# BST 261: Data Science II
# Lecture 9

## Object Localization and Detection,
## Face Recognition,
## Introduction to RNNs

**Heather Mattie**
**Harvard T.H. Chan School of Public Health**
**Spring 2 2020**

# Recipe of the Day!

[Turkey Pot Pie]

# Paper Presentations

# MULTITASK LEARNING AND BENCHMARKING WITH CLINICAL TIME SERIES DATA

Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan

Sci Data. 2019 Jun 17;6(1):96. doi: 10.1038/s41597-019-0103-9

Yanghui Sheng
4/20/20

- Derived a **benchmark dataset** from the publicly available Medical Information Mart for Intensive Care (MIMIC-III) database.

MIMIC-III:
 - A large, open source database comprising information relating to patients
admitted to critical care units at Beth Israel Deaconess Medical Center (Boston,
MA) between 2001 – 2012 (38,597 adult patients, corresponding to 49,785 hospital admissions).

- The benchmark dataset is a subset of MIMIC-III containing more than 31 million clinical events that correspond to 17 clinical variables, covering 42276 ICU stays of 33798 unique patients.

MIMIC-III, a freely accessible critical care database: Sci Data. 2016 May 24;3:160035. doi: 10.1038/sdata.2016.35.

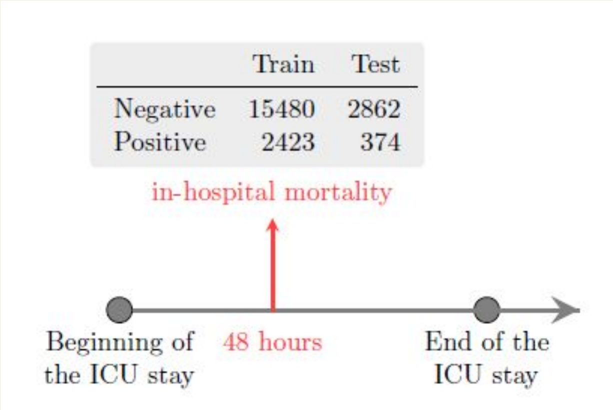
| Variable | MIMIC-III table | Impute value | Modeled as |
|---|---|---|---|
| Capillary refill rate | chartevents | 0.0 | categorical |
| Diastolic blood pressure | chartevents | 59.0 | continuous |
| Fraction inspired oxygen | chartevents | 0.21 | continuous |
| Glascow coma scale eye opening | chartevents | 4 spontaneously | categorical |
| Glascow coma scale motor response | chartevents | 6 obeys commands | categorical |
| Glascow coma scale total | chartevents | 15 | categorical |
| Glascow coma scale verbal response | chartevents | 5 oriented | categorical |
| Glucose | chartevents, labevents | 128.0 | continuous |
| Heart Rate | chartevents | 86 | continuous |
| Height | chartevents | 170.0 | continuous |
| Mean blood pressure | chartevents | 77.0 | continuous |
| Oxygen saturation | chartevents, labevents | 98.0 | continuous |
| Respiratory rate | chartevents | 19 | continuous |
| Systolic blood pressure | chartevents | 118.0 | continuous |
| Temperature | chartevents | 36.6 | continuous |
| Weight | chartevents | 81.0 | continuous |
| pH | chartevents, labevents | 7.4 | continuous |

**Table 3.** The 17 selected clinical variables. The second column shows the source table(s) of a variable from MIMIC-III database. The third column lists the "normal" values we used in our baselines during the imputation step, and the fourth column describes how our LSTM-based baselines treat the variables.

- Derived a **benchmark dataset** from the publicly available Medical Information Mart for Intensive Care (MIMIC-III) database.

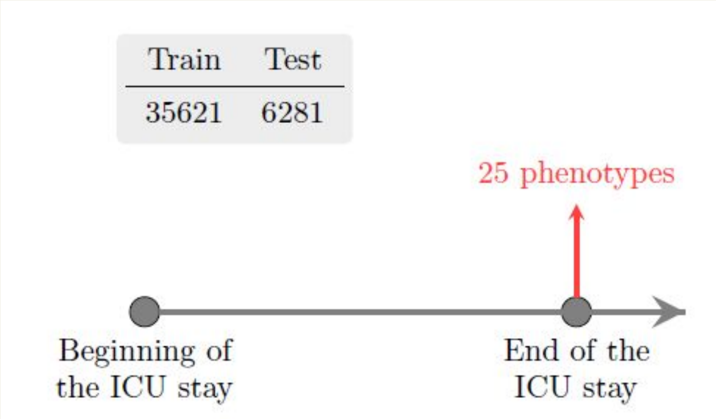
## 4 benchmark tasks:

(a) In-hospital mortality prediction.




- Binary classification: whether patient will die in the hospital.
- Early detection of at-risk patients can improve outcomes.

(b) Phenotype classification.

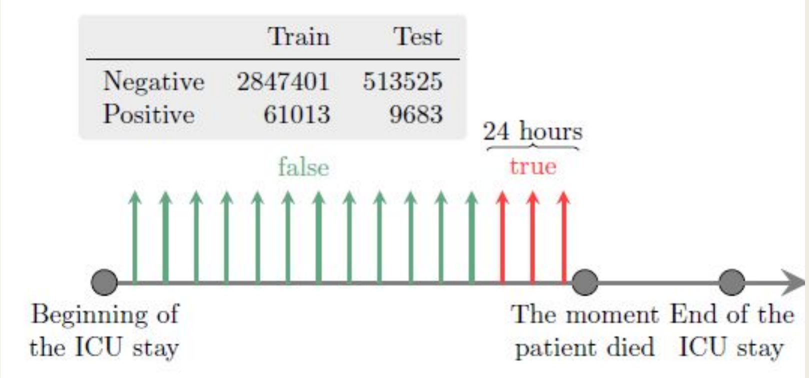



- Multilabel classification: which of the 25 acute care
conditions a patient has.
- Useful for patient treatment and risk monitoring.

- Derived a **benchmark dataset** from the publicly available Medical Information Mart for Intensive Care (MIMIC-III) database.

## <u>4 benchmark tasks:</u>

(c) Decompensation prediction

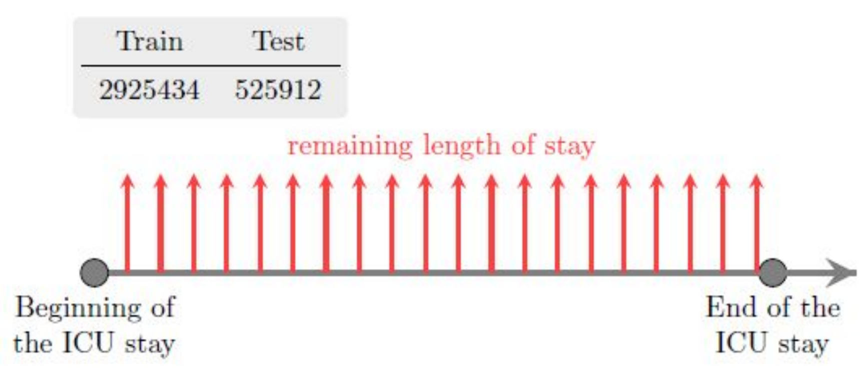


| | Train | Test |
|---|---|---|
| Negative | 2847401 | 513525 |
| Positive | 61013 | 9683 |



- Binary classification: whether patient will die in the next 24 hrs.
- Also for early detection, related to in-hospital mortality.

(d) Length-of-stay prediction.



| Train | Test |
|---|---|
| 2925434 | 525912 |



- Classification: ICU stays < 1 day, each of 7 days,
between 1-2 weeks, > 2 weeks
- Useful for measuring patient acuity and resource

- Compared LSTMs vs. logistic regression using the 4 benchmark tasks on MIMIC III data.

LSTM: long short term memory, a type of recurrent neural network.

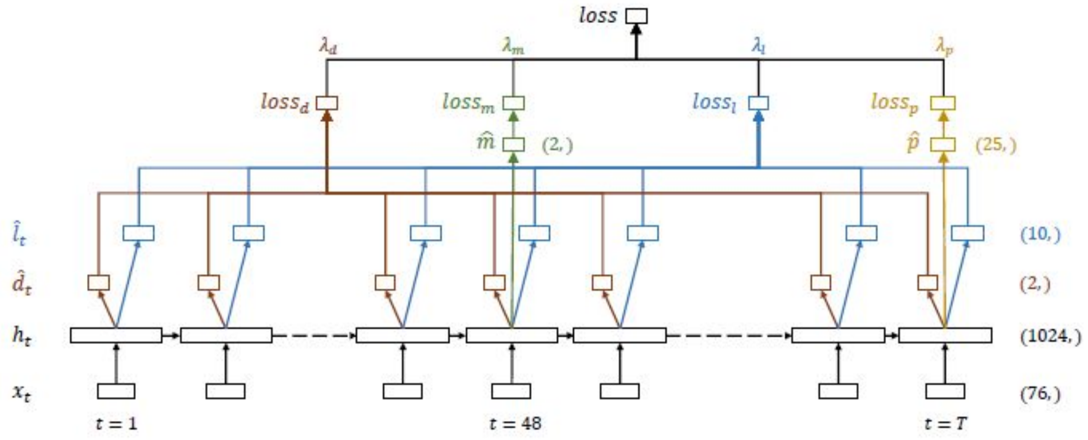
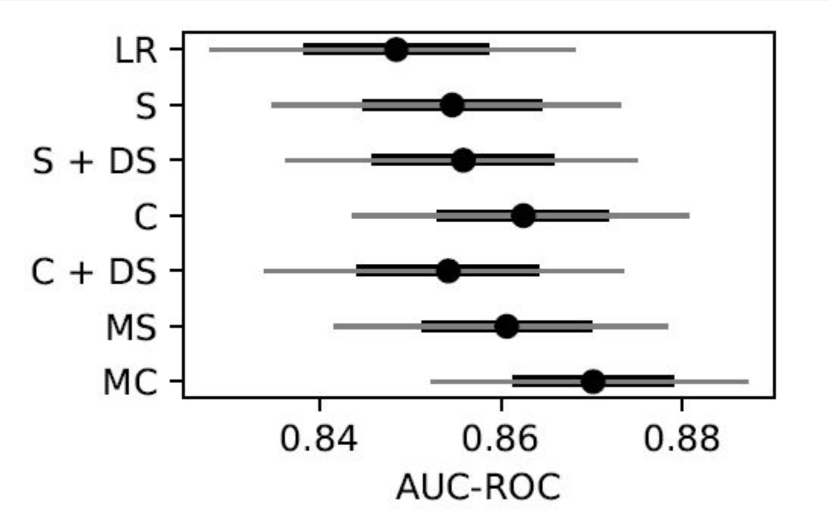


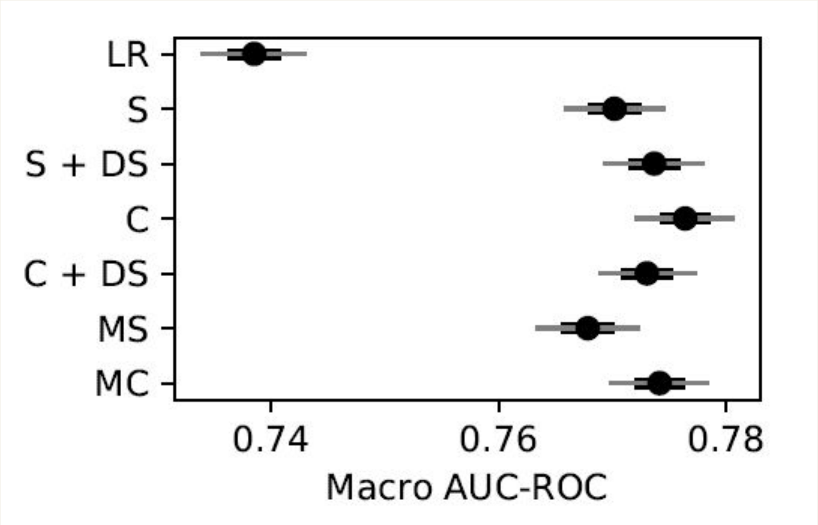

**Figure 10.** LSTM-based network architecture for multitask learning.

- Compared LSTMs vs. logistic regression using the 4 benchmark tasks on MIMIC III data.

(a) In-hospital mortality prediction.




(b) Phenotype classification.




| | | |
|---|---|---|
| LR – logistic regression | C – channel-wise LSTM | MS – multitask standard LSTM |
| S – standard LSTM | DS – deep supervision | MC – multitask channel-wise LSTM |

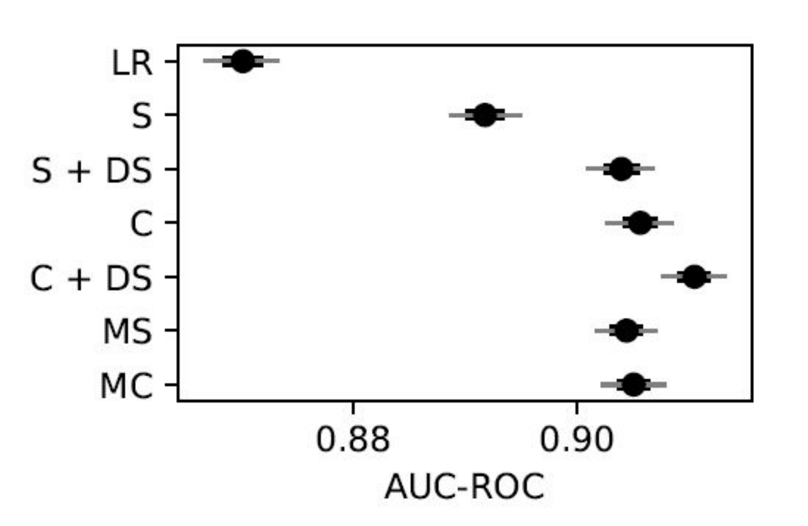
| Phenotype | Type | Prevalence | | AUC-ROC |
|---|---|---|---|---|
| | | Train | Test | |
| Acute and unspecified renal failure | acute | 0.214 | 0.212 | 0.806 |
| Acute cerebrovascular disease | acute | 0.075 | 0.066 | 0.909 |
| Acute myocardial infarction | acute | 0.103 | 0.108 | 0.776 |
| Cardiac dysrhythmias | mixed | 0.321 | 0.323 | 0.687 |
| Chronic kidney disease | chronic | 0.134 | 0.132 | 0.771 |
| Chronic obstructive pulmonary disease | chronic | 0.131 | 0.126 | 0.695 |
| Complications of surgical/medical care | acute | 0.207 | 0.213 | 0.724 |
| Conduction disorders | mixed | 0.072 | 0.071 | 0.737 |
| Congestive heart failure; nonhypertensive | mixed | 0.268 | 0.268 | 0.763 |
| Coronary atherosclerosis and related | chronic | 0.322 | 0.331 | 0.797 |
| Diabetes mellitus with complications | mixed | 0.095 | 0.094 | 0.872 |
| Diabetes mellitus without complication | chronic | 0.193 | 0.192 | 0.797 |
| Disorders of lipid metabolism | chronic | 0.291 | 0.289 | 0.728 |
| Essential hypertension | chronic | 0.419 | 0.423 | 0.683 |
| Fluid and electrolyte disorders | acute | 0.269 | 0.265 | 0.739 |
| Gastrointestinal hemorrhage | acute | 0.072 | 0.079 | 0.751 |
| Hypertension with complications | chronic | 0.133 | 0.130 | 0.750 |
| Other liver diseases | mixed | 0.089 | 0.089 | 0.778 |
| Other lower respiratory disease | acute | 0.051 | 0.057 | 0.694 |
| Other upper respiratory disease | acute | 0.040 | 0.043 | 0.785 |
| Pleurisy; pneumothorax; pulmonary collapse | acute | 0.087 | 0.091 | 0.709 |
| Pneumonia | acute | 0.139 | 0.135 | 0.809 |
| Respiratory failure; insufficiency; arrest | acute | 0.181 | 0.177 | 0.907 |
| Septicemia (except in labor) | acute | 0.143 | 0.139 | 0.854 |
| Shock | acute | 0.078 | 0.082 | 0.892 |
| All acute diseases (macro-averaged) | | | | 0.796 |
| All mixed (macro-averaged) | | | | 0.768 |
| All chronic diseases (macro-averaged) | | | | 0.746 |
| All diseases (macro-averaged) | | | | 0.776 |

25 ICU phenotypes used in the benchmark data set, their prevalence and the per-phenotype classification performance of the best LSTM network

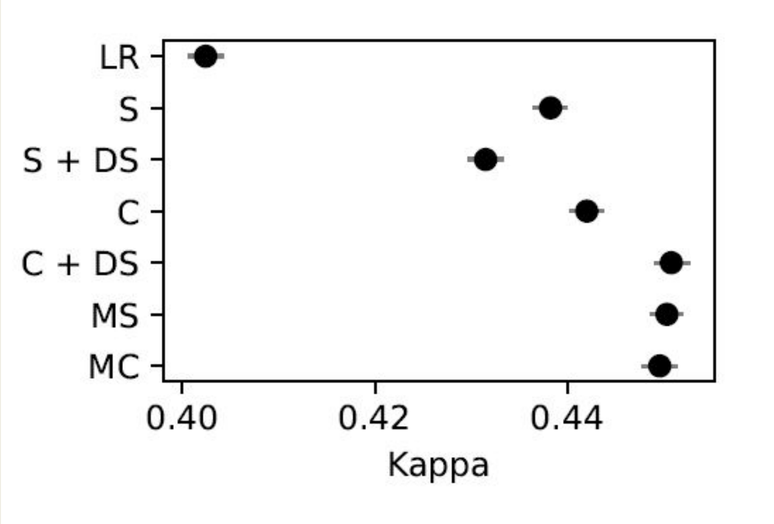
Found better performance for phenotyping acute than chronic conditions.

- Compared LSTM-based neural network and linear models using the 4 benchmark tasks on MIMIC III data.

(c) Decompensation prediction




(d) Length-of-stay prediction.




| | | |
|---|---|---|
| LR – logistic regression | C – channel-wise LSTM | MS – multitask standard LSTM |
| S – standard LSTM | DS – deep supervision | MC – multitask channel-wise LSTM |

# Summary

- Proposed 4 standardized benchmarks for machine learning researchers interested in clinical data problems.
- Described strong linear and neural baselines for these benchmarks.
- LSTM-based models significantly outperform linear models.

# Language Models are Unsupervised Multitask Learners

Carol Wei
April 20, 2020

# GPT-2 Model

## Generative Pre-trained Transformer 2

OpenAI

# INTRODUCTION

- Current models use pre-training and fine tuning
- Goal: language model without supervised fine tuning
  - Transformer-based language model
  - Solve multiple NLP tasks in a zero-shot setting

# TRAINING DATASET

- Web scrape all Reddit outbound links
  - Filter ≥ 3 karma ("interesting, educational, or just funny")
- WebText
  - 45 million links
  - 8 million documents
  - 40 GB of text

# INPUT REPRESENTATION

- Byte Pair Encoding (BPE)
  - Middle ground between character and word level language modeling

*Dictionary*

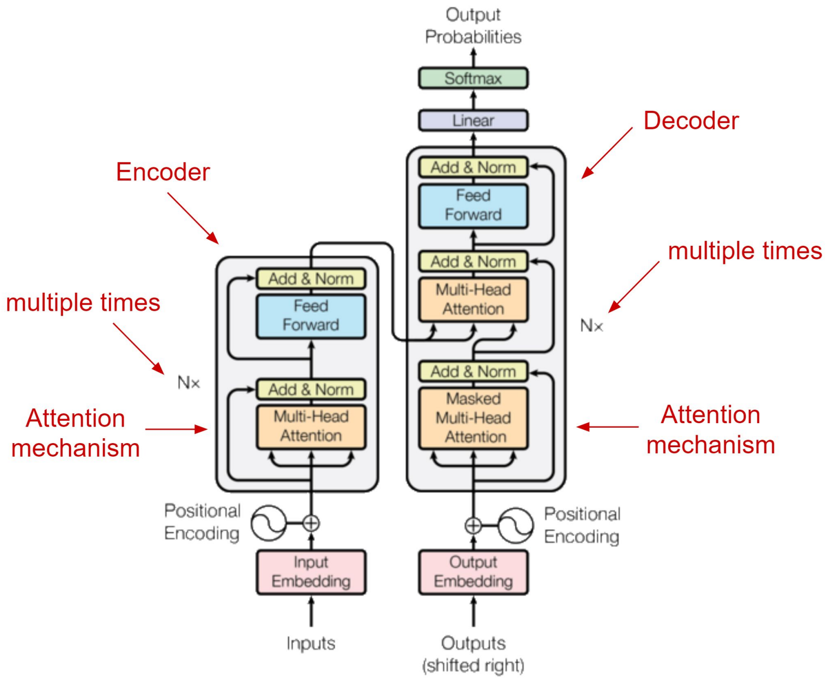
5 **lo** w
2 **lo** w e r
6 n e w est
3 w i d est

*Vocabulary*

l, o, w, e, r, n, w, s, t, i, d, es, est, **lo**

# GPT-2 MODEL

- Improved GPT-1 model
  - Layer normalization
  - Scaled weights of residual layers
  - Expanded vocabulary
  - Increased content size and batchsize
- 1.5B parameter model

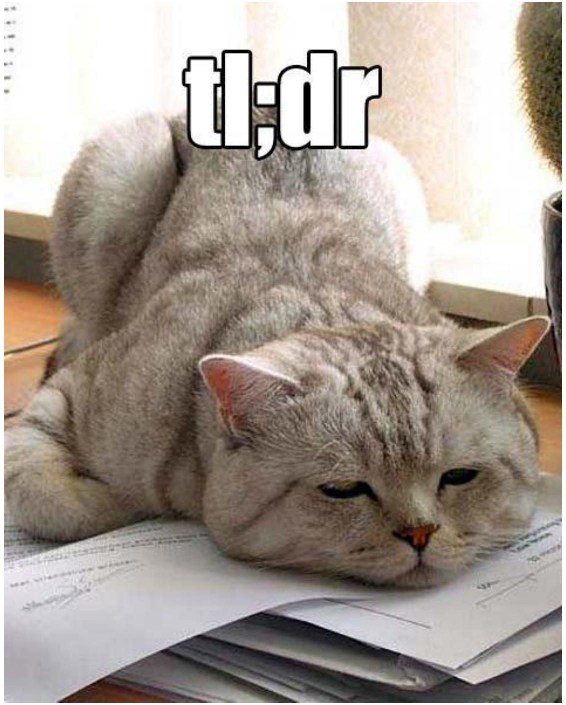


# EXPERIMENTS

1. Language Modeling
2. Children's Book Test: fill in the blank
3. LAMBADA: predict final word
4. Winograd Schema Challenge: reasoning
5. Reading Comprehension
6. Summarization
7. Translation
8. Question Answering

# SUMMARIZATION

- Summarize CNN and Daily Mail dataset
- Induce summarization behavior with “TL;DR:”
- Generate 100 tokens with Top-k random sampling with k=2
- Use first 3 generated sentences as the summary



# EXAMPLE

**Context (human-written):** In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

---

**GPT-2:** The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

# CONCLUSION

- Unsupervised task learning is a promising area of research
- Possible malicious applications
- Model available on GitHub:
    - https://github.com/openai/gpt-2

# Object Detection and Location

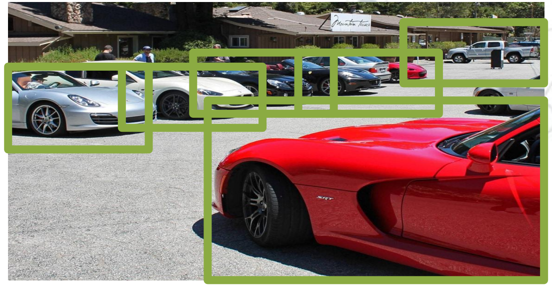
# Localization and Detection

Car

(Classification)

Car, but where?

(Classification with localization)



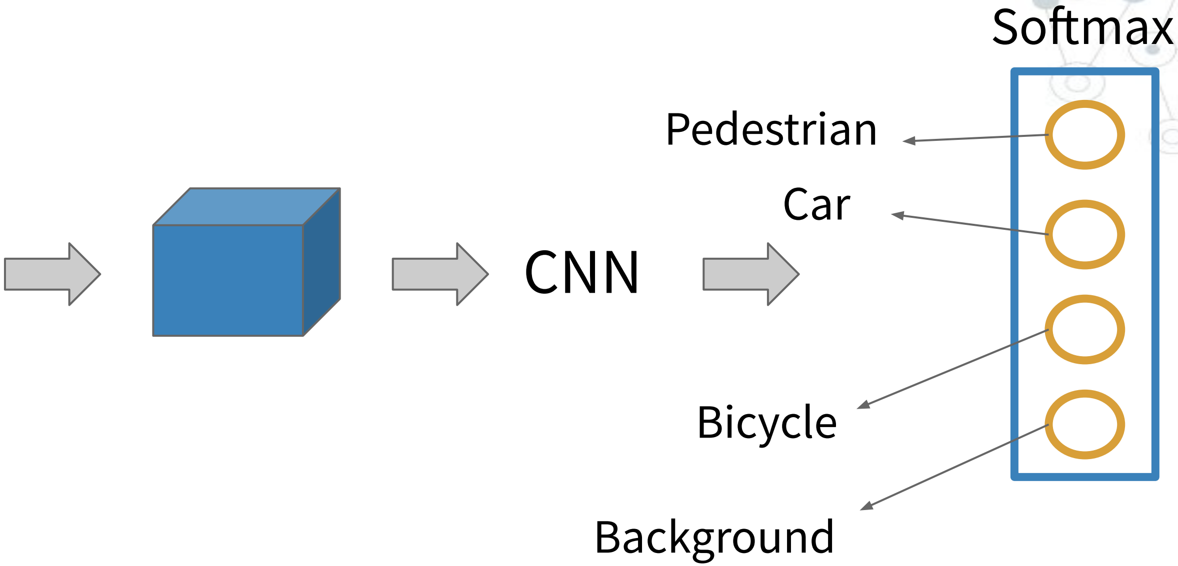
Multiple cars

(Detection)

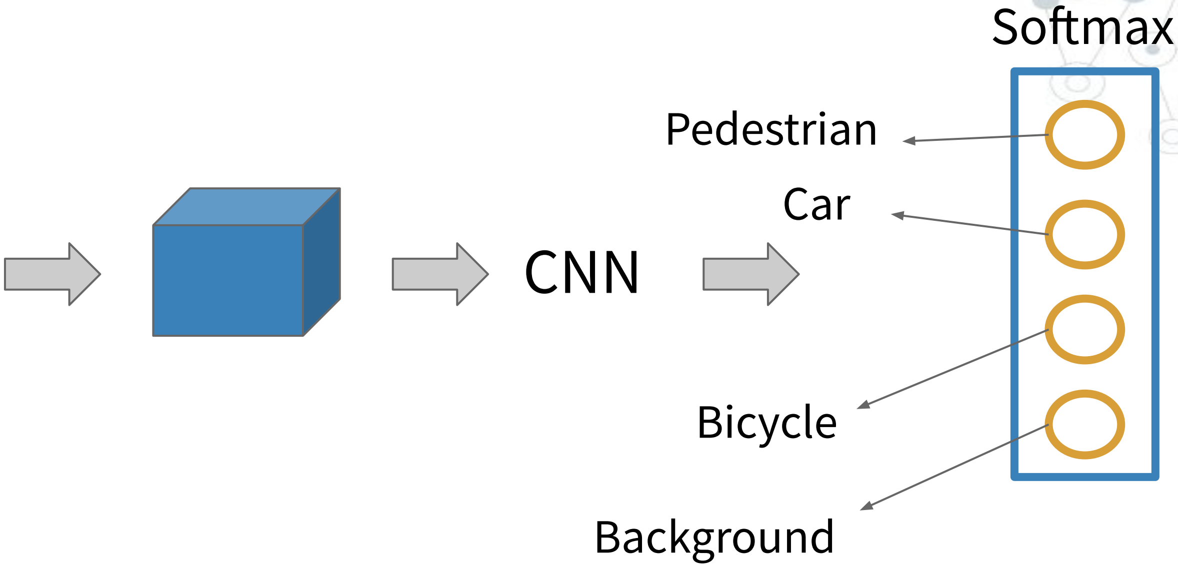
1 object

Multiple objects; could be from different classes

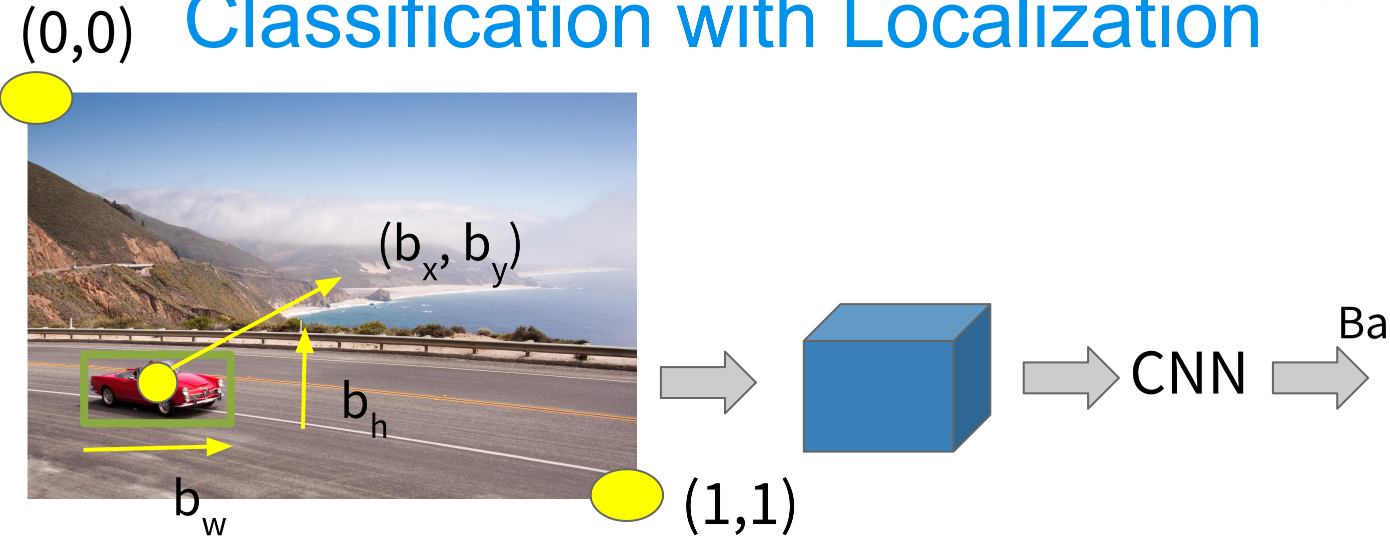
# Classification

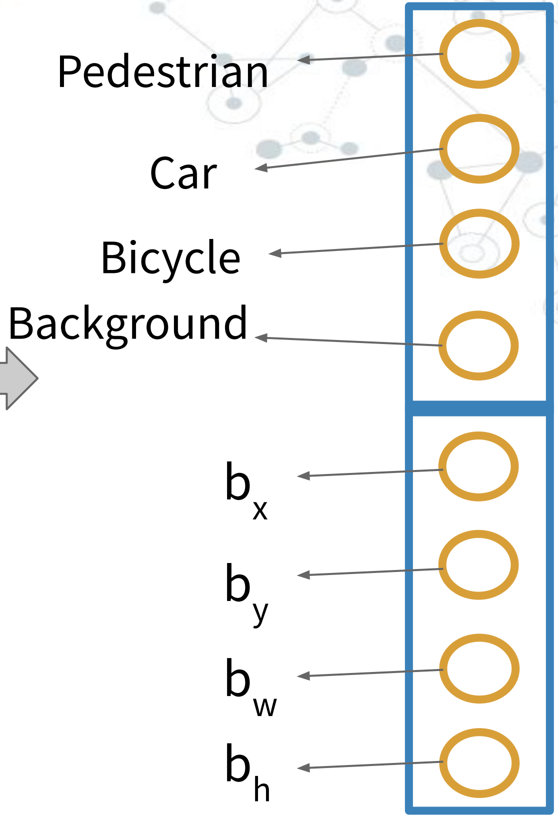


# Classification



# Classification with Localization




- To train a network to detect and locate objects,

  we need a lot of training data with bounding box labels:
  - $(b_x, b_y)$: the x and y coordinates of the center of the object
  - $b_w$: the width of the bounding box
  - $b_h$: the height of the bounding box
  - New image label: [class, $b_x$, $b_y$, $b_w$, $b_h$]

# Classification with Localization

<u>Classes</u>

Pedestrian ($c_1$)

Car ($c_2$)

Bicycle ($c_3$)

Background (no object)

New y label: [$p_d$, $b_x$, $b_y$, $b_w$, $b_h$, $c_1$, $c_2$, $c_3$]

Loss:

$$L(\hat{y}, y) = (\hat{y}_1, y_1)^2 + (\hat{y}_2, y_2)^2 \dots (\hat{y}_8, y_8)^2 \quad \text{if } p_d = 1$$

$$L(\hat{y}, y) = (\hat{y}_1, y_1)^2 \quad \text{if } p_d = 0$$

Note: assuming there is only 1 object in image

Probability there is an object in the image (and not just background)

# Localization and Detection

<u>Classes</u>

Pedestrian ($c_1$)

Car ($c_2$)

Bicycle ($c_3$)

Background (no object)

y = [1, 0.25, 0.75, 0.2, 0.15, 0, 1, 0]

New y label: [$p_d$, $b_x$, $b_y$, $b_w$, $b_h$, $c_1$, $c_2$, $c_3$]

Loss:

$$L(\hat{y}, y) = (\hat{y}_1, y_1)^2 + (\hat{y}_2, y_2)^2 \ldots (\hat{y}_8, y_8)^2$$ if $p_d = 1$

$$L(\hat{y}, y) = (\hat{y}_1, y_1)^2$$ if $p_d = 0$

# Localization and Detection

<u>Classes</u>

Pedestrian ($c_1$)

Car ($c_2$)

Bicycle ($c_3$)

Background (no object)

y = [0, ?, ?, ?, ?, ?, ?, ?]

New y label: [$p_d$, $b_x$, $b_y$, $b_w$, $b_h$, $c_1$, $c_2$, $c_3$]

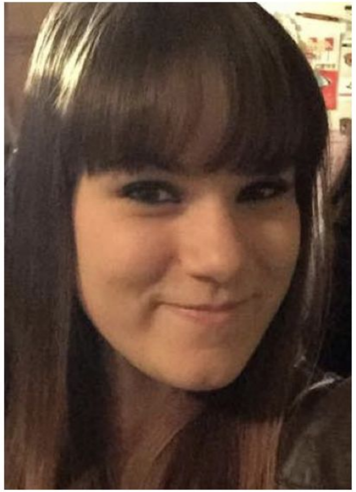
Loss:

$$L(\hat{y}, y) = (\hat{y}_1, y_1)^2 + (\hat{y}_2, y_2)^2 \ldots (\hat{y}_8, y_8)^2 \qquad \text{if } p_d = 1$$

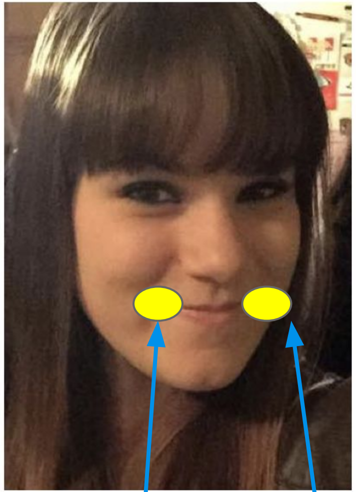
$$L(\hat{y}, y) = (\hat{y}_1, y_1)^2 \qquad \text{if } p_d = 0$$

# Landmark Detection

# Landmark Detection

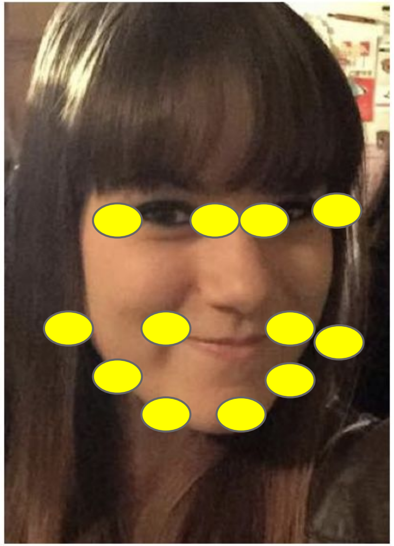


# Landmark Detection



$(l_{x1}, l_{y1})$ $(l_{x2}, l_{y2})$

# Landmark Detection



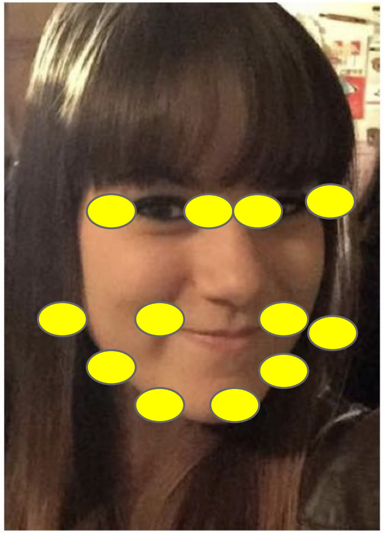
Label every point

Input: image with n landmarks
Output:
$[p_{face}, l_{x1}, l_{y1}, l_{x2}, l_{y2}, \ldots, l_{xn}, l_{yn}]$

Fit CNN to output if the image is of a face and the locations of the landmarks if it is a face

Note: landmarks have to be consistent across all training images, i.e. landmark # 1 is the left corner of the right eye, for example

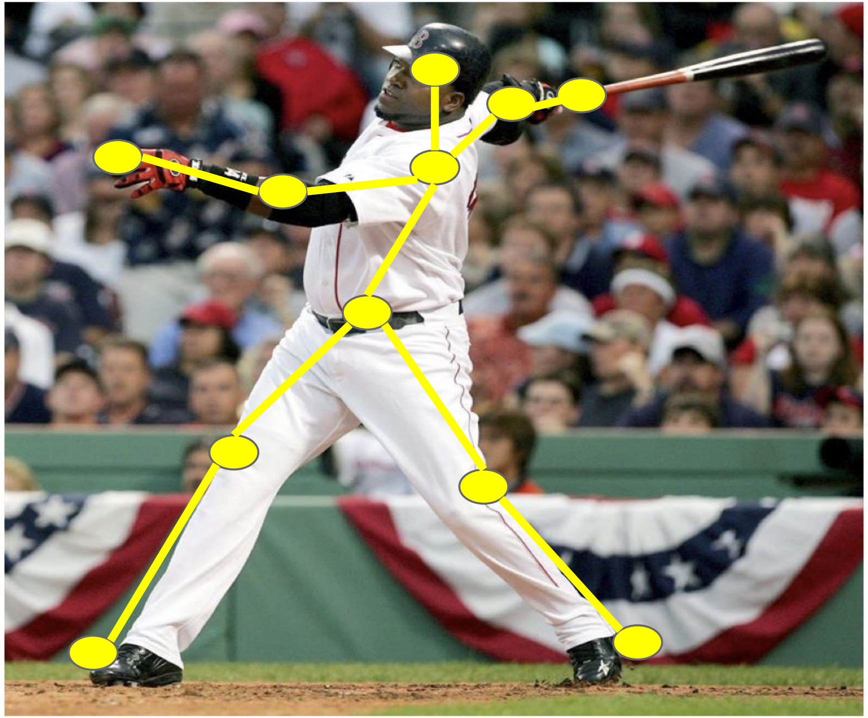
# Landmark Detection



If I can detect where the landmarks are, I can add filters in appropriate places

# Landmark Detection

# Landmark Detection



“Pose” detection

# Object Detection

# Object Detection

- Goal: locate and classify objects in an image
- Train CNN on cropped images of objects, where the object takes up most of the space in the image

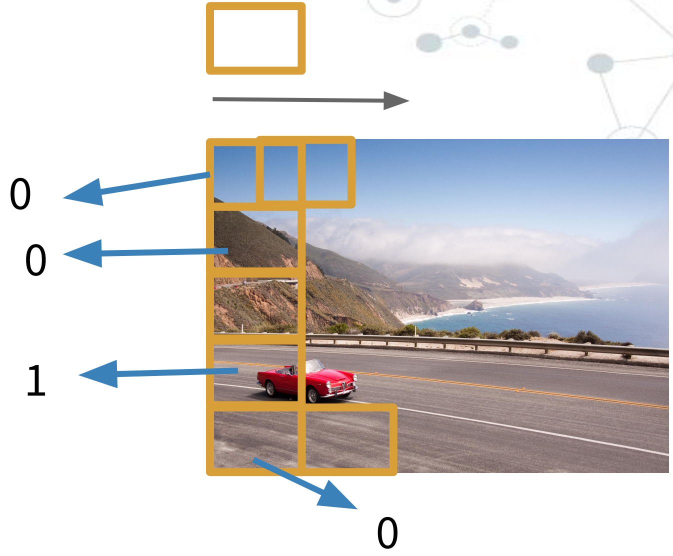
X: image of a car
y: 1

X: image of a car
y: 1

X: image of not a car
y: 0

# Object Detection

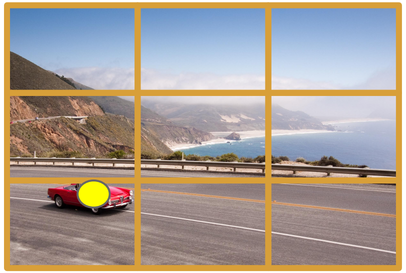
- Sliding windows detection algorithm
  - Slide a window across your image
  - In each region covered by the window, try to detect object (classify every region as containing an object or not)
  - **Very computationally expensive**, especially for small window and small stride
  - Bigger windows or strides result in fewer regions and less computational expense, but could hurt performance
  - Won't output the most accurate bounding boxes



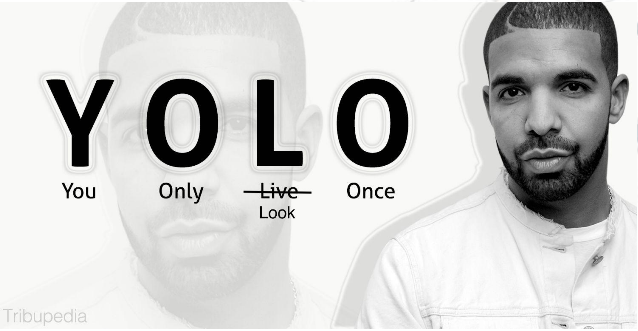

Repeat with larger and larger windows

# Object Detection

- One way to predict more accurate bounding boxes is by implementing the YOLO (You Only Look Once) algorithm
  - Redmon et al. 2015
  - Overly dramatic YOLO video
- Split image into grid cells
- Assign the object to the grid cell containing the midpoint of the object
- Works well when there is only 1 object in a particular cell
- Cuts down on computational cost because it can be run as a single convolutional implementation
  - So fast it performs well for real time object detection
- GitHub repo with easy implementation in Keras

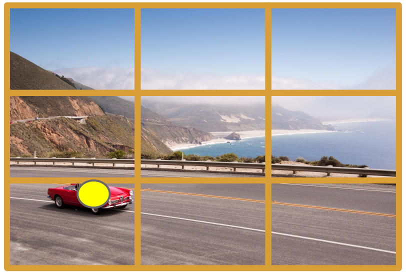


# Object Detection

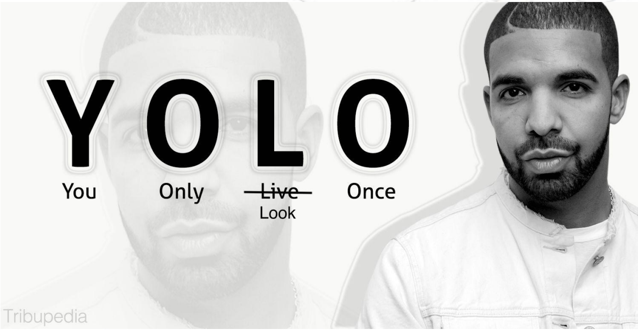



- One way to cut down on computational expense and predict more accurate bounding boxes is by implementing the YOLO (You Only Look Once) algorithm
  - Redmon et al. 2015
  - Overly dramatic YOLO video
- Split image into grid cells
- Assign the object to the grid cell containing the midpoint of the object
- Works well when there is only 1 object in a particular cell
- Cuts down on computational cost because it can be run as a single convolutional implementation
  - So fast it performs well for real time object detection
- GitHub repo with easy implementation in Keras

$y = [0, ?, ?, ?, ?, ?, ?, ?]$

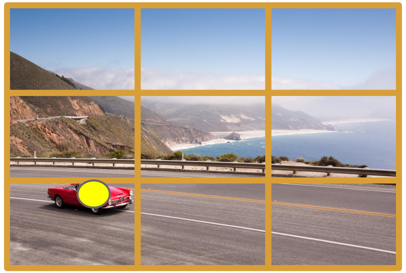


# Object Detection



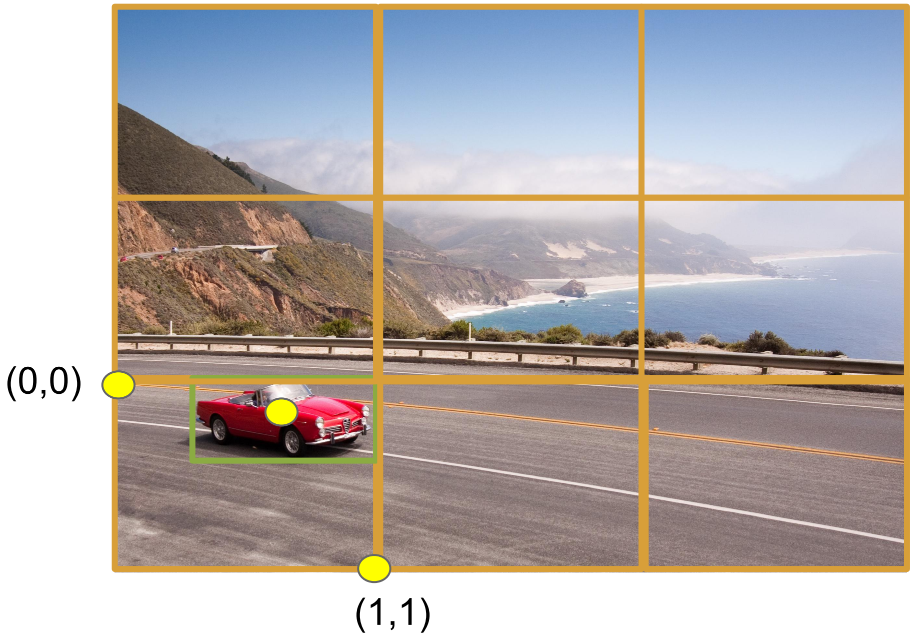

- One way to cut down on computational expense and predict more accurate bounding boxes is by implementing the YOLO (You Only Look Once) algorithm
  - Redmon et al. 2015
  - Overly dramatic YOLO video
- Split image into grid cells
- Assign the object to the grid cell containing the midpoint of the object
- Works well when there is only 1 object in a particular cell
- Cuts down on computational cost because it can be run as a single convolutional implementation
- So fast it performs well for real time object detection
- GitHub repo with easy implementation in Keras

$y = [1, b_x, b_y, b_w, b_h, 0, 1, 0]$ →



# Better Bounding Boxes

- $b_x$, $b_y$, $b_w$, $b_h$ are defined relative to the grid cell
- $b_x$, $b_y$ will be between 0 and 1 by definition
- $b_w$, $b_h$ could be greater than 1, depending on how large the object is and if it spans more than the grid cell with the midpoint



# Evaluating Object Localization

- How well is your algorithm working in terms of finding the bounding boxes?

# Evaluating Object Localization

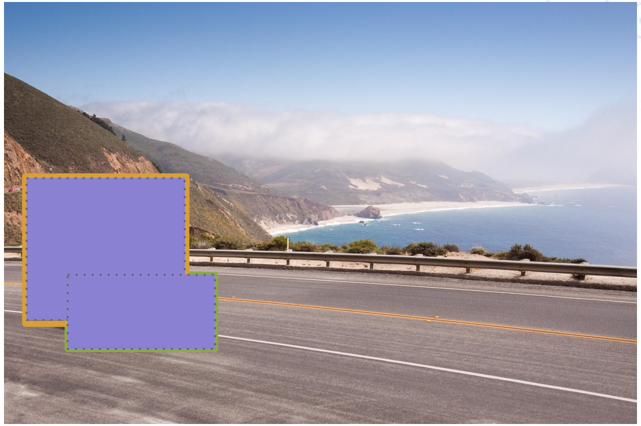
- How well is your algorithm working in terms of finding the bounding boxes?
- One metric to measure the performance of the algorithm is Intersection over Union

$$IoU = \frac{\text{size of intersection}}{\text{size of union}}$$

# Evaluating Object Localization

- How well is your algorithm working in terms of finding the bounding boxes?
- One metric to measure the performance of the algorithm is Intersection over Union

$$IoU = \frac{\text{size of intersection}}{\text{size of union}}$$



# Evaluating Object Localization

- How well is your algorithm working in terms of finding the bounding boxes?
- One metric to measure the performance of the algorithm is Intersection over Union

$$IoU = \frac{\text{size of intersection}}{\text{size of union}}$$

# Evaluating Object Localization

- How well is your algorithm working in terms of finding the bounding boxes?
- One metric to measure the performance of the algorithm is Intersection over Union

$$IoU = \frac{\text{size of intersection}}{\text{size of union}}$$

- "Correct" if , $IoU \geq 0.5$ or some other threshold
- Basically measures the overlap of the predicted bounding box with the ground truth bounding box - more overlap is better

# Non-max suppression

- Your algorithm may detect the same object multiple times
- Non-max suppression is a way to make sure you detect each object only once
- Steps
  - Discard all boxes with $p_d \leq 0.6$ (probability that object is detected)
  - While boxes remain: find box with highest $p_d$
  - Suppress (discard) all other boxes that have IoU $\geq 0.5$ with the box in the previous step
  - Repeat until no more boxes remain
- Repeat this process independently for each type of object you are trying to detect

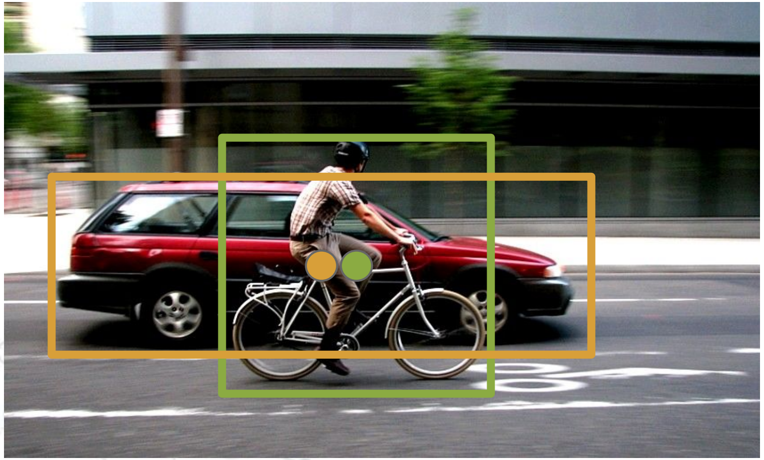
# Anchor Boxes

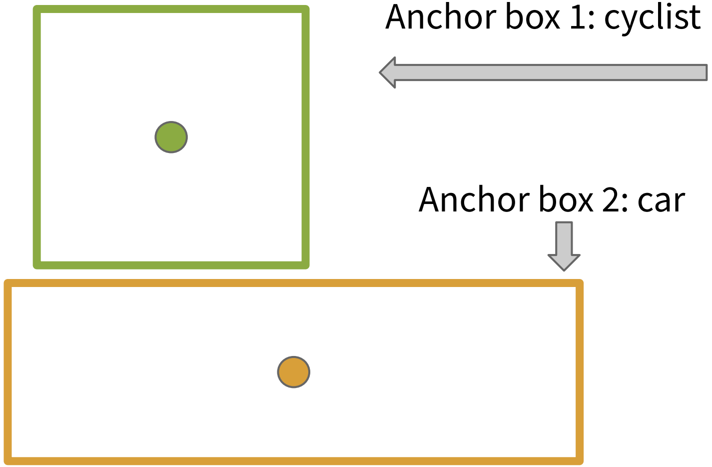
- So far we have assumed a grid cell can only detect one object
- What if you want to **detect multiple objects in the same cell**?
- Originally, we assigned each object in an image to the grid cell that contained its midpoint
- Now, we will assign an object to a grid cell that contains its midpoint and an anchor box for that cell with the highest IoU

# Anchor Boxes

$$y = [p_d, b_x, b_y, b_w, b_h, c_1, c_2, c_3, p_d, b_x, b_y, b_w, b_h, c_1, c_2, c_3]$$

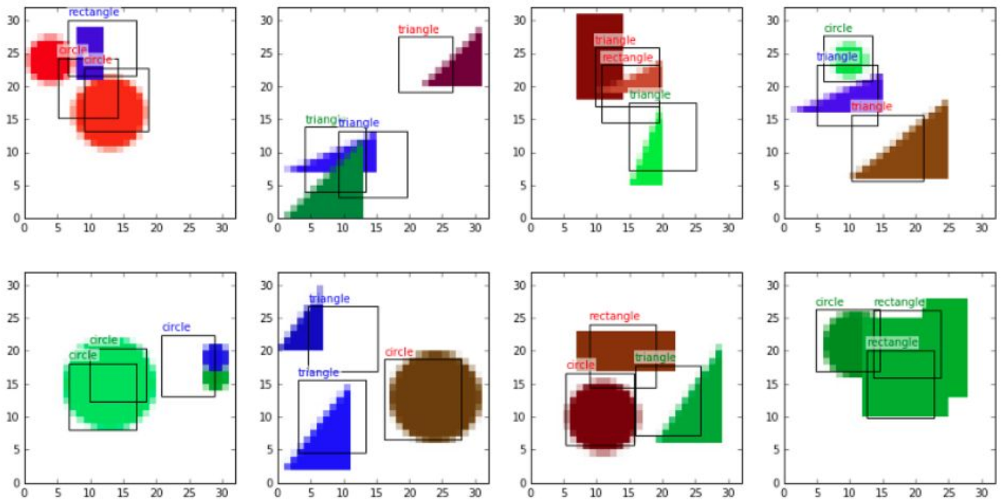
Anchor box 1 | Anchor box 2





# Object Detection Tutorial

- [Object detection with neural networks - a simple tutorial using Keras]()



# Face Recognition

# Terminology

- Recognition
  - Have a database of K persons
  - Get an input image
  - Output ID if the image is any of the K persons, or “not recognized” if not like any of the K persons

- Verification
  - Input image and name/ID
  - Output whether the input image is that of the claimed person

- [Andrew Ng demo video]

# Face Recognition

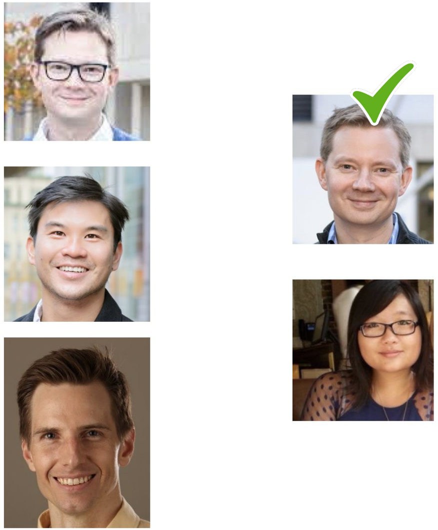
- One-shot Learning: learning from 1 example to recognize that person again
- Major downside: needs to be re-trained every time another person is added to group, only 1 example to learn from





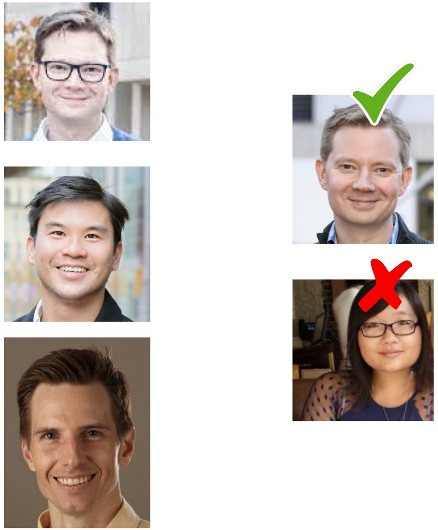
# Face Recognition

- One-shot Learning: learning from 1 example to recognize that person again
- Major downside: needs to be re-trained every time another person is added to group, only 1 example to learn from



# Face Recognition

- One-shot Learning: learning from 1 example to recognize that person again
- Major downside: needs to be re-trained every time another person is added to group, only 1 example to learn from



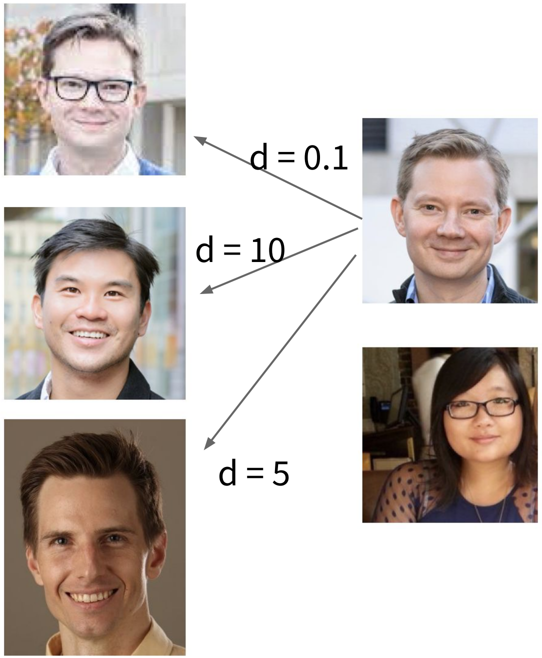
# Similarity Function

- Similarity function: quantify how similar or different two images are
- If difference is large, the images are of two different people
- If difference is small, the images are of the same person
- DeepFace by Taigman et al 2014
- Define the similarity network as

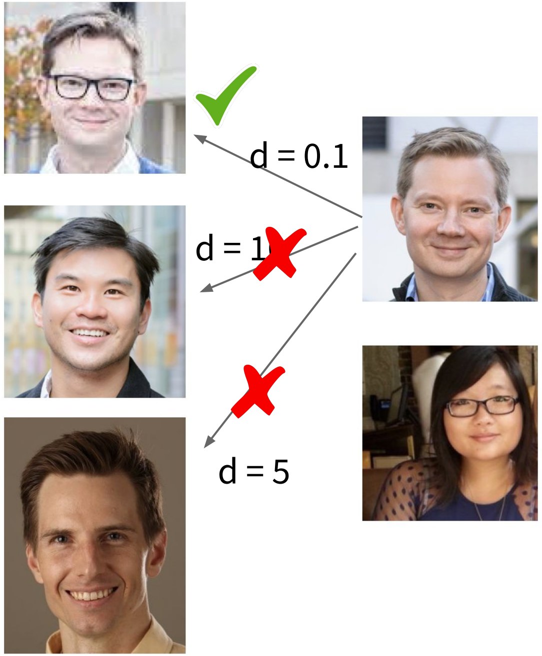
$$d(x^{(1)}, x^{(2)}) = \|f(x^{(1)}) - f(x^{(2)})\|_2^2$$

- Learn parameters such that if $x^{(i)}$ and $x^{(j)}$ are the same person, d is small, and if $x^{(i)}$ and $x^{(j)}$ are different people, d is large
- Come up with threshold of what is “small”

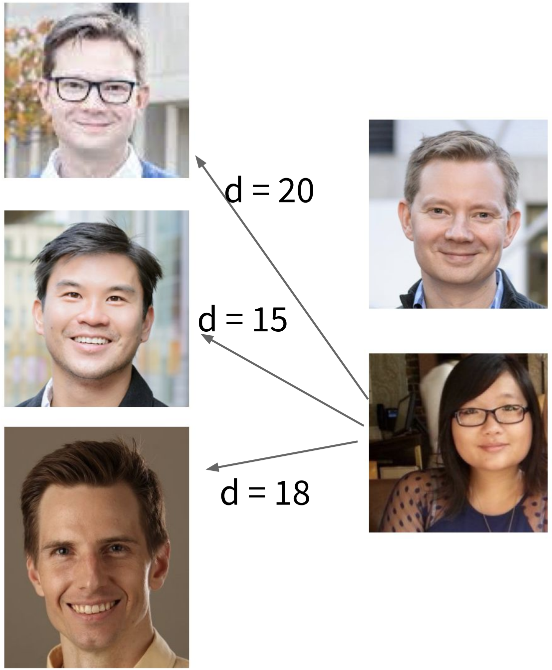
# Similarity Function



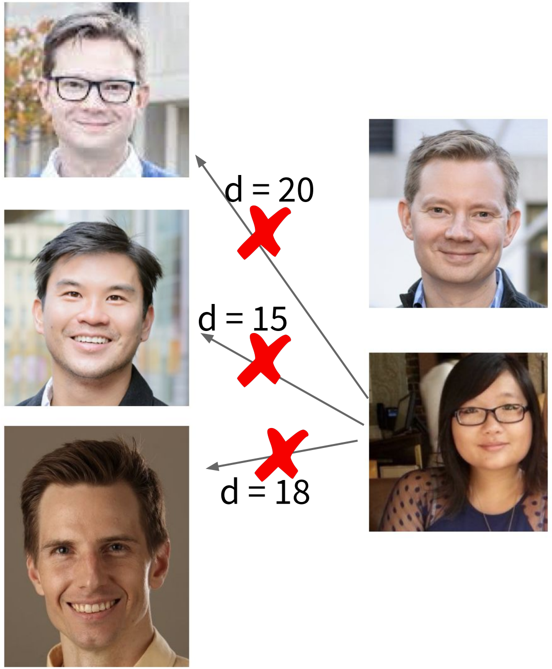
# Similarity Function



# Similarity Function



# Similarity Function




Not in database

# Triplet Loss

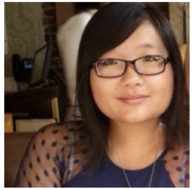
- To learn the parameters of your network (get good encodings for images), can use gradient descent to minimize the triplet loss
- Given 3 images A (anchor), P (positive) and N (negative), can we minimize the “triplet loss”:

$$L(A, P, N) = \max(\|f(A) - f(P)\|_2^2 - \|f(A) - f(N)\|_2^2 + \alpha, 0)$$

- Note that multiple pictures of each person are needed for this to be effective

Anchor
(A)

Positive
(P)



Anchor
(A)

Negative
(N)

# FaceNet

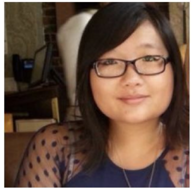
Margin parameter - ensures the network doesn't just label every difference as 0

$$d(A,P) + \alpha \leq d(A,N)$$

- During training, if A, P, and N are chosen randomly, is easily satisfied
  - It's really easy to randomly pick two very different looking people (if your sample is heterogeneous)
  - It's better to choose A, P, and N such that training is more difficult and will be better at recognizing differences on test sets

Anchor (A)

Positive (P)



Anchor (A)

Negative (N)

# Recurrent Neural Networks (RNNs)

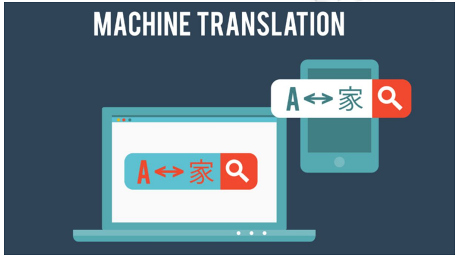
# Neural Networks

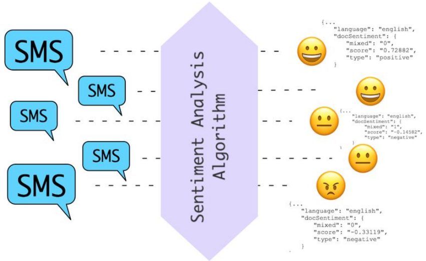
- So far we have seen:
  - Deep feedforward networks (MLPs)
    - Map a fixed length **vector** to a fixed length **scalar/vector**
    - Use case: classical machine learning
  - CNNS
    - Map a fixed length **matrix/tensor** to a fixed length **scalar/vector**
    - Use case: image recognition

- RNNs
  - Map a **sequence** of **matrices/tensors** to a **scalar/vector**
  - Map a **sequence** to a **sequence**
  - Use case: natural language processing (NLP)

# NLP

- The challenge of language for computers:
  - Computers are built to process numbers
  - Language isn’t easily represented by numbers
  - How can we represent human language in a computable fashion?
  - Applications: machine translation, text classification, information retrieval, sentiment analysis and many more
    - You already saw one example: classifying IMDb movie reviews as either positive or negative



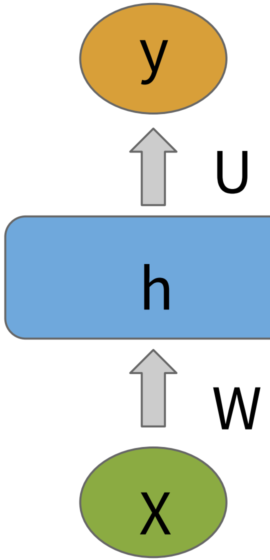
# MLPs → RNNs

- RNNs are a natural extension of MLPs
- MLPs are “memoryless”, but often we need knowledge of the past sequence of events to predict the future

| | Inputs | Output | Probability |
|---|---|---|---|
| MLP | X | y | P(y\|X) |
| RNN | $[x_1, x_2, x_3, \ldots, x_t]$ | y | $P(y \mid x_1, x_2, x_3, \ldots, x_t)$ |

# MLPs ⟶ RNNs

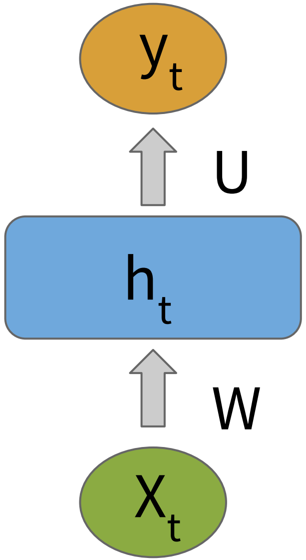
- Recall that the first hidden layer for an MLP is given by h = f(XW + b) where f() is the activation function and W is the weight matrix in the hidden layer, b is the bias term, and U is the weight matrix in the output layer



# MLPs ⟶ RNNs

- RNNs add the concept of "state" to traditional neural networks

- To incorporate the notion of time we will index the hidden layer with t and feed it $X_t$:
  $h_t = f(X_t W + b)$



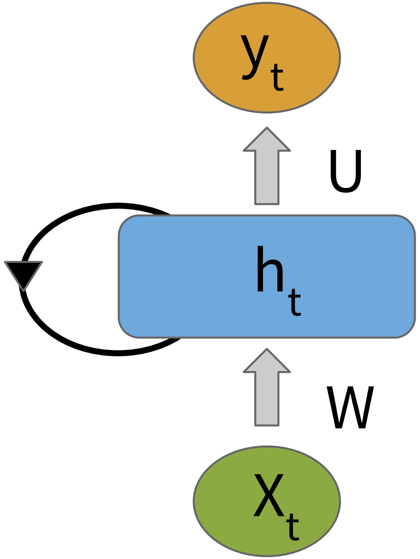
# MLPs ⟶ RNNs

- To incorporate information from the previous state we will make the following modification:

$h_t = f(X_t W + b) \longrightarrow h_t = f(X_t W + h_{t-1} U + b)$

Input at time t

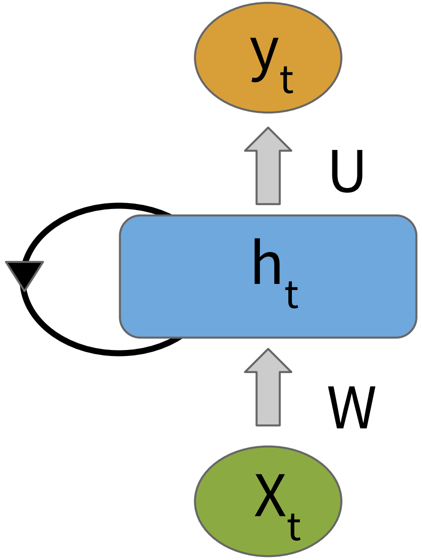
Hidden state from previous time point




- This is equivalent to connecting the hidden state to itself

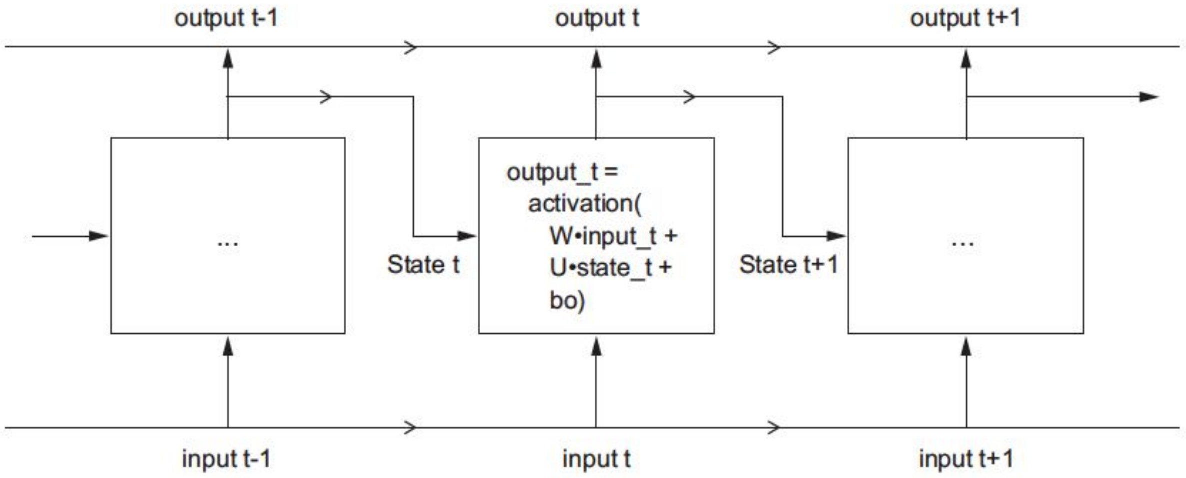
# RNN Backprop

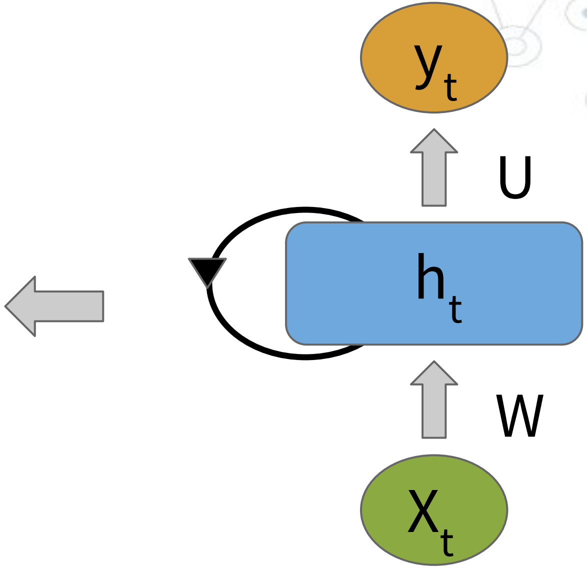
- How do we backprop through something with a loop?
- Have to backprop through depth <u>and</u> time
- This is similar to what we saw with MLPs, but we aren't going to go through it here



# RNNs

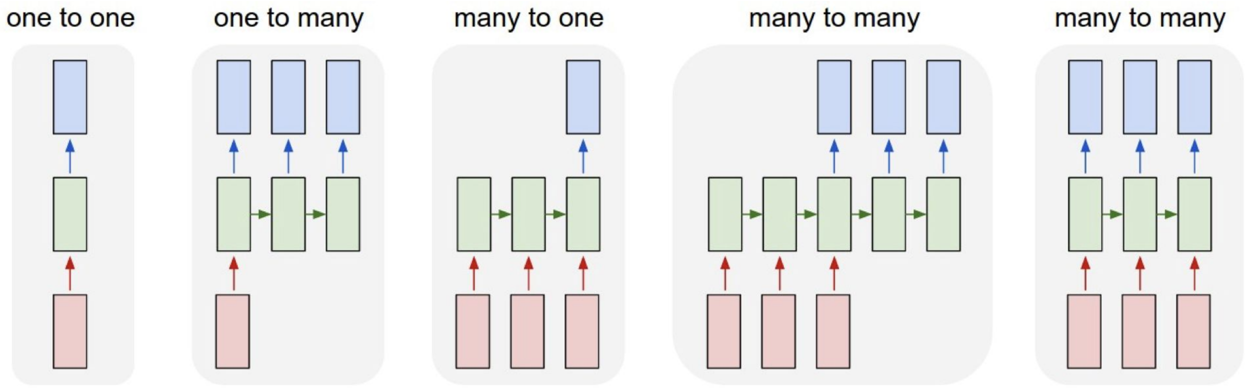
"Unrolled" RNN



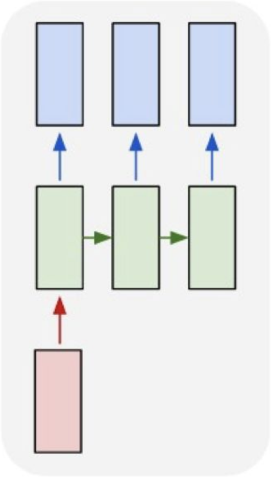
# RNNs

- There are many ways to configure the input ⇨ output mapping

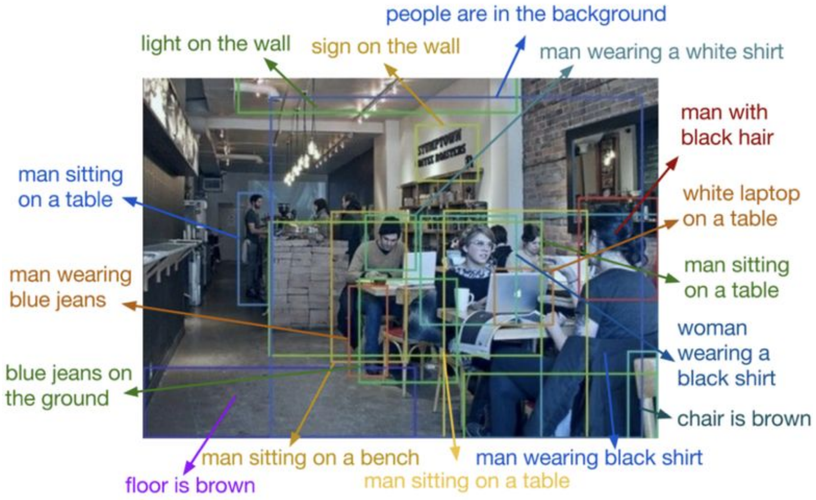


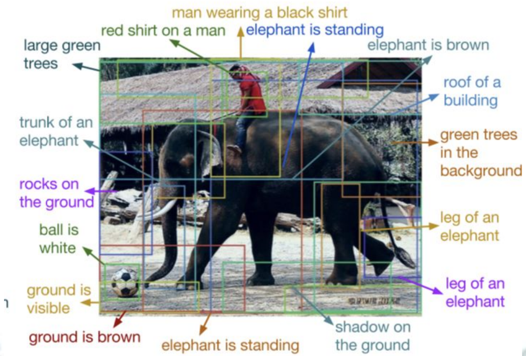
# RNNs

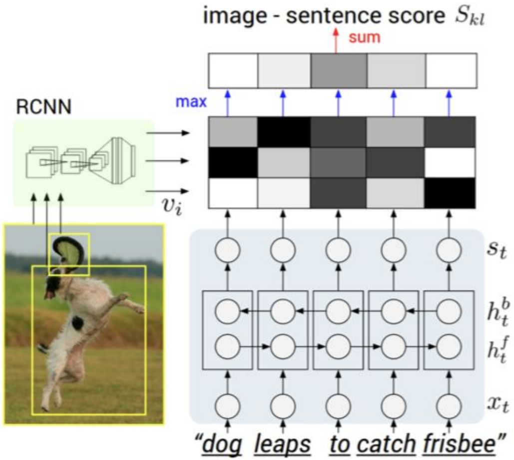
one to many



Ex: image captioning

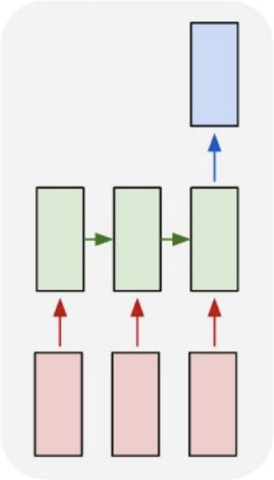


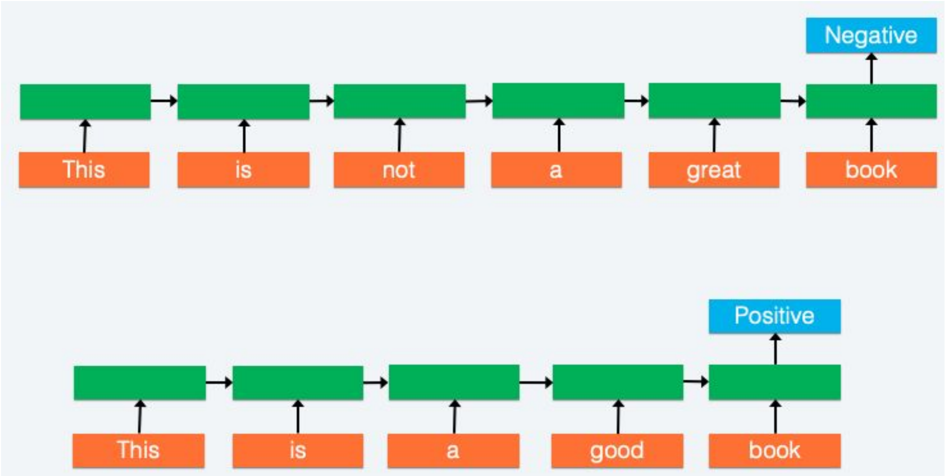



# RNNs

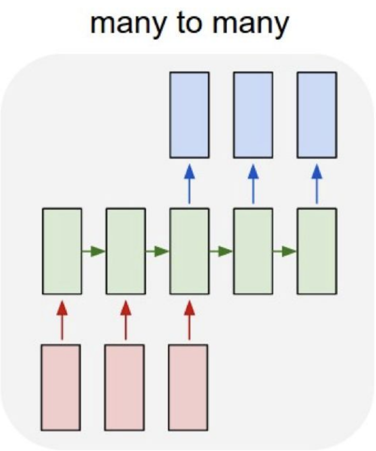
many to one



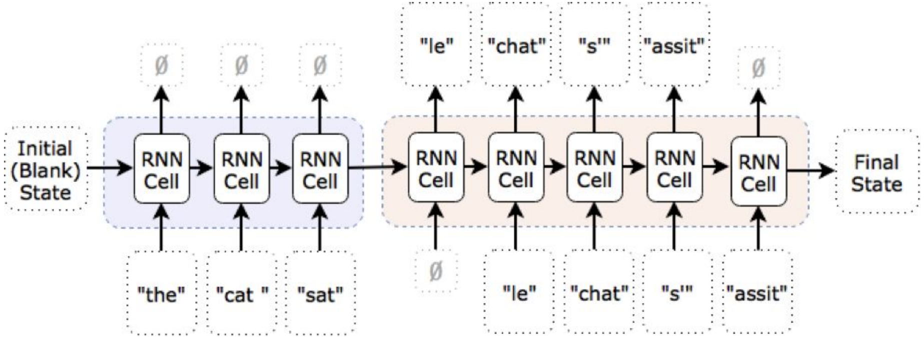
Ex:
Sentiment
Analysis

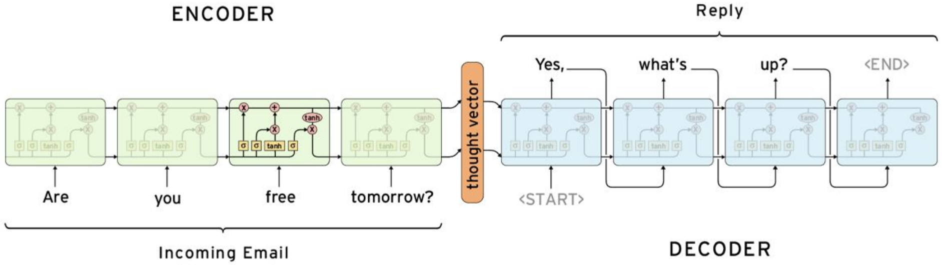


# RNNs

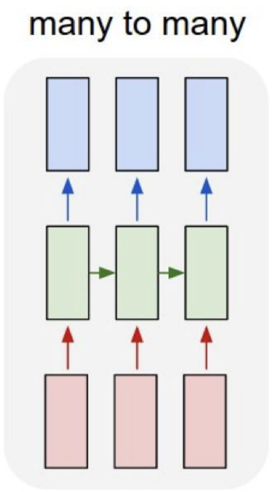



Ex:
Translation,
automated
response

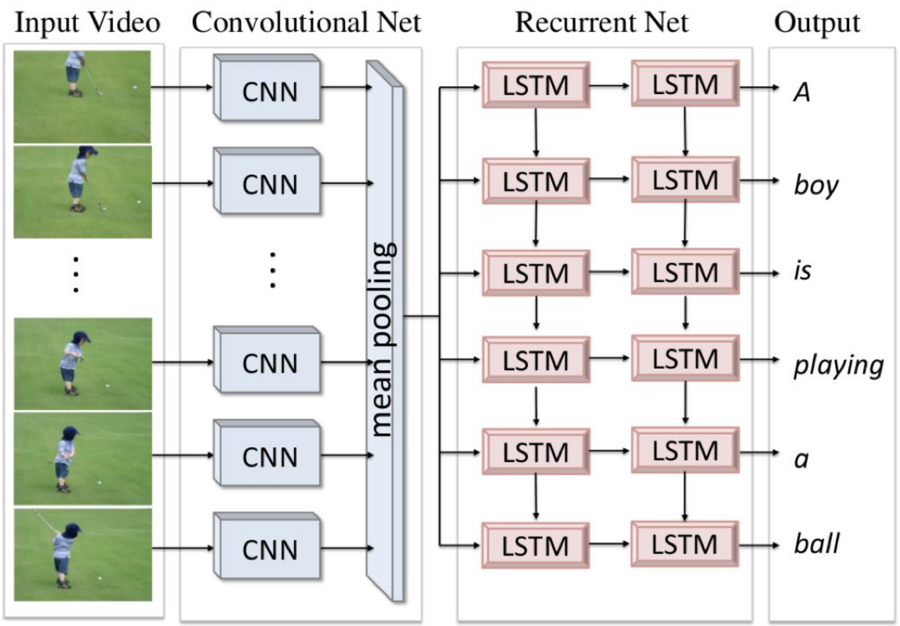


# RNNs




Ex: frame by frame image captioning



# RNNs

- High-level takeaways:
  - RNNs provide a way to handle **sequence** data where the order of events is important
  - Simple modification to MLP model
  - RNNs maintain a “**state**” that reflects current configuration of the “world”

# RNNs

- High-level takeaways:
  - RNNs provide a natural way to “update” your beliefs about the world as new information arrives
  - Really **flexible** and can model many different scenarios that get weird/complicated quickly
  - CNNs = hard to understand but easy to implement; RNNs = easy to understand but hard to implement

# Applications

- Document and time series classification e.g. identifying the topic of an article or the author of a book
- Time series comparisons e.g. estimating how closely related two documents are
- Sentiment analysis
- Time series forecasting e.g. predicting weather (something that needs major improvement for Boston...)
- Sequence-to-sequence learning e.g. decoding an English sentence into Turkish