Object Detection in Urban Environment

prostdfrost

Project Overview

The project goal is to detect objects in urban environments present in Waymo Open dataset. Due to label imbalance detected objects primarily correspond to cars. Pedestrians and cyclists are much less frequently encountered.

Overall, significant improvement relative to the baseline has been achieved. At the same time, models are far away from production quality due to computational resource limitations (4k epochs were ran at max). More epochs would be required to improve performance of models (in the order of ten, hundred thousands; see [Liu et al., 2016] p. 11). Training for such a number of epochs is not feasible in the workspace.

Object detection is the key component of a self-driving car system as it essentially acts as a primary information source guiding decisions of a “Waymo Driver”. Flaws and imperfections of the detection algorithm can potentially lead to catastrophic events. Thus, robust object detection model is the cornerstone for a self-driving system.

Project

Project has been completed using the workspace provided by Udacity

Obtain the project from https://github.com/prostdfrost/cv-urban-env-dfrost

project should be downloaded to /home/workspace/ nd013-c1-vision-starter folder in order to be reproducible:

1) cd /home/workspace/
2) git clone https://github.com/prostdfrost/cv-urban-env-dfrost.git nd013-c1-vision-starter

Set Up: folders with data

The data directory contains trips copied from the workspace. The only difference is the presence of two "*train*" folders:

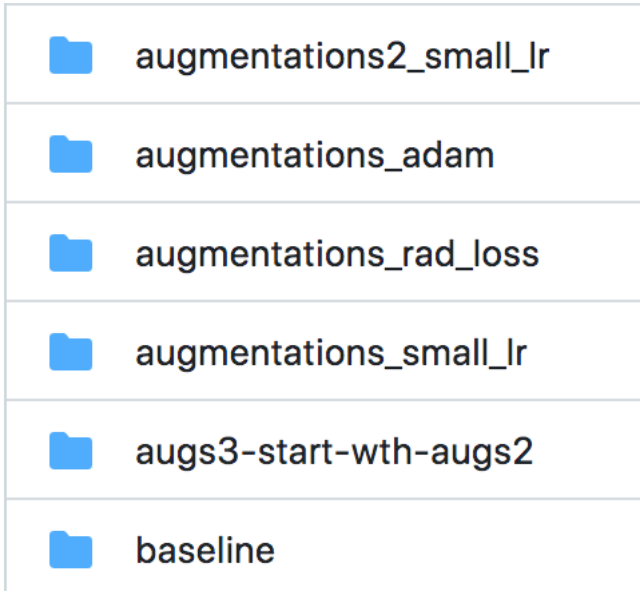
- *train* − contains two trips only, so that the effects of augmentations are more prominent
- *train-all-bc-031323* − contains all the trips originally available for training; this folder has been used for model training

test and *eval* folders are "as provided" in the workspace

The data has been already split for us into training, evaluation and test subsets. While this is the case, I double checked that the data is split on the trip basis: no frames from a single trip were mixed between training, evaluation and test partitions to prevent potential data leakage

Set Up: training and running models

Corresponding configuration files can be found in the following subfolders of the *experiments* folder:



each of which has a configuration file and a short readme. Detailed description of each model is given further.

Configuration files were originally created by edit_config.py but then were manually modified. Please use them as is, they reflect the desired state of the corresponding experiments

Exploratory Data Analysis

Some frames of some trips displayed:

various day/night, weather conditions, objects of different scales

Red – cars; green – pedestrians; blue – cyclists

By looking at 1k random frames there were found 173418 cars, 49672 pedestrians, 1284 bicyclists. Those label distribution suggests strong class imbalance and the dataset bias towards cars

Exploratory Data Analysis

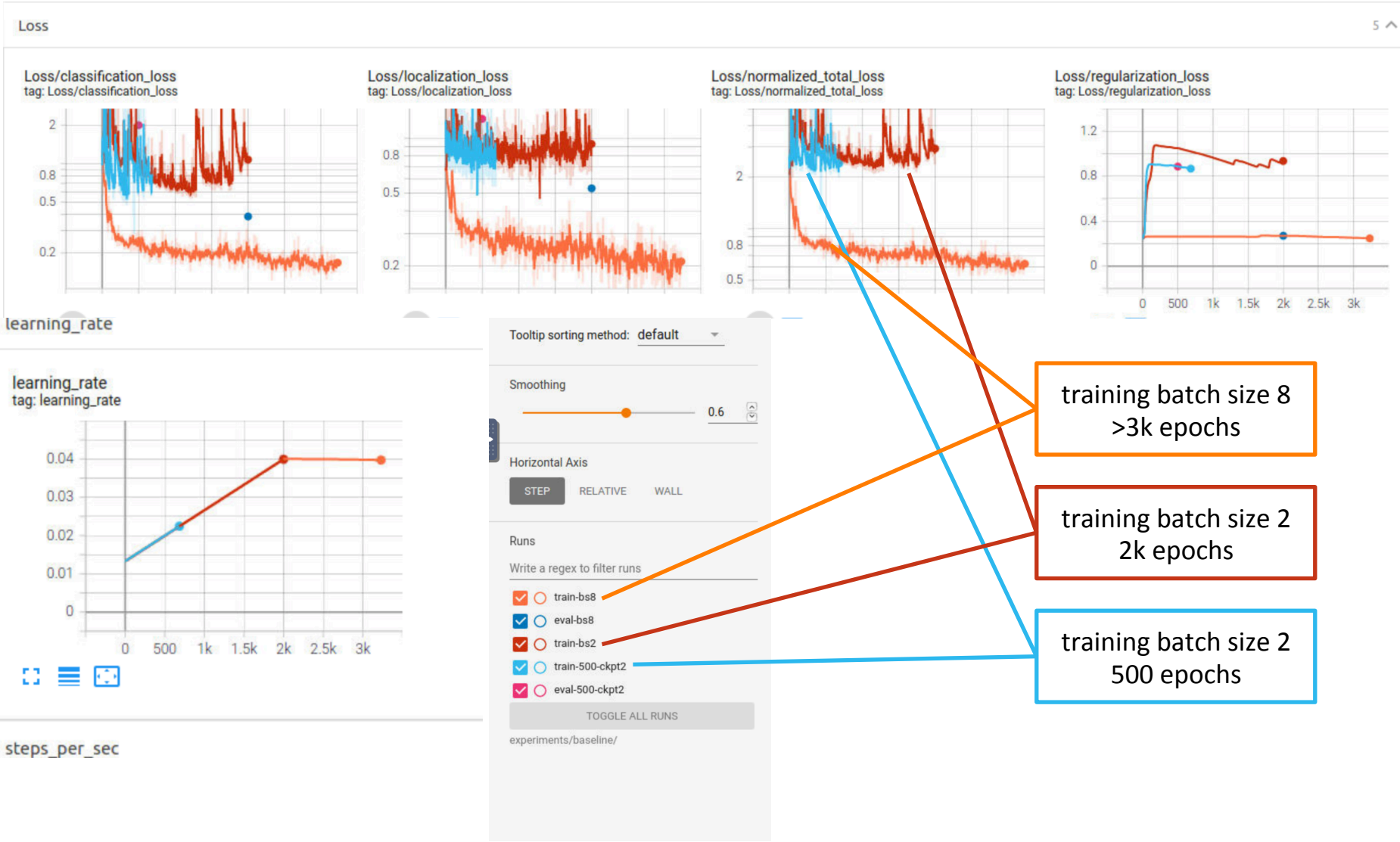
Besides label imbalance, it is worth noting a variety of lighting conditions due to weather, time of day. Blurriness to images is introduced by rainy conditions. Some target objects (cars, pedestrians, cyclists) are obstructed by non-target objects (e.g., trees).

Baseline and batch sizes

Convergence curves on the next slide correspond to the baseline configuration file, which can be retrieved from [nd013-c1-vision-starter]* (batch of 64 is suggested) or by running *“python edit_config.py … --batch_size 2 …”*. As seen from the next slide, batch size of 2 results in “rough” convergence curves, which do not seem to decrease with the epoch number and have a number of local maxima. According to [Google ML Crash Course]** batch size affects the amount of noise in the gradient. I decided to increase a number of examples used simultaneously for gradient calculations, i.e., batch size. Higher batch size results in more consistent derivatives within each epoch since more than 2 examples are seen simultaneously. Thus, derivatives vary less between examples within each epoch, and are more oriented towards consistent error minimization over the whole dataset.

* and ** - see references slide

Baseline and batch sizes



Baseline and batch sizes

As seen from the previous slide, batch size of 8 gives much more consistent convergence curves.

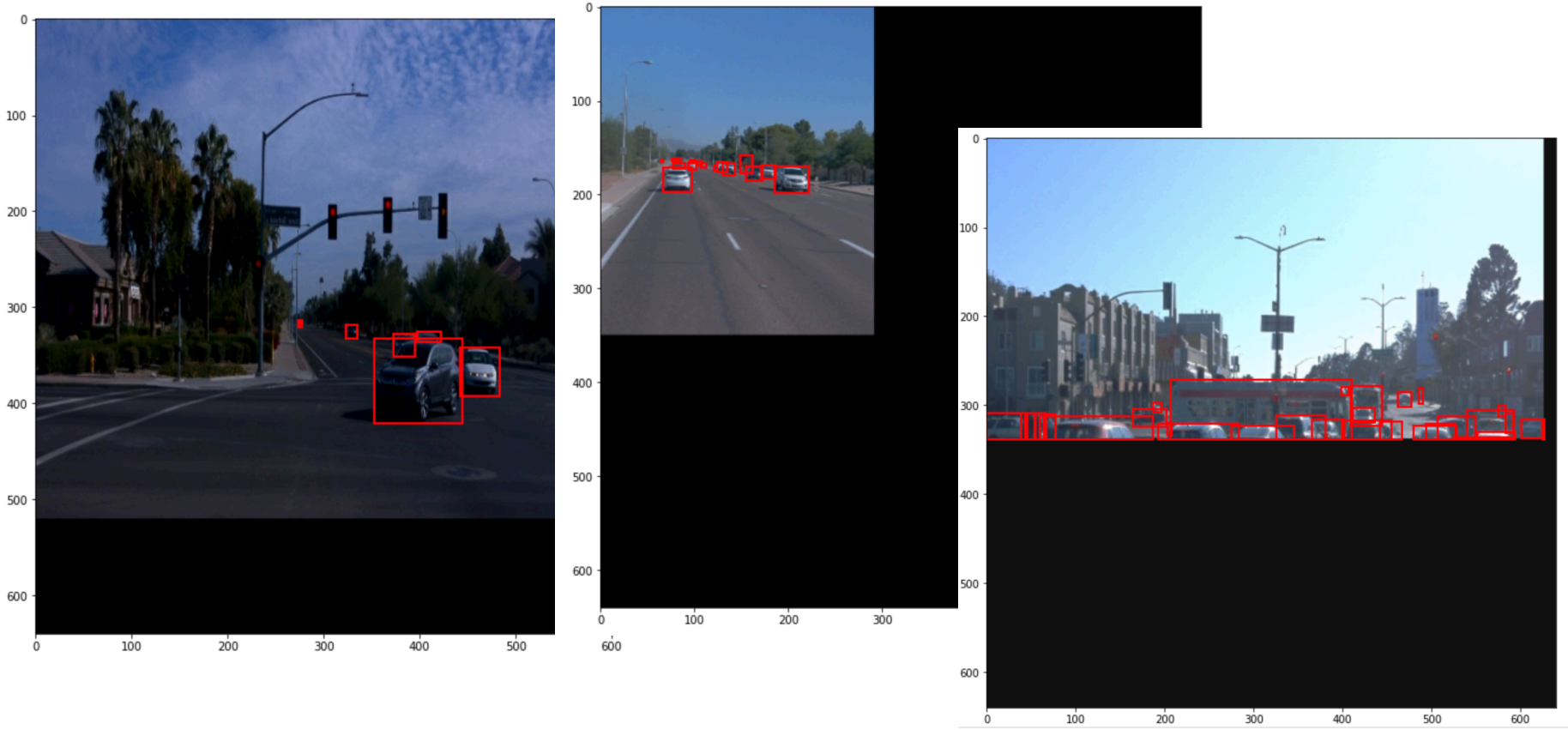
This is the maximum batch size I could use (16 and more would give me out of memory error). Moreover, the following warning already occurs for size of 8:

```
task:0/device:CPU:0',).
WARNING:tensorflow:From /data/virtual_envs/sdc-c1-gpu-augment/lib/python3.7/site-packages/tensorflow/python/util/deprecation.py:574: calling map_fn_v2 (from tensor
flow.python.ops.map_fn) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Use fn_output_signature instead
W0407 17:46:18.226597 140663290373888 deprecation.py:506] From /data/virtual_envs/sdc-c1-gpu-augment/lib/python3.7/site-packages/tensorflow/python/util/deprecation
.py:574: calling map_fn_v2 (from tensorflow.python.ops.map_fn) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Use fn_output_signature instead
2023-04-07 17:46:43.815710: W tensorflow/core/common_runtime/bfc_allocator.cc:246] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.09GiB with freed_by
_count=0. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2023-04-07 17:46:43.920863: W tensorflow/core/common_runtime/bfc_allocator.cc:246] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.14GiB with freed_by
_count=0. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
```

I was further proceeding with a batch size of 8.

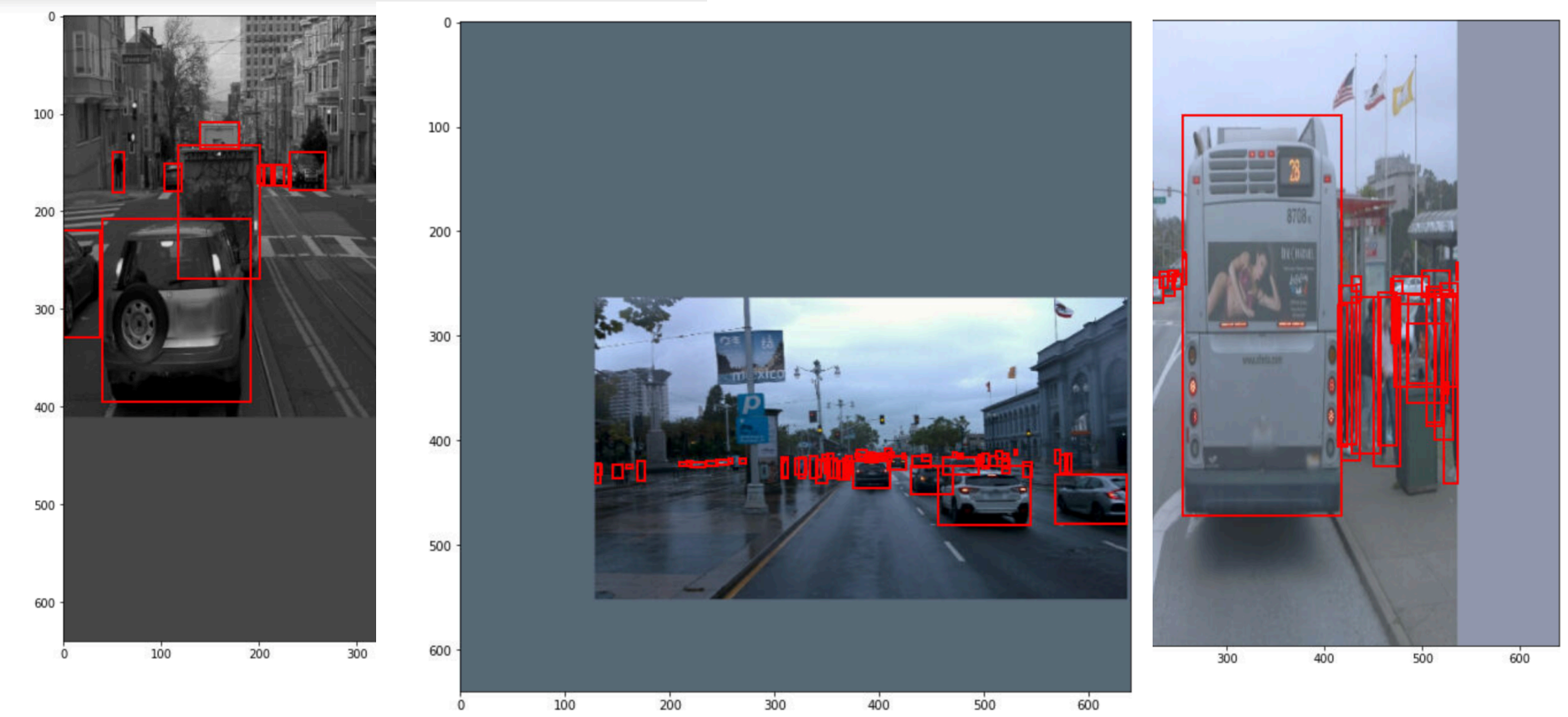
Exploring Augmentations

some augmented examples according to *augmentations_small_lr* experiment



Exploring Augmentations

some augmented examples according to *augmentations2_small_lr* experiment. Notice aspect ratio distortion (attempt to “zoom-out”; see further slides for details)



Exploring Augmentations

The following augmentations have been applied in various combinations:

- random_adjust_saturation/brightness/hue/constrast; random_rgb_to_gray − to emulate different lighting conditions
- random_black_patches − to emulate target object obstruction by non-target objects

I paid particular attention to the following augmentations: random_pad_image; ssd_random_crop_pad_fixed_aspect_ratio; and random_crop_image, which are discussed in Liu et al., (2016) on p. 12. See next slides for more details.

“Zoom In” and “Zoom Out”

Liu et al., (2016) highlight the importance of data augmentation for SSD model performance:

- “Without a follow-up feature resampling step as in Faster-RCNN, … classification … for small objects is relatively hard for SSD …”
- “The data augmentation strategy … helps to improve the performance”, where the data augmentation strategy involves original image sampling into patches of different aspect ratios and sizes. “ The random crops generated by the strategy can be thought of as a “zoom in” operation and can generate many larger training examples”

Then a “zoom out” operation is proposed to increase a number of small training examples: put an original image on a larger canvas and only then perform the crop. Authors claim more iterations are needed due to a larger number of augmentations. 2-3% mAP increase is achieved, which is not drastic, in my opinion, but “the new augmentation trick significantly improves the performance on small objects”.

“Zoom In” and “Zoom Out”

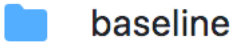
random_crop_image augmentation acts as a “zoom in”, which increases the number of larger objects.

I tried *ssd_random_crop_pad_fixed_aspect_ratio* augmentation, which randomly crops and pads an image with the parameters from Liu et al., (2016). I was hoping to see a prominent zoom-out effect. However, as seen in the notebook (slide 11), usually a big chunk of the image is cropped. Thus, the subsequent padding does not result in a noticeable zooming-out effect.

To explicitly perform zooming out, I tried *random_pad_image* (see slide 12). Adding pads to the original image result in its reduction in size, which achieves consistent zooming out. However, object shapes appear somewhat distorted.

EXPERIMENTS:
AUGMENTATIONS AND CONVERGENCES

Baseline Experiment



Augmentations:

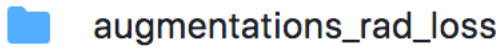
- random_horizontal_flip
- random_crop_image

Optimizer: momentum optimizer

Learning rate: warm-up: 0.01; base: 0.04

For convergence curves please see batch-size-8 metrics on one of the previous slides (slide 9). Although noisy, in my opinion, they look completely reasonable with increasing values throughout the epochs

Augmentations Rad Loss



Augmentations:

- random_horizontal_flip
- random_rgb_to_gray
- ssd_random_crop_pad_fixed_aspect_ratio
- random_jitter_boxes
- random_adjust_saturation/brightness/hue
- random_crop_image

Optimizer: momentum optimizer

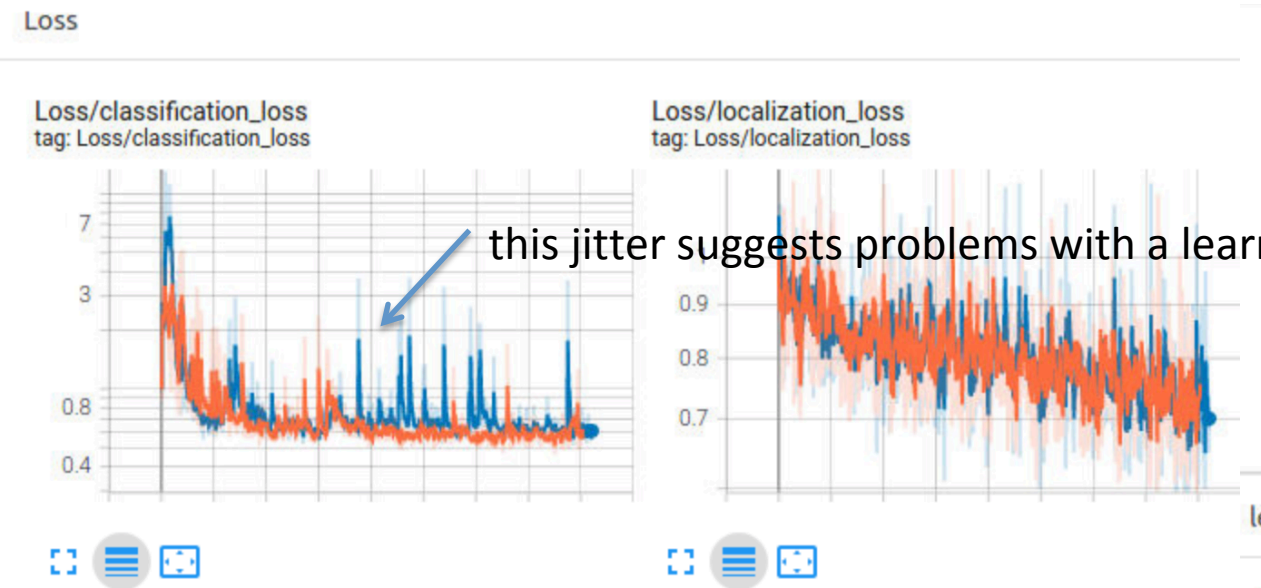
Learning rate: warm-up: 0.01; base: 0.04

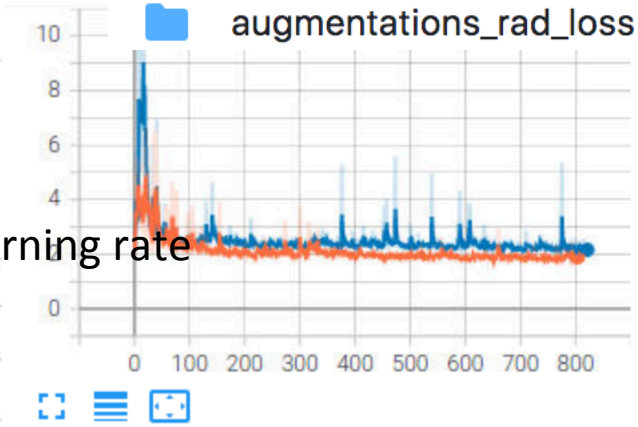
See convergence curves on the next slide

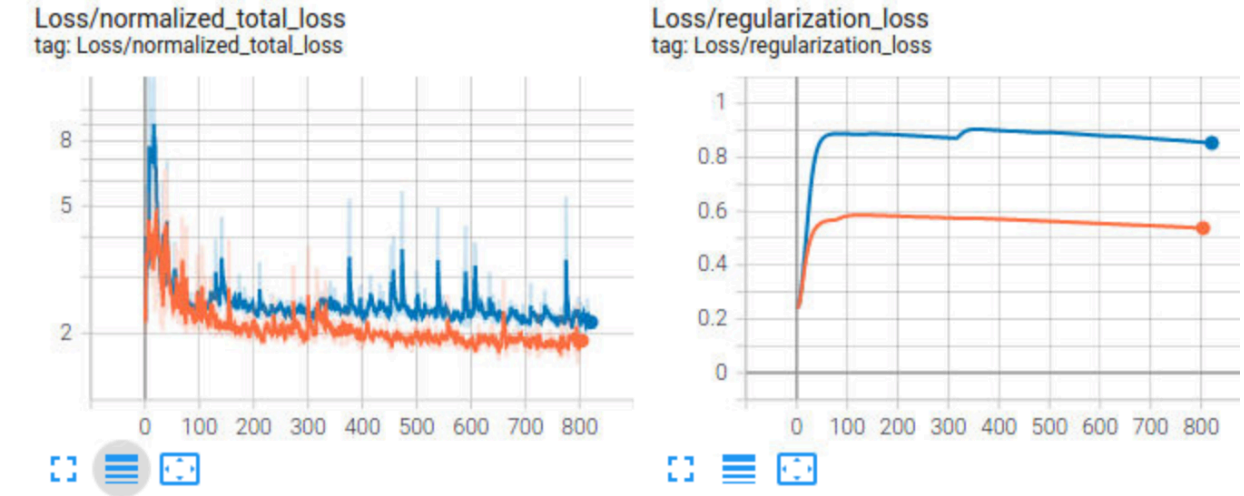
Since the convergence curves (for both runs) have a number of local maxima, I decrease the learning rate and stop both trial runs at about 800 epochs without proceeding

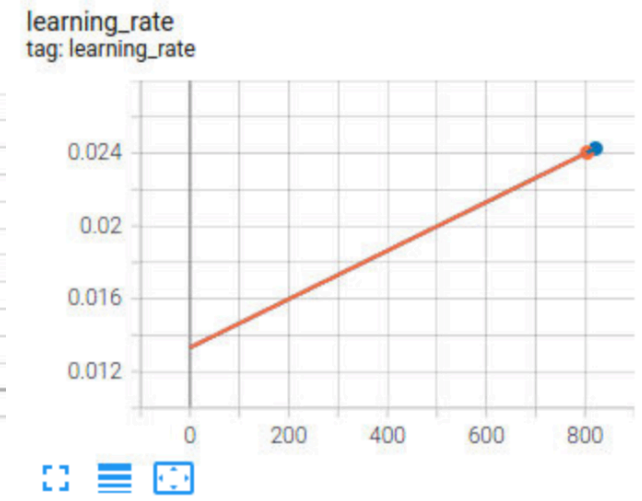
Augmentations Rad Loss

orange and blue correspond to two different runs (both with the same parameters)









Augmentations Small LR

Augmentations:

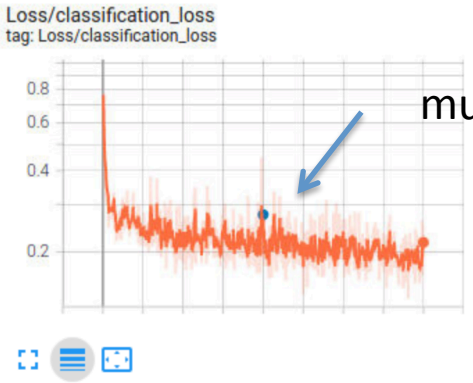
- random_horizontal_flip
- random_rgb_to_gray
- ssd_random_crop_pad_fixed_aspect_ratio
- ~~random_jitter_boxes~~
- random_adjust_saturation/brightness/hue
- random_crop_image

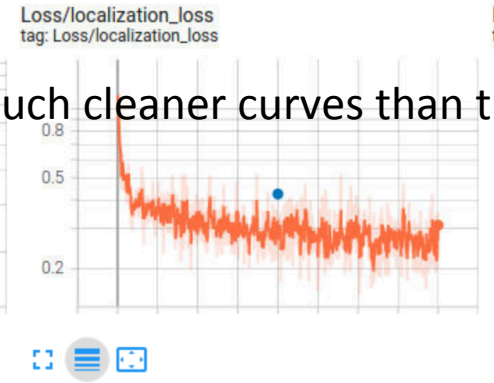
Optimizer: momentum optimizer

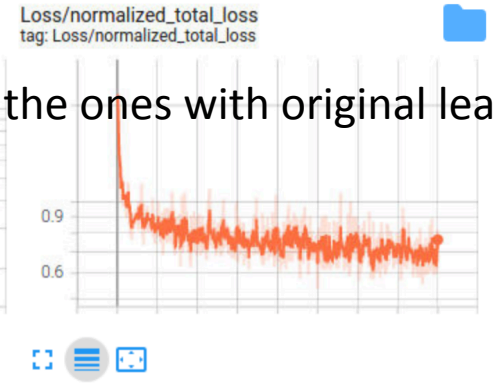
Learning rate: warm-up: 0.001; base: 0.004

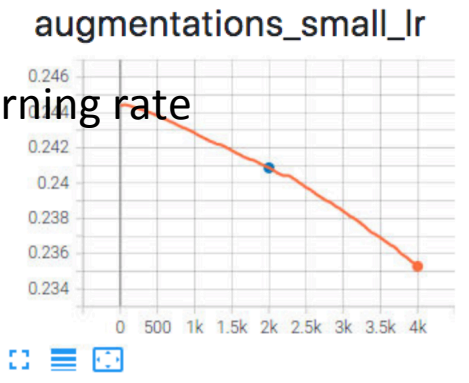
See convergence curves on the next slide. Although noisy, they look reasonable, in my opinion. Losses seem to be decreasing even at iteration of 4000

Augmentations Small LR

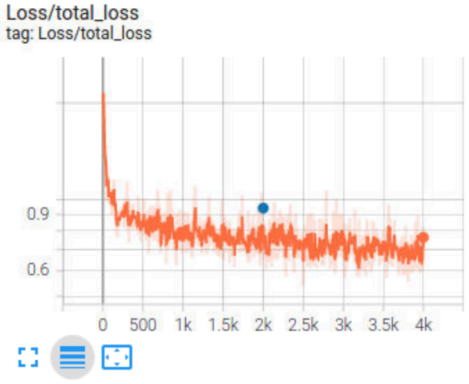


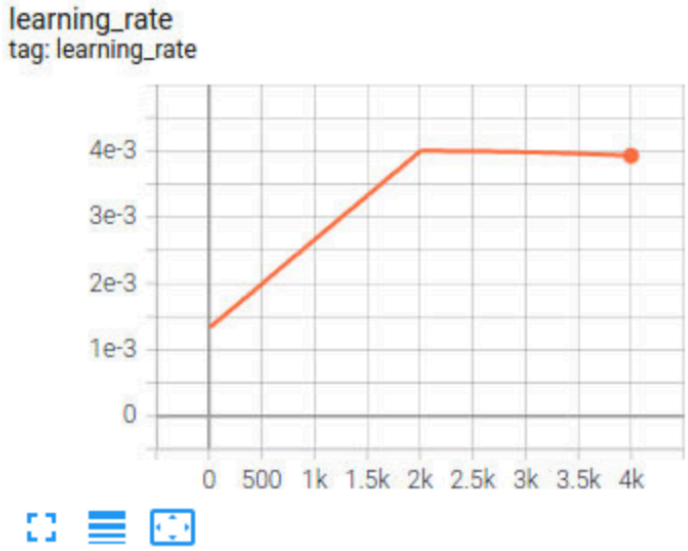






much cleaner curves than the ones with original learning rate





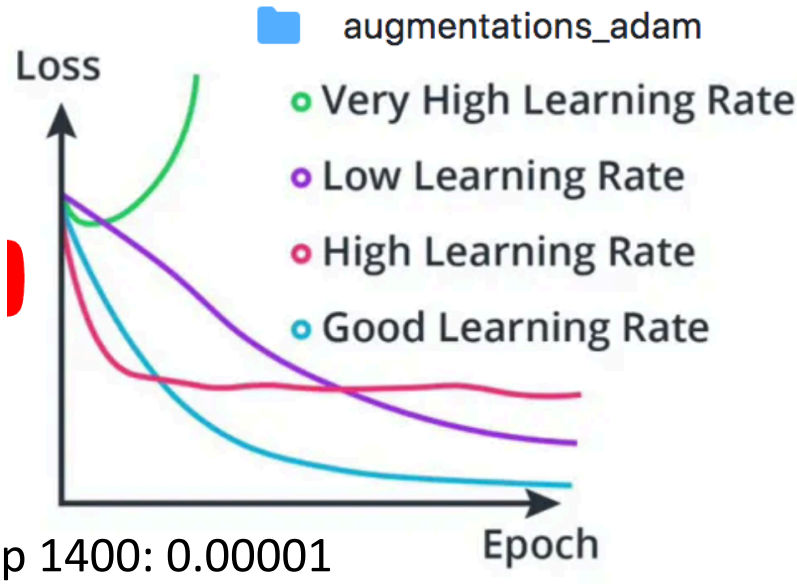
Augmentations Adam

Augmentations:

- random_horizontal_flip
- random_rgb_to_gray
- ssd_random_crop_pad_fixed_aspect_ratio
- ~~random_jitter_boxes~~
- random_adjust_saturation/brightness/hue
- random_crop_image

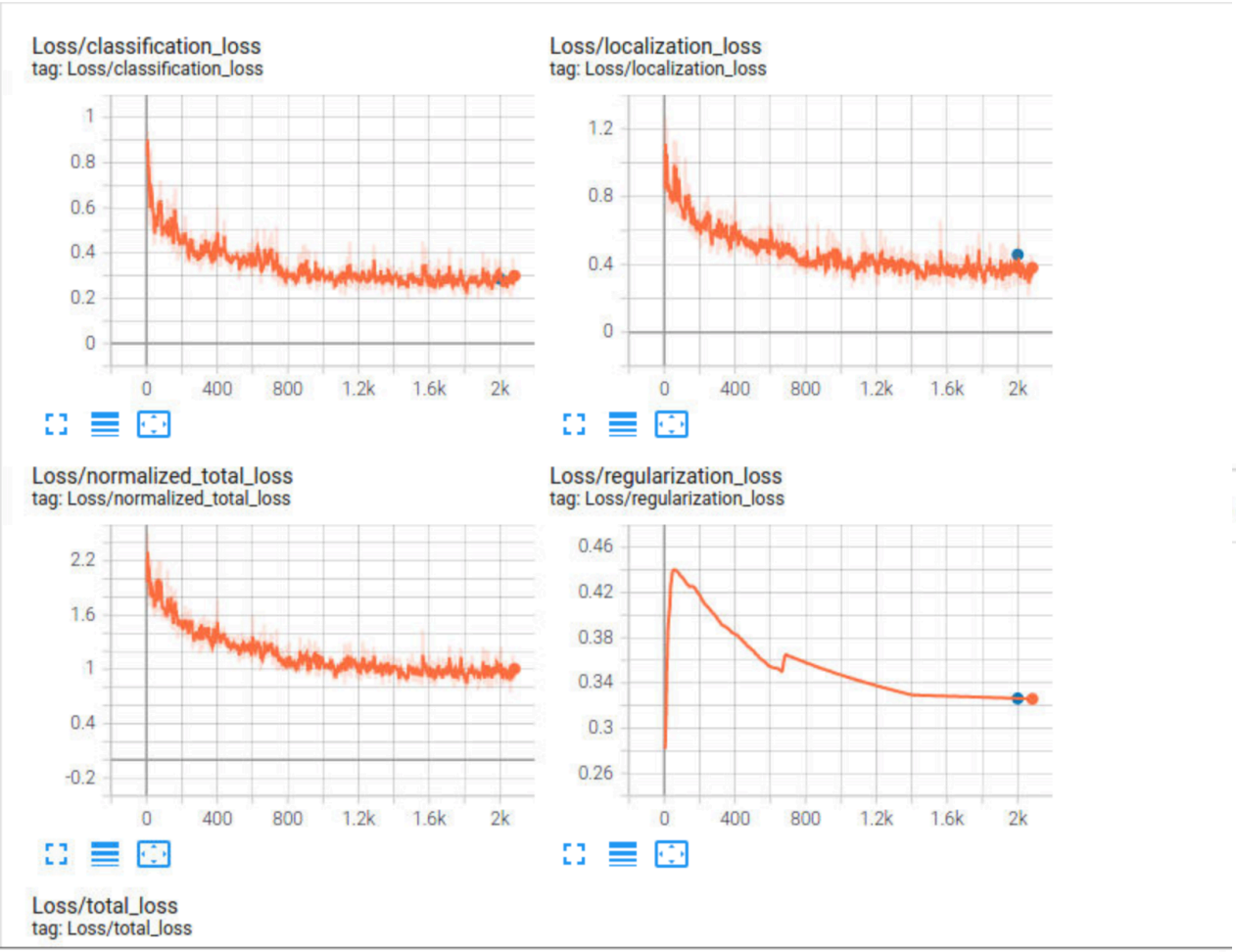
Optimizer: adam optimizer

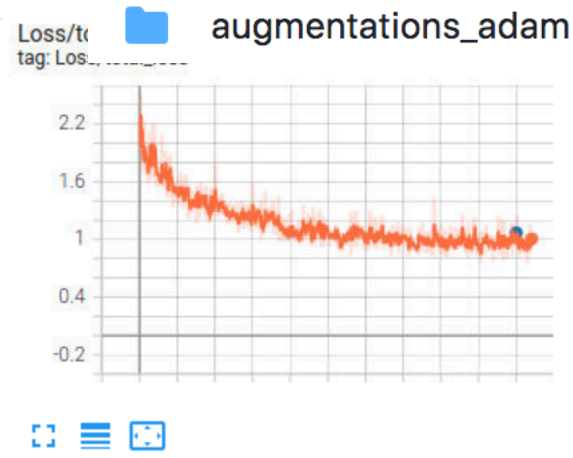
Learning rate: initial: 0.001; step 700: 0.0001; step 1400: 0.00001



See convergence curves on the next slide. Although noisy, they look reasonable, in my opinion. At epoch 2000, losses seem to be "stuck" at a level of ~1 (for normalized total loss). This behavior, in combination with loss shapes looking like a purple curve (see the cartoon from lesson 4, section 21), could be an indicator of a too big of a learning rate. At the same time, non-decreasing loss can be coming from regularization. Also, the Y-axis is not in log format as for other pipelines, which can bias the visual perception

Augmentations Adam





Augmentations2 Small LR


augmentations2_small_lr

Augmentations:

- random_horizontal_flip
- random_black_patches
- random_rgb_to_gray
- random_pad_image
- ~~ssd_random_crop_pad_fixed_aspect_ratio~~
- ~~random_jitter_boxes~~
- random_adjust_saturation/brightness/contrast*
- random_crop_image

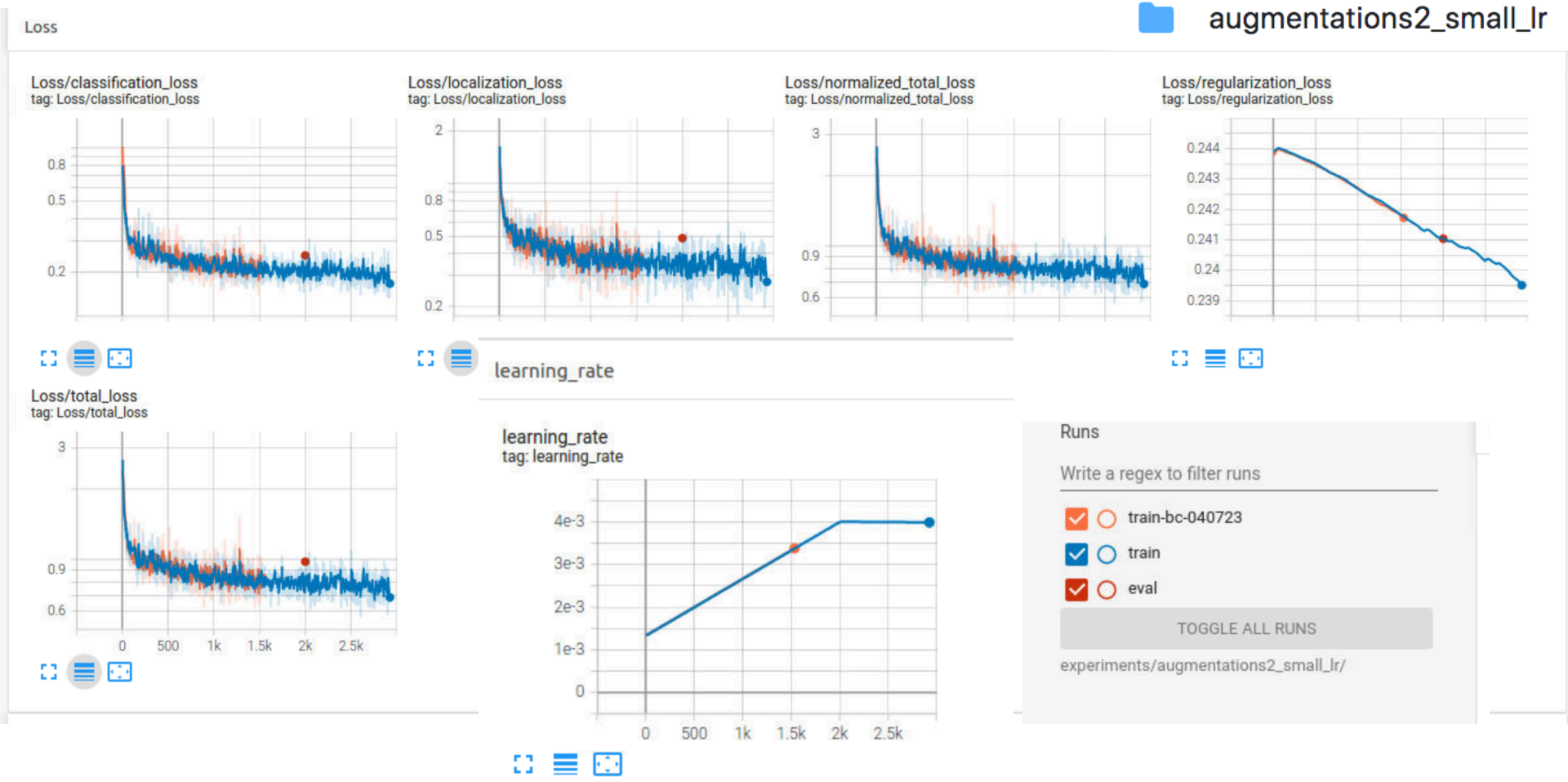
Optimizer: momentum optimizer

Learning rate: warm-up: 0.001; base: 0.004

See convergence curves on the next slide. Although noisy, they look reasonable, in my opinion. Losses seem to be decreasing even at iteration of 3000

*I later realized that hue adjustment in *augmentations_small_lr* was replaced by contrast adjustment. This could affect the performance of *augmentations2_small_lr* but most probably not drastically since displayed augmentations look similar (slides 11-12)

Augmentations2 Small LR



orange is just a shorter run (less epochs) with the same parameters

EXPERIMENTS:
RESULTS EVALUATION AND ANALYSIS

Evaluation Metrics and Cross-Validation

	A	B	C	D	E
1	**Metrics \ Experiments**	baseline with batch size 8	augmentations smaller learning rate	augmentations smaller learning rate with Adam	augmentations2 smaller learning rate
2	*all are approximate and evaluated at different epochs (when training stopped; more than 2000 epochs for all cases)*	about 3k epochs	about 4k epochs	about 2k epochs	about 3k epochs
3	**training class. loss**	0.19	0.2	0.3	0.2
4	**training loc. loss**	0.25	0.3	0.4	0.3
5	**training total loss**	0.6	0.7	1	0.7
6					
7	*evaluation is done at 2000 epoch*				
8	**evaluation class. loss**	0.386	0.275	0.286	0.248
9	**evaluation loc. loss**	0.531	0.425	0.457	0.488
10	**evaluation total loss**	1.186	0.941	1.07	0.978
11					
12	*approximate diff btw training and eval losses*				
13	**classification**	0.196	0.075	-0.014	0.048
14	**localization**	0.281	0.125	0.057	0.188
15					
16	**AP @ IoU 0.5:0.05:0.95 all**	0.081	0.095	0.079	0.091
17	**AP @ IoU 0.5/0.75 all**	0.170/ 0.071	0.188/ 0.090	0.164/ 0.075	0.176/ 0.089
18	**AP @ IoU 0.5:0.05:0.95 small/medium/large**	0.036/ 0.310/ 0.222	0.035/ 0.364/ 0.646	0.029/ 0.296/ 0.322	0.031/ **0.365**/ 0.482
19					
20	**AR @ IoU 0.5:0.05:0.95 max det 1/10/100 all**	0.020/ 0.087/ 0.131	0.023/ 0.093/ 0.135	0.019/ 0.089/ 0.137	0.024/ 0.097/ 0.132
21	**AR @ IoU 0.5:0.05:0.95 small/medium/large max det 100**	0.085/ 0.378/ 0.271	0.077/ 0.425/ 0.698	0.080/ 0.448/ 0.500	0.071/ **0.452**/ 0.495

Evaluation Metrics and Cross-Validation: baseline

	A	B	C	D	E
1	**Metrics \ Experiments**	baseline with batch size 8	sm		
2	*all are approximate and evaluated at different epochs (when training stopped; more than 2000 epochs for all cases)*	about 3k epochs			
3	**training class. loss**	0.19			
4	**training loc. loss**	0.25			
5	**training total loss**	0.6			
6					
7	*evaluation is done at 2000 epoch*				
8	**evaluation class. loss**	0.386			
9	**evaluation loc. loss**	0.531			
10	**evaluation total loss**	1.186			
11					
12	*approximate diff btw training and eval losses*				
13	**classification**	0.196			
14	**localization**	0.281			
15					
16	**AP @ IoU 0.5:0.05:0.95 all**	0.081			
17	**AP @ IoU 0.5/0.75 all**	0.170/ 0.071			
18	**AP @ IoU 0.5:0.05:0.95 small/medium/large**	0.036/ 0.310/ 0.222	0.0		
19					
20	**AR @ IoU 0.5:0.05:0.95 max det 1/10/100 all**	0.020/ 0.087/ 0.131	0.0		
21	**AR @ IoU 0.5:0.05:0.95 small/medium/large max det 100**	0.085/ 0.378/ 0.271	0.0		

Baseline has the lowest training classification and localization losses (0.19 and 0.25). However, evaluation losses are the largest among the pipelines, thus resulting in the largest gap btw training and evaluation metrics. This behavior suggests over-fitting occurring with the baseline pipeline

Observations: augmentations

After introducing augmentations the learning rate needs to be decreased, otherwise, as seen in *augmentations_rad_loss* pipeline, convergence curves will be highly irregular.

Once augmentations are introduced training losses increase, evaluation losses decrease, as well as the difference between them. Thus, augmentations help with the overfitting problem. Training losses increase can be associated with the need to train for more epochs when augmentations are introduced (see Liu et al., 2016, p.12) – "… more training images by introducing … data augmentation, we have to double the training iterations".

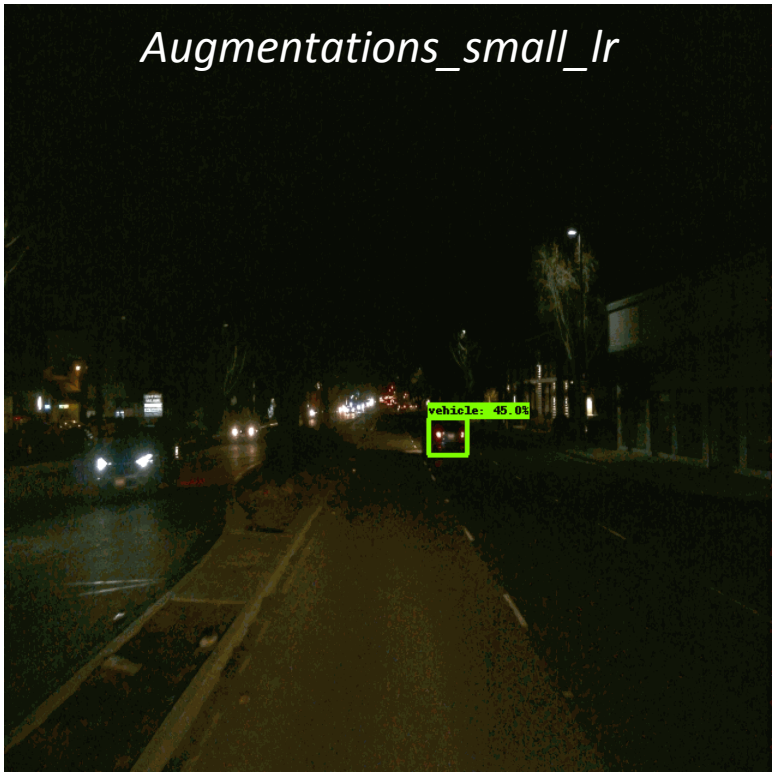
	A	B	C	D	E
1	**Metrics \ Experiments**	baseline with batch size 8	augmentations smaller learning rate	augmentations smaller learning rate with Adam	augmentations2 smaller learning rate
2	*all are approximate and evaluated at different epochs (when training stopped; more than 2000 epochs for all cases)*	about 3k epochs	about 4k epochs	about 2k epochs	about 3k epochs
3	**training class. loss**	0.19	0.2	0.3	0.2
4	**training loc. loss**	0.25	0.3	0.4	0.3
5	**training total loss**	0.6	0.7	1	0.7
6					
7	*evaluation is done at 2000 epoch*				
8	**evaluation class. loss**	0.386	0.275	0.286	0.248
9	**evaluation loc. loss**	0.531	0.425	0.457	0.488
10	**evaluation total loss**	1.186	0.941	1.07	0.978
11					
12	*approximate diff btw training and eval losses*				
13	**classification**	0.196	0.075	-0.014	0.048
14	**localization**	0.281	0.125	0.057	0.188

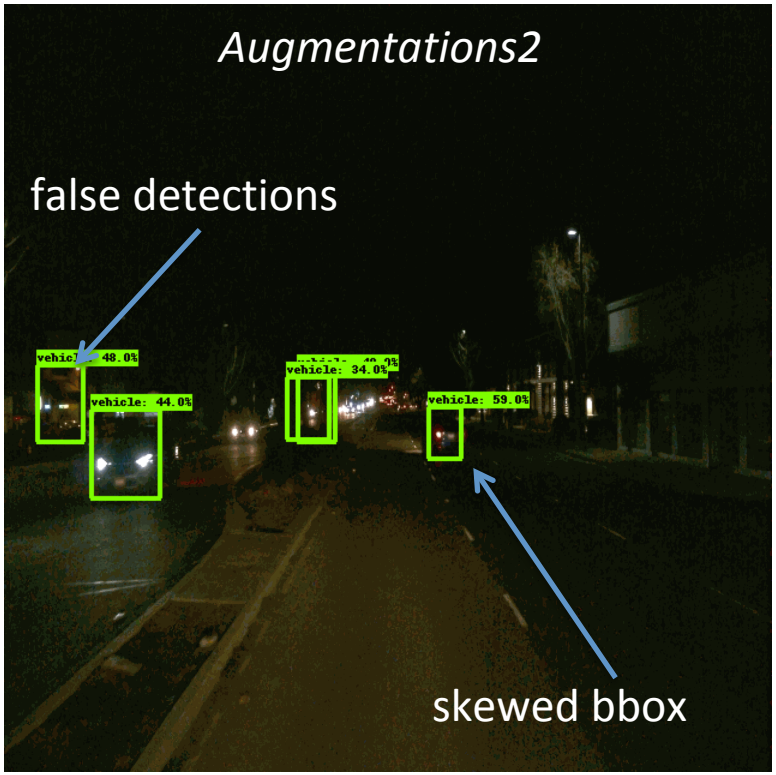
Observations

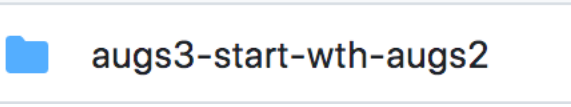
It turns out that *augmentations_adam* pipeline has the least amount of overfitting (smallest difference between training and evaluation losses). However, the best COCO metrics (among 2k epoch models) occur for *augmentations_small_lr* pipeline. Overall, mAP @ 0.5:0.05:0.95 is quite low for all the pipelines, but increases towards reasonable values for large objects (about 0.6). The same happens for mAR.

Adam optimizer in *augmentations_adam* (same augmentations as in *augmentations_small_lr*) does not help with the model performance.

Augmentations2_small_lr pipeline with “zooming-out” augmentations (*random_pad_image* to explicitly reduce the size of original images) overall does not show the best metrics. However, for objects of medium size the corresponding model has the highest performance (mAP@0.5:0.05:0.95 of 0.365 and mAR@0.5:0.05:0.95 of 0.452). This behavior highlights the capability of the “zooming-out” procedure to improve model performance on objects with scales lower than larger. Most probably, low performance comes from object shape distortion introduced by the inconsistent width and height padding. This is supported by the lowest classification score of *augmentations2* pipeline: objects are detected and classified while bounding-box distortion results in low IoU with objects present in the trip. See a particular frame from the testing trip on the next slide: *augmentations2* detects many more cars than *augmentations_small_lr* but the bounding boxes do not reflect true object dimensions. Some false detections can be present as well.


Augmentations_small_lr
vehicle: 45.0%


Augmentations2
false detections
48.0%
vehicle: 44.0%
vehicle: 34.0%
vehicle: 59.0%
skewed bbox



Augmentations3 (augs3-start-wth-augs2 folder): augmentations2 as a starting checkpoint

I really liked how many vehicle objects *augmentations2_small_lr* model has detected on the previous clip. It also has the highest performance on medium-sized objects. I decided to try to improve the bounding box quality output by *augmentations2* by using it as a *fine_tune_checkpoint* to the pipeline based on *augmentations_small_lr*. The resulting pipeline *augs3-start-wth-augs2* has the same parameters (including the set of augmentations (see slide 20)) as *augmentations_small_lr* except the starting checkpoint. Thus, the total number of iterations for the model is 4k. See next slide for convergence metrics.

Essentially, I remove *random_pad_image* and try recovering correct bbox shape while using the model weights exposed to zooming-out operation.

Augmentations2 2k epoch checkpoint exists in the exported folder, which can be found here:

https://drive.google.com/drive/folders/1oOOhnG_BqQTSs-cGAWOwWGdQD5OHZ62r?usp=sharing

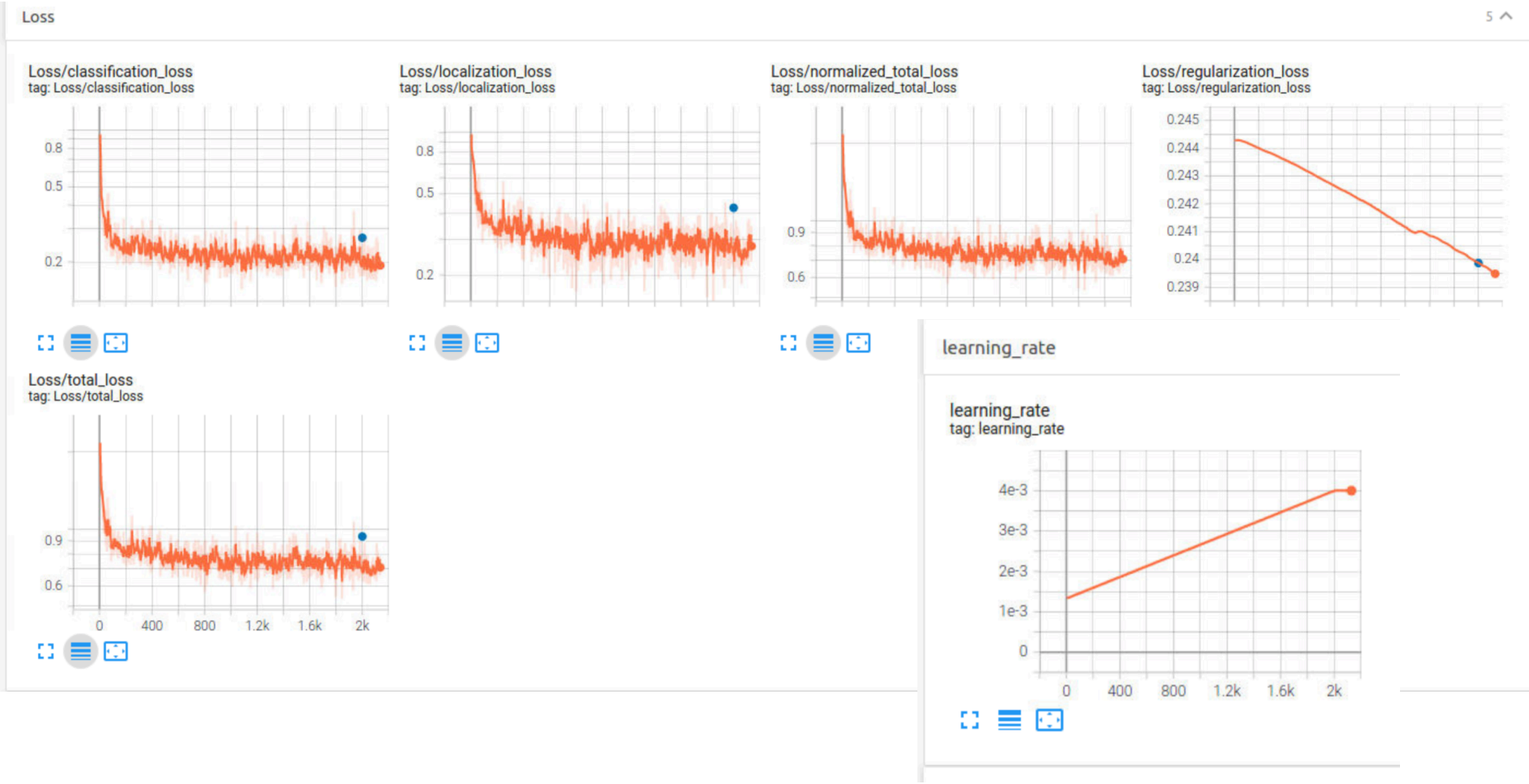
and should be placed in the folder

nd013-c1-vision-starter/augs3-start-wth-augs2/exported

before executing the corresponding *augs3-start-wth-augs2* pipeline

also see https://github.com/prostdfrost/cv-urban-env-dfrost/blob/main/experiments/augs3-start-wth-augs2/pipeline_augmentations.config

augs3-start-wth-augs2



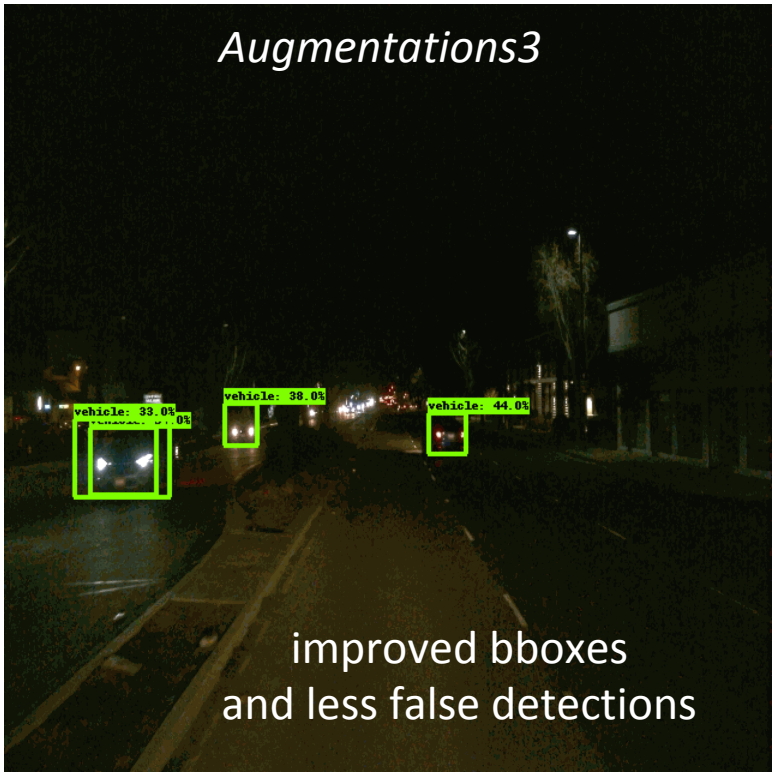
augs3-start-wth-augs2 vs other pipelines

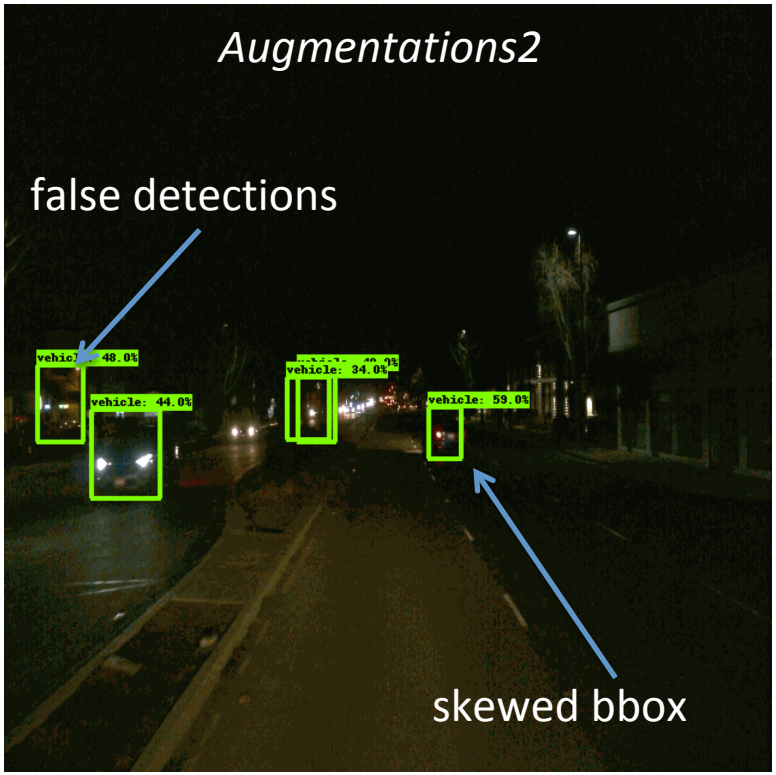
	A	B	C	D	E	F
1	**Metrics \ Experiments**	baseline with batch size 8	augmentations smaller learning rate	augmentations smaller learning rate with Adam	augmentations2 smaller learning rate	augmentations3
2	*all are approximate and evaluated at different epochs (when training stopped; more than 2000 epochs for all cases)*	about 3k epochs	about 4k epochs	about 2k epochs	about 3k epochs	2k epochs
3	**training class. loss**	0.19	0.2	0.3	0.2	0.2
4	**training loc. loss**	0.25	0.3	0.4	0.3	0.29
5	**training total loss**	0.6	0.7	1	0.7	0.685
6						
7	*evaluation is done at 2000 epoch*					
8	**evaluation class. loss**	0.386	0.275	0.286	0.248	0.268
9	**evaluation loc. loss**	0.531	0.425	0.457	0.488	0.428
10	**evaluation total loss**	1.186	0.941	1.07	0.978	0.936
11						
12	*approximate diff btw training and eval losses*					
13	**classification**	0.196	0.075	-0.014	0.048	0.068
14	**localization**	0.281	0.125	0.057	0.188	0.138
15						
16	**AP @ IoU 0.5:0.05:0.95 all**	0.081	0.095	0.079	0.091	0.108
17	**AP @ IoU 0.5/0.75 all**	0.170/ 0.071	0.188/ 0.090	0.164/ 0.075	0.176/ 0.089	0.226/ 0.093
18	**AP @ IoU 0.5:0.05:0.95 small/medium/large**	0.036/ 0.310/ 0.222	0.035/ 0.364/ 0.646	0.029/ 0.296/ 0.322	0.031/ **0.365**/ 0.482	0.048/ 0.413/ 0.381
19						
20	**AR @ IoU 0.5:0.05:0.95 max det 1/10/100 all**	0.020/ 0.087/ 0.131	0.023/ 0.093/ 0.135	0.019/ 0.089/ 0.137	0.024/ 0.097/ 0.132	0.025/ 0.114/ 0.167
21	**AR @ IoU 0.5:0.05:0.95 small/medium/large max det 100**	0.085/ 0.378/ 0.271	0.077/ 0.425/ 0.698	0.080/ 0.448/ 0.500	0.071/ **0.452**/ 0.495	0.106/ 0.510/ 0.670
22						

augs3-start-wth-augs2

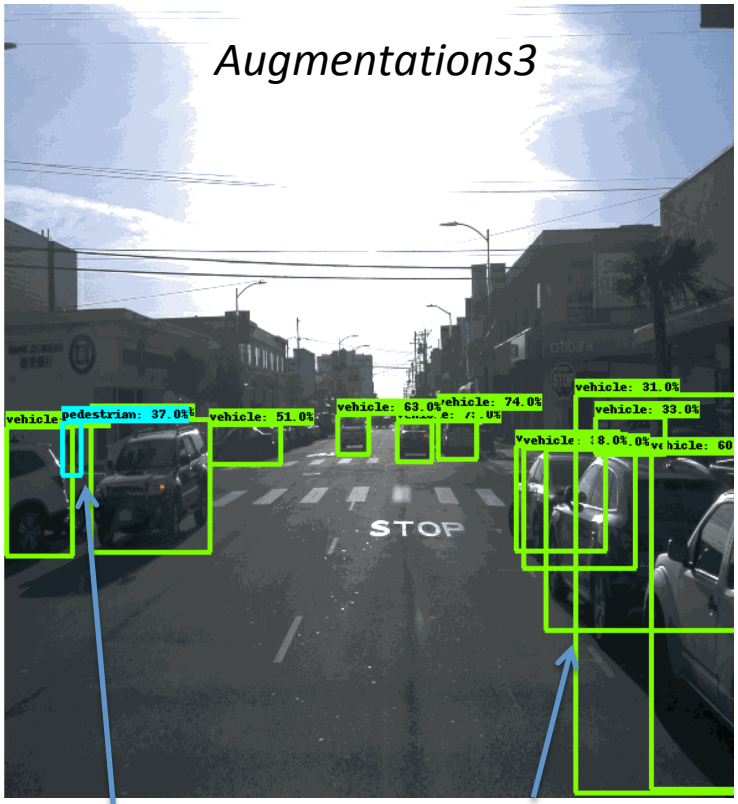
Has one of the lowest differences between training and validation losses (previous slide, rightmost column). The main COCO metric mAP @ 0.5:0.05:0.95 for all objects is the highest (although still pretty low − 0.108). AP and AR for medium-sized objects are the highest. At the same time, the performance for large objects dropped a little bit (AP of 0.381). Except the latter, overall metrics are higher than in the parent pipeline - *augmentations2*.

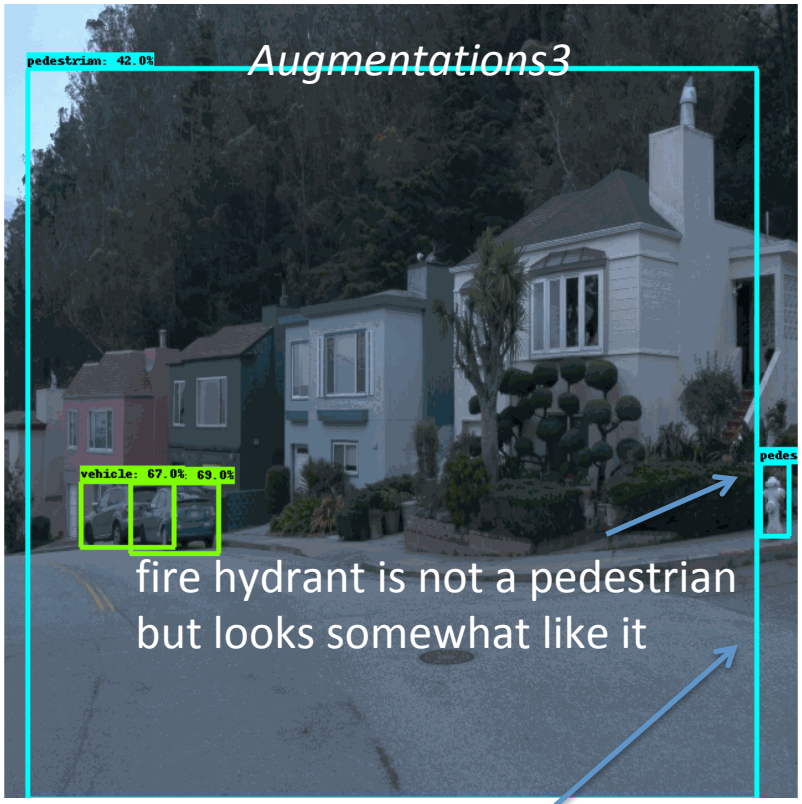
Augmentations3 vs Augmentations2





Augmentations3 on other trips





overall good performance
but multiple boxes per car

perhaps a real pedestrian
but heavily obstructed

from time to time a really big
“pedestrian” object occurs

augs3-start-wth-augs2

Based on test datasets, it indeed can be seen that a better performance than *augmentations2* is achieved (as supported by the metrics). Low AP for large objects can come from false detections (same as for *augmentations2*). Specifically model detects large objects classified by pedestrians, which can reduce large object AP even more. The cause of “giant pedestrian” detections is worth investigating more. As a temporal solution, pedestrian detections with unreasonably large size can be filtered out at a post-processing stage. It also should be noted, that while bounding boxes look more reasonable, a part of the performance gains can come from running more epochs (4k overall).

Summary and Conclusions

- Augmentations are crucial to achieve somewhat reasonable performance compared to a baseline
- Augmentations should be carefully chosen:
  - *ssd_random_crop_pad_fixed_aspect_ratio* does not give a noticeable zoom-out effect, while *random_pad_image* does but distorts bboxes
- We indeed need to babysit losses
- *Augmentations_small_lr* results in best COCO metrics among the other experiments conducted with 2k epochs
- *Augmentations3* allows recovering bounding box distortion introduced in *augmentations2* (mostly by random_pad_image) and overall has the best AP and detections for medium-size objects

Future research

- Running more epochs (more compute resources required); as of now most evaluations are done at 2k epoch (just when the optimizer has finished its warm-up steps)
- Tracking objects across different frames to allow for smoother predictions
- Include augmentations for raining conditions – blurriness
- Implement/utilize a shape-preserving zooming-out augmentation for mAP improvement on smaller objects

References

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg., 2016: SSD: Single Shot Multibox Detector, Proceedings of Computer Vision-ECCV, 14th European Conference, Part I 14, pp. 21-37

nd013-c1-vision-starter:

https://github.com/udacity/nd013-c1-vision-starter/blob/main/pipeline.config

Google ML Crash Course:
https://developers.google.com/machine-learning/crash-course/reducing-loss/stochastic-gradient-descent

References

Videos:

augmentations_small_lr trip1 https://drive.google.com/file/d/1WCy3ZlEaYPp4rbUW-gfrEU-ufVM-_PgD/view?usp=sharing

augmentations_adam trip1 https://drive.google.com/file/d/1T_aMvhPCZbglY-Q8SlgkjnYDmvtlLoXp/view?usp=sharing

augmentations2_small_lr trip1 https://drive.google.com/file/d/1WojUGaBTomwBbP2-JYWB87b8mk2baKqG/view?usp=sharing

augs3-start-wth-augs2 (augmentations3)

trip1 https://drive.google.com/file/d/1Z3QQUrCXrqiGBfzNEKruHJNagz-i2OFx/view?usp=sharing

trip2 https://drive.google.com/file/d/1YQsArsb3e1ykjk2v3ix9FGV-DbLv2z52/view?usp=sharing

trip3 https://drive.google.com/file/d/1dDozkGdeHGYsY-02GhwNSmykD3eFSOlt/view?usp=sharing