Introduction to Deep Learning (I2DL)

Exercise 11: Sentiment Analysis with RNNs

Today's Outline

- Online Exam: February 10, 2021 (register by 15.1)
- Exercise 10: Segmentation
  Case Study of Student's Solutions
- RNNs and LSTMs
- Exercise 11: Sentiment Analysis with RNNs

Exam

- **Due to current university restrictions: Online exam**
  - No retake exam
  - Bonus will be transferred to any future version of this class
- **Organization:** We will release more detailed information on Piazza as soon as possible
- **Exam relevant content:**
  - Lectures
  - Exercises including optional notebooks

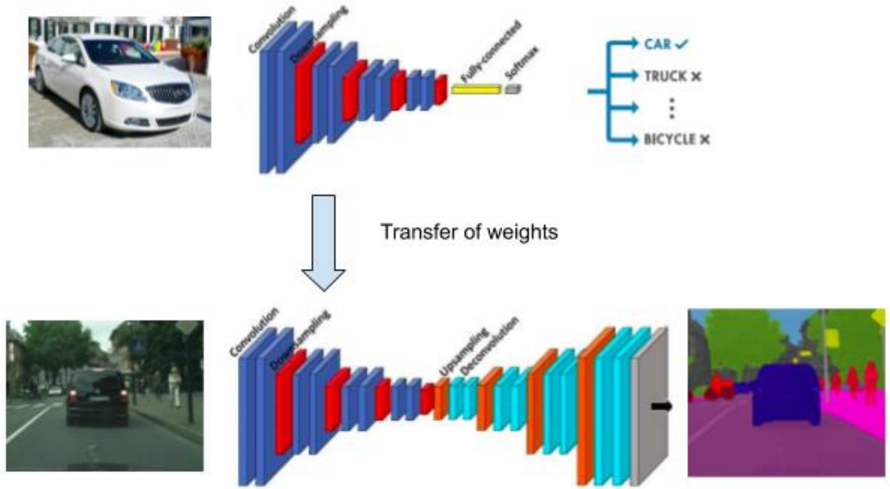
Case Study:
Exercise 10 Semantic Segmentation

Exercise 10: Semantic Segmentation

- **Goal:** Assign a label to each pixel of the image
- **Output of the network:** Segmentation mask with same shape as input image
- **Dataset:** MSRC v2 dataset, 23 object classes, contains 591 images with accurate pixel-wise labeled images

Transfer Learning for Segmentation

- **Idea:** Encoder-Decoder Architecture
- **Transfer Learning:** CNNs trained for image classification contain meaningful information that can be used for segmentation -> Encoder
- **Check out:** pre-trained networks like AlexNet, MobileNets



Leaderboard Submission #10

WS 20/21

Rank	User	Score	Pass
#1	w1459	91.49	✔
#2	w1691	90.80	✔
#3	w1492	90.65	✔
#4	w1258	89.12	✔
#5	w1631	89.06	✔
#6	w1594	88.88	✔
#7	w1463	87.93	✔
#8	w1284	87.46	✔
#9	w1679	87.19	✔
#10	w1468	86.71	✔

SS 21

#	User	Score
1	u0586	92.06
2	u1133	91.72
3	u1055	91.38
4	u0330	91.35
5	u0642	90.87
6	u0623	90.62
7	u0916	89.67
8	u0263	89.31
9	u0255	89.10
10	u0048	89.09

23/06, 20:00

Top 1 Submission: 92,06%

```
self.model_ft = torchvision.models.mobilenet_v2(pretrained=True).features
num_ftrs = 1280  # 1280 of 8x8 features

self.decoder = nn.Sequential(
    nn.Upsample(scale_factor=2, mode='bicubic', align_corners=True),
    nn.ConvTranspose2d(1280, 192, kernel_size=2, stride=2),
    nn.Conv2d(192, 192, kernel_size=3),
    nn.BatchNorm2d(192),
    nn.ReLU(),

    nn.Upsample(scale_factor=2, mode='bicubic', align_corners=True),
    nn.ConvTranspose2d(192, 80, kernel_size=2, stride=2),
    nn.Conv2d(80, 80, kernel_size=3, padding=1),
    nn.BatchNorm2d(80),
    nn.ReLU(),

    nn.Upsample(scale_factor=2, mode='bicubic', align_corners=True),
    nn.Conv2d(80, num_classes, kernel_size=1),
)
```

Summary:

- Architecture: Encoder-Decoder
- Encoder: Pre-trained Mobilenet
- Decoder: Several Blocks consisting of
  - Upsamling Layers
  - Transposed Conv
  - BatchNorm
  - ReLu

Top 2 Submission: 91, 72%

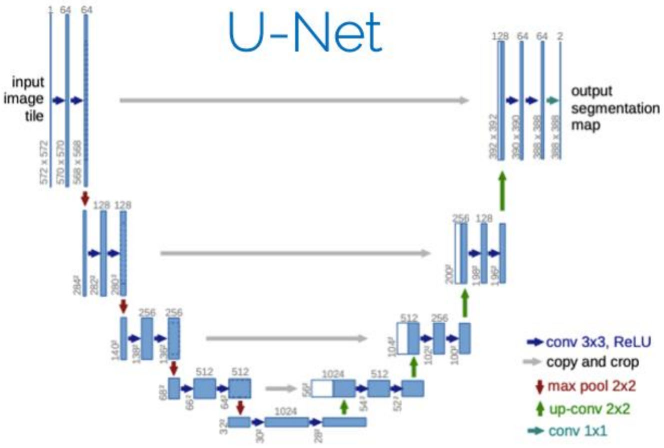
```python
self.encoder = models.mobilenet.mobilenet_v2(pretrained=True).features

self.inner = nn.Sequential(
    nn.Conv2d(1280, l1_depth, 1, 1, 0),
    nn.BatchNorm2d(l1_depth),
    nn.ReLU(inplace=True),
    nn.Conv2d(l1_depth, l1_depth, 3, 1, 1),
    nn.BatchNorm2d(l1_depth),
    nn.ReLU(inplace=True)
)


self.dec_upsample_layers = nn.Sequential(
    nn.ConvTranspose2d(l1_depth, l2_depth, 3, 2, 1),#1-2
    nn.ConvTranspose2d(l2_depth, l3_depth, 3, 2, 1),#2-3
    nn.ConvTranspose2d(l3_depth, l4_depth, 3, 2, 1),#3-4
    nn.ConvTranspose2d(l4_depth, l5_depth, 3, 2, 1),#4-5
    nn.ConvTranspose2d(l5_depth, l6_depth, 3, 2, 1),#5-6
)
```

```python
self.dec_layers = nn.Sequential(
    nn.Sequential(#l2
        nn.ReLU(),
        #nn.Dropout2d(),
        nn.Conv2d(l2_depth+96, l2_depth, 3, 1, 1),#input: 100x15x15
        nn.BatchNorm2d(l2_depth),
        nn.ReLU(),
        #nn.Dropout2d(),

        nn.Conv2d(l2_depth, l2_depth, 3, 1, 1),#input: 100x15x15
        nn.BatchNorm2d(l2_depth),
        nn.ReLU(),
        #nn.Dropout2d(),
    ),


    nn.Sequential(#l3
        nn.ReLU(),
        #nn.Dropout2d(),
        nn.Conv2d(l3_depth+32, l3_depth, 3, 1, 1),#input: 80x30x30
        nn.BatchNorm2d(l3_depth),
        nn.ReLU(),
        #nn.Dropout2d(),

        nn.Conv2d(l3_depth, l3_depth, 3, 1, 1),#input: 80x30x30
        nn.BatchNorm2d(l3_depth),
        nn.ReLU(),
        #nn.Dropout2d(),
    ),
```

...

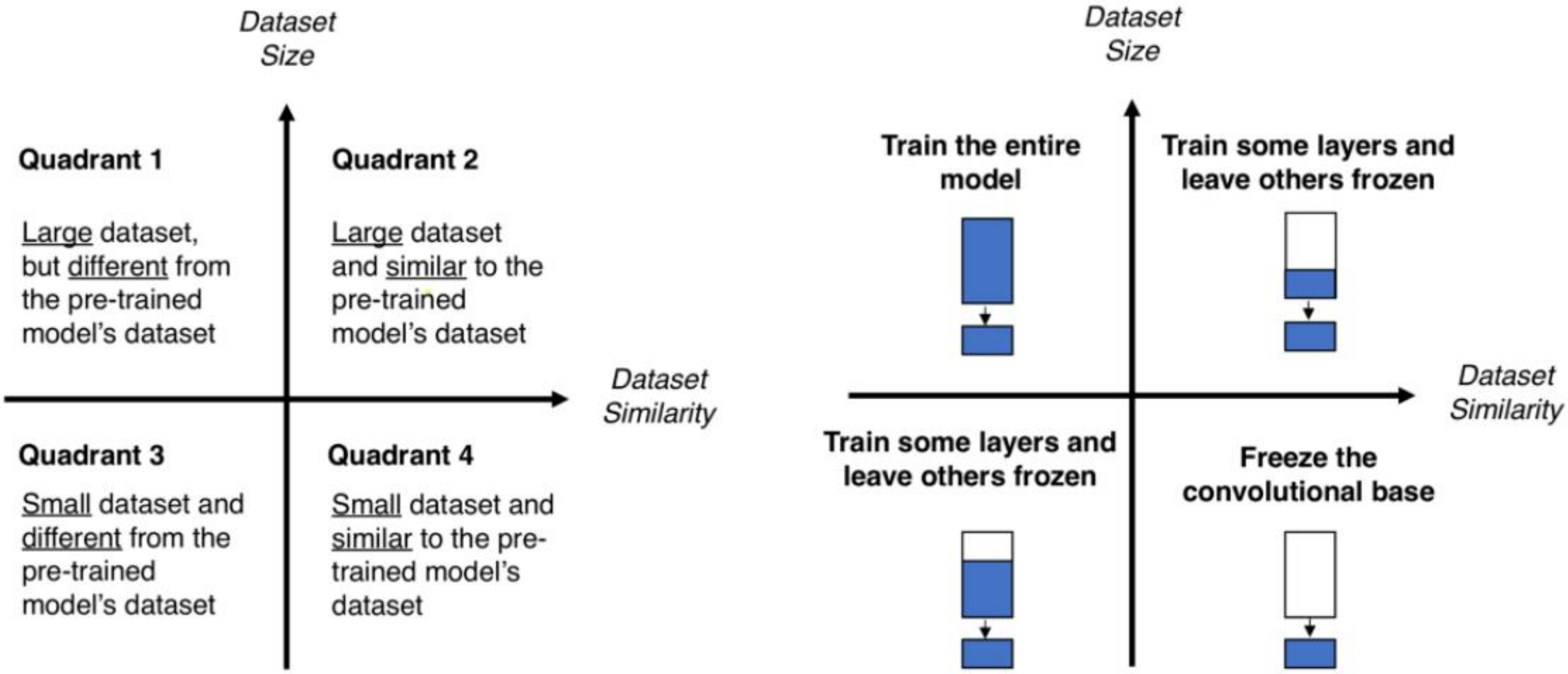
Top 2 Submission: 91, 72%

```python
def forward(self, x):
    ########################################################################
    #                             YOUR CODE                                #
    ########################################################################

    enc_layer_res = []

    res = x
    for i, layer in enumerate(self.encoder):

        new_res = layer(res)

        if res.shape[-2:] != new_res.shape[-2:]:
            enc_layer_res.append(new_res)
        else:
            enc_layer_res[-1] = new_res

        res = new_res


    res = self.inner(res)

    for i, layer in enumerate(self.dec_upsample_layers):
        if len(enc_layer_res)-1 == i:
            output_size = (240, 240)
        else:
            output_size = enc_layer_res[-i-2].shape[-2:]

        res = layer(res, output_size=output_size)

        if len(enc_layer_res)-1 != i:
            res = torch.cat([res, enc_layer_res[-i-2]], dim=1)

        res = self.dec_layers[i](res)

    x = res

    ########################################################################
    #                          END OF YOUR CODE                            #
    ########################################################################
```



Summary:
- Architecture: UNet
- Forward pass:
    - Concatenate encoder features in decoder

When/What to Fine-tune.



https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

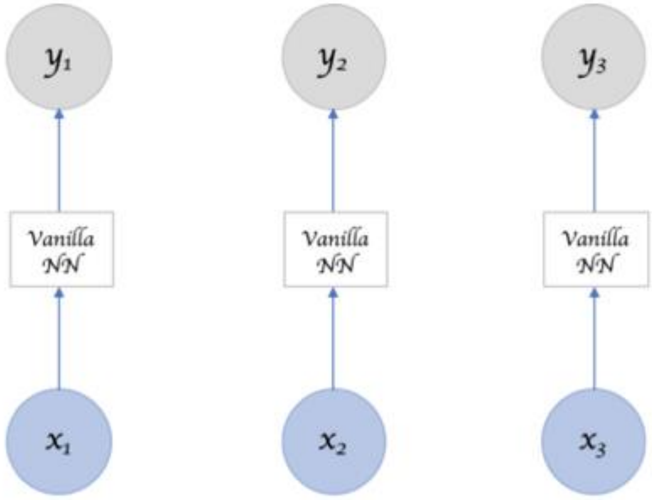
Recurrent Neural Networks

Vanilla Feed-Forward Neural Networks

- **Input:** Fixed size vector as input, e.g. images
- **New type of input:** Series of input with no predetermined limit on size
- **Examples:** Series of images = Video, Series of words = Text

IDEA: Use vanilla network repeatedly for a series of input?

PROBLEM: Series of inputs mean that the components are related to each other



https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Recurrent Neural Networks

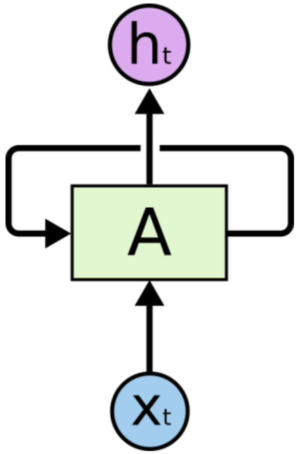
- **Idea:** Network that can capture the relationship between the inputs
- **RNNs:** Learning process is not independent
    - Remember things from processing trainings data
    - Remember things learnt from prior inputs, prior inputs influences decision
- **In other words:** RNNs produce different outputs for same input depending on previous outputs in the series.

$$\boldsymbol{A_t = \theta_c A_{t-1} + \theta_x x_t}$$

Previous hidden state

input

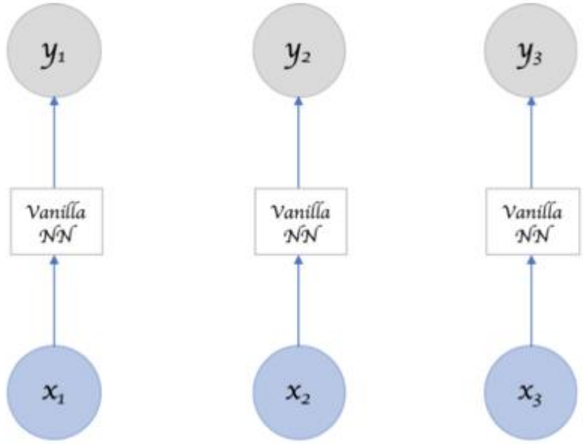
Output / Hidden State

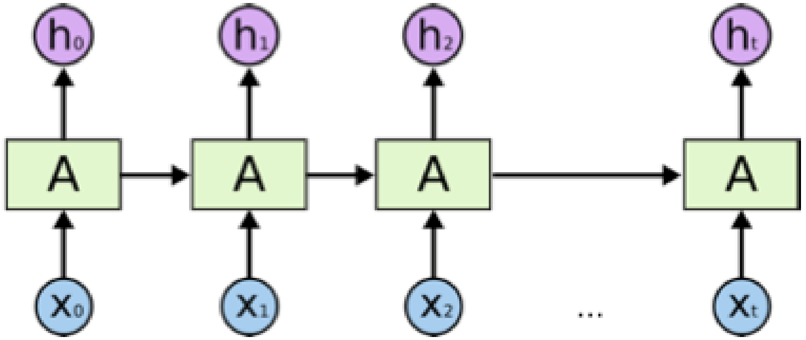


Input

RNNs: Parameter sharing

- **Parameter Sharing:** Weights are shared across all inputs
- **Advantage:** Using same weights allows to process series of input without predefined length

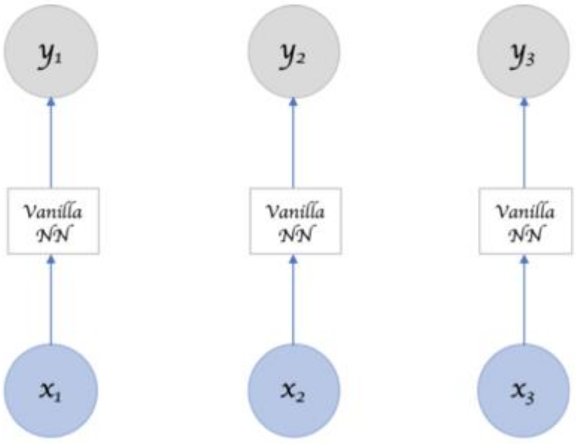


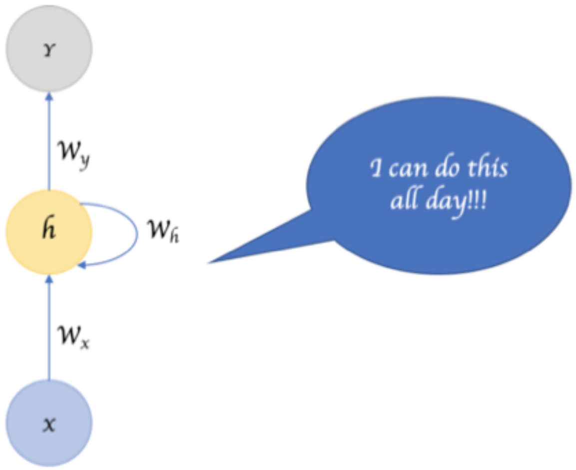


https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

RNNs: Parameter sharing

- **Parameter Sharing:** Weights are shared across all inputs
- **Advantage:** Using same weights allows to process series of input without predefined length

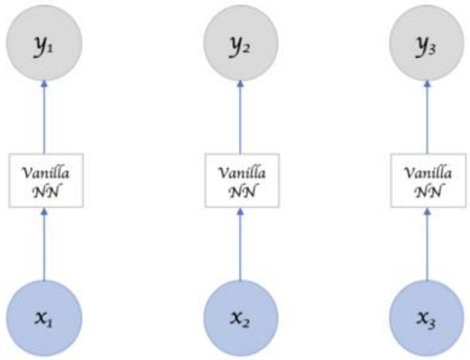


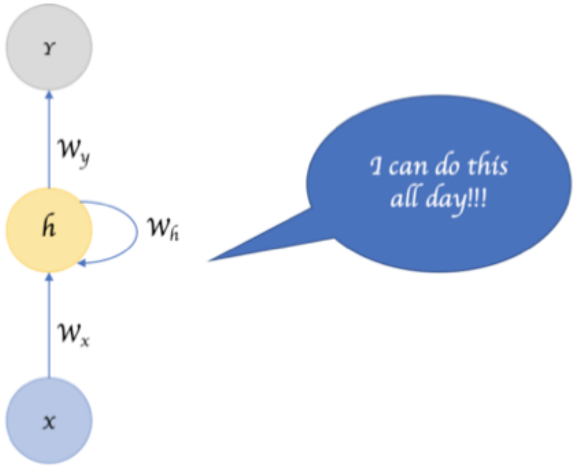


https://towardsdatascience.com/recurrent-neural-networks-

RNNs: Parameter sharing

- **Parameter Sharing:** Weights are shared across all inputs
- **Advantage:** Using same weights allows to process series of input without predefined length
- **Hidden state:** Ensures that each input undergoes a different procedure even with shared weights





RNN Concepts

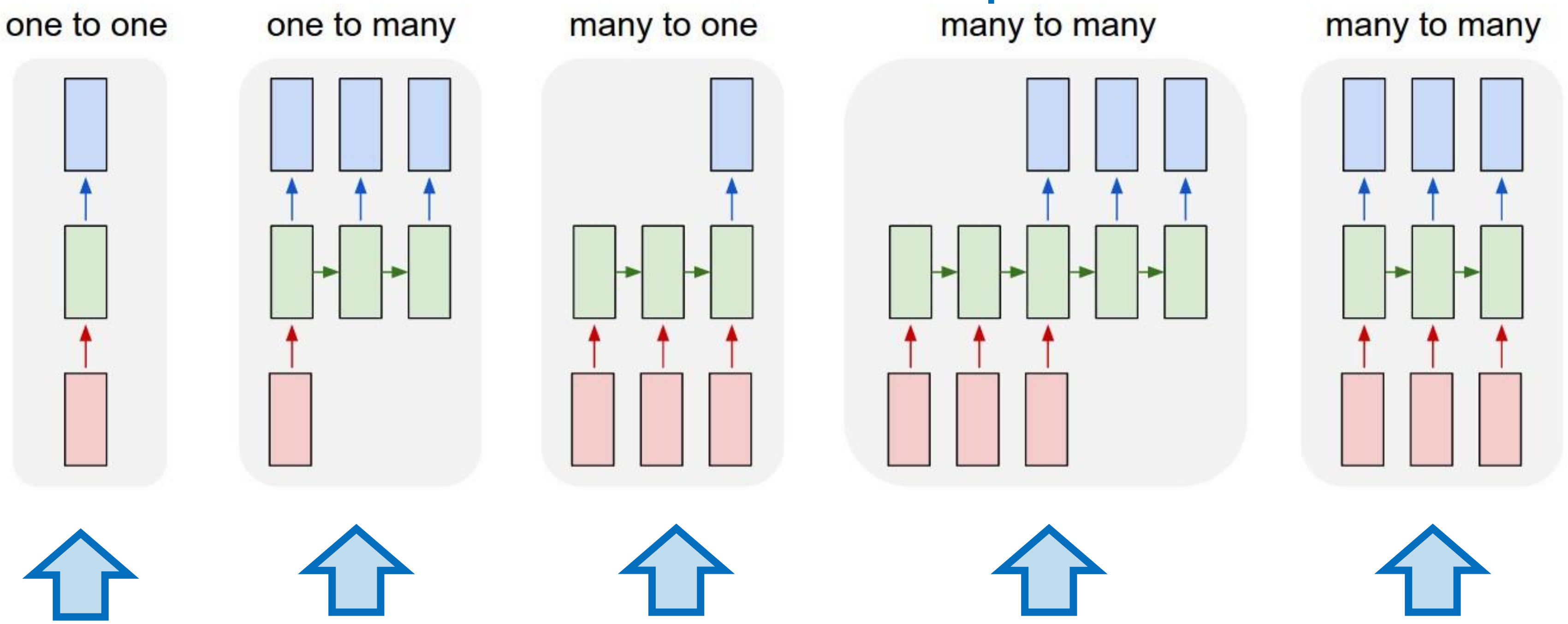


Image Classification

Image Captioning (image -> seq of words)

Sentiment Analysis (seq of words -> sentiment)

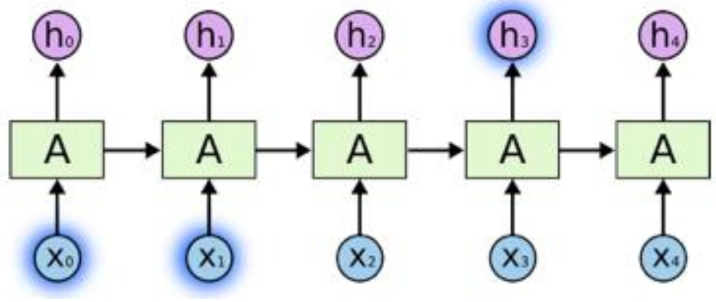
Machine Translation (seq of words -> seq of words)

Video Classification on frame level (seq of frames -> seq of class.)

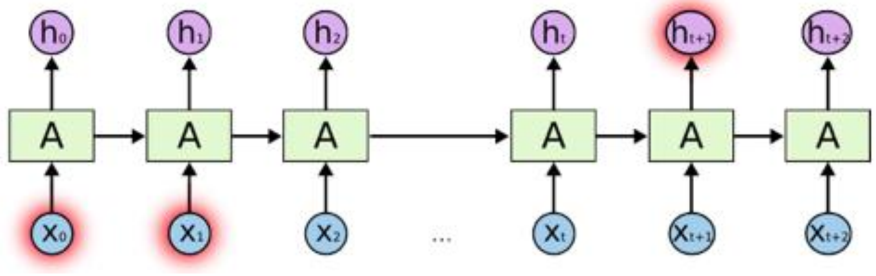
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs and LSTMs

- **Disadvantage:** RNNs struggle to learn long-term dependencies
- **LSTMs (Long Short Term Memory Networks):** special kind of RNNs capable of learning long-term dependencies



The clouds are in the sky.



I moved to Germany … so I speak German fluently.

https://towardsdatascience.com/recurrent-neural-networks-

Exercise 11: Sentiment Analysis

Exercise 11: Goal

Review: I wouldn't rent this one even on dollar rental night.

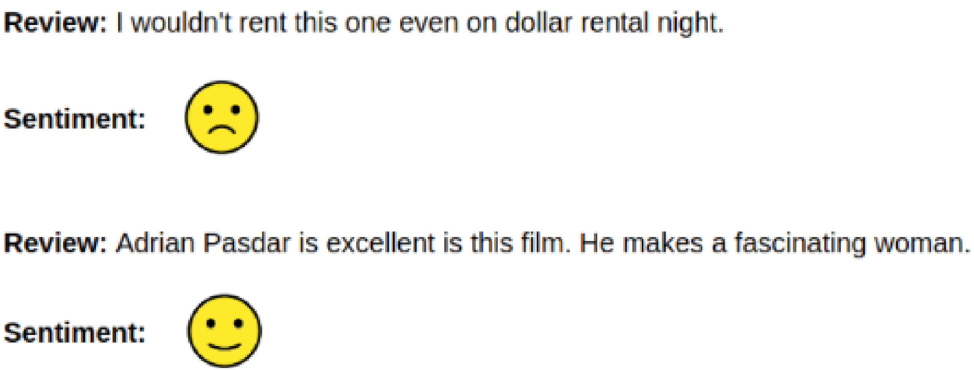
Sentiment: 

Review: Adrian Pasdar is excellent is this film. He makes a fascinating woman.

Sentiment: 

Exercise 11: Content

- Optional Notebook: RNNs and LSTMs
- Notebook 1: Text Preprocessing and Embedding
- Notebook 2: Sentiment Analysis

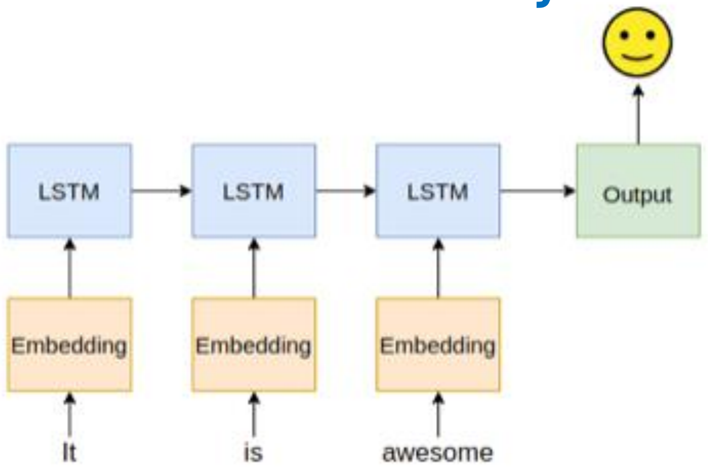


Notebook 1: Text Preprocessing and Embedding

- **Sequential Data:** from image data to text data
- **Dataset:** IMDb sentiment analysis dataset
- **Goal of the notebook:**
  - Data preparation
  - Implementation of Embedding layer

Notebook 2: Sentiment Analysis

- **Network Architecture:**
  - Embedding layer
  - RNN
  - Output layer,
    e.g. fully-connected layer
- **Loss:** Cross-Entropy Loss
- **Performance measure:** Accuracy
- **Goal of the notebook:** Implement and train a recurrent neural network for sentiment analysis



Submission Details

- Submission **Start: January 13**, 2021 13.00
- Submission **Deadline : January 19**, 2021 18.59
- Submit your trained model
- Your model's **accuracy on our test set** is all that counts in order to pass this submission!
  - Threshold to pass: 83%

Summary

- **Monday 17.1**: Watch Lecture 12
  - Advanced DL Topics
- **Wednesday 19.1**: Submit exercise
- This is the last tutorial

Good luck with the exercise! ☺