SIMPLE PRESENTATION FOR COMPETITION HUBMAP + HPA

UW-MADISON GI TRACT IMAGE SEGMENTATION

MINGZIRUI WU BOTAO JIANG

BASELINE COMPARISON

- UW-MADISON Similar Baseline
  - PyTorch + Segmentation models
- FastAI Baseline
  - PyTorch + FastAI
- MONAI Baseline
  - PyTorch + MONAI

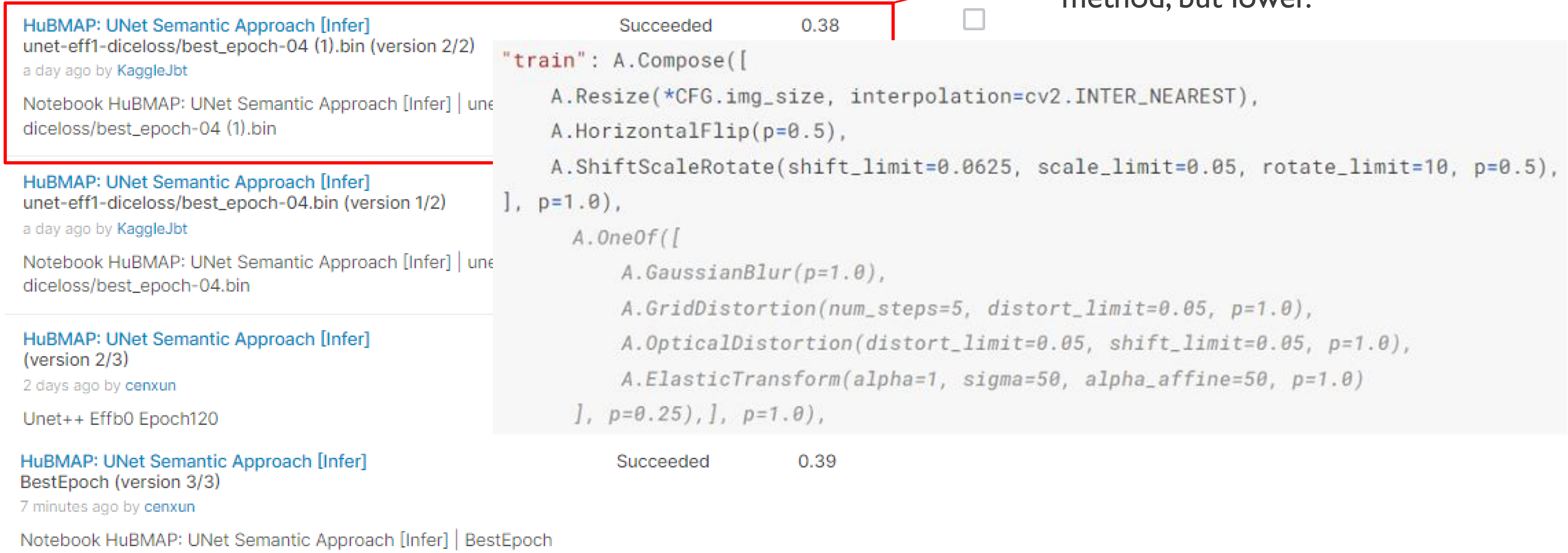
UW-MADISON SIMILAR BASELINE

- Loss Function
  - Dice
  - Training Result: LovaszLoss May Better
- Network
  - UnetPlusPlus + EfficientNet-b0
  - UnetPlusPlus + EfficientNet-b1

UW-MADISON SIMILAR BASELINE

- Result

- Try to add more augmentation method, but lower.



HuBMAP: UNet Semantic Approach [Infer] Succeeded 0.38
unet-eff1-diceloss/best_epoch-04 (1).bin (version 2/2)
a day ago by KaggleJbt

Notebook HuBMAP: UNet Semantic Approach [Infer] | une
diceloss/best_epoch-04 (1).bin

HuBMAP: UNet Semantic Approach [Infer]
unet-eff1-diceloss/best_epoch-04.bin (version 1/2)
a day ago by KaggleJbt

Notebook HuBMAP: UNet Semantic Approach [Infer] | une
diceloss/best_epoch-04.bin

HuBMAP: UNet Semantic Approach [Infer]
(version 2/3)
2 days ago by cenxun

Unet++ Effb0 Epoch120

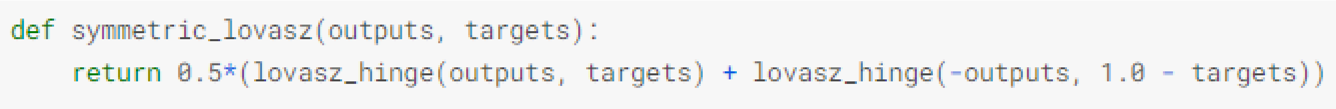
HuBMAP: UNet Semantic Approach [Infer] Succeeded 0.39
BestEpoch (version 3/3)
7 minutes ago by cenxun

Notebook HuBMAP: UNet Semantic Approach [Infer] | BestEpoch

```
"train": A.Compose([
    A.Resize(*CFG.img_size, interpolation=cv2.INTER_NEAREST),
    A.HorizontalFlip(p=0.5),
    A.ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.05, rotate_limit=10, p=0.5),
], p=1.0),
      A.OneOf([
          A.GaussianBlur(p=1.0),
          A.GridDistortion(num_steps=5, distort_limit=0.05, p=1.0),
          A.OpticalDistortion(distort_limit=0.05, shift_limit=0.05, p=1.0),
          A.ElasticTransform(alpha=1, sigma=50, alpha_affine=50, p=1.0)
      ], p=0.25),], p=1.0),
```

FASTAI BASELINE

- Loss Function



```
def symmetric_lovasz(outputs, targets):
    return 0.5*(lovasz_hinge(outputs, targets) + lovasz_hinge(-outputs, 1.0 - targets))
```

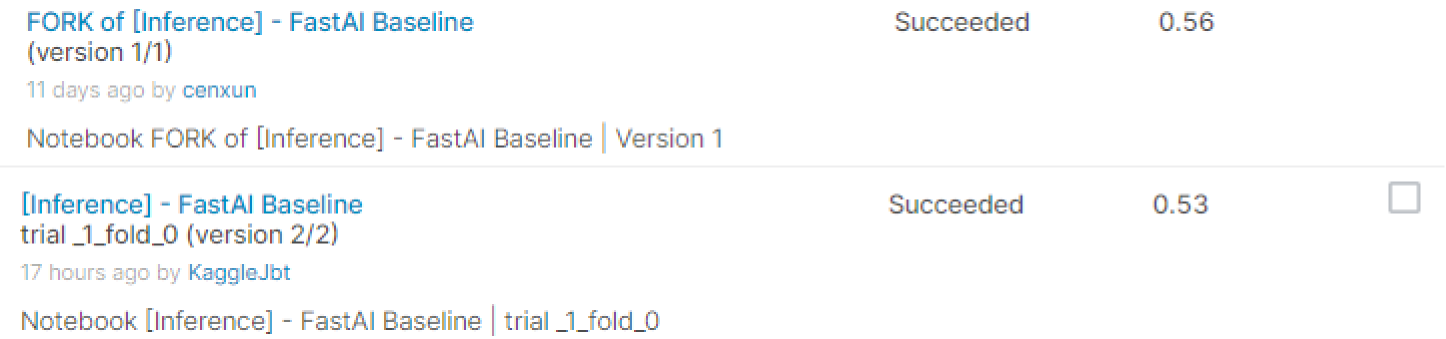
  - Lovász loss
    - Symmetric Lovász loss: consider not only a predicted segmentation and a provided mask but also the inverse prediction and the inverse mask (predict mask for negative case).
- Network
  - UneXt50
    - Unet + ResNeXt50 (ResNet50 + Inception)

UNEXT50

- FPN (Feature Pyramid Network):
  - These skip-connections provide a shortcut for gradient flow improving model performance and convergence speed
- ASPP (Atrous Spatial Pyramid Pooling)
  - The flaw of the traditional U-shape networks is resulted by a small receptive field.
  - Image Size 3000*3000, Confuse for model to see a little part of the image.
  - A way to increase the receptive field and enable interactions between different parts of the image is use of a block combining convolutions with different dilatations (Atrous convolutions with various rates in ASPP block).

FASTAI BASELINE

- Result



FORK of [Inference] - FastAI Baseline (version 1/1) 11 days ago by cenxun Notebook FORK of [Inference] - FastAI Baseline | Version 1	Succeeded	0.56	
[Inference] - FastAI Baseline trial _1_fold_0 (version 2/2) 17 hours ago by KaggleJbt Notebook [Inference] - FastAI Baseline | trial _1_fold_0	Succeeded	0.53	□

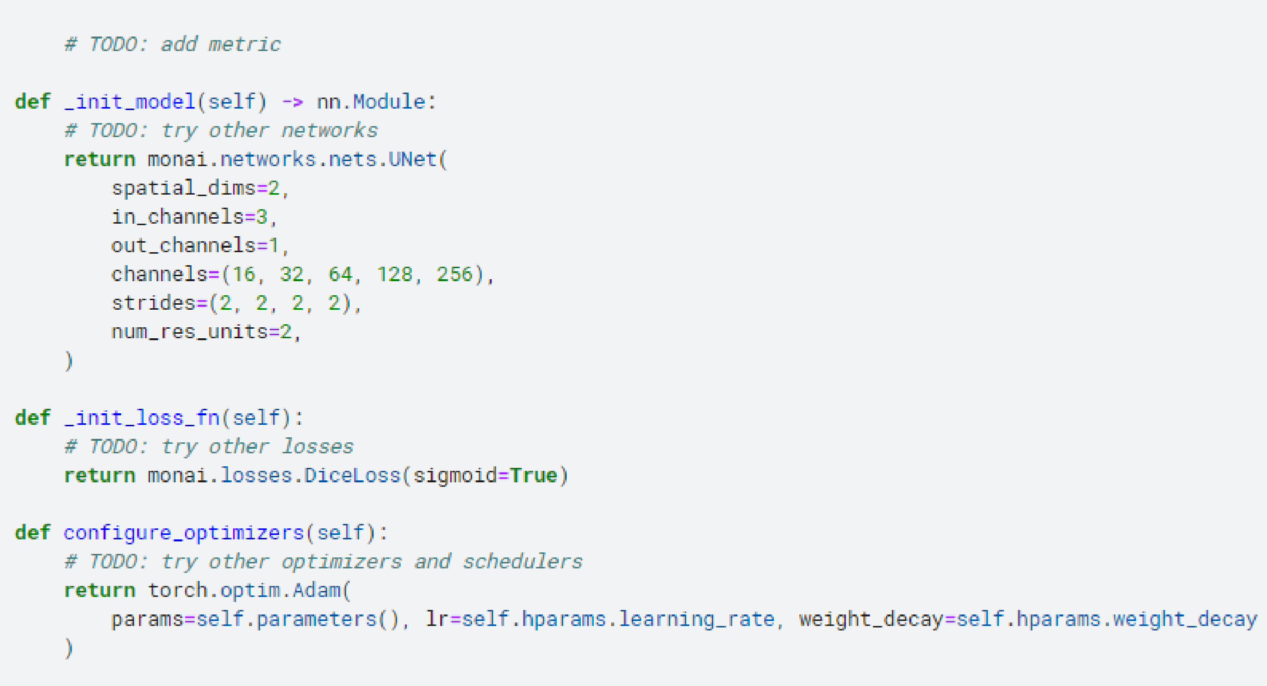
- Complex Network Architecture
- Better performance But harder to understand…

MONAI BASELINE

- Loss Function
  - Dice
  - Training Result: LovaszLoss May Better
- Network
  - Unet

MONAI BASELINE

- Public Score: 0.48
- Advantage:
  - MONAI has more models than Segmentation models
  - Baseline Easy to undertand
  - More intuitive where optimization is possible



```
    # TODO: add metric

def _init_model(self) -> nn.Module:
    # TODO: try other networks
    return monai.networks.nets.UNet(
        spatial_dims=2,
        in_channels=3,
        out_channels=1,
        channels=(16, 32, 64, 128, 256),
        strides=(2, 2, 2, 2),
        num_res_units=2,
    )

def _init_loss_fn(self):
    # TODO: try other losses
    return monai.losses.DiceLoss(sigmoid=True)

def configure_optimizers(self):
    # TODO: try other optimizers and schedulers
    return torch.optim.Adam(
        params=self.parameters(), lr=self.hparams.learning_rate, weight_decay=self.hparams.weight_decay
    )
```

Thank you!