



AMLD 2021 - Food Recognition Challenge

Rohit Midha Shraddhaa Mohan



Overview

1. Problem Statement
2. About the Dataset
3. Data Preprocessing
4. Pre - trained Weights
5. Data Augmentations
6. Training schedule
7. TideCV analysis
8. Label Smooth CrossEntropy Loss
9. Test Time Augmentations
10. Model Ensembling
11. Final Results

Problem Statement

The goal of this challenge is to train models which can look at images of food items and detect the individual food items present in them.

At its core, it is an instance segmentation challenge which aims to train models that can work in the real world since it is trained on real data.





About the Dataset



MY FOOD REPO

It is a novel dataset of food images collected through the MyFoodRepo app where numerous volunteer Swiss users provide images of their daily food intake, and annotators either manually annotate or verify automatic annotations.

It is an evolving dataset:

- Round 1: 40 classes
- Round 2: 61 classes
- Round 3: 273 classes

Data Preprocessing

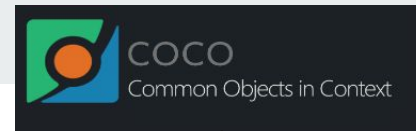
The most important data preprocessing step we used, was to correct the bounding boxes in the dataset to fit the masks correctly.



Erroneous BBox
[122.5, 290.5, 374.5, 351.0]



BBox Generated from Mask
[291, 114, 351, 209]

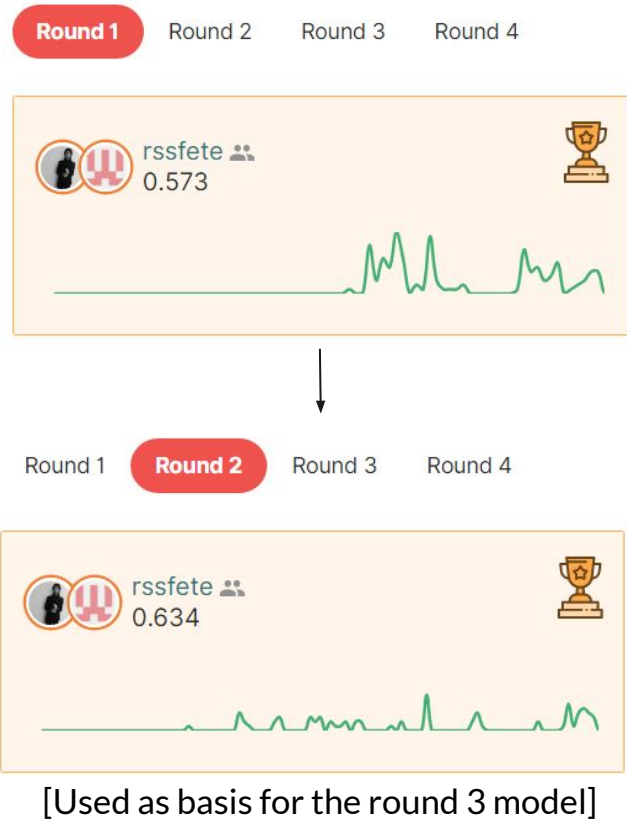


Pre-trained Weights

Our solutions across all rounds used the Hybrid Task Cascade Model with the ResNeXt101 backbone with DCN(64x4d).

In order to help the model converge faster and transfer some of the learnings from previous rounds we always used the previous rounds weights as the base for the next round.

For round 1, we used COCO weights provided by MMDetection.





Data Augmentations



Type	OneOf	params	probability
ShiftScaleRotate		shift_limit=0.0625,scale_limit=0.0,rotate_limit=30,interpolation=2	0.3
RandomRotate90			0.5
RandomBrightnessContrast		brightness_limit=[0.1, 0.3],contrast_limit=[0.1, 0.3]	0.3
Blur	1		0.4
MotionBlur	1		0.4
GaussNoise	1		0.4
ImageCompression	1	quality_lower=75	0.4
CoarseDropout		max_holes=30,max_height=30,max_width=30,min_holes=5,min_height=10,min_width=10,fill_value=[103.53, 116.28, 123.675]	0.4



Training Schedule

```
optimizer = dict(type='SGD', lr=0.0025, momentum=0.9,  
weight_decay=0.0001)  
lr_config = dict(  
    policy='step',  
    warmup='linear',  
    warmup_iters=500,  
    warmup_ratio=0.001,  
    step=[18, 22])  
total_epochs = 24
```

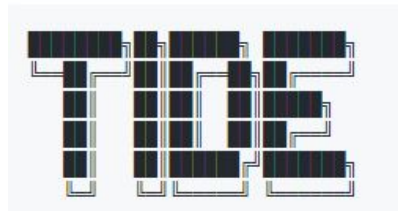
With multiscale training to accommodate varying image sizes with limits of 1600x400 and 1600x1400.
All experiments were conducted on a single V100 GPU

TideCV Analysis

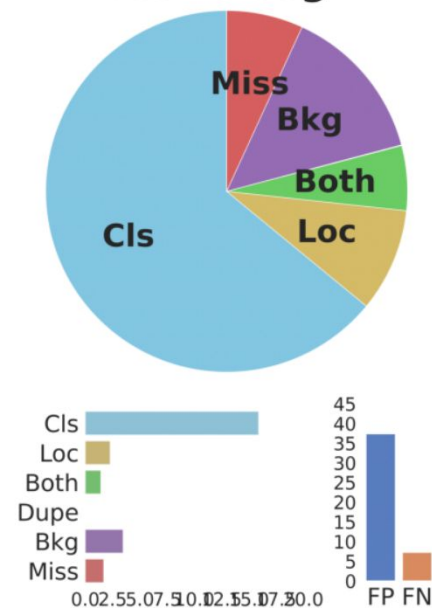
The **tidecv** library breaks down object detection loss into its components (Cls, Loc, Cls+Loc, Duplicate, Bkgd, Missed) and it confirmed that the classification loss was by far the largest source of error.

Why?

Upon closer inspection of the dataset, we noticed that some of the labels are really closely related, for example: [bread-white, bread-whole-wheat, bread-5-grain, ...(20 different types)], [water, water-mineral, water-with-lemon-juice]. As a result it is likely that some of these have been mislabelled.



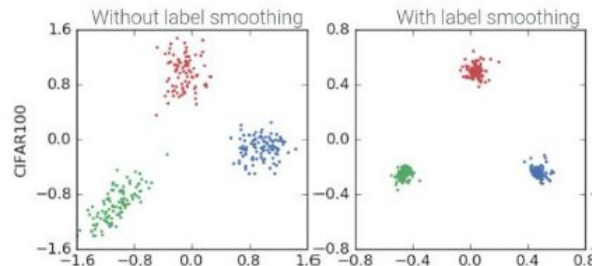
results.segm



Label Smooth Cross-Entropy Loss

Label Smoothing is a regularization technique that introduces noise for the labels. This accounts for the fact that datasets may have mistakes in them.

We re-trained our model using `LabelSmoothCrossEntropyLoss` on the entire set(train + validation) for the loss cls and it lead to an improvement of our score from **0.546 precision** to **0.551 precision**.





Test Time Augmentations

MMdetection's MultiScaleFlipAug was used with flip=True, and an image scale of (1080,720). It was found that this did not cause improvement on the precision score (both gave **0.551 precision**).

We did try multiple image scales, including 2 image scales but these did not evaluate due to the time constraint of 2000 seconds per 1000 images.

```
test_cfg = dict(
    rpn=dict(
        nms_across_levels=False,
        nms_pre=2500,
        nms_post=2000,
        max_num=1500,
        nms_thr=0.8,
        min_bbox_size=0),
    rcnn=dict(
        score_thr=0.0015,
        nms=dict(type='nms', iou_thr=0.35),
        max_per_img=100,
        mask_thr_binary=0.4)
)
```



Model Ensembling

Another thing we ended up trying was ensembling models from different epochs. In this type of ensembling we choose one particular epoch to predict on certain classes with the other predicting on the rest.

Since this approach ran 2 models, due to the time constraint TTA was not used. But it was found that it gave comparable results to the single model approach which had TTA.




Model Ensembling of e11 and e15 with flip=False

- Average precision is **0.546** and average recall is **0.868**

Epoch 15 with flip=True

- Average precision is **0.545** and average recall is **0.869**

Final Results

Δ	#	Participants	Average Precision	Average Recall
●	01	 rssfete  	0.551	0.885



Thank You

Questions? Contact us at:



Rohit Midha : rohit.midha23@gmail.com



Shraddhaa Mohan : shraddhaa.mohan@gmail.com