HuBMAP + HPA – Hacking the Human Body

Progress Meeting 3 Group A

https://www.kaggle.com/competitions/hubmap-organ-segmentation

CONTENTS

1. Data Augmentation

- **The aim of data augmentation:**

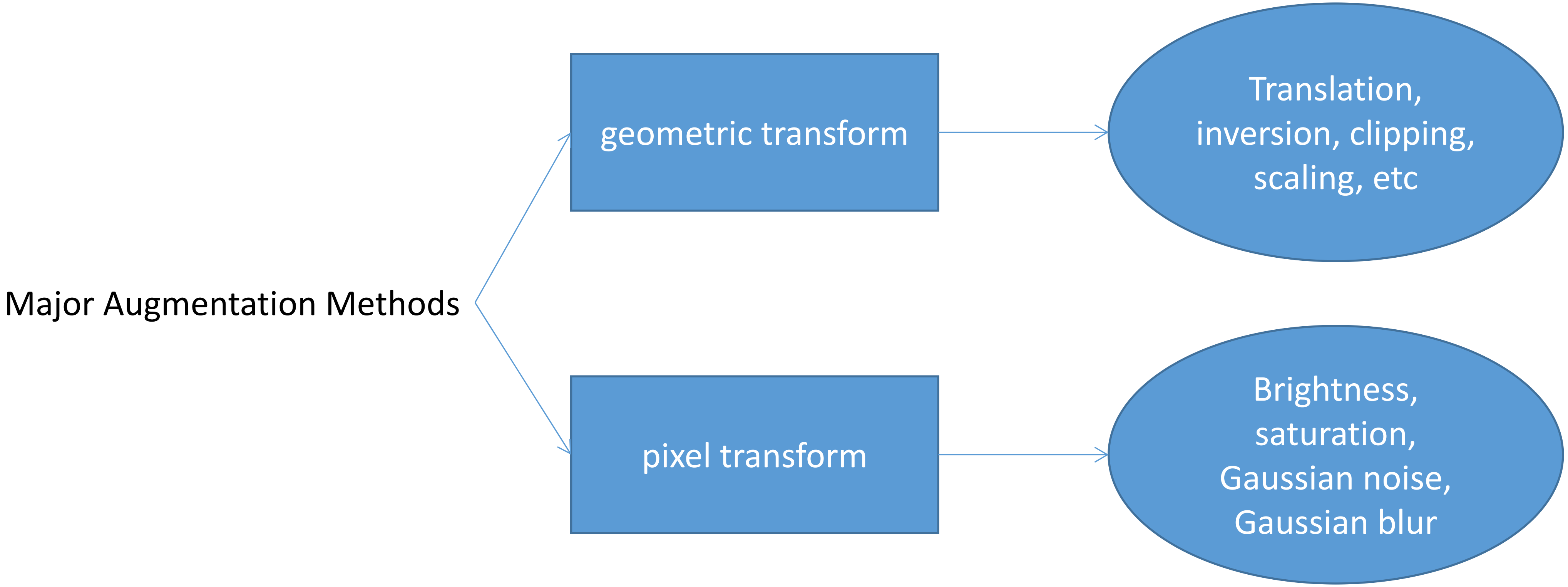
  1. Expand the number of training samples to get more accurate results

  2. Improve the generalization ability of the model

  3. Add noise data to improve the robustness of the model

  4. Aovid sample imbalance(More on the classification problem)

1. Data Augmentation



- **Choice of baseline**

[Training] - FastAI Baseline

FORK of [Inference] - FastAI Baseline

rank high in the leaderboard

perform well in another similar competition, rank 3

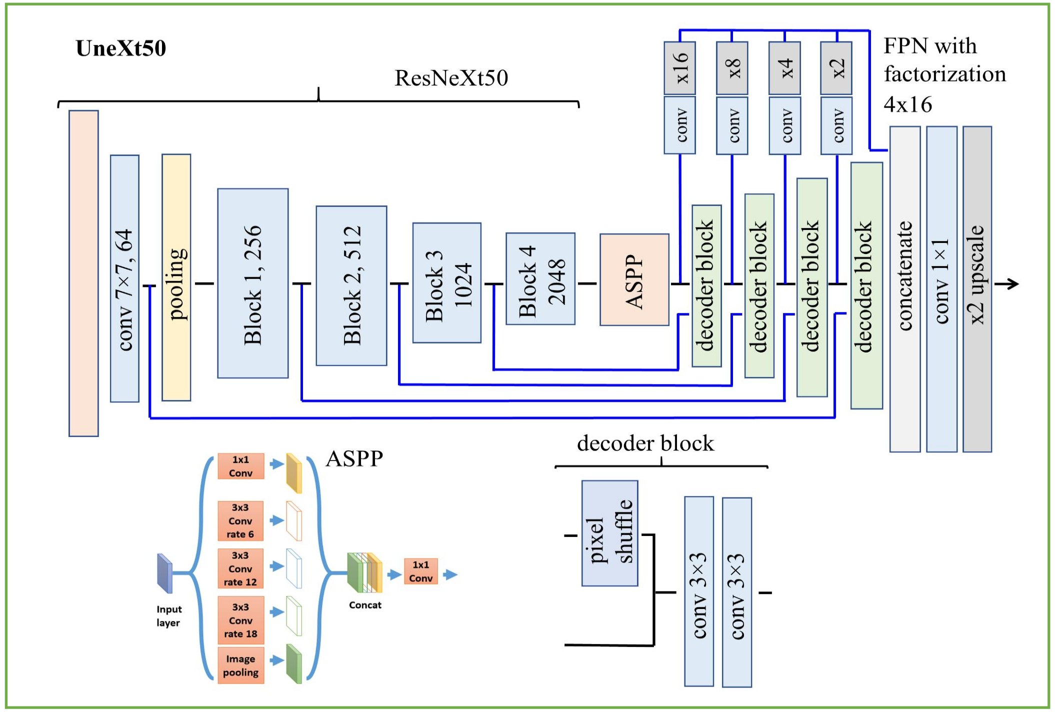
- **A quick review of Unext50 network**

Encoder Part (ResNeXt50)

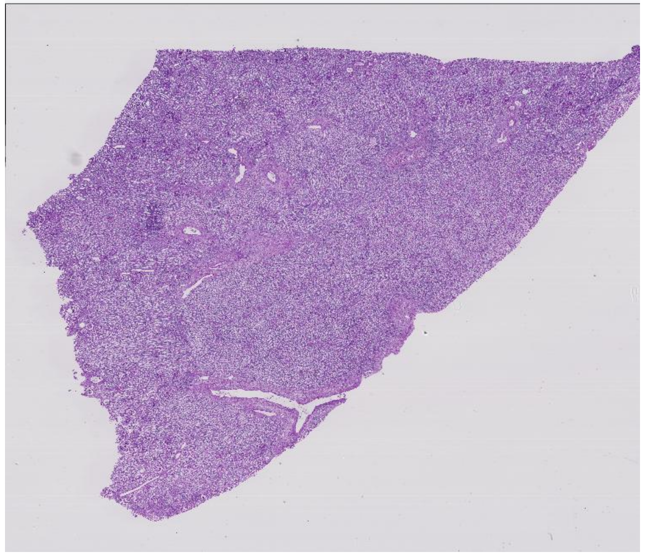
ASPP

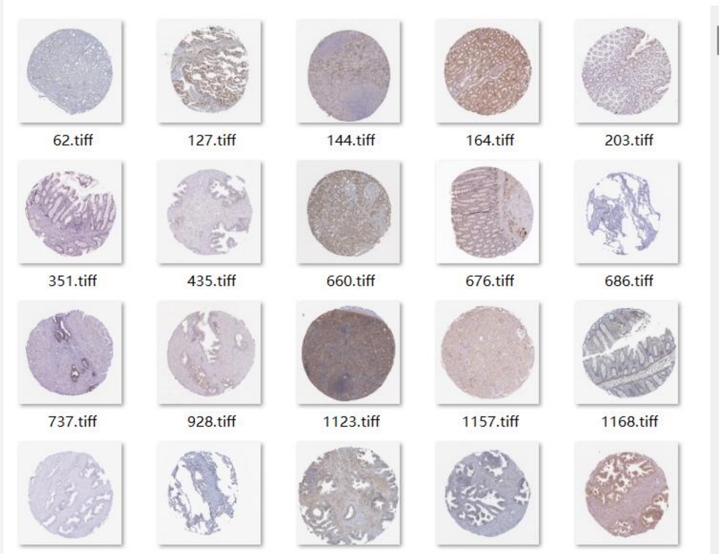
Decoder Part (FPN)

Group C's presentation last time



Similar Competition





First Try:

```
HorizontalFlip(),
VerticalFlip(),
RandomRotate90(),
ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.2, rotate_limit=15, p=0.9,
                 border_mode=cv2.BORDER_REFLECT),
```

Succeeded	0.47

Second Try

```
HorizontalFlip(),
VerticalFlip(),
RandomRotate90(),
ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.2, rotate_limit=15, p=0.9,
                 border_mode=cv2.BORDER_REFLECT),

OneOf([
    OpticalDistortion(p=0.3),
    GridDistortion(p=.1),
    IAAPiecewiseAffine(p=0.3),
], p=0.3),

OneOf([
    HueSaturationValue(10,15,10),
    CLAHE(clip_limit=2),
    RandomBrightnessContrast(),
], p=0.3),
```

Succeeded	0.56

0.54-0.56

https://blog.csdn.net/zhangyuexiang123/article/details/107705311?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216577939321678239531170 3%2522%252C%2522scm%2522%253A%2522 20140713.130102334..%2522%257D&request_id=165779393216782395311703&biz_id=0&ut

■ Third Try:

```
def get_aug(p=1.0):
    return Compose([
        HorizontalFlip(),
        VerticalFlip(),
        RandomRotate90(),
        ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.2, rotate_limit=15, p=0.9,
                         border_mode=cv2.BORDER_REFLECT),
        OneOf([
            ElasticTransform(p=.3),
            GaussianBlur(p=.3),
            GaussNoise(p=.3),
            OpticalDistortion(p=0.3),
            GridDistortion(p=.1),
            IAAPiecewiseAffine(p=0.3),
        ], p=0.3),
        OneOf([
            HueSaturationValue(15,25,0),
            CLAHE(clip_limit=2),
            RandomBrightnessContrast(brightness_limit=0.3, contrast_limit=0.3),
        ], p=0.3),
    ], p=p)
```

Succeeded 0.57

- Forth try: Cutout

```
def cutout(tensor,alpha=0.5):
    x=int(alpha*tensor.shape[2])
    y=int(alpha*tensor.shape[3])
    center=np.random.randint(0,tensor.shape[2],size=(2))
    #perm = torch.randperm(img.shape[0])
    cut_tensor=tensor.clone()
    cut_tensor[:,:,center[0]-x//2:center[0]+x//2,center[1]-y//2:center[1]+y//2]=0
    return cut_tensor
```

Succeeded	0.55

0.53-0.55

- Principle: Cutout cuts out a random square area of the image and adds zeros to the original image.

paper: https://arxiv.org/pdf/1708.04552.pdf
code: https://github.com/uoguelph-mlrg/Cutout

- During training, apply a square matrix to random positions. The authors argue that this technique encourages the web to take advantage of the whole picture, rather than relying on a small number of specific visual features.
- This technique encourages the network to better utilize the full context of the image, rather than relying on the presence of a small set of specific visual features.

- After several rounds of attempts, the optimal setting of different data sets is different, CIFAR10 is 16, CIFAR100 is 8 and SVHN is 20. The cutout parameters are very important.

1. Still work on cutout

2. Continue to study the winning codes(3rd place)

Models

Starting from iafoss's starter notebooks and changing them to pure pytorch with heavier augmentation, I ensembled 2 sets of 5-fold models: one with resnext50 and one with resnext101.

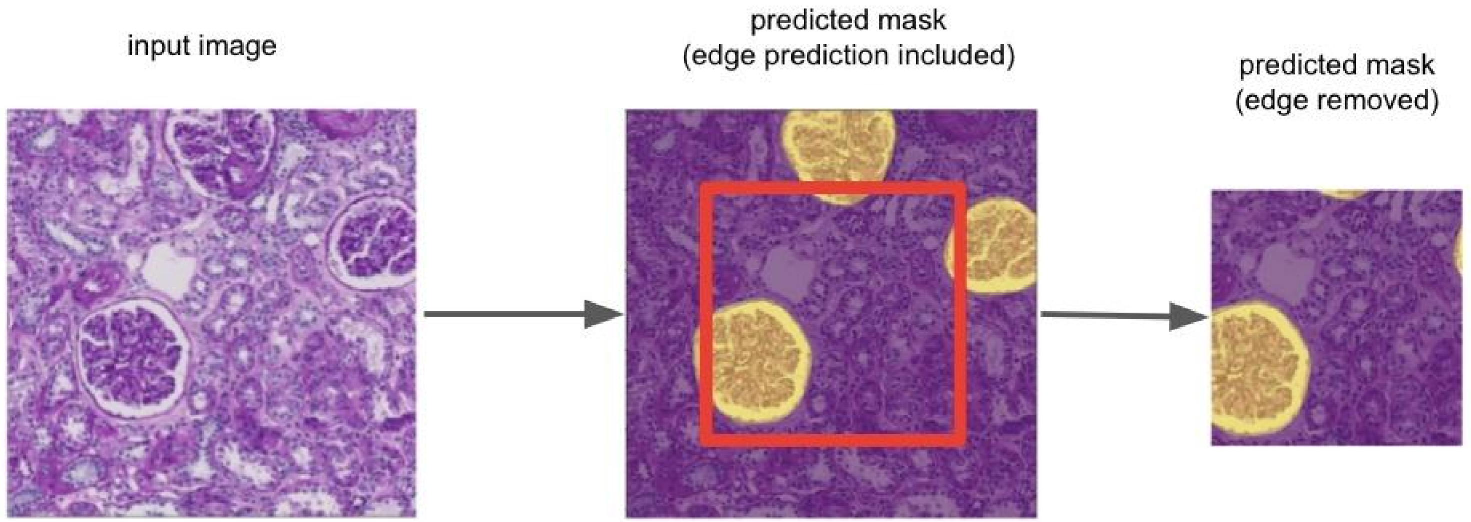
```
models = []
for path in MODELS:
    state_dict = torch.load(path,map_location=torch.device('cpu'))
    model = UneXt50()
    model.load_state_dict(state_dict)
    model.float()
    model.eval()
    model.to(device)
    models.append(model)

del state_dict
```

```
models = []
for path in MODELS_rsxt50:
    state_dict = torch.load(path)
    model = UneXt(m=torchvision.models.resnext50_32x4d(pretrained=False)).cuda()
    model = nn.DataParallel(model)
    model.load_state_dict(state_dict)
    model.float()
    model.eval()
    #model.to(device)
    models.append(model)

for path in MODELS_rsxt101:
    state_dict = torch.load(path)
    model = UneXt(m=ResNet(Bottleneck, [3, 4, 23, 3], groups=32, width_per_group=4)).cuda()
    model = nn.DataParallel(model)
    model.load_state_dict(state_dict)
    model.float()
    model.eval()
    #model.to(device)
    models.append(model)


del state_dict
```



The edge effect is eliminated by using the results in the middle of the predicted results

3. Preprocessing of pathological images—— normalization

Gray histogram normalization

color normalization

Spectral normalization

Thanks for listening