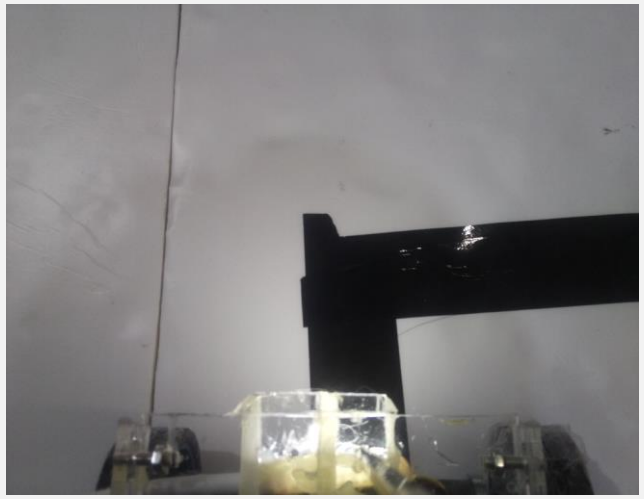
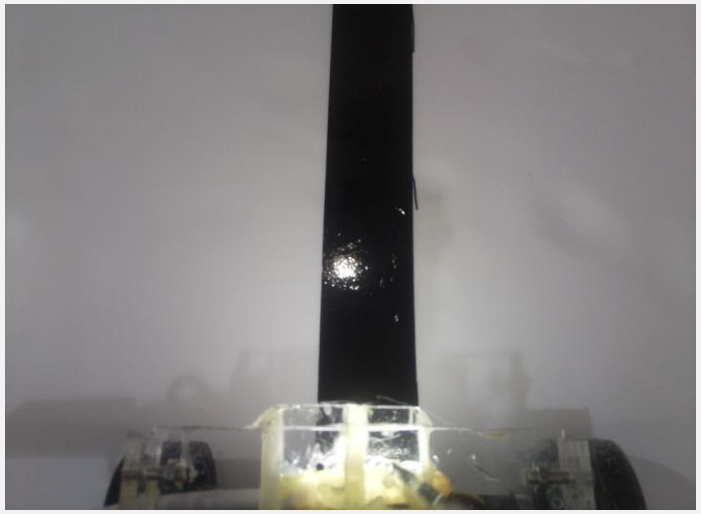
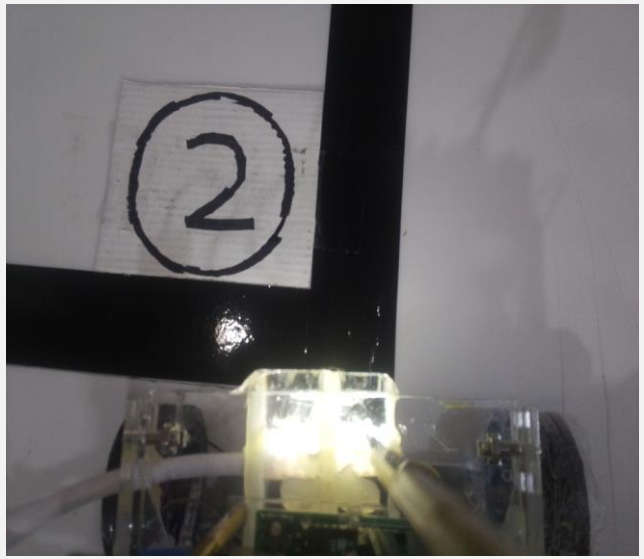
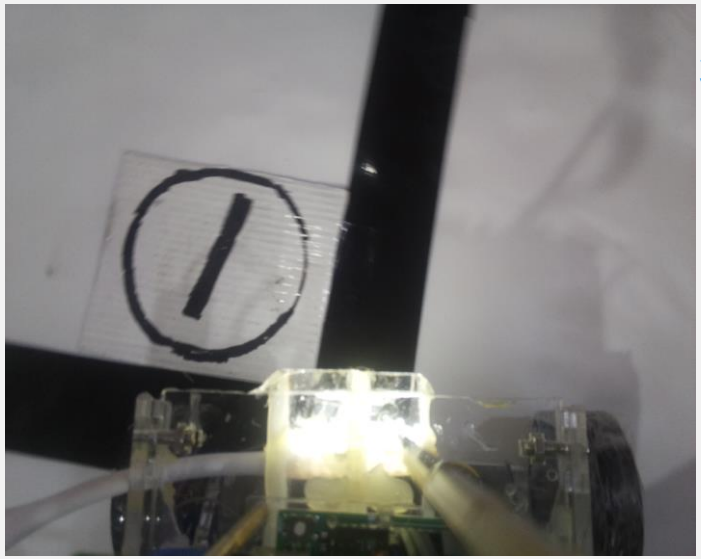
The network structure, visualization and pictures of the robot

Modeling: Image classification









I utilized 3 models:

Model1： line tracking and identify intersections and parking slots

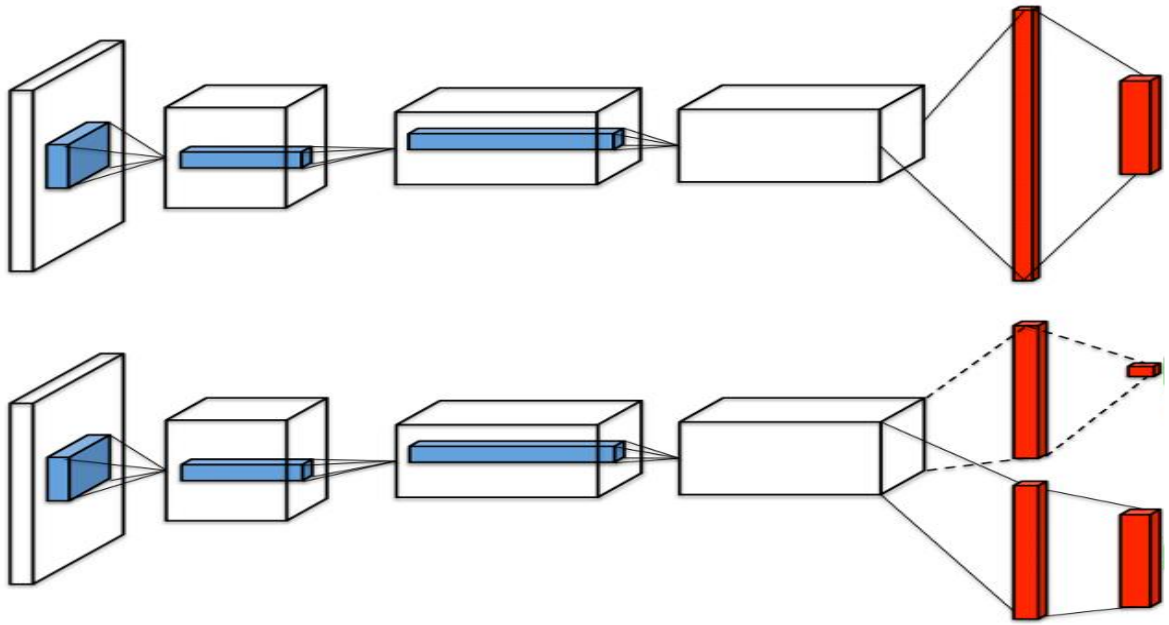
Model2: self parking

Model3: number identification

Choice of network：

CNN + full connection layer

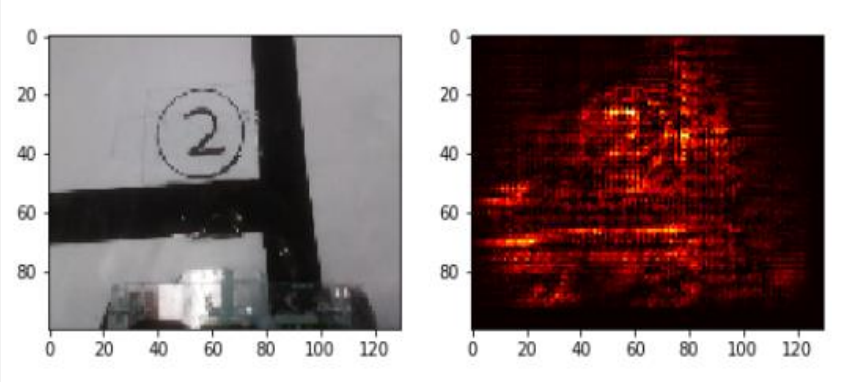
- *Synthesized network: concise codes, sample number need is low, but too much forward propagation time*
- *Seperated network: low forward propagation time.*



Sampling and visualization：

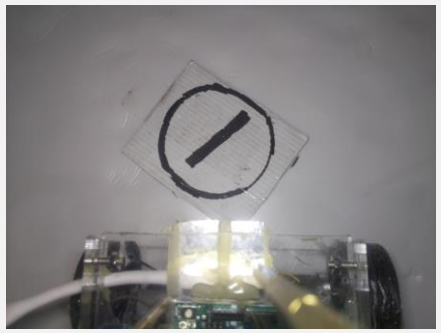
1. Different conditions, different lighting

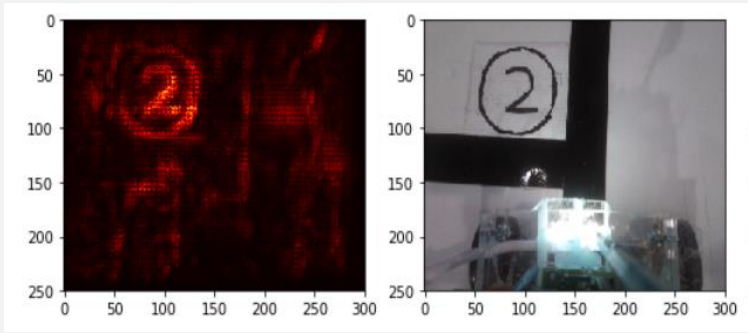
2. Sampling with the lines: failed. Too much points of concentration from the visualized heat colormap.



3. Adding irrelevant samples

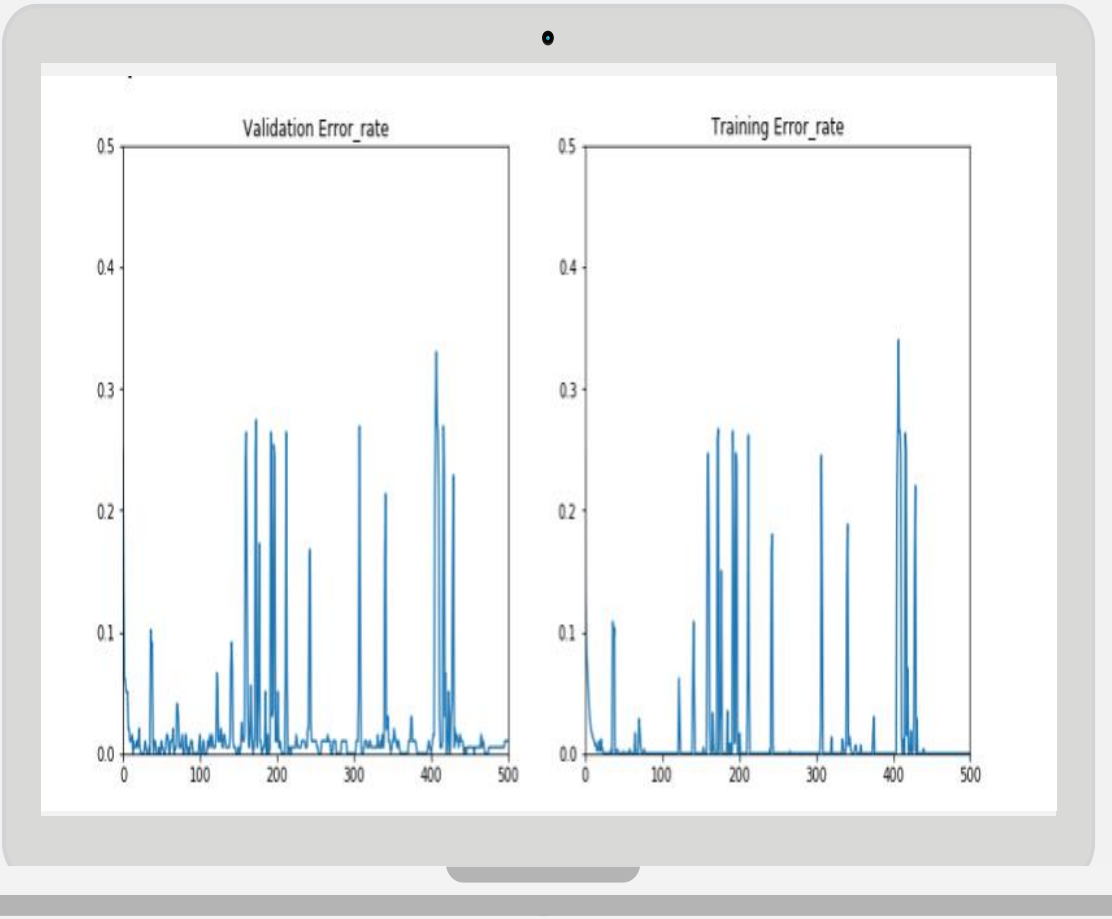
Improved generalization ability, concentrated interest points network

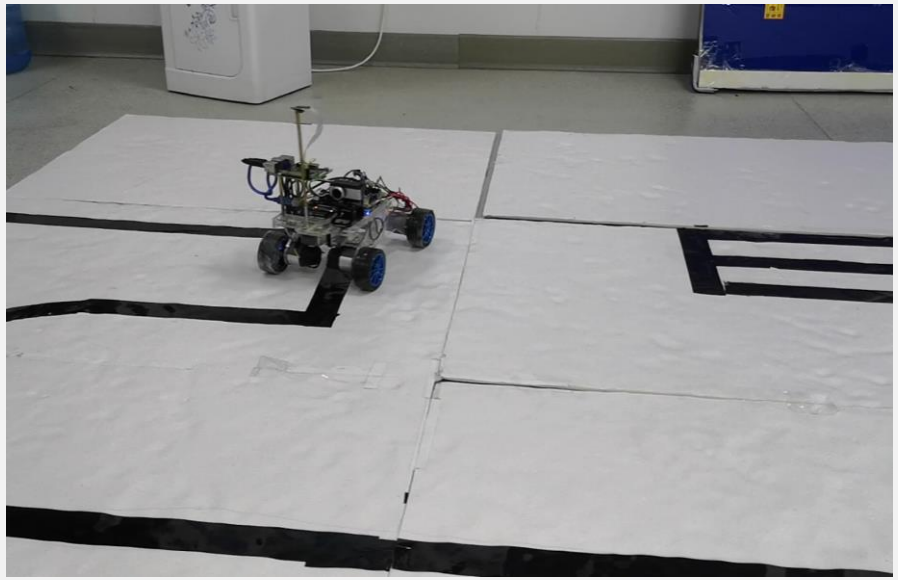


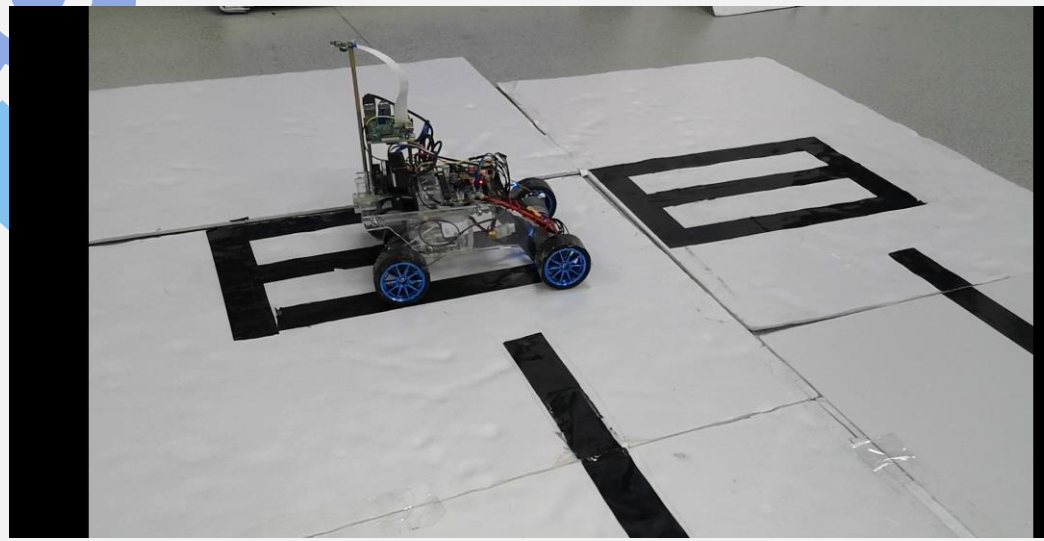


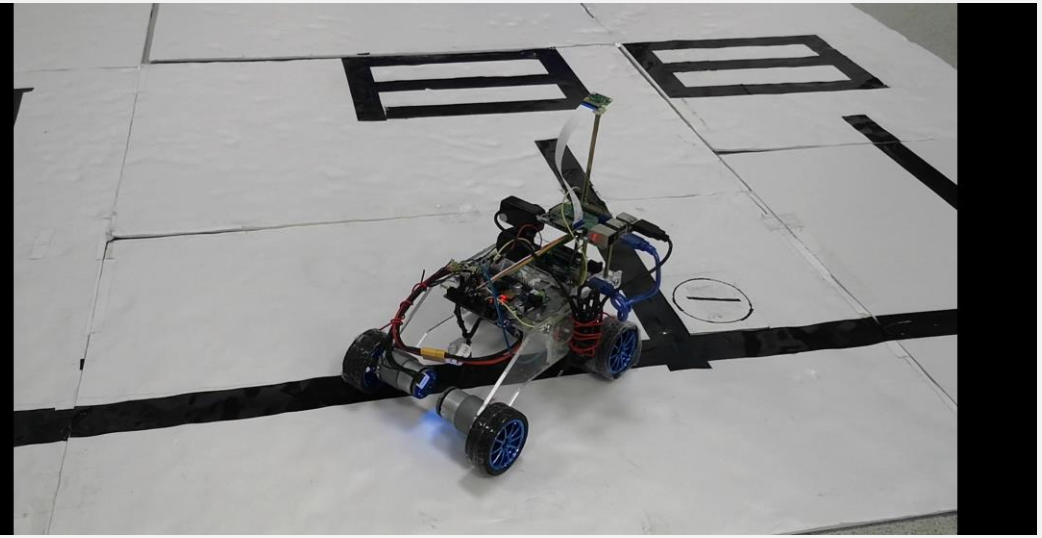
Training and testing the networks

```
#hyperperameters
epochs=500
screen_height=100
screen_width=130##130
#batch_size=10
acc_v=torch.zeros((epochs))
acc_t=torch.zeros((epochs))
policy_net = DQN(screen_height, screen_width,4).to(device)
optimizer = optim.Adam(policy_net.parameters())#RMSprop
policy_net.train()
i=0
import time
tic=time.time()
num=1
for epoch in range(epochs):
    for step,(st,at) in enumerate (loader):
        policy_net.train()
        y_pred=policy_net(st)

        cri = nn.CrossEntropyLoss()
        loss=cri(y_pred,at.long())
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    policy_net.eval()
    y_pred1=torch.argmax(policy_net(_st),1)

    acc=torch.sum(_at.long()==y_pred1.long()).item()/len(_at)
    print('|epoch',epoch,'|loss:',loss,'|accuracy:',acc)
    if epoch%100==99:
        torch.save(policy_net.state_dict(),'F:/Linetrack_project/samples/fii/model%s.h5'%int(num))
        num+=1
    acc_v[epoch]=1-acc
    #if acc>=0.97:
        #break
    y_pred2=policy_net(S)
    y_pred2=torch.argmax(policy_net(S),1)
    acc=torch.sum(A.long()==y_pred2.long()).item()/len(A.long())
    #print('|epoch',epoch,'|loss:',loss,'|accuracy:',acc)
    acc_t[epoch]=acc

toc=time.time()
print('time:',toc-tic)#RMSprop
```









Thanks for reading!