Final Proposal

tags: `NTU_ML2018FALL`

題目

Human Protein Atlas Image Classification

LINK

- HW
  https://docs.google.com/presentation/d/1AqyQoj9JjsLubaTG7cytsdK7pDx3N0PtTPB7BFJKXh8/edit#slide=id.p (https://docs.google.com/presentation/d/1AqyQoj9JjsLubaTG7cytsdK7pDx3N0PtTPB7BFJKXh8/edit#slide=id.p)
- Kaggle
  https://www.kaggle.com/c/human-protein-atlas-image-classification/leaderboard (https://www.kaggle.com/c/human-protein-atlas-image-classification/leaderboard)
- Github
  https://github.com/linnil1/ML2018FALL_FINAL (https://github.com/linnil1/ML2018FALL_FINAL)

Deadline

Time	Title	Grade
11/16 12:00:00 Fri.	Final Project Rules Announcement	
12/14 23:59:59 Fri.	Final Project Proposal Deadline	
12/14 23:59:59 Fri.	Final Project Early Baseline Deadline	Pass Simple +1%
01/04 23:59:59 Fri.	Final Project Ranking and strong baseline Deadline	Pass Strong +2%
01/11 23:59:59 Fri.	Final Presentation (top-3)	
01/14 23:59:59 Mon.	Final Project Github Deadline (Report & Github)	Pass Simple +3% Pass Strong +3%

Members

B04611017 linnil1
D06922023 yan12125
R07943018 denny850306
R06922051 hiiragi4000

Trying some pretrained models

- ResNet
  https://arxiv.org/abs/1512.03385 (https://arxiv.org/abs/1512.03385)
- InceptionV3
  https://arxiv.org/pdf/1602.07261.pdf (https://arxiv.org/pdf/1602.07261.pdf)
  https://arxiv.org/pdf/1512.00567v3.pdf (https://arxiv.org/pdf/1512.00567v3.pdf)
- DenseNet
  https://arxiv.org/pdf/1608.06993.pdf (https://arxiv.org/pdf/1608.06993.pdf)

發現

channel

因為是 4channel 的 image, 所以除了下載pretrained models 套用以外，還必須把第一層改掉

loss function

因為 cross-entropy 並不支援 多種分類
所以自己弄個 FocalLoss 出來
Use FocalLoss as loss function
https://arxiv.org/pdf/1708.02002.pdf (https://arxiv.org/pdf/1708.02002.pdf)

```
    def forward(self, pred, targ):
        x = torch.zeros(targ.size()).cuda()
        x[targ == 1] = pred[targ == 1]
        x[targ == 0] = 1 - pred[targ == 0]
        x[x < self.eps] += self.eps
        return -((1 - x).pow(self.gamma) * x.log()).sum(dim=1).mean()
```

Experiments

1. test1
   用 densenet 去掉 Yellow
   第一個epoch
   train : 0.92893124(28000)
   valid : 0.9419643(3072)
   kaggle score: 0.112
2. test2
   用 densenet 加上第四個 channel
   並使用 data augmentation
   https://github.com/mdbloice/Augmentor (https://github.com/mdbloice/Augmentor)

第11個epoch

```
 28000/ 28000 100% acc: 0.93 loss: 1.87 f1: 0.12864
  3072/  3072 100% acc: 0.93 loss: 1.91 f1: 0.13460
```

然後 fine-tune

```
epoch: 21
Train:  28000/ 28000 100% acc: 0.9594 loss: 0.8668 f1: 0.33122992515563965
Valid:   3072/  3072 100% acc: 0.9545 loss: 0.9480 f1: 0.31907451152801514
```

kaggle score: 0.263

加 normalize
kaggle score: 0.272
差一點點點

add dropout

```
epoch:27/30
Train:  28000/ 28000 100% acc: 0.9637 loss: 0.7422 f1: 0.4714517891407013
Valid:   3072/  3072 100% acc: 0.9602 loss: 0.9357 f1: 0.40074411034584045
```

我發現他只會輸出 `2`
kaggle score =0.005

- test3
  使用 desenet201
  後面加 relu
  lr = 0.01

```
epoch:15/21
Train:  28000/ 28000 100% acc: 0.9350 loss: 1.5912 f1: 0.12758739292621613
Valid:   3072/  3072 100% acc: 0.9387 loss: 1.6050 f1: 0.11201918870210648
```

- test5
  使用其他的 loss function
  https://www.kaggle.com/rejpalcz/best-loss-function-for-f1-score-metric
  (https://www.kaggle.com/rejpalcz/best-loss-function-for-f1-score-metric)
  不過這個很難train
  lr = 0.001

```
epoch:4/11
Train:  28000/ 28000 100% acc: 0.0583 loss: 0.8733 f1: 0.1413094401359558
Valid:   3072/  3072 100% acc: 0.0588 loss: 0.8802 f1: 0.1566707342863083

epoch:15/31
Train:  28000/ 28000 100% acc: 0.8313 loss: 0.8322 f1: 0.18997500836849213
Valid:   3072/  3072 100% acc: 0.8538 loss: 0.8433 f1: 0.22160808742046356
```

而且 train 很慢

- test8
  使用 densenet201
  把新增的取消掉，只留下 7x7x4 跟 1000x28 這兩個

先train多的layer 共 3epoch
lr = 0.01

```
epoch:3/5
Train:  28000/ 28000 100% acc: 0.9417 loss: 1.3850 f1: 0.15147577226161957
Valid:   3072/  3072 100% acc: 0.9379 loss: 1.4734 f1: 0.15859663486480713
```

再來 train 全部
lr = 0.0001

```
epoch:8/15
Train:  28000/ 28000 100% acc: nan loss: 0.8582 f1: 0.3331504762172699
Valid:   3072/  3072 100% acc: nan loss: 0.4307 f1: 0.3795818090438843
```

kaggle score: 0.243

```
epoch:10/15
Train:  28000/ 28000 100% acc: nan loss: 0.8152 f1: 0.3935442566871643
Valid:   3072/  3072 100% acc: nan loss: 0.4218 f1: 0.41416773200035095
```

kaggle scroe: 0.244
應該是overfit了

TODO

目前還沒過 simple baseline

所以我想要 把 loss function 改成 macro F1 score 的 loss 加上 focalloss， 說不定效果更好。

在後面多疊幾層，或是增加 dropout rate，使效果更好。

data argumentation 也要修

Reference

https://www.kaggle.com/iafoss/pretrained-resnet34-with-rgby-0-460-public-lb
(https://www.kaggle.com/iafoss/pretrained-resnet34-with-rgby-0-460-public-lb)