Applied ML & Big Data Analysis
Final Project
Finding Wally in (2D) Images

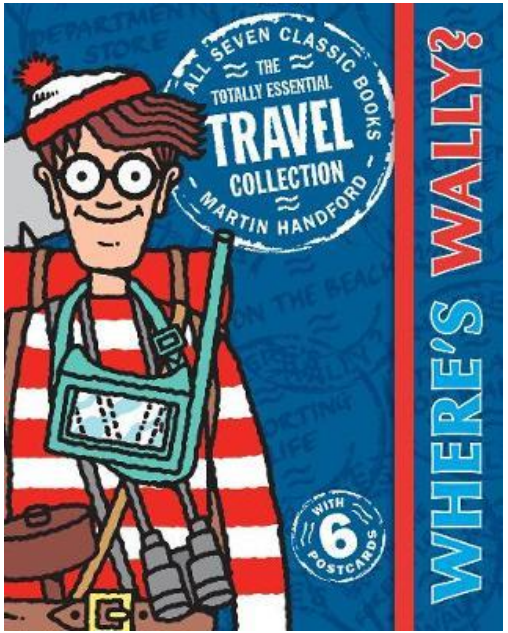
Anastasios M, Kerttu MP, Simon JH, Thomas LT
All group members have contributed evenly to this project

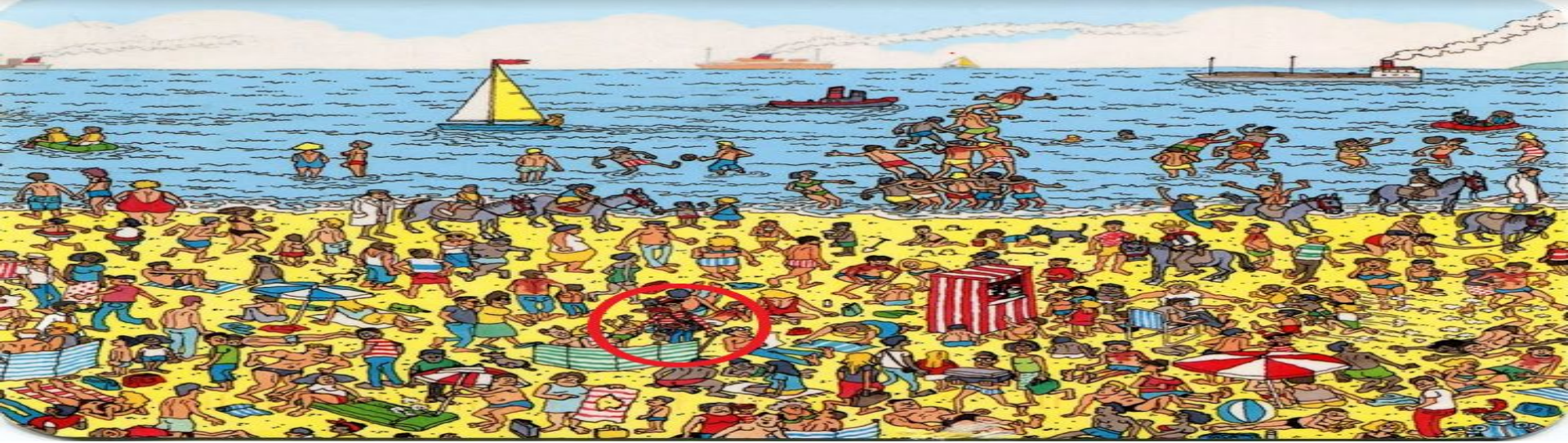
Where's Wally?

A series of children's puzzle books where the reader is challenged to find a character named Wally hidden in a group of illustrated characters.

Our Goal:

Build ML algorithms capable of finding Wally.





Finding Wally is not always an easy task...

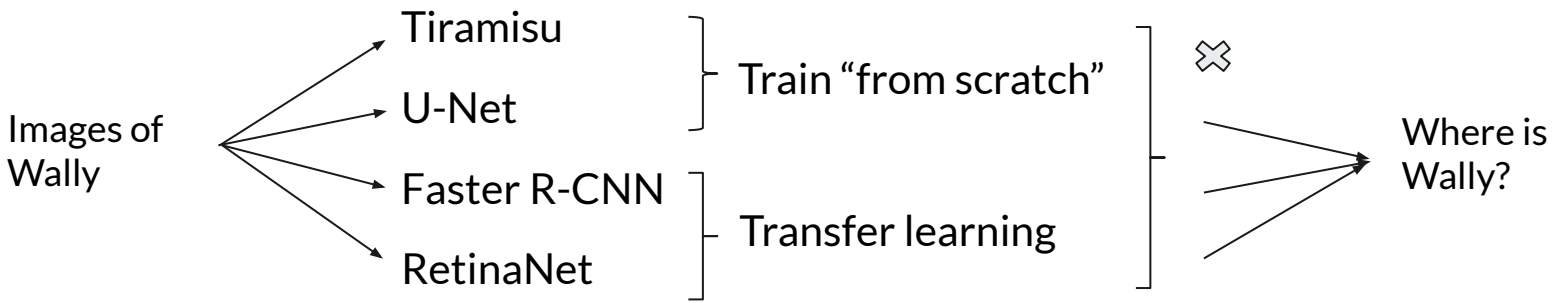
What makes finding Wally difficult?

- Distinguish between Wally and Wally's look-alikes.
- Wally is not always fully visible.
- Wally can get distorted during upscaling /downscaling.
- Class imbalance (1 Wally in dozens of no-Wallys per image).

- Small amount of data (~30 pictures initially).
- Badly annotated data.

Our approach

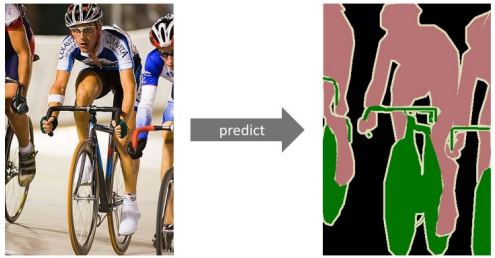
Convolutional Neural Networks for object detection



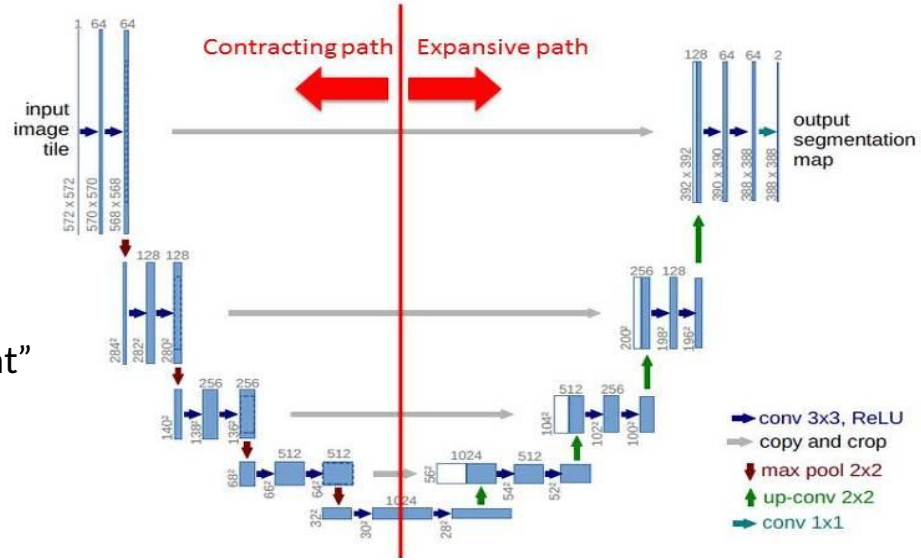
Created our own annotated dataset
(81 images in total / 70 for training / 10 + 1 for testing)

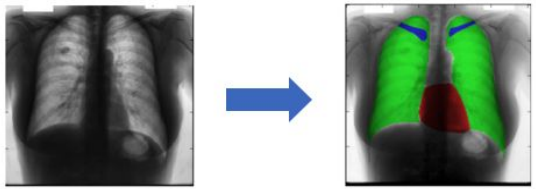
U-Net

- Training “from scratch”
- Semantic segmentation (classification of each pixel)
- Saves the “where” along with the “what”
- First used in medical image analysis



Person
Bicycle
Background





Input Image

Segmented Image

“U-Net: Convolutional Networks for Biomedical Image Segmentation“, May 2015: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

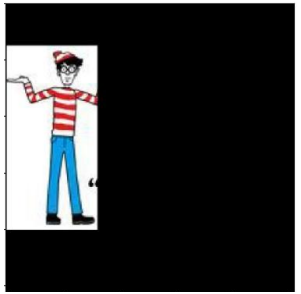
Arxiv: https://arxiv.org/abs/1505.04597

U-Net

First iteration:
Annotated, small dataset

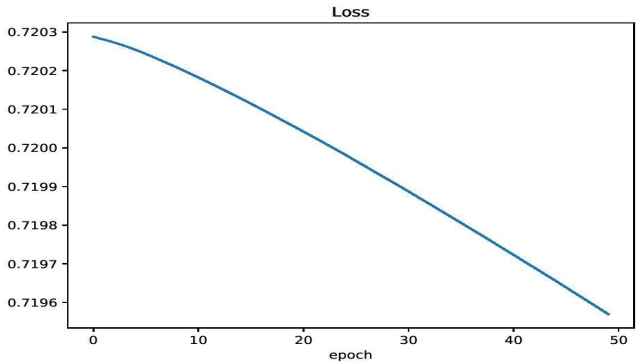
Data

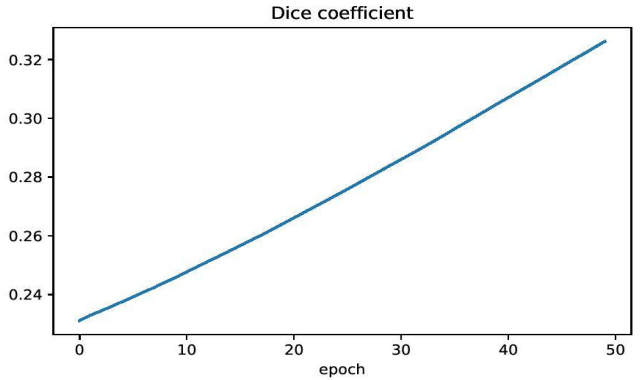
Second iteration:
Bounding boxes, large dataset



U-Net

- Computationally heavy
  - resize to 512x512
  - only use 8 filters
- 50 epochs ~ 10 hours
- Needs a lot more training time





Transfer Learning Approach: Pretrained Models Applied to New Problem

- Initial dataset: COCO (Common Objects in COntext)
- Custom Retraining Dataset: 81 Wally images (70 training, 11 evaluation)

How to deal with complex Wally pictures?

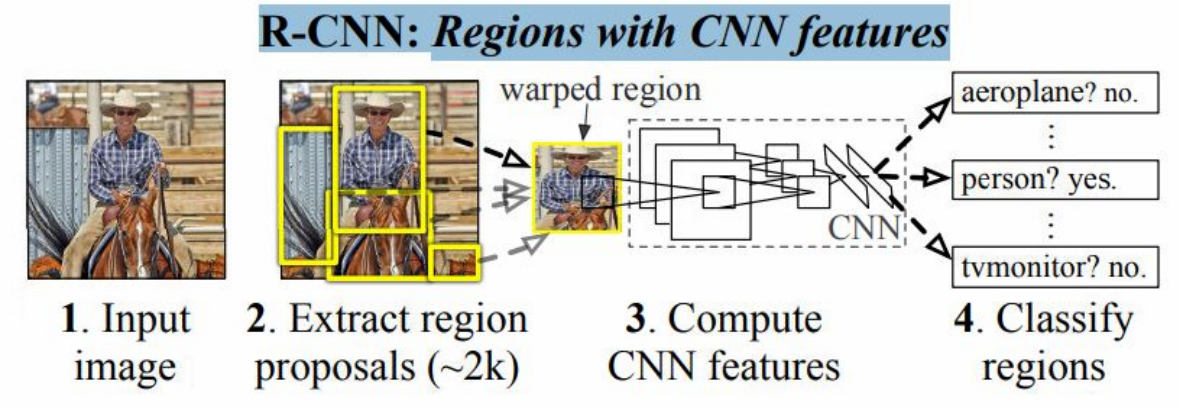
- Data Augmentation: rotation, translation, shear, scaling, horizontal flip, …
- Resizing Input Images for Easier Feature Extraction: 1800x3000

Faster R-CNN Inception v2

- 3rd Generation of R-CNN (Regions with CNN features) Models, i.e. an Object Detection model using Bounding Boxes for detection
- Implemented through **Tensorflow Object Detection API**, run on local GPU

Simple overview of 1st Generation R-CNN, Oct. 2014

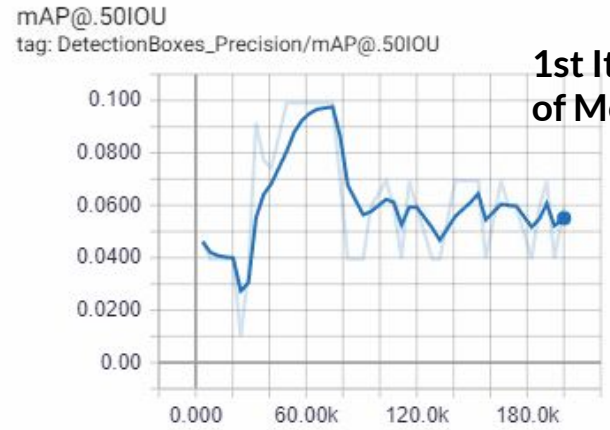
https://arxiv.org/pdf/1311.2524.pdf



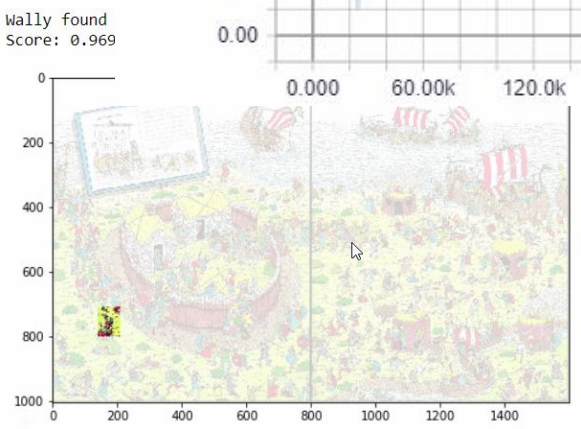
Faster R-CNN Inception v2

- Initial Model: Downloaded from GitHub and applied straight away (trained for 200k steps on 36 images)
  - **Bad at large images, cannot find large Wally**
- 1st Iteration of Model: Retrained base model with new data (trained for 200k steps, 70 images)
  - **Bad at large images, finds large Wally**
- 2nd Iteration of Model: Retrained base model with new params (crashed after 22k steps, 70 images)
  - **Better at large images, finds large Wally**

Example of train/eval. metrics obtained with Tensorboard (here mAP)



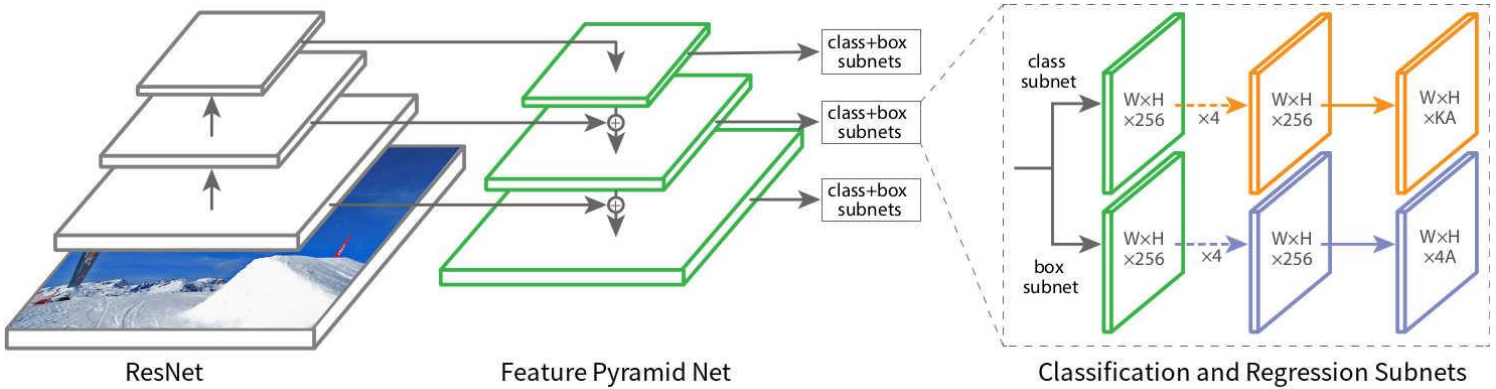
1st Iter. of Model



2nd Iter. of Model -better!

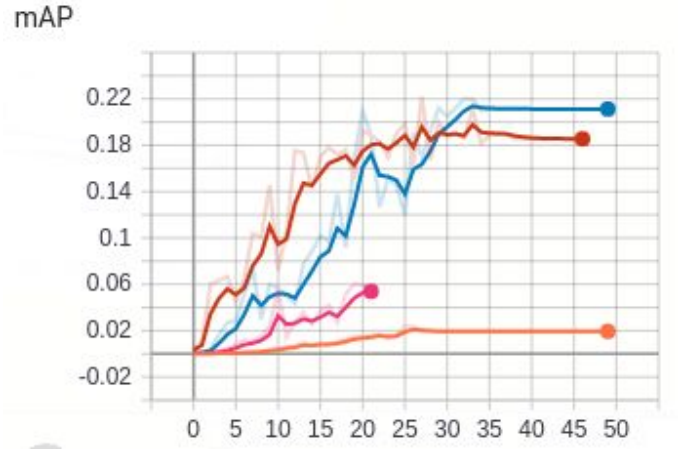
RetinaNet

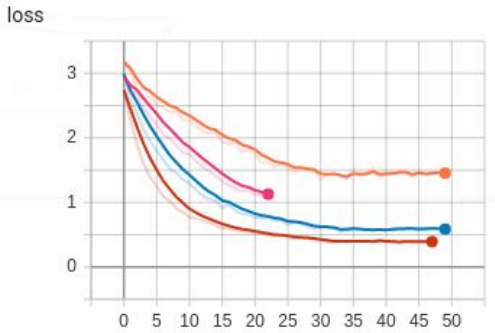
- Architecture: Residual Net (ResNet) and modified Feature Pyramid Net (FPN)
- Object Detection with bounding boxes
- Loss functions: Smooth_L1 loss (regression), focal loss (classification)



RetinaNet

- Run on Google Colab
- 53 training and 19 validation annotations
- 11 test images
- Iterations with 50 epochs and
  - **1000 steps ~ 11.25 hours**
  - **500 steps in ~ 6 hours** (trained model lost due to runtime limitations)
  - **300 steps ~ 1.6 hours** (trained model lost due to runtime limitations)
  - **150 steps ~ 2 hours**





RetinaNet: improvements

- Cross-validation
- Faster GPU for experimenting

Results

Model:	**Evaluation Metric: (Dice Coefficient)**	**Training Time: (Best Predictor)**
Tiramisu	NaN	NaN
U-Net	0.41	~ 10 hours (50 epochs, 3 steps, CPU)
Faster R-CNN	0.45	~ 5 hours (22k steps, 1 batch, GPU)
RetinaNet	0.27	~ 11 hours (50 epochs, 1000 steps, GPU)

```
1
Waldo is not found
```

References

- Data: https://github.com/cparrarojas/find-wally
- Labelling: https://tzutalin.github.io/labelImg/
- Models: https://github.com/fizyr/keras-retinanet; https://towardsdatascience.com/how-to-find-wally-neural-network-eddbb20b0b90; https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/
- Object Detection API: https://www.dlology.com/blog/how-to-train-an-object-detection-model-easy-for-free/?fbclid=IwAR0BMi56uzQx1XKs2frcdKqKlWGoTOkBRukpBnrcTLj18_UWm5eW6xsHKh8
- Pictures for presentation: https://blog.zenggyu.com/en/post/2018-12-05/retinanet-explained-and-demystified/