Effective Charging Planning for Electric Vehicles using Deep Reinforcement Learning

By: (Group- 9) 1801CS13 (Balbeer) , 1801CS16 (Mangesh) , 1801CS22 (Hrishabh) , 1801CS30 (Kunj)

Mentor: Shivendu Mishra

Problem Statement

We consider the scenario where electricity prices (directly affecting charging cost in real time) and owner's uncertain commute behaviour make charging the vehicle a practical optimization problem. Aim is to utilize the fluctuation in electricity price to **minimize the cost**.

For example, if the EV is charged when electricity price is low and discharged when the electricity price is high, the reduction in charging costs for the EV owner can be achieved.

EV owner has an intelligent charging device (ICD) at home. When the battery is connected to the ICD, the ICD can perform charging/discharging action according to the proposed method.

Formulation of the Problem

We define the charging process as a Markov Decision Process (MDP), which has unknown transition probabilities due to the randomness of EV owner’s commuting behavior and electricity price

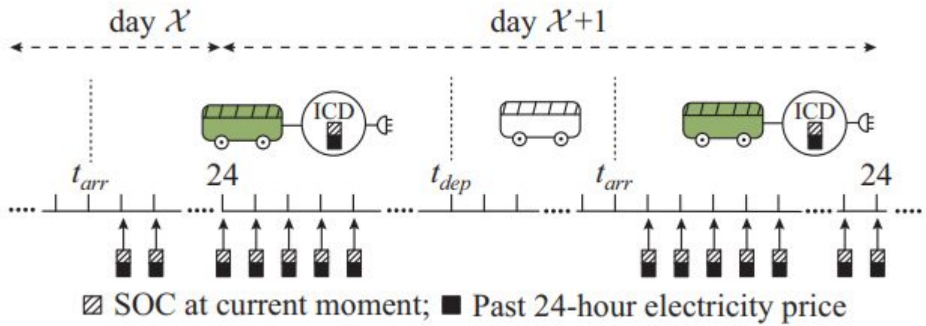


Fig. 1. Single EV charging management model.

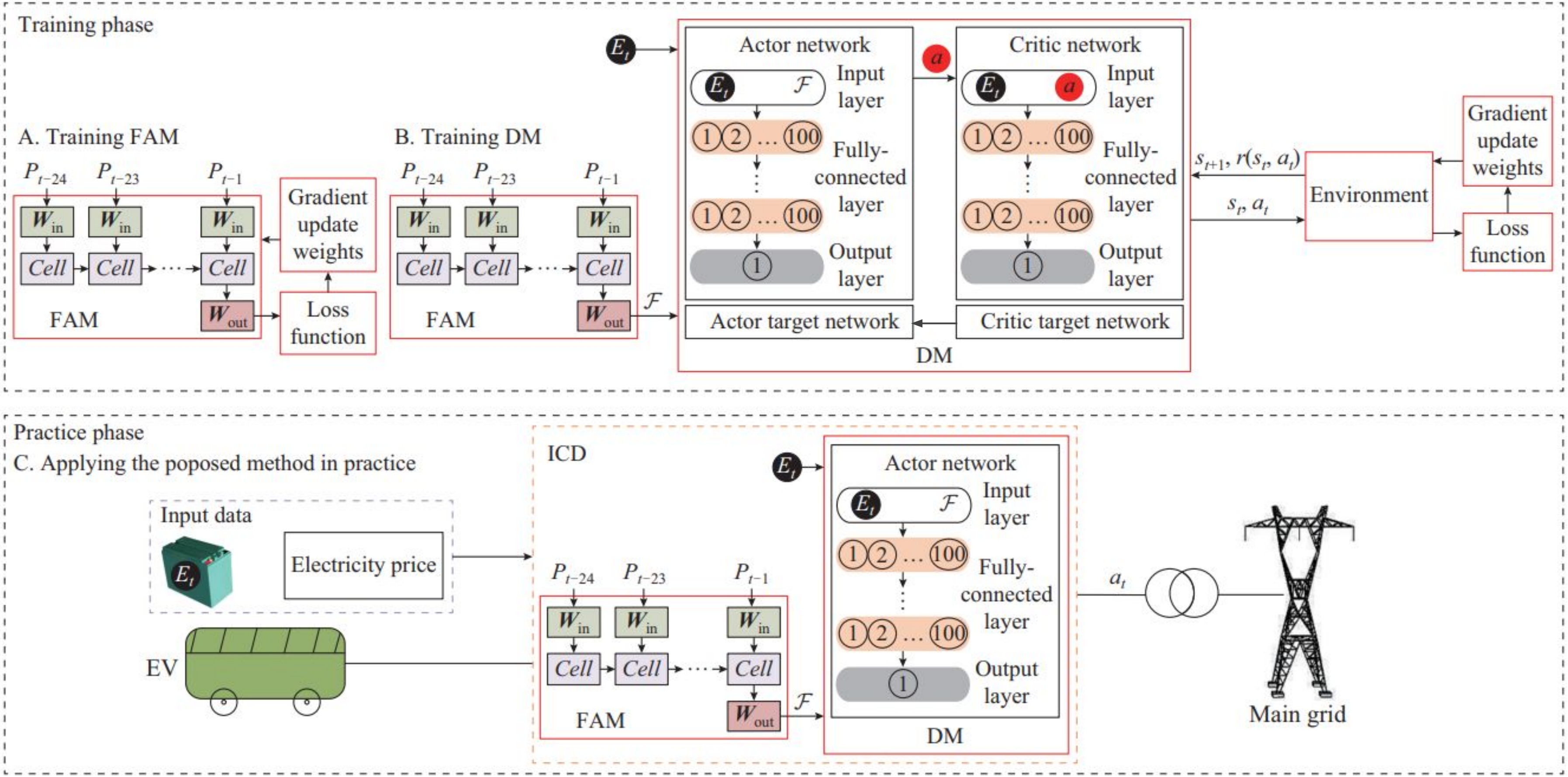
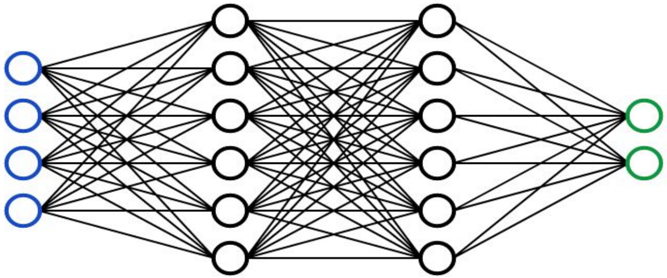


Fig. 3. Complete workflow of proposed method.

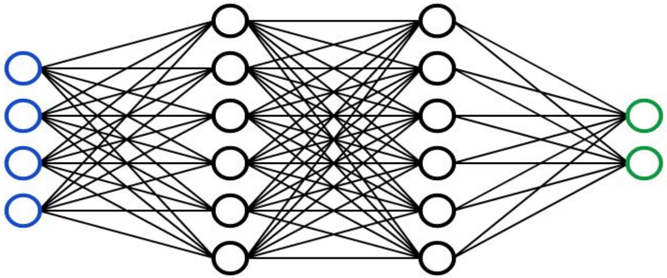
Implementation Highlights

Neural Network



```
class Net(nn.Module):
    def __init__(self, s_dim, a_dim):
        super(Net, self).__init__()
        self.s_dim = s_dim
        self.a_dim = a_dim
        self.pi1 = nn.Linear(s_dim, 128)
        self.pi2 = nn.Linear(128, a_dim)
        self.v1 = nn.Linear(s_dim, 128)
        self.v2 = nn.Linear(128, 1)
        set_init([self.pi1, self.pi2, self.v1, self.v2])
        self.distribution = torch.distributions.Categorical
```

Neural Network



```python
def forward(self, x):
    #print(x.shape)
    pi1 = torch.tanh(self.pi1(x))
    logits = self.pi2(pi1)
    v1 = torch.tanh(self.v1(x))
    values = self.v2(v1)
    return logits, values
```

Model Summary

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1               [-1, 1, 128]           1,152
            Linear-2                [-1, 1, 11]           1,419
            Linear-3               [-1, 1, 128]           1,152
            Linear-4                 [-1, 1, 1]             129
================================================================
Total params: 3,852
Trainable params: 3,852
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.01
Estimated Total Size (MB): 0.02
```

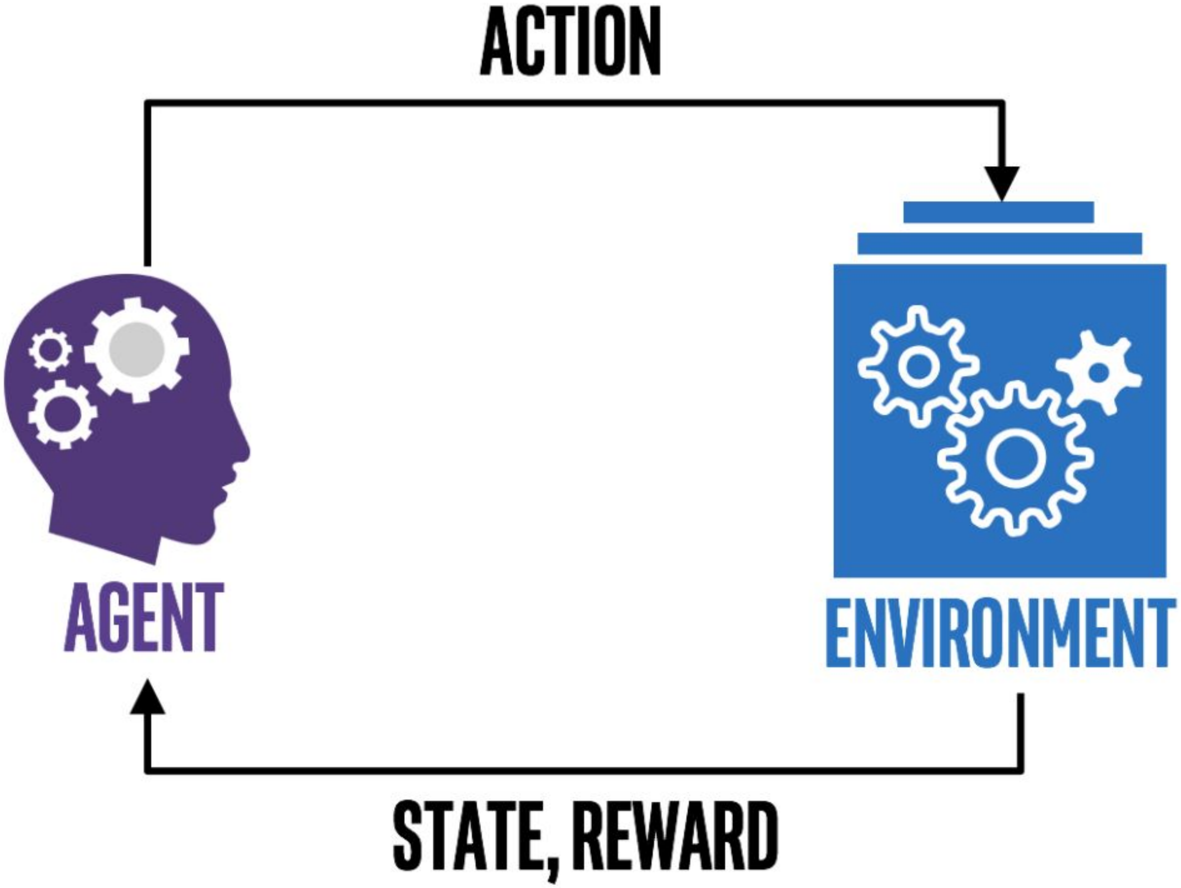
Loss Function

```
def loss_func(self, s, a, v_t):
    self.train()
    logits, values = self.forward(s)
    td = v_t - values
    c_loss = td.pow(2)

    #print(logits[:,:max_charging_rate])
    #print(logits[:,max_charging_rate:])

    prob1 = F.softmax(logits[:,:max_charging_rate], dim=-1).data
    prob2 = F.softmax(logits[:,max_charging_rate:], dim=-1).data
    m1 = self.distribution(prob1)
    m2=self.distribution(prob2)
    #print(a.shape)
    #print(a[:,0])
    #print(a[:,1])
    exp_v = m1.log_prob(a[:,0])*m2.log_prob(a[:,1])* td.detach().squeeze()
    a_loss = -exp_v
    total_loss = (c_loss + a_loss).mean()
    return total_loss
```

RL Network



RL Network

```
for t in range(MAX_EP_STEP):
    a = self.lnet.choose_action(s)
    r, real_state_, s_= self.env(a,real_state,t)
    r=np.expand_dims(np.expand_dims(r, 0), 0)
    s_=s_.reshape((1,N_S)).unsqueeze(0).float()
    ep_r += r
    buffer_a.append(np.array(a))
    buffer_s.append(s.squeeze().numpy())
    buffer_r.append(r.squeeze())
    done=False
    if t == MAX_EP_STEP - 1:
        done = True
    if total_step % UPDATE_GLOBAL_ITER == 0 or done:  # update global and assign to local net
        # sync
        push_and_pull(self.opt, self.lnet, self.gnet, done, s_, buffer_s, buffer_a, buffer_r, GAMMA)
        buffer_s, buffer_a, buffer_r = [], [], []

        if done:  # done and print information
            record(self.g_ep, self.g_ep_r, ep_r, self.res_queue, self.name)
            break
    s = s_
    real_state=real_state_
    total_step += 1
```

RL Network

```python
def env(action,residual_demand,iternum):
    if action[1]>residual_demand.shape[0]:
        action[1]=residual_demand.shape[0]

    ###Charging Station Start to Charge
    if residual_demand.shape[0]>0.5:
        #return reward,residual_demand,torch.tensor([0,0,0,0,0])
        least=residual_demand[:,1]-residual_demand[:,0]
        order=[operator.itemgetter(0)(t)-1 for t in sorted(enumerate(least,1), key=operator.itemgetter(1), reverse=True)]
        residual_demand[order[:action[1]],0]=residual_demand[order[:action[1]],0]-1

        residual_demand[:,1]=residual_demand[:,1]-1
    ###EV Admission
    reward=0
    for i in range(out1[iternum]):
        dem=beta1[0]*action[0]+beta2[0]
        if dem<0:
            dem=0
        reward+=dem*action[0]
        residual_demand=demand_update(residual_demand,np.array([dem,deadline[0]]).reshape((1,2)))
```

Execution Details :
Data & Results

Electricity Price Data (California ISO Dataset)

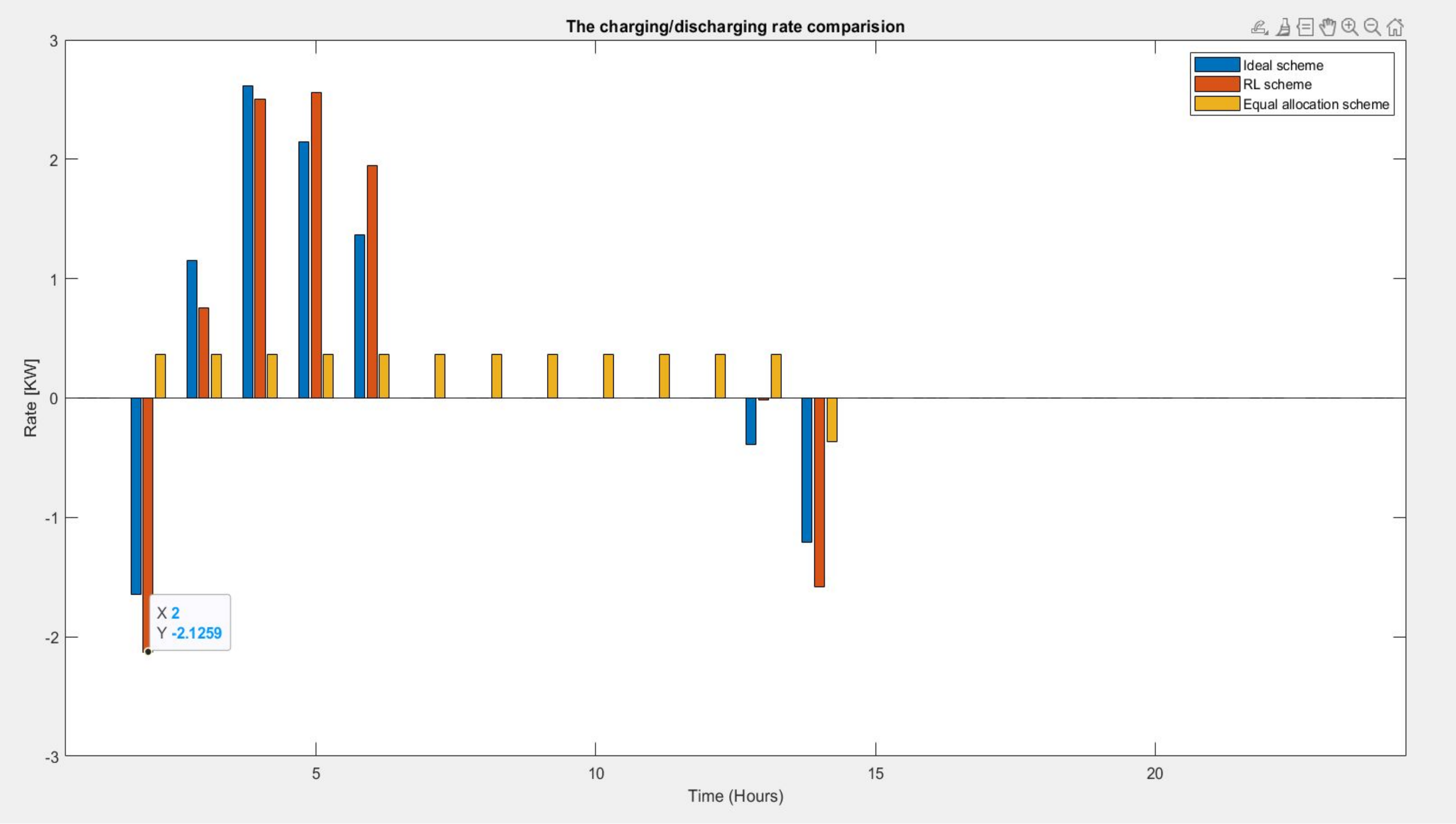
TRADE_DATE	TRADE_HOUR	LOAD_AGGREGATE_POINT	PRICE
01/01/2017	1	DLAP_PGAE-APND	40.92
01/01/2017	1	DLAP_SCE-APND	35.59
01/01/2017	1	DLAP_SDGE-APND	34.48
01/01/2017	1	DLAP_VEA-APND	32.35
01/01/2017	1	ELAP_AZPS-APND	28.71
01/01/2017	1	ELAP_NEVP-APND	29.70
01/01/2017	1	ELAP_PACE-APND	27.29
01/01/2017	1	ELAP_PACW-APND	17.43
01/01/2017	1	ELAP_PSEI-APND	17.41
01/01/2017	2	DLAP_PGAE-APND	48.22
01/01/2017	2	DLAP_SCE-APND	46.55
01/01/2017	2	DLAP_SDGE-APND	73.20
01/01/2017	2	DLAP_VEA-APND	50.10
01/01/2017	2	ELAP_AZPS-APND	21.28
01/01/2017	2	ELAP_NEVP-APND	33.67
01/01/2017	2	ELAP_PACE-APND	30.97
01/01/2017	2	ELAP_PACW-APND	17.52
01/01/2017	2	ELAP_PSEI-APND	18.09
01/01/2017	3	DLAP_PGAE-APND	25.14
01/01/2017	3	DLAP_SCE-APND	24.93
01/01/2017	3	DLAP_SDGE-APND	26.53
01/01/2017	3	DLAP_VEA-APND	30.45
01/01/2017	3	ELAP_AZPS-APND	15.60
01/01/2017	3	ELAP_NEVP-APND	18.99
01/01/2017	3	ELAP_PACE-APND	18.73
01/01/2017	3	ELAP_PACW-APND	14.22

EV data

EV info matrix:
1) arrival, 2) departure time, 3) initial energy, 4) charging period, 5) min charging time

2.0000000e+000	1.4000000e+001	1.0361574e+001	1.3000000e+001	2.0723149e+000
2.0000000e+000	1.4000000e+001	4.6783690e+000	1.3000000e+001	9.3567381e-001
1.1000000e+001	2.3000000e+001	8.9445108e+000	1.3000000e+001	1.7889022e+000
1.3000000e+001	2.4000000e+001	8.1078522e+000	1.2000000e+001	1.6215704e+000
1.8000000e+001	2.4000000e+001	1.2346613e+001	7.0000000e+000	2.4693225e+000
1.2000000e+001	2.4000000e+001	2.5701974e+000	1.3000000e+001	5.1403947e-001
8.0000000e+000	2.0000000e+001	7.6566751e+000	1.3000000e+001	1.5313350e+000
1.3000000e+001	2.4000000e+001	1.0714041e+000	1.2000000e+001	2.1428081e-001
1.0000000e+000	1.3000000e+001	6.9685083e+000	1.3000000e+001	1.3937017e+000
5.0000000e+000	1.7000000e+001	1.0830709e+001	1.3000000e+001	2.1661418e+000
1.5000000e+001	2.4000000e+001	5.2093861e-001	1.0000000e+001	1.0418772e-001
1.8000000e+001	2.4000000e+001	4.2158136e+000	7.0000000e+000	8.4316273e-001
1.1000000e+001	2.3000000e+001	9.0796757e+000	1.3000000e+001	1.8159351e+000
1.0000000e+000	1.3000000e+001	1.1011979e+001	1.3000000e+001	2.2023959e+000
1.0000000e+000	1.3000000e+001	6.4115023e+000	1.3000000e+001	1.2823005e+000
1.9000000e+001	2.4000000e+001	1.2732923e+001	6.0000000e+000	2.5465845e+000

Results



The charging/discharging rate comparision
Ideal scheme
RL scheme
Equal allocation scheme
Rate [KW]
3
2
1
0
-1
-2
-3
5
10
15
20
Time (Hours)
X 2
Y -2.1259

Future Scope and Deployment

- Can be deployed as a centralised distributed network and trained using Federated Learning for each individual EV
- Once the data flow becomes fast enough (seconds instead of hourly data), Loss Functions and Activation units can be optimised to give faster output by minimal compromise in accuracy.
- For Time-Series based prediction, LSTM based NNs are generally more effective so instead of FC NN, we can use LSTM.

References:

- Electric Vehicle Charging Management Based on Deep Reinforcement Learning (2021) - Sichen Li, Weihao Hu, Di Cao, Tomislav Dragičević, Qi Huang, Zhe Chen, and Frede Blaabjerg

- IoT Based Smart Parking System Using Deep Long Short Memory Network (2020) - Ghulam Ali 1 , Tariq Ali 2 , Muhammad Irfan 3 , Umar Draz 4 , Muhammad Sohail 1 , Adam Glowacz 5 , Maciej Sulowicz 6, Ryszard Mielnik 6 , Zaid Bin Faheem 7 and Claudia Martis

Thank You