FADE: A Task-Agnostic Upsampling Operator for Encoder-Decoder Architectures

Hao Lu, Wenze Liu, Hongtao Fu, Zhiguo Cao

CMPE593 Term Project
Progress Presentation
Kutay Eroğlu

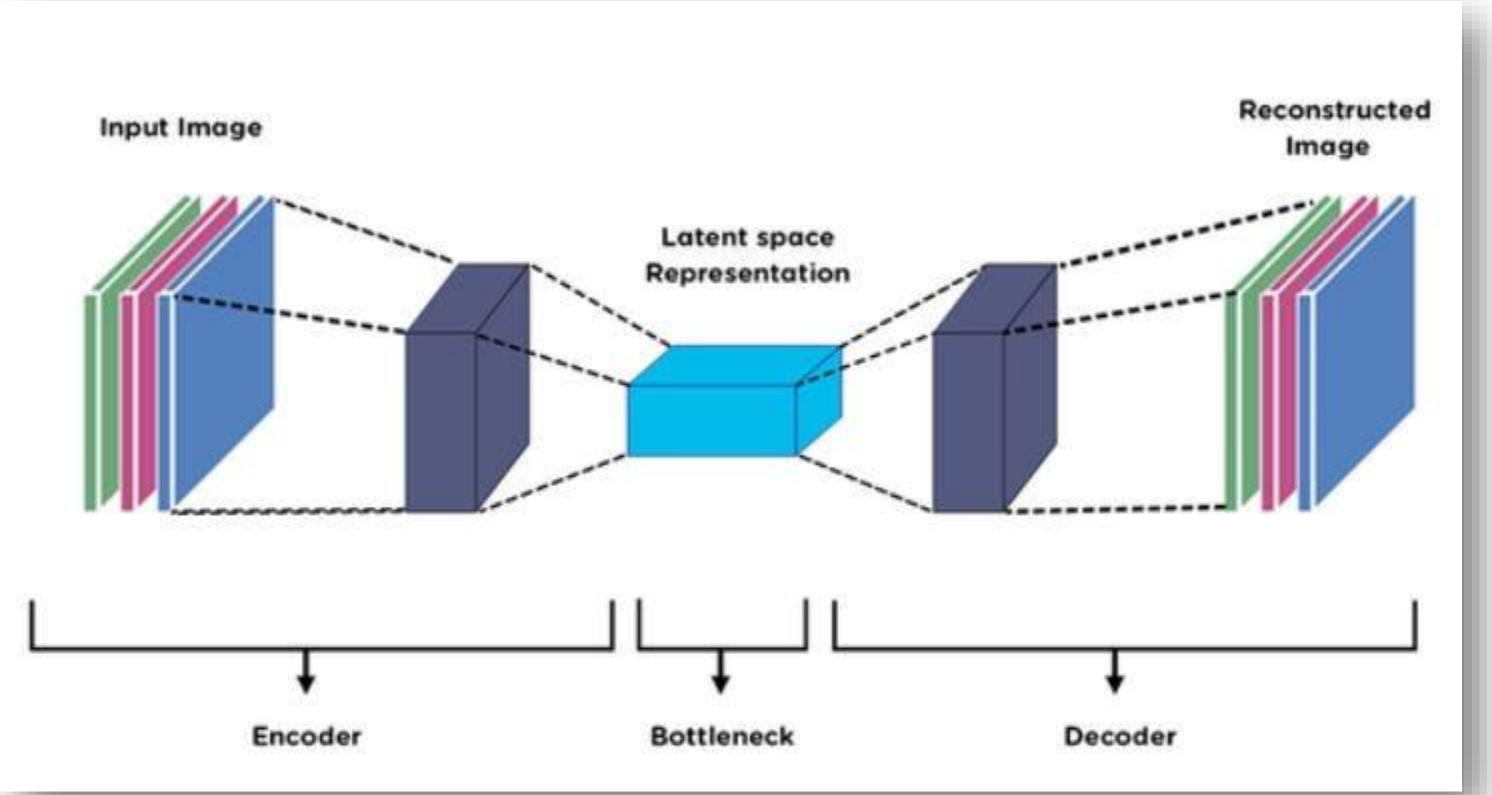
Published in
International Journal
of Computer Vision (IJCV)
22 July 2024

Problem

- No task-agnostic upsampling operators exist for encoder-decoder architectures.

Problem

- No task-agnostic upsampling operators exist for encoder-decoder architectures.



Problem

- No task-agnostic upsampling operators exist for encoder-decoder architectures.

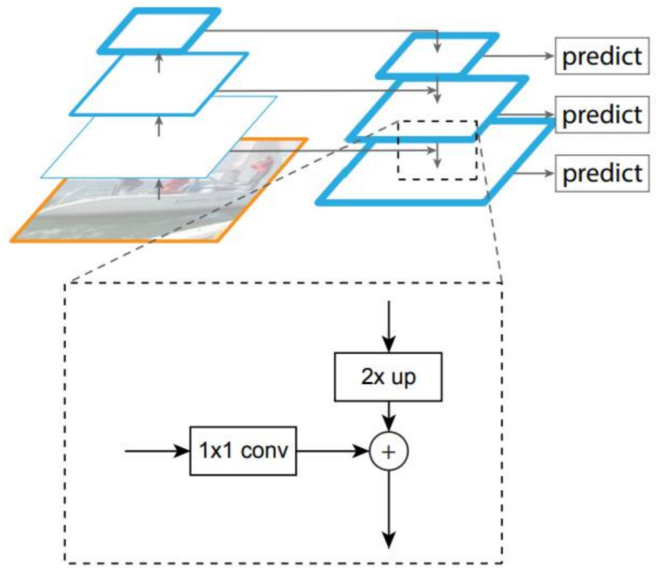


Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.

Feature Pyramid Networks for Object Detection*

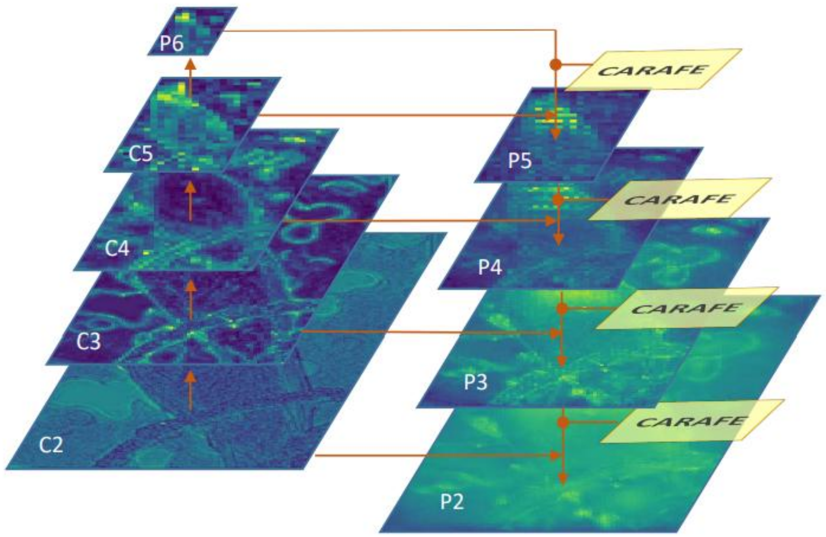
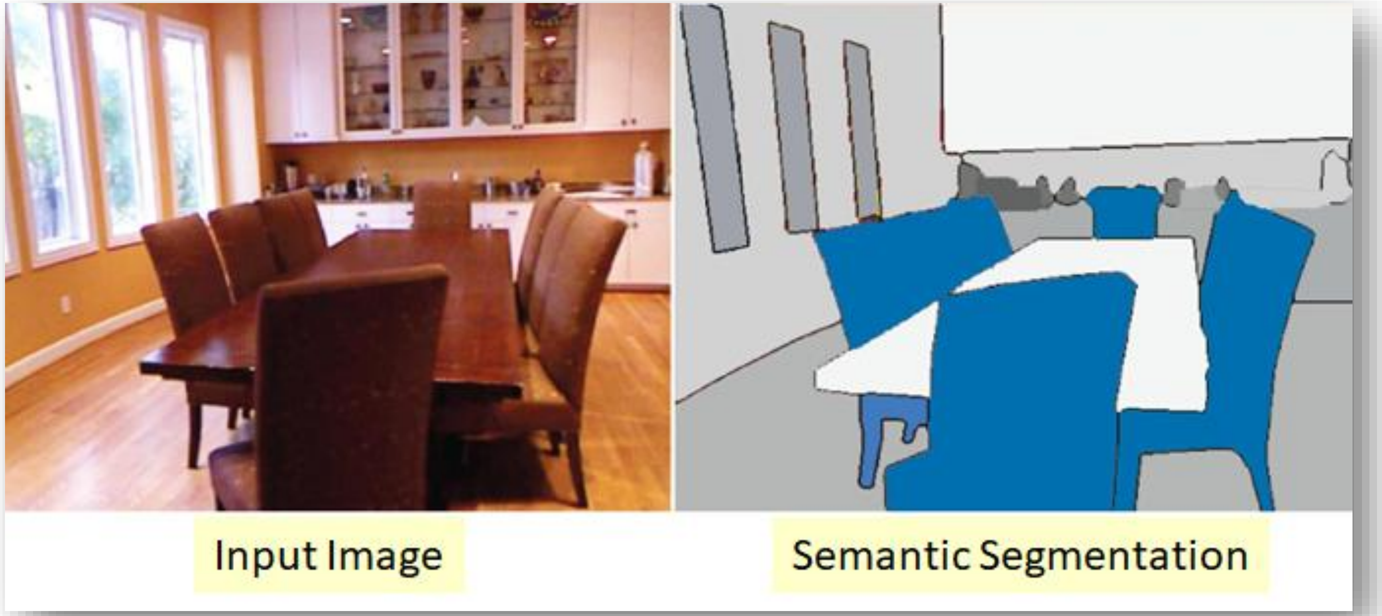


Figure 3: **FPN architecture with CARAFE**. CARAFE upsamples a feature map by a factor of 2 in the top-down pathway. It is integrated into FPN by seamlessly substituting the nearest neighbor interpolation.

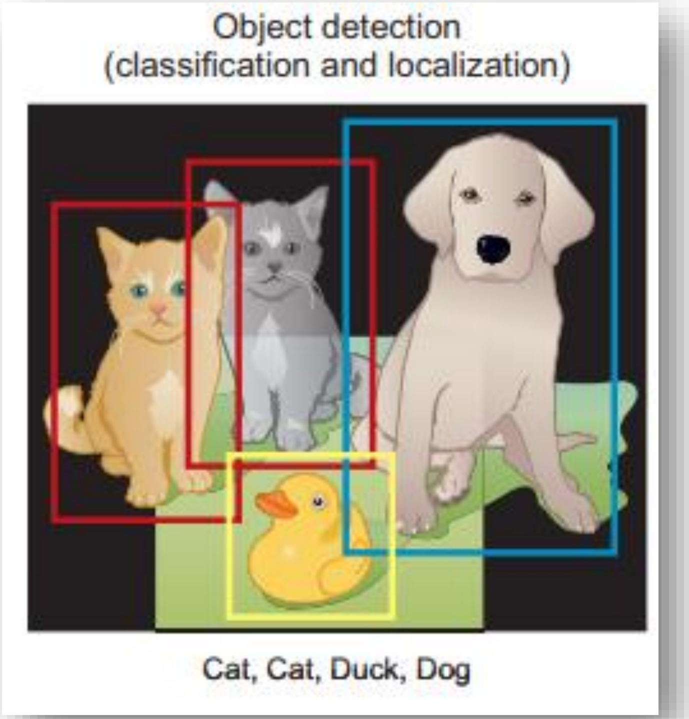
CARAFE: Content-Aware ReAssembly of FEatures*

Problem

- No task-agnostic upsampling operators exist for encoder-decoder architectures.

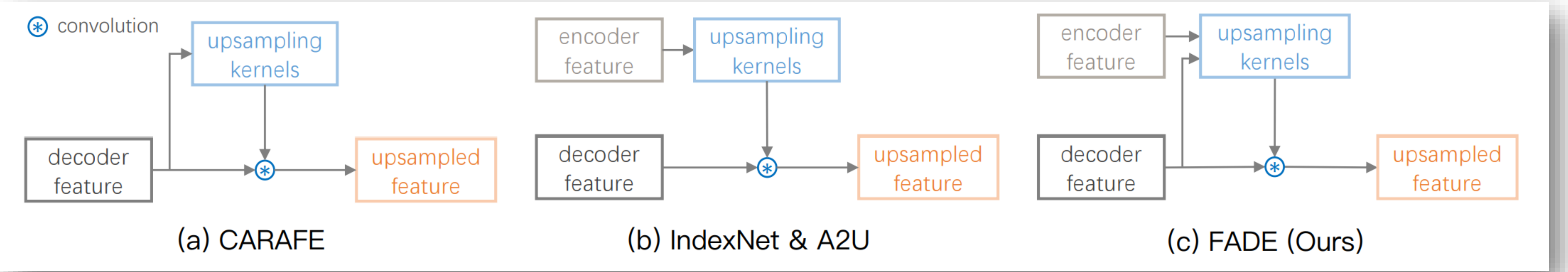


Example of a semantic segmentation task



Example of an object detection task

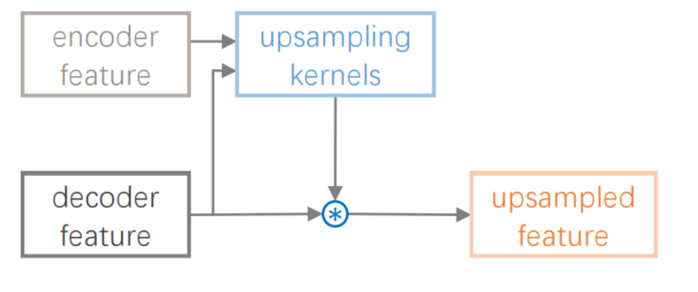
Solution: Main Difference
Dynamic Feature Upsampling



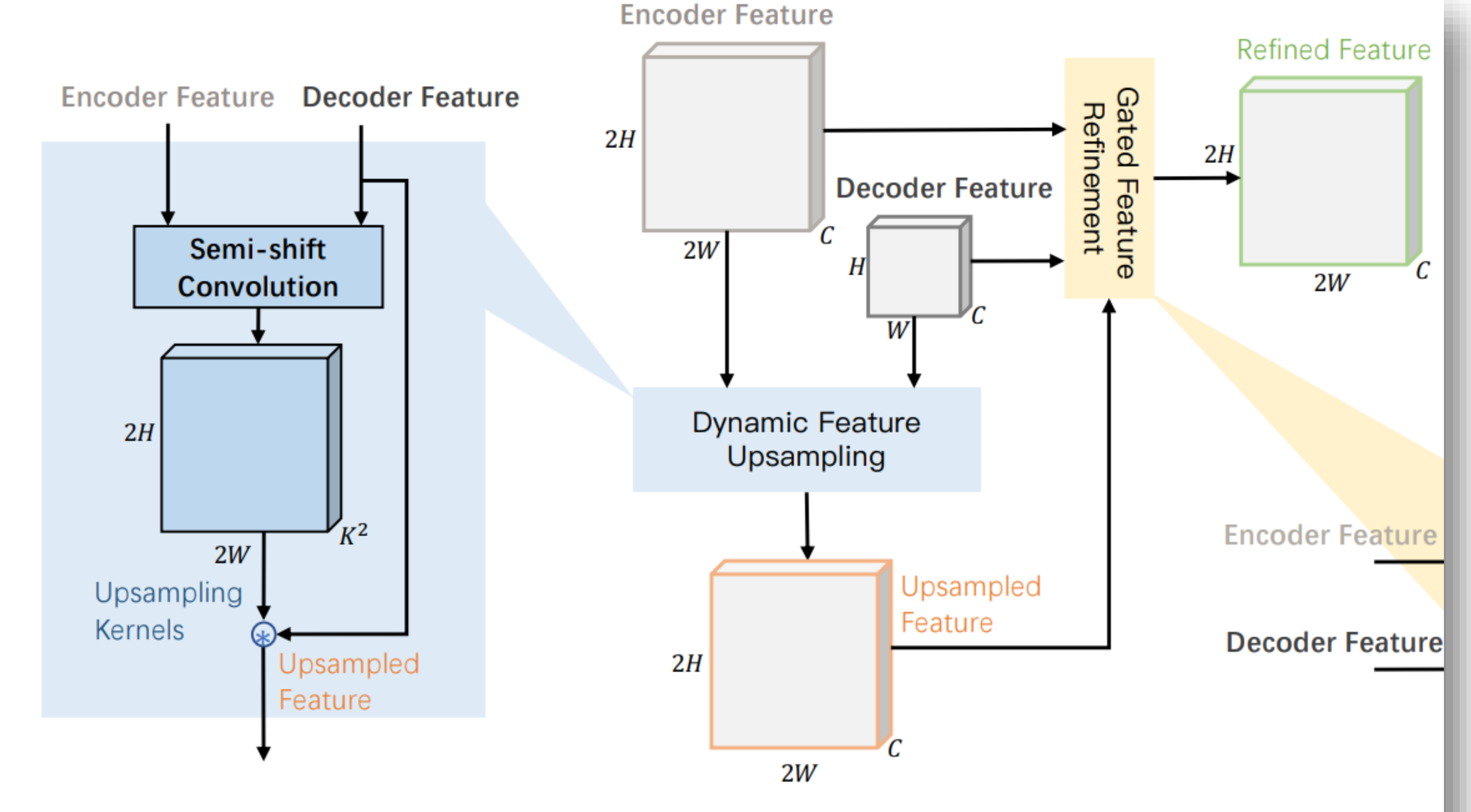
Main difference between dynamic upsampling operators on the use of encoder and/or decoder features

Overview of FADE
Dynamic Feature Upsampling

```
self.kernel_generator = SemiShift(in_channels_en, in_channels_de,
                                  up_kernel_size=up_kernel_size, scale=scale)
```

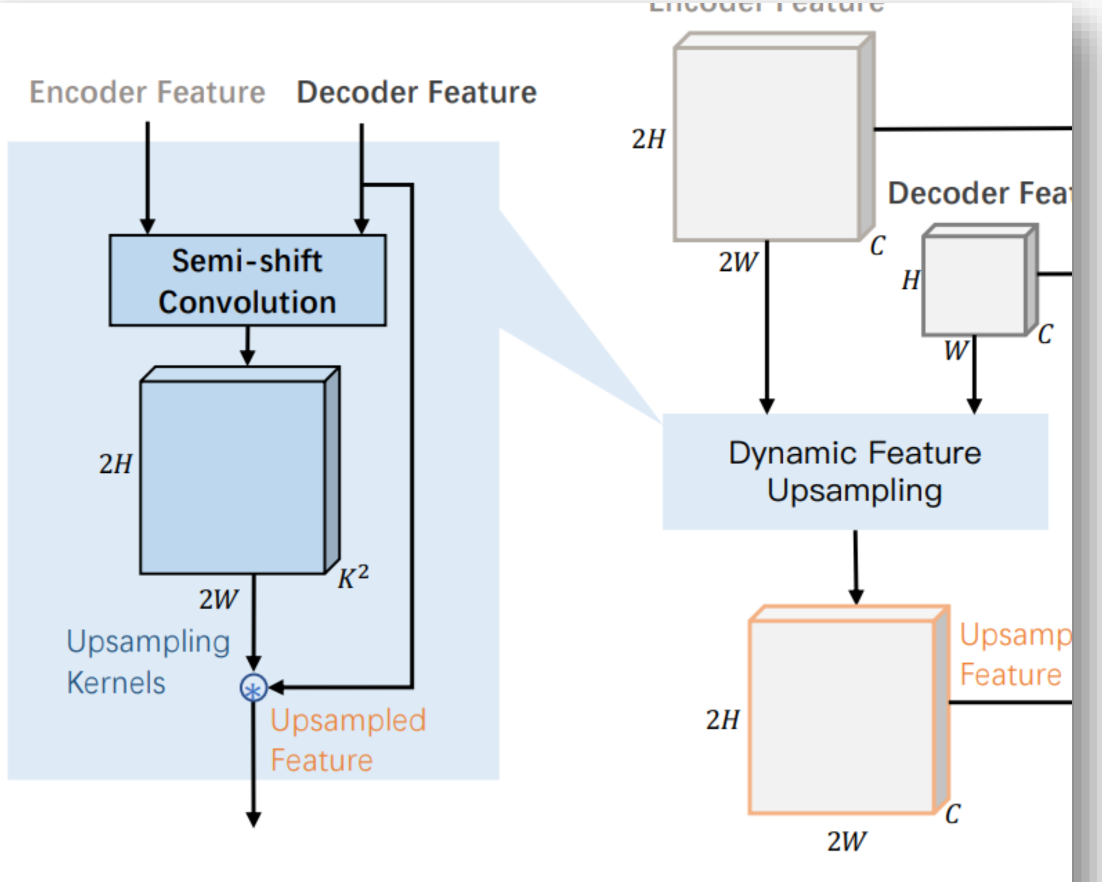


(c) FADE (Ours)



(a) Dynamic Feature Upsampling (b) Overview of FADE

Dynamic Feature Upsampling
Semi-shift Convolution



(a) Dynamic Feature Upsampling (b) Overview of FA

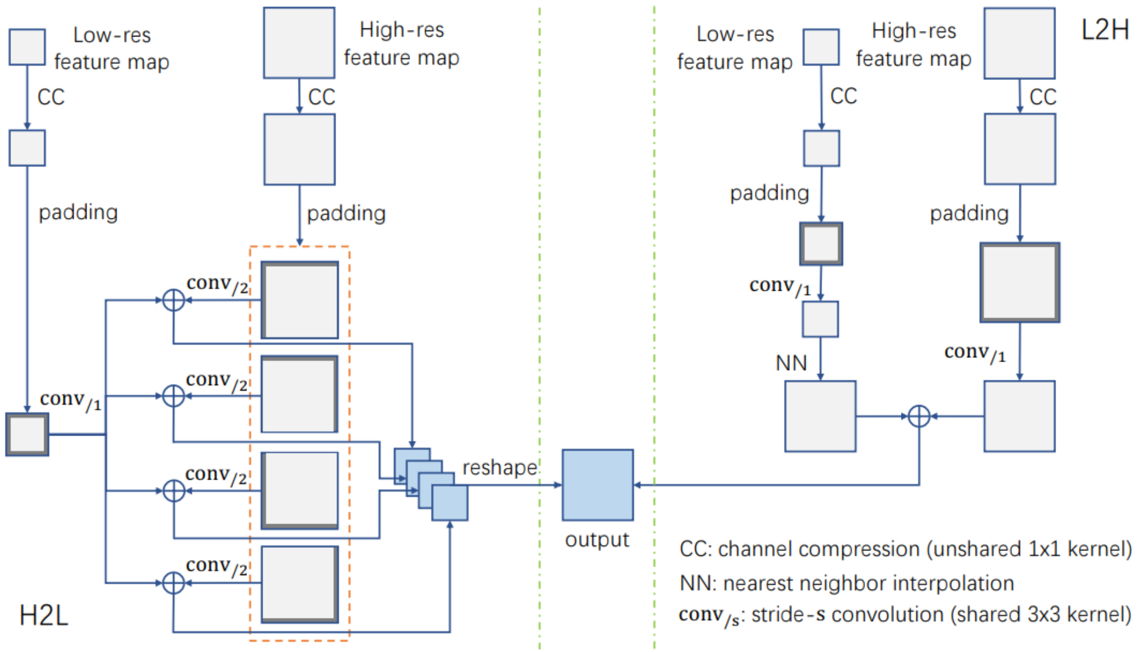
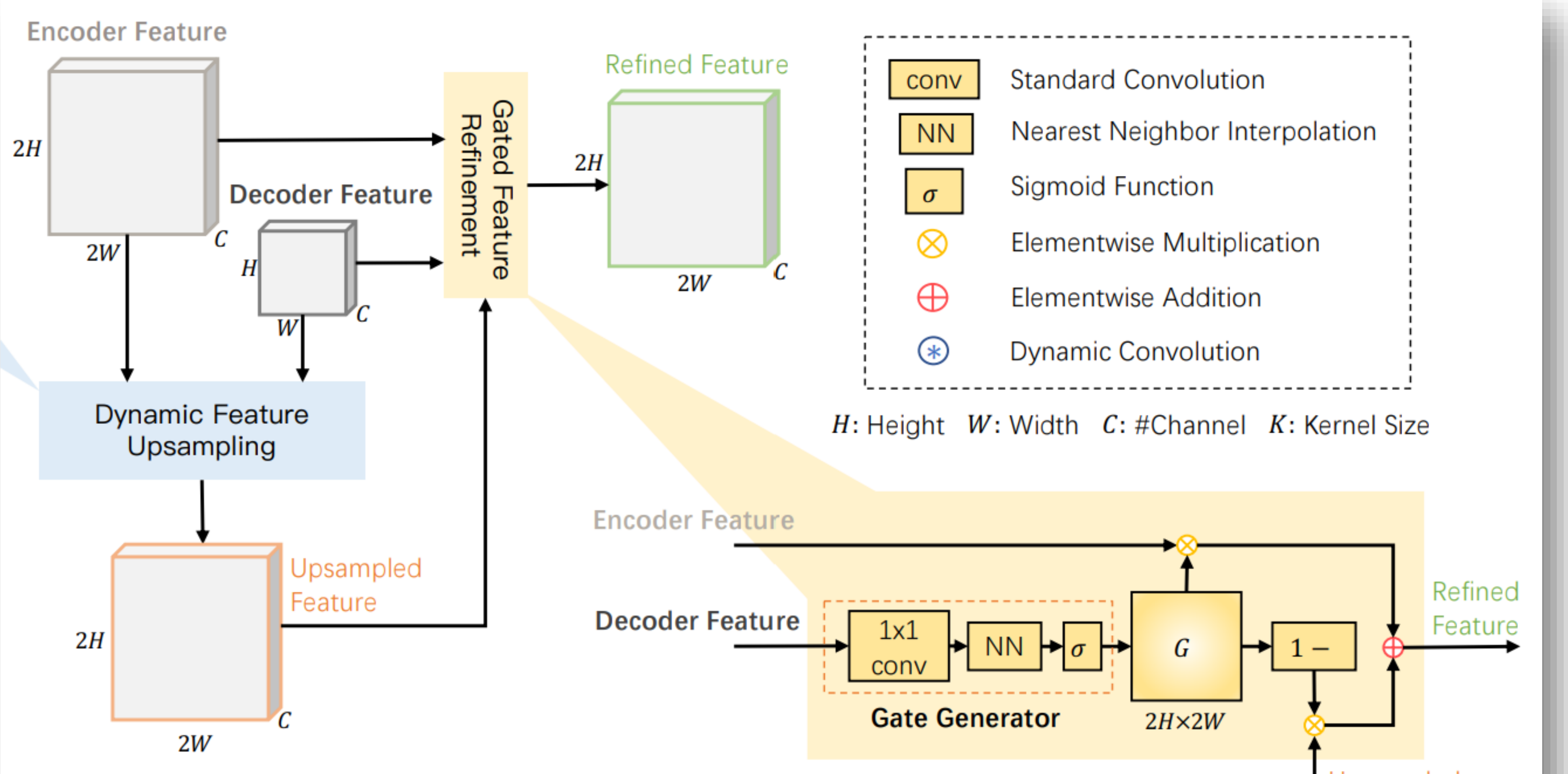


Figure 7 Fast implementations of semi-shift convolution. We present two forms of fast implementations: (left: H2L) high resolution matches low resolution, which is presented in our conference version (Lu et al., 2022b), and (right: L2H) low resolution matches high resolution, which is more memory efficient.

Overview of FADE

Gated Feature Refinement

```python
def forward(self, en, de):
    gate = self.gate_generator(de)
    kernels = F.softmax(self.kernel_generator(en, de), dim=1)
    return gate * en + (1 - gate) * self.carafe(de, kernels, self.up_kernel_size, self.scale)
```



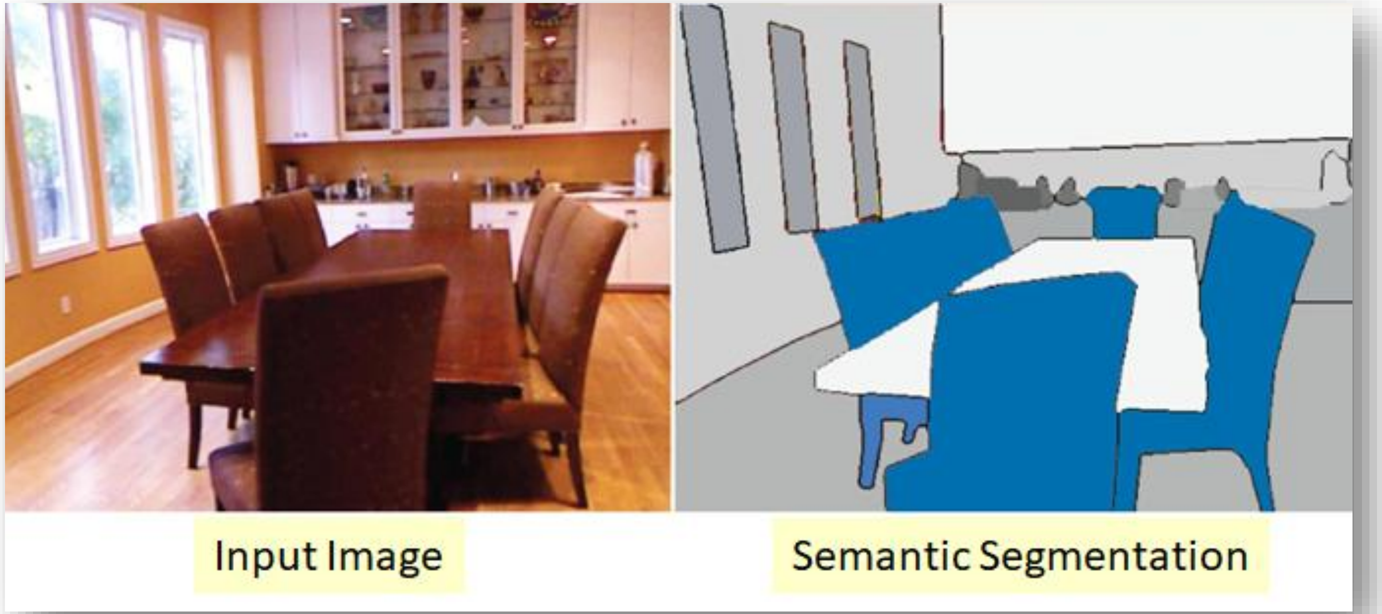
(b) Overview of FADE

(c) Gated Feature Refinement

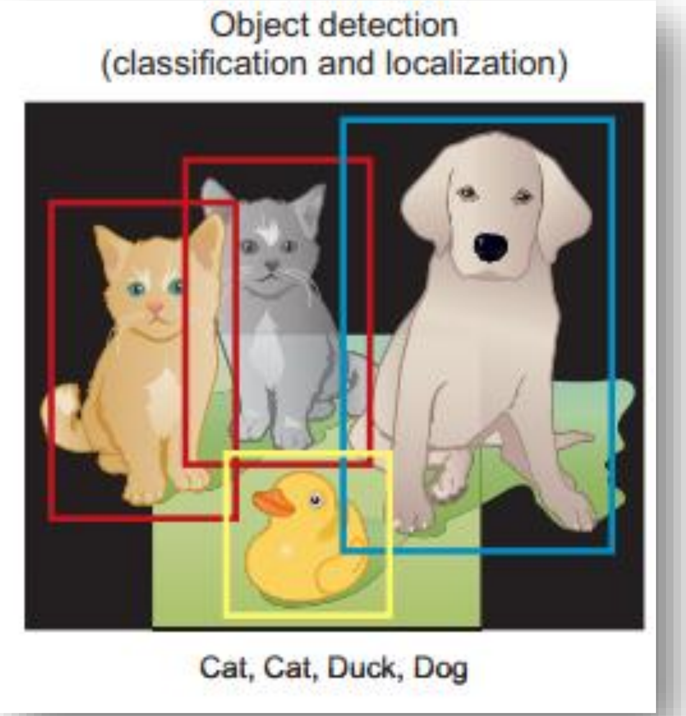
Experimentation
& Results

Disclaimer

- To test task-agnostic property
  - Focus: Semantic Segmentation & Object Detection



Example of a semantic segmentation task

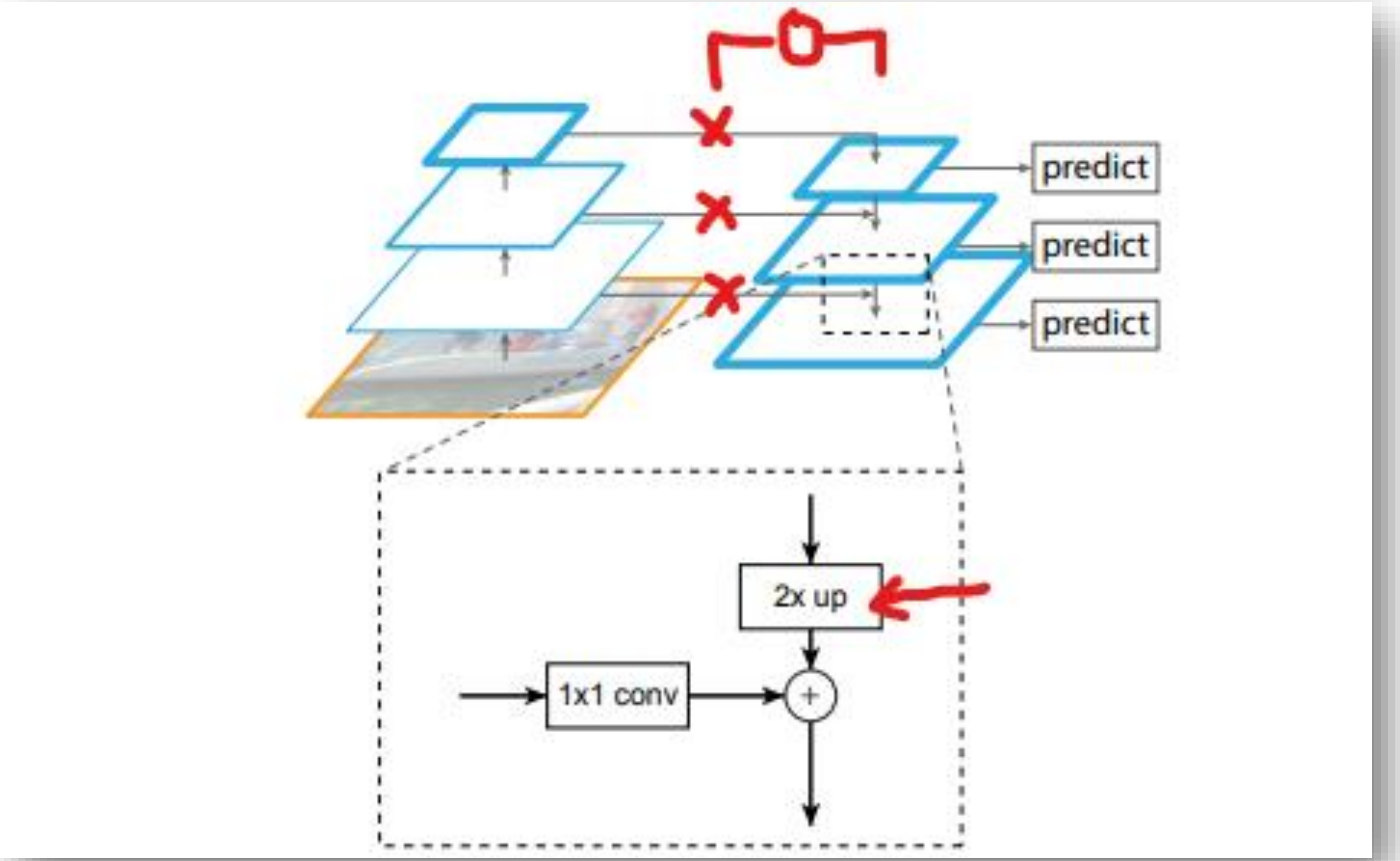


Example of an object detection task

Object Detection

- Model(s): Faster R-CNN (R50, R101)
    - Paper: *mmdetection*
    - My work: **torchvision.models.detection.fasterrcnn_resnet50_fpn**
  - Modify *only* upsampling stages in Feature Pyramid Network and remove skip connections.
- Dataset: MS COCO (Lin et al., 2014)
  - Obtained test/validation set through
    - Original website
      - Missing/corrupted annotation issue when put to Google Drive (%40 does not match)
    - FiftyOne*
      - Data/time efficient, harder to customize.
- Metrics: AP, AP_{50} , AP_{75}, AP_{S}, AP_{M}, AP_{L}

Changes Made to FPN



Baseline Predictions Visualized
torchvision.models.detection.fasterrcnn_resnet50_fpn

Baseline Metrics

```
Accumulating evaluation results...
DONE (t=6.87s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.370
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.585
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.398
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.211
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.403
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.482
```

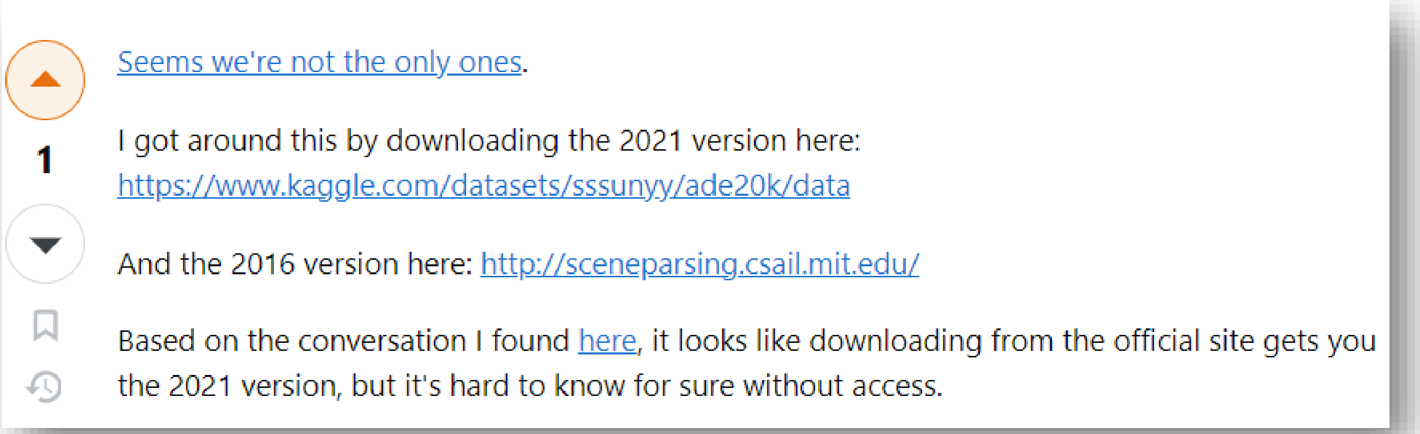
Faster RCNN (Ren et al., 2015)	backbone	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FA2M (Wu et al., 2022)	R50	+23K	37.9	58.8	40.9	22.1	41.7	48.8
FAM (Li et al., 2020b)	R50	+0.8M	37.8	58.6	41.0	21.8	41.2	48.8
GD-FAM (Li et al., 2023)	R50	+0.5M	38.1	59.2	41.3	22.7	41.5	<u>49.6</u>
Nearest	R50	46.8M	37.4	58.1	40.4	21.2	41.0	48.1
CARAFE (Wang et al., 2019)	R50	+0.3M	**38.6**	**59.9**	**42.2**	**23.3**	**42.2**	**49.7**
IndexNet (Lu et al., 2022a)	R50	+8.4M	37.6	58.4	40.9	21.5	41.3	49.2
A2U (Dai et al., 2021)	R50	+0.1M	37.3	58.7	40.0	21.7	41.1	48.5
SAPA (Lu et al., 2022c)	R50	+0.1M	37.8	59.2	40.6	22.4	41.4	49.1
FADE	R50	+0.2M	37.8	58.8	40.8	21.2	41.2	49.4

FADE Metrics

- Hardware: T4 GPU (Google Colab)
- Number of datapoints: Around 70k.
  - Reduced size due to mismatch between annotations and images.
- Loss decreasing throughout iterations steadily.

Semantic Segmentation

- Model(s):
  - SegFormer (Xie et al., 2021) as transformer baseline (B1, B3, B4, B5)
    - Code: available via MMsegmentation
  - UPerNet (Xiao et al., 2018) as convolutional baseline (R50, R101)
    - Code: available via MMsegmentation
- Dataset: ADE20K (Zhou et al., 2017)
- Metrics: mIoU, bIoU



Future Directions

- Use already implemented data subsampler on train data to obtain a more manageable size.
- Idea:
  - Use L2H instead of H2L
- Train both the baseline and modified model on subset.
  - Obtain more trustworthy comparison.
- Wishful thinking:
  - Plug FADE into SegFormer and compare results with baseline.
    - Train both models.
      - Possible issues: Time and computation

Appendix

- Feature Pyramid Networks for Object Detection
  - [1612.03144] Feature Pyramid Networks for Object Detection
- CARAFE: Content-Aware ReAssembly of Features
  - [1905.02188] CARAFE: Content-Aware ReAssembly of FEatures
- “FiftyOne”: open-source tool facilitating visualization and access to COCO data resources.
  - FiftyOne — FiftyOne 1.0.2 documentation

End of Progress Presentation
Start of Final Presentation

Disclaimer: For the rest of this presentation, Faster R-CNN model with a ResNet-50-FPN backbone* will be referred to as the baseline model while the same model with only its upsampling operator changed from FPN to FADE will be referred to as the custom model.

*fasterrcnn_resnet50_fpn — Torchvision main documentation

Summary of Progress Checkpoint

Already Implemented Components

- Model training and validation flow
    - Extraction and transformation of MS COCO dataset
    - Training implementation
    - Validation implementation

Missing Parts

- Trustworthy comparison between custom and baseline model
    - Baseline model was fully pretrained, imported from PyTorch, which is more than likely to outperform the custom model due to optimized training procedure and amount of data used for training.
- Exception handling during training loop
    - Training halted when an out-of-bounds bounding box prediction was made. (This exception occurred *only* with initial set of hyperparameters)

How is the “Custom Model” Created?

- “Modify only the upsampling stages in FPN of the custom model.”
- “Original skip connection in FPN is removed due to inclusion of gating mechanism.”
- “All other settings remain unchanged.”

Improving the performance of a deep learning model is a highly iterative process. To fully capture the thought process behind the experimentation, results are also presented iteratively alongside their corresponding adjustments.

Disclaimer:

<u>Unless stated otherwise</u>

- Training data is *randomly sampled* from train2017 split of MSCOCO* (total:118k, available: ~71k)

- No layers were frozen

- Evaluations are carried out with entire val2017 split of MSCOCO (total: 5k)

[*COCO - Common Objects in Context]

1. Train Custom Model for More Than 1 Epoch
(train size: 5k)

Hyperparameters

```
"lr_scheduler": null,
"lr": 0.01,
"momentum": 0.9,
"weight_decay": 0.0005,
"num_epochs": 3,
```

Standard Average Precision (AP) metrics for Custom Model

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.003
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.009
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.001
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.002
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.004
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.004
```

📌 To validate the training procedure, the baseline model was trained using the exact setup after obtaining unsatisfactory results with the custom model.

2. Train Baseline Model with Same Setup to Validate Training Procedure (train size: 5k)

AP metrics in order (Baseline, Custom)

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.022
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.047
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.018
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.018
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.027
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.030
```

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.003
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.009
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.001
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.002
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.004
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.004
```

📌 Baseline model doesn't perform as poorly with the same setup:

- FADE's learnable gating mechanism starts from scratch, leading to higher initial loss.

Idea: Extending training to give the custom model enough time to learn the additional parameters introduced by FADE.

Hyperparameters

```
"lr_scheduler": null,
"lr": 0.01,
"momentum": 0.9,
"weight_decay": 0.0005,
"num_epochs": 3,
```

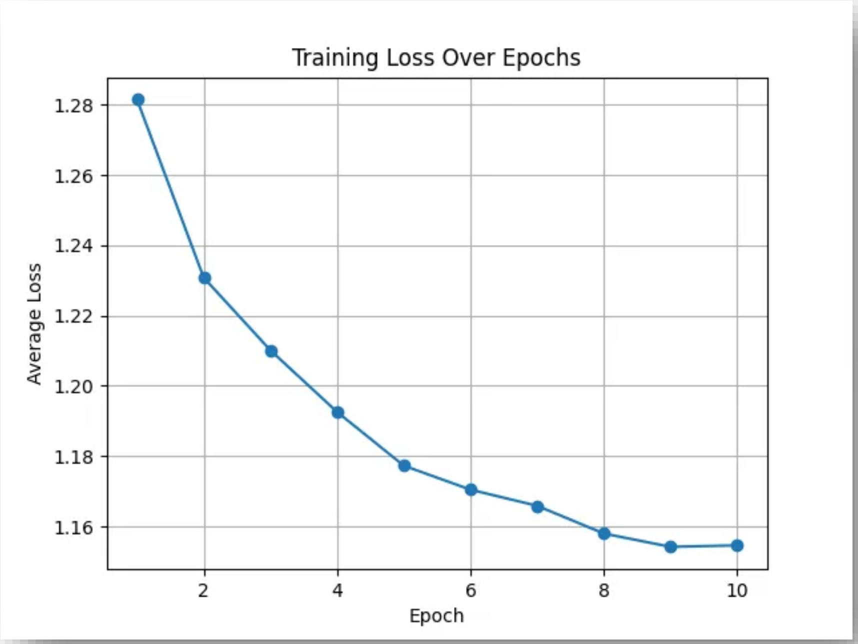
3. Increase Epochs and Training Set Size (train size: 10k)

AP metrics in order [Best Custom model from prior experiment vs current experiment (Prior Custom), and (Current Custom)]

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.003
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.009
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.001
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.002
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.004
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.004
```

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.010
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.024
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.006
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.006
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.011
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.014
```

Average Loss over Epochs for Current Custom



Naturally,results improved due to increased training time and dataset size. However, the loss began to stagnate between epochs 4 and 6.

Training logs showed oscillating loss around epoch 4, suggesting that the model's updates might be unstable during this phase. Introducing a learning rate decay at this point could help stabilize the training process.

Hyperparameters

```
"lr": 0.01,
"momentum": 0.9,
"weight_decay": 0.0005,
"num_epochs": 10,
```

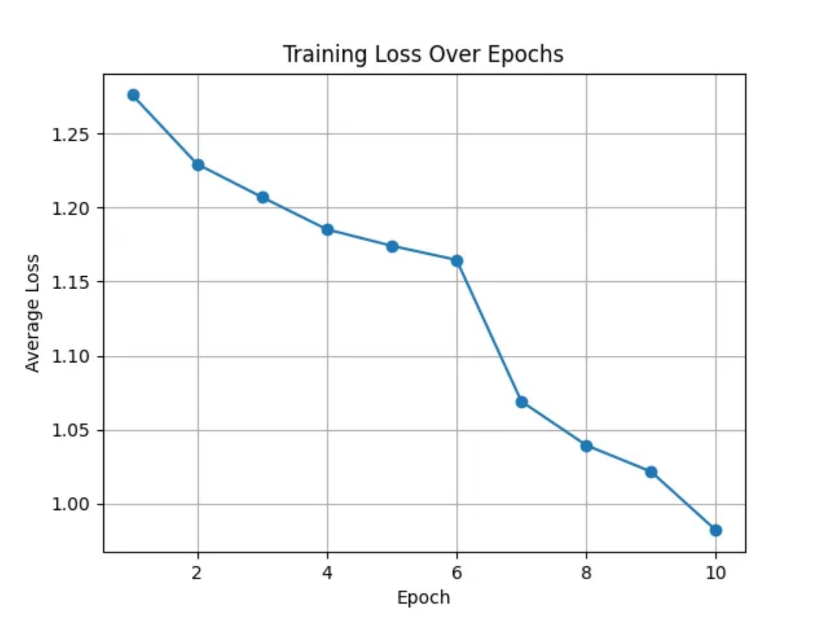
4. Introduce Learning Rate Decay & Warmup
(train size: 10k)

AP metrics in order Prior Custom and Current Custom

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.010
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.024
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.006
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.006
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.011
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.014
```

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.019
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.041
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.017
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.011
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.021
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.027
```

Average Loss over Epochs for Current Custom



⚠️ **StepLR** (step_size=3, gamma=0.1) activated at ep6 ,

Warmup

- Epoch 1: LR = 0.003333
  Epoch 2: LR = 0.006667

StepLR's effect is clearly visible as there is a significant drop in loss between epoch 6~8

However, initial learning phase is less steep, might need to reconsider warmup.

Hyperparameters

```
"lr": 0.01,
"momentum": 0.9,
"weight_decay": 0.0005,
"num_epochs": 10,
```

5. Introduce Batch Normalization for Channel Compression and Feature Map Refinement

(train size: 10k)

AP metrics in order Prior Custom and Current Custom

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.019
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.041
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.017
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.011
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.021
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.027
```

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.024
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.048
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.021
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.018
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.026
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.031
```

Average Loss over Epochs for Current Custom

⚠️ Disclaimer: Batch normalization (BN) is already applied in original FPN implementation. However, during the implementation of FADE, BN was omitted due to a misunderstanding on my part regarding FADE's normalization process.

📌 All parts except for batch normalization stayed the same with prior implementation.

Noticeable improvements are observed in the AP metrics

Hyperparameters

```
"lr": 0.01,
"momentum": 0.9,
"weight_decay": 0.0005,
"num_epochs": 10,
```

6. Modify Learning Rate Decay Structure (Scheduled LR + StepLR)

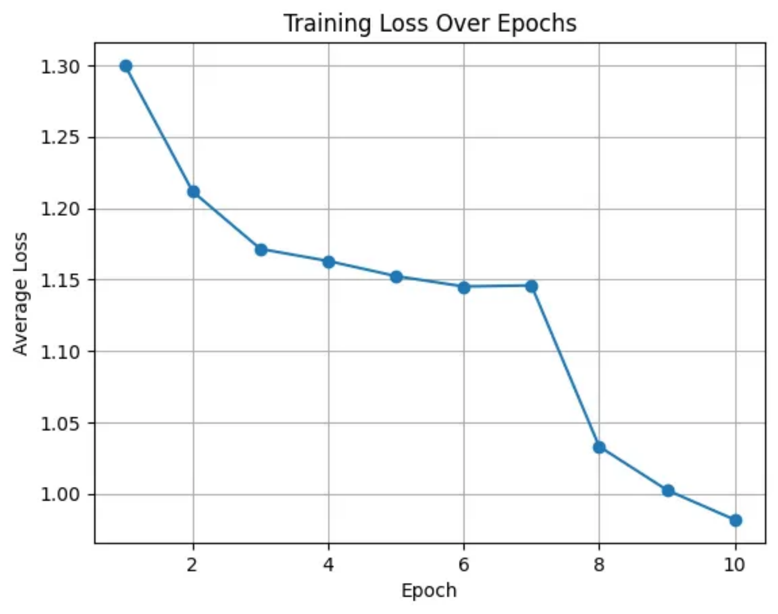
(resulting learning rates shared below) (train size: 10k)

AP metrics in order Prior Custom and Current Custom

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.024
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.048
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.021
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.018
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.026
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.031
```

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.023
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.047
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.020
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.015
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.025
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.030
```

Average Loss over Epochs for Current Custom



📌 Resulting learning rates over epochs with the modified approach

[0.005, 0.008, 0.009, 0.01, 0.01, 0.01, 0.001, 0.001, 0.001, 0.0001]

A slight drop in performance was observed.

Epoch 1-3: Higher learning rate for epoch 1 could be a better starting point considering initial lower loss levels without warmup. Also, a smoother rate change between epoch 1-2 could improve learning.

Epoch 4-6: Model could benefit from a lower learning rate considering the stagnating average loss.

Hyperparameters

```
"lr": 0.01,
"momentum": 0.9,
"weight_decay": 0.0005,
"num_epochs": 10,
```

It is worth noting that additional experiments were carried out to inform parameter tuning.

The intermediate steps are not detailed in the main experimentation section in order to maintain a clear narrative flow.

More information can be found in the appendix section.

*Best Iteration

7. Modify Learning Rate Decay Structure* (Scheduled LR only)

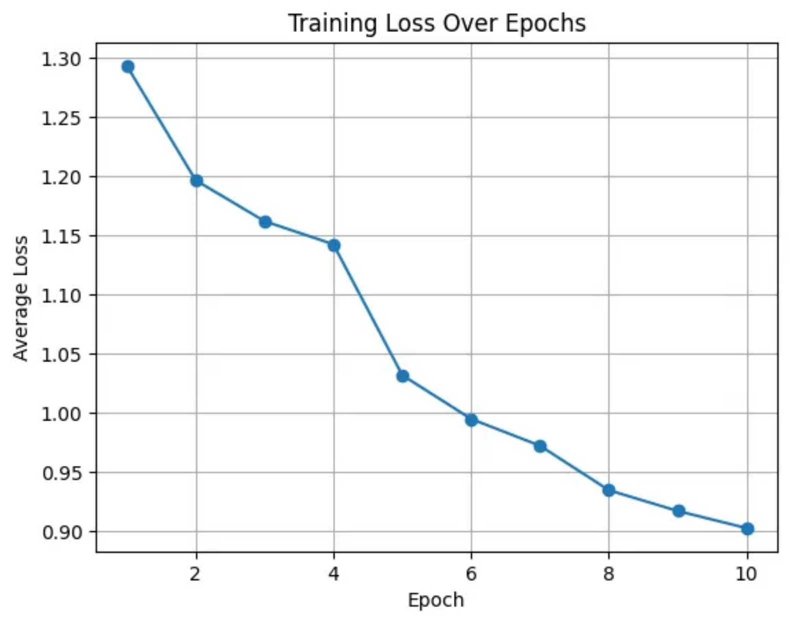
(resulting learning rates shared below) (train size: 10k)

AP metrics in order Prior Custom and Current Custom

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.024
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.048
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.021
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.018
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.026
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.031
```

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.028
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.057
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.025
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.018
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.027
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.038
```

Average Loss over Epochs for Current Custom

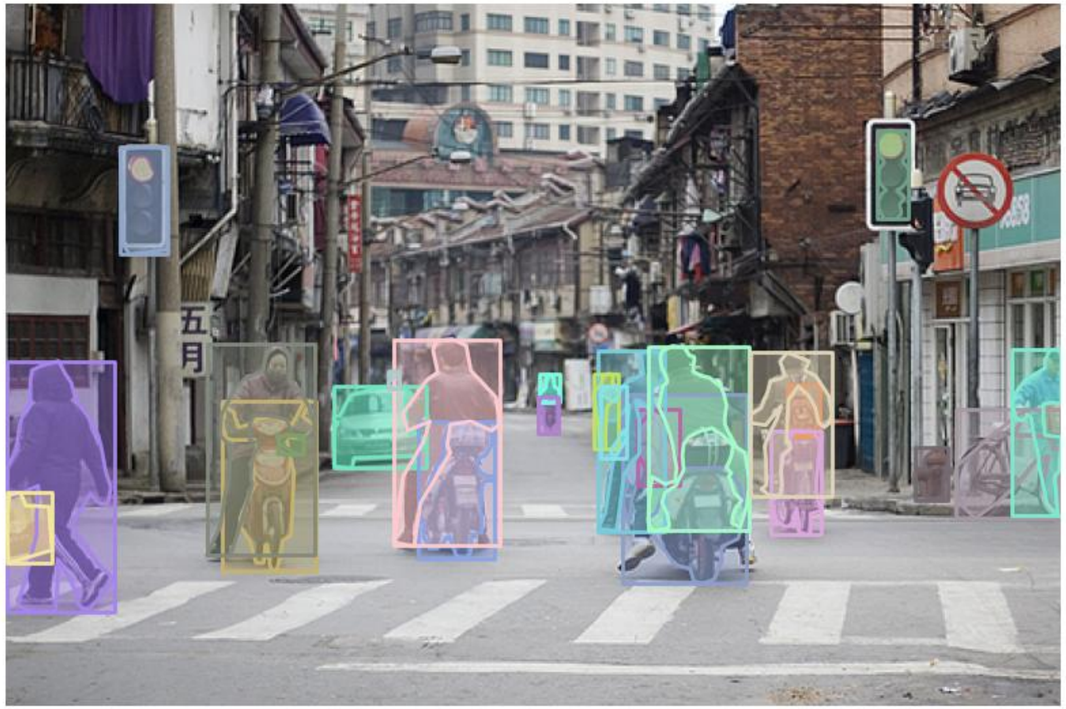


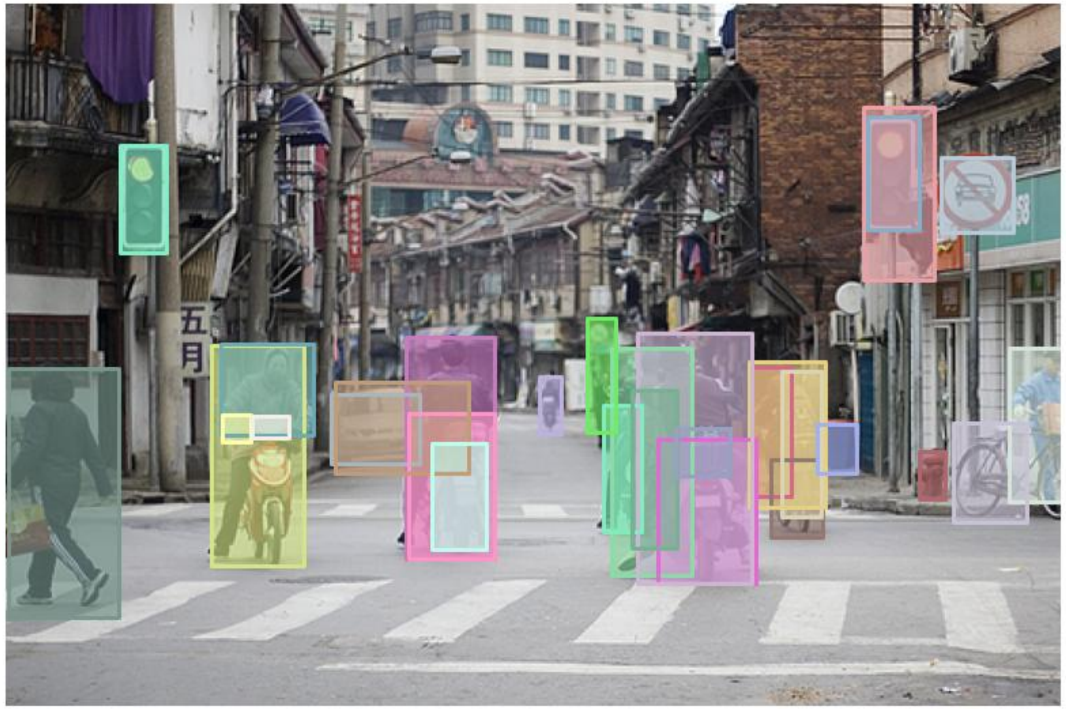

```python
def lr_schedule(epoch):
    '''
    Actual scheduler function is scaled by 0.01 (initial lr)
    This function showcases the resulting learning rates for simplicity.
    '''
    epoch += 1
    lr = 1.0

    if epoch in (1, 2):
      lr = 1e-2
    elif epoch in (3, 4, 5, 6):
      lr = 4e-3
    elif epoch in (7, 8, 9):
      lr = 1e-3
    else:
      lr = 5e-4

    return lr
```

Hyperparameters

```
"lr": 0.01,
"momentum": 0.9,
"weight_decay": 0.0005,
"num_epochs": 10,
```

Ground Truth vs Example Prediction



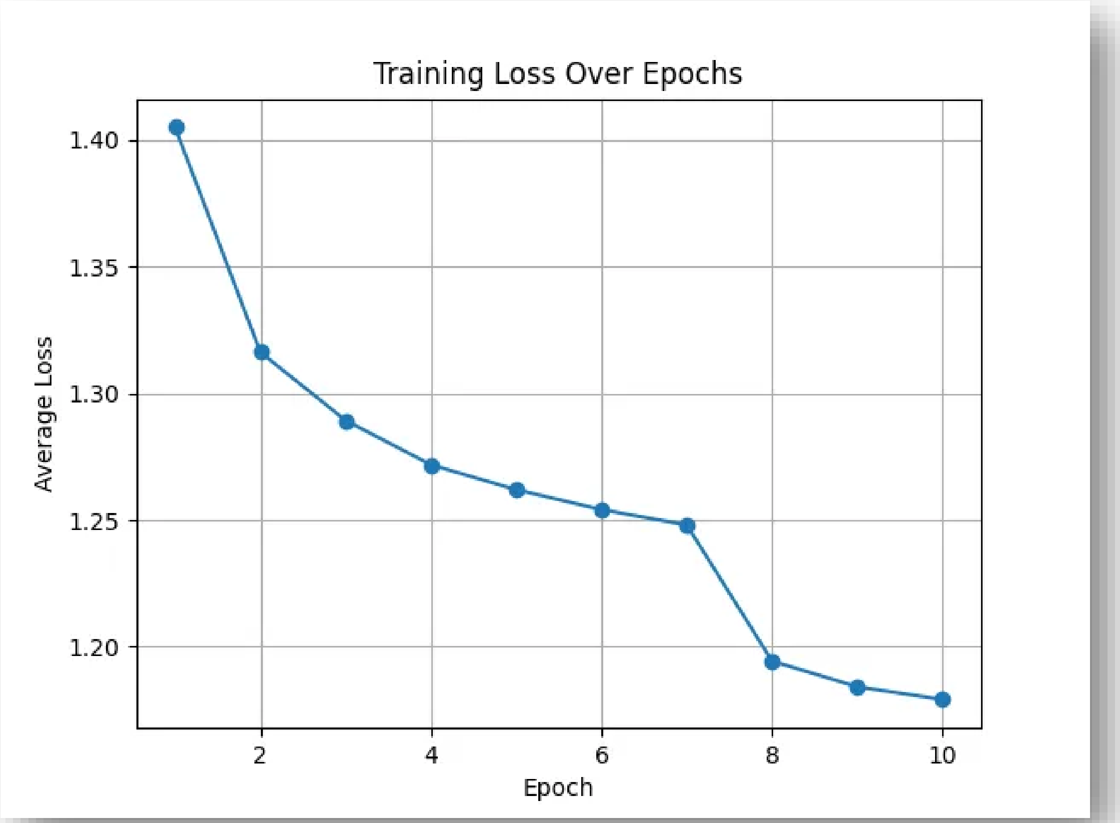


This is an example of a confident prediction.
Low confidence predictions (<0.5) are suppressed.
Custom model is not successful at distinguishing small object crowded by a bigger object.

Appendix

Other Experiments I

- Freeze backbone completely throughout entire training and remove learning rate scheduling completely. Use 0.01 constant learning rate.
  - The aim was to prevent diverging from learned weights of the pretrained model and only learn for FADE layers



Other Experiments II

- Use lr_schedule*
- No metrics or plot available for this setup since important information was obtained from change in average loss between epochs. Consequently, training was halted to begin a new experiment immediately, resulting in *Experiment 7*.

*

```python
def lr_schedule(epoch):
    '''
    Actual scheduler function is scaled by 0.01 (initial lr)
    This function showcases the resulting learning rates for simplicity.
    '''
    epoch += 1
    lr = 1.0

    if epoch == 1:
      lr = 6e-3
    elif epoch == 2:
      lr = 8e-3
    elif epoch in (3, 4):
      lr = 7e-3
    elif epoch in (5, 6, 7):
      lr = 1e-4
    else:
      lr = 5e-4

    return lr
```

Other Experiments III

- Increasing dataset size to 20k with lr_schedule*
  - Average loss @ epoch 10: 0.9229

```
Epoch [1/20] completed in 1:07:46
Learning Rate: Average Loss: 1.2521
Epoch [2/20] completed in 0:22:27
Learning Rate: Average Loss: 1.1223
Epoch [3/20] completed in 0:22:28
Learning Rate: Average Loss: 1.0924
Epoch [4/20] completed in 0:22:30
Learning Rate: Average Loss: 1.0800
Epoch [5/20] completed in 0:22:34
```

```
Epoch [6/20] completed in 0:22:34
Learning Rate: Average Loss: 0.9732
Epoch [7/20] completed in 0:22:35
Learning Rate: Average Loss: 0.9494
Epoch [8/20] completed in 0:22:34
Learning Rate: Average Loss: 0.9364
Epoch [9/20] completed in 0:22:33
Learning Rate: Average Loss: 0.9299
Epoch [10/20] completed in 0:22:30
Learning Rate: Average Loss: 0.9229
```

*

```python
def lr_schedule(epoch):
    epoch += 1
    lr = 1.0

    if epoch in [1]:
      lr = 1.0
    elif epoch in [2, 3, 4, 5]:
      lr = 0.4
    elif epoch in [6, 7, 8, 9, 10]:
      lr = 0.1
    else:
      lr = 0.05

    return lr
```

Other Experiments IV

- Modify learning rate of the highest scoring experiment's (7) scheduler to lr_schedule*

```
def lr_schedule(epoch):
    epoch += 1
    lr = 1.0

    if epoch in [1]:
      lr = 1.0
    elif epoch in [2, 3]:
      lr = 0.4
    elif epoch in [4, 5]:
      lr = 0.1
    elif epoch in [6, 7]:
      lr = 0.03
    elif epoch in [8, 9]:
      lr = 0.01
    else:
      lr = 0.001

    return lr
```

```
Epoch [1/20] completed in 0:26:44
Learning Rate: Average Loss: 1.3052
Epoch [2/20] completed in 0:11:14
Learning Rate: Average Loss: 1.2081
Epoch [3/20] completed in 0:11:15
Learning Rate: Average Loss: 1.1645
Epoch [4/20] completed in 0:11:15
Learning Rate: Average Loss: 1.0812
Epoch [5/20] completed in 0:11:14
Learning Rate: Average Loss: 1.0530
Epoch [6/20] completed in 0:11:17
Learning Rate: Average Loss: 1.0038
Epoch [7/20] completed in 0:11:17
Learning Rate: Average Loss: 0.9893
Epoch [8/20] completed in 0:11:17
Learning Rate: Average Loss: 0.9653
Epoch [9/20] completed in 0:11:19
Learning Rate: Average Loss: 0.9586
Epoch [10/20] completed in 0:11:18
Learning Rate: Average Loss: 0.9496
```