Recurrent Neural Networks for object detection

Bin Qasim Ahmad
Technical University of Munich
Department of Informatics
Munich, Germany
ahmad.qasim@tum.de

Pettirsch Arnd
Technical University of Munich
Department of Informatics
Munich, Germany
a.pettirsch@outlook.de

***Abstract*—ToDo**

***Index Terms*—TBD.**

I. Introduction

A. Image and Video Object Detection in general

- Image object detection history.
  - Bayesian methods before deep learning
  - ImageNet challenge and VID [15]
  - Deep Learning and AlexNet [16]
- Single stage and 2-stage image object detectors.
  - A two-stage pipeline firstly generates region proposals, which are then classified and refined. [17]
  - A single-stage method is often more efficient but less accurate. Directly regress on bounding boxes and classes. [18], [19]
- Why is video object detection harder?
  - Large size
  - Motion blur
  - Quality of the dataset
  - Partial occlusion
  - Unconventional Poses

B. Recurrent Neural Networks in general

1) Delimitation to non-recurrent Neural Networks: Non-recurrent Neural Networks process on single inputs. In the field of image classification this could be for example a single image.
Recurrent Neural Networks process on sequences of date. In the field of classification those sequences consist often multiple frames. Recurrent Neural Network's core concept to enable the sequence processing is parameter sharing across different parts of a model. Parameter sharing can be reached by cycles in the architecture. [11]

2) Common Types of Recurent Neural Networks: Most of the paper which are described in this work use two pretty common approaches of Recurrent Neural Networks. The first one are LSTMs, first mentioned in [18] in 1997. The key of LSTMs is the inner state unit. LSTMs consist of four layers. Those four layers are able to remove or add information to the inner cell state.

The second type of Recurrent Neural Networks is Gated Recurrent Units (GRUs). The main difference to LSTMs is, that those GRUs consist out of a single gated unit which can simultaneously control the forgetting factor and decide to update the inner cell state. [11]

II. Feature-based Video Object Detection

First, we want to introduce feature-based Video Object Detection methods. As defined, for example, in [1-2] feature-based Video Object Detection methods fuse detectors which integrate features from multiple frames into their video detection. In most of our papers those detectors use recurrent units to integrate the visual features from different frames.

A. Recurrent Multi-frame Single Shot Detector for Video Object Detection [1]

Broad, Jones and Lee have in [1] and [12] the idea to design a multi-frame video object detector by inserting a fusion-layer into a classical Single Shot Detector (SSD) meta-architecture. Based on this idea they research in two main fields: On the one hand, they investigate different fusion methods and on the other hand they try several SSD meta-architectures [20-2.1.1; 20-3.6.1]. They test their approaches on the KITTI dataset [21] for autonomous driving and their model improves a state-of-the-art SSD model by 2.7 % mAP. Finally they evaluate their best approach on Caltech Pedestrians Dataset [22] and find similar improvements of 5 % in maP.

SSDs is generally consist of two main components. First, an so called Feature Extractor which consist of convolutional and pooling layers. As an input the feature extractor gets an image and it outputs feature-maps. The other component is an Detection head which consists of convolutional layers and creates bounding boxes and class probabilities out of the feature maps created by the feature extractor. [1-3]

As mentioned above Broad, Jones and Lee have the idea to insert a fusion layer in between those two components. As fusion techniques they test simple element-wise operations (e.g.: add or max), simple concatenations of features maps and recurrent layers [1-3]. Their experiments show that the recurrent layers perform best. Element-wise operations do not lead to an improvement in the mAP. And concatenations just add another depth to the network, which increases the mAP slightly.Only the Recurrent Units add new parameters, to learn temporal context, to the Network. In addition, they observe that recurrent units do not slow down the computational speed significantly. The state of the art SSD, which is used to test the different fusion techniques performs on 55 fps the recurrent one on 50 fps [1-4.1]. As a recurrent unit they use GRU because they observe that the results were similar to the results when they use LSTMs but the GRUs are faster to train [1-3.1.1]. Their final architecture is shown in Figure 1.

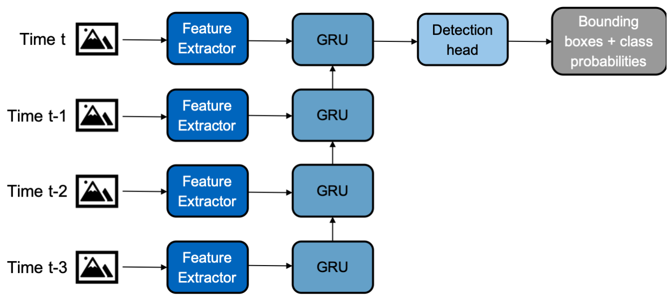

Fig. 1: Architecture: Recurent Multi-frame single shot detector [1]. The Feature Extractor generates feature maps for four frames and feeds those feature maps into the GRUs. Detection Head uses the final feature maps which are created by the GRU with respect to the temporal context to create Bounding Boxes and Class probabilities

In addition to the investigation on the type of fusion layer, Broad, Jones and Lee test different types of SSDs as a baseline for their architecture. They find out that for all baseline SSDs the mAP was higher in comparison to the non-recurrent models. The mAP increase by 1.4 to 4.4 percent on KITTI dataset. The best mAP is made with SqueezeDet+ [23] as a baseline SSD network [1-4.2].

Finally Broad, Jones and Lee use the Caltech Pedestrians dataset to expolore the effect of the number of prior frames and the frame-rate. They compare the single-frame SSD model, with the RMf-SSD using the prior frames and the RMf-SSD model using the frames t-2, t-4 and t-6. The improvement was 3 percentage points higher by using the frames t-2, t-4 and t-6 in comparison to use the prior 3 frames [1-4.4].

There is only a little information on the training of the Recurrent Multi-frame Single Shot Detector. They use the SequeezeDet Training strategies [23-3.3] and a pre-trained version of the baseline SSD. Finally, they use the SequzzeDet fine tuning strategy to train the whole network afterwards [12-2].

Overall Broad, Jones and Lee show that they are able to improve single-frame models on Kitti dataset and also on Caltech Predestrians dataset by adding recurrent units. Finally they reach an mAP of 83 % on KITTI dataset [1-4.2] and a miss-rate of 29 % on the Caltech Pedestrians dataset [1-4.1].

This architecture is comparable to the architectures described in 2-B and 2-E. Main differences are that 2-B uses more than one recurrent unit and 2-E different feature extractors for different frames. But all of them feed feature maps of different frames into recurrent units.

B. Mobile Video Object Detection with Temporally Aware Feature Maps [2]

Liu and Zhu have in [2] the goal to design a video object detection architecture, which can run real-time (15 fps) on low-powered mobile and embedded devices. The key of their method is to combine convolutional layers with convolutional LSTMs. On this core idea they do some further research. They test the benefit from adding an LSTM into the baseline SSD, different types of Recurrent Layers (LSTM, GRU and bottleneck LSTMs), different dimensions of their bottleneck LSTMs and different LSTM placement strategies (Single LSTM placement and multiple LSTM placement). They test their model on Imagenet VID 2015 dataset and reached a mAP of 54.4 % while performing on 15 fps.

In the beginning they simply add one LSTM to their baseline SSDs architecture - MobileNet [24]. They observe that adding the LSTM improves the mAP in comparison to their baseline SSD architecture. Moreover, they investigate that the greatest improve is by adding the LSTM after the 13th convolutional Layer of the SSD. [2-4.2]

Afterwards, they compare LSTMs, GRUs and Bottleneck LSTMs as different types of Recurrent units by placing them after the 13th convolutional layer. Bottleneck-LSTMs have been designed by Liu and Zhu to increase the efficiency of LSTMs. For that purpose they use slightly unusual the ReLU function as activation for the LSTM. Moreover they, compute a so called Bottleneck feature map with less input channel than the originally feature map and feed this map into the LSTM to reduce computational power. They come to the conclusion that Bottleneck-LSTMs are more effective the LSTMs and in the case of a convolutional kernel greater 1x1 even more effective than GRUs while attaining comparable perfomance. [2-3.4; 2-4.2]

In addition to the bottleneck LSTMs Liu and Zhu extend their network with multipliers. They use α_{base}, α_{ssd} and α_{lstm} as multiplier to scale the channel dimension of each layer. During their research they find out that the accuracy of the model remains near constant up to $\alpha_{ssd} = 0.25\alpha$ which means that the output of each LSTM is one-fourth the size of the input. For the other multipliers they use the values: $\alpha_{base} = \alpha$ and $\alpha_{ssd} = 0.5\alpha$. [2-3.3; 2-4.2]

As shown in Fig. 2 the final model uses LSTMs after all feature maps, because Liu and Zhu observe that there a slight performance improve by adding LSTMs after every feature map and nearly no change in computational cost [2-4.2].

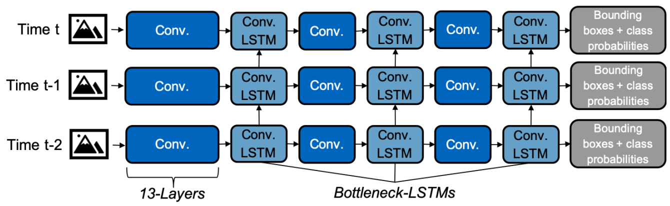

Fig. 2: Architecture Mobile Video Object Detection with temporally aware feature maps [2]. First 13 Convolutional Layeer create Feature Maps. Those Feature Maps are fed into the Bottleneck-LSTMs (faster LSTMs). Afterwards Convolutional-Layer create new Feature Maps and feed them again into Bottleneck-LSTMs.

On the training strategy and the loss function Liu and Zhu do not provide any information.

Liu's and Zhu's final model reaches a mAP of 54.4 percent on Imagenet VID dataset. They compare their model with some single frame SSD networks. Especially the comparison with the Mobilenet-SSD model is interesting because this model also works on a mobile machine in real-time. Liu and Zhu outperform the Mobilenet-SSD network by 4.1 percent in mAP. Moreover they reach shorter run times on the Pixel 2 phone then the Mobilenet-SSD architecture.

This architecture can be especially compared to the model described in "Looking Fast and Slow" because both of them are processing on a mobile device and are feature-based architectures. But "Looking Fast and Slow" differs because it does not treat every frame in the same way and proceeds with different features extractors on different frames.

C. Feature Selective Small Object Detection via Knowledge-based recurrent attentive neural networks [6]

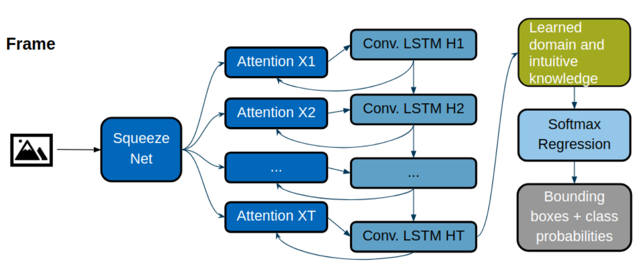

Fig. 3: Architecture Feature Selective Small Object Detection via Knowledge-based recurrent attentive neural networks

The aim of this paper is develop a video object detector for the purpose of autonomous driving. The network developed in this paper is termed as Knowledge-Based Recurrent Attentive Neural Network (KB-RANN).
The main contributions of the authors are:

- An attention mechanism that works like human cognition. The attention mechanism can detect the salient features, which are important for the object detection problem and propagate them forward. This attention mechanism is based on previous work by Ashish Vaswani et. al. [30].
- A domain and intuitive knowledge module which can use the knowledge about traffic signals to produce feature maps.
- A model which has good ability for transfering knowledge. The authors obtained good results by evaluating the KB-RANN model on BTSD dataset [29], which was trained on KITTI dataset [21].

The KB-RANN, model is tested on KITTI and MS COCO17 dataset [31], achieving 0.813 mAP and 0.578 mAP on all classes on those datasets respectively. The model performs better then several popular objection detection algorithms like Faster R-CNN [15], RetinaNet [32] and SqueezeDet [33]. The authors successfully compressed and accelerated their proposed model and deployed it to their own self-developed autonomous car.

The authors use SqueezeNet [34] for feature extractor because SqueezeNet can provide AlexNet level of accuracy but with 50 times less parameters. Although there are better performing models like VGGNet and ResNet, but these models are computationally expensive. They make a few changes in the SqueezeNet architecture, which include changing the kernel size and also fine-tuning the backend by adding two fire modules at the end. This modified SqueezeNet architecture is given the term, SqueezeNet+.

Attention mechanism is used to find the saliency maps from the deep feature maps obtained from SqueezeNet. Attention module obtains the input tensor X and outputs the saliency map $\tilde{X}_t$.

LSTM [18] is used as a memory mechanism in order to find long term dependencies between different frames and the characteristics of Attention and LSTM are exploited together by fusing them with each other, this fused module is termed as RANN. As saliency feature extraction is applied on the original deep feature maps, it is possible that some of the information is lost. Multiple RANNs are cascaded together into a chain to minimize the effect of feature information loss. Moreover, the output H_t of each LSTM is concatenated with the original feature map and used as an input for the next attention module to refine the saliency feature extraction process.

Lasty, The authors also add a domain and intuitive knowledge module to the KB-RANN architecture. It is assumed that traffic signs detection is one of the most important aspect of autonomous driving. The major focus of attention of a driver is at the center of vision and the traffic lights are always located at a bias from that central region of peoples' gazes. With these assumptions in place, domain knowledge about traffic signals is learned from the data itself by constraining the distribution to a 2D Gaussian function and learning the mean and covariance matrices from the data.

For training and evaluation of the model, the authors use the KB-RANN model and MS COCO17 dataset. They compare the results with other popular object detection models. On the KITTI dataset the KB-RANN achieves the mAP of 0.813, compared to the 0.763 of SqueezeDet, 0.702 of Faster R-CNN and 0.601 of RetinaNet. The frames-per-second (FPS) at which KB-RANN operates are higher then the compared models. In order to signify the accuracy gain from the attention mechanism and the knowledge module, authors also train and evaluate different architectures, namely KB-RCNN in which the attention mechanism is replaced with convulational layers and RANN, which does not have knowledge-based module. KB-RANN achieves better accuracy then KB-RCNN and RANN on KITTI and MS COCO17 dataset, although the FPS achieved by KB-RCNN due to its recurrent nature are higher. Lastly, a KB-RANN model with parameters trained on the KITTI dataset is also tested on BTSD dataset to demonstrate the knowledge transfer capabilities.

D. Looking fast and slow: memory-guided mobile video object detection [7]

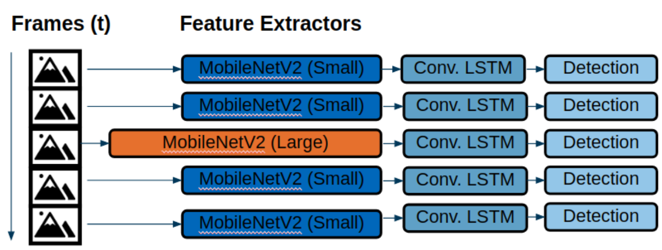

Fig. 4: Architecture Looking fast and slow: memory-guided mobile video object detection

The authors aim to, replicate the capability of a human visual system of obtaining the ”gist” of a scene. Using this sparse information, and amalgamating it with the more thorough information, a human visual system can effectively detect objects in its field of vision. In light of this, the main contributions of this paper are:

- The introduction of multiple feature extractors. Some of those feature extractors are very light-weight but can provide a ”gist” of the frame, while others can provide more accurate representation of a frame but at the cost of performance.
- A memory module, which can fuse the outputs of those feature extractors.
- An adaptive interleaving policy which uses reinforcement learning to find to decide, which feature extractor should be executed.
- The capability of executing multiple feature extractors asynchronously. So that, the light-weight feature extractors don’t have to wait for the more expensive feature extractors.

The model is evaluated on the ImageNet VID 2015 dataset [35], with extra data from the ImageNet DET [35] and MS COCO17 datasets [31]. The model manages to achieve a mAP of 0.593 at 72.3 FPS on a Pixel3 mobile device.

Let m be the number of feature extractors. For the implementation of this paper, the authors constraint $m = 2$, a d let f_1 be the feature extractor which is computationally more expensive while f_2 be the feature extractor which is cheaper. Both feature extractors use MobileNetV2 architecture [36]. f_1 uses a depth of 1.4 with $320 * 320$ input frames resolution while f_2 uses a depth of 0.35 with a lower $160 * 160$ input frames resolution.

A modified LSTM [18] is used as a memory mechanism to preserve long term dependencies. The modifications in the LSTM are responsible for making the memory mechanism faster and also better at preserving long term dependencies. For faster memory mechanism, the authors make three modifications. They introduce bottlenecking and add a skip connection between the bottleneck and the output. Lastly, the LSTM states are grouped together and convulations are applied on each group seperately and the resultant states are then concatenated together to obtain the final states. The grouped convulations provide a speed-up. In order to improve the preservation of long term dependencies by the LSTM, the LSTM states are only updated when f_1 is run and not the f_2, as feature maps obtained from f_2 are not of a higher quality as of f_1 and this can result in loss of important state information in the LSTM.
SSD-style [17] detection is applied on refined feature maps obtained from the LSTM for classification and bounding boxes.

The interleaving policy which defines the feature extractor that should be used next is based on reinforcement learning. The state space of the reinforcement learning policy network π consists of the LSTM states, c_t, h_t, as well as the differences between the states at different timestamps i.e. $c_t - c_{t-1}$, $h_t - h_{t-1}$ and lastly the action history η_t. The action space has length m and the action a means that the feature extractor f_a should be run.

It is observed by the authors that, despite the employment of the interleaving policy the real-time detection is limited by the execution of the expensive f_1 feature extractor. They introduce an asynchronous framework for running the feature extractors f_1 and f_2 extractors in parallel. During testing, this asynchronous framework provides better results.

As mentioned before, the authors use ImageNet VID 2015 dataset for training and evaluation, which has 30 classes in total, along with the addition of extra data from ImageNet DET and MS COCO17, but this extra data is limited to the classes contained within ImageNet VID. All results are reported, using a Pixel 3 mobile device. The results are compared with, baseline single-frame detection model i.e. MobilenetV2-SSDLite (mAP: 0.420, FPS: 14.4), LSTM-based model i.e. MobilenetV2-SSDLite+LSTM (mAP: 0.451, FPS: 14.6) and the state of the art mobile video object detection model of Zhu et. al. (mAP: 0.602, FPS: 25.6). The proposed model by authors manages to achieve a mAP of 0.593 at 72.3 FPS. The authors also perform evaluation using slight variations of the proposed architecture i.e. Non-interleaved, Interleaved only, Interleaved+Async and Interleaved+Adaptive+Async, in order to test the significance of different components of the architecture on the end results. The Interleaved+Adaptive+Async provides the most balanced end result. [37]. Although the

E. Detect to Track and track to detect

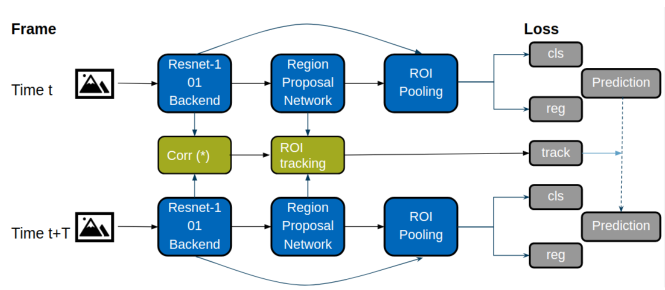

Fig. 5: Architecture Detect to Track and track to detect

In this paper, the authors aim to perform, detection and tracking simultaneously using a fully convulational network by infering a "tracklet" over multiple frames. This paper is based on the R-FCN [19] model and extends it for multiple frame and tracking. The main contributions of this paper are:

- Finding correlation maps of two feature maps of adjacent frames. These correlation maps are used to estimate the local displacement between frames.
- Using ROI-tracking layer to regress bounding boxes over multiple frames.
- Linking the detections based on tracklets to infer long-term tubes of objects over the course of the video.

The proposed model is trained and evaluated on the ImageNet VID dataset [35], consisting of 30 classes. The model achieves an overall mAP of 0.82.

The R-FCN [19] detection process consists of two stages, firstly a Region Proposal Network (RPN) [38] is used to find the candidate region of interests (ROIs) and then a ROI pooling layer [39] is used to perform classification of the regions into classes or the background. The input to the ROI pooling layer comes from a convulational layer put in place at the end of the ResNet-101 feature extractor [40] with output x^t_{cls}. The ROI layer outputs position sensitive feature maps and using a softmax layer these position senstive feature maps can be converted to class probablities for each ROI. On a second branch, R-FCN places a convulational layer at the end of the ResNet-101 feature extractor which outputs x^t_{reg} and this ouput is again used as an input to ROI pooling layer which generates the bounding boxes.

For each pair of adjacent frames I^t, $I^{t+\tau}$, a bounding box regression layer is introduced that performs position senstive ROI pooling on the concatentation of bounding box regression features x^t_{reg}, x^{t-1}_{reg}, which are also stacked with correlation maps, to perform bounding box transformation regression between frames. The correlation maps between the two frames are obtained by finding correlation between the feature maps of the two frames. Finding correlation on all the features in the feature maps will result in an explosion of dimensionality so the correlation maps are only limited to the local neighbors. Like mentioned before, the correlation maps are stacked with the bounding box features maps.

In practical uses, using all frames of the video is not the most efficient way of detection and tracking, as a lot of information between adjacent frames is redundant and also due to GPU memory and computational restrictions, only a certain number of frames can be processed in the GPU at the same time. Due to this, the authors employ a technique similar to action localization [41] to obtain an optimal path through a video.

As mentioned above, the proposed model is trained and evaluated on the ImageNet VID dataset. A comparison is made with the R-FCN detector (mAP: 0.742), which the proposed model is based on, the ILSVRC 2015 [42] (mAP: 0.738) winner and the ILSVRC 2016 [43] (mAP: 0.762) winner. DT performs the best with a mAP of 0.82. A comparison between slight variations of the DT model are also evaluated. Firstly, using different feature extractor backbones: ResNet-50 (mAP: 0.767), ResNet-101 (mAP: 0.80) and Inception-v4 (mAP: 0.821) [44] and secondly using the ResNet-101 backbone but a temporal sampling rate τ of 10 (mAP: 0.786).

III. Box-Level-based Video Object Detection

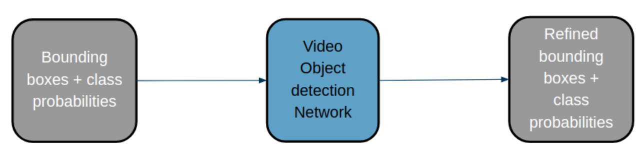

Fig. 6: Box-Level-based Video Object Detection

Bounding Boxes and Class probabilities are fed into the network and are refined temporally and/or spatially.

A. Optimizing Video Object Detection via Scale-Time Lattice

The aim of this paper, is to propose an architecture which is balanced and flexible enough to allow prioritisation of accuracy or performance with minimal effort. The primary contributions of this paper are:

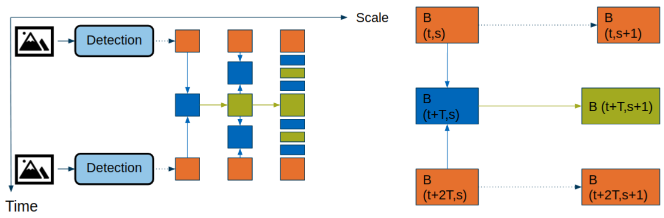

Fig. 7: Architecture Optimizing Video Object Detection via Scale-Time Lattice

- The Scale-Time Lattice, which provides a rich design space.
- A detection framework which provides good accuracy to performance trade-off.
- A novel key-frame extraction policy, which is based on the ease of detection.

The Scale-Time Lattice allows coarse detection, both temporally and spatially and then use temporal propagation and spatial refinement to go from coarse to fine detection.

The Scale-time lattice is composed of structures which connect together to perform the temporal and spatial operations. These structures are called Propagation and Refinement Units (PRUs). PRUs work on the basis of two operators F_τ and F_S, and by carefully allocating resources to the two operators, a balance between the detection accuracy and the performance can be acheived. Assuming that two frames are sampled from a video at time t and $t+2\tau$, both at scale s, the F_τ operator tries to model the movement of the bounding boxes from time t to $t+\tau$ and from time $t+2\tau$ to $t+\tau$ irrespective of the scale offset between the scale s and the ground truth frame scale $s+1$. The modeling of the bounding boxes offset from scale s to $s+1$ is the task of the F_S operator. The figure depicting the PRU gives more information on this.

The most straight-forward keyframe extraction policy is a uniform i.e. to use keyframes from the video after uniform intervals but an intelligent keyframe extraction policy can be used to increase the accuracy of the results. To that effect, the authors introduce a ease of detection cofficient e. If during detection, the value of e falls beneath a certain threshold that the sample rate of the frames from the video is increased. This can happen if there are a lot of objects on the screen or the objects are moving too quickly.

The proposed model has an mAP of 0.79 at 60 FPS on the ImageNet VID dataset.

B. Context Matters: Refining Object Detection in Video with Recurrent Neural Networks [3]

In [4] Tripathi, Lipton, Belongie, Nguyen come up with a neural network architecture for video-based object detection. Their architecture consists two parts: A pseudo-labeler, which assigns labels to all video frames and a recurrent unit which refines those pseudo-labels by using the contextual information. Moreover, they describe a training strategy for their architecture and compare their approach to other models on the YouTube-Objects dataset (v2.0), which consists of the ten categories airplane, bird, boat, car, cow, dog, horse, mbike, train. Their model reaches an mAP of 68.73 percent which improves the strongest image-based baseline for Youtube-Video Objects dataset of 7.1% [4-Abstract; 25].

The final architecture can be found in **Fig 2**. Tripathi, Lipton and Belongie first train the pseudo-labeler, which is an YOLO object detection network [26] originally trained for 20-class PASCAL VOC [27] on the YouTube-Video Dataset. As specified in YOLO they minimize the weighted squared detection loss and optimize classification and localization error simultaneously [4-3].

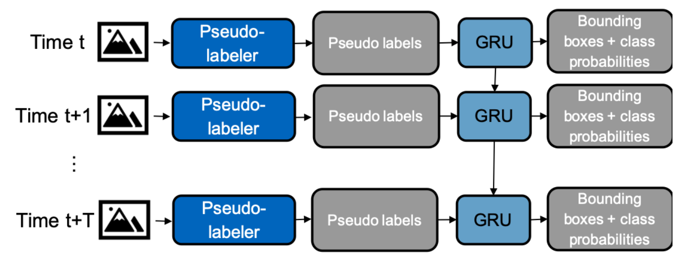

Fig. 8: Architecture Context Matters: Refining Object Detection in Video with Recurrent Neural Networks [4]. First the "Pseudolabler" creates bounding boxes and class probabilities for every input frame. Afterwards the GRU fuses the output of the current and some past frames and refines the bounding boxes and class probabilities.

After training the pseudo-labeler they train the Recurrent Neural Network, which takes as an input the pseudo-labels and outputs improved prediction. The RNN consists of two GRU layers [4-2].

For training the whole network they use the following loss function to take both accuracy at the target frame and consistency of predictions across adjacent time steps into consideration. They choose the values of $\alpha = 0.2$, $\beta = 0.2$ and $\gamma = 0.1$ based on the detection performance on the validation set [4-2.1]:

$$loss = d_{loss} + \alpha \cdot s_{loss} + \beta \cdot c_{loss} + \gamma \cdot pc_{loss}$$

For the final output Tripathi, Lipton, Belongie and Nguygen use the object detection loss as described in YOLO [26, 4-2.1.1]:

$$detection_{loss} = \lambda_{corr}\ todo$$

The similarity loss considers the dissimilarity between the pseudo-labels and prediction at each frame t [4-2.1.2]:

ToDo

The category loss takes wrong class probabilities into consideration [4-2.1.3]:

-

And the prediction-consistency loss regularizes the model by encouraging smoothness predictions across the time-steps [4-2.1.4]:

-

During the evaluation they find two possible areas of improvement for their approach. On the one hand the RNN is not able to recover from wrong predictions made by the pseudo-labeler after they have been fed into the RNN. On the other hand, they observe that their network is not robust to motion [4-3.4].

Tripathi, Lipton, Belongie and Nguyen test their model on the Youtube-Objects dataset. Overall they outperform the best no-recurrent architecture (DA Yolo) in their comparison by 7% mAP. It's notable that DA YOLO reaches gut results in detection of airplanes and birds but comparatively bad result in detection of cats, cows, dogs, horses and mbike. In those categories the recurrent networks in the comparison outperform the non-recurrent ones [4-3.1].

The architecture mentioned in ”Context Matters: Refining Object Detection in Video with Recurrent Neural Networks” is pretty similar to the one used in [4]. Both are using the YOLO network architecture as a baseline and feed its output into an Recurrent Unit. Main Differences is that [4] feeds the bounding boxes and in addition some visual features into the recurrent unit.

C. Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking [5]

Ning, Zhang, Huang, He, Ren and Wang design in [5] a combination of box-level and feature-level based Video detectors to track objects in video frames. They use the YOLO [**?**] network to create high-level visual features and preliminary location inferences and feed both into a recurrent unit. They test their approach on the OTB-30 dataset [28] and compared it with 9 different state-of-the-art trackers. Their architecture, called ROLO, is shown in **figure**. [5-Abstract; 5-4]

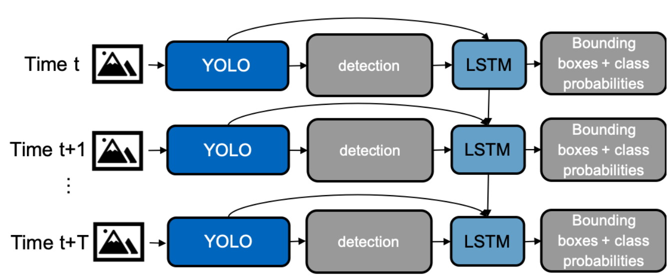

Fig. 9: Architecture ROLO [5]. The architecture feeds feature representations made by the YOLO network as well as bounding boxes also made by the YOLO Network into a recurrent unit. The recurrent unit uses the temporal context of those inputs to finally output bounding boxes and class probabilities.

The architecture consists two main parts. The YOLO [27] networks which collects visual features and outputs also preliminary location inferences and the LSTM which is used as a tracking module [5-2].

They use three phases to train their model: The pre-training phase of convolutional layers for feature learning, the traditional YOLO training phase for object proposal and the LSTM training phase for object tracking. [5-3]

In the pre-training phase convolutional layers, which create feature maps are pretrained on ImageNet data. Afterwards the YOLO-architecture is adopted as detection module and trained by using the traditional YOLO training phase. Last they add LSTMs for tracking. The LSTMs are fed with, the feature representations from the convolutional layers, the Bounding boxes from the detection module and the output states from the LSTM in the previous time-step. For training they use the Mean Squared Error (MSE) [5-3]:

$$L_{MSE} = \frac{1}{n} TODO$$

As an alternative, Ning, Zhang, Huang, He, Ren and Wang mention a Heatmap as input for the LSTM. Therefor the prediction and the visual features are concatenated before adding them to the LSTM. This is helpful to visualize the intermediate results. [5-3.3]

Special to this architecture is, that LSTM does regression in two folds. There is a regression within one frame between the location inferences and the high-level features and there is also regression over the different frames of the sequence by taking the temporal context into account [5-3.4].

In 2015 Wu, Lim and Yang published a comparison of different online object trackers on their one data set [28]. The best 9 architectures (STRUCK; CXT; OAB; CSK; VTD; VTS; LSK; TLD; RS) were used by Ning, Zhang, Huang, He, Ren and Wang to compare them with their own architecture. They use 30 videos with different objects (e.g.: car, human, Skater, Couple,..) out of the benchmark [28] dataset. They get the best tracking result in 13 videos and the second best in 3 videos. There is no other tracker that reaches the best results in more then 4 videos.

This approach combines box-level and feature-level methods and reaches good results based on the combination of spatial regression between features and location proposals and the temporal regression in the LSTM. The paper shows a potential way to combine the best of both methods (feature based and box-level approaches) which are shown in the other papers.

IV. Flow-based Object Detection

A. Definition

Another type of architectures for Video Object Detection defined, for example in [], are architectures which use Flow-Networks to consider the temporal context. The flow network estimates the optical flow which means it projects back the location in the current frame to an earlier frame.

B. Deep Feature Flow for Video Recognition

Zhu, Xiong, Dai, Yuan and Wei develop in [2] a way to use flow nets to detect objects in video frames. As shown in figure 10 their architecture consists of three main parts: A network to create visual features, a network to create class probabilities and bounding boxes out of feature maps, and a network which estimates the optical flow.

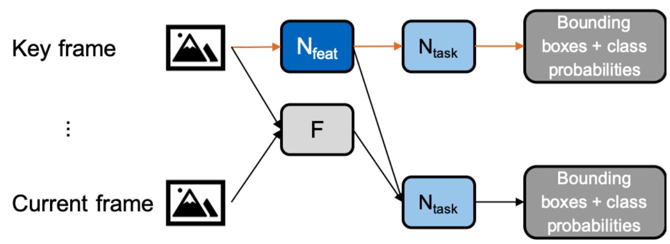

Fig. 10: Architecture Deep Feature Flow for Video Recognition

The paper differs between key frames and other frames. The network to create visual features N_{feat} only processes on key frames. It is a fully convolutional network, which takes as an input the image and outputs a feature maps. As a feature network they use a pretrained version of ResNet.

The second network N_{task} does the recognition task over the feature maps. It performs on every frame. Zhu, Xiong, Dai, Yuan and Wei use an R-FCN network for the recognition task.

For all non-key frames the feature maps are not constructed by the feature network. Instead, they are propagated by a flow network N_{flow}. Zhu, Xiong, Dai, Yuan and Wei use a state-of-the-art CNN based FlowNet architecture as a flow network. The flow estimation is given by:

$$f_i = W(f_k, M_{i->k}, S_{i->k})$$

f_k is the key frame's feature map, $M_{i->k}$ is a two dimensional flow field, it projects back the location of an object in the current frame to the location in the key frame by using bilinear interpolation and $S_{i->k}$ is scale field.

Zhu, Xiong, Dai, Yuan and Wei use a fixed key frame scheduling and they already see a potential improvement in changing the key frame policy.

V. Comparison of different approaches

A. KITTI Dataset

TABLE I: Results on KITTI Dataset

Model	MAP	FPS	Machine	Architecture
Recurrent [1]	86.0	50	Nvidia TITAN X	Feature-Level
Feature Selective [6]	81.3	30.8	Nvidia TITAN X	Feature-Level

As seen in Table I the architecture mentioned in "Recurrent Multi-frame Single Shot Detector for Video Object Detection" [1] outperforms the results of the paper "Feature Selective Small Object Detection via Knowledge-based Recurrent Attentive Neural Network" [6] in regard to the detection quality (mAP) and the computational speed (fps).

The main difference between those two architectures is that [6] performs on a single frame and uses LSTMs to include the spatial context within this frame. [1] instead is processing on a sequence of input frames and uses recurrent units to take the temporal context into consideration.

That leads to the hypothesis that performing on multiple frames is more beneficial than performing on only one frame. Which means that temporal context is more important than spatial context.

B. ImageNet Dataset

TABLE II: Results on ImageNet Dataset

Model	MAP	FPS	Machine	Architecture
DT [8]	82.0	7	Nvidia TITAN X	Feature-Level
DT [8]	78.5	55	Nvidia TITAN X	Feature-Level
Scale-Time Lattice [10]	79.6	20	Nvidia TITAN X	Box-Level
Scale-Time Lattice [10]	79	62	Nvidia TITAN X	Box-Level
DeepFeature Flow [3]	73.9	3	-	Flow-Based
DeepFeature Flow [3]	73.1	20.5	-	Flow-Based
Looking Fast and Slow [7]	60.7	48.8	Pixel Phone	Feature-Level
Object Detection with Temporally-Aware [2]	54.4	15	Pixel Phone	Feature-Level

As seen in Table 2 both "Detect to Track and Track to Detect" and "Optimizing Video Detection with spatial-time lattice" reach better results than the other papers. Both "Detect to Track and track to Detect" and "Optimizing Video Detection with spatial-time lattice" perform on multiple frames in parallel. That leads to the advantage that both of them can use temporal context from the past and also from the future. The doubling of the temporal context information is probably one reason for their comparatively good performance.

In addition to the advantage of performing on multiple frames in parallel ”Optimizing Video Detection with spatial-time lattice” uses not only the temporal context. In addition to the temporal context, this approach takes also different scales into consideration. This is a possible further reason for the good results.

The Flow-based approach [3] has comparatively bad results on ImageNet. We only evaluated one flow-based paper, but the comparatively bad performance could be an evidence that flow-based approaches’ benefit in comparison to recurrent approaches does not exist or is very small.

”Looking Fast and Slow: Memory-Guided Mobile Video Object Detection” and ”Mobile Video Object Detection with Temporally-Aware Feature Maps” cannot be compared to the other 3 approaches because they are running on mobile devices, which have pretty less computational power than the GPUs which are used by the other papers. Evaluating the comparatively bad results of ”Mobile Video Object Detection with Temporally-Aware Feature Maps”, leads to the hypothesis that the use of multiple LSTMs after every feature map is not so beneficial.

C. *Results on COCO Dataset*

TABLE III: Results on COCO Dataset

Model	MAP	FPS	Machine	Architecture
Feature Selective [6]	57.8	37.5	Nvidia TITAN X	Feature-Level

D. *Results on YouTube Dataset*

TABLE IV: Results on YT Dataset

Model	MAP	FPS	Machine	Architecture
Context Matters [4]	68.73	-	-	Box-Level

”Context Matters” is the only paper of those which we have done some research on, which uses the YouTube Dataset to test their architecture. Unfortunately the results cannot be directly compared to the other papers. Also Tripathi, Lipton, Belongie and Nguygen only compare their model with non-recurrent ones.

E. *Results on OTB Challenge Dataset*

TABLE V: Results on OTB Challenge Dataset

Model	Success Rate	IoU	FPS	Machine
Spatially Supervised [5]	0.564	0.455	20/60	Nvidia TITAN X

Out of the paper which are mentioned in this paper only ”Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking” uses the OTB Challenge Dataset to evaluate their results. Unfortunately, they are only doing a comparison with non-recurrent approaches.

VI. Outro

A. *Conclusion*

The main conclusion of our research is that temporal context matters. All the papers came to the results that their models, which use more than one frame to detect objects outperform the models with similar baseline architecture which only proceed on single frames.

Moreover, we have seen that operating on multiple-frames at the same time is a beneficial approach. It doubles the amount of temporal context information, which leads to higher mAPs.

With respect to the computational speed we noticed that the recurrent units should not be too deep. And in addition working only on some keyframes can be beneficial to increase the speed. Therefor a good key-frame policy is needed.

For detection quality and also for computational speed it is beneficial to work on different scales. This enables us to use recurrency to take the spatial and the temporal context into consideration.

B. *Further work*

Todo

References

[1] Alexander Broad, Michael Jones, Teng-Yok Lee. Recurrent Multi-frame Single Shot Detector for Video Object Detection. 2018.

[2] Mason Liu, Menglong Zhu. Mobile Video Object Detection with Temporally-Aware Feature Maps. 2018.

[3] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, Yichen Wei. Deep Feature Flow for Video Recognition. 2017.

[4] Subarna Tripathi, Zachary C. Lipton, Serge Belongie, Truong Nguyen. Context Matters: Refining Object Detection in Video with Recurrent Neural Networks.

[5] Guanghan Ning, Zhi Zhang, Chen Huang, Zhihai He, Xiaobo Ren, Haohong Wang. Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking. 2016.

[6] Kai Yi, Zhiqiang Jian, Shitao Chen, Nanning Zheng. Feature Selective Small Object Detection via Knowledge-based Recurrent Attentive Neural Network. 2019.

[7] Mason Liu, Menglong Zhu, Marie White, Yinxiao Li, Dmitry Kalenichenko. Looking Fast and Slow: Memory-Guided Mobile Video Object Detection. 2019.

[8] Christoph Feichtenhofer, Axel Pinz, Andrew Zisserman. Detect to Track and Track to Detect. 2017.

[9] Kai Kang, Wanli Ouyang, Hongsheng Li, Xiaogang Wang. Object Detection from Video Tubelets with Convolutional Neural Networks. 2016.

[10] Kai Chen, Jiaqi Wang, Shuo Yang, Xingcheng Zhang, Yuanjun Xiong, Chen Change Loy, Dahua Lin. Optimizing Video Object Detection via a Scale-Time Lattice. 2018.

[11] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning (Adaptive Computation and Machine Learning). 2017.

[12] Alexander Broad, Michael Jones, Teng-Yok Lee. Supplementary Material for Recurrent Multi-frame Single Shot Detector for Video Object Detection. 2018.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database.

[14] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. 2012.

[15] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 2016.

[16] Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector. 2016.

[18] Sepp Hochreiter, Jürgen Schmidhuber. Long short-term memory In: Neural Computation (journal), vol. 9, issue 8, p. 1735-1780. 1997.

[19] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In NIPS, 2016.

[20] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al.. Speed/accuracy trade-offs for modern convolutional object detectors . In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR*, 2017.

[21] Andreas Geiger, Philip Lenz and Raquel Urtasun. Are we ready for autonomous driving? The kitti vision benchmark suite. In *Conference on Computer Vision and Patern Recognition (CVPR)*, 2012.

[22] P. Dollár, C. Wojek, B. Schiele and P. Perona. Pedestrian detection: A benchmark. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009.

[23] Bichen Wu, Forrest Iandola, Peter H. Jin, Kurt Keutzer. SequeezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. In CVPR *Workshops*, 2017.

[24] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[25] Alexander Prest, Vicky Kalogeiton, Christian Leistner, Javier Civera, Cordelia Schmid, Vittorio Ferrari. Youtube-objects dataset v2.0, 2014. URL calvin.inf.ed.ac.uk/datasets/youtube-objects-datasets. University of Edinbough (CALVIN), INRIA Grenoble (LEAR), ETH Zurich (CALVIN).

[26] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshink, Ali Farhadi. You only look once: Unified, real-time object detection, *in CVPR 2016*.

[27] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303-338, June 2010, ISSN 0920-5691

[28] Yi Wu, Jongwoo Lim, Ming-Hsuan Yang. Object tracking benchmark. Pattern Analysis and Machine Intelligence. *IEEE Transaction on, 37(9):1834-1848*, 2015

[29] Radu Timofte, Karel Zimmermann, Luc Van Gool. Multi-view traffic sign detection, recognition, and 3D localisation. 2009.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit. Attention Is All You Need. 2017.

[31] Tsung-Yi Lin, Michael Maire, Serge Belongi, Lubomir Bourde, Ross Girshick et. al. Microsoft COCO: Common Objects in Context. 2015.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss ´ for dense object detection. 2017.

[33] B. Wu, F. Iandola, P. H. Jin, K. Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. 2016.

[34] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer. Squeezenet: Alexnet-Level Accuracy With 50X Fewer Parameters And ¡0.5MB Model Size. 2016.

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. 2015.

[36] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018.

[37] X. Zhu, J. Dai, X. Zhu, Y. Wei, and L. Yuan. Towards high performance video object detection for mobiles. 2018.

[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. 2016.

[39] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. 2016.

[40] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. 2016.

[41] S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. 2016.

[42] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-CNN: tubelets with convolutional neural networks for object detection from videos. 2016.

[43] J. Yang, H. Shuai, Z. Yu, R. Fan, Q. Ma, Q. Liu, and J. Deng. ILSVRC2016 object detection from video: Team NUIST. 2016.

[44] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. 2016.