DEMO BY DEMO

Chapter A
RECURRENT NEURAL NETWORK

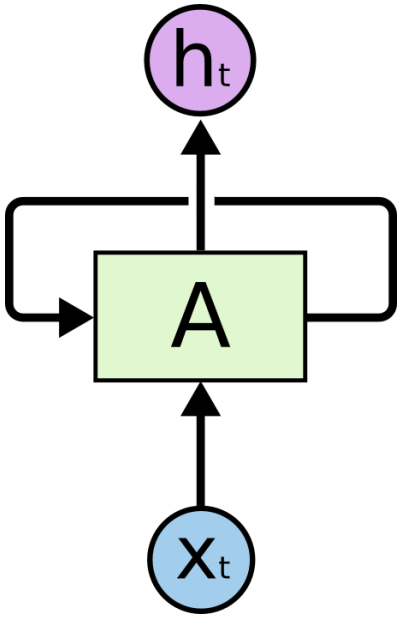
Loaii abdalslam

النهاردة هنتكلم عن ال Recurrent Neural Network بشيء من الاستفاضة والتوضيح , في سلسلة Demo by Demo الهدف الأساسي أننا نحاول نبسط ونسهل علي بعض كل المواضيع عشان المعلومة توصل بسهولة شديدة .

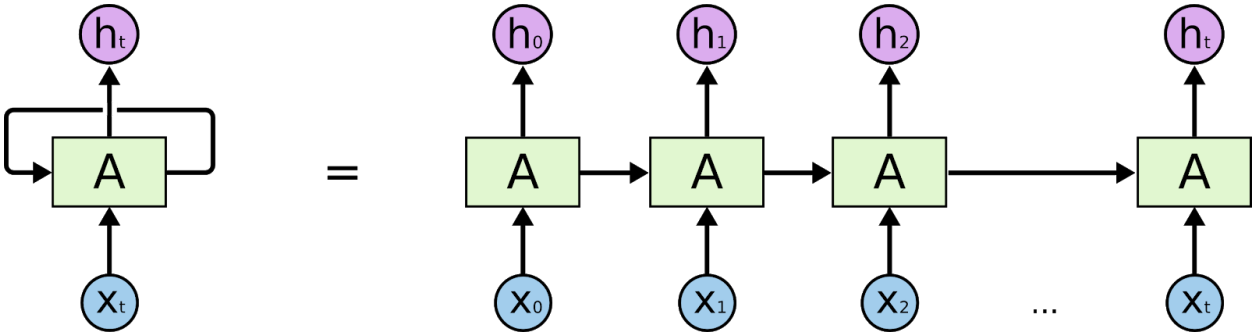
أنت وانت بتفكر مش بتبدأ تفكر كل مرة من البداية , انت عقلك شغال بقاله فترة كبيرة جدا تقريبا من يوم ما اتولدت , كل ما تبدأ تفكر في حاجة مش بتبدأ تفكر من البداية , بتستعمل ذاكرتك في انك توصل المعلومة المخزنة بداخلها

تخيل معايا لو عايز تعرف السما هتمطر او مش هتمطر ,كل الي هتعمله انك تبص للسما وهتشوف لو لون السحاب غامق فيه احتمالية للمطر لو السحاب تصادم , مش بتحتاج ساعتها انك تعرف يعني ايه سحاب ويعني ايه بخار ماء ويعني ايه أمطار , كل الي بتعمله انك بتستخدم ال Memory بتاعتك في انك تستخدمها علشان تصنف السحابة هل هي سحابة ممطرة ام سحابة غير ممطرة .

هو ده أبسط مثال علشان تقدر تعرض فيه ال Recurrent Neural Network - الشبكات العصبية الالتفافية وسميت الالتفافية لأنها ملتفة حول نفسه , او ممكن تقول عليها تكرارية لأنها بتتكرر حول نفسها وبشكل مبسط أكتر هي تعتبر جواها Loops لأن ده يسمح باستمرارية المعلومات جواها .



عايزك تتخيل معايا الأتي شايف ال Input ده x_t (إكس تي) دا يعتبر هو يعتبر الـ Input بتاع ال Neural Network الي نرمز لها بالرمز A وال Output بتاعنا هيكون ال h_t لو ركزت شوية هتلاقي إن ال RNN تعتبر هي هي ال NN العادية ولكن متكررة , الهدف من التكرار هو ضمان تدفق المعلومات من شبكة واحدة للشبكة الثانية الداخلية - كالمثال في الصورة - أثناء عملية التدريب Training كلها , فى خلال ال Loop داخل الشبكة أثناء عملية الإلتفاف او التكرار هنسمي العملية دي بالخطوة Step .



طبيعة الشبكة إنها شبه السلسلة مرتبطة إرتباط قوي جدا بالتسلسلات والقوائم وتعتبر $[n1, n2, n3, n4, n5]$دي البنية التحتية للبيانات المستخدمة في ال RNN انها لازم تكون Sequence وبشكل أدق انها تكون على شكل Array :

$$[n1, n2, n3, n4, n5]$$

$$[[n1, n2, n3], [n1, n2, n3]]$$

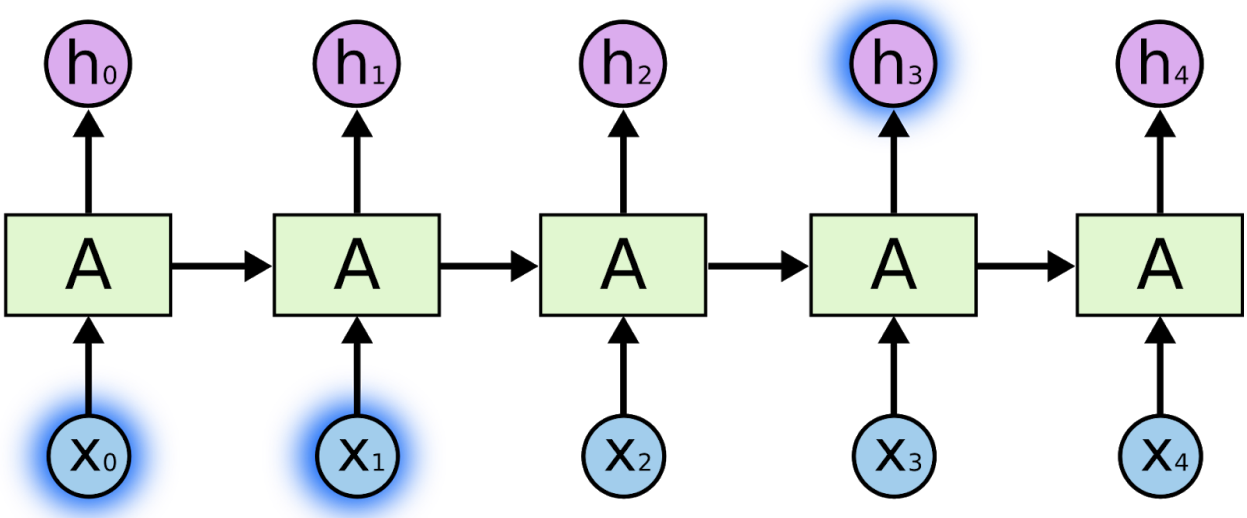
أشكال الـ Sequence هتتغير باختلاف الداتا بتاعتك ونوع ال Task الي بتحاول انك تنفذه ولكن هنرجع بشئ من التفصيل للجزئية دي , الشبكة تعتمد علي الداتا المتتابعة لأن الذاكرة هتقوم فإنها تخزن المعلومات في الذاكرة بشكل ما يحافظ على الترتيب .

تطبيقات ال RNN واسعة جدا في السنين الي فاتت قدرت انها تنفذ تطبيقات قوية على سبيل المثال , التعرف على الكلام , نمذجة اللغة , الترجمة , التنبؤ بالبورصة , التنبؤ بحركة المرور , صاحب الفضل في نجاح ال RNN كفكرة وتطبيق نوع معين منها اسمه ال LSTM وهو الي يرمز له اختصارا LONG SHORT TERM MEMORY وهندخل فيه دلوقتي بشئ من التفصيل

ممكن تحس لثواني أن ال RNN & LSTM متشابهيين ومفيش اي فرق بينهم ولكن هو في الواقع فيه فرق , فيه فرق بين تدفق المعلومات وتذكرها وتذكرها بشكل عام على الأمد الطويل والقصير .

ال LSTM بتتيح ليك انك توصل لأي معلومة داخل الشبكة على مدار فترة التمرين سواء كانت طويلة الأجل او قصيرة الأجل وتستخدمهم تاني من أول وجديد

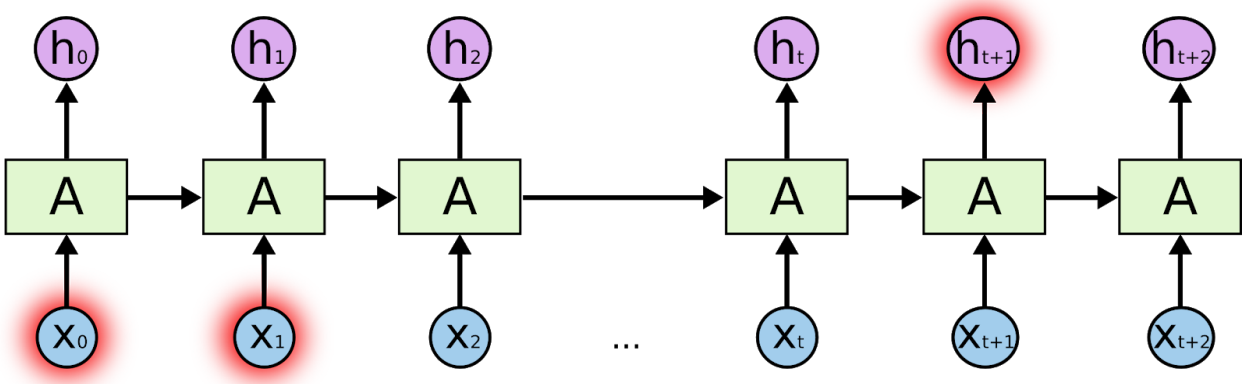
ولو طبقناها على مثال بسيط مثلا فعلى سبيل المثال الفيديو يتكون من مجموعة من الFRAMES وال Frames دي هي مجموعة من الصور كبيرة جدا في ثواني معدودة , وأثناء عملية التعليم الموديل بيحاول انه يفهم ال Frame رقم 3 بيرمز لأيه , قولنا قبل كده ان ال Frames بيكون مجموعة من اللقطات في وقت صغير ومن فهم الModel لل Frame رقم 2 يقدر انه يستعين بيه في فهمه Frame رقم 2 ويقدر يستعين داخليا ببعض المعلومات في ال Frame رقم 3 نفسه علشان ميقعش فيها تاني وهو يستعين الداتا بتاعت ال Frame رقم 2 , يعني هو بيحاول يسجل الداتا على الأمد الطويل الي هي بتكون Long Term (طويلة الأمد) على مدار الـ Frames الي بتتغير بداخل الفيديو , وعلي ال Short Term أثناء عملية ال Training علي ال Frame الداخلي بحيث ميغلطش فيها تاني .



وكما في الصورة الي فوق بالظبط ممكن نستعين بداتا قديمة علشان تحسين عملية التعليم بتاعتنا علي الداتا الجديدة , كل ده اثناء عملية ال Training دون تدخلنا تماما.

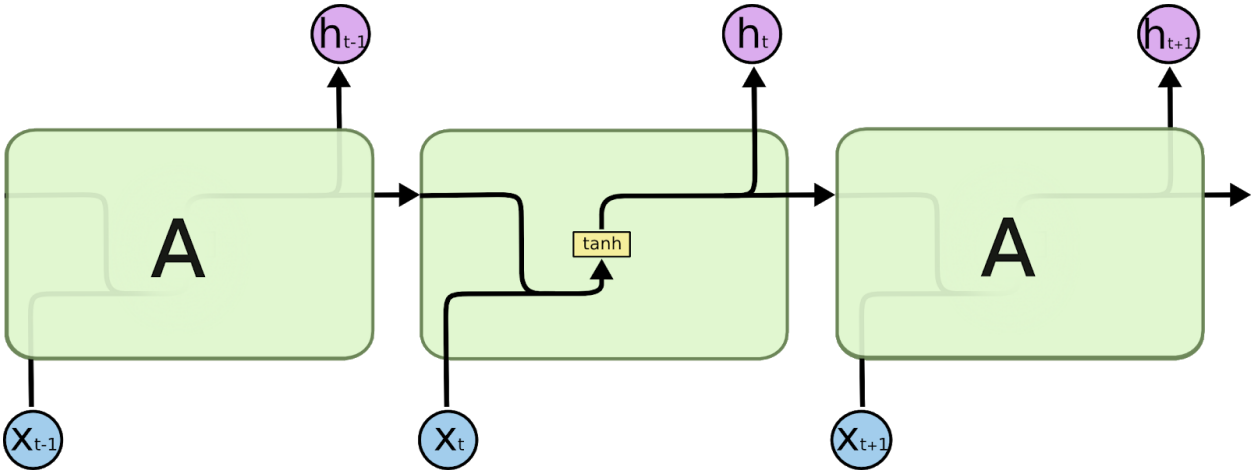
طبعا هنحاول اننا نبسط المثال بمثال أكثر عملية شوية وهو لو قلت اننا عايزين نبني نموذج يتنبأ بالكلمة التالية يعني احطله 4 كلمات يقولي الكلمة ال5 , بافتراض أننا في الداتا بتاعتنا عندنا جملتين الأولى :
" انا احب مصر جدا " والثانية " انا أحب فرنسا أكثر من أي شئ " , أثناء عملية التدريب وهو بيحاول انه يكمل الجمل الي فوق دي , لما يوصل لجملة أنا أحب

فرنسا يذكر أن سياق الكلام بيقول ان لازم "أسم البلد" يكون موجود بعد كلمة "بحب"
وهنا هيكون وفر علي نفسه عناء انه يتعامل مع كل كلمة بشكل منفصل , هو بيحاول انه يجمل اكبر قدر من المعلومات من كل البيانات الموجودة ويحتفظ بيها لأنه ممكن يستخدمها قدام او ممكن يستخدمها كإضافة بسيطة تحسن من الأداء العام بتاعه في المستقبل .

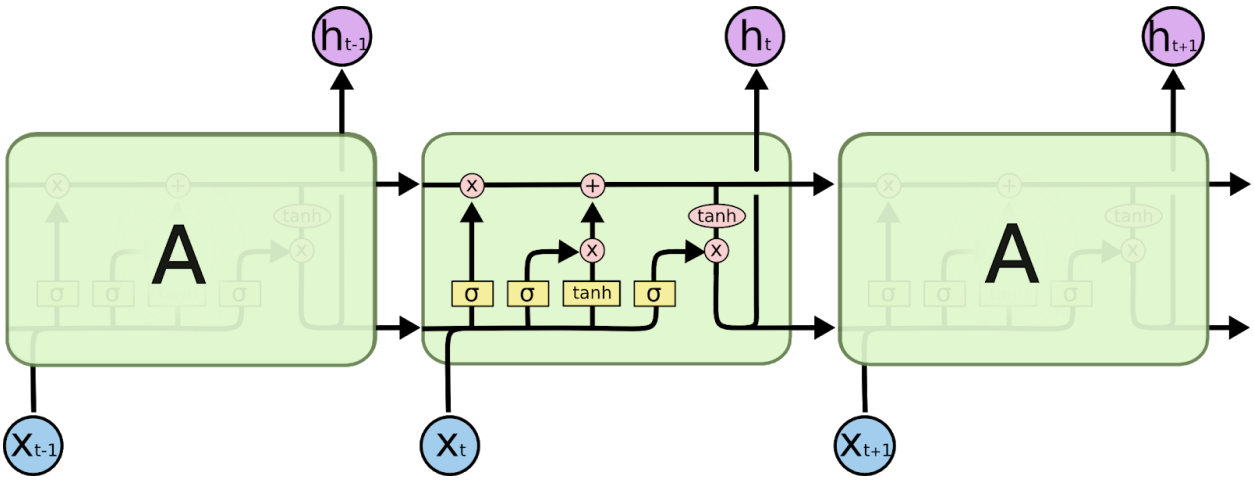


شبكات ال LSTM اتصممت علشان تحاول انها تظبط التبعية طويلة الأجل ودي كانت مشكلة بتعاني منها ال RNN لأن ال RNN مكنتش مصممة انها تتذكر الأنماط والتجارب والنواتج علي تبعية من النوع الطويل وهنا جه دور ال LSTM انها تحل محل ال RNN في حل المشكلة دي .

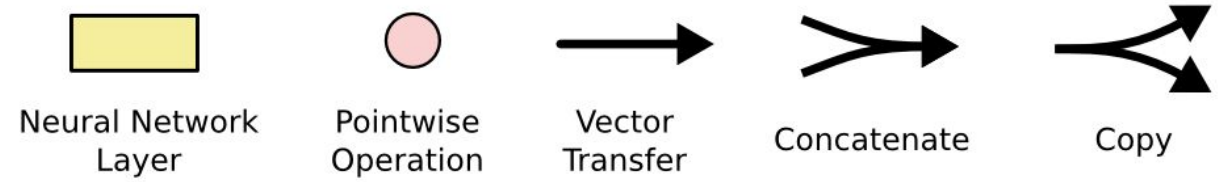
نحاول نتعمق شوية في البنية الداخلية للشبكات ال Rnn والفرق بينها وبين ال LSTM واللي هتكون عندها الشكل التالي لو حاولنا اننا نقرب شوية هنلاقي ان هنا عندنا تكرار من 3 شبكات من نفس النوع تربطهم Activation من النوع Tanh



ولو حاولنا اننا نتعمق داخل ال LSTM هتلاقي إن الموضوع مختلف شوية عن بنية ال RNN , هو أيوة يتبع نفس فكرة السلسلة ولكن الفرق هنا أنه بدل من Neural Network واحدة متكررة هنا عندنا 4 متكررين مربوطين بطريقة خاصة



التفاصيل عن كيفية تكوين ال LSTM نفسها مش مهمة إطلاقا دلوقتي بالنسبالك ولكن المهم انك تكون عارف إنها تفرق تماما عن ال RNN , ومعاك بعض الرموز الي هتقدر تستعين بيها علشان تفهم الرسمة الي فوق .



التطبيق العملي :

الفقرة دي هنحاول اننا نطبق بشكل عملي مبسط جدا المعلومات الي عرضناها فوق علشان نحاول نوسع الإستيعاب بمزجه بتطبيقات عملية , حاول علي قد ما تقدر تطبق عملي وتتوسع بالتطبيقات العملية .

إحنا هنستخدم Keras في المثال ده وهنعمل Importing لكل ال Modules بتاعتنا ومنهم ال LSTM

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from numpy import array
from keras.models import load_model
```

في الخطوة دي المفروض اننا هنعمل Function هتعملنا ال Data , لاحظ اننا محتاجين نستخدم ال Reshape لل X بتاعتنا لأن ال LSTM او ال RNN عموما لازم ال INPUT بتاعها يكون 3D يعني ثلاثي الأبعاد بحيث انها بتقبل ثلاث قيم :

```
def get_train():
  seq = [[0.0, 0.1], [0.1, 0.2], [0.2, 0.3], [0.3, 0.4]]
  seq = array(seq)
  X, y = seq[:, 0], seq[:, 1]
  X = X.reshape((len(X), 1, 1))
  return X, y
```

المفروض بعد كده اننا نبدأ نبني ال Model بتاعتنا وهو بيكون Sequential Model يتكون في البداية من LSTM وبنحط فيها الـ Node Number بتاعنا ونحط فيها ال Input Shape بتاعنا وهنا هو بيكون (1,1) وال Output هيكون حاجة واحدة بس هتخلي الـ Node بتاعته تساوي 1 , وهنحط ال Loss Function بتاعتنا المرة دي تساوي MSE وال Optimizer يساوي Adam من غير ما نظبط أي Configuration فيهم حاليا لأن ده يعتبر Baseline Model احنا محتاجين نطبق عملي بس وبعد كده ندوس علي آل Model Summary هيطلع Table فيه كل تفاصيل وعدد ال Params بتاعتنا داخل ال Model .

```
# define model
model = Sequential()
model.add(LSTM(10, input_shape=(1,1)))
model.add(Dense(1, activation='linear'))
# compile model
model.compile(loss='mse', optimizer='adam')
model.summary()
```

هنجيب الداتا بتاعتنا وهنبدأ ندرب الModel بتاعنا ونلاحظ هنا اننا محتاجين جدا يكون ال Shuffle =False لأنك تعاني معاناة كبيرة جدا , شبه ما قولنا ال- RNN LSTM بيعتمدو علي ال Sequence Data وتكون بالترتيب الصحيح , لأنها هيتبني عليها نتايج بعد كده بناء علي الترتيب وكم المعلومات الي نقدر نستخلص منه.

```
X,y = get_train()
model.fit(X, y, epochs=300, shuffle=False, verbose=2)
```

بعد كده نقدر اننا نعمل Predict للداتا بتاعتنا .

```
yhat = model.predict(X, verbose=0)
print(yhat)
```

سلسلة ال LSTM ممتدة جدا لأكثر من تطبيق ومنهم تطبيقات عملية ودي كانت البداية بشكل بسيط جدا .