Deep Learning for Drone Vision in Cinematography

Anastasios Tefas
tefas@aiia.csd.auth.gr

Contributors: P. Nousi, N. Passalis, D. Triantafyllidou, M. Tzelepi

Artificial Intelligence and Information Analysis Laboratory
Department of Informatics
Aristotle University of Thessaloniki
Introduction
Convolutional Neural Networks

• Deep Convolutional Networks (CNNs) [1] are among the state-of-the-art techniques for Visual Information Analysis

Convolutional Neural Networks

- Composed of a series of convolutional and pooling layers
- Usually a fully connected layer is used for classification
  - Fully convolutional architectures do exist!
- Capable of learning hierarchies of increasingly abstract visual features (from simple edges to object parts and concepts)
Assisting Cinematography Tasks

- Unmanned Aerial Vehicles (UAVs), also known as drones, are becoming increasingly popular for video shooting tasks
  - Flexible!
  - They can capture spectacular shots!
Assisting Cinematography Tasks

- Flying drones in a professional shooting setting requires the coordination of several people
  - One for controlling the flight path of each drone
  - One for controlling the main shooting camera of each drone
  - At least one director, technician, etc.
Assisting Cinematography Tasks

• Several parts of the **shooting process can be automated**, reducing the load of human operators

• **Goal**: One human controls multiple drones
Shooting Pipeline

• A drone must be able to **quickly identify whether one or more objects of interest exist in a scene**
  • Apart from the main drone camera, multiple smaller resolution cameras might be also available to aid this task
Shooting Pipeline

• Decide whether the detected humans are part of the crowd or are persons of interest (e.g., cyclists)
  • The drone must flight away from the crowd
  • The drone must follow the detected persons of interest
Shooting Pipeline

- After detecting a person of interest the camera must be appropriately rotated toward the person of interest
  - Different shot types have different specifications
  - We need the position of the person w.r.t. the camera and its pose
Assisting Cinematography Tasks

• Several (quite demanding) subtasks!
• **Detect** whether and where an object of interest exist (cyclist, boat, monuments, etc.)
Assisting Cinematography Tasks

- **Track** a detected (or selected) object
Assisting Cinematography Tasks

- Detect where crowd exists
  - Comply with legislation
  - Detect emergency landing points
  - Provide **heatmaps** of the estimated probability of crowd presence in each location
Assisting Cinematography Tasks

- Detect where crowd exists
Assisting Cinematography Tasks

- **Identify** a detected person (e.g., a well-known cyclist)
Assisting Cinematography Tasks

- **Estimate the pose** of a detected object
  - Allows for appropriately controlling the camera according to the specifications of each shot type (e.g., orbit around a target or acquire a profile shot)
Assisting Cinematography Tasks
• Camera control was traditionally handled as a purely geometric problem
• We can also perform **camera control** using only visual information
Assisting Cinematography Tasks

- The aforementioned tasks can be solved using deep Convolutional Neural Networks (CNNs)
- Deploying a deep CNN on a drone is not straightforward
- Significant memory and processing power constraints exist
  - State-of-the-art CNNs, such as the VGG-16, consist of hundreds of millions parameters making them unsuitable for handling real-time tasks on-board.
Assisting Cinematography Tasks

- Light-weight models are needed!
- Slight delays can result to control lag
- Different illumination conditions can affect the performance of the models
  - Training set augmentation!
Object Detection and Tracking
Object Detection

- **Faster R-CNN** [1]
  - Region-based object detectors are plagued by inefficient external **region proposal** schemes
  - Key idea: **Utilize CNN feature maps** for both detection and region proposal in a fully convolutional network
  - Also assumes prior "anchor" boxes, and region proposals are **fine-tuned anchor boxes**

Object Detection

- **Faster R-CNN** [1]
  - Trained using a double objective of *classification loss* plus *bounding box regression loss*
  - **Precision** increases with number of region proposals but...
  - ... at 5fps on a K40 GPU, it is among the **slowest** deep object detectors

Object Detection

- **Single Shot Detection (SSD)** [1]
  - Fully convolutional object detection at multiple scales
  - **Predicts center x, y** coordinates for multiple objects as well as their class
  - **Several feature maps of different resolutions** are used for the final prediction
  - Adjusts priors on bounding boxes instead of outright predicting the width and height

Object Detection

- **Single Shot Detection (SSD)** [1]
  - Performs hard negative examples mining, thus not all unannotated regions are considered as negatives
  - On an NVIDIA Titan X: **46fps for the 300x300 version, 19fps for the 512x512 version**

---

Object Detection

- You Only Look Once (YOLO v2) [1] [2]
  - Fully convolutional object detection at multiple scales
  - Predicts center x, y coordinates for multiple objects as well as their class
  - Adjusts priors on bounding boxes instead of outright predicting the width and height
  - All unannotated regions in the input image are considered as negative examples
  - On an NVIDIA Titan X: 67fps for the 416x416 version, 40fps for the 544x544 version

Object Detection

- We evaluated the faster detector (YOLO) on an GPU accelerated embedded system (NVIDIA TX-2) that will be 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>
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).
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)*
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 $32 \times 32$ 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

Face detection examples
Face detection examples
Face detection examples
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
Bicycle detection
- Bicycle detection
Football player detection
Limitations

• **Speed vs accuracy** trade-off:
  • lightweight models don’t perform as well as heavier architectures (think YOLO and tiny YOLO variant)
  • in our approach, accuracy is increased by the use of **domain-specific object detectors**
  • as well as a strategic training methodology of **progressive positive and hard negative mining**, which mimics the natural learning process
Limitations

- **Train with fixed size images:**
  - detection of larger or smaller objects requires the forward-pass of a spatial pyramid of the input
  - which is made **efficient through the fully convolutional architecture** of the detectors
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
In the recent Visual Object Tracking (VOT) challenges CNNs have been taken up the first places in terms of either speed or accuracy.

In VOT2015 [1], the two best scoring trackers, MDNet and DeepSRDCF, were both based on CNNs.

Another CNN-based tracker, SODLT, also achieved a high performance on the benchmark.

However, precise tracking comes at the cost of slow models with online updates or heavy architectures.

But they certainly served to show CNNs can be effectively used in tracking tasks.

Object Tracking

- In VOT2016 [1], eight of the submitted trackers were CNN-based and another six combined convolutional features with Discriminative Correlation Filters
- Five out of the top ten ranked trackers were CNN-based
- The winner of the challenge, C-COT is based on a VGG-16 architecture and computes convolutions in continuous space via learnable, implicit interpolation
- Of the runner ups, Siam-FC is a somewhat lighter CNN-based model which deployed a learnable correlation layer to measure the similarity between target and various candidates in a fully convolutional fashion

Fully Convolutional Image Segmentation
Crowd Detection for Safe Autonomous Drones

• There are limited previous efforts on crowd detection, using computer vision techniques
• Related research works involving crowds, e.g., crowd understanding, crowd counting, and human detection and tracking in crowds, consider crowded scenes
Crowd Detection for Safe Autonomous Drones

- State-of-the-art approaches on crowd analysis utilize deep learning techniques
  - In [1] an effective Multi-column Convolutional Neural Network architecture is proposed to map the image to its crowd density map
  - In [2] a switching convolutional neural network for crowd counting is proposed, aiming to leverage the variation of crowd density within an image

Crowd Detection for Safe Autonomous Drones

Deep Detectors on Drones

- In [1] a **Fully Convolutional Model** for crowd detection is proposed
  - Complies with the computational requirements of the crowd detection task
  - Allows for handling input images with arbitrary dimension
- **Subspace learning inspired Two-loss Convolutional Model**
  - Softmax Loss preserves between class separability
  - Euclidean Loss aims at bringing the samples of the same class closer to each other

Deep Detectors on Drones
Deep Detectors on Drones
Deep Detectors on Drones
Deep Detectors on Drones
Reducing the Complexity of CNNs
Convolutional Bag-of-Features Pooling

- **Global polling** (GMP [1], SPP [2]) techniques can be used to reduce the size of the fully connected layers and allow the network handle arbitrary sized images.
- A Bag-of-Features-based approach was used to provide a trainable **global pooling layer** that is capable of:
  - reducing the size of the model,
  - increasing the feed-forward speed,
  - increasing the accuracy and the scale invariance,
  - adjust to the available computational resources.

---

Convolutional Bag-of-Features Pooling

- The whole network, including the proposed layer, is optimized end-to-end towards the task at hand.
- The proposed method can be readily implemented with the existing Deep Learning Frameworks (Tensorflow, Caffe, PyTorch, etc.)
Convolutional Bag-of-Features Pooling

More information can be found in our paper “Learning Bag-of-Features Pooling for Deep Convolutional Neural Networks” (ICCV 2017, Friday, poster session 8)
Convolutional Bag-of-Features Pooling

- The method was evaluated on a **pose estimation task**
  - Estimate the pose (yaw, pitch, roll) of the main actors (e.g., cyclists, boats, etc)
  - Allows for appropriately controlling the camera according to the specifications of each shot type (e.g., orbit around a target or profile shot)
Convolutional Bag-of-Features Pooling

- Use an object detector to **locate** and **crop** the object.
- Train a CNN to **directly regress the pose** of the cropped object.
- Advantages:
  - No need for 3D models (only a training set of pose-annotated objects are needed).
  - More robust to variations of the object (especially if the training set is appropriately augmented).
Convolutional Bag-of-Features Pooling

- Comparing the proposed pooling technique to other state-of-the-art techniques (ALFW dataset)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>14.50</td>
<td>(13.91*)</td>
<td>61,954k</td>
</tr>
<tr>
<td>GMP</td>
<td>13.70</td>
<td>12.83</td>
<td>514k</td>
</tr>
<tr>
<td>SPP</td>
<td>12.37</td>
<td>11.96</td>
<td>2,562k</td>
</tr>
<tr>
<td>CBoF (0, 128)</td>
<td>16.23</td>
<td>13.84</td>
<td>196k</td>
</tr>
<tr>
<td>CBoF (1, 32)</td>
<td>11.92</td>
<td>11.25</td>
<td>196k</td>
</tr>
</tbody>
</table>

The first number in the CBoF technique indicates the spatial segmentation level.
Convolutional Bag-of-Features Pooling

- Demonstrating the ability of global pooling techniques to adjust to the available computational resources on-the-fly by altering the input image size (the results are reported on a concept detection task)

<table>
<thead>
<tr>
<th>Image size</th>
<th>Clas. Accuracy</th>
<th>Clas. Time per Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>224x224</td>
<td>82.43%</td>
<td>9.11 msec</td>
</tr>
<tr>
<td>200x200</td>
<td>82.62%</td>
<td>6.91 msec</td>
</tr>
<tr>
<td>176x176</td>
<td>82.12%</td>
<td>5.10 msec</td>
</tr>
<tr>
<td>152x152</td>
<td>81.69%</td>
<td>3.68 msec</td>
</tr>
<tr>
<td>128x128</td>
<td>79.13%</td>
<td>2.52 msec</td>
</tr>
<tr>
<td>104x104</td>
<td>74.65%</td>
<td>1.65 msec</td>
</tr>
</tbody>
</table>
Knowledge Transfer

- **Knowledge transfer techniques** (e.g., distillation, hint-based training) also allows for **increasing the performance of smaller and more lightweight models**
- **Neural Network Distillation** [1]
  - Train a large and complex model
  - Train a smaller model to regress the output of the larger model
  - The temperature of the softmax activation function is increased to maintain more information
- **Hints for Thin Deep Nets** [2]
  - The basic distillation idea is followed
  - A random projection is used to provide hints for intermediate layers

---

Knowledge Transfer

• **Similarity Embedding-based Knowledge Transfer**
  - Instead of matching the output of the layers, the smaller model is trained to match the similarities between the training samples.
  - Similarity Embeddings [1] were used to this end.
  - This allows for directly transferring the knowledge, even when different number of neurons are used in each layer, without regressing the output of the layer.

Knowledge Transfer

- Preliminary results on a pose classification task are reported.
- The nearest centroid classifier was used to evaluate the “quality” of the knowledge transfer on an intermediate layer.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (NCC)</th>
<th>Accuracy (NN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distillation</td>
<td>78.21%</td>
<td>81.76%</td>
</tr>
<tr>
<td>Sim. Emb.</td>
<td>77.99%</td>
<td>-</td>
</tr>
<tr>
<td>Sim. Emb. + Distill.</td>
<td><strong>80.81%</strong></td>
<td><strong>83.11%</strong></td>
</tr>
</tbody>
</table>
Reinforcement Learning
Camera Control

- Deep Reinforcement Learning techniques can be used to provide optimal end-to-end control of the camera
  - Deep Q Learning [1] (discrete control)
  - Policy Gradients [2] (continuous control)
- The reward function can be used to measure the quality of the obtained shots according to cinematography objectives


Proof of concept

• To examine whether it is possible to directly control the camera using visual information, we used a simple PID controller.
• The aim was to keep the detected bounding box to a specific position and appropriately adjust the zoom.
• Very good results were obtained in our simulations.
Proof of concept
Training and Deployment
Deep Learning Frameworks

• There are several deep learning frameworks that can be used to train and deploy deep learning models!
  • Deploy-oriented frameworks/libraries: Caffe, Tensorflow, Darknet
    • Darknet is not well documented, but the code is quite simple and it is easy to use it in deploy-oriented code
  • Other frameworks/libraries, e.g., PyTorch, are more research-oriented than deploy-oriented
• Training the models usually requires high-end GPUs (e.g., GTX-1080, Titan X, etc.)
  • Training on CPU is infeasible!
Drone Deployment

• **GPU-accelerated hardware**, e.g., the NVIDIA TX-2 module, must be used during the deployment to ensure adequate performance.

**TX2 example**
• integrated 256-core NVIDIA Pascal GPU
• a hex-core ARMv8 64-bit CPU complex
• 8GB of LPDDR4 memory
Drone Deployment

• The Robotic Operating Systems (ROS) is used to provide a seamless integration platform
  • Other solutions are also possible, but ROS is well established in the robotics community!
• Each deep learning algorithm can be executed as a ROS node
• Grouping several deep learning tasks into the same node can reduce the communication overhead in some cases and improve the performance of the system
Drone Deployment

• NVIDIA also provides a set of deploy optimization tools that can further accelerate the models.
• Using the TensorRT library with a crowd detection Caffe model leads to a significant speedup
  • Without TensorRT: 45 fps
  • With TensorRT: 100 fps
NVIDIA Redtail Project

- Drones that fly autonomously and **follow a trail through a forest using only visual information**
- The **technical and implementation details** of this project are provided in their paper "Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness" [1]
- The implementation is **open-source** and available at https://github.com/NVIDIA-Jetson/redtail/
  - ROS nodes, interface with the Pixhawk flight controller, etc.
  - TensorRT implementation of YOLO

NVIDIA Redtail Project

- Object detection @ 1fps using a 16bit YOLO variant
- TrailNet used to provide orientation and lateral offset (treated as classification problem)
  - Over-confident networks perform worse
  - Learning the dataset vs. performing well in deployment
    - TrailNet was the only one able to fly autonomously even though it didn't achieve the best accuracy
- Modifications to the used networks to be able to run on limited resources devices (TX-1)
  - Removing layers, etc.
Q & A

Thank you very much for your attention!

Contact: Prof. A. Tefas, tefas@aiia.csd.auth.gr
www.multidrone.eu