Pruning the YOLO network

Kevin Strauß[1] and Ravil Dorozhinskii[1]

[1]Technical University of Munich

Motivation

Object detection is one of the most challenging problems in the field of computer vision especially when one has to detect and classify objects in **run time**. Examples include **autonomous car driving, driver-assistance systems, video content analysis**, etc, where predictions with relatively high accuracy have to be made almost instantly. **Deep CNN architectures** have been successfully applied for solving this sort of problem but they are accompanied by a significant increase in **computation and parameter storage costs**. The motivation of this project is to find and optimize an existing object detection algorithm that shows decent accuracy results and **improvements in execution time**.

Introduction and Background

Primarily the project goal is to implement a modern, state-of-the-art object detection algorithm, namely: **YOLO (You Only Look Once)**. Unlike Fast R-CNN or Faster R-CNN, **YOLO** predicts bounding boxes and class probabilities directly from full images in **one evaluation using a single neural network** [1]. To optimize our original network architecture, we prune the network. The goal of pruning is compressing the weights of various layers without hurting the original accuracy [2]. Our **baseline** model with **0%** of pruning has **46.3% mAP** (standard object detection accuracy measure) (for IOU = 0.5). The **best accuracy** achieved using **"MSCoco"** dataset is about **50%** according to [3].

Dataset

- A well-known **"MSCoco"** dataset has been chosen for our project [4]. The dataset contains over **80k labeled images** with both pixel and box segmentation information on objects from almost **80 categories**. The main feature of the dataset is that all images have different spacial dimension along x and y axes. It is worth mentioning that the entire data set (for instance "2014") takes almost **25 GB memory space**.
- We **reduced** (squeezed) the dataset leaving only **5 classes**, namely **"horse"**, **"bear"**, **"zebra"**, **"cow"** and **"giraffe"**.
- Additionally, all **image metadata was removed**, thus **less than a third the memory space** of the original dataset is required.

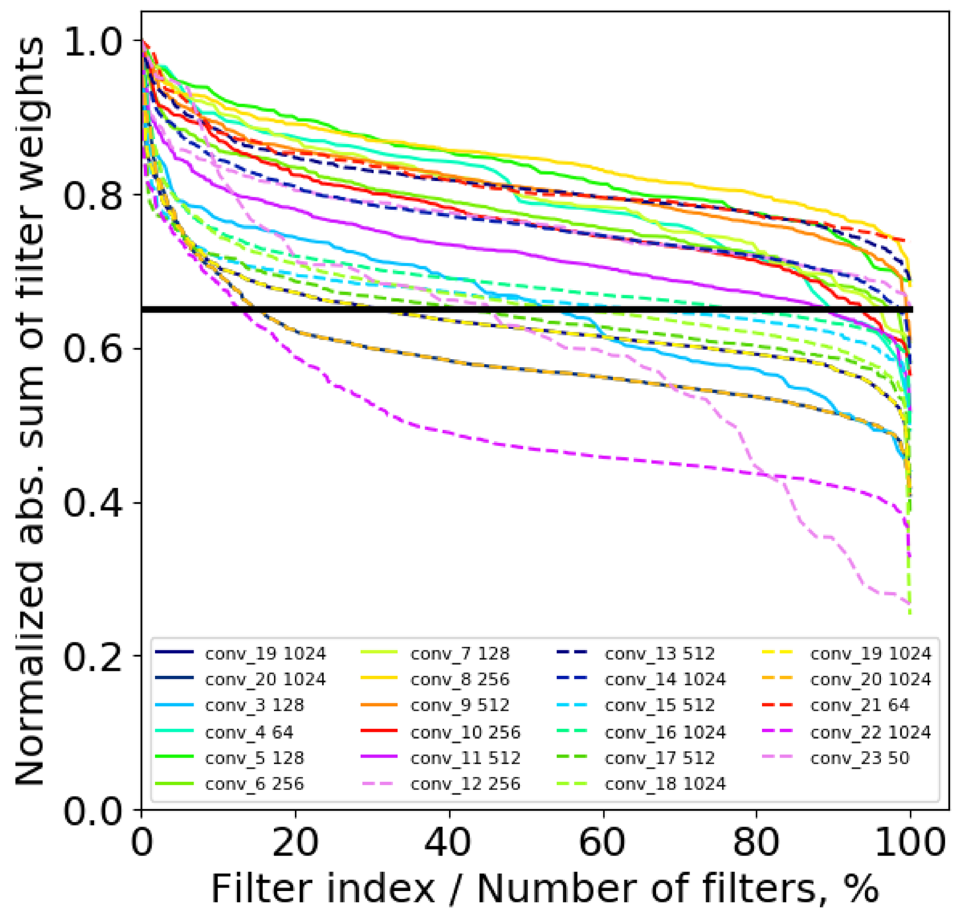
Network Architecture

- The **starting point** of our project was a **YOLO version 2** architecture [5].
- The architecture consists of **23 convolutional layers** and **5 max pooling layers** with 2x2 kernels which reduce the original image dimension by a factor of 5. An additional **skip connection** is also included to improve the gradient flow through the network.
- **Different spacial dimensions** of all images are transformed to the **fixed image size, namely: 416x416**.
- **The final output tensor has the following shape**: (**batch size**, 13,13, 5, 4 + 1 + **number of classes**) where 4 represents **x, y, height and width** of a bounding box, 1 – confidence of an object detection, 13, 13 – discretization of an original image into **13x13 grids** and 5 – number of predictions per grid cell (**number of anchors**).

Fig. 1: Illustration of the Yolo network

Pruning Approach

- **Idea**: The magnitude of the kernel weights give an **expectation** of the **magnitude of the output feature map** for each layer [2]. Filters with smaller kernel weights tend to produce feature maps with weak activations and vice versa.
- The relative importance for each filter is calculated as the sum of absolute weights as the L_1 norm $||\mathcal{F}||_1 = \sum |\mathcal{F}_{i,j}|$.
- In [2] it was found that **pruning the smallest filters works better**.
- We performed pruning across **multiple layers** using the **"independent pruning"** strategy and retrained the network following the **"prune once and retrain"** strategy [2].



Results

- The highest **mAP** of **58.2%** was achieved by pruning of the original network by **8 %** with subsequent retraining. (with a speedup of mean exection time per image by **8.54 %**).
- Pruning of the original network by **13.8%** and retraining it again showed the **mAP** of **47.7 %** and thus gave us our original **mAP**. (with speedup of mean exection time per image by **10.2 %**).
- **Pruning too much at once** has a negative effect on the model peformance.

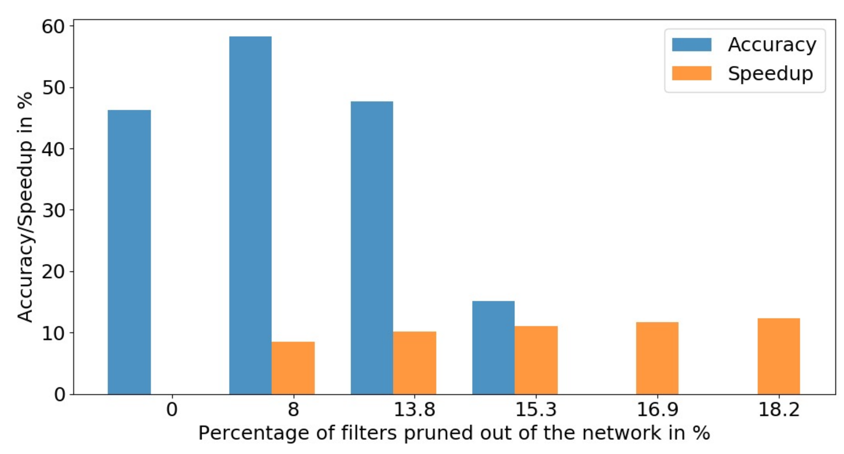


Fig. 2: Accuracy and Speedup for different pruning scenarios

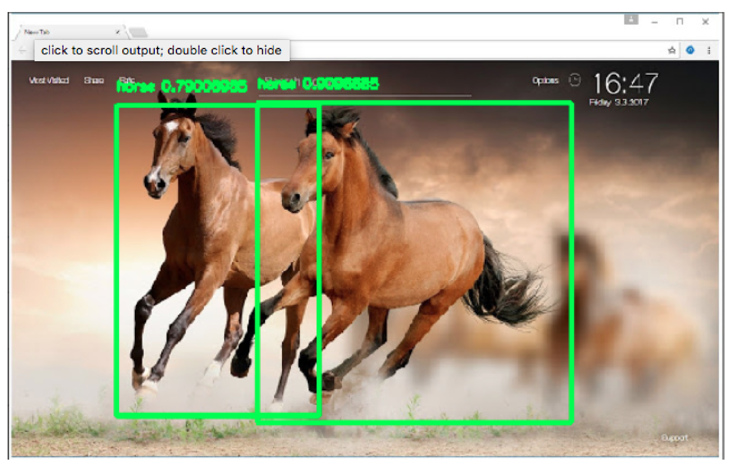


Fig. 3: Output of a network pruned by 13.6 %

Conclusion and Future Work

- We **successfully** fulfilled all our original proposals: **data augmentation**, **object detection** and we managed to perform several rounds of **pruning** and **retraining** with regaining the original mAP results of our baseline model.
- We observe that if **large portions** of the networks are **pruned away** at once (which may include sensitive layers), it may **not** be **possible** to **recover** the **original mAP** [2].
- **Multi-scale training technique** can be considered as a further **improvement**. It allows our network to be more robust to size fluctuations (e.g. very small and far away object) in the objects we are trying to detect.
- Using the **"prune and retrain iteratively"** strategy may yield better results, but the iterative process requires many more epochs especially for deep networks [2].

References

References

[1] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[2] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016.

[3] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[4] http://cocodataset.org.

[5] https://github.com/experiencor/basic-yolo-keras.