Target detection

Contributors: V. Nousi, D. Triantafyllidou, A. Tefas, I. Pitas (Aristotle University of Thessaloniki)
Presenter: Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@aiia.csd.auth.gr
www.multidrone.eu
Presentation version 1.1.3

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)
Target detection
Target detection
Target detection

- Target/object examples: athletes, boats, bicycles.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE).
Target detection
Object detection

• Single view object detection
  • Deep learning (CNN) object detection.
  • Light weight CNNs for object detection.
• Multiple view object detection.
Object detection

- Object detection = classification + localization:
- Find **what** is in a picture as well as **where** it is.
Object detection

- **Input**: an image.
- **Output**: bounding boxes containing depicted objects.
  - Each image contains a different number of objects (outputs).
- Typical approach: train a specialized classifier and deploy in sliding-window style to detect all objects of that class.
  - Very inefficient, quite ineffective.
- **Goal**: combine classification and localization into a single architecture for multiple, multiclass object detection.
Classification and Regression

• **Classification:** If we have a class label set $\mathcal{C} = \{c_1, ..., c_L\}$, train a NN model to assign a class label vector $\hat{y} \in [0, 1]^L$ to an object $x$:
  $$\hat{y} = f_{NN}(x, \theta),$$
  where $\theta$ are the CNN trainable parameter vector.
• Essentially, we assign (predict) probabilities $P(\hat{y} | x)$ that an object $x$ belongs to each of the $L$ classes.

• **Training:** Given $N_{\text{training}}$ ground truth pairs $\{x_i, y_i\}, i = 1, ..., N_{\text{training}}$, estimate $\theta$ by minimizing an error function $\min_\theta J(y - \hat{y})$.

• **Testing:** Given $N_{\text{test}}$ ground truth validation pairs $\{x_i, y_i\}, i = 1, ..., N_{\text{test}}$, calculate (predict) $\hat{y}_i, i = 1, ..., N_{\text{test}}$ and calculate a performance metric.
Classification and Regression

• **Classification:**
  • Two class \((L=2)\) and multiple class \((L>2)\) classification.
  • **Example:** *Face detection* (two classes), *face recognition* (many classes).
Classification and Regression

**Regression:** If we have a function $y = f(x)$, train a NN model to predict real-valued quantities (vector $y$ entries), $\hat{y} = f_{NN}(x, \theta)$, so that an error function $\min_{\theta} J(y - \hat{y})$ is minimized.

**Training:** Given $N_{\text{training}}$ ground truth pairs $\{x_i, y_i\}, i = 1, ..., N_{\text{training}}$, estimate $\theta$ by minimizing an error function $\min_{\theta} J(y - \hat{y})$.

**Testing:** Given $N_{\text{test}}$ ground truth validation pairs $\{x_i, y_i\}, i = 1, ..., N_{\text{test}}$ calculate (predict) $\hat{y}_i, i = 1, ..., N_{\text{test}}$ and calculate an error function $J(y - \hat{y})$, e.g. MSE.
Classification and Regression

• Regression:
  • **Example:** In object detection, regress object ROI parameters (width $W$, height $H$, offsets $X$, $Y$).
  • **Function approximation:** it is essentially regression, when the function $y = f(x)$ is known.
Object detection

• **Classification**: train a model to assign a category to an object, i.e., predict a probability of the object belonging to a certain class, i.e. $Pr(class \ | \ object)$.

• **Regression**: train a model to predict **real-valued quantities**, e.g., ROI width $W$, height $H$, $x$ and $y$ offsets.
Object detection performance metrics

- **IoU**: Intersection over Union of predicted ROI (bounding box) $A$ with ground truth ROI $B$:
  \[ J(A, B) = \frac{|A \cap B|}{|A \cup B|} \]
- Also called **Jaccard Similarity Coefficient**.
Object detection performance metrics

- Object detection output: bounding boxes $A_{ij}$ with corresponding confidence scores $s_{ij}$.

- If $A_{ij}$ is matched to a groundtruth box $B_{ik}$, according to $J(A_{ij}, B_{ik}) > T(B_{ik})$, then $z_{ij} = 1$.

- The threshold $T(B_{ik})$ depends on the box size:
  $$T(B_{ik}) = \min(0.5, H \times W / (H+1) \times (W+1)).$$
Object detection performance metrics

- **Recall, Precision** definitions.
- For a confidence threshold $t$ (real number):

  \[
  \text{recall}(t) = \sum_{ij} 1[s_{ij} \geq t]z_{ij}/C, \quad (C \text{ is the number of classes})
  \]

  \[
  \text{precision}(t) = \sum_{ij} 1[s_{ij} \geq t]z_{ij} / \sum_{ij} 1[s_{ij} \geq t].
  \]

- **Mean Average Precision** is calculated over $n=1,\ldots, N$ levels of recall, by varying the confidence threshold:

  \[
  \text{mAP} = 1/N \sum_{n} \text{precision}(t_n).
  \]
Object detection

- **False Positive** (FP) vs **True positive** (TP) plots, as a function of detection threshold e.g., for various training stages.
- The closer the curve is to the upper left corner, the better.
Training

- All approaches treat localization as a regression problem to find \([H, W, X, Y]\) using a CNN.
- All these CNNs utilize a mixed classification + localization loss of the form:

\[
\mathcal{L}(a, I; \theta) = \beta_1 \cdot 1[a \text{ is positive}] \cdot \ell_{loc}(\phi(b_a; a), f_{loc}(I; a, \theta)) + \\
+ \beta_2 \cdot \ell_{cls}(y_a, f_{cls}(I; a, \theta))
\]

\(\beta_1\) and \(\beta_2\) balance the localization and classification losses.

- \(\alpha\) is the best matching ground truth ROI (anchor box) for the detected ROI (box) \(b_\alpha\).
- \(1[\alpha \text{ is positive}]\): indicator vector (vector of ones, if \(\alpha\) matches \(b_\alpha\) with good IoU).
- \(f_{loc}\) is the localization CNN function, \(f_{cls}\) is the classification CNN function.
- \(\ell_{loc}, \ell_{cls}\) are loss functions, e.g., MSE, cross-entropy.
Object detection with CNNs

• **Deep Learning** (DL) approach: train a classifier on, say, 1000 classes of ILSVRC.

• **OverFeat** (2013) was one of the first DL approaches to object detection. Its convolutional method made multi-scale sliding window efficient.

• Based on AlexNet architecture.
Object detection with CNNs

Overfeat: Object detection at increasing image resolutions


This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)
Object detection with CNNs

• Impact of Deep Learning.
• Pascal VOC (object detection)
R-CNN

- R-CNN: Regions with CNN features.

- Three step approach:
  - Extract region proposals using an external proposal method (i.e., Selective Search). Cropped and resized proposed input image regions form crops, always having the same size.
  - Extract CNN features for each crop.
  - Classify features with an SVM.
  - Regress Region Of Interest (ROI) height (H) and width (W) based on the proposed and validated crops.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE).
R-CNN

R-CNN

- **Region selection based on Selective Search:**
  - Alternative to exhaustive, sliding-window search.
  - Based on *region segmentation* techniques.
  - Starting from *over-segmentation*, merge similar regions and produce region proposals.
  - Merged regions are proposed.
R-CNN

Slow R-CNN

Apply bounding-box regressors
Classify regions with SVMs
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

Girshick et al. CVPR14.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)
Fast R-CNN

- R-CNN weaknesses:
  - Use of multiple overlapping proposed regions (crops) of the input image.
    - Too many duplicate computations.
  - Three stage architecture.
Fast R-CNN

- **Fast R-CNN** dealt with these weaknesses:
  - Input image is passed once from a CNN (ConvNet) to generate a **CNN feature map** (big speedup).
  - Selective search is used to generate region proposals, but crops were taken from the CNN feature map, instead of the input image (**ROI pooling**). Crops have always the same size.
  - The pooled features are then fed to the remainder of the network, consisting of **fully connected layers** for classification and ROI H, W refinement through regression.
Fast R-CNN

- **Fast R-CNN weaknesses:**
  - Multiple **overlapping RoIs**
    - duplicate computations.
  - **Externally** computed region proposals (selective search).

Faster R-CNN

- Faster R-CNN: The Region Proposal Network shares layers with the feature extracting network and internally produces region proposals (no selective search).

Faster R-CNN

- The Region Proposal Network (RPN) produces proposals based on an *objectness* score, computed based on the **feature map activations**.
- The feature map is trained using ground-truth objects to produce high objectness score in their ROIs.
- The proposals are used by the RoI pooling and fed into the remainder of the (fully connected) layers for classification.
- Still, *part of the computation depends on the number of proposals*, making it quite slow.
Faster R-CNN

**CNNs used:** VGG16, Inception, Resnet etc


This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE).
R-FCN

• **R-FCN**: Region-based Fully Convolutional Networks.
• Only convolutional layers, thus **fully convolutional**.
• Like **Faster R-CNN**, but crops features from the **last layer prior to prediction** (region classification and refinement).
• This **immensely decreases the per-region computation** that must be done.
SSD

- SSD: Single-Shot Detector.

- **Region-based object detection** (R-CNN, Fast R-CNN, Faster R-CNN, R-FCN): accurate, but **too slow for real-time** applications.

- SSD approach: **Combine a classification network and bounding box regression** into single architecture, without any external steps or duplicated computations.
SSD

• It uses anchors (ROIs of precomputed size and aspect ratio). No region proposals are used.

• Anchors are overlapped at various spatial locations, aspect ratios and scales of the feature maps on various CNN layers.

• During training, anchor location and size are refined via regression to better fit objects.
SSD

- Example: The cat has 2 anchors (ROIs) that match on the 8x8 feature map, but none match the dog. We choose the one having biggest IoU and refine it.
- On the 4x4 feature map there is one anchor that matches the dog and is refined.

• YOLO: You Only Look Once.

• The first version of YOLO was published before SSD, and lacked in precision of localization.
YOLO

• **Simpler YOLO architecture:** Darkenet19 convolutional network plus FC layer.
• Prediction only at the final convolutional feature map.

YOLO

- YOLO divides the input image into an $S \times S$ grid.

- If the center of an object falls within a cell of the grid, that cell is responsible for detecting that object.

- $N$ is the maximal number of bounding boxes that each grid cell can detect.

- Each cell predicts $N$ bounding boxes and confidence and classification scores for those boxes.
YOLO

- **Confidence** is measured as $Pr(\text{Object}) \times \text{IoU(truth, pred)}$, corresponding to a) how confident the model is that the box contains an object and b) how accurate it thinks the predicted box is.
- Each ROI is assigned **five object predicted values**: $H$, $W$, $X$, $Y$ and *confidence*.
- The maximal number of detected objects is $N \times S \times S$, where $N$ is the maximal number of detections per grid cell.
Object Detection

- We evaluated the faster detector (YOLO) on an GPU accelerated embedded system (NVIDIA TX-2) that is available on our drone
- Adjusting the input image size allows for increasing the throughput
- Real-time detection is not yet possible with satisfactory accuracy

<table>
<thead>
<tr>
<th>Model</th>
<th>Input Size</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLO v.2</td>
<td>604</td>
<td>3</td>
</tr>
<tr>
<td>YOLO v.2</td>
<td>544</td>
<td>4</td>
</tr>
<tr>
<td>YOLO v.2</td>
<td>416</td>
<td>7</td>
</tr>
<tr>
<td>YOLO v.2</td>
<td>308</td>
<td>10</td>
</tr>
<tr>
<td>Tiny YOLO</td>
<td>604</td>
<td>9</td>
</tr>
<tr>
<td>Tiny YOLO</td>
<td>416</td>
<td>15</td>
</tr>
</tbody>
</table>
YOLO v2

- **Fully convolutional**, no densely-connected layers:
  - It may be run at varying input sizes.
- It can utilize **multi-scale capabilities** during training as well.
- **Very fast** architecture and implementation.
- Uses **precomputed anchors**.
YOLO v2

• Anchors: precompute a predefined number of anchors (typically 5) by running k-means on all object ROIs of training set, using IoU as the error to minimize.
YOLO v2

- **Input**: Image of arbitrary size $H \times W$ - best to be a multiple of 32, as the network downsamples by 32.
- **Output**: $(H/32) \times (W/32) \times ((C+5)\times N)$
  - $C$ is the number of classes to predict
  - $N$ is the number of precomputed anchors to fit bounding boxes.
  - Depth $d=(C+5)\times N$.
- For one class and five anchors (e.g., bicycles): $(H/32) \times (W/32) \times 30$. 
Using object detectors for drone-based shooting

- Reducing the input image size can also increase the detection speed
- However, this can significantly impact the accuracy when detecting very small objects (which is the case for drone shooting)

<table>
<thead>
<tr>
<th>Model</th>
<th>Input Size</th>
<th>Pascal 2007 test mAP*</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLO v.2</td>
<td>544</td>
<td>77.44</td>
</tr>
<tr>
<td>YOLO v.2</td>
<td>416</td>
<td>74.60</td>
</tr>
<tr>
<td>YOLO v.2</td>
<td>288</td>
<td>67.12</td>
</tr>
<tr>
<td>YOLO v.2</td>
<td>160</td>
<td>48.72</td>
</tr>
<tr>
<td>YOLO v.2</td>
<td>128</td>
<td>40.68</td>
</tr>
</tbody>
</table>

*Using unofficial evaluation code (results might slightly differ)
YOLO v2/ TinyYOLO

- **Input RGB image**: arbitrary size H, W. Preferable sizes: odd multiples of 32. (416x416 = 13x32x32x13 pixels).

- **Output feature map**: H/32xW/32xd=13x13x125 (d=125). Depth d (depends on the number of object classes and number of anchors used).

- **TinyYOLO** has same input/output but half the number of the convolutional layers. Therefore, it is much faster.

![YOLO v2/TinyYOLO Diagram](image)
YOLO v3

• **Deeper ResNet-based architecture:** 53 convolutional layers with skip connections.

• **Multiscale Detection:** detection occurs at multiple layers at different points in the architecture, to detect objects of different scales.

• Much **better mAP**, but **significantly slower**.

• Much **better at detecting small objects**.
RetinaNet

- **RetinaNet**: Dense detection, trained with **Focal Loss**:
  \[
  FL(p_t) = -(1 - p_t)\gamma \log(p_t)
  \]

  where
  \[
  p_t = \begin{cases} 
  p & \text{if } y = 1 \\ 
  1 - p & \text{otherwise}
  \end{cases}
  \]

- Focal Loss forces the model to **focus on hard negative examples**, reducing the significance of the overwhelmingly many easy negative examples.

- Significantly enhances the detection performance, using a **one-stage detector**.
RFBNNet

• Architecture inspired by the structure of **Receptive Fields in human visual systems**.

• Use of **multiple dilated convolutions with different kernel sizes** in each convolutional layer.

• **State-of-the-art** results and **fast** inference time.
Using object detectors for drone-based shooting

- **Fine-tuning a pretrained model** on a new domain (e.g., boat/bicycle detection), instead of training from scratch usually yields better results.
- **Tiny versions of the proposed detectors** (e.g., Tiny YOLO) can increase the detection speed (but at the cost of accuracy).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Positive</th>
<th>Negative</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crowd</td>
<td>40000</td>
<td>20000</td>
<td>20000</td>
<td>11550</td>
</tr>
<tr>
<td>Football</td>
<td>80000</td>
<td>40000</td>
<td>40000</td>
<td>10000</td>
</tr>
<tr>
<td>Bicycles</td>
<td>51200</td>
<td>25600</td>
<td>25600</td>
<td>7000</td>
</tr>
<tr>
<td>Face</td>
<td>140000</td>
<td>70000</td>
<td>70000</td>
<td>7468</td>
</tr>
</tbody>
</table>
Network parameters

- **Any feature extractor** can be used for the feature extraction step of detection:
  - VGG16
  - ResNet, ResNet-101
  - Inception, Inception V2, V3
  - MobileNets.
- **MobileNets** were recently introduced by Tensorflow as a lightweight alternative:
  - The standard convolution is replaced by depth-wise separable convolutions, reducing the number of parameters and FLOPs.
MobileNets

- **Standard convolutional filters:**
  - $N \times D_k \times D_k$ filters, depth $M$ ($M=3$ for RGB images).

- **Depth-wise separable convolutional filters (MobileNet):**
  - $M \times D_k \times D_k$ filters of depth 1, one per input channel ($M=3$ for RGB images) and
  - $N \times 1 \times 1$ filters (essentially weighted averaging of the $M$ channels produced by the previous step).

CNN comparison


This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)
CNN comparison


This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE).
Number of Region Proposals


This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE).
CNN comparison

• Faster R-CNN is more accurate but slower.

• YOLO, SSD are much faster but not as accurate.

• YOLO, SSD make more mistakes when objects are small and have trouble correctly predicting the exact location of such objects.
Object detection acceleration

- Examples of acceleration techniques:
  - Input size reduction.
  - Specific object detection instead of multi-object detection.
  - Parameter reduction.
  - Post-training optimizations with TensorRT (NVIDIA), including FP16 (floating point 16 bit) computations.
Object detection on NVIDIA jetson TX2

- YOLO: good precision in general, but too heavyweight:
  - small objects are more challenging.
- Evaluation on VOC:

<table>
<thead>
<tr>
<th>Input Image Size</th>
<th>FPS</th>
<th>mAP</th>
<th>Forward time (ms) No TensorRT</th>
<th>Forward time (ms) TensorRT</th>
<th>Forward time (ms) FP16</th>
</tr>
</thead>
<tbody>
<tr>
<td>608x608</td>
<td>2.9</td>
<td>71.26</td>
<td>241.5</td>
<td>128.8</td>
<td>69.3</td>
</tr>
<tr>
<td>544x544</td>
<td>3.2</td>
<td>73.64</td>
<td>214.4</td>
<td>121.2</td>
<td>64.3</td>
</tr>
<tr>
<td>480x480</td>
<td>5.4</td>
<td>74.50</td>
<td>155.4</td>
<td>62.3</td>
<td>35.7</td>
</tr>
<tr>
<td>416x416</td>
<td>6.4</td>
<td>73.38</td>
<td>155.3</td>
<td>56.5</td>
<td>32.5</td>
</tr>
<tr>
<td>352x352</td>
<td>7.8</td>
<td>71.33</td>
<td>111.0</td>
<td>45.0</td>
<td>24.3</td>
</tr>
<tr>
<td>320x320</td>
<td>8.5</td>
<td>70.02</td>
<td>103.0</td>
<td>40.4</td>
<td>22.8</td>
</tr>
</tbody>
</table>
Object detection on NVIDIA jetson TX2

- Tiny YOLO: low precision, but very lightweight.
- Evaluation on VOC:

<table>
<thead>
<tr>
<th>Input Image Size</th>
<th>FPS</th>
<th>mAP</th>
<th>Forward time (ms) No TensorRT</th>
<th>Forward time (ms) TensorRT</th>
<th>Forward time (ms) FP16</th>
</tr>
</thead>
<tbody>
<tr>
<td>608x608</td>
<td>6.5</td>
<td>51.28</td>
<td>76.5</td>
<td>37.5</td>
<td>22.1</td>
</tr>
<tr>
<td>544x544</td>
<td>8.2</td>
<td>52.93</td>
<td>68.4</td>
<td>34.8</td>
<td>20.5</td>
</tr>
<tr>
<td>480x480</td>
<td>13.4</td>
<td>55.00</td>
<td>50.1</td>
<td>17.2</td>
<td>11.7</td>
</tr>
<tr>
<td>416x416</td>
<td>16.5</td>
<td>56.28</td>
<td>49.9</td>
<td>15.7</td>
<td>10.3</td>
</tr>
<tr>
<td>352x352</td>
<td>20</td>
<td>55.05</td>
<td>37.1</td>
<td>13.0</td>
<td>7.9</td>
</tr>
<tr>
<td>320x320</td>
<td>23</td>
<td>53.81</td>
<td>34.0</td>
<td>11.7</td>
<td>7.2</td>
</tr>
</tbody>
</table>
Object detection on NVIDIA jetson TX2

- SSD: generally good precision, not as fast, still prone to mistakes when objects are small.
- MobileNets and Inception V2 can be used as feature extractors to provide speed ups.
- Tensorflow Implementation (subject to Tensorflow’s memory mishandling).
## Object detection on NVIDIA jetson TX2

### MobileNets

<table>
<thead>
<tr>
<th>Input Image Size</th>
<th>FPS</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>300x300</td>
<td>11.6</td>
<td>84.1</td>
</tr>
<tr>
<td>224x224</td>
<td>13.4</td>
<td>81.1</td>
</tr>
<tr>
<td>192x192</td>
<td>18.2</td>
<td>80.0</td>
</tr>
<tr>
<td>160x160</td>
<td>21.0</td>
<td>77.4</td>
</tr>
<tr>
<td>128x128</td>
<td>23.8</td>
<td>71.4</td>
</tr>
</tbody>
</table>

### Inception V2

<table>
<thead>
<tr>
<th>Input Image Size</th>
<th>FPS</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>300x300</td>
<td>8.5</td>
<td>85.2</td>
</tr>
<tr>
<td>224x224</td>
<td>12.3</td>
<td>84.0</td>
</tr>
<tr>
<td>192x192</td>
<td>13.8</td>
<td>81.1</td>
</tr>
<tr>
<td>160x160</td>
<td>15.6</td>
<td>76.6</td>
</tr>
<tr>
<td>128x128</td>
<td>17.1</td>
<td>73.6</td>
</tr>
</tbody>
</table>
Object detection comparisons

- SSD w/ MobileNets and Inception V2 for various input image sizes in Face Detection in FDDB facial data base. Curves are created by changing the confidence threshold.
- (Face recall in parentheses).
Object detection

• Single view object detection
  • Deep learning (CNN) object detection.
  • Light weight CNNs for object detection.
• Multiple view object detection.
Object detection

- **State-of-the-art** object detectors (YOLO, SSD, etc) are based on **very Deep** and **multiple-channel CNNs**.

- **Light weight** architectures can provide equally satisfactory results.

- Such architectures are trained with incremental positive and negative example mining methods.
Lightweight Approach to Object Detection

• Our approach: train **lightweight fully convolutional object-specific** (e.g., face, bicycle, football player) detectors
  
  • e.g., for face detection we trained a **7-layer fully convolutional** face detector on 32 × 32 positive and negative examples [1]
  
  • During **deployment on larger images** the network very **efficiently produces a heatmap** indicating the probability of a face as well as its location in the image

Lightweight Approach to Object Detection

- **Domain-specific** knowledge may be exploited to train such lightweight object detector for specific events
- e.g., for cycling races, train detector to recognize professional bicycles
Object detection

- **Light weight** deep CNN architecture (~76K parameters).
- The network is **fully convolutional**.
- Input 32x32 pixel RGB image. Output d=2 (2 classes).

<table>
<thead>
<tr>
<th>layer</th>
<th>kernel</th>
<th>filters</th>
<th>input</th>
<th>output</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>3 x 3</td>
<td>24</td>
<td>32 x 32 x 3</td>
<td>30 x 30 x 24</td>
<td>648</td>
</tr>
<tr>
<td>prelu1</td>
<td></td>
<td></td>
<td>30 x 30 x 24</td>
<td>30 x 30 x 24</td>
<td>24</td>
</tr>
<tr>
<td>conv2</td>
<td>4 x 4</td>
<td>24</td>
<td>30 x 30 x 24</td>
<td>14 x 14 x 24</td>
<td>9216</td>
</tr>
<tr>
<td>prelu2</td>
<td></td>
<td></td>
<td>14 x 14 x 24</td>
<td>14 x 14 x 24</td>
<td>24</td>
</tr>
<tr>
<td>conv3</td>
<td>4 x 4</td>
<td>32</td>
<td>14 x 14 x 24</td>
<td>11 x 11 x 32</td>
<td>12288</td>
</tr>
<tr>
<td>prelu3</td>
<td></td>
<td></td>
<td>11 x 11 x 32</td>
<td>11 x 11 x 32</td>
<td>32</td>
</tr>
<tr>
<td>conv4</td>
<td>4 x 4</td>
<td>48</td>
<td>11 x 11 x 32</td>
<td>8 x 8 x 48</td>
<td>24576</td>
</tr>
<tr>
<td>prelu4</td>
<td></td>
<td></td>
<td>8 x 8 x 48</td>
<td>8 x 8 x 48</td>
<td>48</td>
</tr>
<tr>
<td>conv5</td>
<td>4 x 4</td>
<td>32</td>
<td>8 x 8 x 48</td>
<td>5 x 5 x 32</td>
<td>24576</td>
</tr>
<tr>
<td>prelu5</td>
<td></td>
<td></td>
<td>5 x 5 x 32</td>
<td>5 x 5 x 32</td>
<td>32</td>
</tr>
<tr>
<td>conv6</td>
<td>3 x 3</td>
<td>16</td>
<td>5 x 5 x 32</td>
<td>3 x 3 x 16</td>
<td>4608</td>
</tr>
<tr>
<td>prelu6</td>
<td></td>
<td></td>
<td>3 x 3 x 16</td>
<td>3 x 3 x 16</td>
<td>16</td>
</tr>
<tr>
<td>conv7</td>
<td>3 x 3</td>
<td>2</td>
<td>3 x 3 x 16</td>
<td>1 x 1 x 2</td>
<td>288</td>
</tr>
</tbody>
</table>
Object detection

• Detection with **Light weight** deep CNNs.
Object detection

• Detection with **Light weight** deep CNNs.
Object detection

• Detection with **Light weight fully** CNNs.
• The network is trained with **32x32** pixel training samples.
• It is **fully convolutional** and accepts images of **arbitrary** size.
• The network outputs a classification heatmap containing probability scores for each **32x32** pixel region of the input.
Object detection

• Detection with Light weight deep CNNs.
• The method develops an image pyramid representation of varying resolutions and performs detection at multiple scales.
Object detection

• Output heat maps are thresholded at the various pyramid layers.
• The image ROIs of size 32x32xI are created (I: pyramid level).
• The ROI parameters $[X,Y,H,W]$ are clustered to create the final object ROIs.
Object detection

- Evaluation of the model on a bicycle benchmark of RAI videos.
- True positive rate and False positive rate are evaluated for:
  - Different heatmap thresholds,
  - **Hard negative/positive mining**: increased training examples.
Object detection

- **Test execution time** for a 32x32 pixel object ROI of the proposed CNN architecture using NVIDIA’s tensorRT library (in msec):

<table>
<thead>
<tr>
<th></th>
<th>NVIDIA TX2</th>
<th>GeForce GTX 1080</th>
</tr>
</thead>
<tbody>
<tr>
<td>tensorRT</td>
<td>0.491933</td>
<td>0.652406</td>
</tr>
<tr>
<td>no tensorRT</td>
<td>2.84615</td>
<td></td>
</tr>
</tbody>
</table>
Face detection examples
Face detection examples
Face detection examples
Bicycle detection
Bicycle detection
Football player detection
Boat detection
Combining Detectors with Trackers on Drones

• The deployed detector can be combined with fast trackers to achieve satisfactory real-time performance.
• The detector can be called only a few times per second, while the used tracker provides the “detections” in the intermediate frames.
• We evaluated several trackers on the NVIDIA TX-2:

<table>
<thead>
<tr>
<th>Model</th>
<th>Device</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASMS [1]</td>
<td>CPU</td>
<td>81</td>
</tr>
<tr>
<td>STRUCK [2]</td>
<td>CPU</td>
<td>7</td>
</tr>
<tr>
<td>THUNDERSTRUCK [2]</td>
<td>GPU</td>
<td>100</td>
</tr>
<tr>
<td>GOTURN [3]</td>
<td>GPU</td>
<td>30</td>
</tr>
</tbody>
</table>

Object detection

- Single view object detection
  - Deep learning (CNN) object detection.
  - Light weight CNNs for object detection.
- Multiple view object detection.
Multiview human detection

• Problem statement: Use information from multiple cameras to detect bodies or body parts, e.g. head.

• Applications:
  • Human detection/localization in postproduction.
  • Matting/segmentation initialization.

Camera 4

Camera 6
Multiview human detection

• Head or body detection in two stages:
  • Use a head/face/body detector to derive ROIs in each view separately.
  • Insert these ROIs to an algorithm utilizing 3D information.

• Use of camera calibration parameters.
Multiview human detection

- Output: a rectified set of ROIs for each view that contains:
  - fewer false positives;
  - fewer false negatives
    - especially those due to occlusion are eliminated;
  - associations across views
    - all ROIs corresponding to the same human head/body are associated.
Multiview human detection

• Detected ROIs are projected back in the 3D space.
• A “probability volume” is created collecting “votes” from individual ROIs.
• High probability voxels correspond to the most probable head/body VOIs.
Multiview human detection

- The retained voxels are projected to all views.
- For every view we reject ROIs that have small overlap with the regions resulting from the projection.

Camera 2

Camera 4
Multiview human detection

• ROI association across different views:
  • A voting scheme is used to find ROIs across views containing projections of the same voxels.
  • These ROIs are associated across views.
  • ROIs that are not associated are rejected.

• Further elimination of false positives may be achieved.
Multiview human detection

Camera 2

Camera 4

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)
Multiview human detection

• ROI rectification:
  • Using 3D information we create ROIs for a certain head/body in views lacking in them.
  • False negatives elimination.
Multiview human detection

After ROI Rectification
Q & A

Thank you very much for your attention!

Contact: Prof. I. Pitas
pitas@aiia.csd.auth.gr
www.multidrone.eu