Aprendizagem Profunda
Memória de Curto Prazo Longa

Aluizio Fausto Ribeiro Araújo
Universidade Federal de Pernambuco
Centro de Informática

UFPE

Centro de Informática UFPE

Conteúdo

- Introdução
- Redes neurais recorrentes
- Memória de Curto Prazo Longa (LSTM)
  - LSTM Vanilla
- Treinamento de uma LSTM
- Exemplos de LSTM
- Aplicações

Introdução

- Em uma rede de múltiplas camadas, os gradientes para camadas mais próximas da entrada são proporcionais a produtos de muitos gradientes das funções de ativação:
  - Se há gradientes muito pequenos ou nulos, haverá decaimento acentuado do gradiente resultante;
  - Se os termos de gradientes forem maiores que 1, pode haver crescimento acentuado do gradiente resultante;
- Logo, é muito difícil treinar redes profundas ou redes recorrentes simples (SRN) por retropropagação original;
- Em particular, é muito difícil treinar SRNs que lidem com dependências de longa distância;

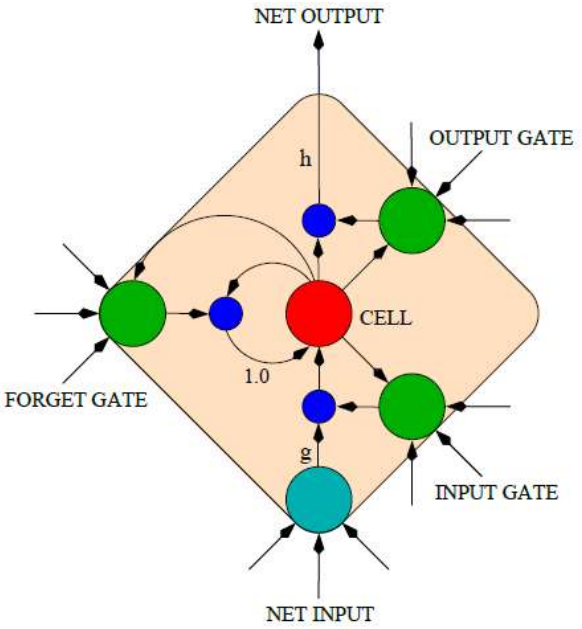
Introdução

- Redes Memória de Curto Prazo Longa (LSTM) são um tipo de rede neural recorrente (RNN) que aprendem dependências de longo prazo:
  - Uma LSTM pode aprender a transpor intervalos de tempo superiores a 1.000 passos de tempo;
- LSTM evita o desaparecimento e explosão do gradiente e permite reter as informações de estado (STM) por tempo mais longos:
  - A existência da estrutura de célula, uma instância de rede recorrente, que mantém o sinal de erro dentro da célula de cada unidade;
  - A ativação realimentada é multiplicada pela saída do *gate* de esquecimento, que decide quando e quanto esquecer;
  - A ativação da célula é realimentada para ela mesma, mantendo um valor ativo por longo tempo;

Introdução

- LSTM usa células de memória cujas ativações são atualizadas a cada passo de uma sequência temporal;

- Três portais (*gates*) controlam o fluxo de sinais que entram e saem de uma célula de memória:
  - *Gate* de entrada: Protege o passo atual de entradas irrelevantes;
  - *Gate* de saída: Impede que sinais atuais irrelevantes sejam passados para passos posteriores;
  - *Gate* de esquecimento: Limita os sinais passadas de uma célula para a próxima.

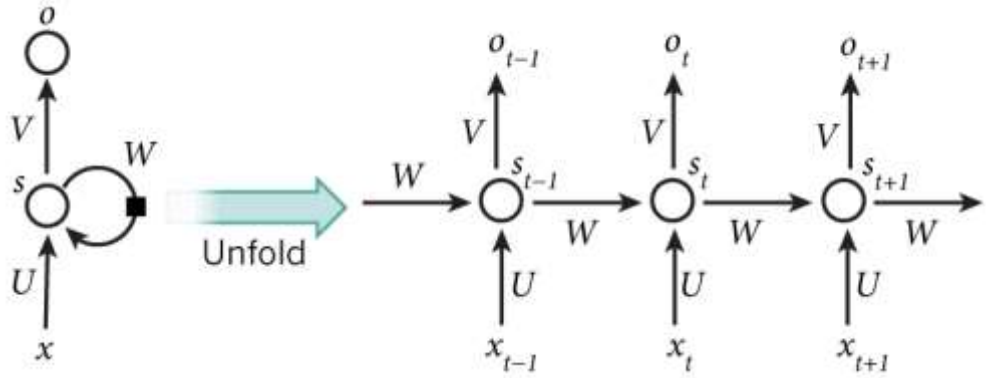


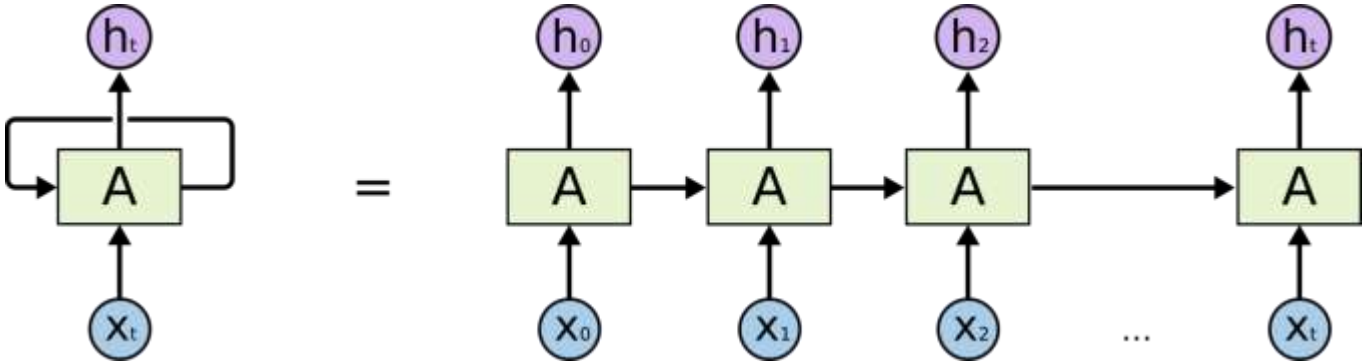
Introdução

- LSTM é amplamente usada para processamento de sequências incluindo tarefas de processamento de linguagem natural tais como:
  - Reconhecimento de fala no *Google*;
  - Traduções automáticas no *Google* tradutor;
  - Respostas de *Alexa* da *Amazon*;
  - Traduções de conteúdo pelo *Facebook*;
  - Em jogos como *Alphastar* da *Google Deepmind*;

Redes Neurais Recorrentes

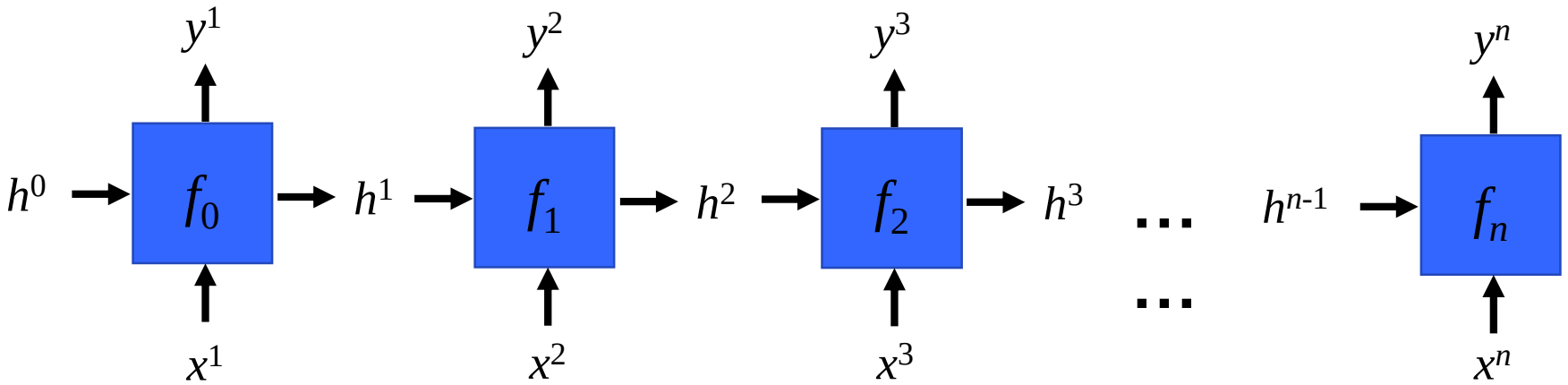
- Nodo com recorrência:
  - Realimentação local de estado (s) para entrada (x) gerando saída (o);
  - A rede é desdobrada para treinamento;





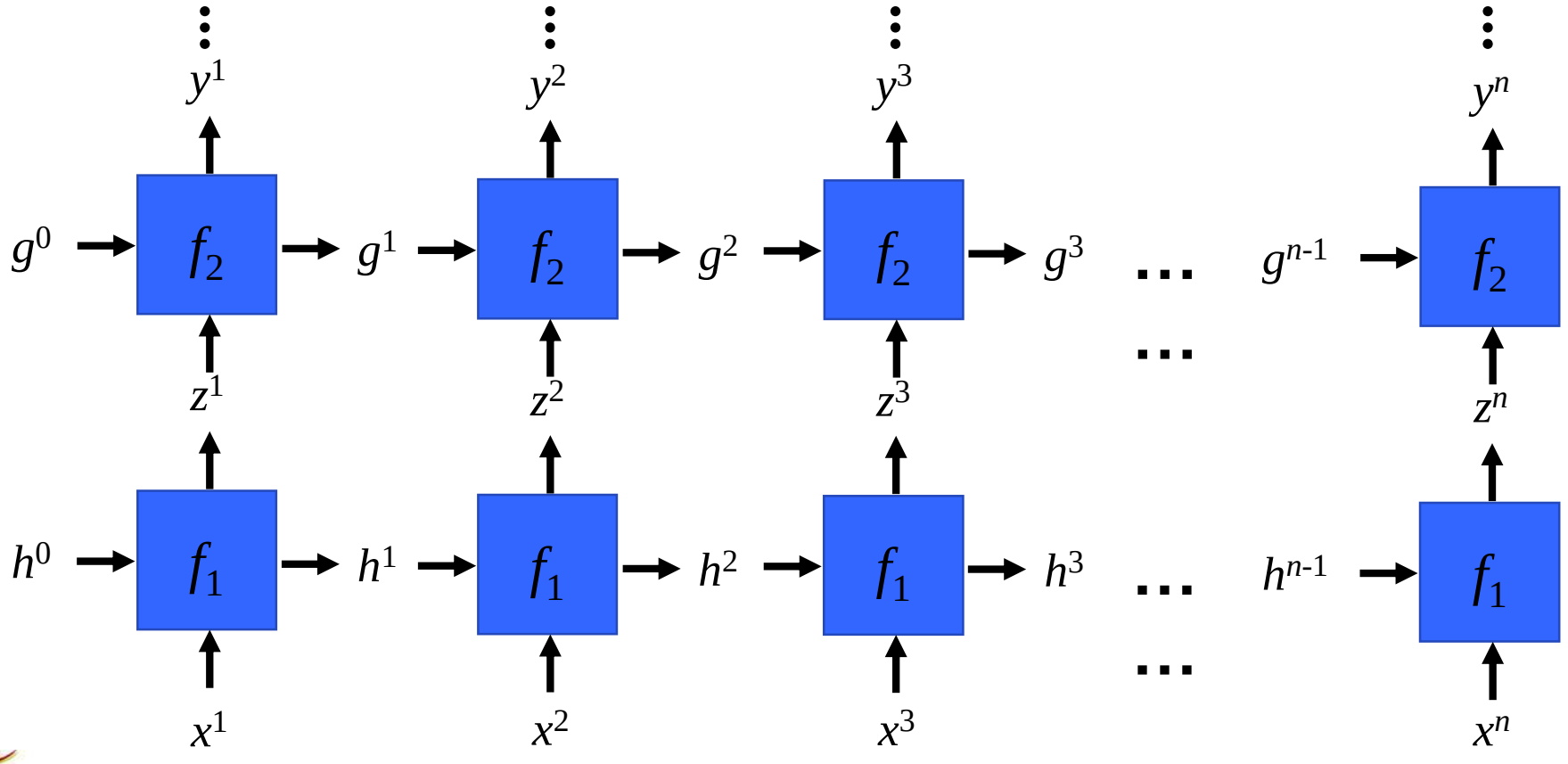
Redes Neurais Recorrentes

- Redução da complexidade de uma RNN:
  - Faz-se $f_0 = f_1 = f_2 = f_3 = \ldots = f_n$;
- As saídas são definidas como $y^t = f_i\,(h^{t-1}, x^t)$;



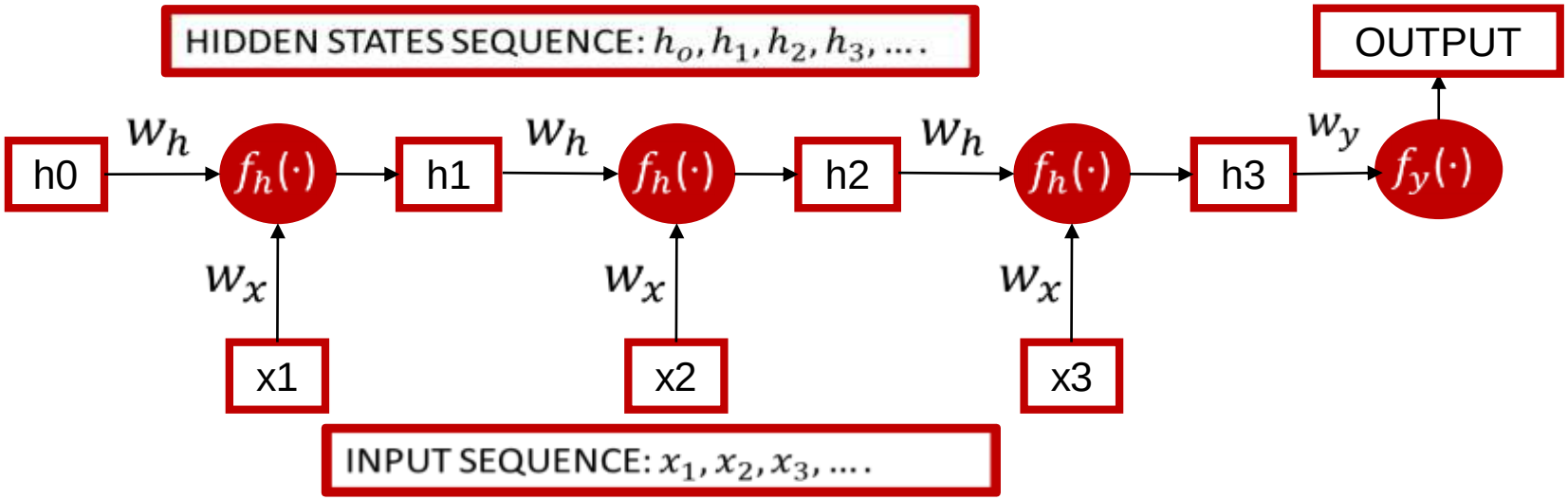
Redes Neurais Recorrentes

- Tornando a RNN profunda: $y^t = f_2(g^{t-1}, z^t) = f_2(g^{t-1}, f_1(h^{t-1}, x^t))$;



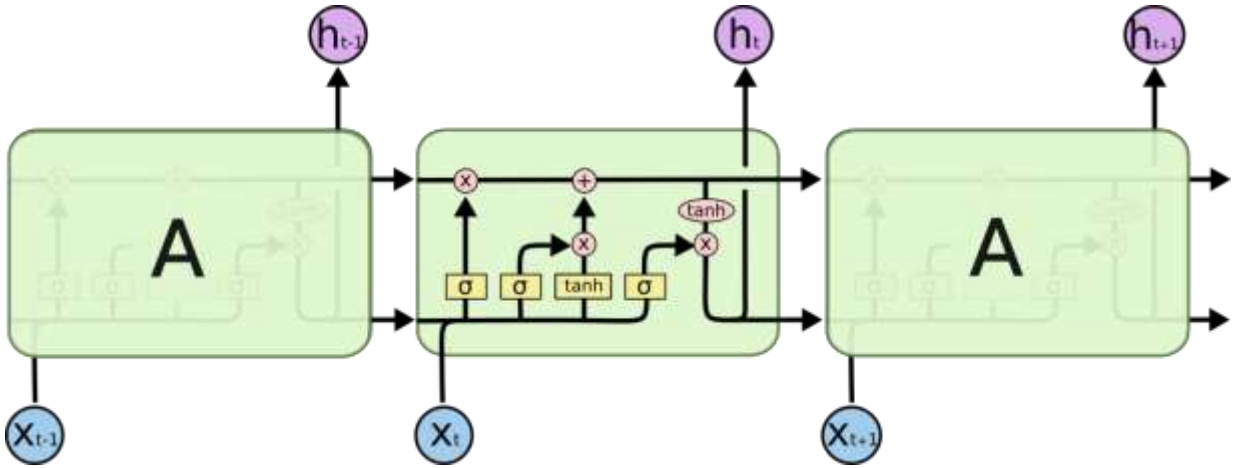
Redes Neurais Recorrentes

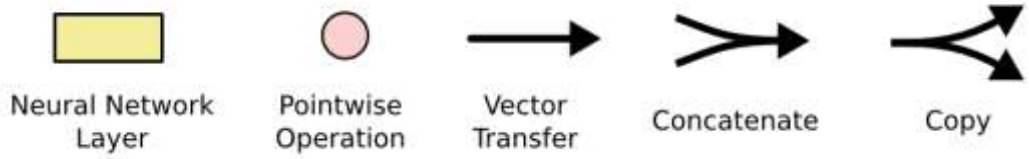
- Arquitetura típica:
  - Emprega os mesmos pesos W_h e W_x para todos os passos de tempo;
  - Memória RNN: sequência de estados escondidos aprendidos (h_0, h_1, h_2 ...);
  - $h(t) = f_h\,(W_h * h(t\text{-}1) + W_x * x(t))$
  - A função de ativação, f_h (.) é não linear, e.g., ReLU ou tanh;



Memória de Curto Prazo Longa

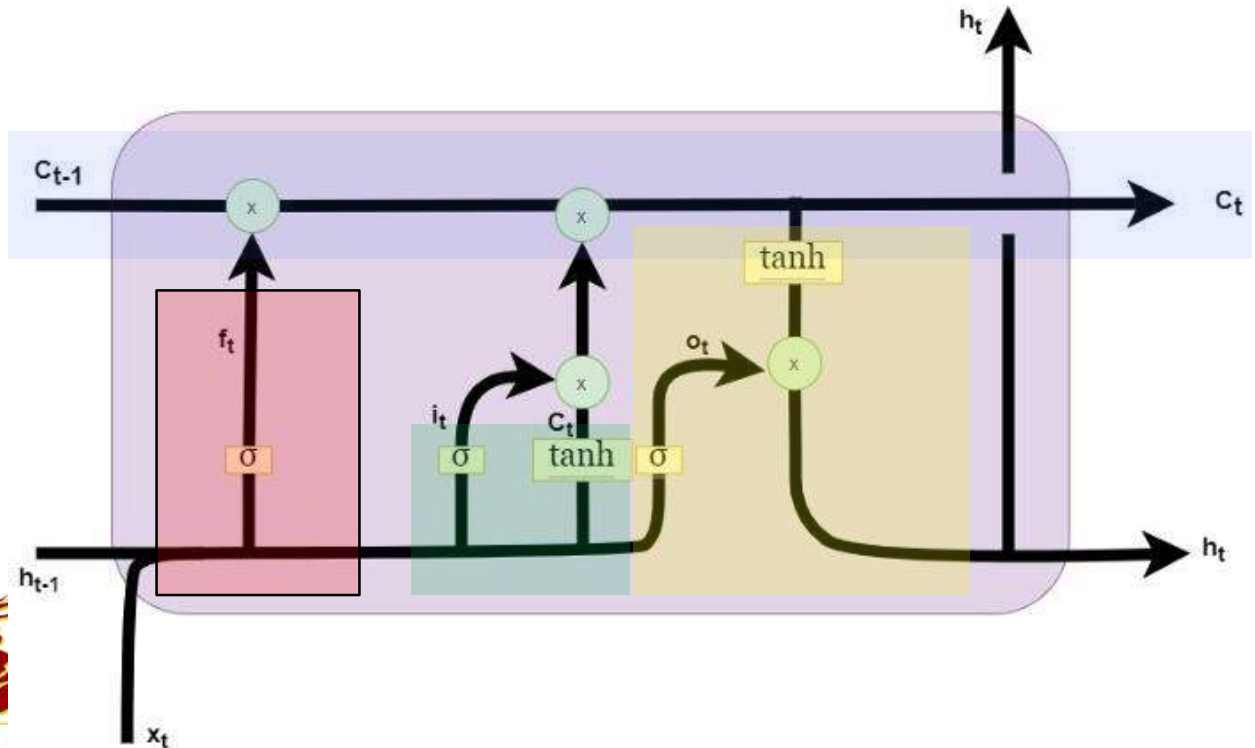
- Arquitetura:
  - Definida por conjunto de sub-redes conectadas recorrentemente;
  - Estes blocos de memória mantêm seu estado ao longo do tempo e regulam o fluxo de informações por meio de unidades de controle não-lineares;
  - Existência de uma ou mais células de memória e unidades multiplicativas;





Memória de Curto Prazo Longa

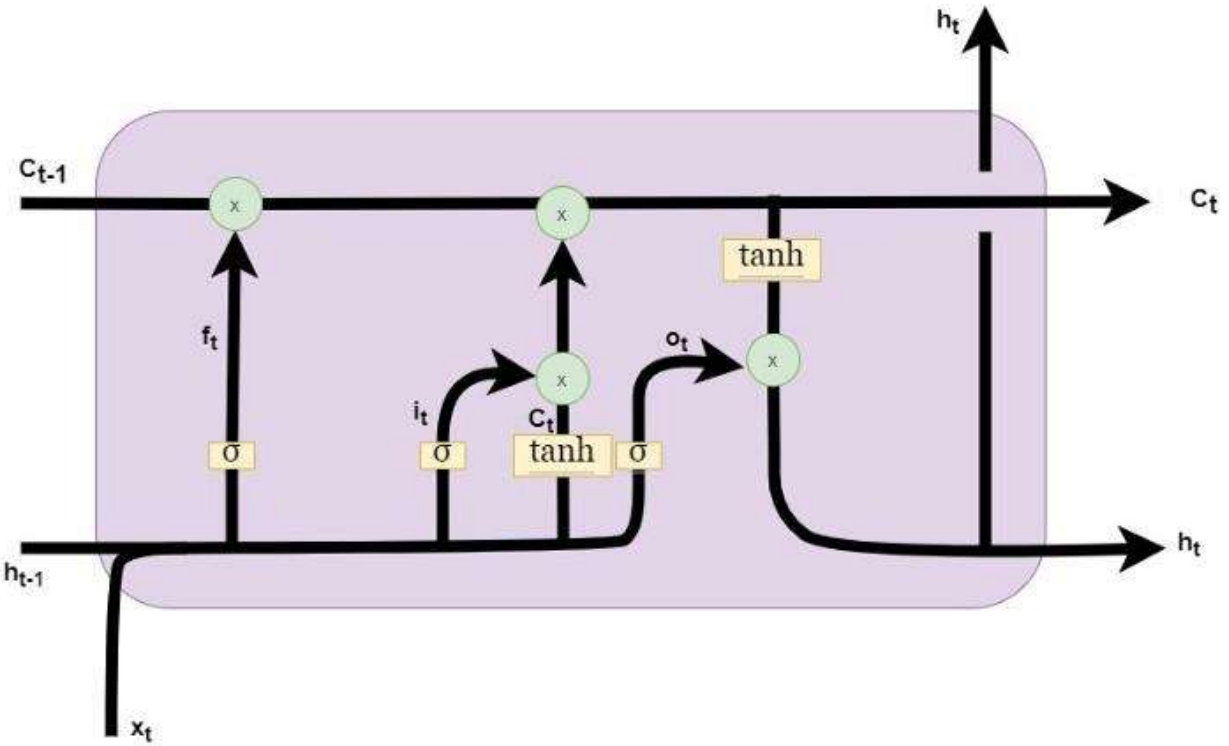
- Unidade LSTM padrão possui uma célula, um *gate* de entrada, um *gate* de saída e um *gate* de esquecimento,
  - O gate de esquecimento foi proposto por Gers et al. (2000) para permitir que a rede redefina seu estado;



A célula armazena valores em intervalos de tempo arbitrários e os 3 *gates* regulam o fluxo de informações associadas à célula;

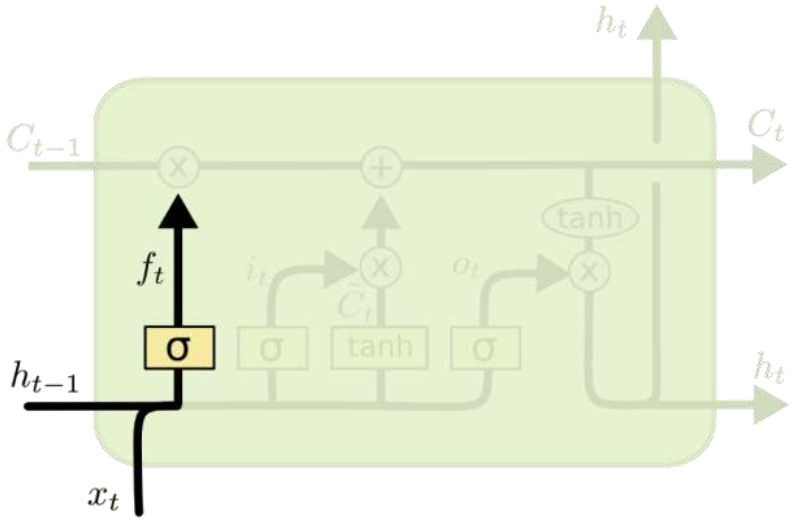
Memória de Curto Prazo Longa

- Célula de memória:
  - Sinais recebidos: Ativação (c_{t-1}) e estado oculto anterior (h_{t-1}) da célula, e entrada atual (x_t);
  - Gates: esquecimento, entrada e saída;
  - Sinais produzidos: Ativação e estado oculto atuais da célula e saída atual;



Memória de Curto Prazo Longa

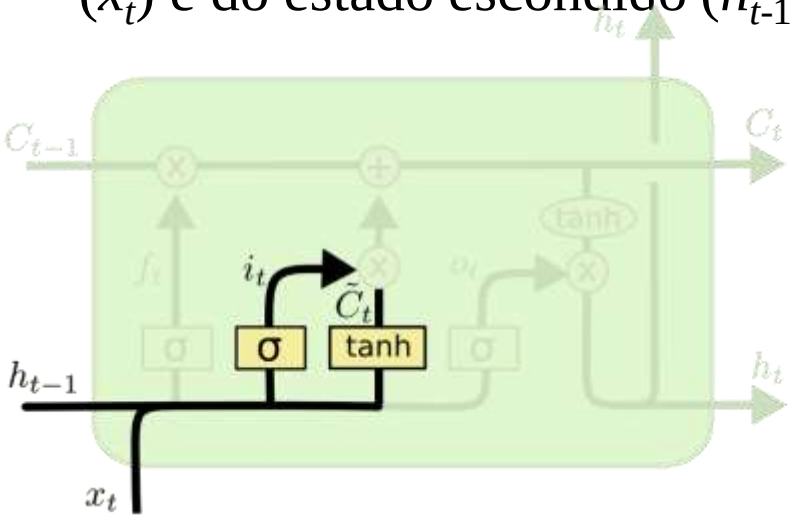
- *Gate* de esquecimento:
  - Calcula o valor $0 \leq f_t \leq 1$ por uma função sigmoide logística para dois argumentos: uma entrada x_t e o estado escondido anterior h_{t-1}:
  - Multiplica a saída de f_t pelo estado anterior da célula (C_{t-1}), este decresce (é esquecido) quando o *gate* produz saída próxima a 0;



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

Memória de Curto Prazo Longa

- *Gate* de entrada:
  - Determina quais entradas no estado da célula devem ser atualizadas calculando a saída da função sigmoidal (i_t), valor entre 0 e 1;
  - Determina ativação temporária da célula ($\tilde{C}_t$), sem considerar os *gates* de esquecimento e de entrada, calculado por uma função tanh da entrada (x_t) e do estado escondido (h_{t-1});

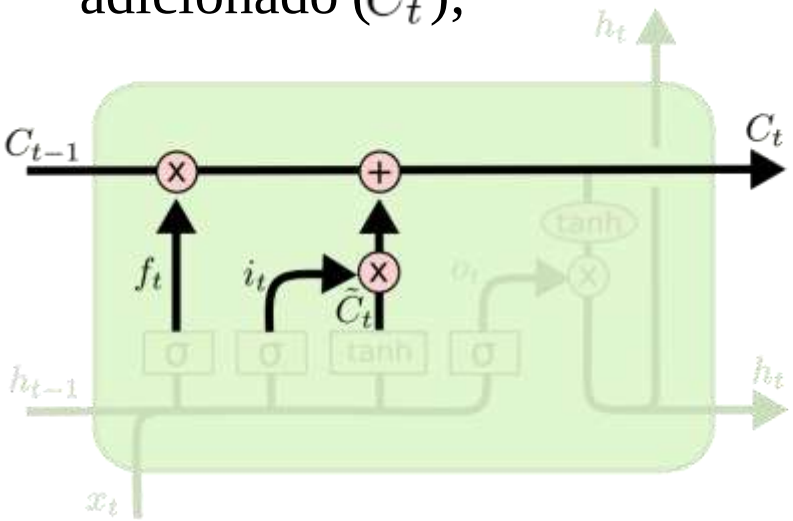


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

Memória de Curto Prazo Longa

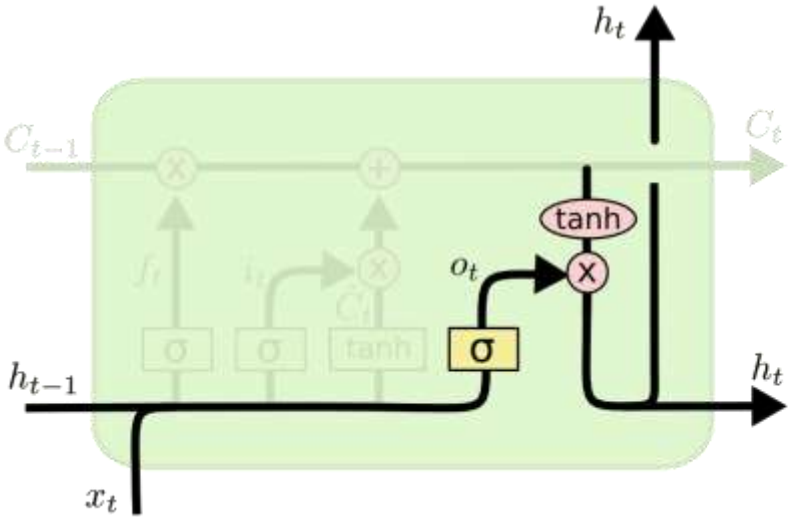
- Atualização do estado da célula:
  - Vetor C_t com mesma dimensionalidade que o estado escondido, h_t;
  - O estado da célula é atualizado pela soma de duas parcelas: produto do componente de esquecimento (f_t) com o estado anterior da célula (C_{t-1}) somado ao produto das entradas a serem atualizadas (i_t) e o valor a ser adicionado ($\tilde{C}_t$);



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

Memória de Curto Prazo Longa

- *Gate* de saída:
  - O *gate* de saída produz uma saída como função (σ) da entrada atual e do estado escondido anterior;
  - O estado escondido (saída da célula) é atualizado com base em uma versão "filtrada" do estado da célula, empregando a função tanh;

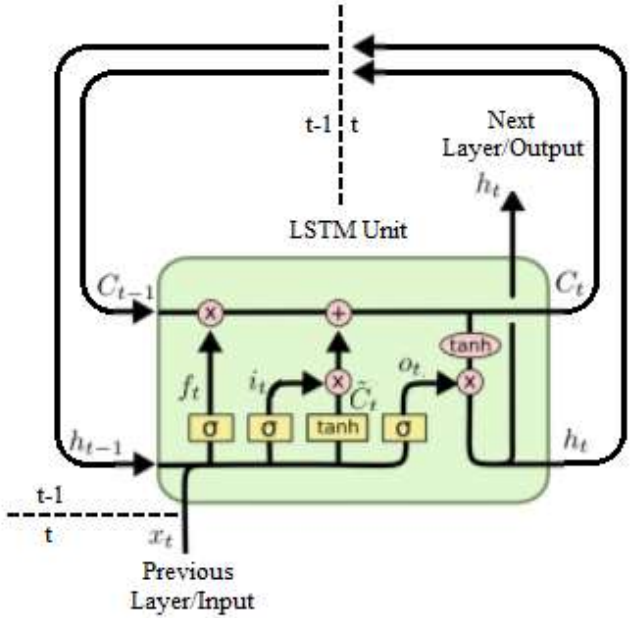


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

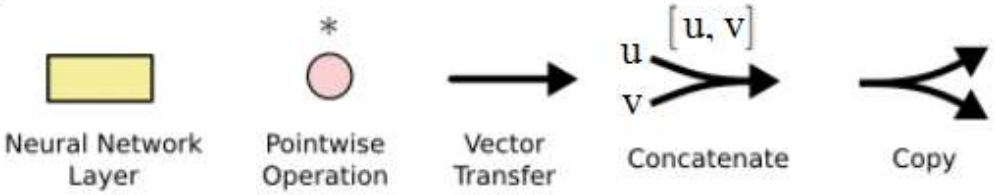
Memória de Curto Prazo Longa

- Esboço do modelo completo:



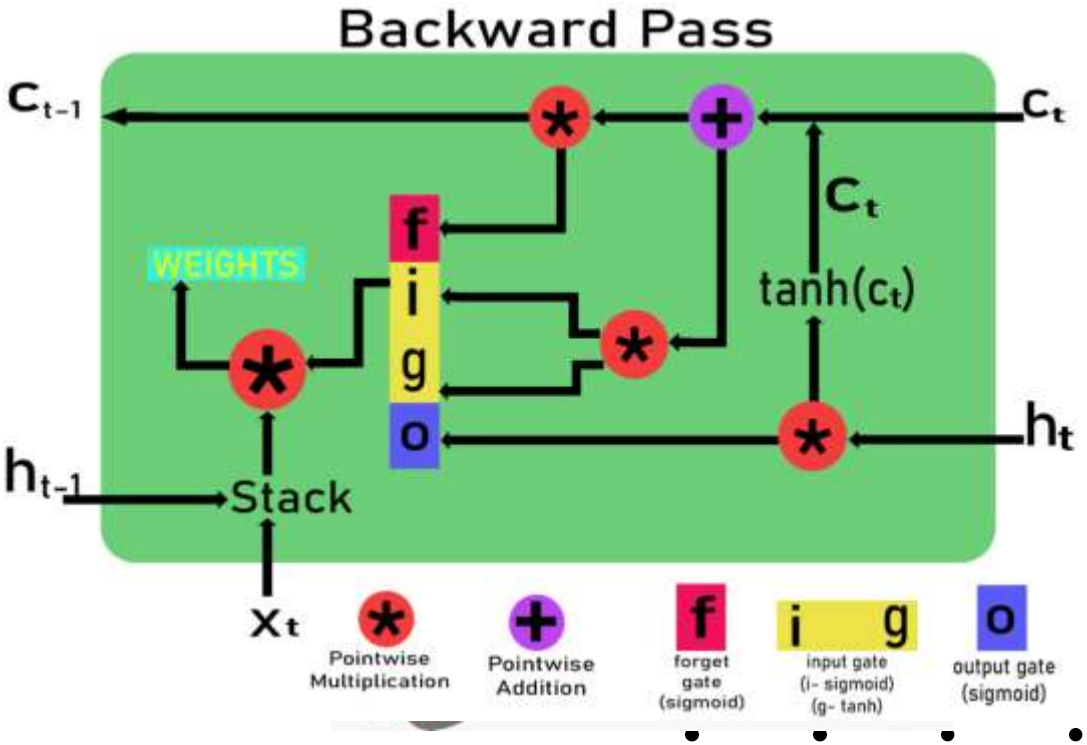
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Gate de esquecimento;
Gate de entrada
Estado temporário da célula
Novo estado da célula
Gate de saída
Saída da LSTM



Treinamento de uma LSTM

- Cada célula tem muitos parâmetros (W_f, W_i, W_C, W_o):
  - Geralmente requer muitos dados de treinamento;
  - Requer muito tempo de computação e usa clusters de GPU;
  - https://www.geeksforgeeks.org/lstm-derivation-of-back-propagation-through-time/



Treinamento de uma LSTM

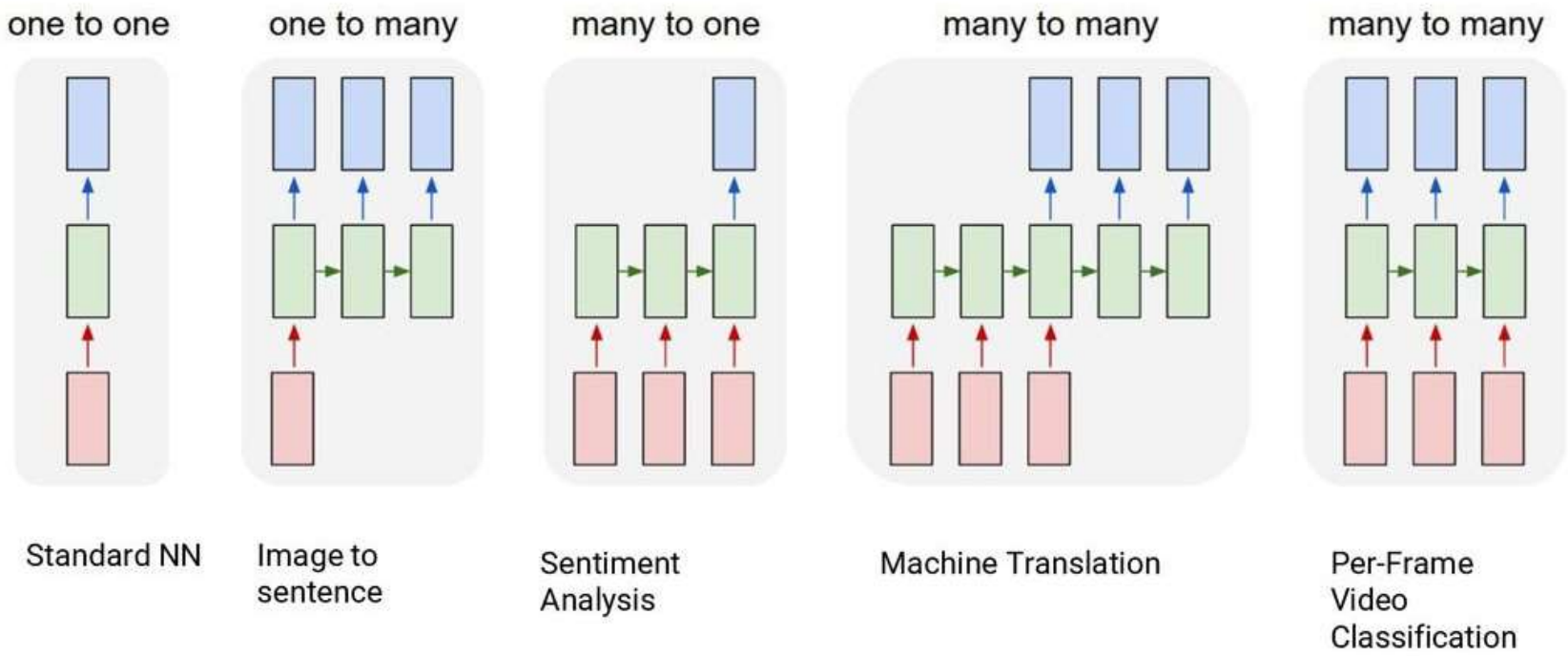
- Treinamento com modelos empregando retropropagação:
  - Gradiente descendente estocástico (SGD) com ordem aleatória de instâncias de treinamento em cada época e momento, tende a convergir para mínimos locais se a função objetivo não for convexa;
  - Gradiente acelerado de Nesterov permite adaptar as atualizações com o gradiente à inclinação (termo do momento) e acelerar o SGD;
  - Adagrad permite adaptar as atualizações para cada parâmetro ajustável: ajustes maiores (menores) para parâmetros menos (mais) frequentes,
    - Aumenta a velocidade, escalabilidade e robustez do SGD e pode ser usado para treinar redes neurais de grande escala;
    - Torna a taxa de aprendizagem muito pequena, impedindo o ajuste significativo de pesos;

Treinamento de LSTM

- Treinamento com modelos empregando retropropagação:
  - Otimizador AdaDelta visa eliminar o problema da taxa de aprendizgem diminuta, calculando o gradiente passado acumulado em uma janela de tamanho fixo;
  - RMSProp também visa eliminar o problema da taxa de aprendizgem diminuta,
    - Necessita da determinação de dois parâmetros: gama e a taxa de aprendizagem, valores sugeridos: $\gamma = 0{,}9$ e $\eta = 0{,}001$;
  - Adam tem o mesmo objetivo que os dois acima,
    - Necessita de hiperparâmetros, cujos valores sugeridos são: $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$ e $\varepsilon = 10^{-8}$;

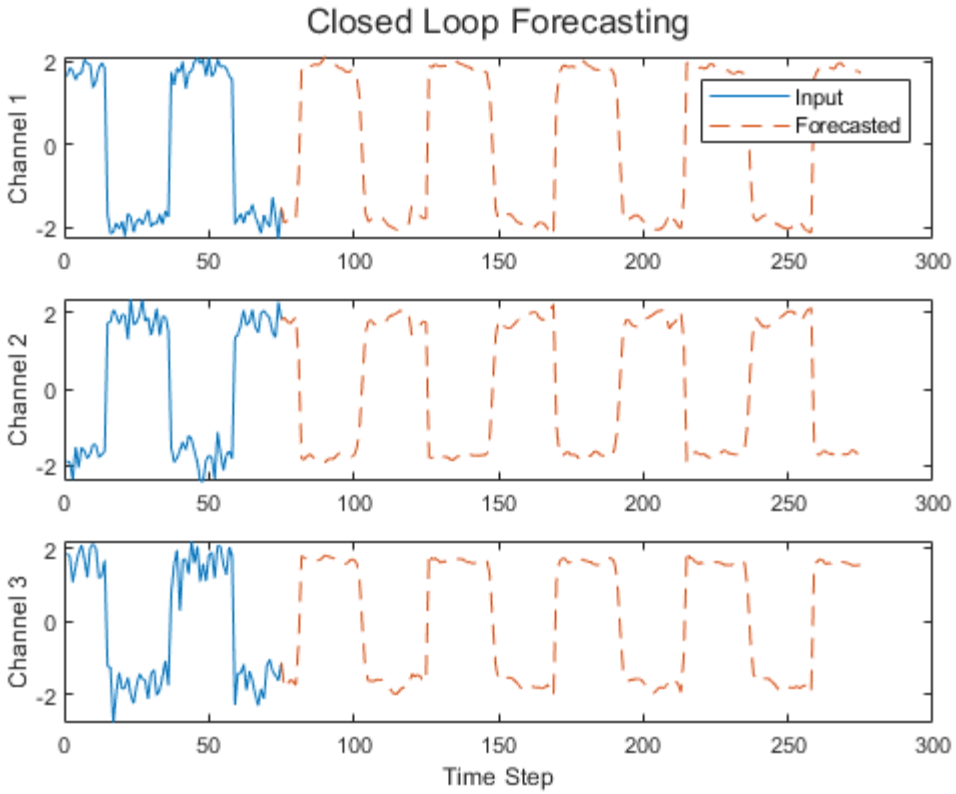
Exemplo de LSTM

- Arquiteturas com LSTM para diferentes finalidades



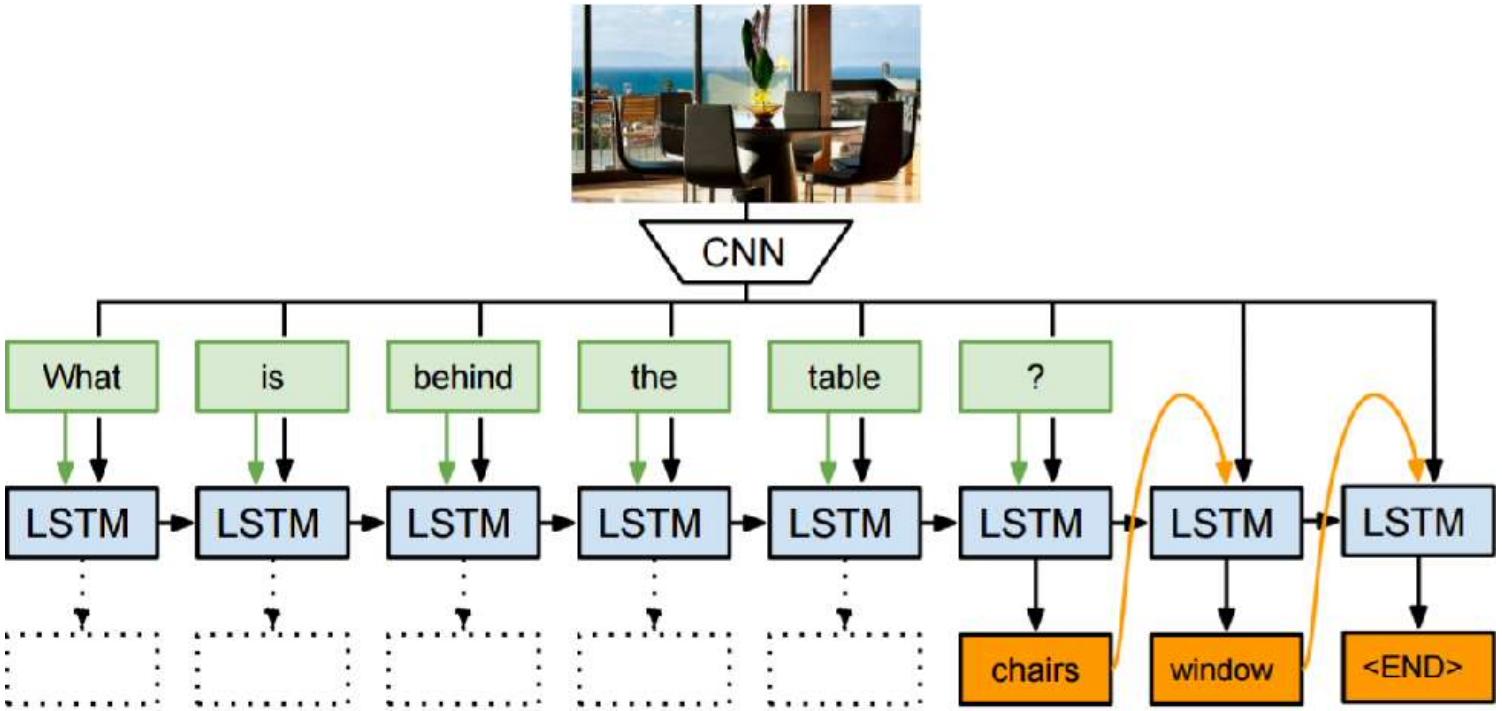
Aplicações de LSTM

- Predição de séries temporais:
  - Empregar uma LSTM para prever valores futuros de uma série temporal a partir de uma sequência de dados em instantes de tempos anteriores;



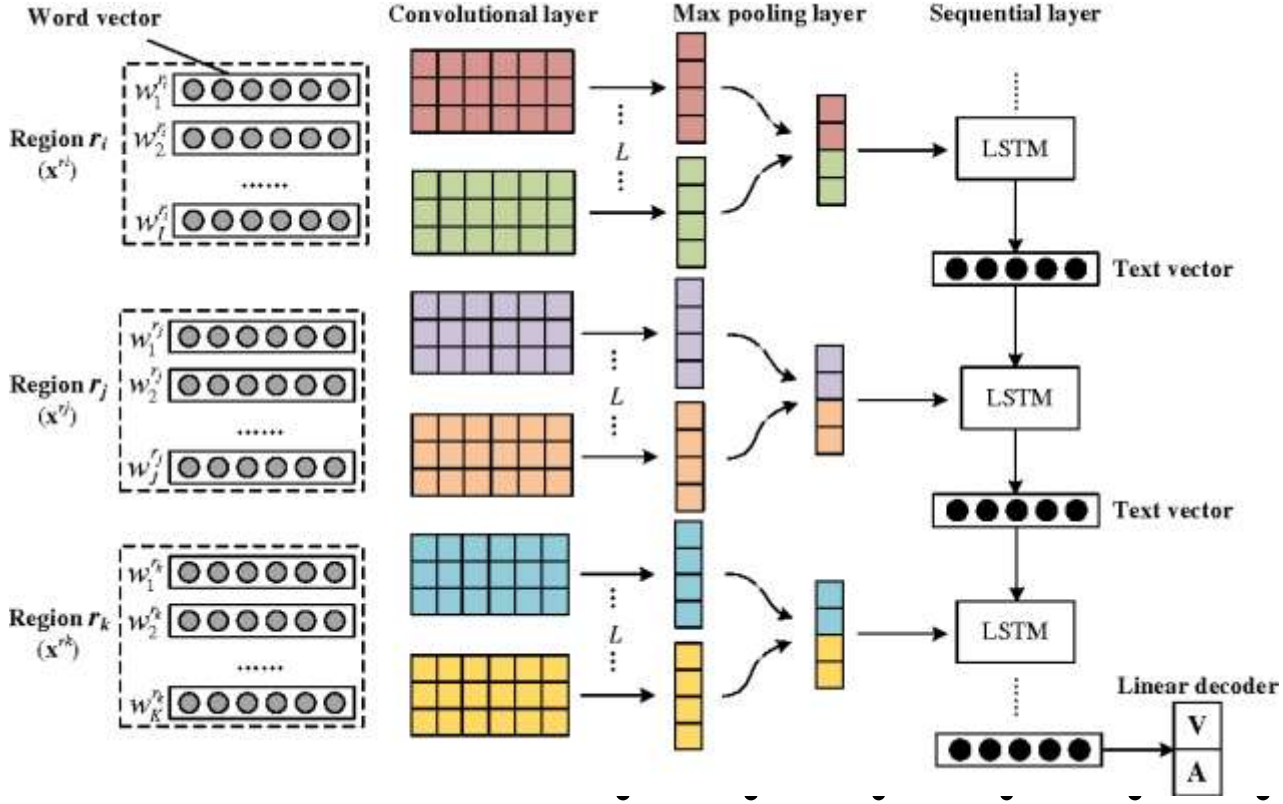
Aplicações de LSTM

- Respondendo perguntas sobre imagens:
  - Demanda geração de linguagem com base em pistas textuais e visuais;
  - Malinowski et al. previram respostas com palavras vinculadas a uma imagem de entrada modelada por CNN e texto modelado por LSTM;



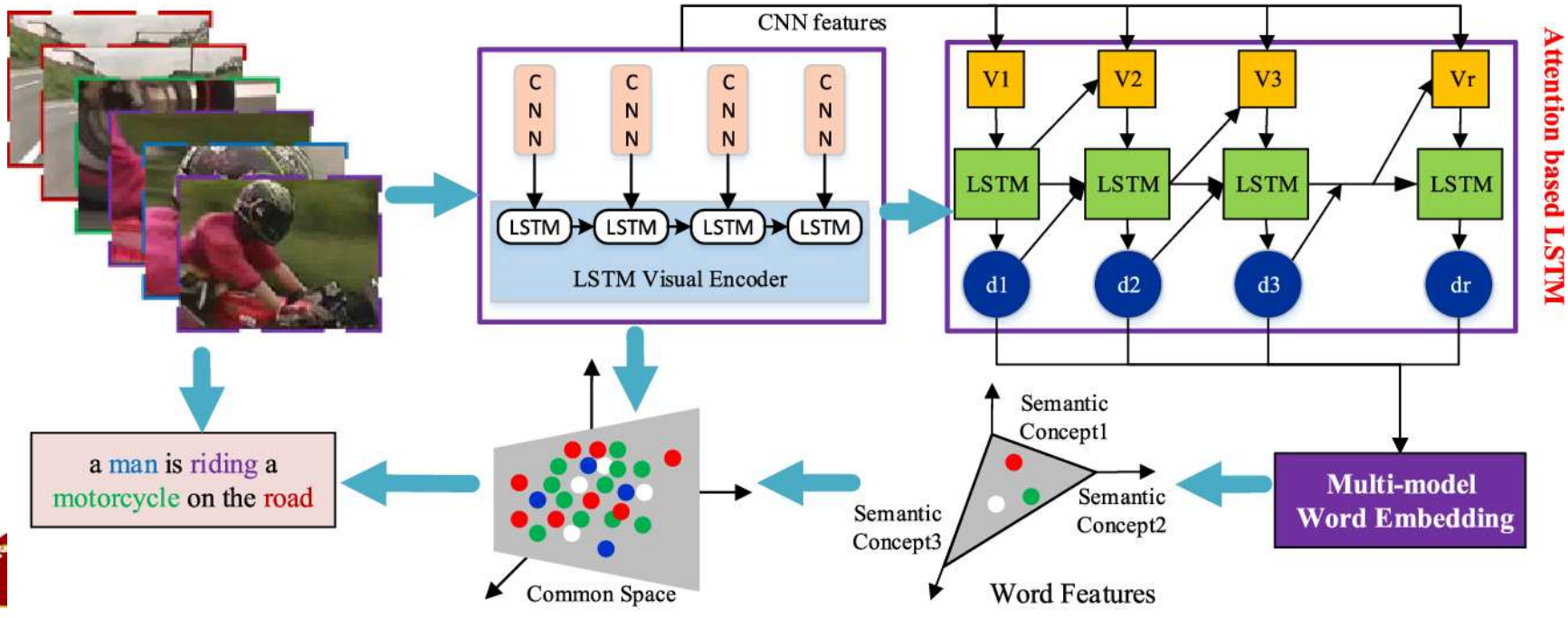
Aplicações de LSTM

- Análise de sentimentos:
  - Modelo regional CNN-LSTM para prever as classificações valência-excitação de textos: Captura informações locais (regionais) dentro de sentenças e dependência de longa distância entre sentenças;



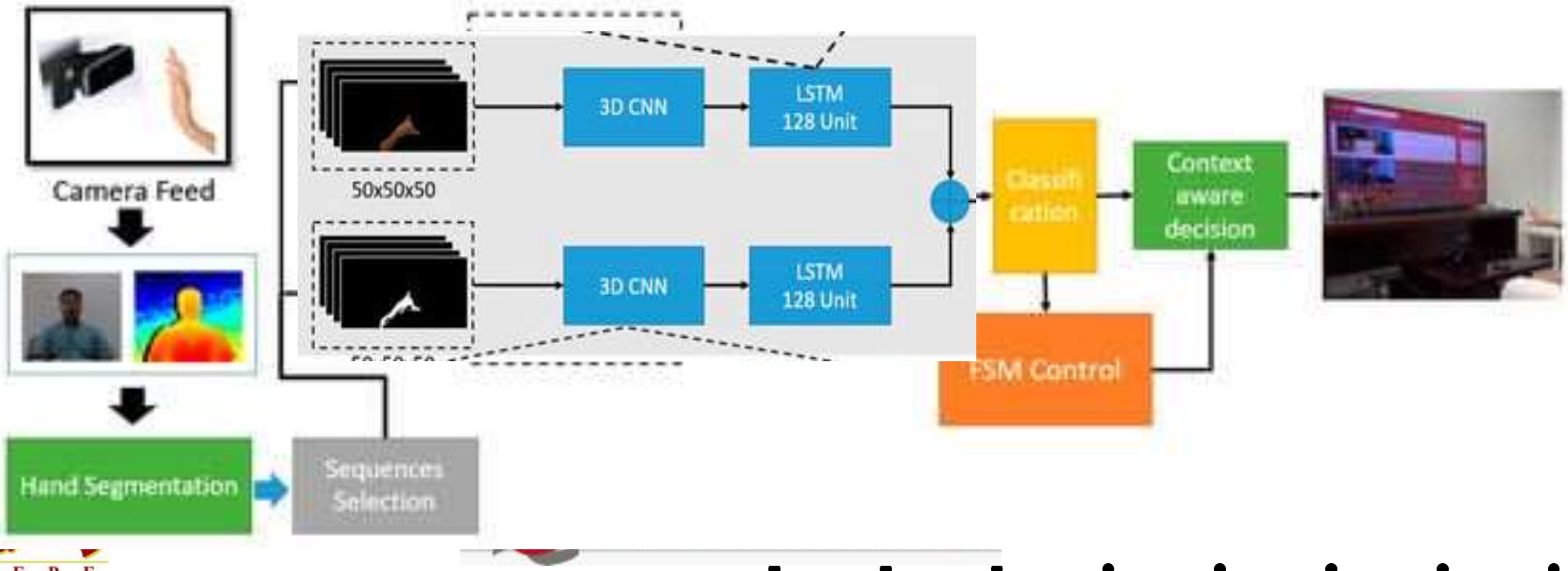
Aplicações de LSTM

- Legendagem de imagem e vídeo:
  - Gao et al. (2017) propuseram LSTM baseada em atenção (aLSTMs);
  - Integra mecanismo de atenção e LSTM que captura estruturas salientes de vídeo e explora correlação entre representações multimodais (palavras e conteúdo visual) para transferir vídeos para sentenças;



Aplicações de LSTM

- Visão computacional:
  - Reconhecimento de gestos em situação em tempo real, combinando as dados RGB e profundidade como entrada para o modelo DNN;
  - Hakin et al. (2019) usaram 3D-CNN + LSTM para extração de características espaço-temporais da sequência de gestos;



Aplicações de LSTM

- Ilustrações de diferentes aplicações:

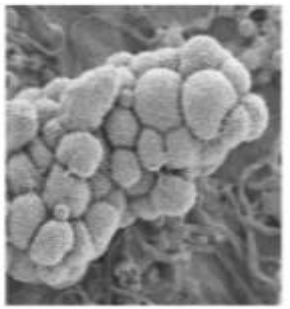

Object localization

Object detection


Image or video Segmentation


Autonomous Car


Security and Defense


Medicine and biology

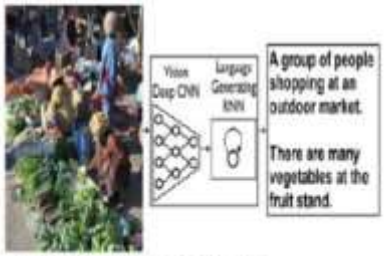
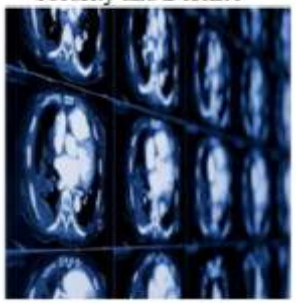


Image or Video Captioning


Media and entertainment

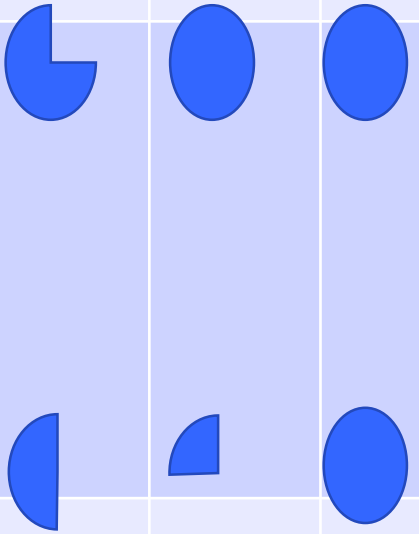
Machine translation

Speech recognition


Brian Cancer Detection


Skin cancer recognition

Frameworks para LSTM

Frame work	Easy of Use	Flexibility	Scalability	Description
TensorFlow				**TensorFlow** was developed by researchers and engineers from the Google Brain team. It is far and away the most commonly used software library in the field of deep learning (though others are catching up quickly). **TensorBoard:** Helps in effective data visualization using data flow graphs **TensorFlow:** Useful for rapid deployment of new algorithms/experiments
Keras				**Keras** is written in Python and can run on top of TensorFlow (as well as CNTK and Theano). The TensorFlow interface can be a bit challenging as it is a low-level library and new users might find it difficult to understand certain implementations.
PyTorch				**Torch** is a scientific computing framework that offers wide support for machine learning algorithms. It is a Lua based deep learning framework and is used widely amongst industry giants such as ***Facebook, Twitter and Google***. It employs CUDA along with C/C++ libraries for the processing and was basically made to scale production of building models and overall flexibility. **PyTorch** is basically a port to Torch deep learning framework used for constructing deep neural networks and executing tensor computations that are high in terms of complexity.
Caffe				**Caffe** is another popular deep learning framework geared towards the image processing field. It was developed by Yangqing Jia during his Ph.D at the University of Claifornia, Berkeley. Caffe's support for **recurrent networks**

Referências

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
- Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222-2232.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
- Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019.
- Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, *53*, 5929-5955.
- Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, *31*(7), 1235-1270.