[Pratice Exercise]

Given a pre-defined net, train and obtain validation and testing results.

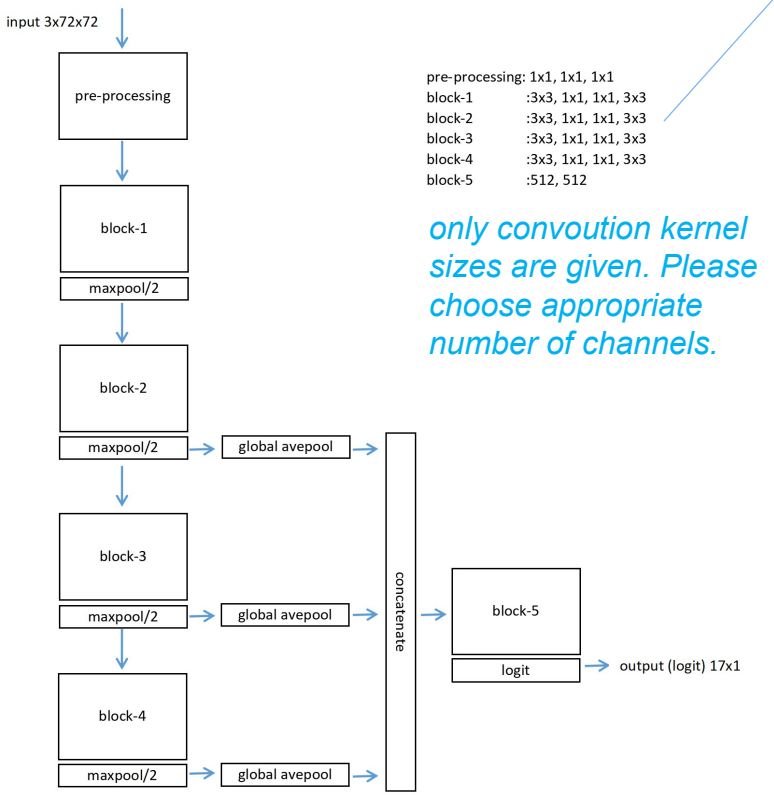
---

step.1

- resize the jpg train and test set from 256x256 to 72x72. Divide train set into 37479 training and 3000 validation samples.

step.2

- implement the following net in pytorch.



pre-processing: 1x1, 1x1, 1x1
block-1 :3x3, 1x1, 1x1, 3x3
block-2 :3x3, 1x1, 1x1, 3x3
block-3 :3x3, 1x1, 1x1, 3x3
block-4 :3x3, 1x1, 1x1, 3x3
block-5 :512, 512

only convoution kernel sizes are given. Please choose appropriate number of channels.

```
class MyNet(nn.Module):

    def __init__(self, in_shape=(3,72,72), num_classes=17):
        super(MyNet, self).__init__()
        in_channels, height, width = in_shape
        stride=1

        self.preprocess = nn.Sequential(
            nn.Conv2d(in_channels, ??, kernel_size=1, stride=1, padding=0),
            ???  #choose approiate activation, etc
            nn.Conv2d(??, ??, kernel_size=1, stride=1, padding=0),
            ???
            nn.Conv2d(??, ??, kernel_size=1, stride=1, padding=0),
            ???
        )

        self.block1 = nn.Sequential(
            ???
            nn.MaxPool2d(kernel_size=2, stride=2),
        )

        self.block2 = nn.Sequential(
            ???
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        ...

        self.block5 = nn.Sequential(
            nn.Linear(???, 512),
            ???
            nn.Linear(512, 512),
            ???
        )

        self.logit = nn.Linear(512, num_classes)


    def forward(self, x):

        out = self.preprocess(x)
        out = self.block1(out)
        out = self.block2(out)
        ...
        out = self.logit(out)
        return out
```

- Hints:
  - choose appropriate number of channels
  - choose appropriate activiation functions
  - you may need to add dropout, batch nornmalisation, etc
  - what is global average pooling?

    https://www.quora.com/What-is-global-average-pooling

step.3

- Implement training code to learn the net using stocastic gradient descent

```
optimizer = optim.SGD(net.parameters(), lr=?? , momentum=0.9, weight_decay=0.0005)

...

for epoch in range(num_epoches):  # loop over the dataset multiple times

    ...
    for it, (images, labels, indices) in enumerate(train_loader, 0):

        logits = net(Variable(images))
        loss  = criterion(logits, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

- Hints:
  - choose appropriate loss function
  - choose appropriate batch size
  - choose appropriate number of epoches or stopping criteria
  - choose appropriate learning rates (may need learning rate scheduler)

- print your results as

epoch	iter	rate	train_loss	f2	valid_loss	f2	min
			0.157	0.855	0.172	0.820	0.5 min
			0.131	0.877	0.155	0.850	0.5 min
			0.159	0.855	0.135	0.870	0.5 min
	???		0.118	0.888	0.138	0.875	0.5 min
			0.120	0.891	0.125	0.890	0.5 min
			0.141	0.868	0.147	0.855	0.5 min
			0.109	0.892	0.125	0.888	0.5 min
			0.117	0.904	0.148	0.860	0.5 min
			0.142	0.853	0.132	0.878	0.5 min
			0.114	0.918	0.143	0.855	0.5 min
			0.108	0.913	0.137	0.868	0.5 min
			0.103	0.915	0.135	0.877	0.5 min
			0.106	0.907	0.107	0.90	
			0.126	0.889	0.112	0.90	
	???		0.105	0.895	0.111	0.90	
			0.110	0.907	0.110	0.90	
			0.102	0.913	0.120	0.89	
			0.102	0.912	0.120	0.88	
			0.145	0.876	0.147	0.86	
			0.097	0.927	0.122	0.88	
			0.127	0.885	0.106	0.91	
			0.107	0.908	0.096	0.92	
			0.097	0.925	0.095	0.91	
			0.103	0.904	0.093	0.921	0.5 min
			0.115	0.887	0.094	0.918	0.5 min
			0.109	0.913	0.093	0.920	0.5 min
			0.117	0.905	0.096	0.917	0.5 min
			0.089	0.924	0.094	0.920	0.5 min
			0.089	0.925	0.092	0.921	0.5 min
	???		0.081	0.942	0.093	0.922	0.5 min
			0.095	0.934	0.092	0.921	0.5 min
			0.087	0.936	0.091	0.923	0.5 min
			0.105	0.895	0.090	0.923	0.5 min
			0.088	0.913	0.090	0.922	0.5 min
			0.114	0.915	0.090	0.923	0.5 min
			0.104	0.895	0.090	0.922	0.5 min
			0.082	0.935	0.090	0.923	0.5 min

you need to implement f2 measure

https://www.kaggle.com/c/planet-understanding-the-amazon-from-space#evaluation

Submissions will be evaluated based on their mean F_2 score. The F score, commonly used in information retrieval, measures accuracy using the precision p and recall r. Precision is the ratio of true positives tp to all predicted positives $tp + fp$. Recall is the ratio of true positives to all actual positives $tp + fn$. The F_2 score is given by

$$(1+\beta^2)\frac{pr}{\beta^2 p + r} \text{ where } p = \frac{tp}{tp+fp}, \quad r = \frac{tp}{tp+fn}, \quad \beta = 2.$$

Note that the F_2 score weights recall higher than precision. The mean F_2 score is formed by averaging the individual F_2 scores for each row in the test set.

reference results on validation set is f2=0.923

step.4

- After training the net, apply it on the test set. Submit your results to the kaggle server and report your leader board f2 score.

[Passing mark]

f2 >0.90 on validation set.
leader board score should be within 0.03 from your validation f2 score.

[Bounes]

f2 >0.92 on validation set.
leader board score should be within 0.008 from your validation f2 score.