CS167: Machine Learning

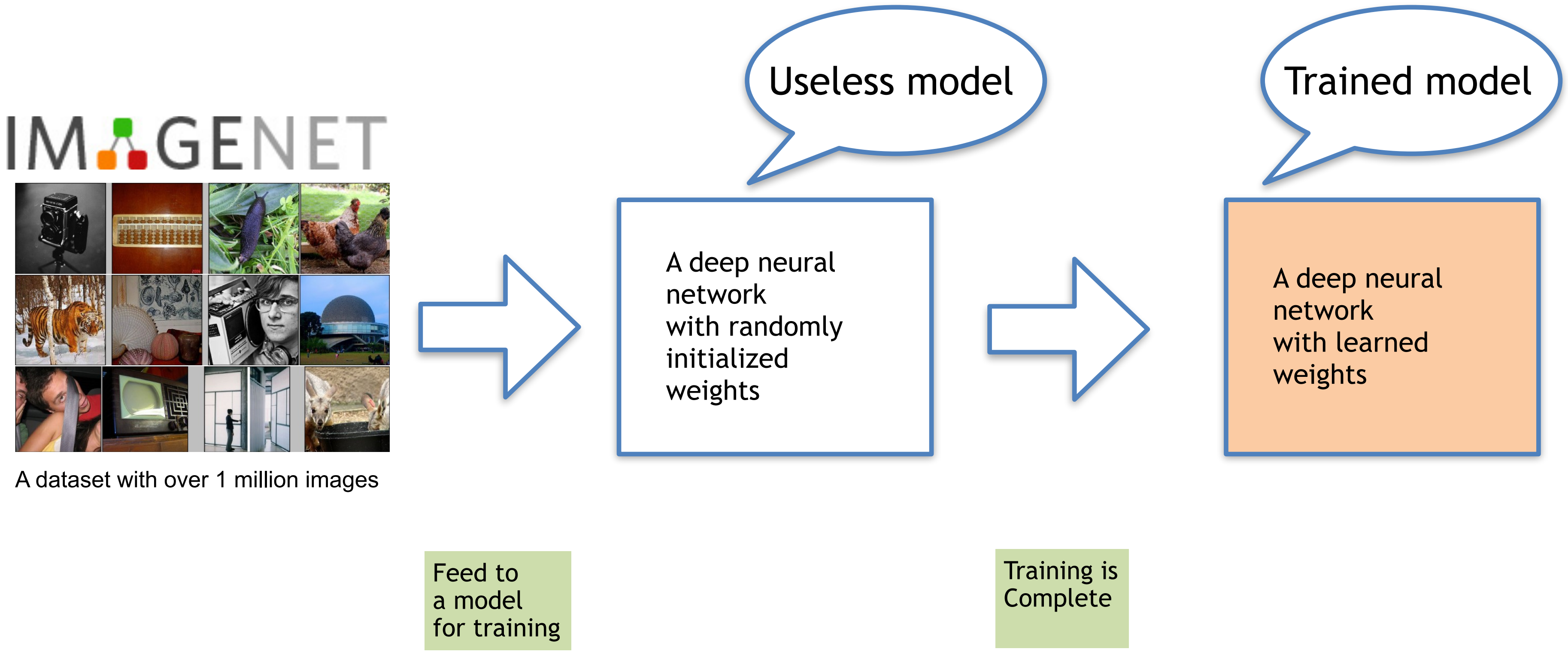
Recurrent Neural Networks (RNN)

Thursday, April 25th, 2024

Drake UNIVERSITY

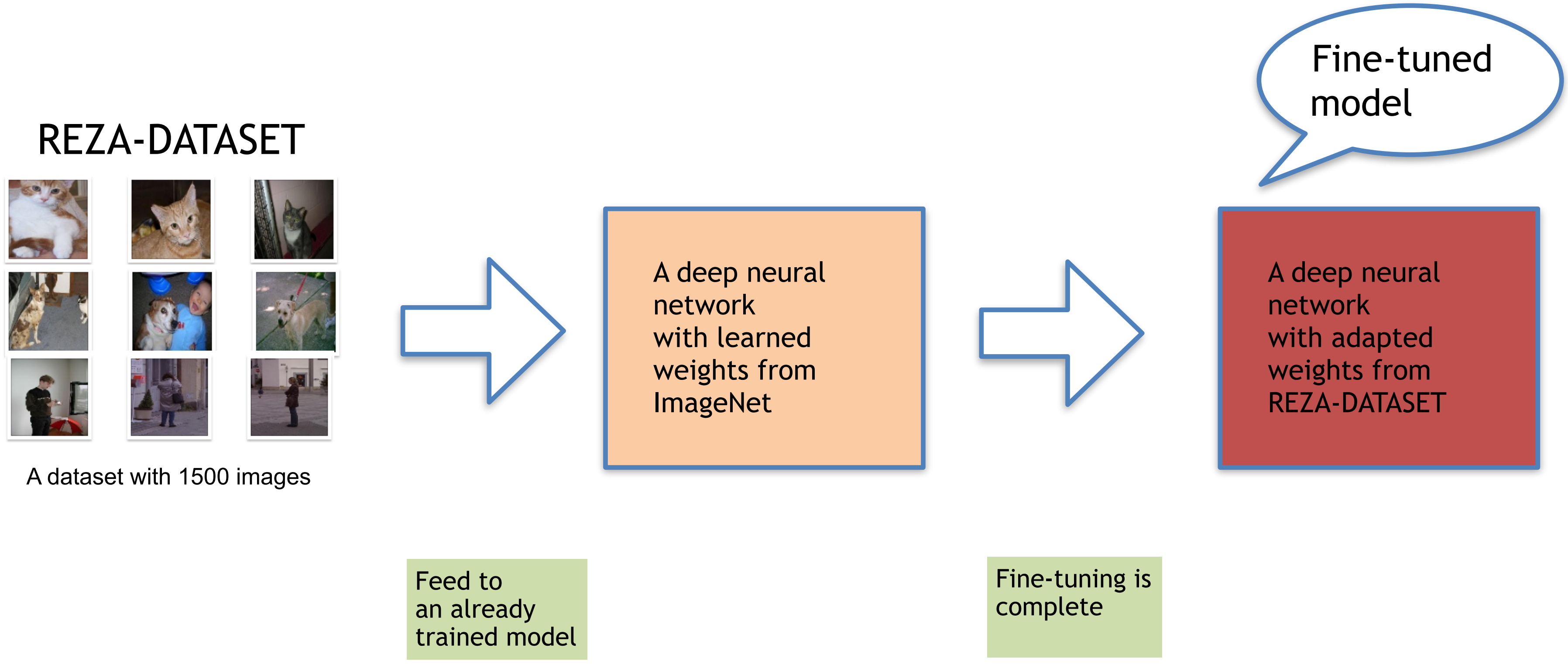
Recap: Training a Model

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

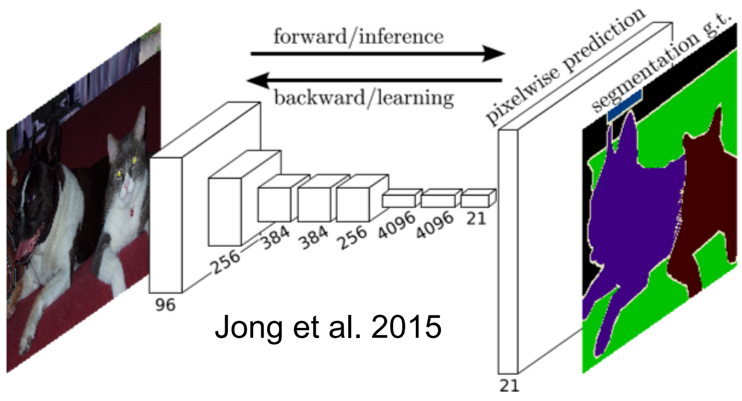

IMAGENET
A dataset with over 1 million images
Feed to a model for training
Useless model
A deep neural network with randomly initialized weights
Training is Complete
Trained model
A deep neural network with learned weights

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.

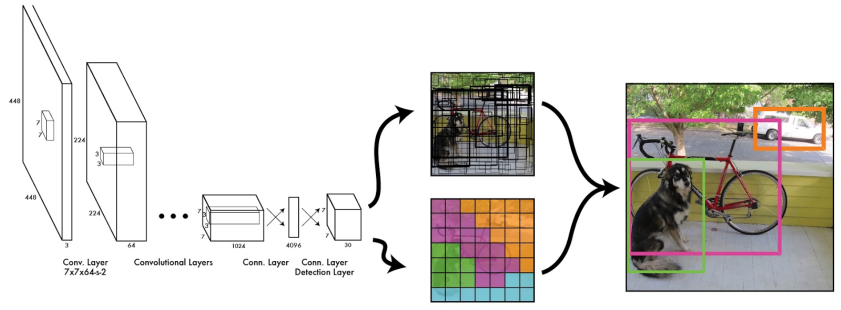

REZA-DATASET
A dataset with 1500 images
A deep neural network with learned weights from ImageNet
A deep neural network with adapted weights from REZA-DATASET
Fine-tuned model
Feed to an already trained model
Fine-tuning is complete

What else is CNN good for beyond image recognition?

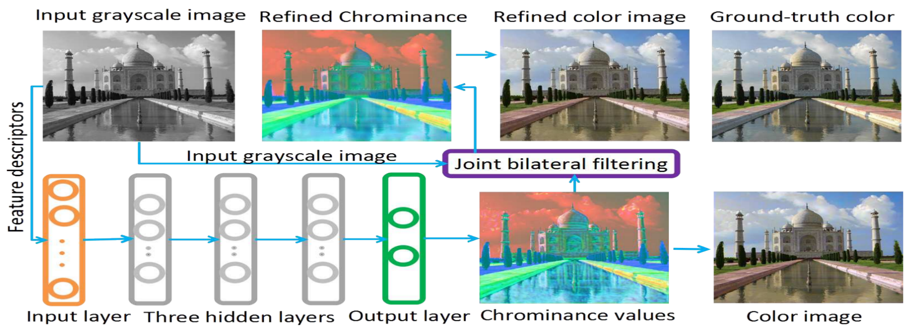


Jong et al. 2015

Semantic Segmentation

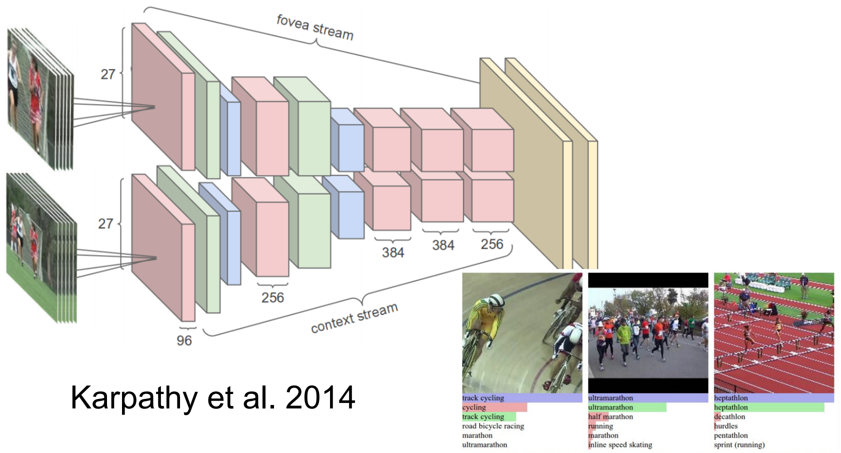


Object Detection



Cheng et al. 2015

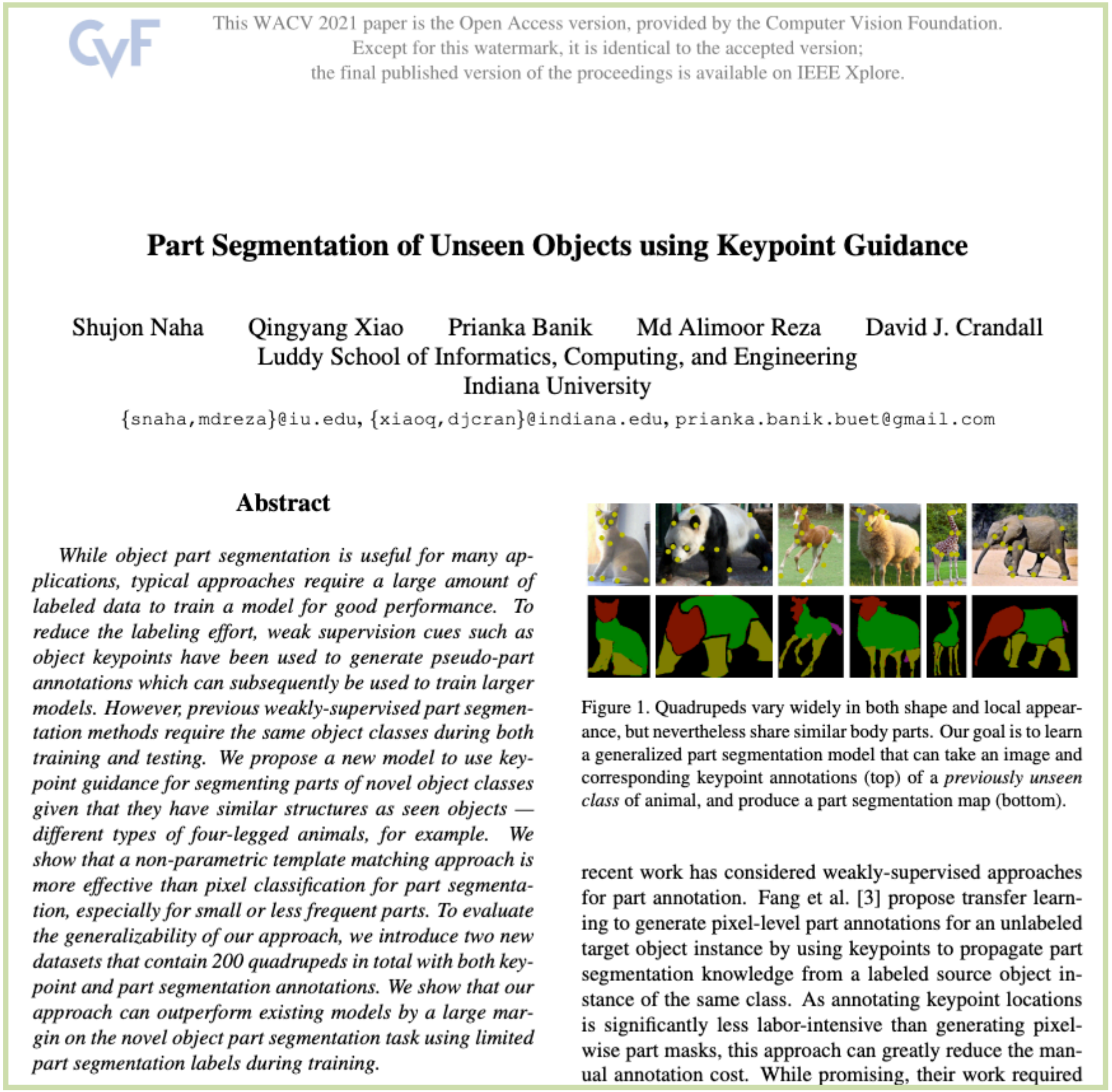
Image colorization



Karpathy et al. 2014

Video / Action Recognition

What else is CNN good for beyond image recognition?



Part Segmentation of Unseen Objects using Keypoint Guidance

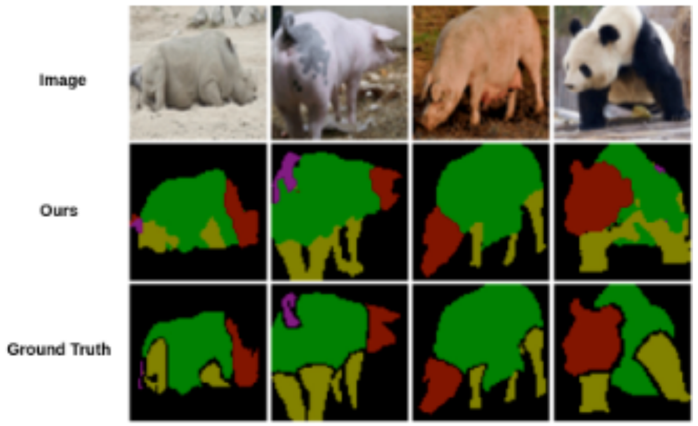
Shujon Naha Qingyang Xiao Prianka Banik Md Alimoor Reza David J. Crandall
Luddy School of Informatics, Computing, and Engineering
Indiana University
{snaha,mdreza}@iu.edu, {xiaoq,djcran}@indiana.edu, prianka.banik.buet@gmail.com

Abstract

While object part segmentation is useful for many applications, typical approaches require a large amount of labeled data to train a model for good performance. To reduce the labeling effort, weak supervision cues such as object keypoints have been used to generate pseudo-part annotations which can subsequently be used to train larger models. However, previous weakly-supervised part segmentation methods require the same object classes during both training and testing. We propose a new model to use keypoint guidance for segmenting parts of novel object classes given that they have similar structures as seen objects — different types of four-legged animals, for example. We show that a non-parametric template matching approach is more effective than pixel classification for part segmentation, especially for small or less frequent parts. To evaluate the generalizability of our approach, we introduce two new datasets that contain 200 quadrupeds in total with both keypoint and part segmentation annotations. We show that our approach can outperform existing models by a large margin on the novel object part segmentation task using limited part segmentation labels during training.

Figure 1. Quadrupeds vary widely in both shape and local appearance, but nevertheless share similar body parts. Our goal is to learn a generalized part segmentation model that can take an image and corresponding keypoint annotations (top) of a *previously unseen class* of animal, and produce a part segmentation map (bottom).

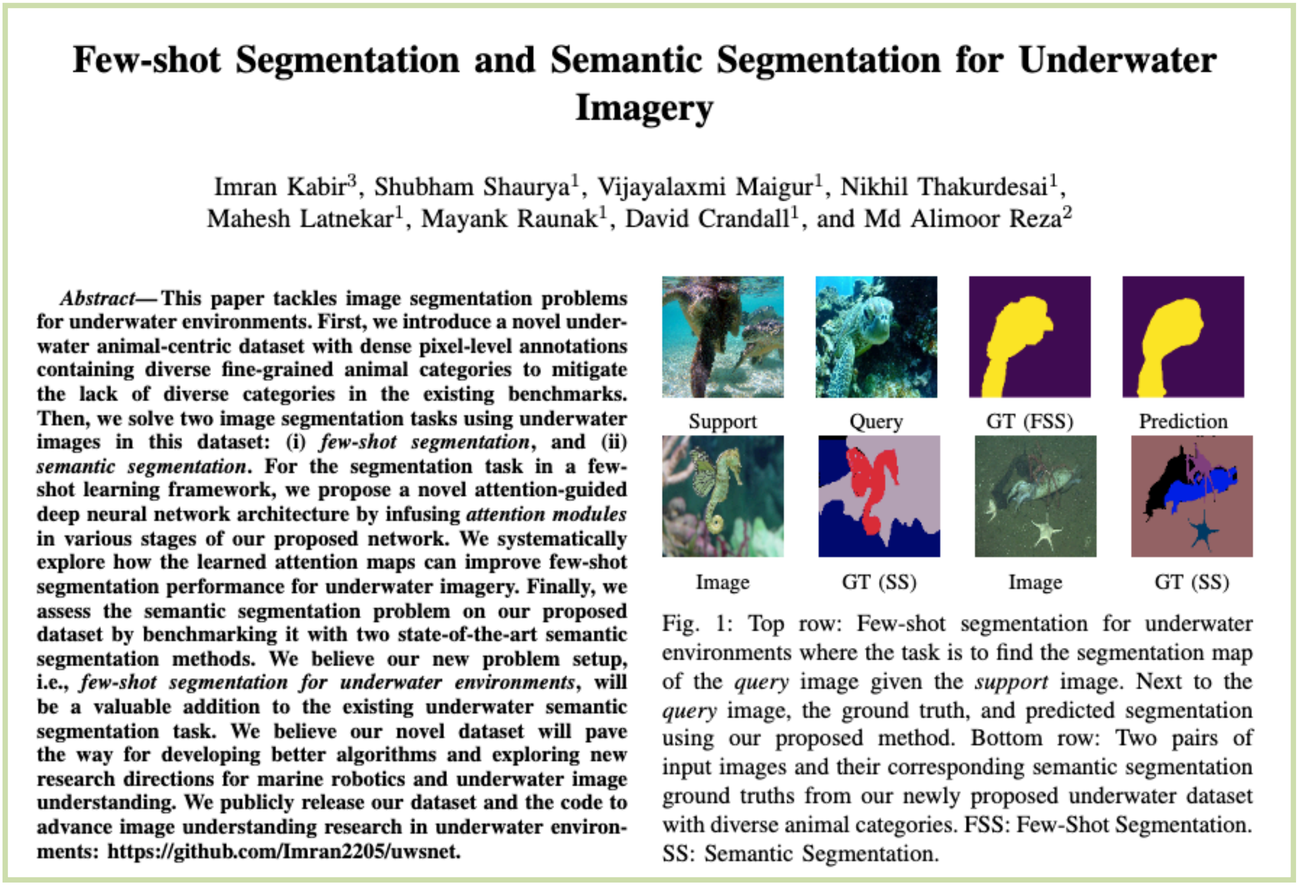
recent work has considered weakly-supervised approaches for part annotation. Fang et al. [3] propose transfer learning to generate pixel-level part annotations for an unlabeled target object instance by using keypoints to propagate part segmentation knowledge from a labeled source object instance of the same class. As annotating keypoint locations is significantly less labor-intensive than generating pixel-wise part masks, this approach can greatly reduce the manual annotation cost. While promising, their work required



[Naha et al. WACV'2021](#)

Animal Body Part Segmentation

What else is CNN good for beyond image recognition?



Few-shot Segmentation and Semantic Segmentation for Underwater Imagery

Imran Kabir[3], Shubham Shaurya[1], Vijayalaxmi Maigur[1], Nikhil Thakurdesai[1], Mahesh Latnekar[1], Mayank Raunak[1], David Crandall[1], and Md Alimoor Reza[2]

***Abstract*— This paper tackles image segmentation problems for underwater environments. First, we introduce a novel underwater animal-centric dataset with dense pixel-level annotations containing diverse fine-grained animal categories to mitigate the lack of diverse categories in the existing benchmarks. Then, we solve two image segmentation tasks using underwater images in this dataset: (i) *few-shot segmentation*, and (ii) *semantic segmentation*. For the segmentation task in a few-shot learning framework, we propose a novel attention-guided deep neural network architecture by infusing *attention modules* in various stages of our proposed network. We systematically explore how the learned attention maps can improve few-shot segmentation performance for underwater imagery. Finally, we assess the semantic segmentation problem on our proposed dataset by benchmarking it with two state-of-the-art semantic segmentation methods. We believe our new problem setup, i.e., *few-shot segmentation for underwater environments*, will be a valuable addition to the existing underwater semantic segmentation task. We believe our novel dataset will pave the way for developing better algorithms and exploring new research directions for marine robotics and underwater image understanding. We publicly release our dataset and the code to advance image understanding research in underwater environments: https://github.com/Imran2205/uwsnet.**

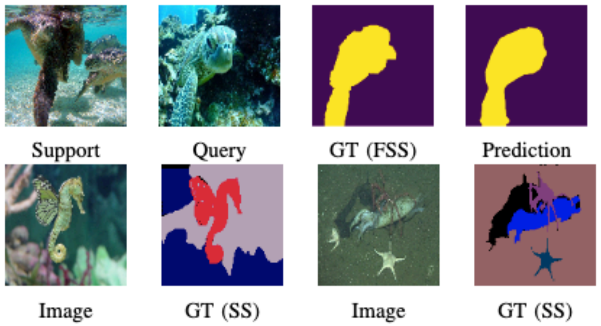


Fig. 1: Top row: Few-shot segmentation for underwater environments where the task is to find the segmentation map of the *query* image given the *support* image. Next to the *query* image, the ground truth, and predicted segmentation using our proposed method. Bottom row: Two pairs of input images and their corresponding semantic segmentation ground truths from our newly proposed underwater dataset with diverse animal categories. FSS: Few-Shot Segmentation. SS: Semantic Segmentation.

Kabir et al. IROS'2023

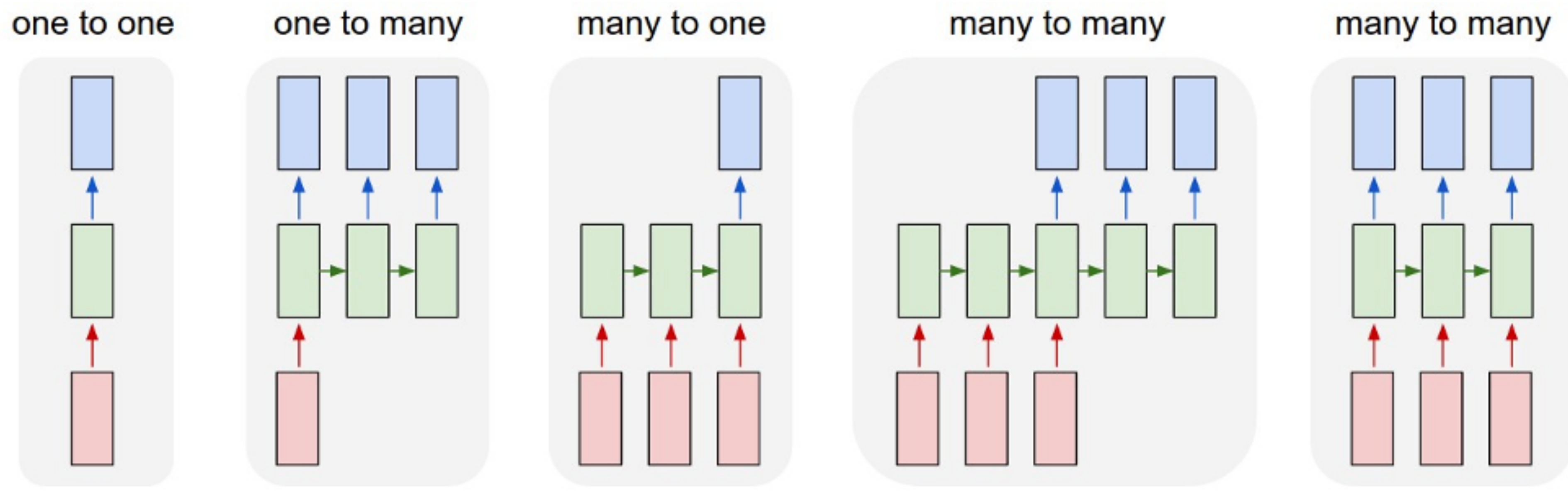
Underwater Animal Segmentation

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**

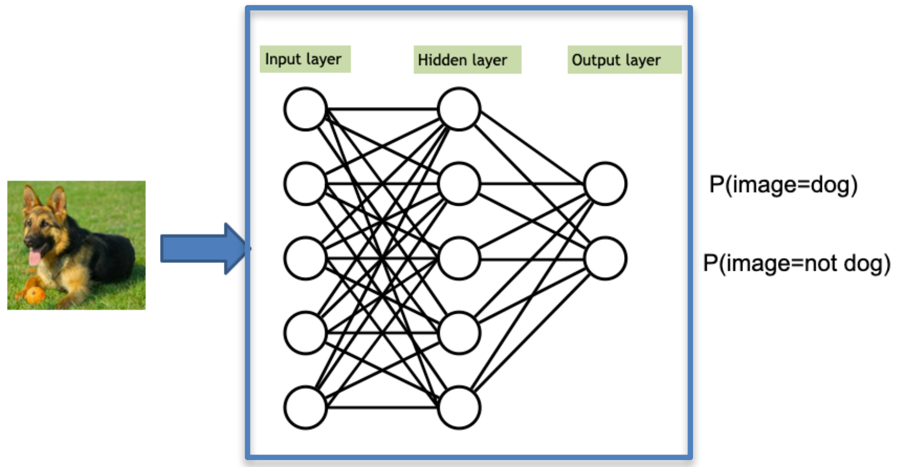
- Can you think of other kinds of target variables we might be interested in modeling?

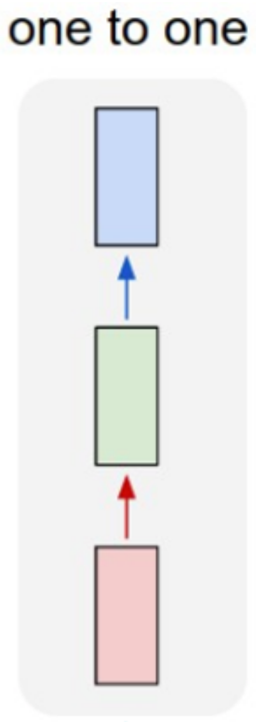
What if my input isn't the same size of my output?



One to one mapping task

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

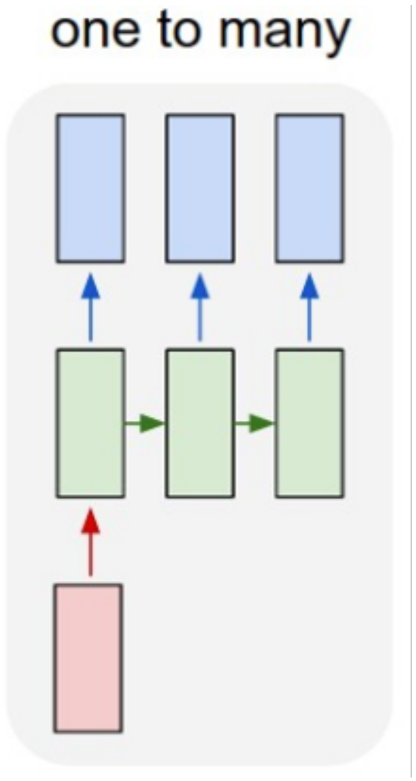




One to many mapping task

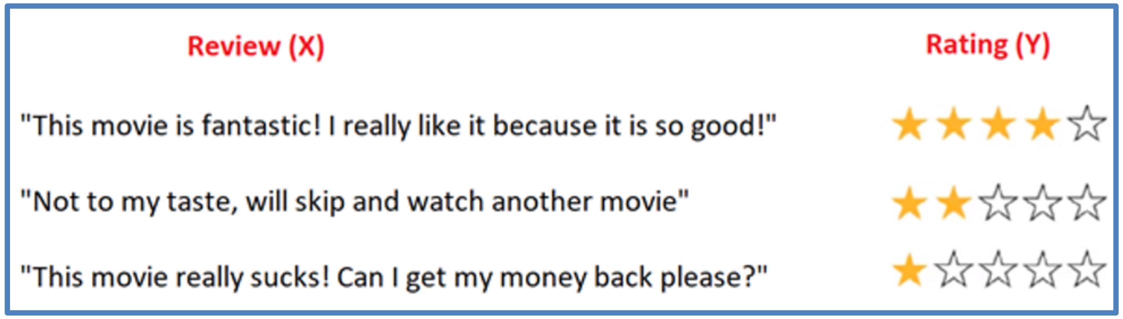
- Can anyone think of an example of a ML model that is **one to many**?
    - we input one training/test example, and output many predictions

"A Dog catching a ball in mid air"

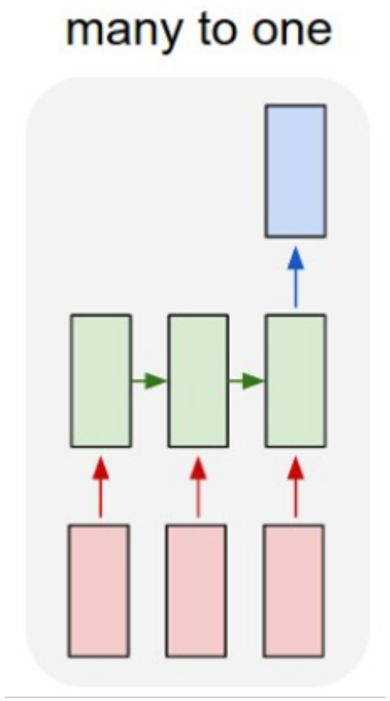


Many to one mapping task

- 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

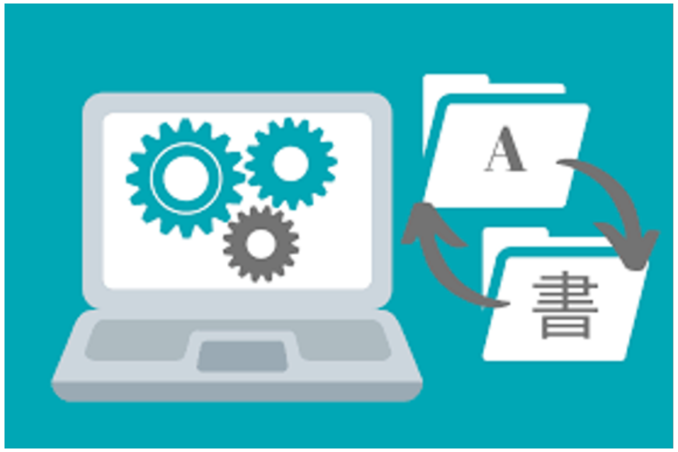


Review (X)	Rating (Y)
"This movie is fantastic! I really like it because it is so good!"	★★★★☆
"Not to my taste, will skip and watch another movie"	★★☆☆☆
"This movie really sucks! Can I get my money back please?"	★☆☆☆☆



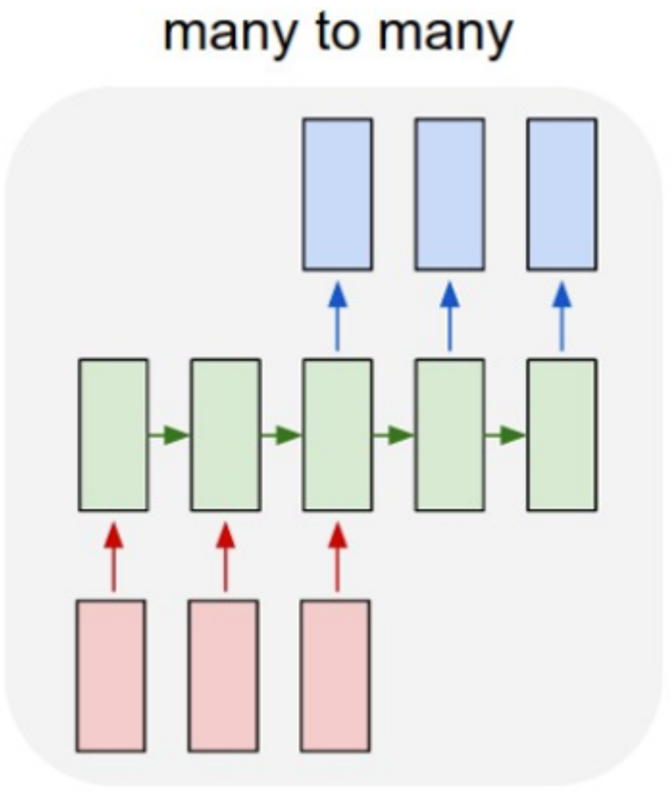
Many to many mapping task

- 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"



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)

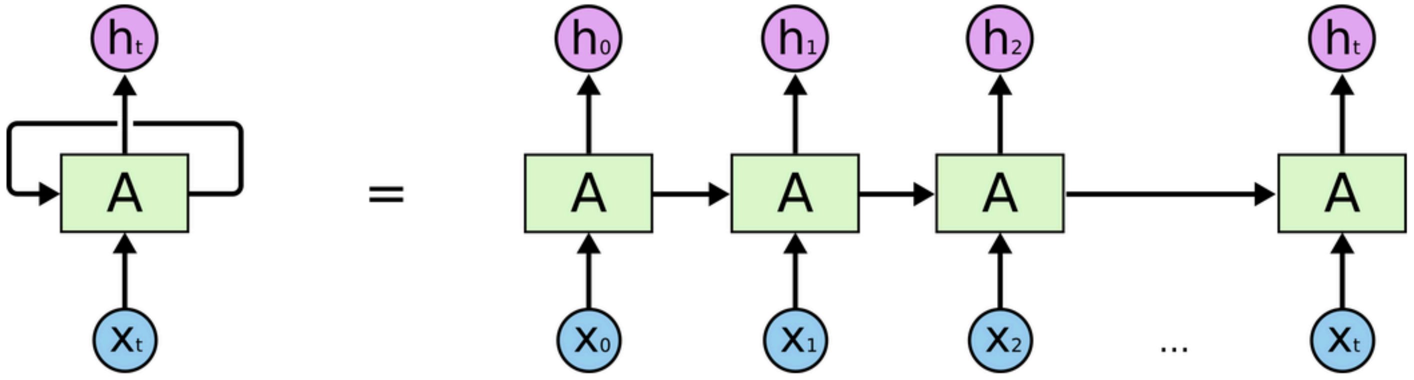
Recurrent Neural Network (RNN)

- 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



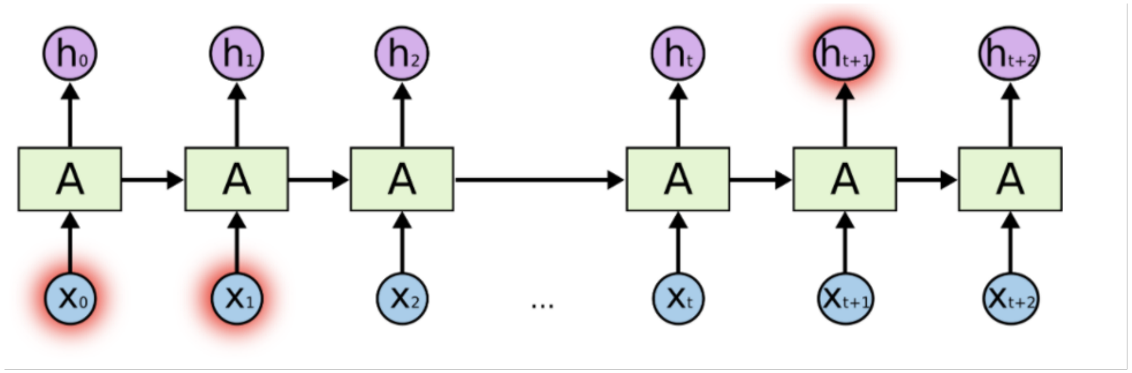
RNN model (rolled-up)

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 ________"`

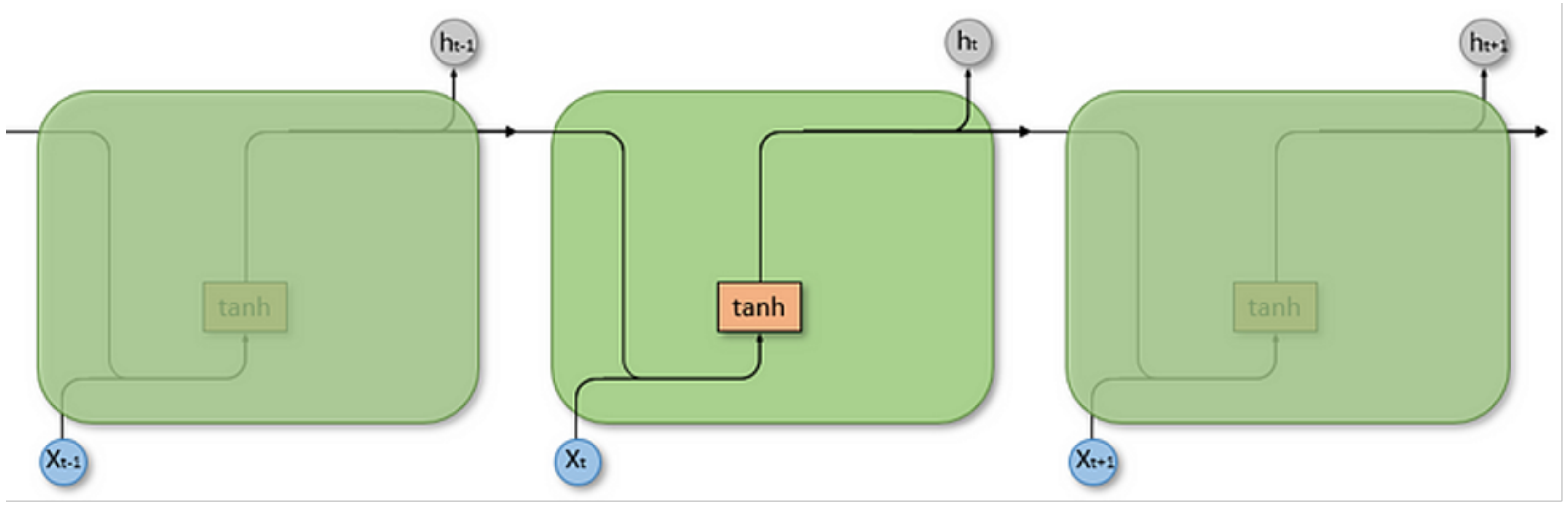
Figure credit: colah.gihub.oi

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



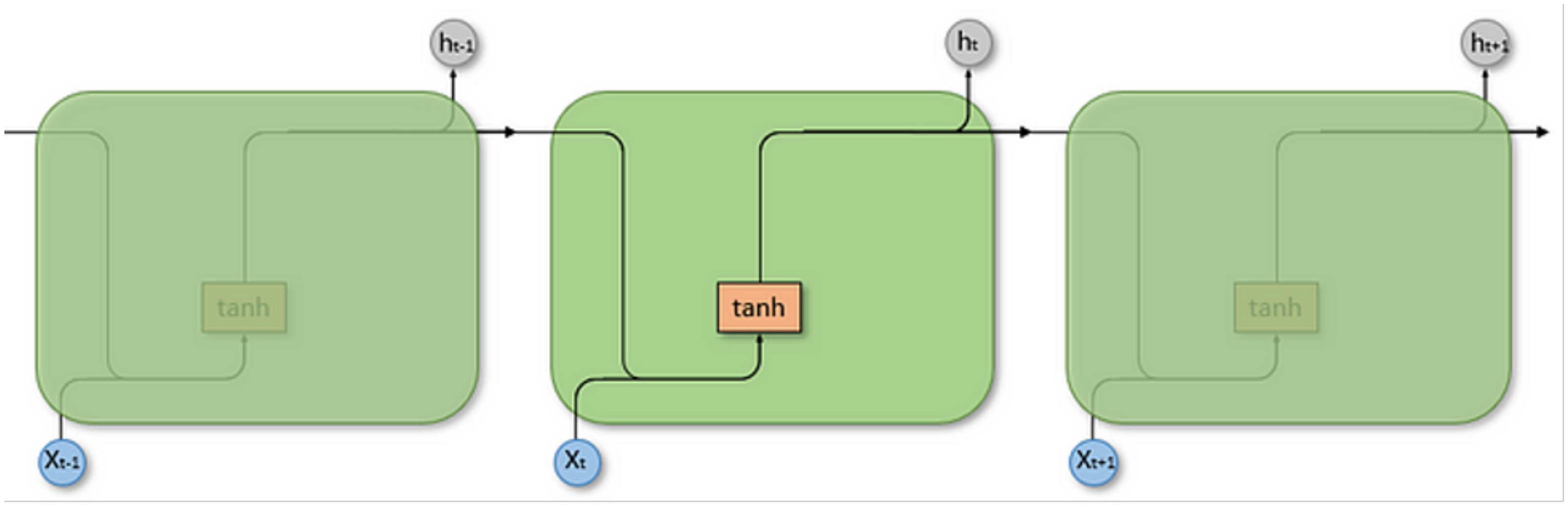
- This figure represents an unfolded RNN to help understand what is going on. The length of the cells is equal to the number of time steps of the input sequence.

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)

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

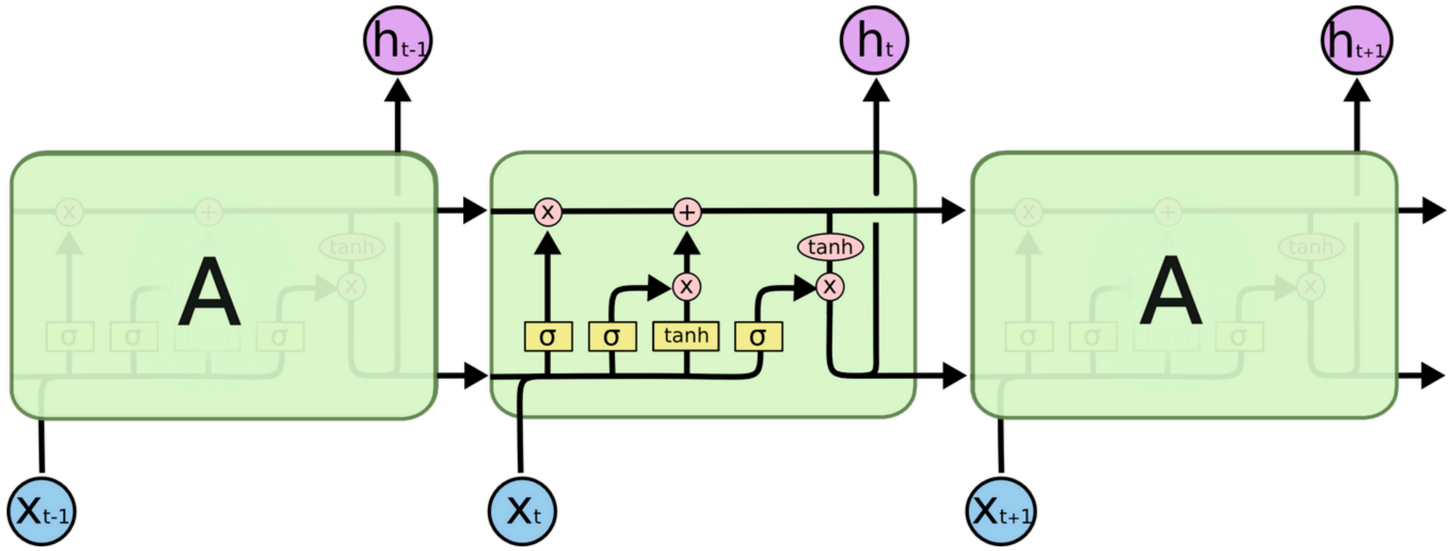
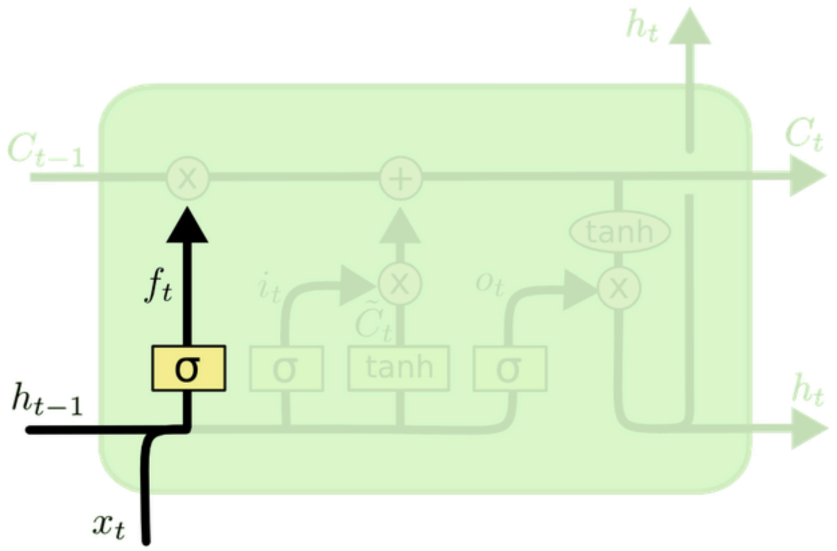


Figure credit: colah.gihub.oi

Long Short-Term Memory (LSTM)

- **Forget gate:** decides how much memory shall be kept



forget gate

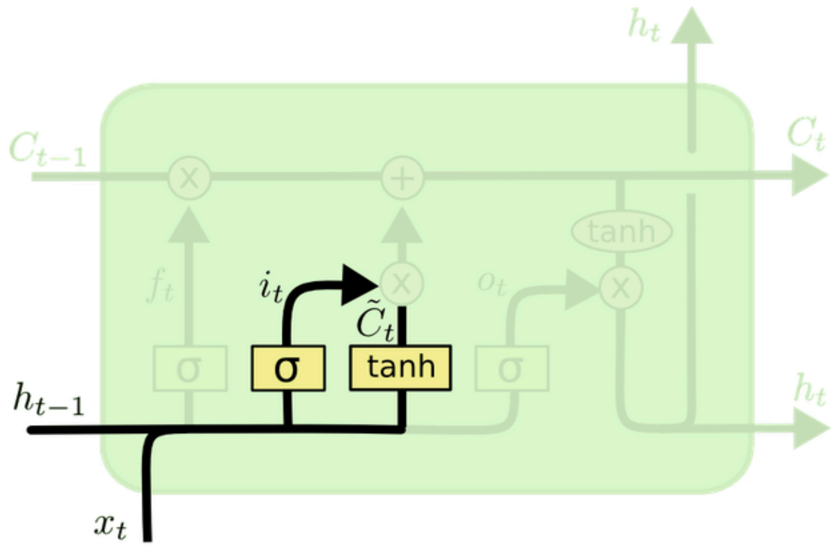
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Learnable parameters

Figure credit: colah.gihub.oi

Long Short-Term Memory (LSTM)

- **Input gate** which decides which information shall be added to the long-term memory



Input gate

Learnable parameters

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

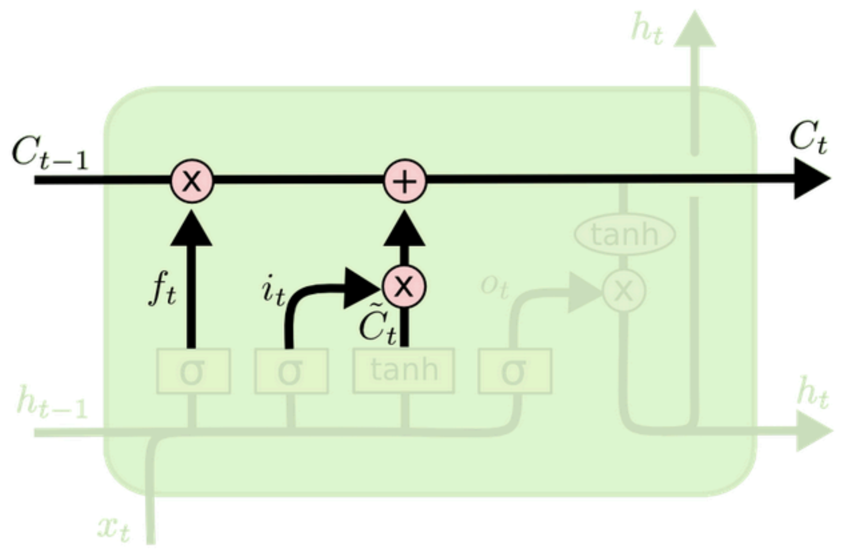
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Learnable parameters

Figure credit: colah.gihub.oi

Long Short-Term Memory (LSTM)

- combining old cell state using input gate + new information using forget gate



forget gate

Input gate

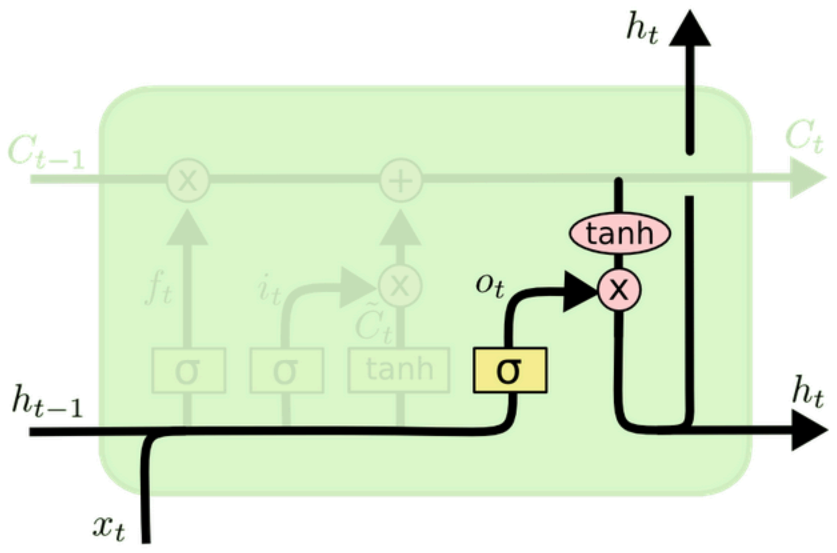
$$\underbrace{C_t}_{\text{New cell state}} = \overset{\text{forget gate}}{f_t} * C_{t-1} + \overset{\text{Input gate}}{i_t} * \tilde{C}_t$$

New cell state

Figure credit: colah.gihub.oi

Long Short-Term Memory (LSTM)

- Finally computing **output gate** (that decides which parts of the cell state build the output, i.e. it's responsible for the short term memory) and hidden state update:



Output gate

Learnable parameters

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

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

Output gate

New cell state

Figure credit: colah.gihub.oi

Long Short-Term Memory (LSTM)

LSTM Advantages

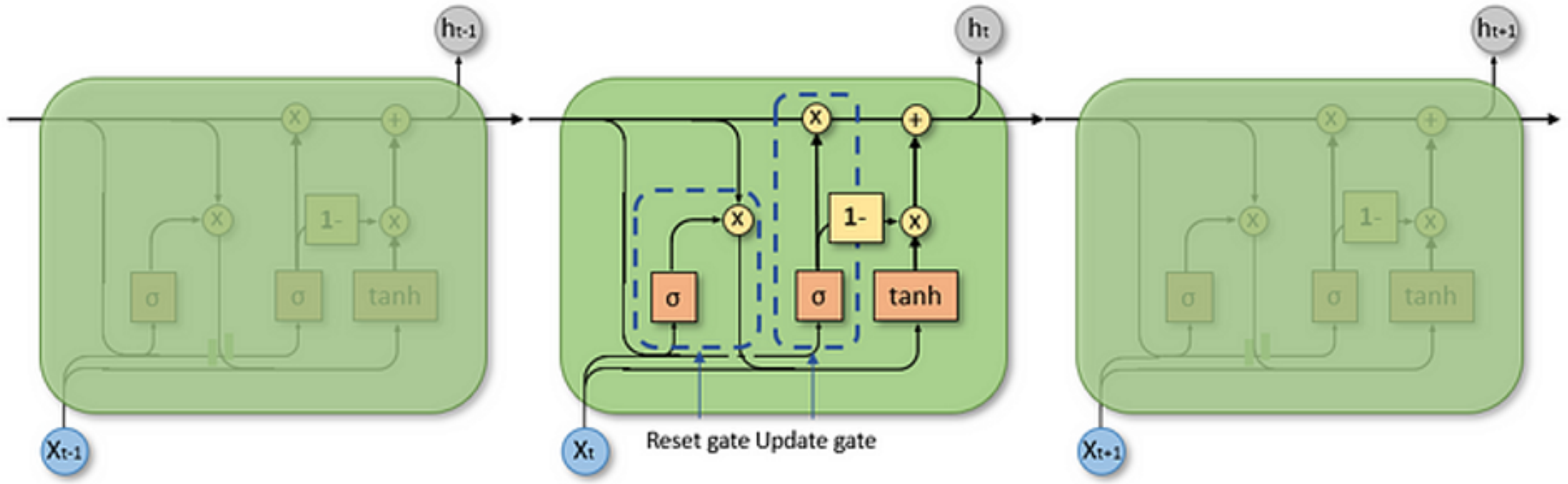
- Can capture both the short and long term patterns of a sequence.

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

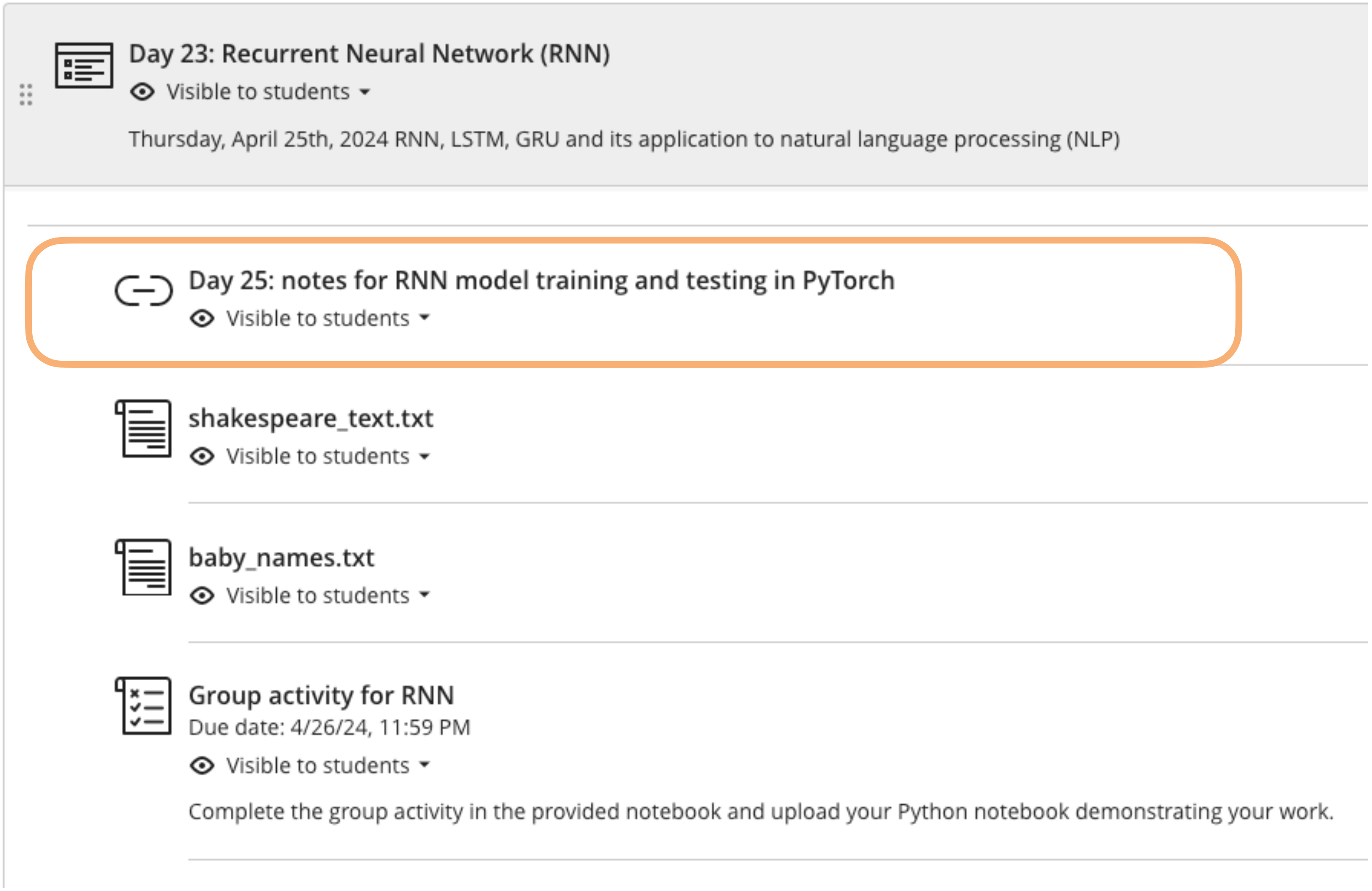
Today's Agenda

- Recurrent neural network (RNN)

- RNN implementation in PyTorch

RNN Implementation in PyTorch

- Go to Blackboard and follow the notebook as shown below:



Day 23: Recurrent Neural Network (RNN)

Visible to students

Thursday, April 25th, 2024 RNN, LSTM, GRU and its application to natural language processing (NLP)

Day 25: notes for RNN model training and testing in PyTorch

Visible to students

shakespeare_text.txt

Visible to students

baby_names.txt

Visible to students

Group activity for RNN

Due date: 4/26/24, 11:59 PM

Visible to students

Complete the group activity in the provided notebook and upload your Python notebook demonstrating your work.