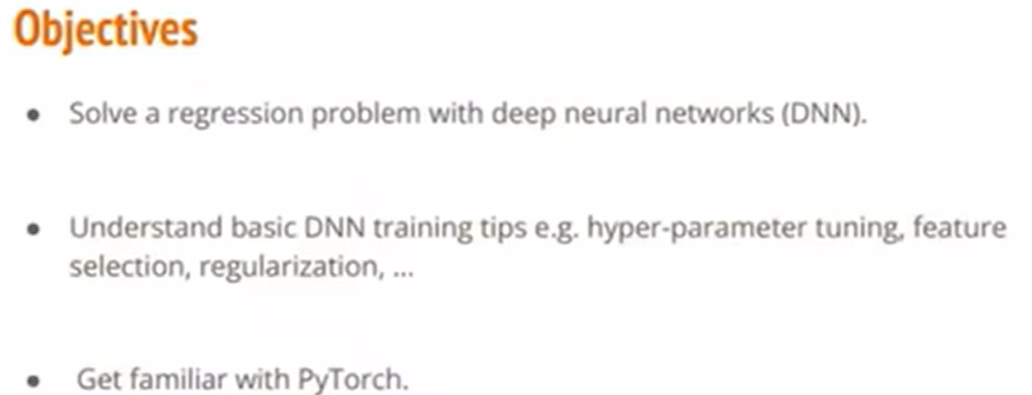
# 第一次作业：

要求：

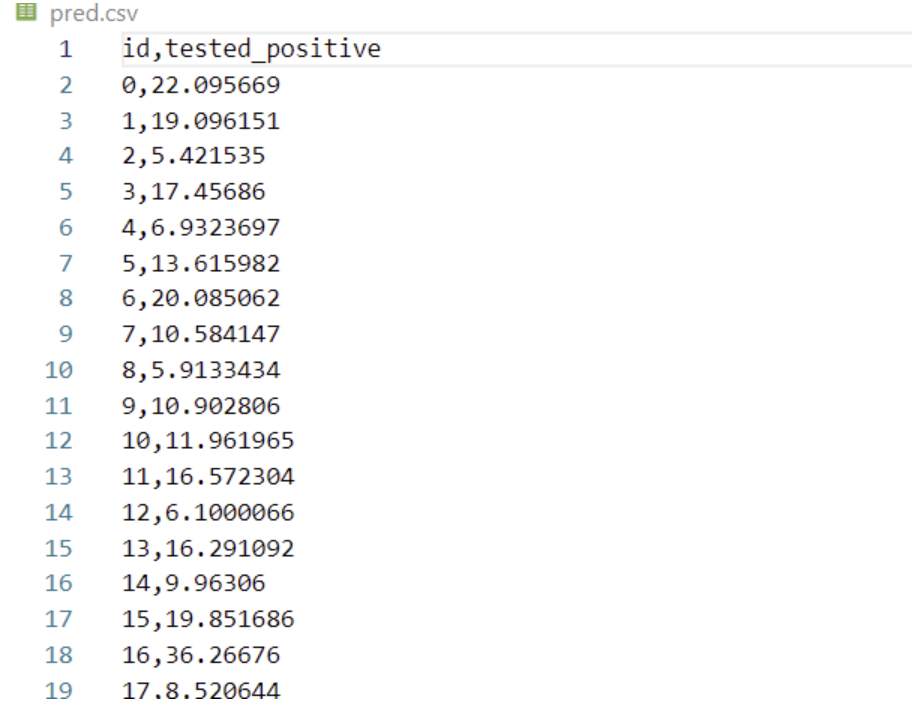


训练：



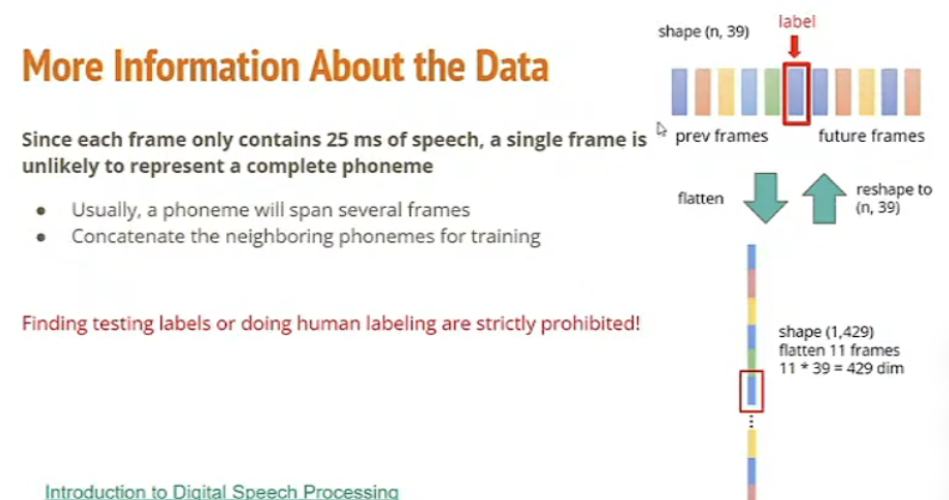
输出：



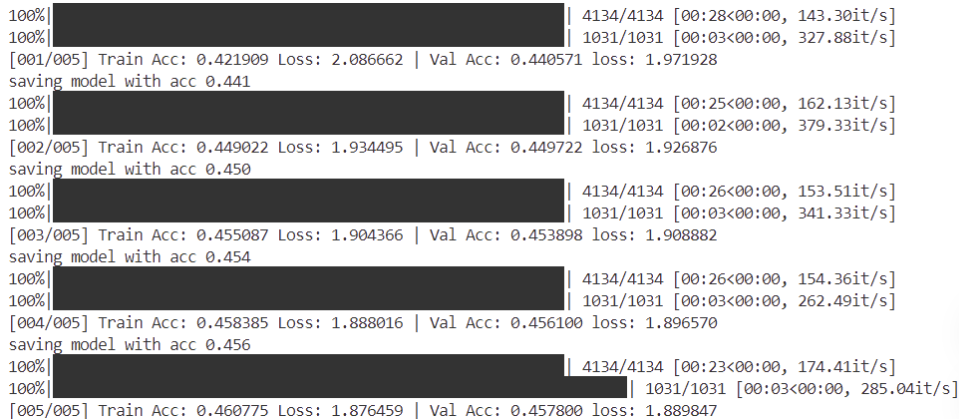


# 第二次作业

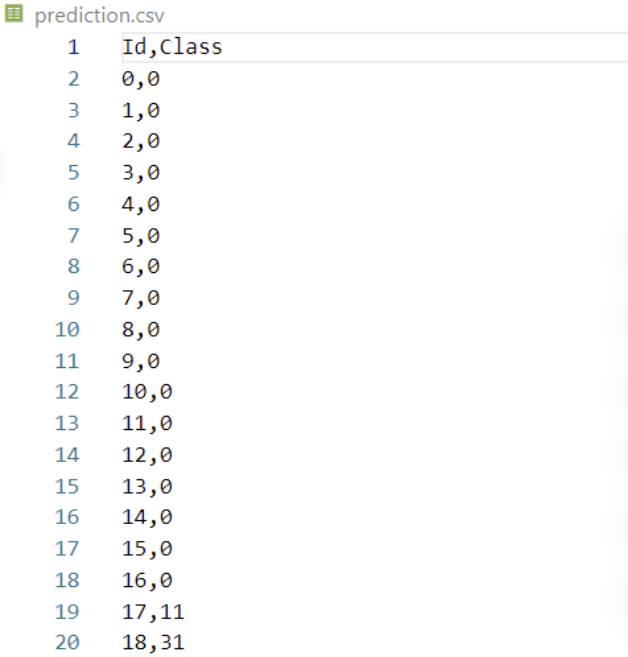
要求：



训练：

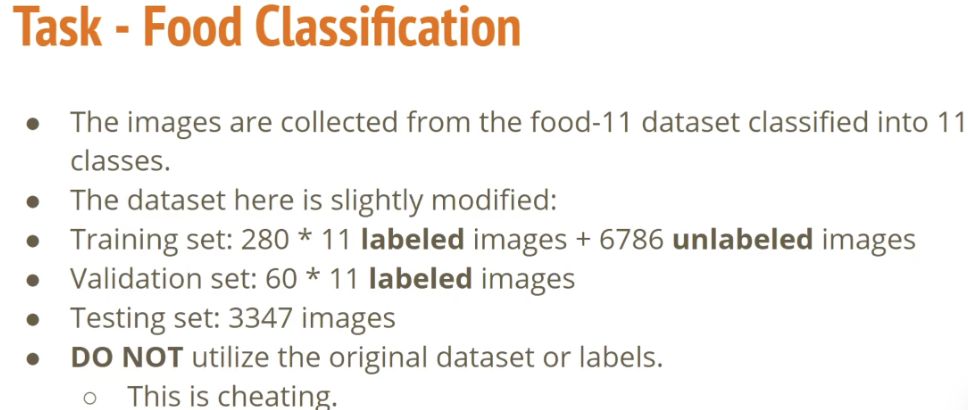


输出：



# 第三次作业

要求：



代码（定义网络）：

**class** **Classifier**(nn.Module):

**def** **\_\_init\_\_**(self):

super(Classifier, self).\_\_init\_\_()

*# torch.nn.Conv2d(in\_channels, out\_channels, kernel\_size, stride, padding)*

*# torch.nn.MaxPool2d(kernel\_size, stride, padding)*

*# input 维度 [3, 128, 128]*

self.cnn = nn.Sequential(

nn.Conv2d(3, 64, 3, 1, 1), *# [64, 128, 128]*

nn.BatchNorm2d(64),

nn.ReLU(),

nn.MaxPool2d(2, 2, 0), *# [64, 64, 64]*

nn.Conv2d(64, 128, 3, 1, 1), *# [128, 64, 64]*

nn.BatchNorm2d(128),

nn.ReLU(),

nn.MaxPool2d(2, 2, 0), *# [128, 32, 32]*

nn.Conv2d(128, 256, 3, 1, 1), *# [256, 32, 32]*

nn.BatchNorm2d(256),

nn.ReLU(),

nn.MaxPool2d(2, 2, 0), *# [256, 16, 16]*

nn.Conv2d(256, 512, 3, 1, 1), *# [512, 16, 16]*

nn.BatchNorm2d(512),

nn.ReLU(),

nn.MaxPool2d(2, 2, 0), *# [512, 8, 8]*

nn.Conv2d(512, 512, 3, 1, 1), *# [512, 8, 8]*

nn.BatchNorm2d(512),

nn.ReLU(),

nn.MaxPool2d(2, 2, 0), *# [512, 4, 4]*

)

self.fc = nn.Sequential(

nn.Linear(512\*4\*4, 1024),

nn.ReLU(),

nn.Linear(1024, 512),

nn.ReLU(),

nn.Linear(512, 11)

)

**def** **forward**(self, x):

out = self.cnn(x)

out = out.view(out.size()[0], -1)

**return** self.fc(out)

1234567891011121314151617181920212223242526272829303132333435363738394041424344

训练：

