Agenda

Open–CE discussion and demonstration

Computer Vision Demo : Social Distancing Use Case

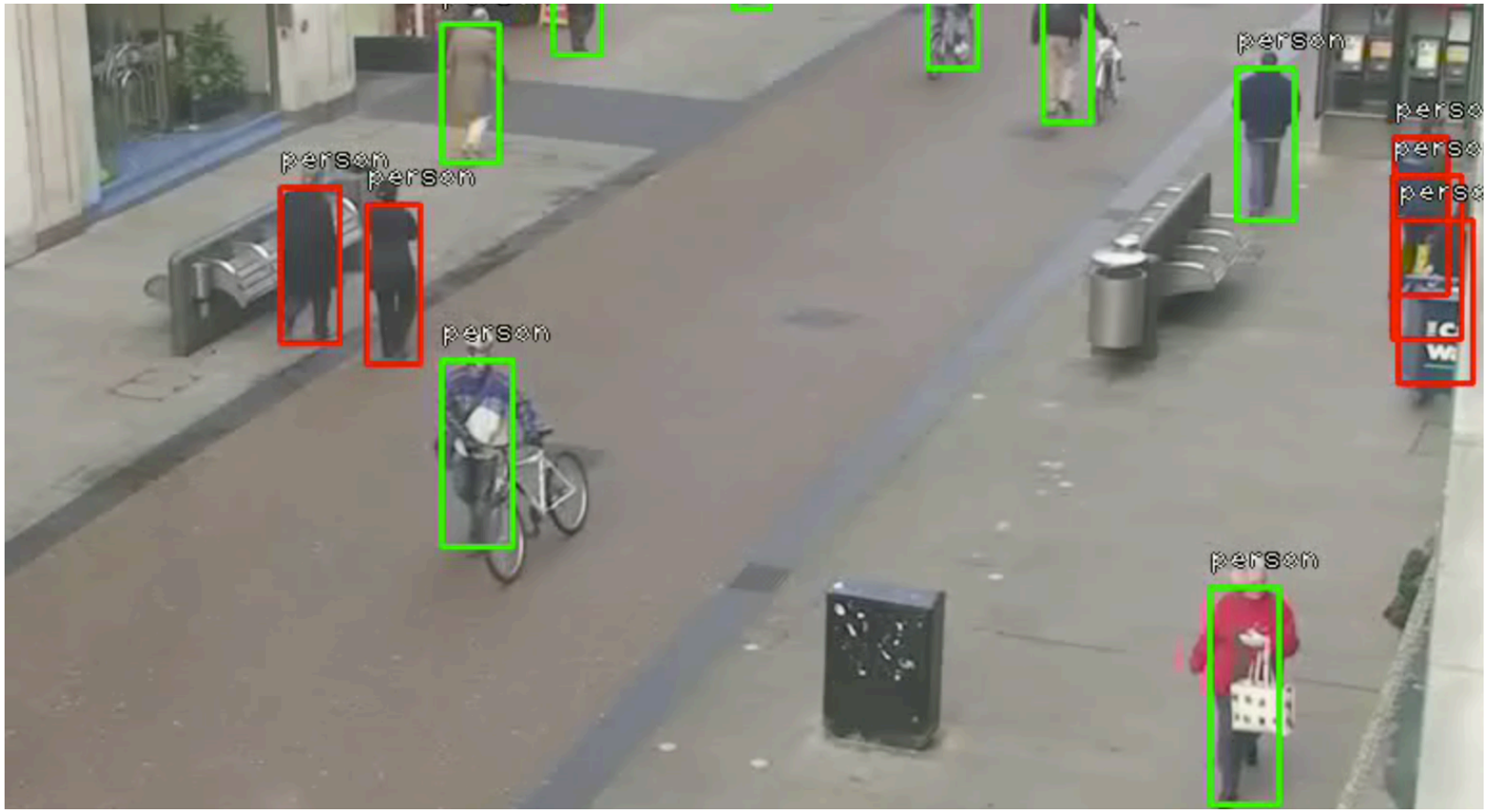
Natural Language Demo : Sentiment Analysis using FastAI

Content Links :
https://github.com/dustinvanstee/2020-11-install-opence-fastai

Vision
https://github.com/dustinvanstee/2020-11-wisc
https://github.com/yhenon/pytorch-retinanet

NLP
https://github.com/fastai/fastbook

Video Social Distancing Use Case



Requires

Object Detection of Person

- labelled dataset
- Pre-trained model
- RetinaNet Implementation

Pixel to World Coordinate Mapping

- homography
- kd-tree for selecting pairs

Image Processing Tools

- Numpy
- OpenCV / ffmpeg
- matplotlib

Goal : can you identify areas in image where social distancing hotspots exist ?
Uses : Could be used in public areas//factories to identify hot spot zones

What are localization and detection ?

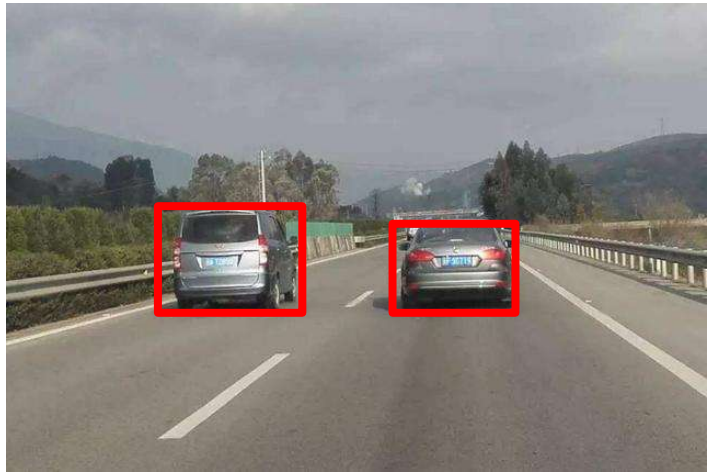
Image classification

Is there a car ?
Give possibility of a car

Classification with localization

Where is the car ?
Locate the object

Detection



Is there any target objects ?
- Multi-object (Car, pedestrian , Motocycle)
- Multi-instance

Label their locations

Multi-object , Single-instance

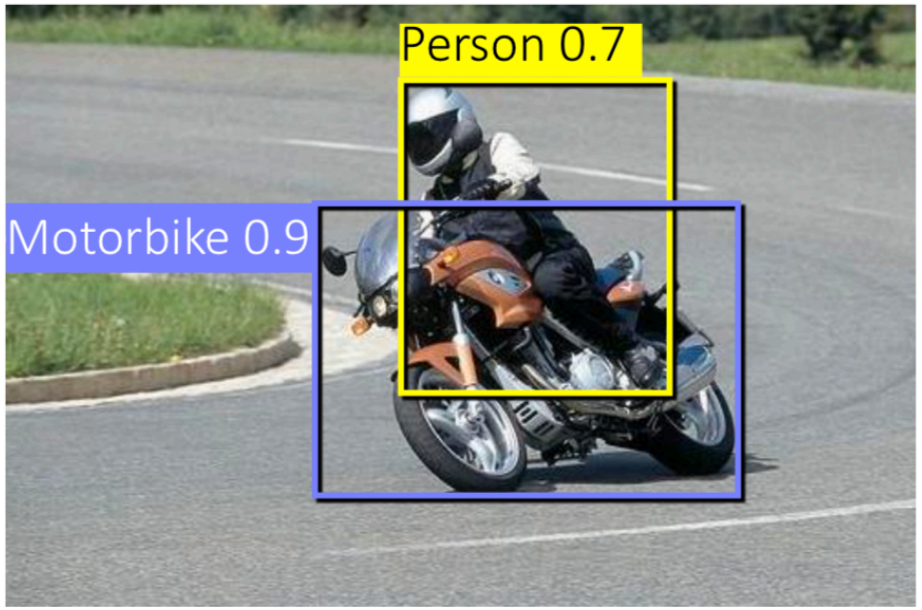
Multi-object , Multi-instance

Object Detection : Problem Formulation

Identify K classes of objects in images and predict bounding boxes :

`{person, bird, motorcycle, car}` ~ this list is configurable based on the application.

Input



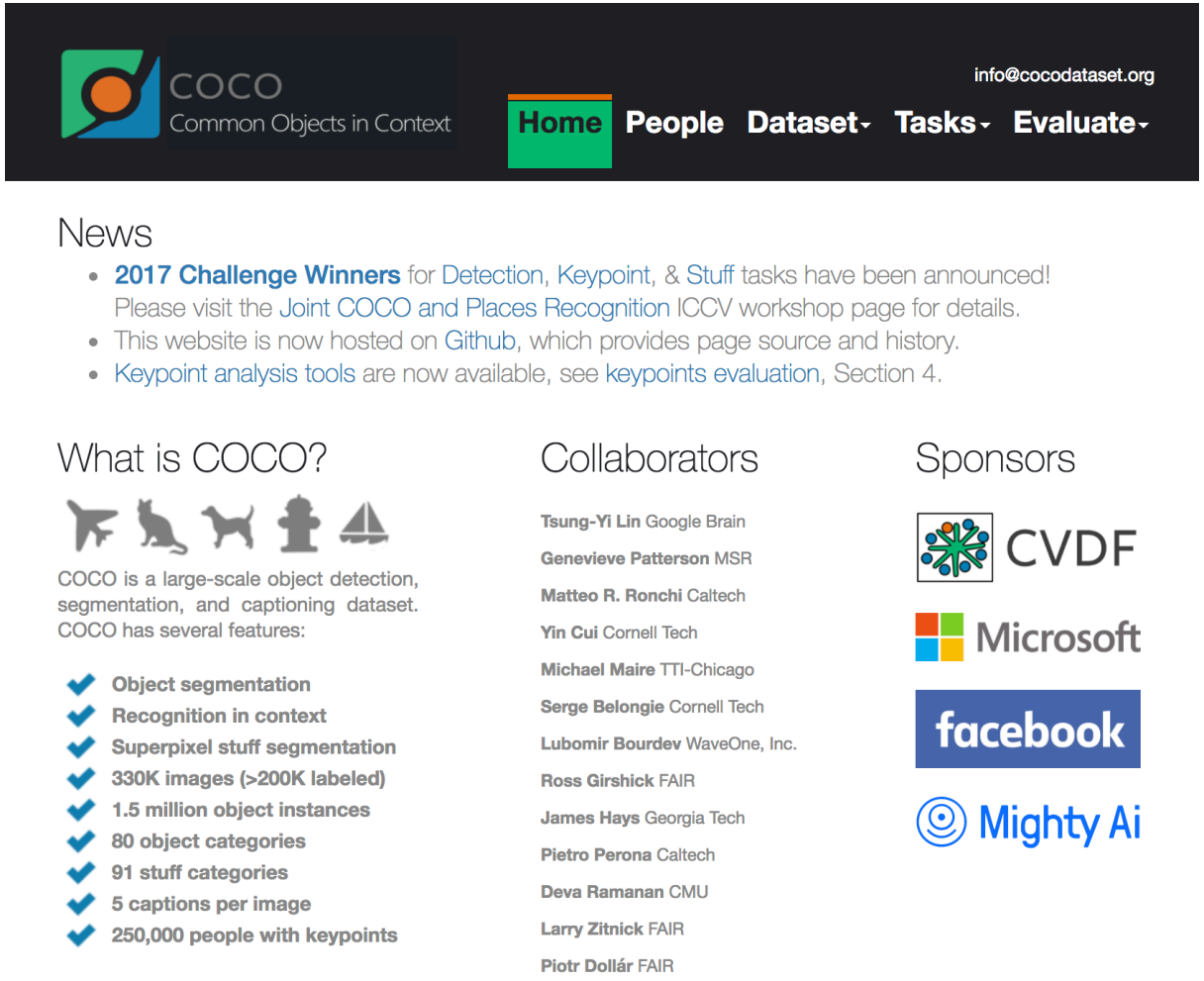
Desired output

*Actual results may vary

Object Detection Datasets

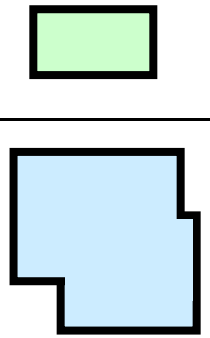
Lots of freely available data sets, here are some of the most popular

- MS COCO - http://cocodataset.org/#home
  - 80 Classes, 1.5 M labelled bounding boxes
- Pascal VOC - http://host.robots.ox.ac.uk/pascal/VOC/
  - 20 Classes, ~52K bounding boxes
- Imagenet - http://image-net.org/download-bboxes
  - 3000 classes, ~450K labelled bounding boxes



Evaluating object localization

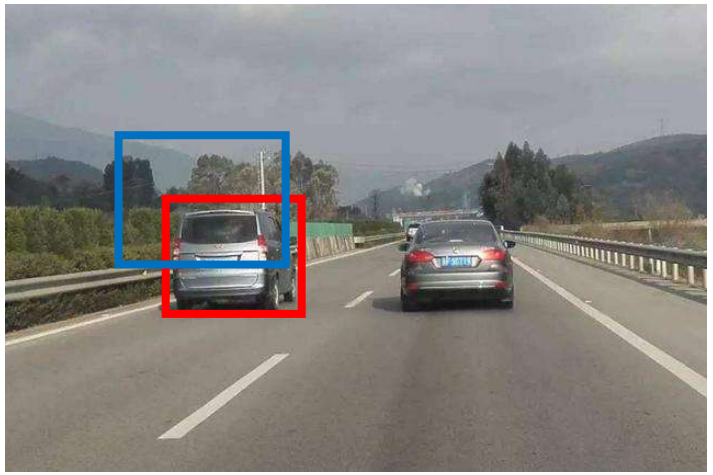
- Intersection over Union（IoU）
  - Evaluate accuracy of object location prediction
- eg :
  - Object real position
  - Object predict position
  - $\text{IoU} = \frac{\text{Intersection}}{\text{Union}} = \frac{\quad}{\quad} \geq 0.5$



Usually IoU > 0.5

Union

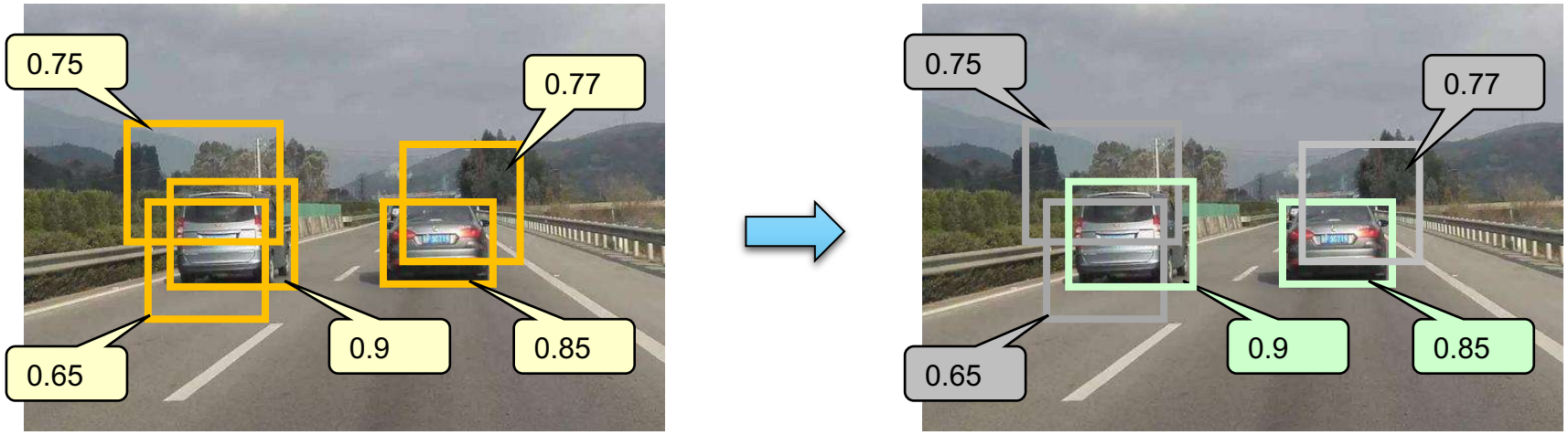
Intersection



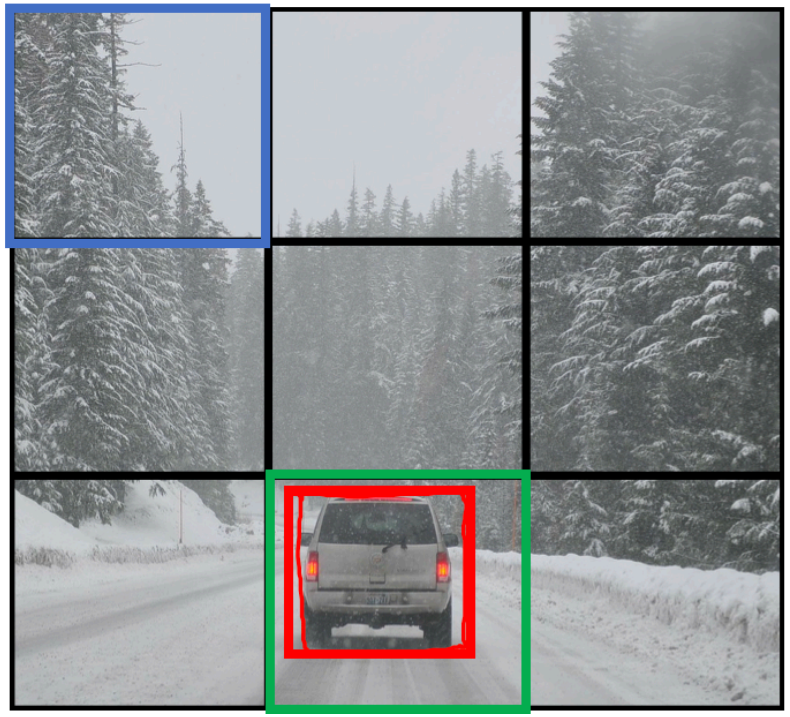
More generally, IoU is a measure of the overlap between two bounding boxes.

Non-max suppression example

- Problem: The same object may be positioned multiple times
  - In classification with localization, many cell / grids have high probability （p_c） of thinking they contains midpoint （b_x, b_y）
- Solution
  - Use the box of highest probability（b_x, b_y, b_h, b_w）
  - And, suppress other boxes with high overlap（large IoU）



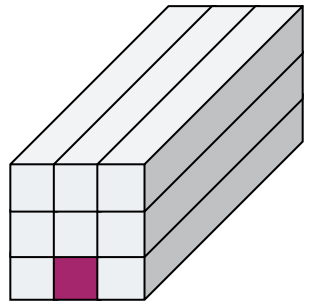
Obj Detection Example – Encode Image



Foreach grid cell build this ->

Pc	1	Pc = probability there is an object in grid cell
Bx	0.5	Bx = x center location
By	0.8	By = ycenter location
Bh	.28	Bh = height of object
BL	.28	BL = width of object
C1	0	
C2	1	Cx = one hot class vector
C3	0	
…	…	

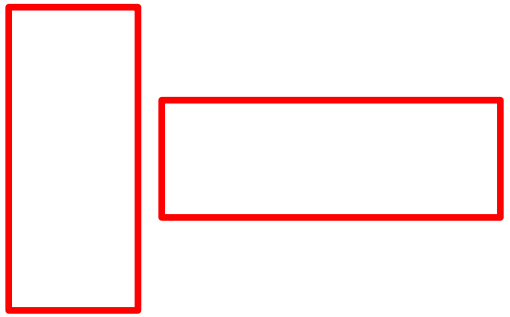
End result ~ (3 x 3 x 8) tensor for each image



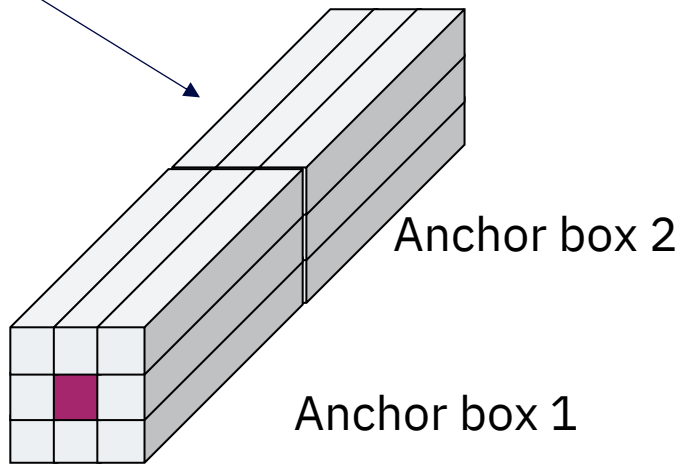
Key Concept - Anchor Boxes [advanced]

What if there are multiple objects at the same grid location?
Solution ->

1. Define K anchor boxes of different aspect ratio
2. Use IOU to determine which anchor box best fits the object during labelling
3. Encode object in proper anchor box
4. Anchor boxes multiply the number of outputs to predict
5. Train network with anchor boxes



Example of image with 2 objects in same grid cell
Without Anchor Boxes, only one box would be drawn!



OBJ Detection Example – Inference

Bounding Box prediction workflow

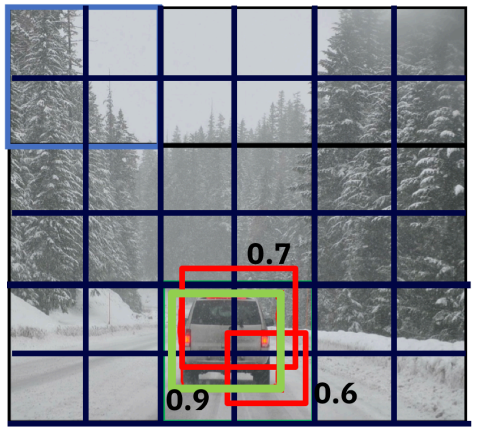
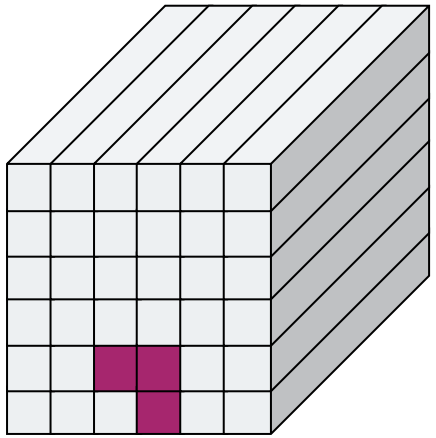


Image of X x Y Pixels

Image scaling + Deep CNN



Each grid cell consists of box predictions

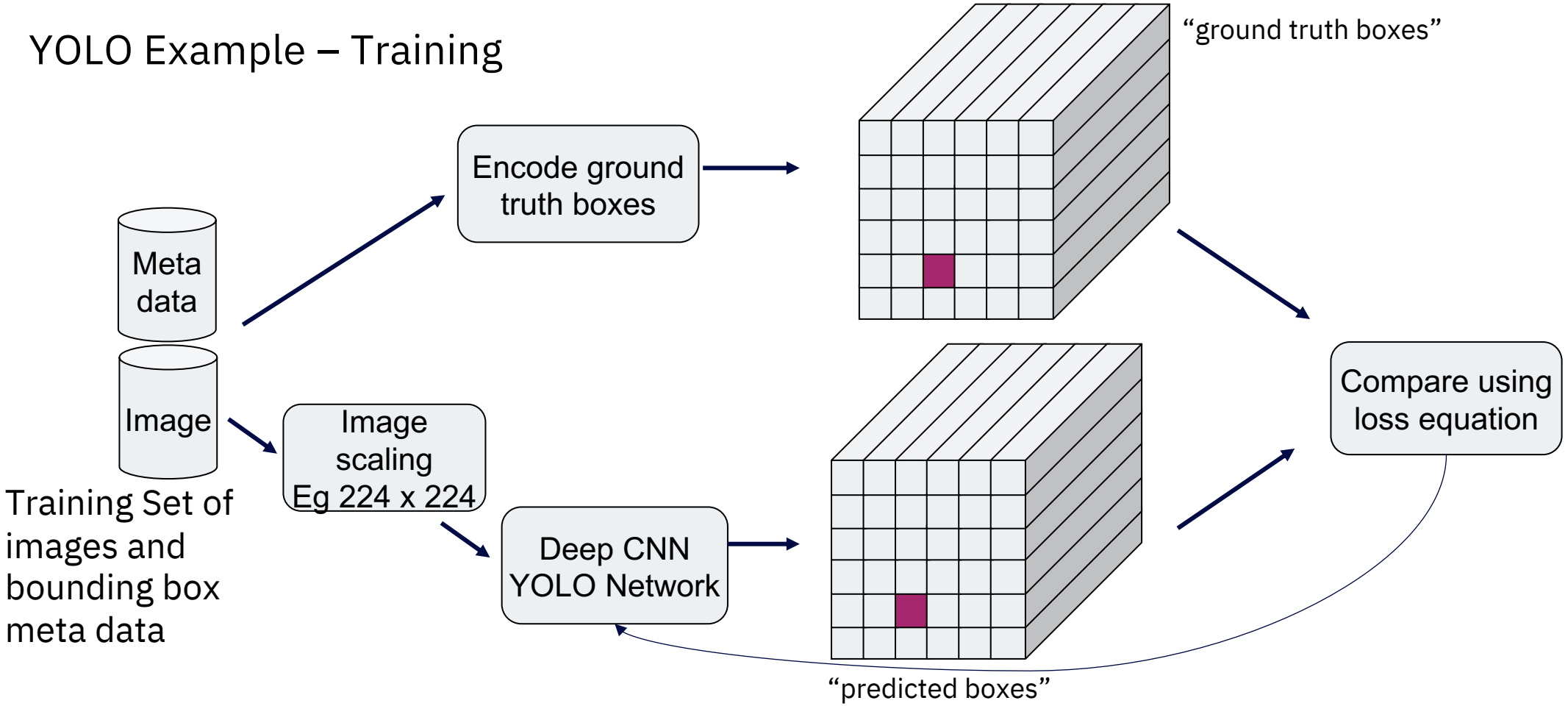
Box Pruning

Remove low probability boxes

Remove overlapping boxes

Variable length List of boxes

YOLO Example – Training

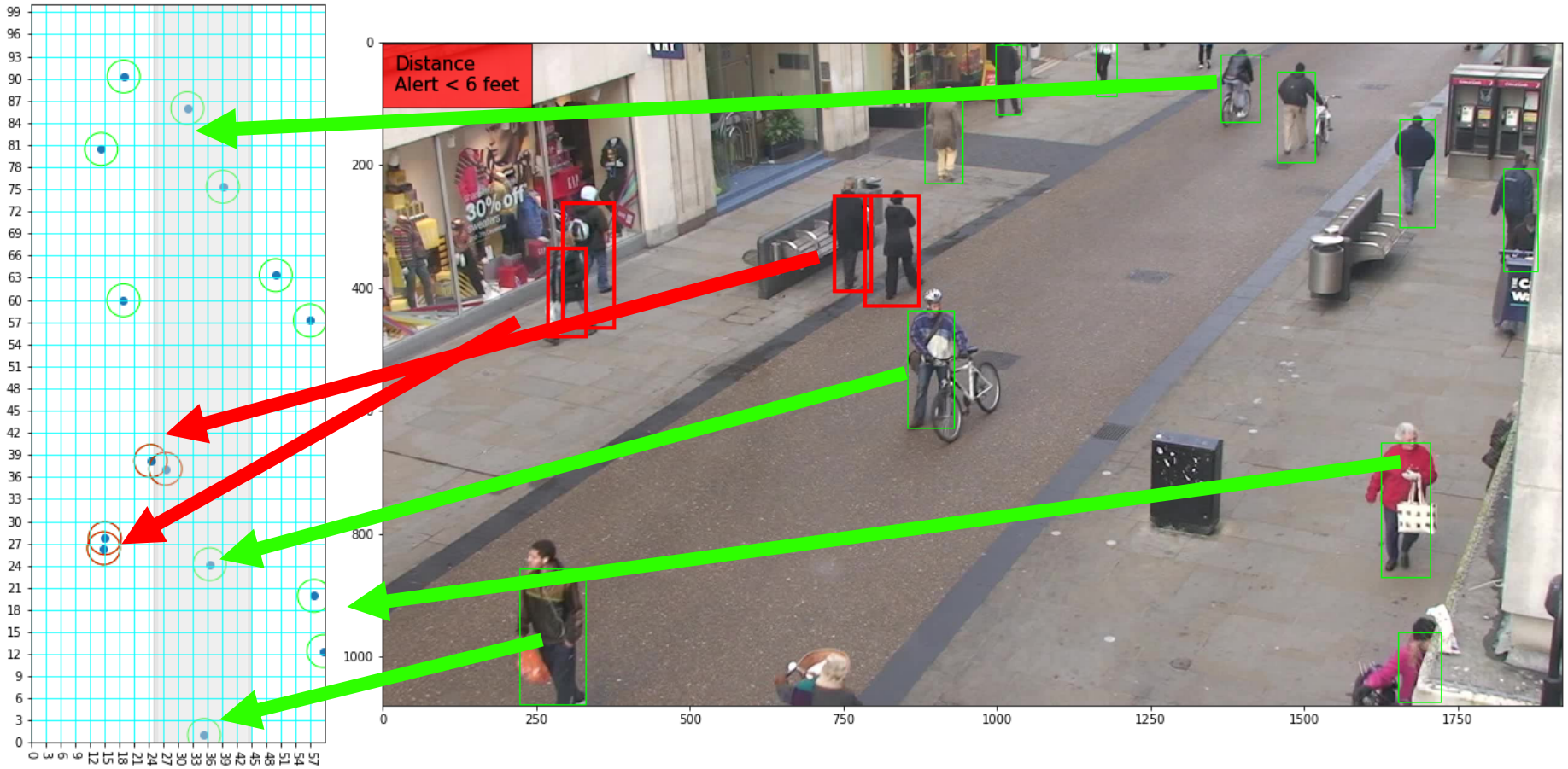


Training Set of images and bounding box meta data

"update weights using backpropagation technique"

Homograpy Trick

Mapping pixel coordinates to real world coordinates



Rectangle color coded to indicate distances in real world smaller than 6 feet

Homography

- **Compute perspective Transform – Homography**

```
pts_src = np.array([[143, 468], [789,178], [541,544], [1194, 671], [1083,214], [1560, 273]])
pts_dst = np.array([[10,25],    [10,65],   [24.7,25], [44.7,25],   [24.7, 65], [44.7,65]])

# calculate matrix H. DO this once for an image to compute H for camera setup,
# resuse H over and over for rest of images

homograpy, status  = cv2.findHomography(pts_src, pts_dst)
homograpy_inv, status = cv2.findHomography(pts_dst, pts_src)
```

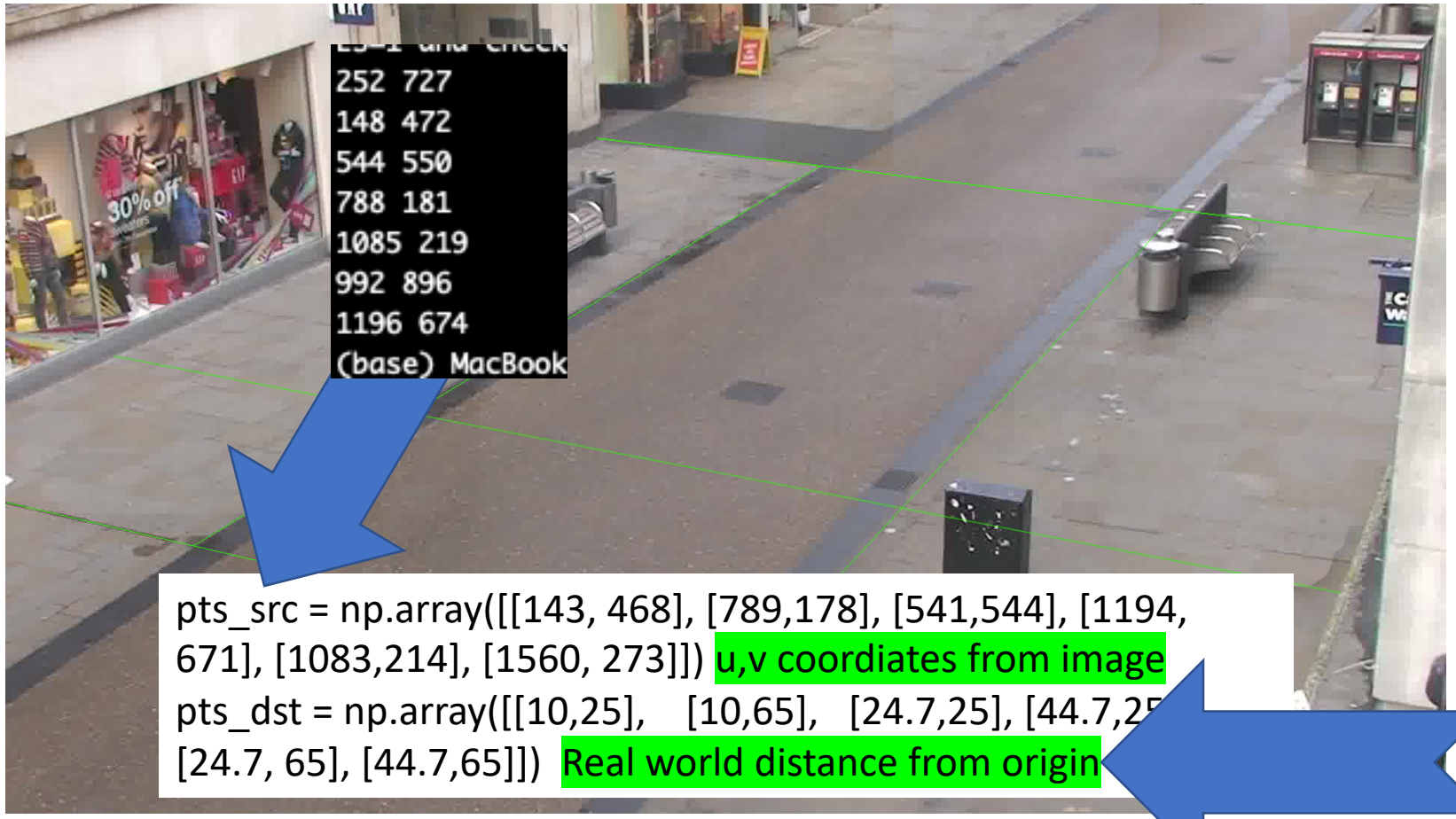
- **Use Homography to compute world coordinates**

```
footPositionPx = np.array([[[143, 468], [789,178], [541,544], [1194, 671], [1083,214], [1560, 273]]], dtype='float32')
print(footPositionPx.shape)
FootPostionWorldWc = cv2.perspectiveTransform(footPositionPx, homograpy)
FootPostionWorldWc
```

```
(1, 6, 2)
array([[[10.013301, 24.920633],
        [10.161524, 65.04503 ],
        [24.677074, 25.029789],
        [44.709366, 25.050034],
        [24.425364, 65.02927 ],
        [44.813377, 64.925255]]], dtype=float32)
```

Prepare Data: register world coordinates

One time camera calibration: establish correlation between real world coordiantes and camera image points



python ImageMouseClick.py

Numbers here are a little different from the original calibration values I encoded into program

Originally I used point 541, 544 (u,v)

This time I clicked in new areas and got values like 544, 550 (see the black text)

Compared to values from Google Earth, or measurements onsite, or CAD drawings

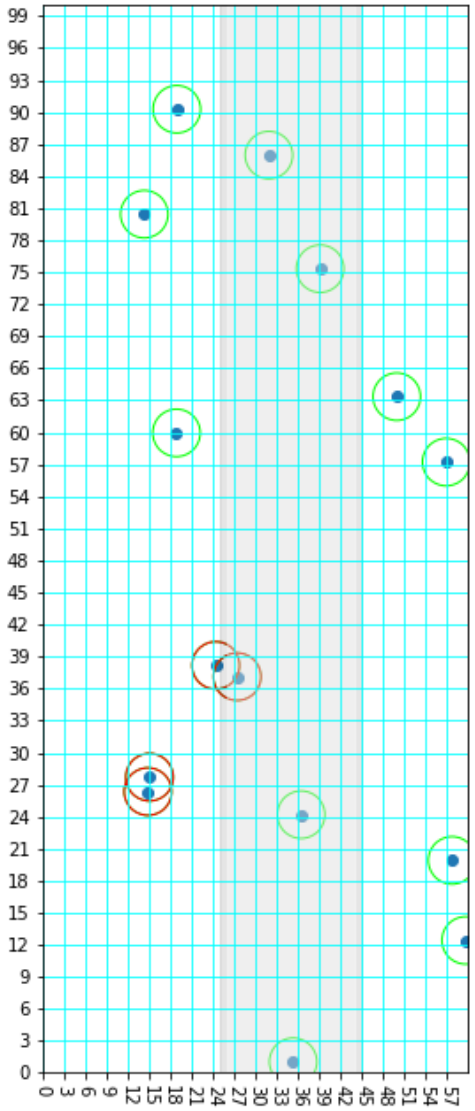
Finding closest pairs – use KDTree

- Use KDTree to find closest neighbors within radius of 6 feet
  - Outputs a set with tuples of the indexes
    {(0, 6), (3,5)}

```
R = 6
T = KDTree(oneFrameOut[0,:])
pairs = T.query_pairs(r=R)
pairs
```

- x1, y1 = pointsOut[0,pair[1],0], pointsOut[0,pair[1],0]

Result: World coordinates on left: Distances Flagged, alert could be sent



Attribution:

- Dataset: https://megapixels.cc/oxford_town_centre/
- @online{megapixels.cc, author = {Harvey, Adam. LaPlace, Jules.}, title = {MegaPixels.cc: Origins, Ethics, and Privacy Implications of Publicly Available Face Recognition Image Datasets}, year = 2019, url = {https://megapixels.cc/}, urldate = {2019-04-18} }

- blog/PR from April 16 from company called Landing AI - describing similar approach but without PAIV https://landing.ai/landing-ai-creates-an-ai-tool-to-help-customers-monitor-social-distancing-in-the-workplace/

NLP Use Case : IMDB sentiment prediction

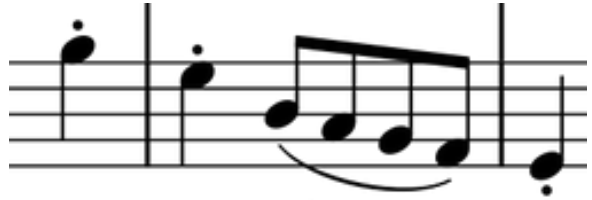
Examples of sequence data

Speech recognition

→ “The quick brown fox jumped over the lazy dog.”

Music generation

Ø

→ 

Sentiment classification

“There is nothing to like in this movie.”

→ 

DNA sequence analysis

AGCCCCTGTGAGGAACTAG

→ AGCCCCTGTGAGGAACTAG

Machine translation

Voulez-vous chanter avec moi?

→ Do you want to sing with me?

Video activity recognition

→ Running

Name entity recognition

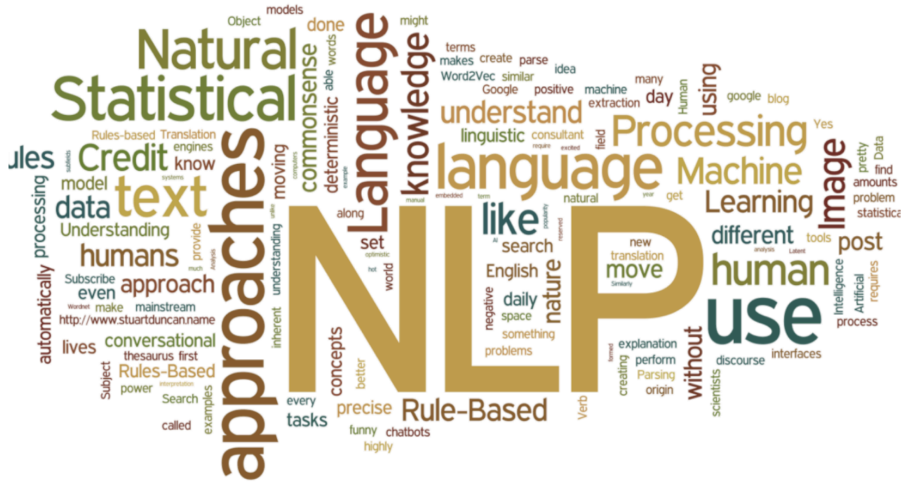
Yesterday, Harry Potter met Hermione Granger.

→ Yesterday, Harry Potter met Hermione Granger.

NLP Use Cases

- Uses
  - Sentiment Analysis
  - Document classification
  - Spam filtering
  - Topic modelling
  - Word Embeddings

- Classical Method Used
  - Naive Bayes
  - SVM
  - Logistic Regression
  - Tree Methods

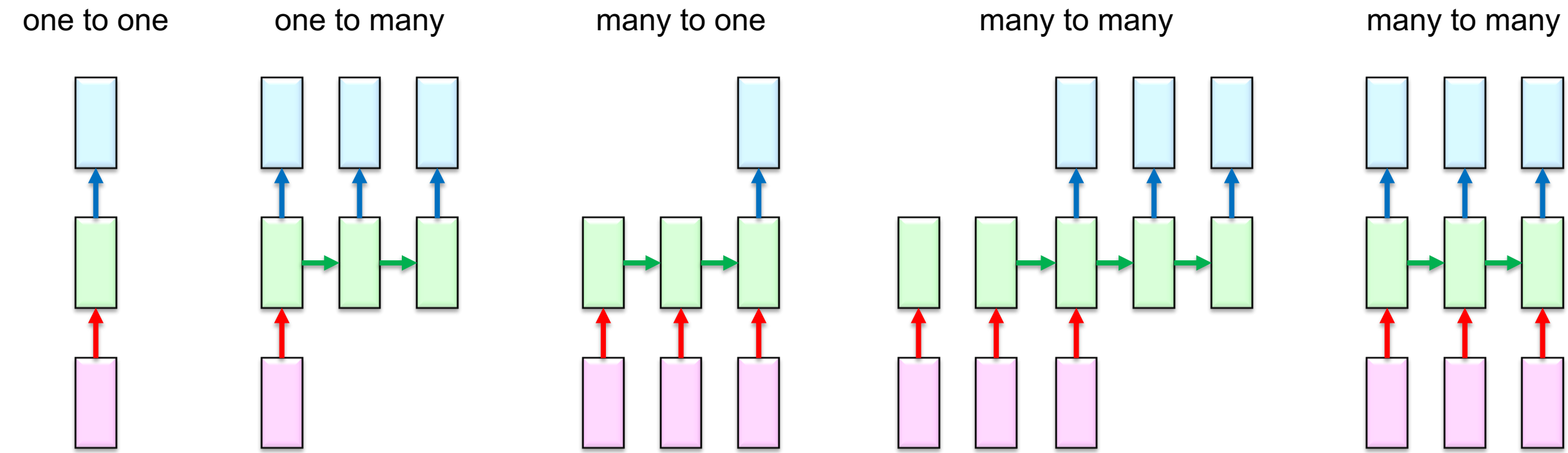


The issue with these methods is that they don't know anything about the English language!

Can we apply large data, transfer learning and deep learning neural networks in this space ??

YES!

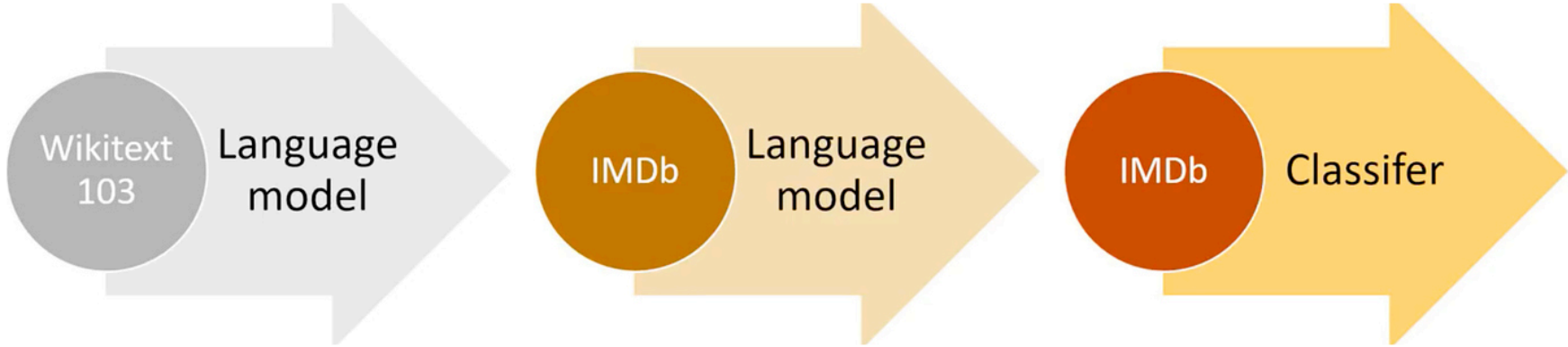
RNN types


one to one
one to many
many to one
many to many
many to many
Vanilla Neural Networks
eg: Image Captioning
Image → sequence of words
eg: Sentiment Classification
sequence of words → sentiment
eg: Machine Translation
seq of words → seq of words
eg: Video classification
on frame level

Transfer Learning with NLP – motivating example

- MovieLens dataset
  - Contains movie ratings data from users
  - 100k ratings
  - The issue here is that there isn't enough data to sufficiently learn language subtleties

- What if we first train NN to "learn" english semantics using a much larger corpus of data (like wikipedia) ?
- Then "fine tune" language model with our data ..
- Then take the knowledge [embeddings] from the language and use for classification

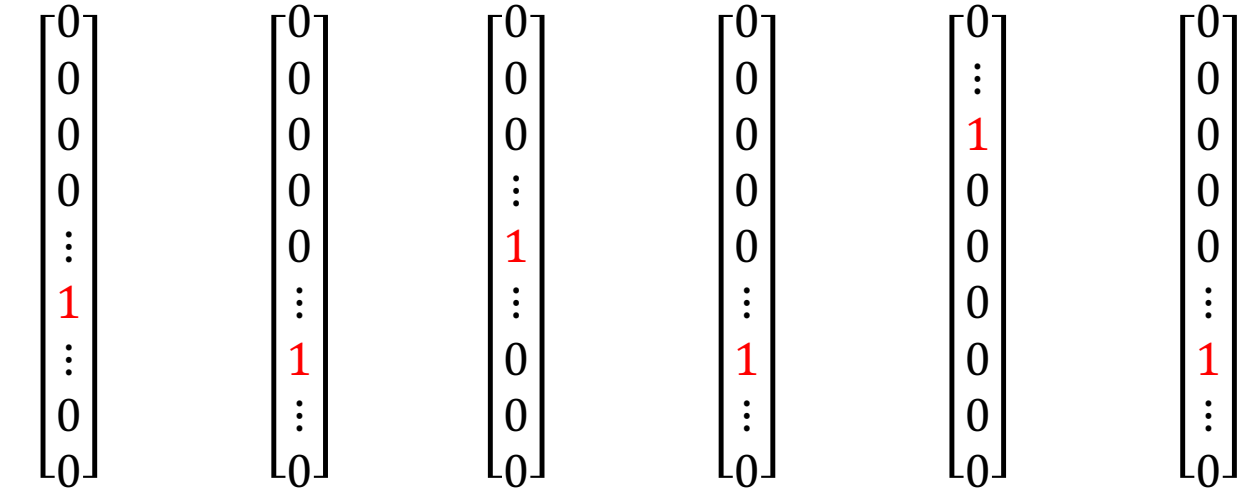


Word representation（One-hot）

- Word Vector
  - Vocabulary includes 10,000 words　　V = [a, aaron, …, zulu, <UNK>]
  - Each word is 10,000-d vector,　One-hot coding,　O_{num}

  - eg :

Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
O_{5391}	O_{9853}	O_{4914}	O_{7157}	O_{456}	O_{6257}



Featurized representation（Word embedding）

- If we can build connections between word and feature, words with the same characteristics are closer and easier to analogy than other words, called “Word embedding”, i.e. e_{num}

Vocabulary size (10000)

Features (300)

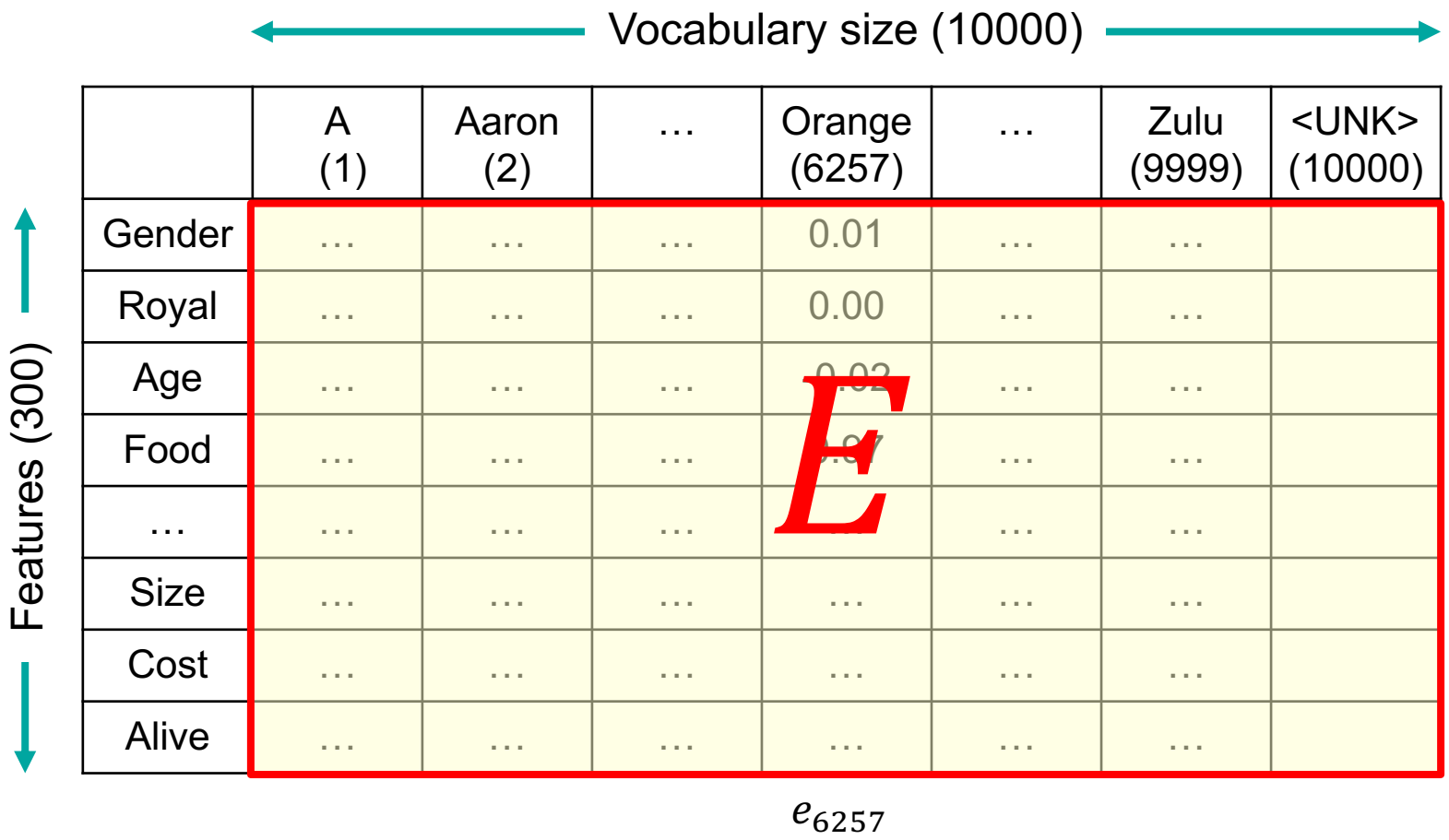
	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97
…	…	…	…	…	…	…
Size	…	…	…	…	…	…
Cost	…	…	…	…	…	…
Alive	…	…	…	…	…	…
	e_{5391}	e_{9853}	e_{4914}	e_{7157}	e_{456}	e_{6257}

I want a glass of apple juice

I want a glass of orange juice

Embedding matrix

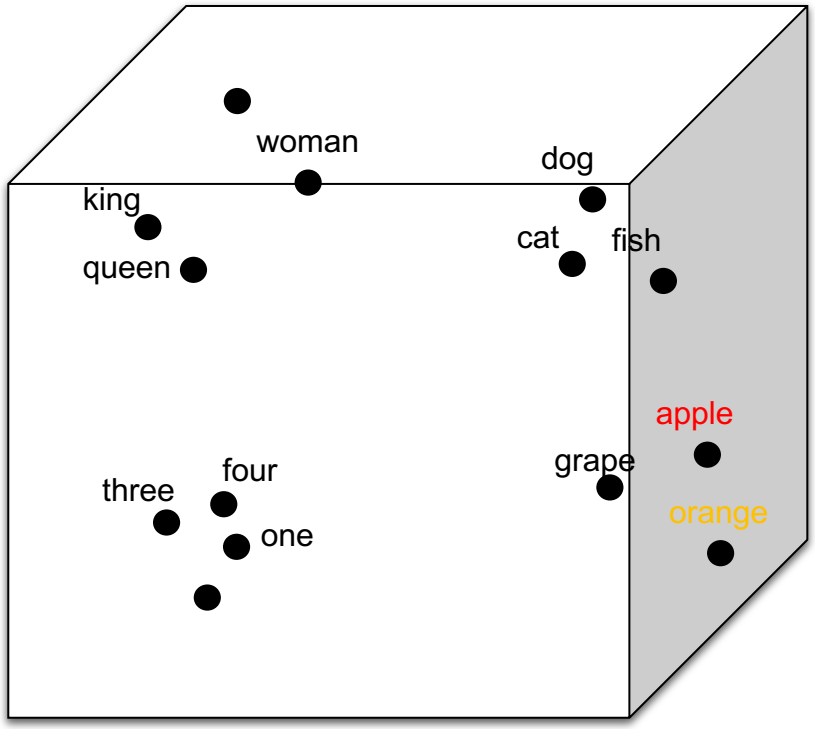
- Learn word embeddings for whole vocabulary, as Embedding Matrix (E)



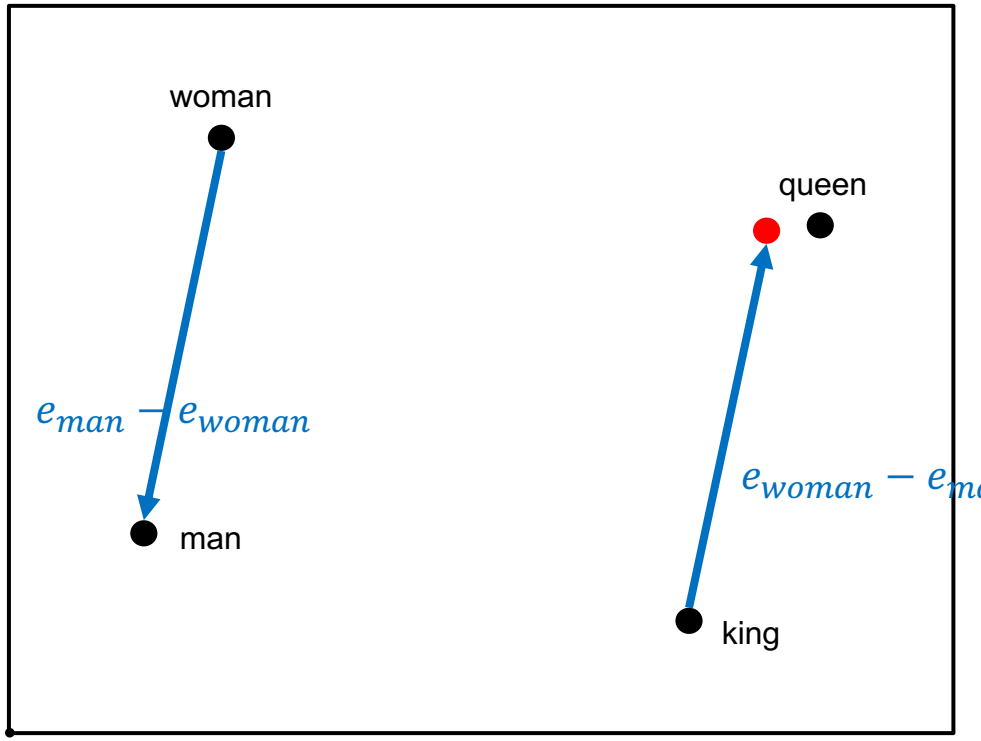
Analogies using word vectors

- Cast 300D embedding vector to 2D by linear transformation, “similar feature” can be present as "parallel vector"
  - $e_{man} - e_{woman} \approx e_{king} - e_{queen}$

- So, the problem transfer to
  - Find a word, which is most similar with $e_{woman} - e_{man} + e_{king}$



N Dimensions



Linear transformation

A few words FastAI

- World class results!

- Incorporates best practices learned from experts (training / simplicity)
- Easy to use examples
- Flexibility to extend to custom problems
- Built on top of PyTorch framework
- Vibrant user community

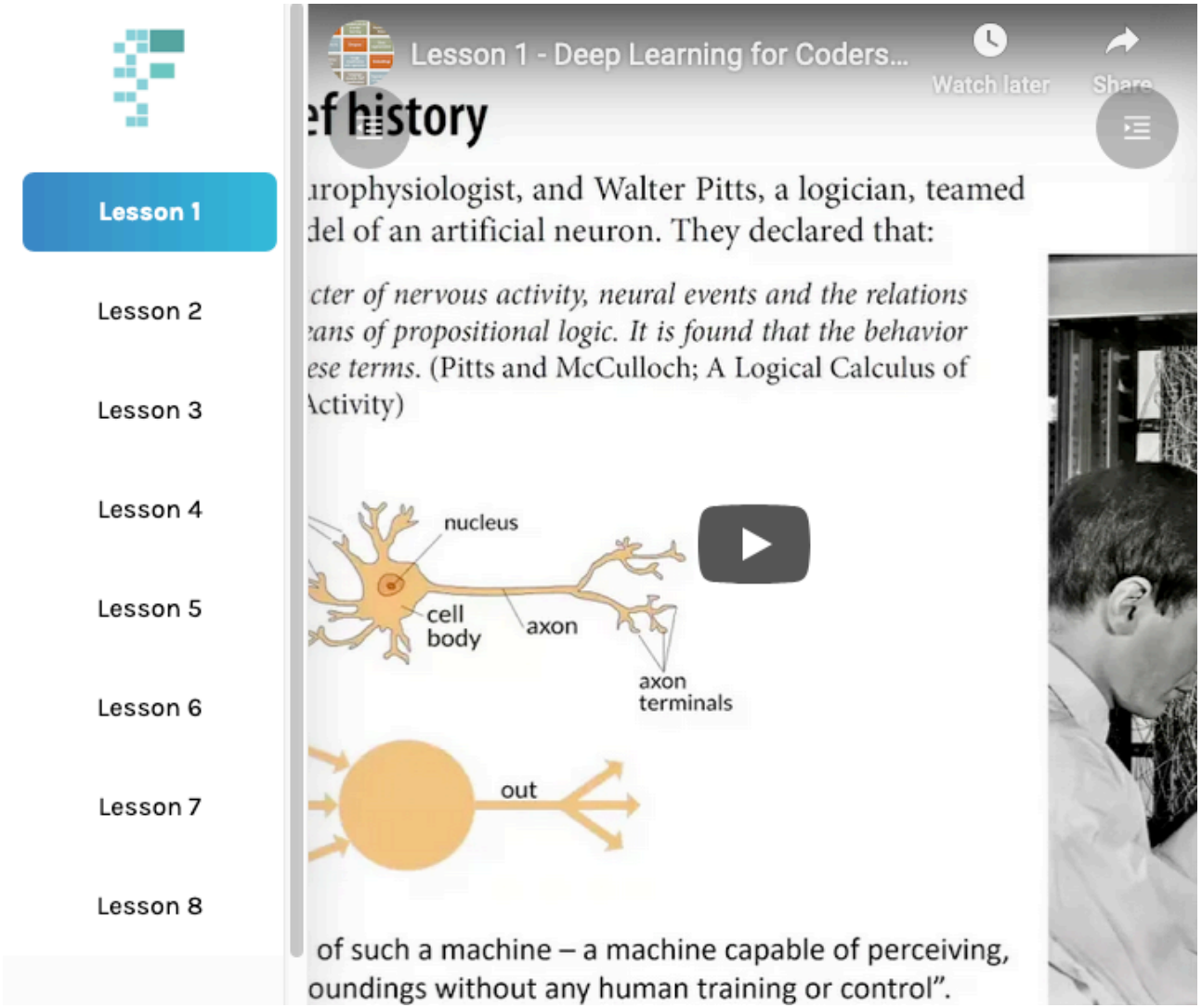
- Excellent educational material
  - videos : https://course.fast.ai/
  - repo : https://github.com/fastai/course-v3

- Supports classification / obj det / tabular / nlp / gans and more!

Dogs vs. Cats

	fastai resnet34*	**fastai** resnet50	Keras
Lines of code (excluding imports)	5	5	31
Stage 1 error	0.70%	0.65%	2.05%
Stage 2 error	0.50%	0.50%	0.80%
Test time augmentation (TTA) error	0.30%	0.40%	N/A*
Stage 1 time	4:56	9:30	8:30
Stage 2 time	6:44	12:48	17:38

FastAI is more than a Python Library



Lessons (Part 1) ▲

1 - Image classification
2 - Production; SGD from scratch
3 - Multi-label; Segmentation
4 - NLP; Tabular data; Recsys
5 - Backprop; Neural net from scratch
6 - CNN deep dive; Ethics
7 - Resnet; U-net; GANs

Lessons (Part 2) ▲

Part 2 overview
8 - Backprop from the foundations
9 - The training in depth
10 - Looking inside the model
11 - Data Block API; generic optimizer
12 - Advanced training; ULMFiT
13 - Swift: Deep Learning Basics
14 - Swift: Putting it all together

Thankyou !

Open–CE discussion and demonstration

Computer Vision Demo : Social Distancing Use Case

Natural Language Demo : Sentiment Analysis using FastAI

Content Links :
https://github.com/dustinvanstee/2020-11-install-opence-fastai

Vision
https://github.com/dustinvanstee/2020-11-wisc
https://github.com/yhenon/pytorch-retinanet

NLP
https://github.com/fastai/fastbook