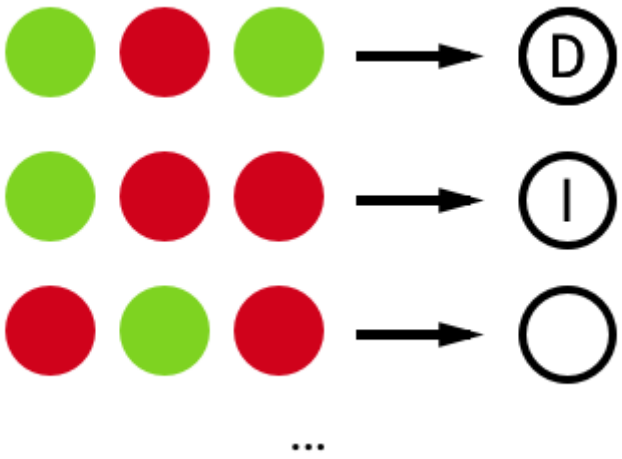
In [2]:

```
import torch
from torch import nn, optim
import numpy as np
import socket
```

NeuroCar

El problema a resolver?



...

In [3]:

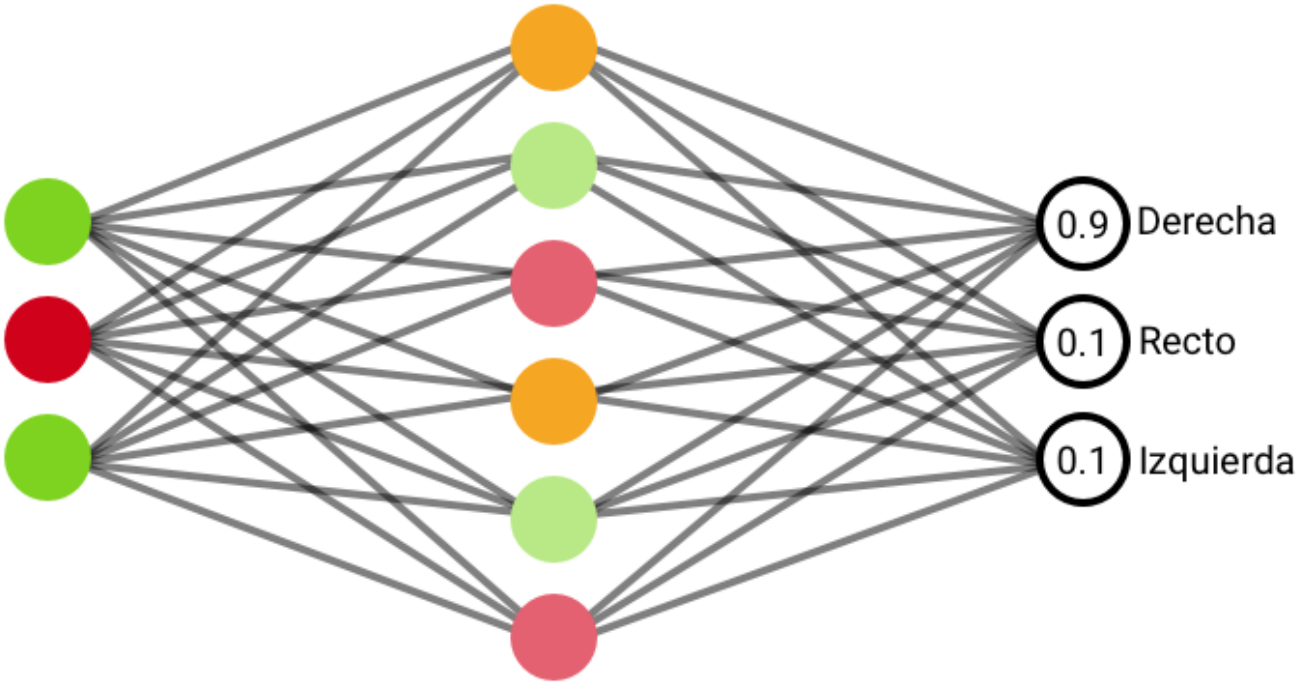
```
with open("NeuroCar/records_working.txt", "r") as file:
    records = file.readlines()

records = torch.tensor([list(map(int, r.split())) for r in records]).float()
records[:5]
```

Out[3]:

```
tensor([[0., 0., 0., 1.],
        [1., 0., 1., 0.],
        [0., 0., 1., 0.],
        [1., 0., 1., 0.],
        [0., 0., 1., 0.]])
```

Nuestra Red Neuronal Profunda



In [4]:

```
mlp = nn.Sequential(
    nn.Linear(3, 6),
    nn.ReLU(),
    nn.Linear(6, 3)
)
```

In [7]:

```
# mlp.load_state_dict(torch.load("model_working.pt")) #model_working.pt
```

Entrenamos nuestra red neuronal

In [ ]:

```
epochs = 10000

optimizer = optim.Adam(mlp.parameters(), lr=0.01)
criterion = nn.CrossEntropyLoss()

for e in range(1, epochs+1):
    x = records[:, :-1]
    y = records[:, -1].long() + 1
    pred = mlp.forward(records[:, :-1])
    loss = criterion(pred, y)

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

    if e % 1000 == 0:
        print(f"EPOCH {e} - loss {loss.item()}")
```

In [ ]:

```
torch.save(mlp.state_dict(), "model.pt")
```

Cómo de bien funciona?

In [8]:

```
mlp.eval()
with torch.no_grad():
    pred = mlp.forward(records[:, :-1])
    pred = pred.argmax(dim=-1) - 1
    acc = (pred == records[:, -1].long()).float().mean()
    print(f"Accuracy {100*acc:.2f}%")
    for s, e, p in list(zip(records[:, :-1], records[:, -1:], pred))[:50]:
        print(f"{s.int().tolist()} E: {int(e.item())}\t P: {p.item()} {'wrong' i
f p.item() != e.item() else ''}")
```

```
Accuracy 97.27%
[0, 0, 0] E: 1   P: 0 wrong
[1, 0, 1] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[1, 0, 1] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[1, 0, 1] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[0, 1, 1] E: -1  P: -1
[1, 0, 1] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[1, 1, 0] E: 1   P: 1
[1, 0, 1] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[0, 1, 1] E: -1  P: -1
[0, 0, 1] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[1, 0, 1] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[1, 0, 0] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[1, 0, 0] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[1, 0, 0] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[0, 1, 0] E: -1  P: -1
[1, 0, 1] E: 0   P: 0
[1, 0, 0] E: 0   P: 0
[1, 1, 0] E: 1   P: 1
[1, 0, 0] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[1, 0, 1] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[0, 1, 0] E: 1   P: -1 wrong
[1, 0, 1] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[1, 0, 1] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[0, 1, 1] E: -1  P: -1
[0, 0, 1] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[0, 0, 1] E: 0   P: 0
[0, 0, 0] E: 0   P: 0
[0, 1, 0] E: -1  P: -1
[1, 0, 1] E: 0   P: 0
```

Conectamos con el juego

In [9]:

```
HOST = ''   # Symbolic name, meaning all available interfaces
PORT = 5204 # Arbitrary non-privileged port

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print('Socket created')

try:
    s.bind((HOST, PORT))
except socket.error as msg:
    print('# Bind failed. ')

print('Socket bind complete')

s.listen(1)
print('Socket now listening')

conn, addr = s.accept()
print('# Connected to ' + addr[0] + ':' + str(addr[1]))

while True:
    data = conn.recv(1024)
    if not data: break
    line = data.decode('UTF-8')
    line = list(map(int, line.replace("\n","").split()))
    line = torch.tensor([line]).float()
    with torch.no_grad():
        pred = mlp.forward(line)[0].softmax(-1)
        if pred.max() > 0.8:
            pred = (pred.argmax(dim=-1) - 1).item()
        else:
            pred = (torch.multinomial(pred / 1.2, 1) - 1).item()
    conn.sendall(str(pred).encode('utf-8'))
    # print(line[0].tolist(), ">>" ,pred)

conn.close()
print("Connection closed")
```

```
Socket created
Socket bind complete
Socket now listening
# Connected to 127.0.0.1:56643
Connection closed
```

In [ ]:

```
conn.close()
```

In [ ]:

```

```