CS167: Machine Learning

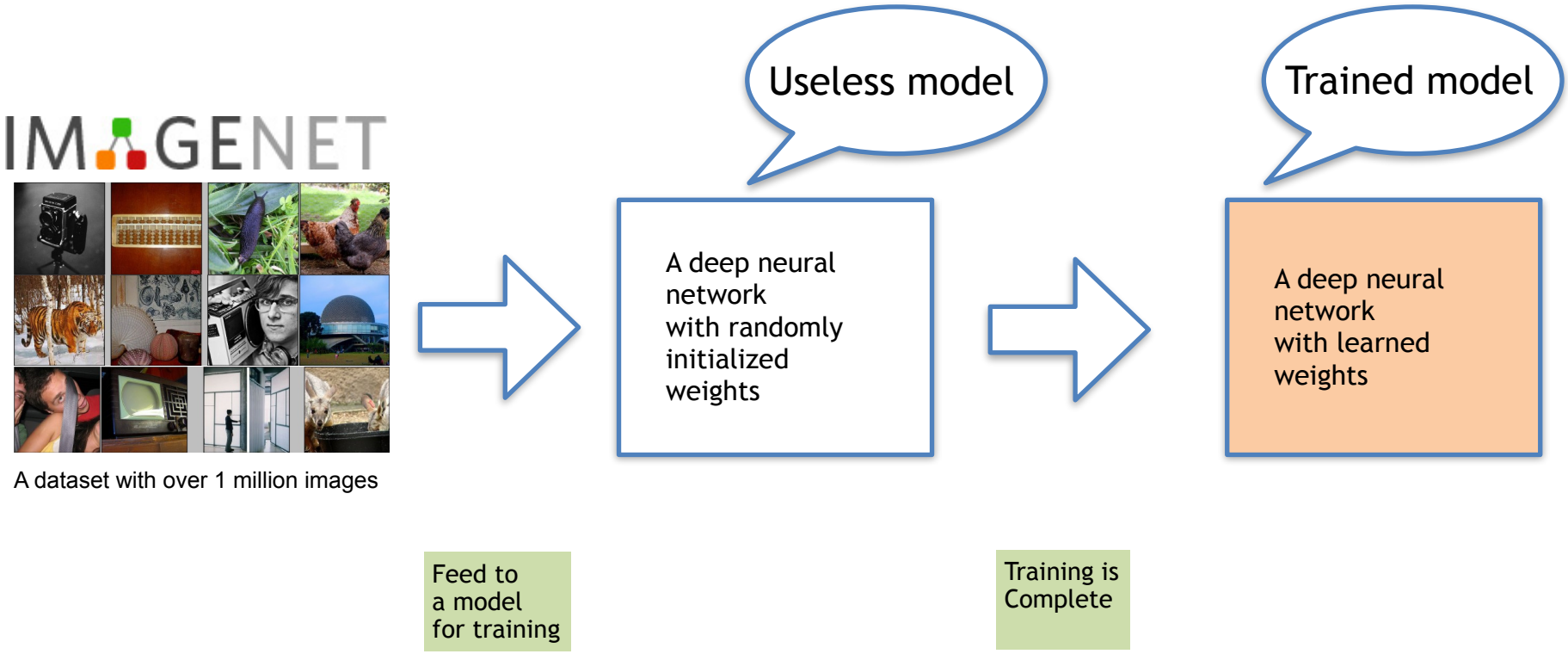
Recurrent Neural Networks (RNN)

Monday, December 2nd, 2024

Drake UNIVERSITY

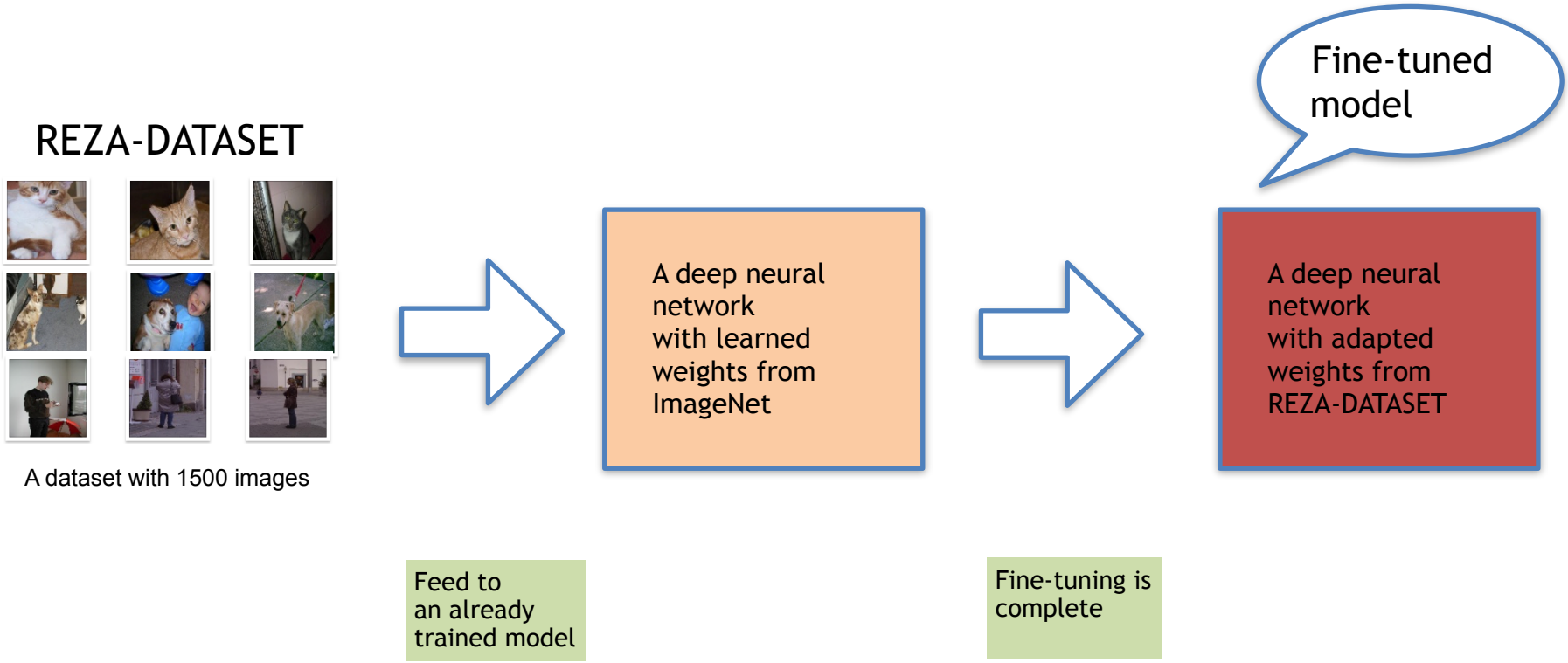
Recap: Training a Model

- **Training** refers to the process of training a model from scratch, often on a large and general dataset (e.g., ImageNet for image classification).



Recap: Fine-tuning a Model

- **Fine-tuning** refers to the process of taking a pre-trained model and further training it on a new or specific dataset. The initial model is often trained on a large and general dataset, e.g., ImageNet, and fine-tuning adapts the model to perform well on a more specific task or dataset.

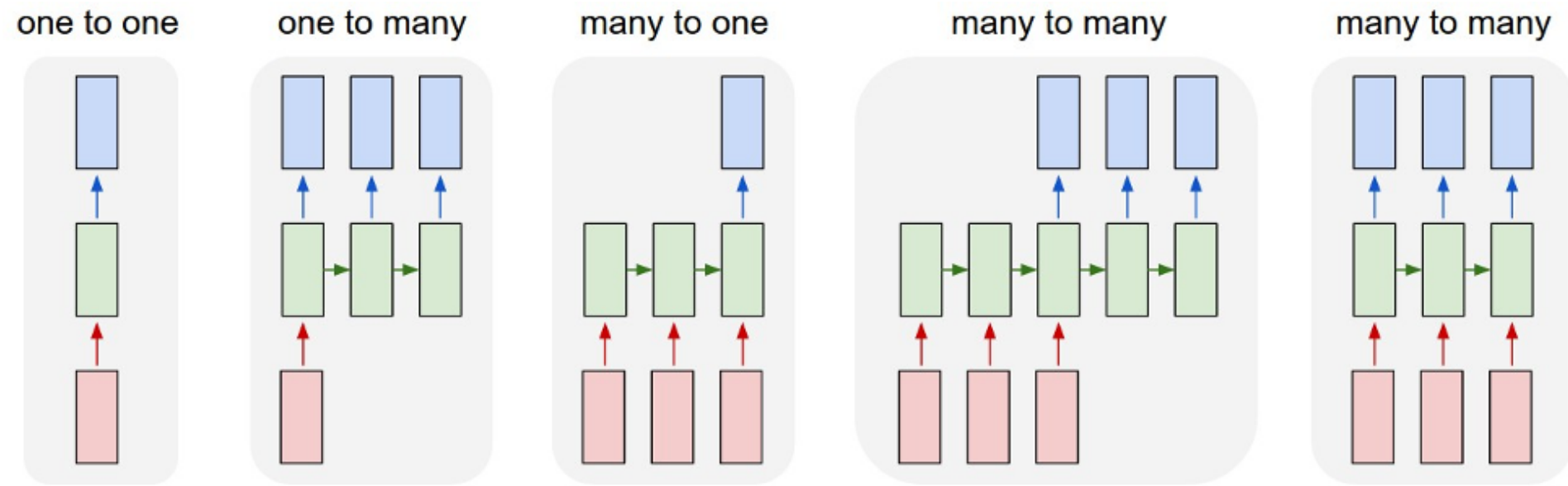


So far our modeling has been limited

- We've only worked with a certain type of data:
  - tabular data or image
  - no sequential data (the data does not need to be in a specific order)
  - the kind of predictions we have made have either been **classifications** (binary or multi-class) or **regressions**

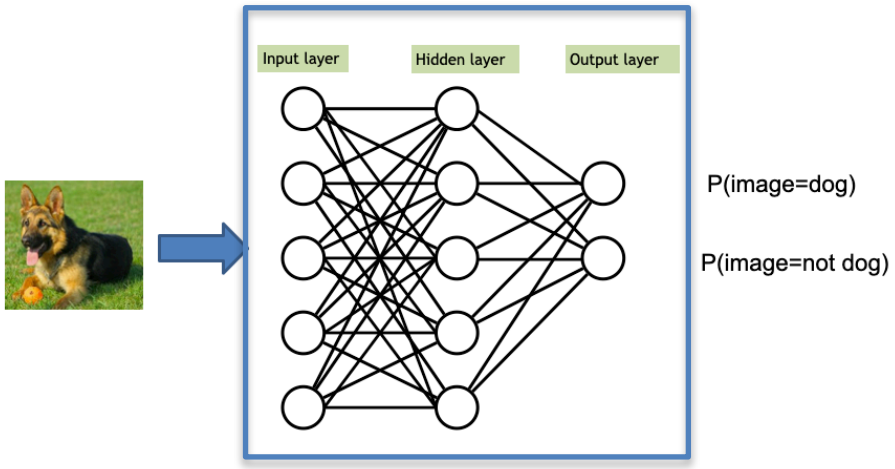
- Can you think of other kinds of target variables we might be interested in modeling?

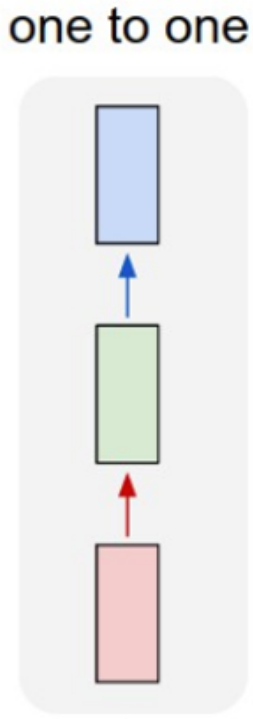
What if my input isn't the same size of my output?



One to one mapping task

- This is the mapping we have used so far
  - we input one training/testing example and make one prediction.

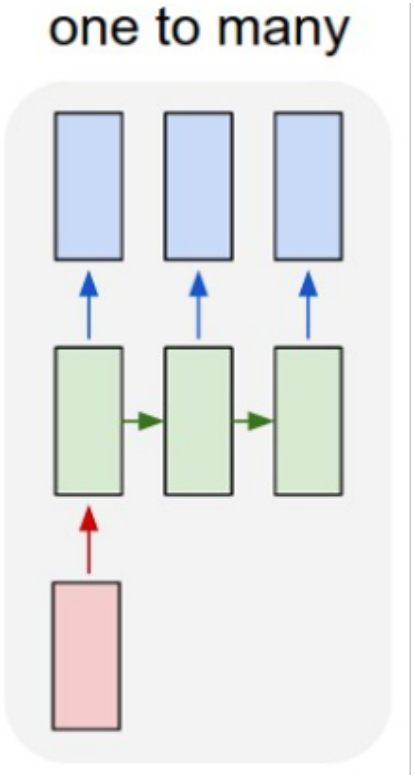




One to many mapping task

- Can anyone think of an example of a ML model that is **one to many**?
  - we input one training/test example, and output many predictions

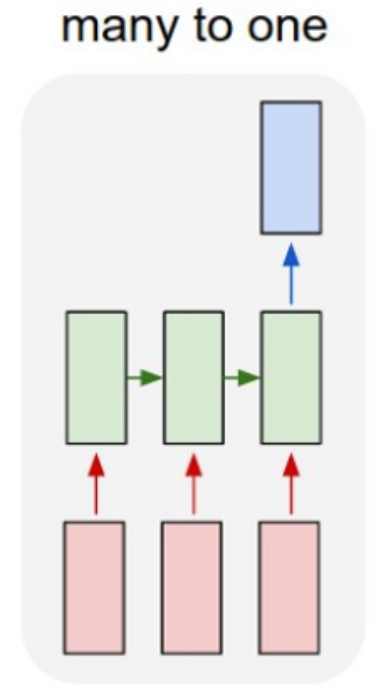
"A Dog catching a ball in mid air"



Many to one mapping task

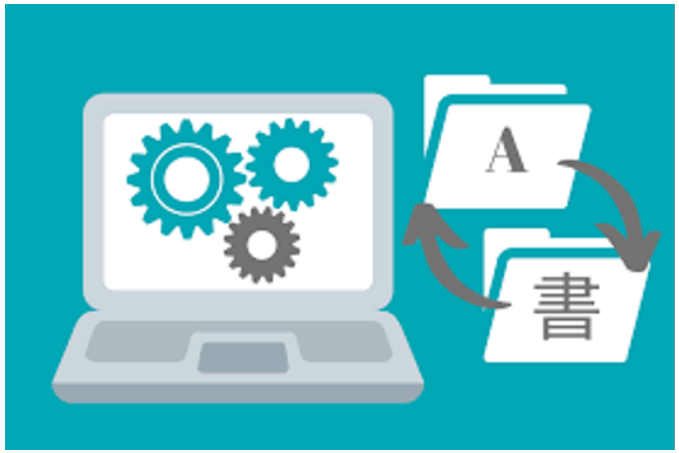
- Can anyone think of an example of a ML model that is **many to one**?
  - we input multiple things, and make one prediction from it

Review (X)	Rating (Y)
"This movie is fantastic! I really like it because it is so good!"	★★★★☆
"Not to my taste, will skip and watch another movie"	★★☆☆☆
"This movie really sucks! Can I get my money back please?"	★☆☆☆☆



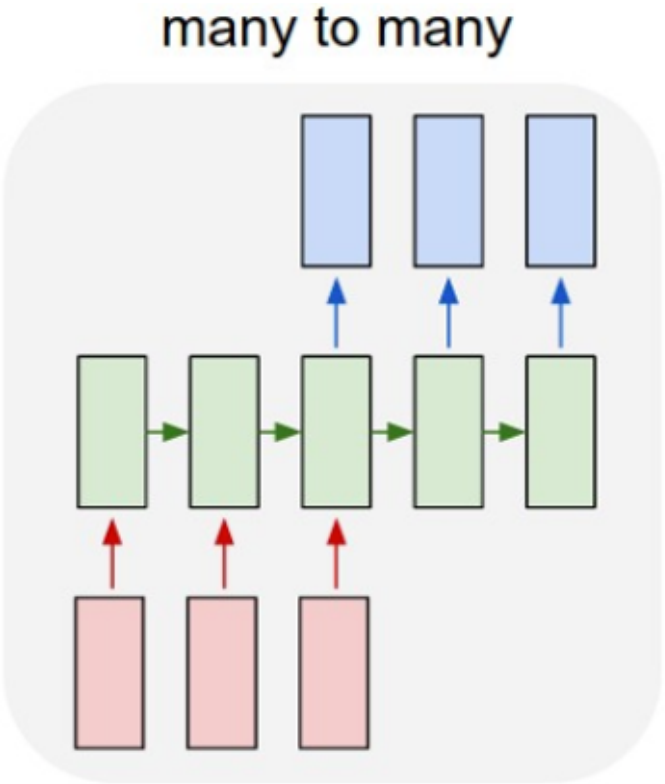
Many to many mapping task

- Can anyone think of an example of a ML model that is **many to many**?
  - we input multiple things, and make multiple predictions from it
  - the input and output size do not need to be the same length

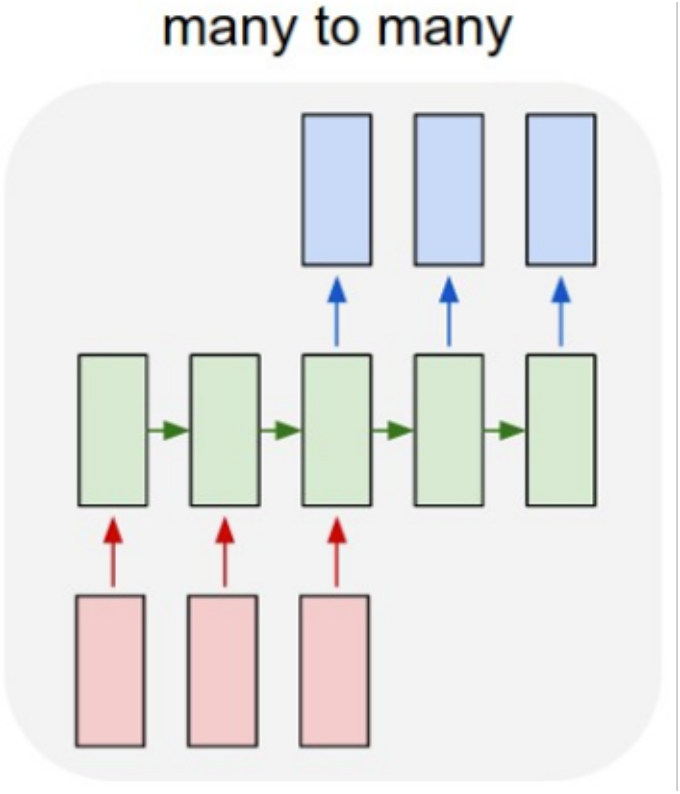


Machine Translation (translating from one language to another)

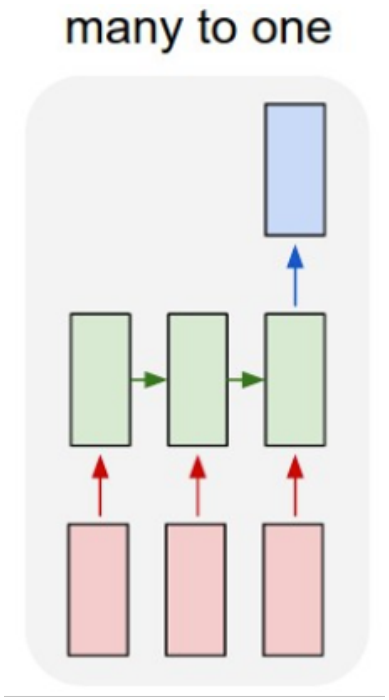
“Hello my name is" --> "Hola me llamo"



Many to many mapping task



Many to one mapping task



Today's Agenda

- Recurrent neural network (RNN)

- RNN implementation in PyTorch

Recurrent Neural Network (RNN)

- So far we've looked at feed-forward neural networks
  - Multilayer Perceptron (MLP)
  - Convolutional Neural Network (CNN)

- Recurrent neural networks are a class of neural networks that *allow previous outputs to be used as inputs* (ie, **"feedback loops"**) while having **hidden states**. They must use **context** when making a prediction.
  - Vanilla RNN
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)

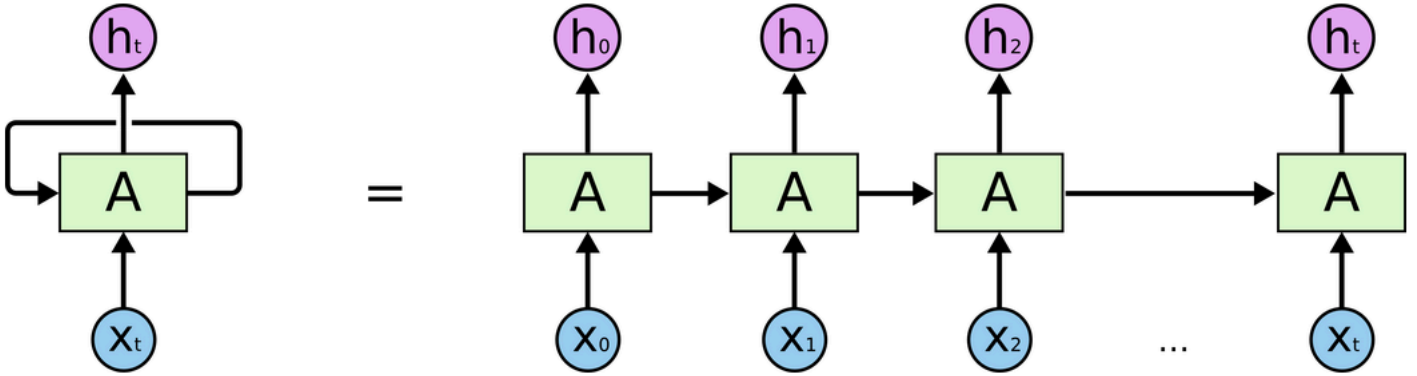
Recurrent Neural Network (RNN)

- Recurrent neural networks (RNN) include **"feedback loops"**
  - Vanilla RNN
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)

- RNNs are useful for learning and predicting sequences, e.g.
  - translations from one language to another
  - Sentiment prediction of a given text (eg, predicting positive or negative reviews)

Recurrent Neural Network (RNN)

- RNN contains “feedback loops”
  - Input X_t
  - Output h_t
- **Feedback loop** allows information to be passed from one step of the network to the next



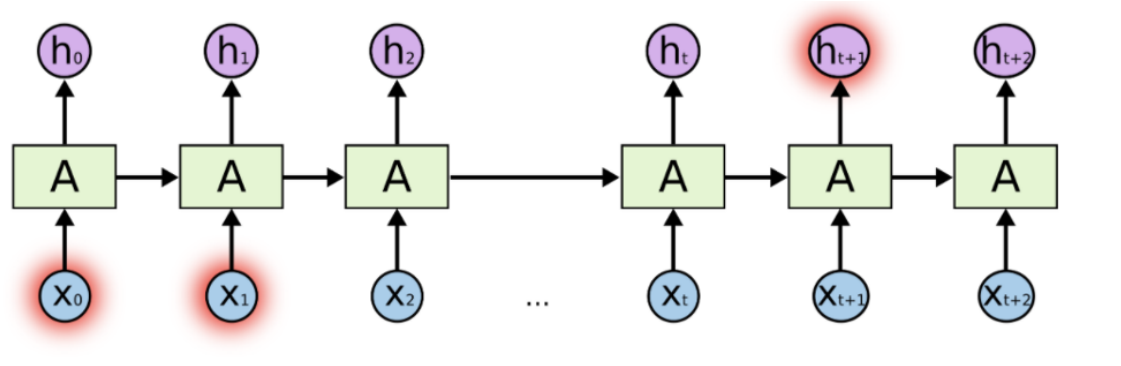
RNN model (rolled-up)

Unrolled version of the same RNN model

Figure credit: colah.gihub.oi

Recurrent Neural Network (RNN)

- Sometimes we only need to look at recent information to make a prediction:



- Consider a language model trying to predict the next word based on the previous ones. Let's predict the last word to this sequence:
  - `"The clouds are in the _____"`

Figure credit: colah.gihub.oi

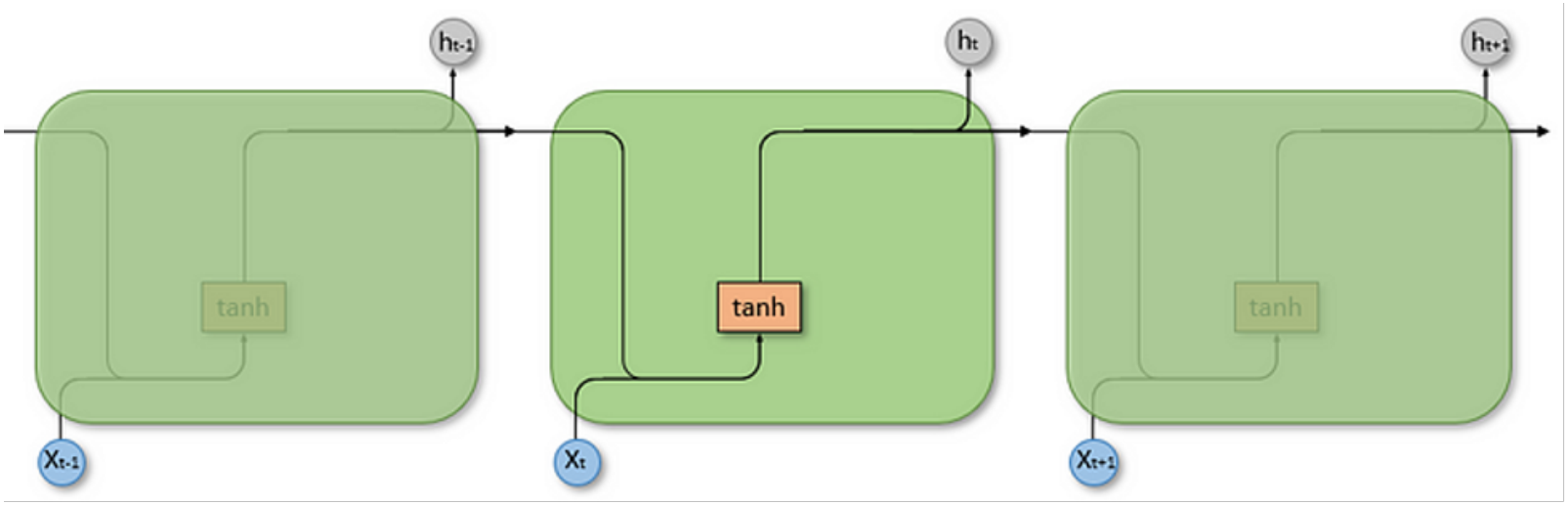
Recurrent Neural Network (RNN)

- We don't need much extra context to make a prediction for the previous example. However, consider the following sentence:
  - `"I grew up in France, I speak fluent _____"`

- We can guess from recent information that it will be the name of a language; we need the context of **France** to make a prediction.
  - `"I grew up in France, I speak fluent french”`

Simple Recurrent Neural Network (RNN)

- Each cell has **2 inputs**:
  - the past and the present
  - the output from the previous cell h_{t-1} and the current input into the cell X_t



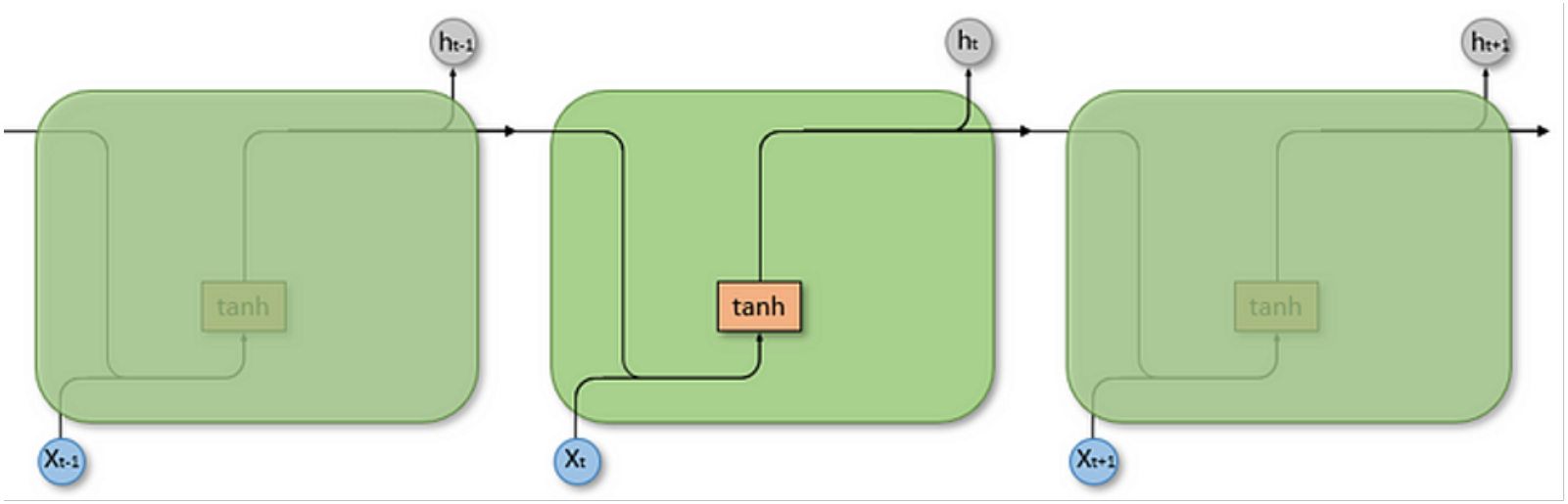
- This figure represents an unfolded RNN to help understand what is going on. The length of the cells is equal to the number of time steps of the input sequence.

Simple Recurrent Neural Network (RNN)

- In each cell, the input of the current time step x_t , the hidden state of the previous cell h_{t-1} and a bias are combined and then put through a $\tanh$ activation function as follows:

Learnable parameters

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$$

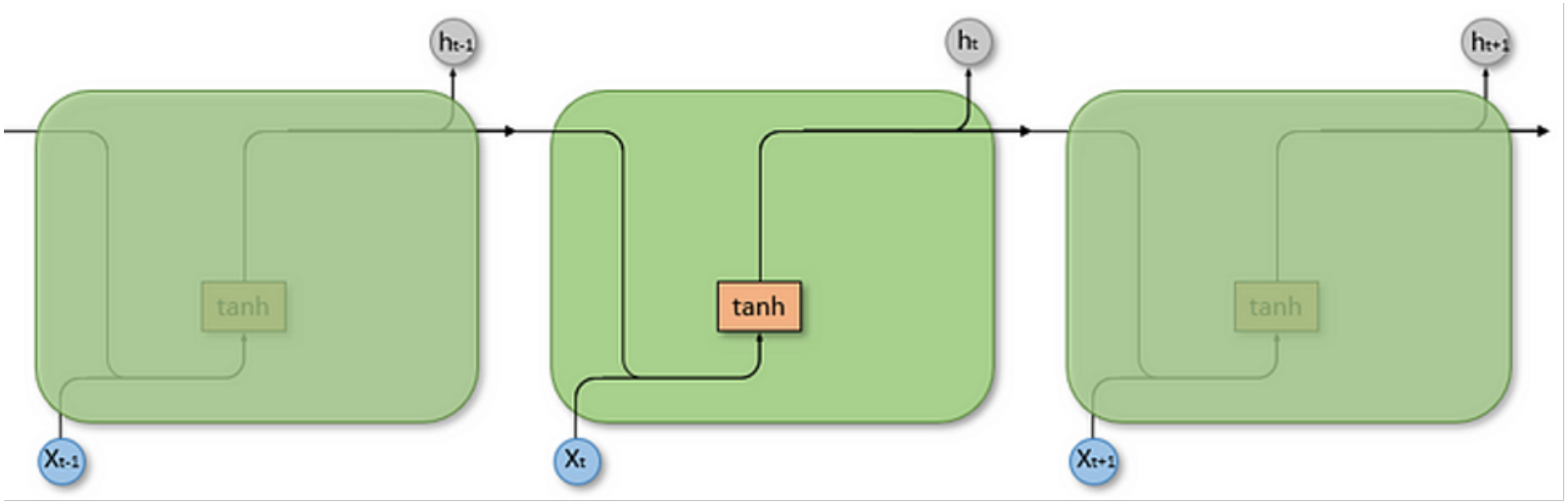


- here W_x and W_h are learnable weights for the RNN.

Simple Recurrent Neural Network (RNN)

- In each cell, the input of the current time step x_t , the hidden state of the previous cell h_{t-1} and a bias are combined and then put through a $\tanh$ activation function as follows:

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$$



- here W_x and W_h are learnable weights for the RNN.

Simple Recurrent Neural Network (RNN)

RNN Advantages

- Due to their short term memory, RNNs can handle sequential data and identify patterns in historical data.
- They can also handle inputs of varying lengths.

RNN disadvantages

- Suffers from the **vanishing gradient** problem:
  - The gradients that are used to update the weights during backpropagation become very small
  - Multiplying weights with a gradient that is so close to zero prevents the network from learning new weights
  - This stops the learning and results in the RNN forgetting what is seen in longer sequences
  - This problem gets worse with the more layers the network has

Long Short-Term Memory (LSTM)

- Each LSTM cell has **3 gates**:
  - **forget gate:** decides how much memory shall be kept
  - **input gate:** decides which information shall be added to the long-term memory
  - **output gate**: decides which parts of the cell state build the output, i.e. it's responsible for the short term memory

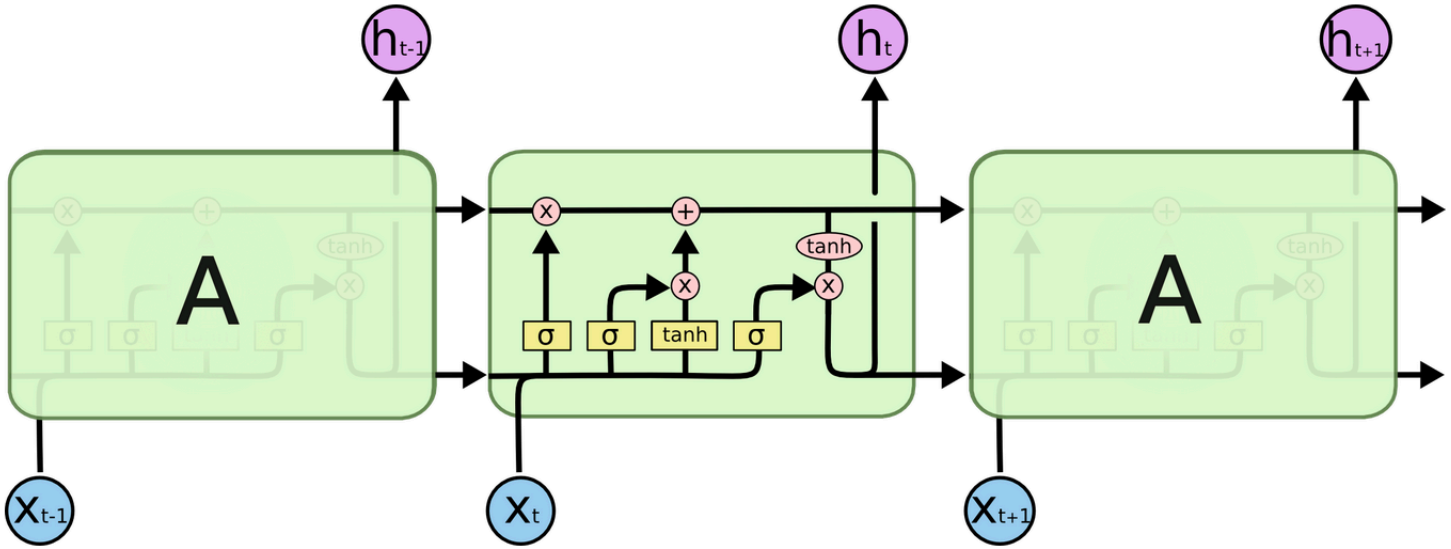


Figure credit: colah.gihub.oi

Long Short-Term Memory (LSTM)

LSTM Advantages

- Can capture both the short and long term patterns of a sequence.

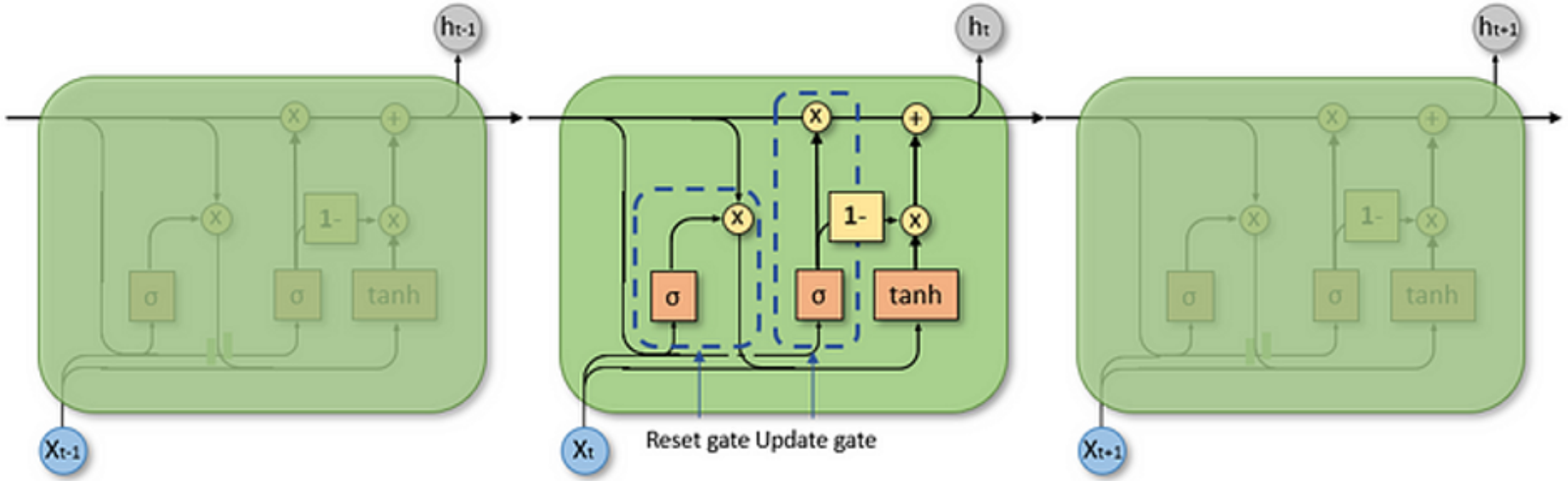
LSTM disadvantages

- Because LSTMs add complexity, they also are computationally more expensive, leading to longer training times.

Gated Recurrent Unit (GRU)

- Similar to LSTMs, the GRU solves the vanishing gradient problem, but they do so using fewer gates — making them effective, and fast.

- GRUs have 2 gates:
  - a **reset gate** which is responsible for the short-term memory as it decides how much past information is kept and disregarded
  - an **update gate** which is responsible for the long-term memory and is comparable to the LSTMs forget gate



Gated Recurrent Unit (GRU)

GRU Advantages

- Solve vanishing gradient problem
- Less computationally expensive than LSTMs, which makes them faster to train

GRU disadvantages

- GRUs do not have a separate hidden and cell state, so they might not be able to consider observations as far into the past as the LSTM