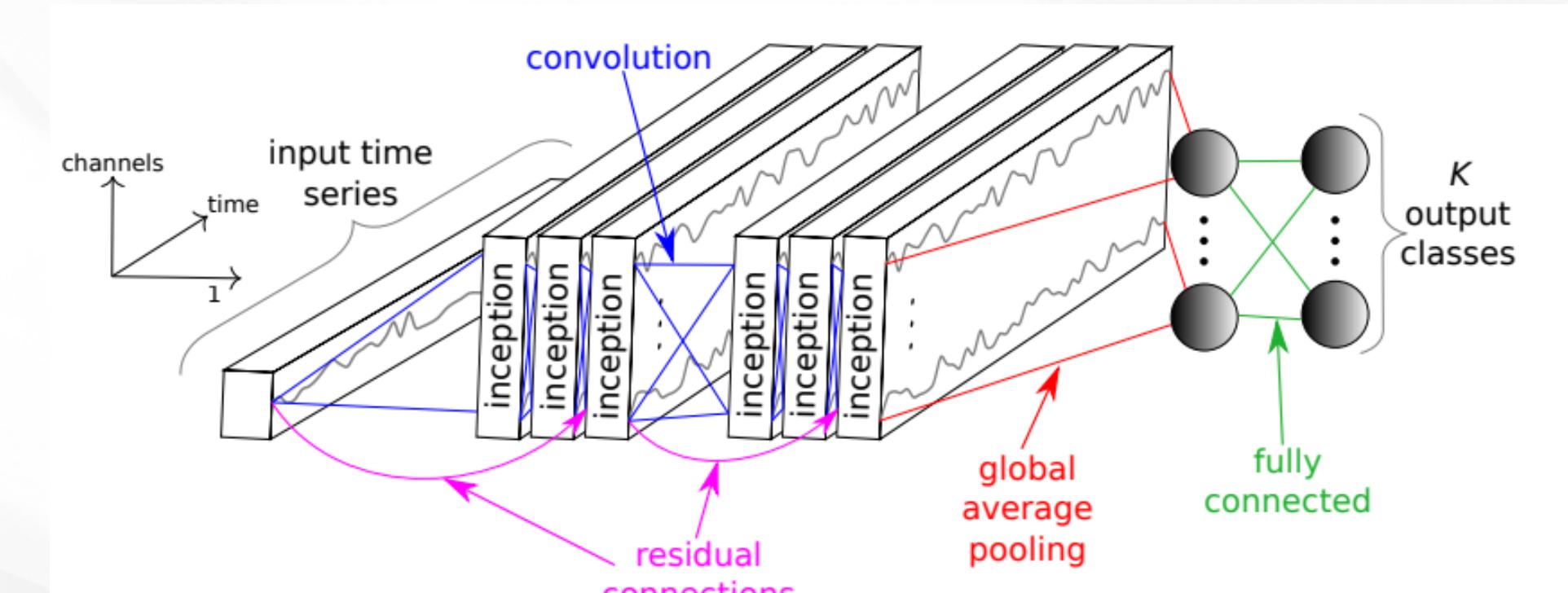


Fig. 1: Our Inception network for time series classification

Source: <https://doi.org/10.1007/s10618-020-00710-y>.

LSTMS + MOTIVATION

INCEPTION-TIME : ARCHITECTURE HYBRIDE (LTSM ET CNN) FOR TIME SERIES CLASSIFICATION

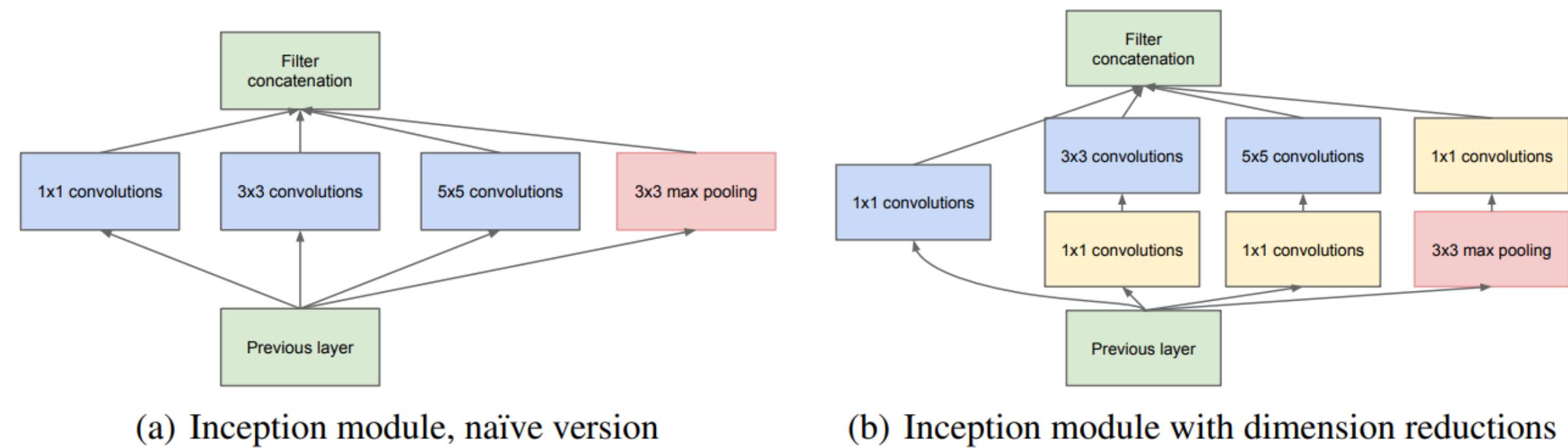


Figure 2: Inception module

Paper :Going deeper with convolutions

Source: <https://doi.org/10.48550/arXiv.1409.4842>

Cité 56610 fois

LSTMS + MOTIVATION

INCEPTION-TIME : ARCHITECTURE HYBRIDE (LTSM ET CNN) FOR TIME SERIES CLASSIFICATION

```
from tensorflow.keras import layers
from keras.models import Model
from tensorflow.keras.utils import plot_model

def _inception(input_layer):
    # layer1
    conv1_1 = layers.Conv1D(filters=32,kernel_size=1,padding='same',activation='relu')(input_layer)
    # layer2
    conv2_1 = layers.Conv1D(filters=32,kernel_size=1,padding='same',activation='relu')(input_layer)
    conv2_2 = layers.Conv1D(filters=32,kernel_size=3,padding='same',activation='relu')(conv2_1)
    # layer2
    conv3_1 = layers.Conv1D(filters=32,kernel_size=3,padding='same',activation='relu')(input_layer)
    conv3_2 = layers.Conv1D(filters=32,kernel_size=3,padding='same',activation='relu')(conv3_1)
    # layer3
    pool4_1 = layers.MaxPooling1D(3,strides=2,padding='same')(input_layer)
    conv4_2 = layers.Conv1D(filters=32,kernel_size=5,padding='same',activation='relu')(pool4_1)

    concat= layers.concatenate([conv1_1,conv2_2,conv3_2], axis=2)
    return concat

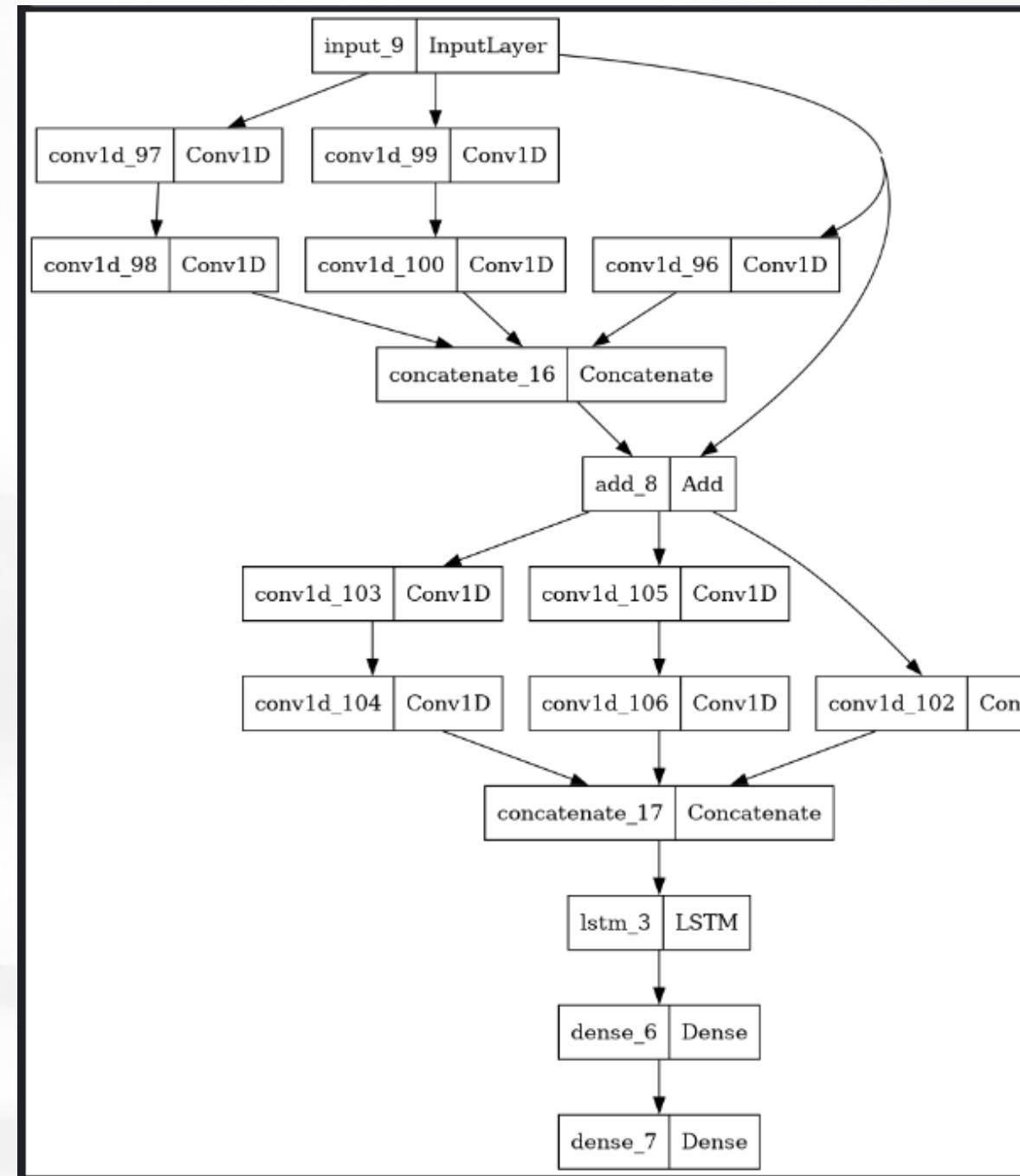
def _shortcut_layer(input_layer,output_layer):
    return layers.Add()([input_layer,output_layer])
```

```
input_layer      = layers.Input(shape=(100,1))
inception1_block1 = _inception(input_layer)
shortcut_layer   = _shortcut_layer(inception1_block1,input_layer)
inception1_block2 = _inception(shortcut_layer)
# global_averge_layer = layers.GlobalAveragePooling1D()(inception1_block2)
lstm            = layers.LSTM(100,return_sequences=True)(inception1_block2)
fc1             = layers.Dense(100,activation='relu')(lstm)
output_layer    = layers.Dense(10,activation='softmax')(fc1)

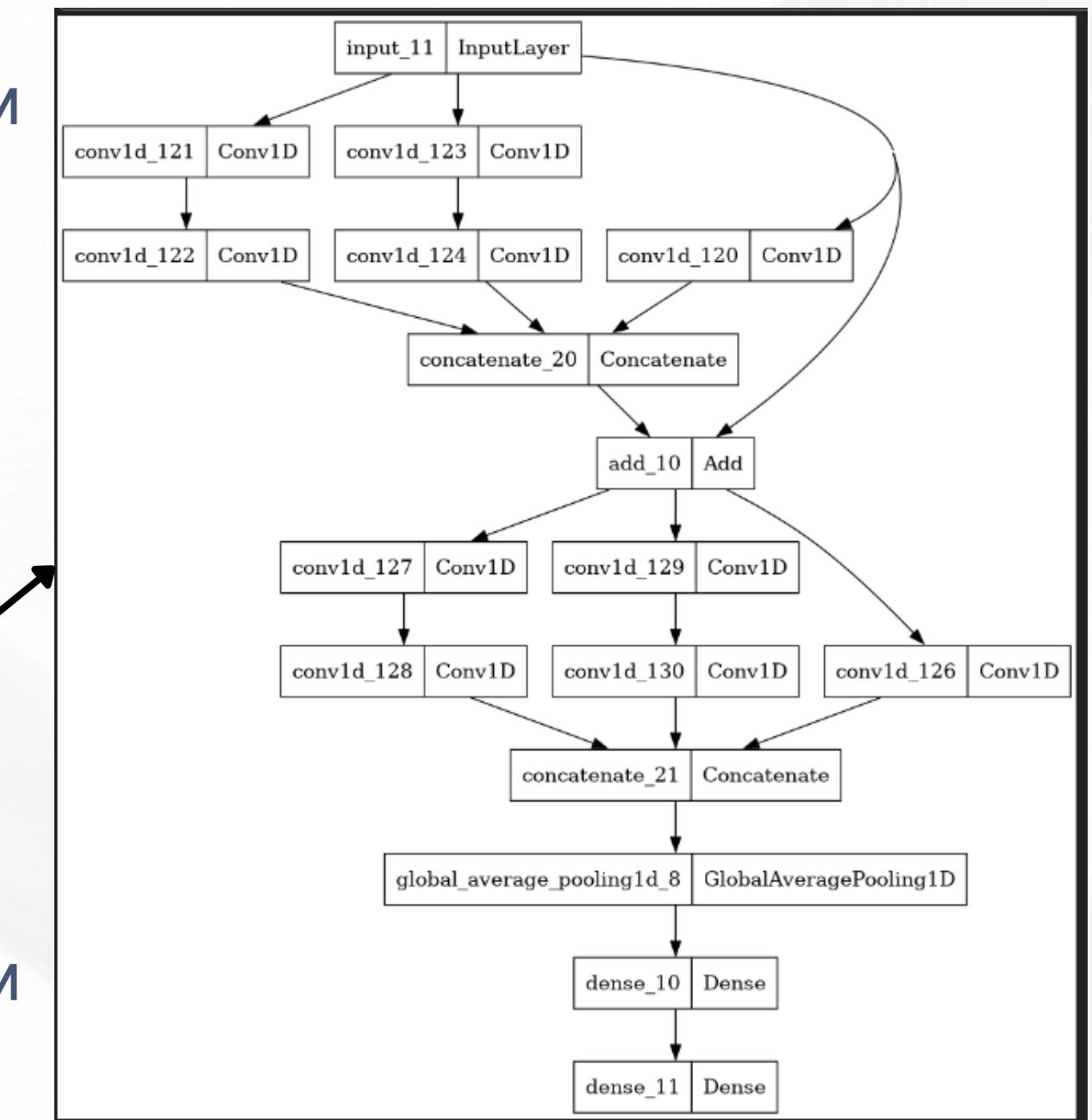
model = Model(inputs=input_layer,outputs=output_layer)
model.compile(optimizer='adam',loss="binary_crossentropy" ,metrics=["accuracy"])
plot_model(model)
```

LSTMS + MOTIVATION

INCEPTION-TIME : ARCHITECTURE HYBRIDE (LSTM ET CNN) FOR TIME SERIES CLASSIFICATION : **ARCHETECTURE DETAILLÉ**



INCEPTION-TIME MODIFIÉ :
2 INCEPTION MODELS + LSTM
+ 2 FC



INCEPTION-TIME (PAPIER):
2 INCEPTION MODELS + LSTM
+ 2 FC

LSTMS + MOTIVATION

QUELQUES POINTS D'INTÉRÊT SPÉCIFIQUES À L'UTILISATION DES LSTMS AVEC CNNS

CONVLSTM : EXEMPLE AVEC PROBLEME DE PREDICTION IMMEDIATE (NOWCASTING) AVEC MOVING-MNIST DATASET

Model	Number of parameters	Cross entropy
FC-LSTM-2048-2048	142,667,776	4832.49
ConvLSTM(5x5)-5x5-256	13,524,496	3887.94
ConvLSTM(5x5)-5x5-128-5x5-128	10,042,896	3733.56
ConvLSTM(5x5)-5x5-128-5x5-64-5x5-64	7,585,296	3670.85
ConvLSTM(9x9)-1x1-128-1x1-128	11,550,224	4782.84
ConvLSTM(9x9)-1x1-128-1x1-64-1x1-64	8,830,480	4231.50

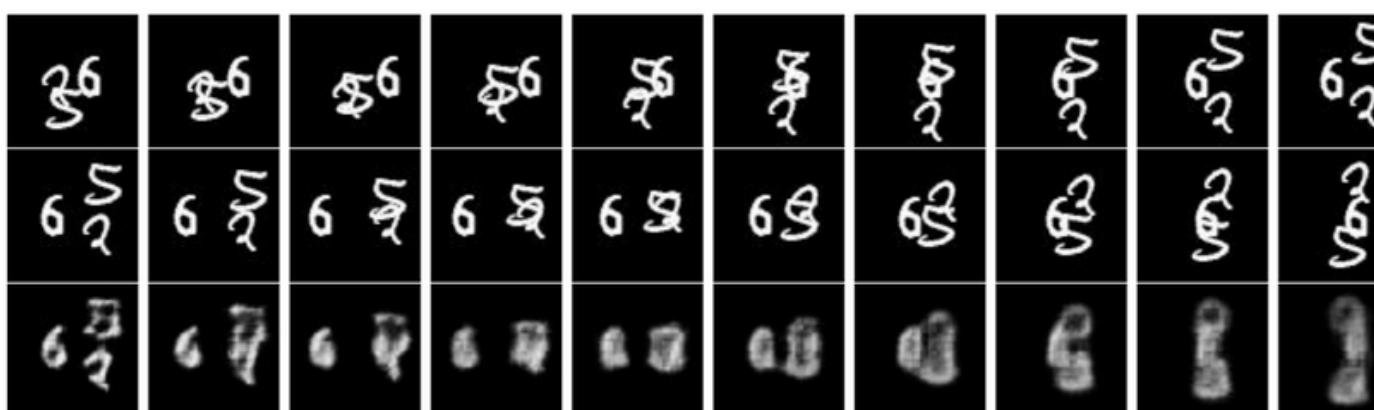


Figure 4: An example showing an “out-of-domain” run. From left to right: input frames; ground truth; prediction by the 3-layer network.

Papier: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting

Source: <https://doi.org/10.48550/arXiv.1506.04214>

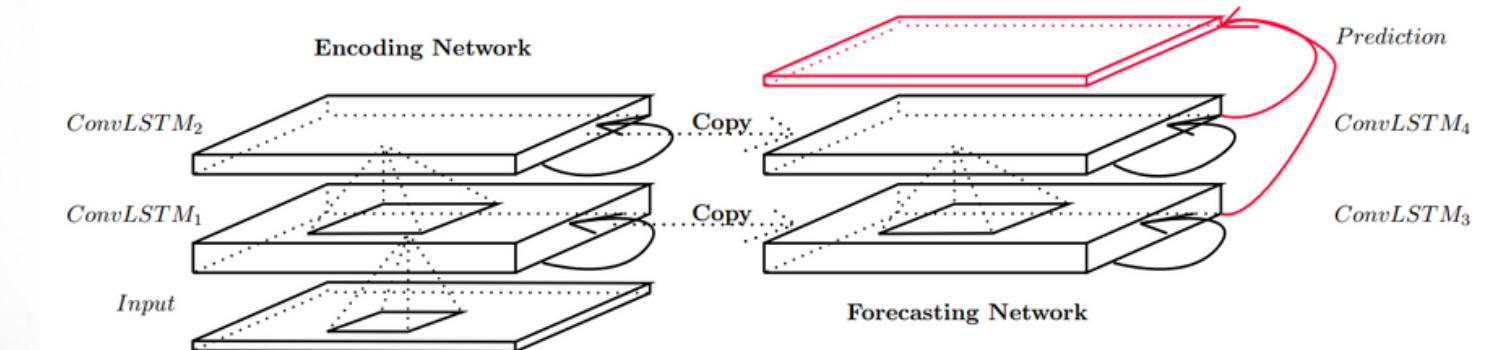
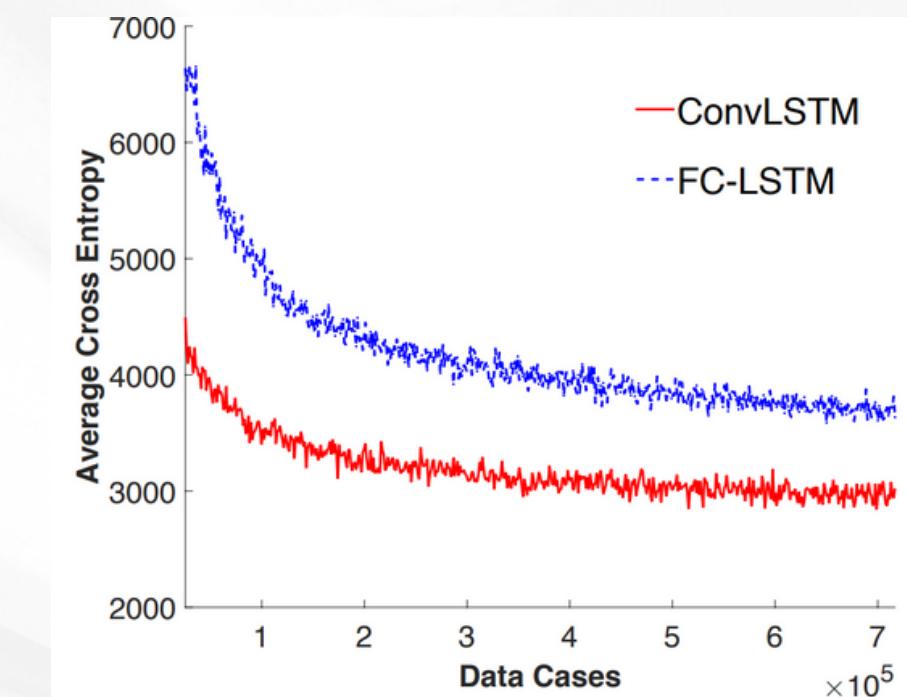


Figure 3: Encoding-forecasting ConvLSTM network for precipitation nowcasting



11: Comparison of the 3-layer ConvLSTM and FC-LSTM in the online setting.

ConvLSTM ont moins loss que FC-LSTM

LSTMS + MOTIVATION

QUELQUES POINTS D'INTÉRÊT SPÉCIFIQUES À L'UTILISATION DES LSTMS AVEC CNNS

Two-Stream Convolutional Networks with LSTMs (Two-Stream CNN-LSTM) : Dans le domaine de la reconnaissance d'action dans des vidéos, on peut utiliser une architecture à deux flux où un flux est basé sur des CNN pour l'information spatiale et l'autre sur des LSTMs pour l'information temporelle. Ces deux flux sont ensuite combinés pour la classification finale.

EXEMPLE : EFFICIENT TWO-STREAM NETWORK FOR VIOLENCE DETECTION USING SEPARABLE CONVOLUTIONAL LSTM 2021

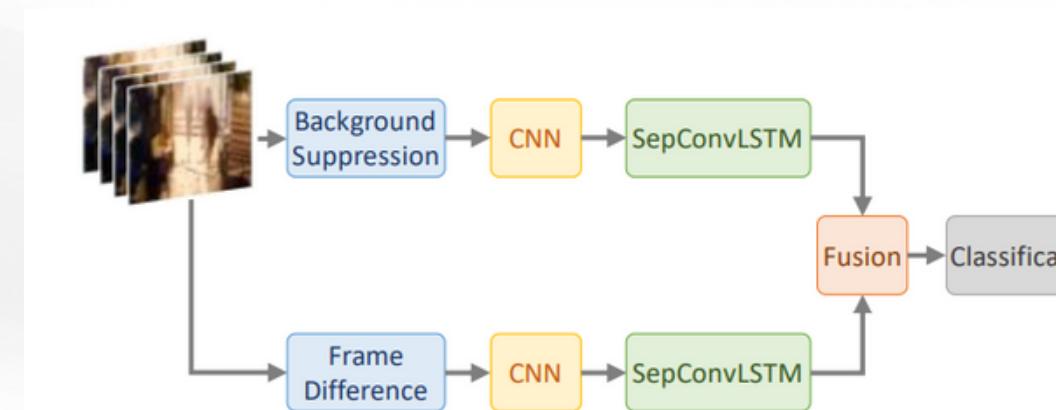
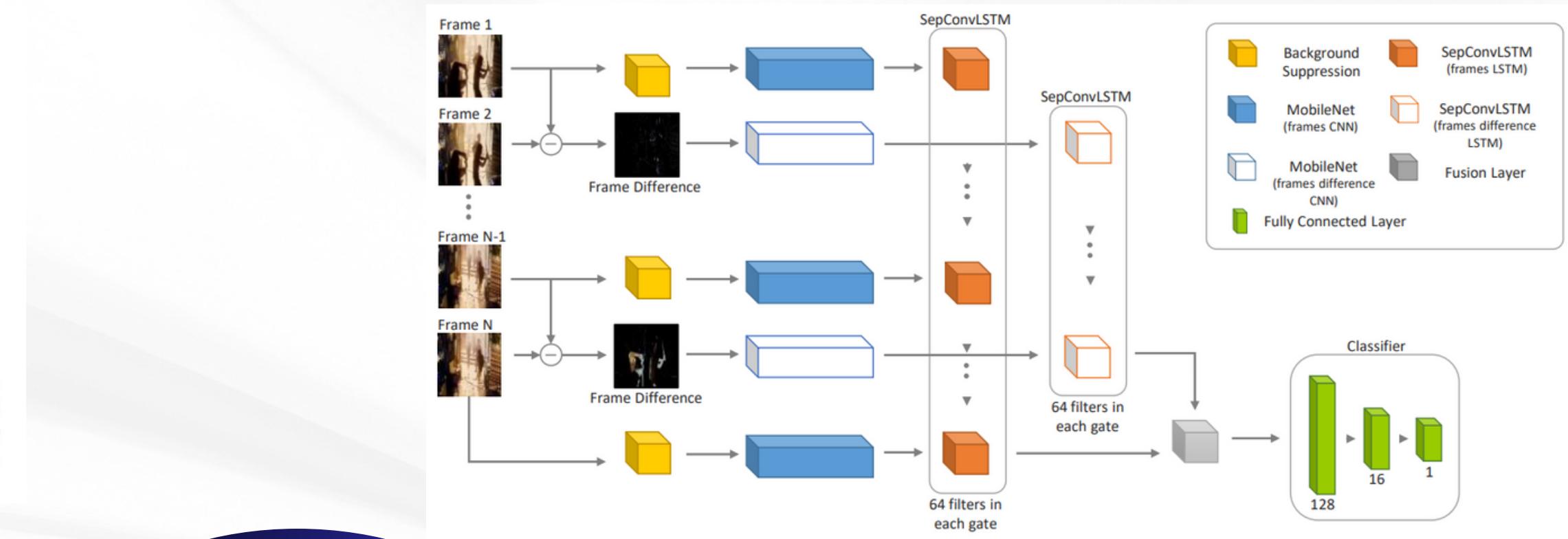


Fig. 1. A schematic overview of the proposed method for violence detection. The proposed pipeline has two streams consisting of CNN and SepConvLSTM modules. Background suppression and Frame difference are pre-processing modules. The output of the two streams are fused to produce robust Spatio-temporal features.

ARCHITECTURE GENERALE



ARCHITECTURE DETAILE

Source: <https://doi.org/10.1109/IJCNN52387.2021.9534280>

LSTMS + MOTIVATION

QUELQUES POINTS D'INTÉRÊT SPÉCIFIQUES À L'UTILISATION DES LSTMS AVEC CNNS

CRNN (Convolutional Recurrent Neural Network) : Le modèle CRNN est souvent utilisé pour des tâches de reconnaissance d'images où l'information temporelle est cruciale (vidéos).

CRNN a été utilisé avec succès dans des tâches telles que la reconnaissance optique de caractères (OCR)

EXEMPLE : OCR USING CRNN: A DEEP LEARNING APPROACH FOR TEXT RECOGNITION

Source: <https://doi.org/10.1109/INCET57972.2023.10170436>

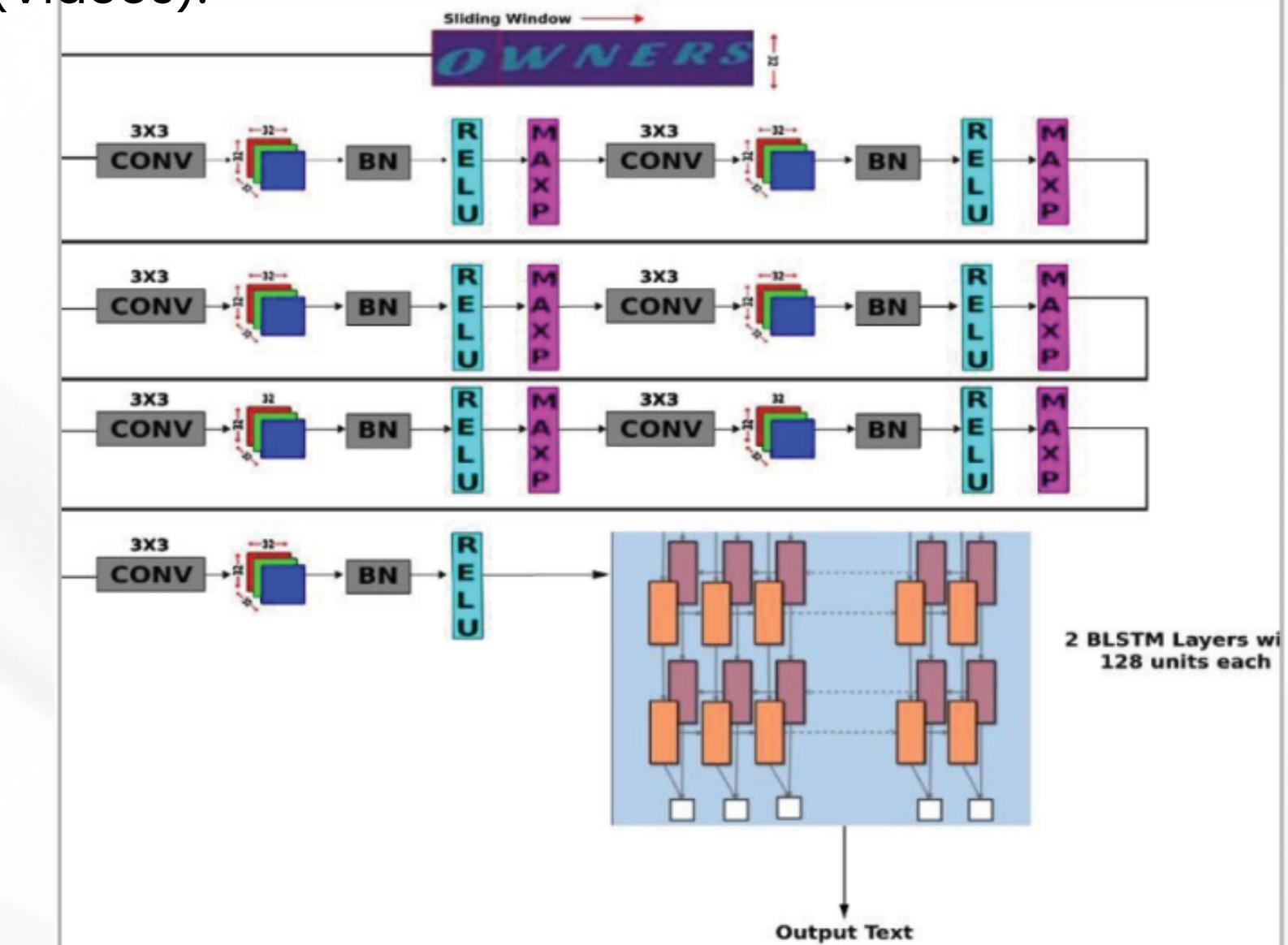


Fig. 1. System Architecture of OCR

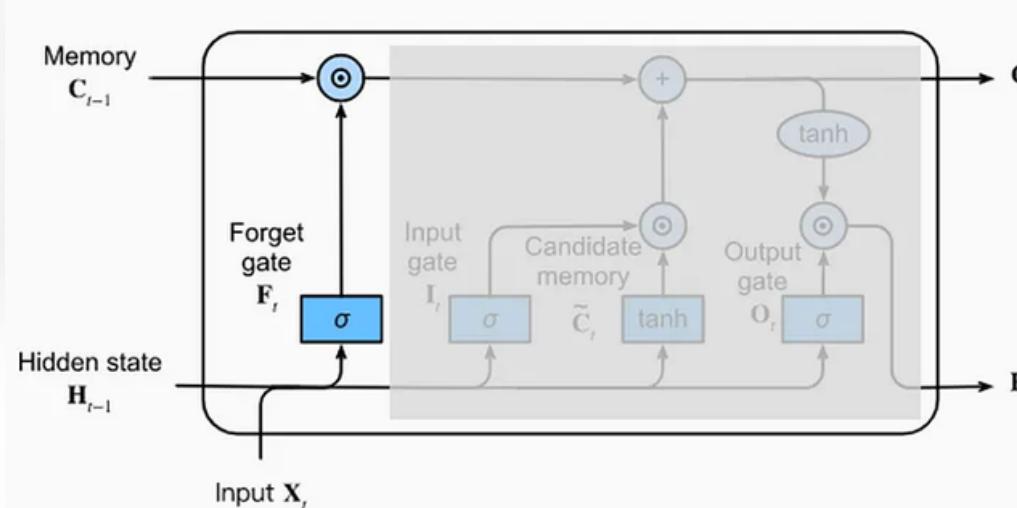
LSTMS

+ ARCHITECTURE

À tout moment t, un LSTM reçoit un vecteur d'entrée ($x_{[t]}$) en tant qu'entrée. Il reçoit également les vecteurs d'état caché ($h_{[t-1]}$) et d'état de cellule ($c_{[t-1]}$) déterminés à l'instant précédent (t-1).

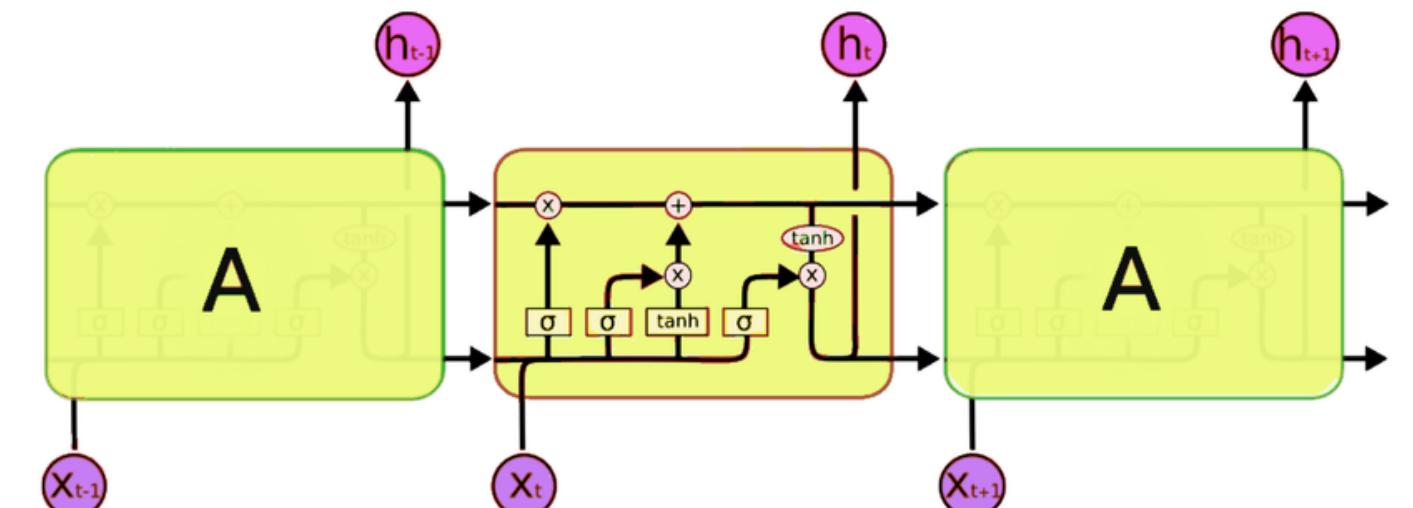
Cellule LSTM présente 3 gates:

Porte d'oubli forget gate : $f_t = \sigma_g (W_f \times x_t + U_f \times h_{t-1} + b_f)$



La fonction d'activation
sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



La porte d'oubli décide (en fonction des vecteurs Input $x_{[t]}$ et Hidden state $h_{[t-1]}$) quelle information retirer du l'état de cellule (cell state $c_{[t-1]}$) provenant du temps t-1.

Wf , Uf bf: Poids de la porte oubli : matrice change durant la Back Propagation Through Time

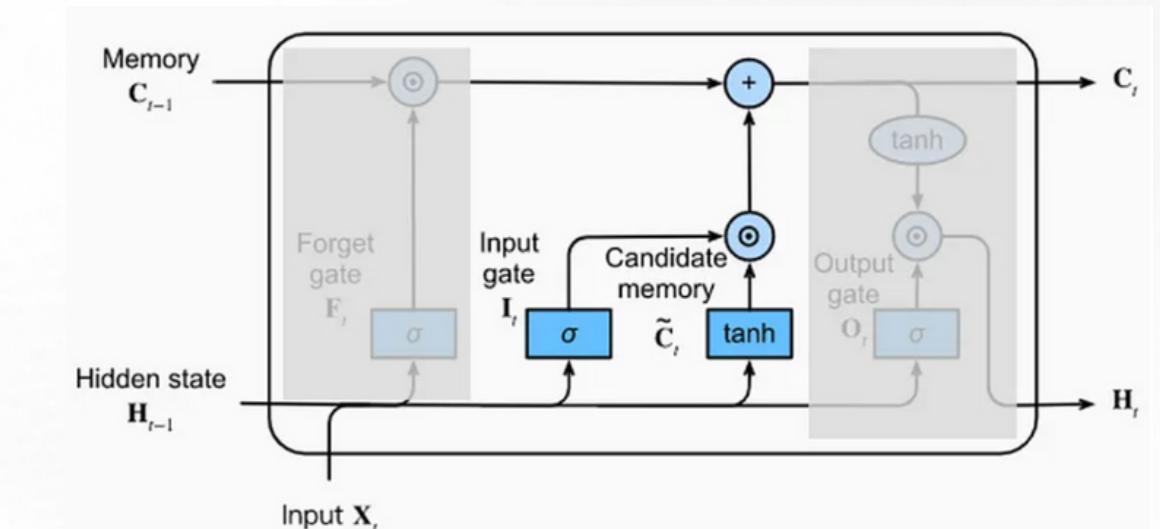
LSTMS

+ ARCHITECTURE

Les LSTM présentent également une porte d'entrée (**Input gate I_t**) et une mémoire candidate (**Candidate memory $C^*[t]$**) qui sont responsables de sélectionner quelle information sera ajoutée à la mémoire de la cellule (**Cell memory $C[t]$**).

Porte d'entrée: Input gate i_t . $i_t = \sigma_g (W_i \times x_t + U_i \times h_{t-1} + b_i)$

La mémoire candidate: $c'_t = \sigma_c (W_c \times x_t + U_c \times h_{t-1} + b_c)$
Candidate Memory : $C^*[t]$.



La fonction tangente hyperbolique (tanh) a été utilisée ici pour normaliser l'information (valeurs de $H_{[t-1]}$ et $x_{[t]}$) entre -1 et 1, qui sera ensuite multipliée par la valeur de $I_{[t]}$. Cette opération aide à déterminer quelle information **sera ajoutée** à la mémoire de la cellule $C_{[t-1]}$.

La fonction
d'activation $\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$

LSTMS + ARCHITECTURE

La cellule LSTM présente porte de sortie qui va choisir quel information va être **réinjecter*** à la cellule dans l'instant $(t+1)$.

$C_{[t]}$ représente l'information sauvegardée dans la mémoire de la cellule pour être réinjectée à la cellule à l'instant $t+1$.

Avec $C_{[t]}$ on peut garder les informations précédentes (instants t_0 à l'instant t)

$H_{[t]}$ représente l'état de la cellule à l'instant actuel.

Porte de sortie: Output gate : $o(t)$

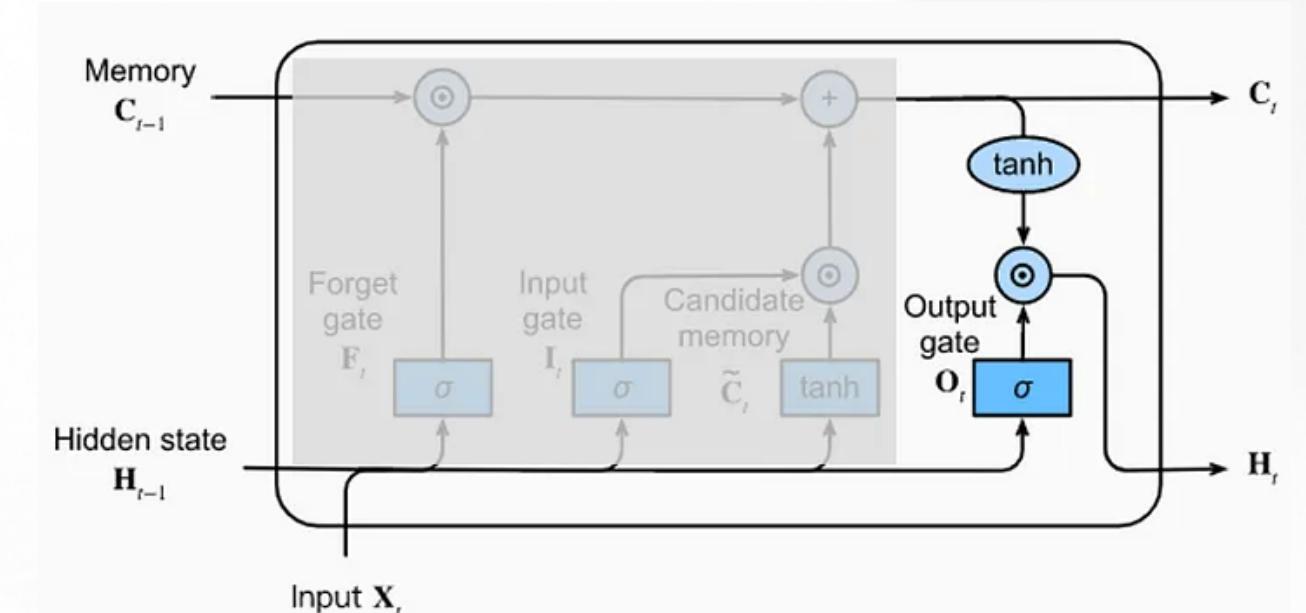
$$o_t = \sigma_g (W_o \times x_t + U_o \times h_{t-1} + b_o)$$

Mémoire de cellule: Cell Memory : $c(t)$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t$$

Etat caché: Hidden state : $h(t)$

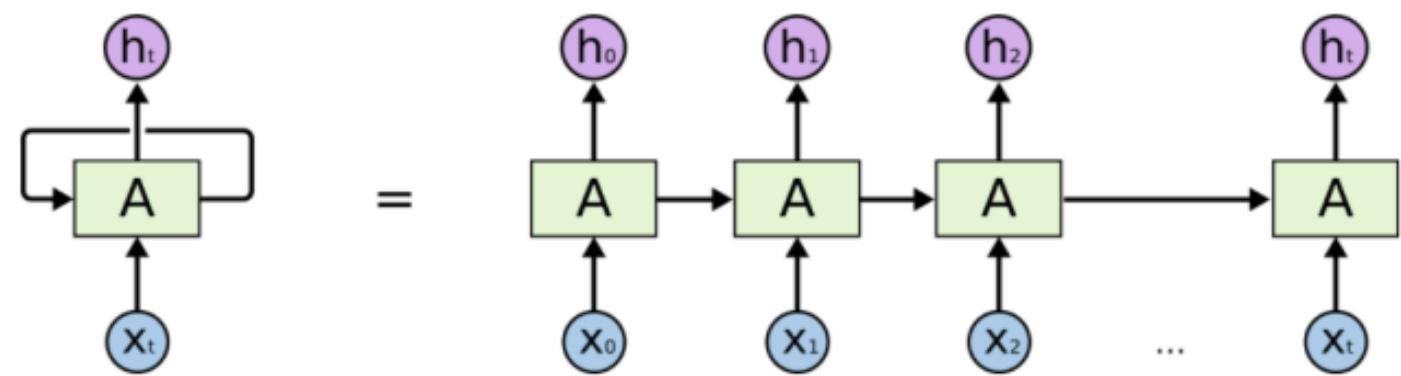
$$h_t = o_t \cdot \sigma_c(c_t)$$



LSTMS

+ REINJECTER? UNE SEULE CELLULE LSTM

- A chaque instant t une nouvelle information $x_{[t]}$ va être injecter à la cellule.
- On calcule $h_{[t]}$ et $c_{[t]}$
- On injecte à l'instant $t+1$ nouvelle information $x_{[t+1]}$ avec $h_{[t]}$ et $c_{[t]}$ à la même cellule.



LSTMS

LES TYPES DE RÉSEAUX RÉCURRENTS:

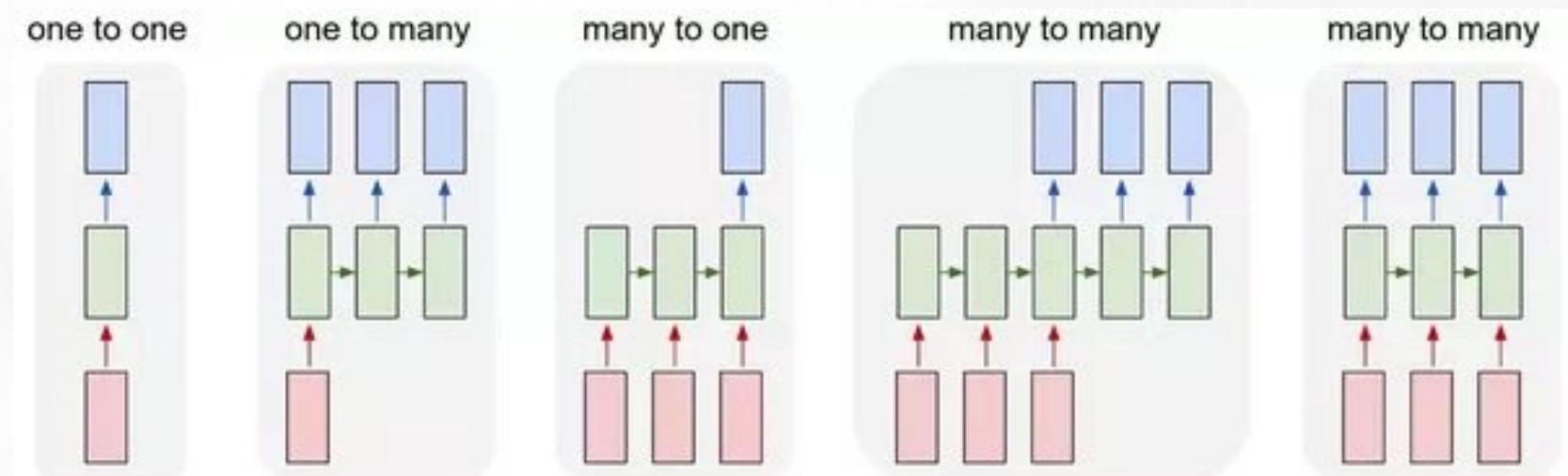
One to One : cas image classification (image -> classe)

One to many : Image Captioning (image -> texte)

Many to one : L'analyse des sentiments (texte -> classe)

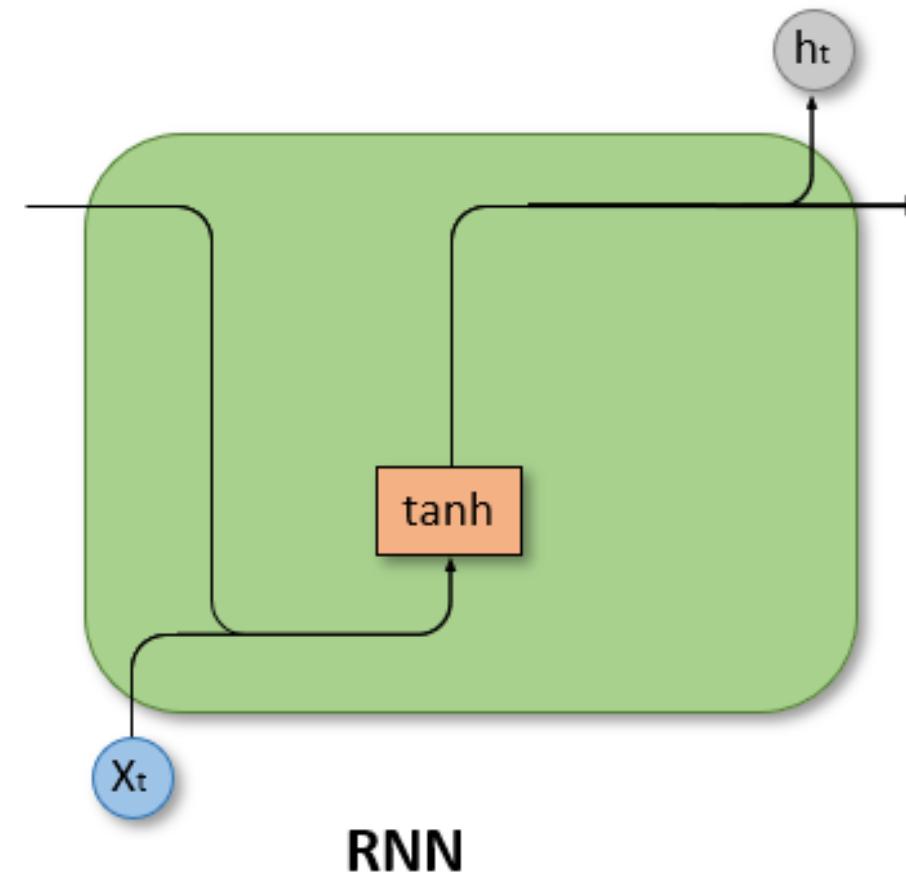
Many to many (1) : traduction (texte -> texte)

Many to many (2) : image captioning(vidéo -> texte)

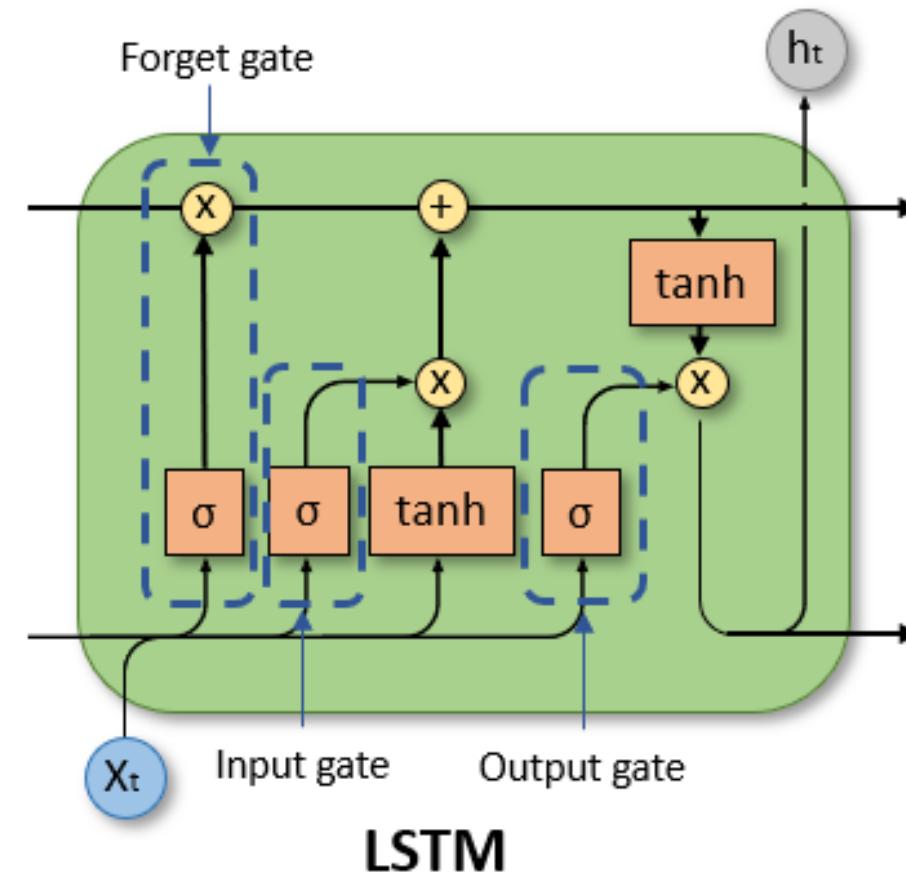


LSTMS

RNN | LSTM | GRU:

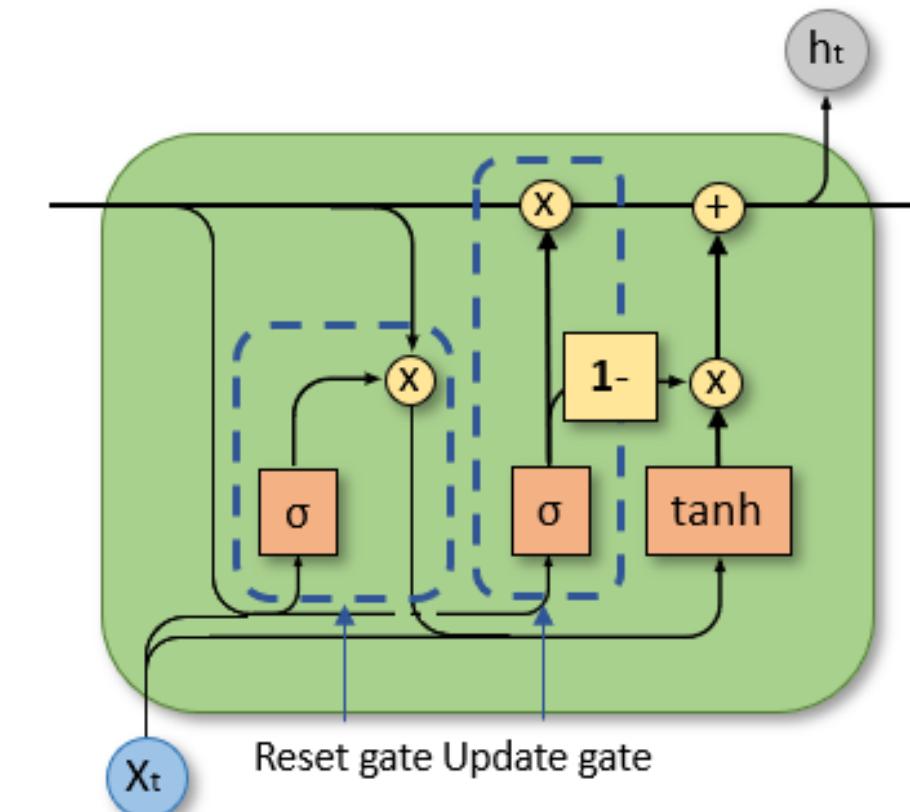


ni de protes, ni mémoires



LSTM

Mémoire et 3 portes:
- Forget gate
- Input gate
- Ouput gate



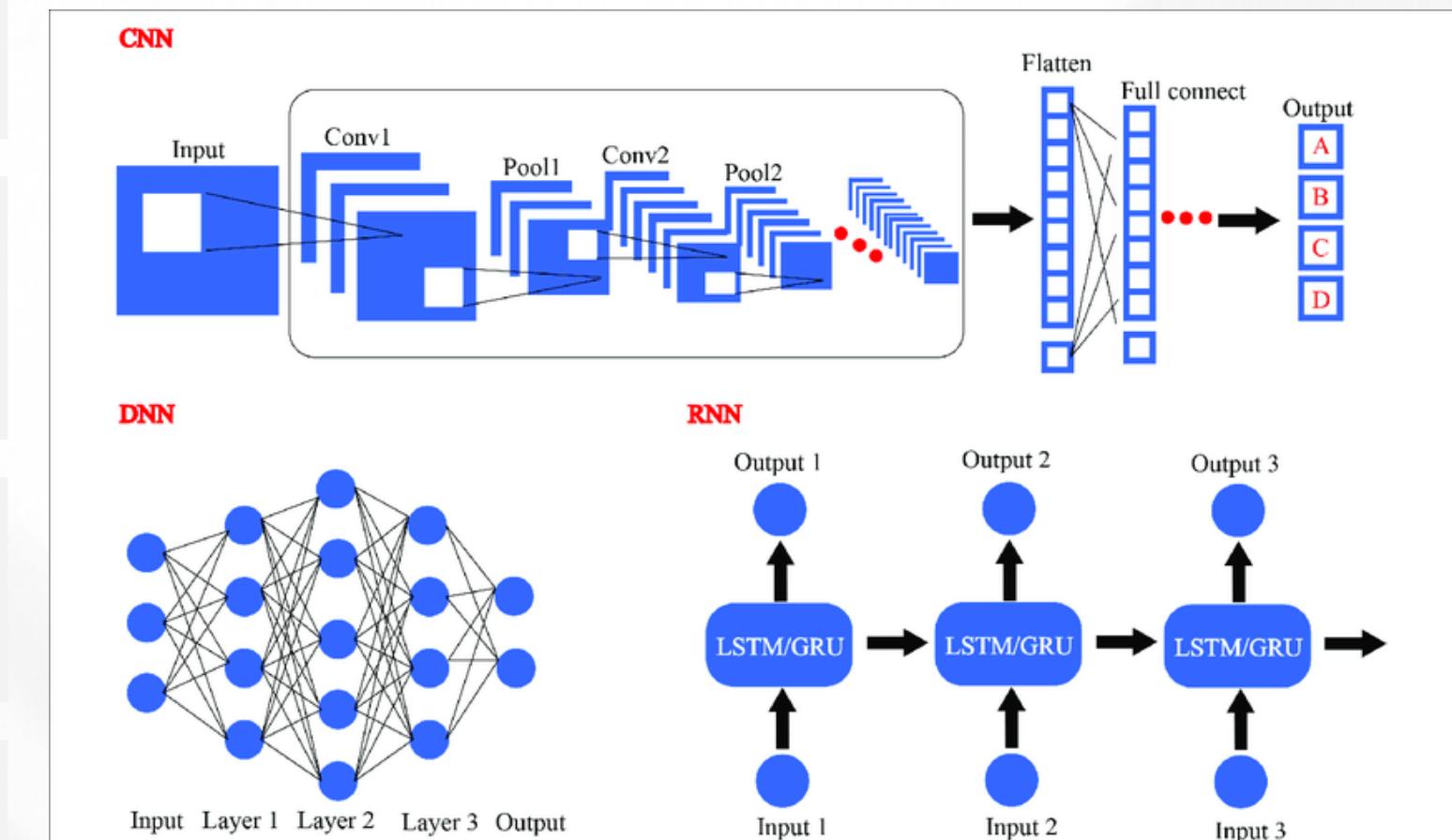
GRU

Mémoire et 2 portes:
- Reset gate
- Update gate

LSTMS

DNN | CNN | LSTM:

	DNN	CNN	LSTM/RNN/GRU
Information spatiale	Non	Oui	Oui
Information temporelle	Non	Non	Oui
Back Propagation through time	Non	Non	Oui



LSTMS

NOTES ET ANNEXES:

Le Blog suivant présente: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

- Recurrent Neural Networks de manière générale
- Le Problème des Dépendances à Long Terme
- LSTM Networks détaillé : (le rôle de chaque porte(gates))
- Différentes architectures/types de LSTMs.

Pour future recherches, on peut:

- Comparer les résultats de quelques variantes de LSTMs:
commençant par les papiers suivants:

<https://arxiv.org/pdf/1503.04069.pdf>

<https://arxiv.org/pdf/1402.3511v1.pdf>

QUESTIONS ?