南开大学 Nankai University | 国家示范性软件学院 College of Software

南开大学软件学院

人工智能导论

三种神经网络的比较

作者:
崔家贺

讲师:
张玉志

An Assignment submitted for the NKU:

AI

2023 年 1 月 15 日

目录

1 背景介绍 1

2 网络设计 1

2.1 全链接网络 2

2.2 CNN 3

2.3 LSTM 4

3 测试结果 5

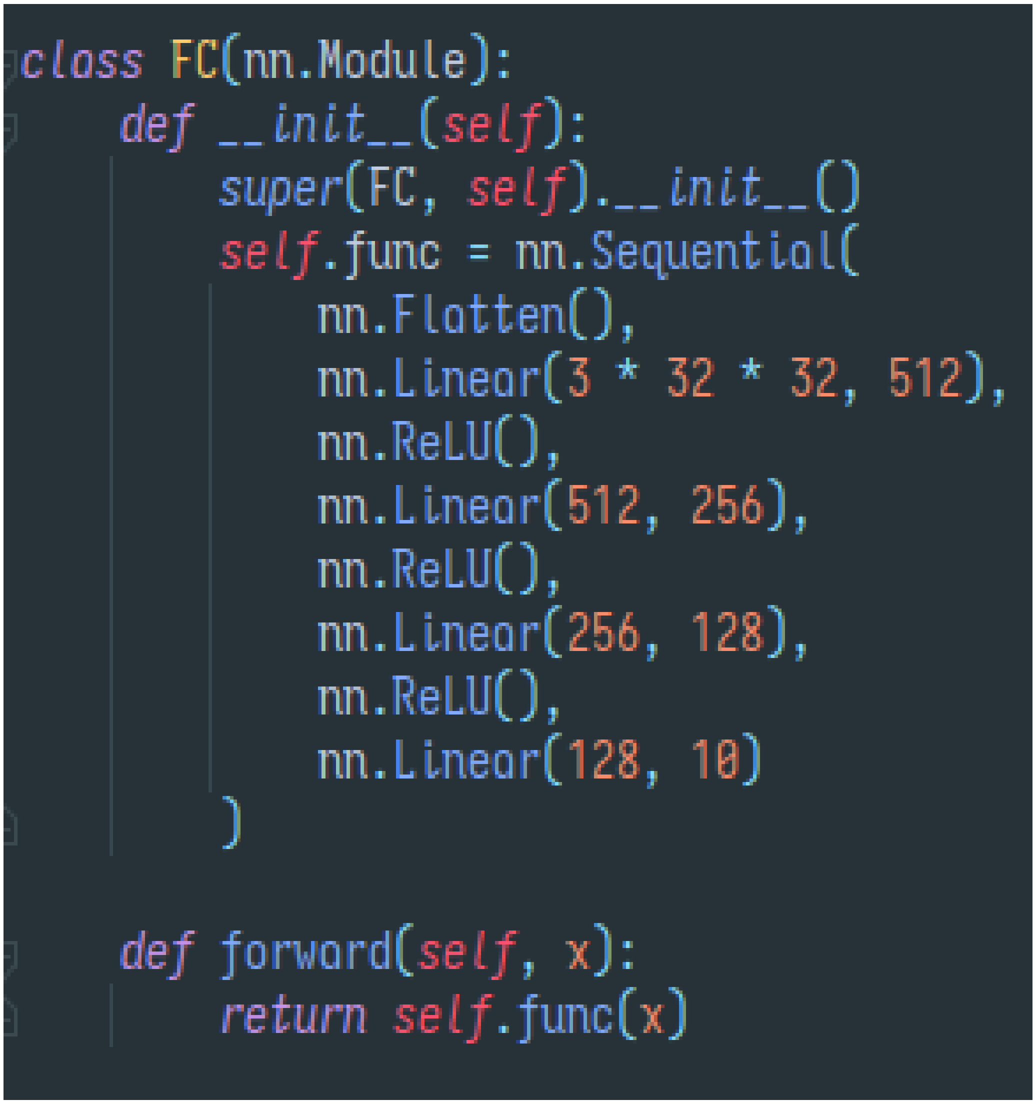
1 背景介绍

对于 CIFAR-10 数据集，我分别使用普通全链接网络，CNN 和 LSTM 训练并进行测试

2 网络设计

所有网络使用 adam 优化器

2.1 全链接网络



```python
class FC(nn.Module):
    def __init__(self):
        super(FC, self).__init__()
        self.func = nn.Sequential(
            nn.Flatten(),
            nn.Linear(3 * 32 * 32, 512),
            nn.ReLU(),
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Linear(128, 10)
        )

    def forward(self, x):
        return self.func(x)
```

图 1: fc

使用 4 层隐含层设计，直接把图片像素点对应到结果

2.2 CNN

采用 resnet18 的配置并进行简化，因为这个数据集的图片像素并不大，所以去掉两层残差连接块，代码如下

```python
class ResBlock(nn.Module):
    def __init__(self, inchannel, outchannel, stride=1):
        super(ResBlock, self).__init__()
        self.left = nn.Sequential(
            nn.Conv2d(inchannel, outchannel, kernel_size=3, stride=stride, padding=1, bias=False),
            nn.BatchNorm2d(outchannel),
            nn.ReLU(inplace=True),
            nn.Conv2d(outchannel, outchannel, kernel_size=3, stride=1, padding=1, bias=False),
            nn.BatchNorm2d(outchannel)
        )
        self.shortcut = nn.Sequential()
        if stride != 1 or inchannel != outchannel:
            self.shortcut = nn.Sequential(
                nn.Conv2d(inchannel, outchannel, kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(outchannel)
            )

    def forward(self, x):
        out = self.left(x)
        out = out + self.shortcut(x)
        out = F.relu(out)
        return out
```

图 2: rb

```python
class ResNet18(nn.Module):
    def __init__(self, rb=ResBlock, num_classes=10):
        super(ResNet18, self).__init__()
        self.in_channel = 64
        self.conv1 = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1, bias=False),
            nn.BatchNorm2d(64),
            nn.ReLU()
        )
        self.layer1 = self.make_layer(rb, 64, 1, stride=1)
        # self.layer2 = self.make_layer(rb, 128, 2, stride=2)
        # self.layer3 = self.make_layer(rb, 256, 2, stride=2)
        self.layer4 = self.make_layer(rb, 32, 1, stride=2)
        self.fc = nn.Linear(512, num_classes)

    def make_layer(self, block, channels, num_blocks, stride):
        strides = [stride] + [1] * (num_blocks - 1)
        layers = []
        for stride in strides:
            layers.append(block(self.in_channel, channels, stride))
            self.in_channel = channels
        return nn.Sequential(*layers)

    def forward(self, x):
        out = self.conv1(x)
        out = self.layer1(out)
        # out = self.layer2(out)
        # out = self.layer3(out)
        out = self.layer4(out)
        out = F.avg_pool2d(out, 4)
        out = out.view(out.size(0), -1)
        out = self.fc(out)
        return out
```

图 3: cnn

2.3 LSTM

采用每行是一个 embeed 设计，直接输入 lstm

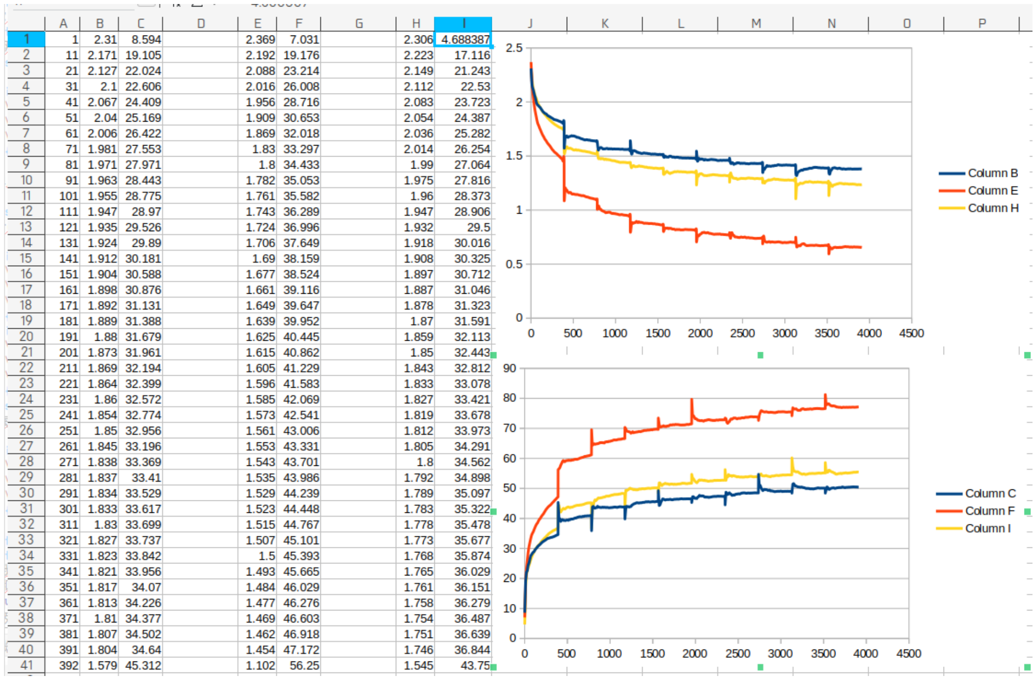
```python
class LSTM(nn.Module):
    def __init__(self):
        super(LSTM, self).__init__()
        self.flat1 = nn.Flatten(1, 2)
        self.flat2 = nn.Flatten()
        self.lstm = nn.LSTM(32, 32, batch_first=True)
        self.l1 = nn.Linear(3072, 64)
        self.l2 = nn.Linear(64, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.flat1(x)
        x, _ = self.lstm(x)
        x = self.flat2(x)
        x = self.relu(self.l1(x))
        x = self.l2(x)
        return x
```

图 4: lstm

3 测试结果

从左到右分别是全链接, CNN, LSTM, 图中上边的图是 loss, 下边是准确率



	A	B	C	D	E	F	G	H	I
1	1	2.31	8.594		2.369	7.031		2.306	4.688387
2	11	2.171	19.105		2.192	19.176		2.223	17.116
3	21	2.127	22.024		2.088	23.214		2.149	21.243
4	31	2.1	22.606		2.016	26.008		2.112	22.53
5	41	2.067	24.409		1.956	28.716		2.083	23.723
6	51	2.04	25.169		1.909	30.653		2.054	24.387
7	61	2.006	26.422		1.869	32.018		2.036	25.282
8	71	1.981	27.553		1.83	33.297		2.014	26.254
9	81	1.971	27.971		1.8	34.433		1.99	27.064
10	91	1.963	28.443		1.782	35.053		1.975	27.816
11	101	1.955	28.775		1.761	35.582		1.96	28.373
12	111	1.947	28.97		1.743	36.289		1.947	28.906
13	121	1.935	29.526		1.724	36.996		1.932	29.5
14	131	1.924	29.89		1.706	37.649		1.918	30.016
15	141	1.912	30.181		1.69	38.159		1.908	30.325
16	151	1.904	30.588		1.677	38.524		1.897	30.712
17	161	1.898	30.876		1.661	39.116		1.887	31.046
18	171	1.892	31.131		1.649	39.647		1.878	31.323
19	181	1.889	31.388		1.639	39.952		1.87	31.591
20	191	1.88	31.679		1.625	40.445		1.859	32.113
21	201	1.873	31.961		1.615	40.862		1.85	32.443
22	211	1.869	32.194		1.605	41.229		1.843	32.812
23	221	1.864	32.399		1.596	41.583		1.833	33.078
24	231	1.86	32.572		1.585	42.069		1.827	33.421
25	241	1.854	32.774		1.573	42.541		1.819	33.678
26	251	1.85	32.956		1.561	43.006		1.812	33.973
27	261	1.845	33.196		1.553	43.331		1.805	34.291
28	271	1.838	33.369		1.543	43.701		1.8	34.562
29	281	1.837	33.41		1.535	43.986		1.792	34.898
30	291	1.834	33.529		1.529	44.239		1.789	35.097
31	301	1.833	33.617		1.523	44.448		1.783	35.322
32	311	1.83	33.699		1.515	44.767		1.778	35.478
33	321	1.827	33.737		1.507	45.101		1.773	35.677
34	331	1.823	33.842		1.5	45.393		1.768	35.874
35	341	1.821	33.956		1.493	45.665		1.765	36.029
36	351	1.817	34.07		1.484	46.029		1.761	36.151
37	361	1.813	34.226		1.477	46.276		1.758	36.279
38	371	1.81	34.377		1.469	46.603		1.754	36.487
39	381	1.807	34.502		1.462	46.918		1.751	36.639
40	391	1.804	34.64		1.454	47.172		1.746	36.844
41	392	1.579	45.312		1.102	56.25		1.545	43.75

图 5: res

可以看出 CNN 效果最好, lstm 次之, 全链接最次