HuBMAP + HPA - Hacking the Human Body Competition Progress Report 6

Presentation of a Report

By

Group C

Xi'an Jiaotong-Liverpool University
西交利物浦大學

Overview

- **Changing the Model Backbone:**
  1. ResNext50 to EfficientNet b5

- **Ensemble Learning:**
  1. *Combining the Predicted Results Of UNEXT50 And EfficientUNet*

- **Dataset Preprocessing:**
  1. *Balanced tile sampling for training (EDIT : masked area is balanced)*

- **Future Plan:**
  1. Continue to improve the Ensembled Model

Part 1: Changing the Model Backbone

```python
class EffUnet(nn.Module):
    def __init__(self, model_name, stride=1):
        super().__init__()

        cfg = efficient_net_encoders[model_name]
        stage_idxs = cfg['stage_idxs']
        out_channels = cfg['out_channels']

        self.encoder = EfficientNetEncoder(stage_idxs, out_channels, model_name)

        #aspp with customized dilatations
        self.aspp = ASPP(out_channels[-1], 256, out_c=384,
                         dilations=[stride*1, stride*2, stride*3, stride*4])
        self.drop_aspp = nn.Dropout2d(0.5)
        #decoder
        self.dec4 = UnetBlock(384, out_channels[-2], 256)
        self.dec3 = UnetBlock(256, out_channels[-3], 128)
        self.dec2 = UnetBlock(128, out_channels[-4], 64)
        self.dec1 = UnetBlock(64, out_channels[-5], 32)
        self.fpn = FPN([384, 256, 128, 64], [16]*4)
        self.drop = nn.Dropout2d(0.1)
        self.final_conv = ConvLayer(32+16*4, 1, ks=1, norm_type=None, act_cls=None)

    def forward(self, x):
        enc0, enc1, enc2, enc3, enc4 = self.encoder(x)[-5:]
        enc5 = self.aspp(enc4)
        dec3 = self.dec4(self.drop_aspp(enc5), enc3)
        dec2 = self.dec3(dec3,enc2)
        dec1 = self.dec2(dec2,enc1)
        dec0 = self.dec1(dec1,enc0)
        x = self.fpn([enc5, dec3, dec2, dec1], dec0)
        x = self.final_conv(self.drop(x))
        x = F.interpolate(x,scale_factor=2,mode='bilinear')
        return x
```

- **Inference Trick:**
  1. Expansion Tile

- **Data Augmentation Strategy:**
  1. The Same As UNEXT50

[Inference]-HuBMAP fast.ai starter (EfficientNet) (version 1/16) — Succeeded — 0.68
4 days ago by Juntuo Wang
Notebook [Inference]-HuBMAP fast.ai starter (EfficientNet) | Version 1

[Inference] - FastAI Baseline (version 13/20) — Succeeded — 0.66
9 days ago by Juntuo Wang
Notebook [Inference] - FastAI Baseline | Version 13

Part 1: Changing the Model Backbone

```python
class EffUnet(nn.Module):
    def __init__(self, model_name, stride=1):
        super().__init__()

        cfg = efficient_net_encoders[model_name]
        stage_idxs = cfg['stage_idxs']
        out_channels = cfg['out_channels']

        self.encoder = EfficientNetEncoder(stage_idxs, out_channels, model_name)

        #aspp with customized dilatations
        self.aspp = ASPP(out_channels[-1], 256, out_c=384,
                         dilations=[stride*1, stride*2, stride*3, stride*4])
        self.drop_aspp = nn.Dropout2d(0.5)
        #decoder
        self.dec4 = UnetBlock(384, out_channels[-2], 256)
        self.dec3 = UnetBlock(256, out_channels[-3], 128)
        self.dec2 = UnetBlock(128, out_channels[-4], 64)
        self.dec1 = UnetBlock(64, out_channels[-5], 32)
        self.fpn = FPN([384, 256, 128, 64], [16]*4)
        self.drop = nn.Dropout2d(0.1)
        self.final_conv = ConvLayer(32+16*4, 1, ks=1, norm_type=None, act_cls=None)

    def forward(self, x):
        enc0, enc1, enc2, enc3, enc4 = self.encoder(x)[-5:]
        enc5 = self.aspp(enc4)
        dec3 = self.dec4(self.drop_aspp(enc5), enc3)
        dec2 = self.dec3(dec3,enc2)
        dec1 = self.dec2(dec2,enc1)
        dec0 = self.dec1(dec1,enc0)
        x = self.fpn([enc5, dec3, dec2, dec1], dec0)
        x = self.final_conv(self.drop(x))
        x = F.interpolate(x,scale_factor=2,mode='bilinear')
        return x
```

- **Pretrain The Model On Similar Dataset:**

  1. Hacking The Kdiney

[Inference]-HuBMAP fast.ai starter (EfficientNet) (version 1/16) — Succeeded — 0.68

4 days ago by Juntuo Wang

Notebook [Inference]-HuBMAP fast.ai starter (EfficientNet) | Version 1

[Inference]-HuBMAP fast.ai starter (EfficientNet) (version 8/16) — Succeeded — 0.62

2 days ago by Juntuo Wang

Notebook [Inference]-HuBMAP fast.ai starter (EfficientNet) | Version 8

- **Need Further Exporetion In The Future:**

Part 2: Ensemble Learning

```python
models = []
for path in MODELS:
    state_dict = torch.load(path,map_location=torch.device('cpu'))

    model = EffUnet('efficientnet-b5')
    model.load_state_dict(state_dict)
    model.float()
    model.eval()
    model.to(device)
    models.append(model)

for path in MODELS1:
    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
```

- **Combine The Results Of Different Model:**

  1. *Combining the Predicted Results Of UNEXT50 And EfficientUNet*

[Inference]-HuBMAP fast.ai starter (EfficientNet) (version 1/16) 4 days ago by Juntuo Wang Notebook [Inference]-HuBMAP fast.ai starter (EfficientNet) | Version 1	Succeeded	0.68	☐
[Inference]-HuBMAP fast.ai starter (EfficientNet) (version 20/20) 19 hours ago by Juntuo Wang Notebook [Inference]-HuBMAP fast.ai starter (EfficientNet) | Version 20	Succeeded	0.70	☐

Part 3: Dataset Preprocessing

The Problem Of The Dataset:

Large empty spaces in labeled areas

5

Posted in hubmap-organ-segmentation 21 days ago

In preprocessing the data I noticed that there are often large empty spaces labeled as foreground. They happen most often in prostate, lung images and sometimes in intestine as well.

In my own preprocessing I am trying to discard those labels at empty pixel locations (colored blue) since they probably will confuse the model. But I am not sure if this would be the best strategy for doing inference on test data.

Has anyone tried any strategy for these "empty" labels?

Question about tiles

0

Posted in hubmap-organ-segmentation 10 days ago

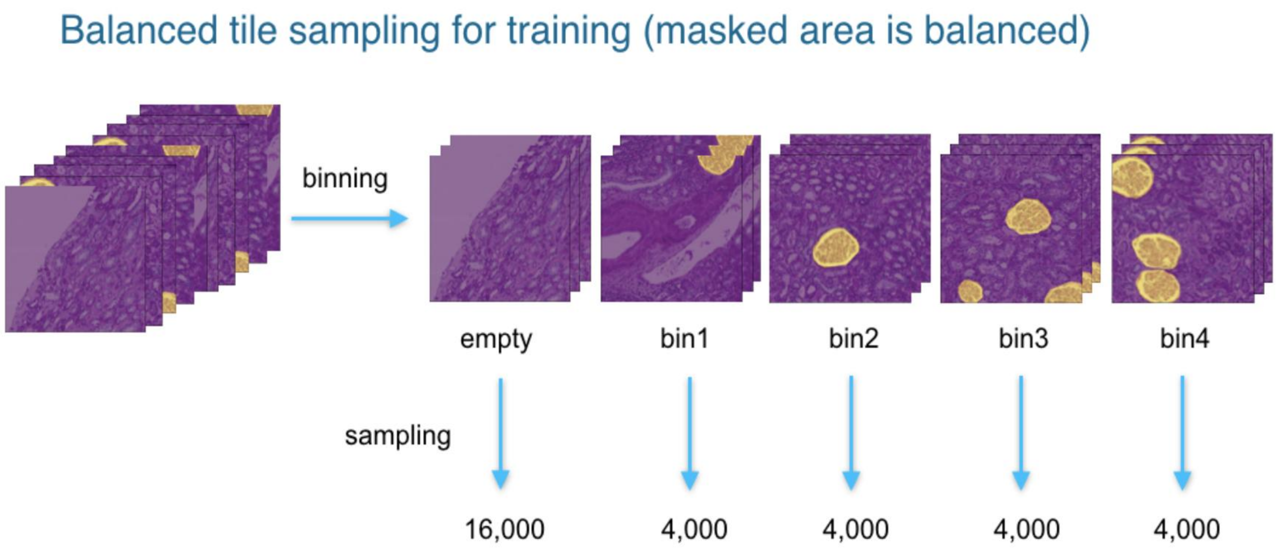
Thanks for this https://www.kaggle.com/code/thedevastator/converting-to-256×256, I learned the way that a large medical image can be divided into small tiles.
This approach works well in some other competitions, but it doesn't work for me in this competitions.
I did spilt the large image into tiles and use them to train, but this way lead my model a negative effect, which is worse than I remain the large image.
I don't know if any one else has the same situation as me. Any advice is greatly appreciated.

Data Imbalance:

There is an imbalance in the number of Empty tiles and Mask tiles

Part 3: Dataset Preprocessing (Further Improvment)

Possible Solution:



Processing:

- Sample tiles from these masked and unmasked data equally
- Sample tiles from these tiles have very small masked area,
- tiles have small masked area,
- tiles have some masked area,
- tiles have large masked area equally

Part 3: Dataset Preprocessing (Further Improvement)

- ## The Problem Of The Dataset:

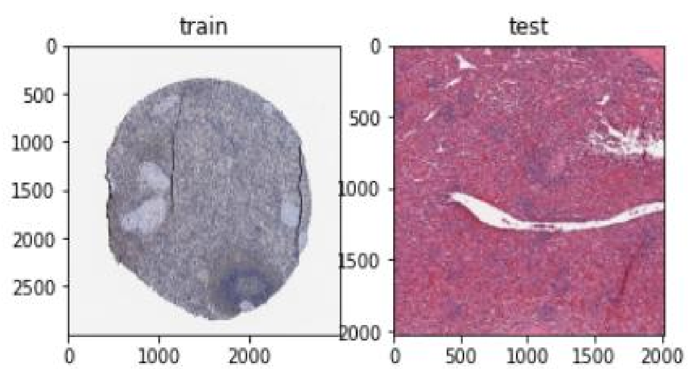
An overview: Stain Normalization Techniques

Posted in hubmap-organ-segmentation 4 days ago

18

Biology | Adversarial Learning | Deep Learning

We might have noticed that there are not only texture differences but also color differences due to different stain protocols in the train and test set. So, I wanted to create this post to have an overview of varying normalization techniques for histopathological images.



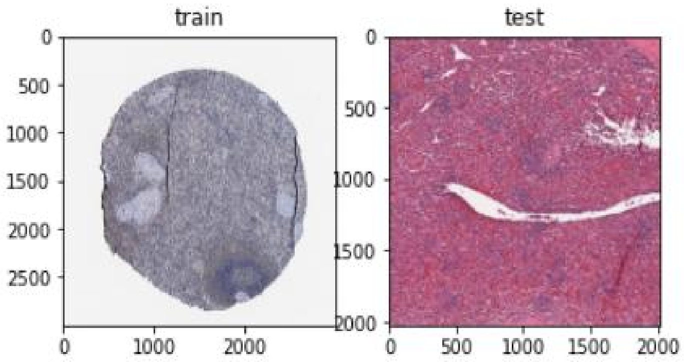
- ## Color Differences:

Color differences due to different stain protocols in the train and test set

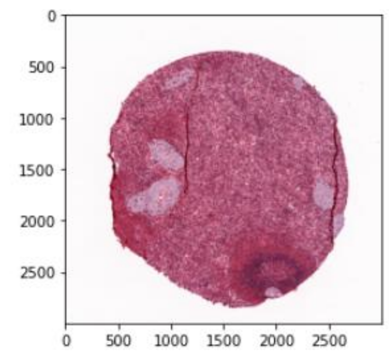
Part 3: Dataset Preprocessing (Further Improvement)

- Possible Solution
- Stain Normalization

- Before



- After



Part 4: Future Plan

- Try to implement the Preprocessing Introduced In Part 3

- Continue to Improve the Ensemble Model