pofc2

September 19, 2024

0.1 Can mlp learn to classifier clean modulated signals just as well as convnet?

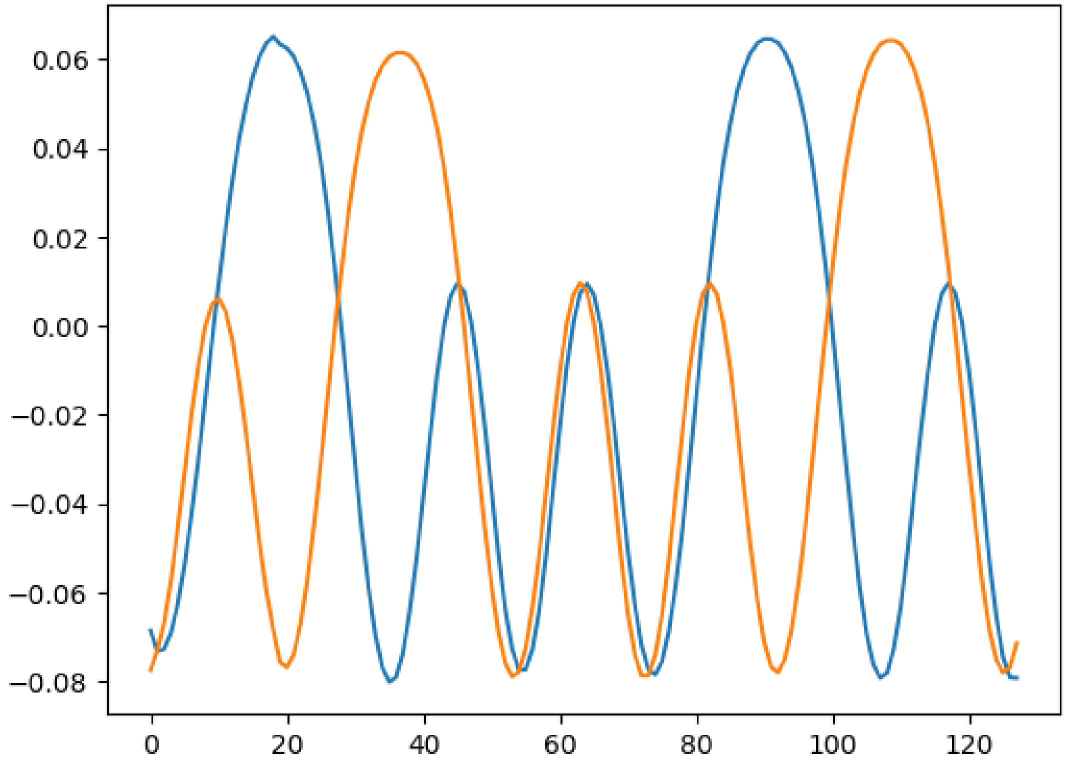
[27]:
```
import pickle
import matplotlib.pyplot as plt
```

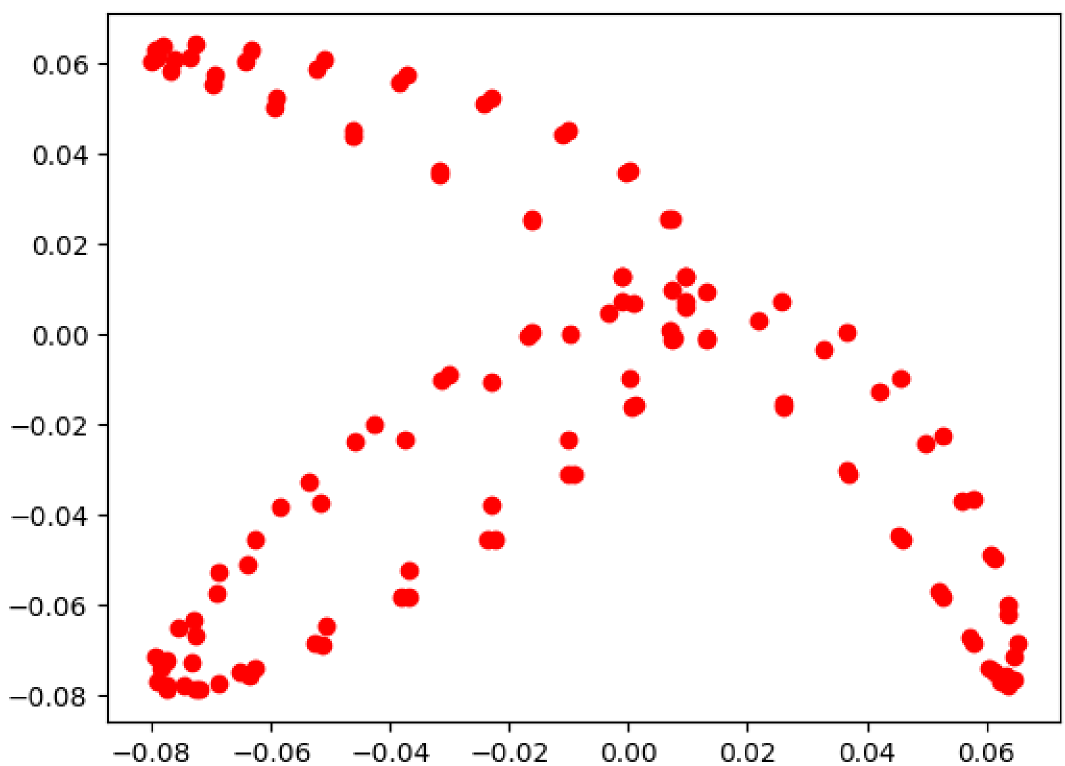
[28]:
```
with open('./Data/Unimpaired_diff_BW_mod22.pickle', 'rb') as f:
    data = pickle.load(f, encoding='latin1')

print(data.keys())
```

```
dict_keys(['oqpsk', 'gmsk', '64apsk', '2fsk', 'ook', '128apsk', '4fsk', '8psk',
'16psk', 'bpsk', 'qam16', 'cpfsk', 'qam256', '256apsk', '16apsk', 'pam4',
'32psk', 'gfsk', 'qam64', 'qpsk', '32apsk', 'qam32'])
```

[29]:
```
oqpsk = data['oqpsk']
plt.figure(1)
plt.plot(oqpsk[0].T)
plt.figure(2)
plt.scatter(oqpsk[0][0], oqpsk[0][1], c='r')
```

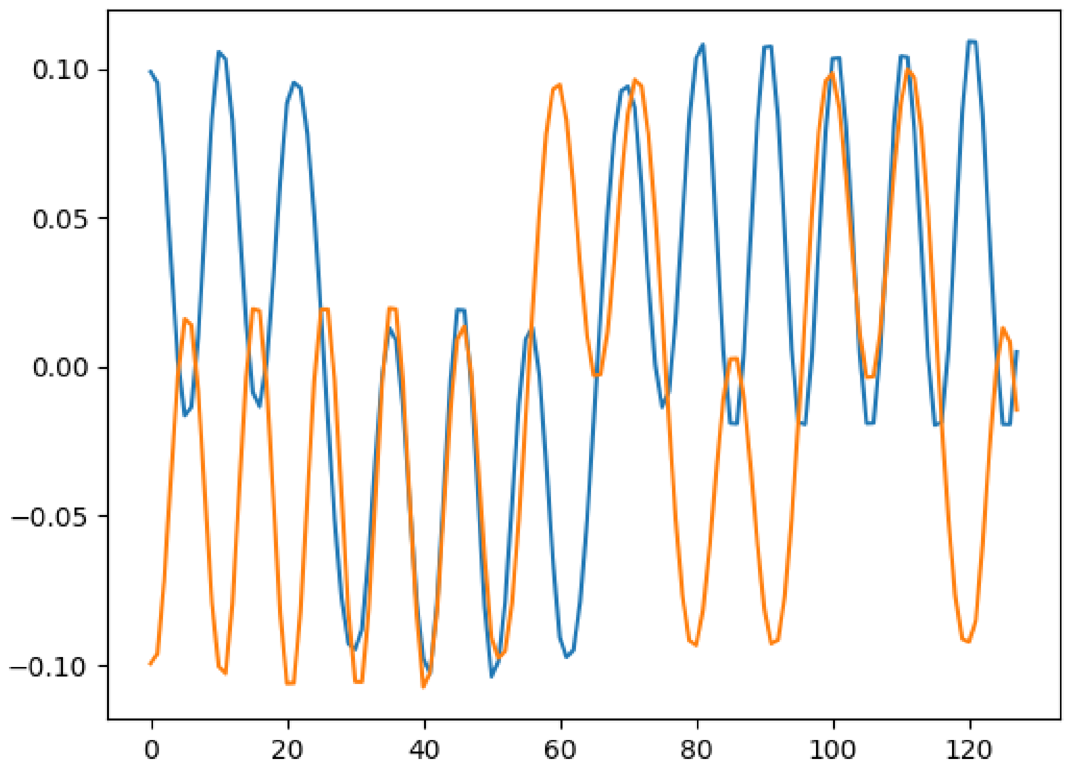
[29]:
```
<matplotlib.collections.PathCollection at 0x7ad99eb20a60>
```

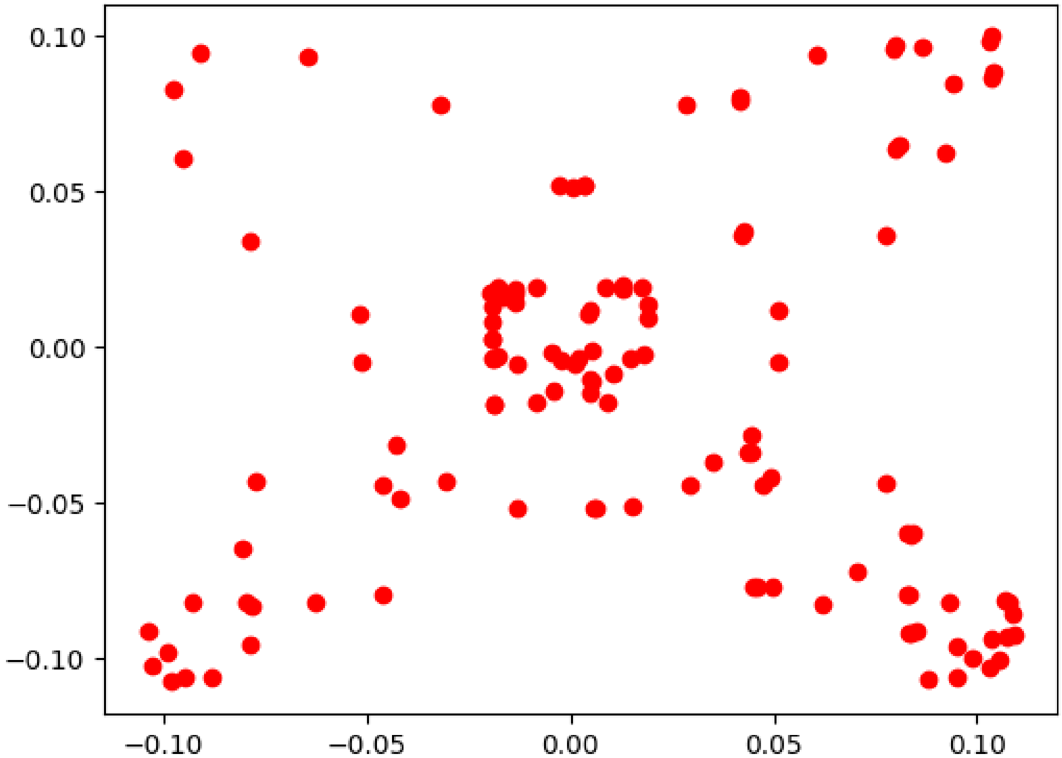

0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
20
40
60
80
100
120


0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.08
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06

[30]:
```
gmsk = data['gmsk']
plt.figure(1)
plt.plot(gmsk[0].T)
plt.figure(2)
plt.scatter(gmsk[0][0], gmsk[0][1], c='r')
```

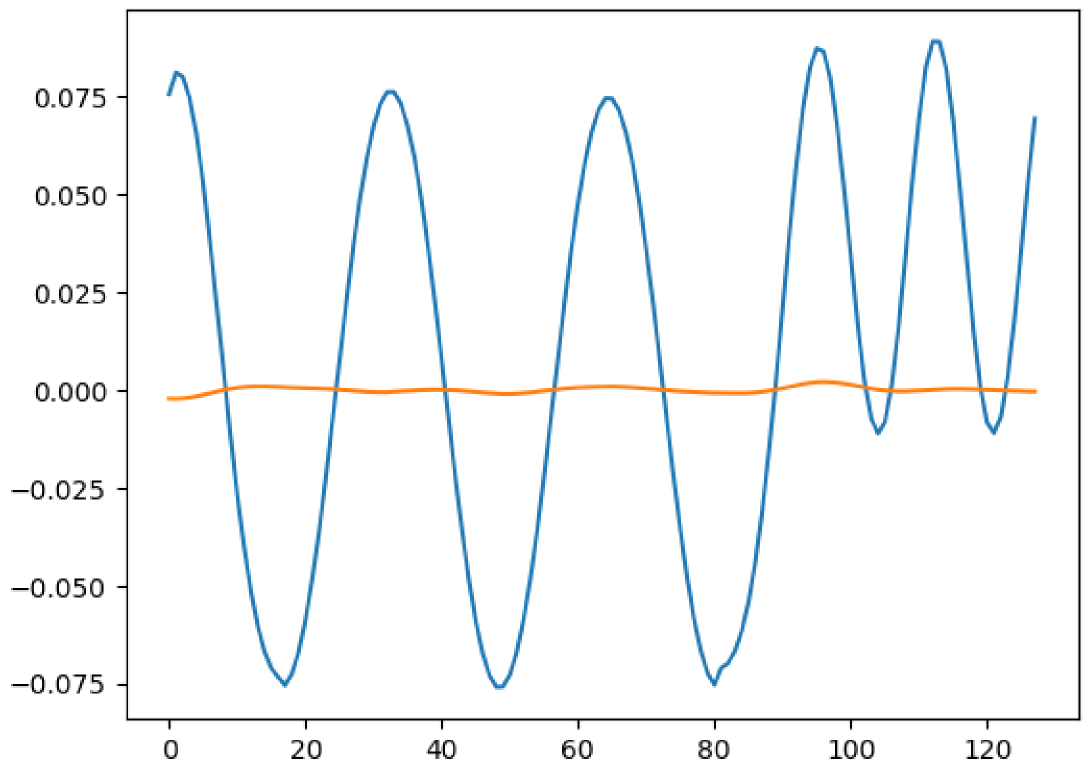
[30]: <matplotlib.collections.PathCollection at 0x7ad9a57c33a0>

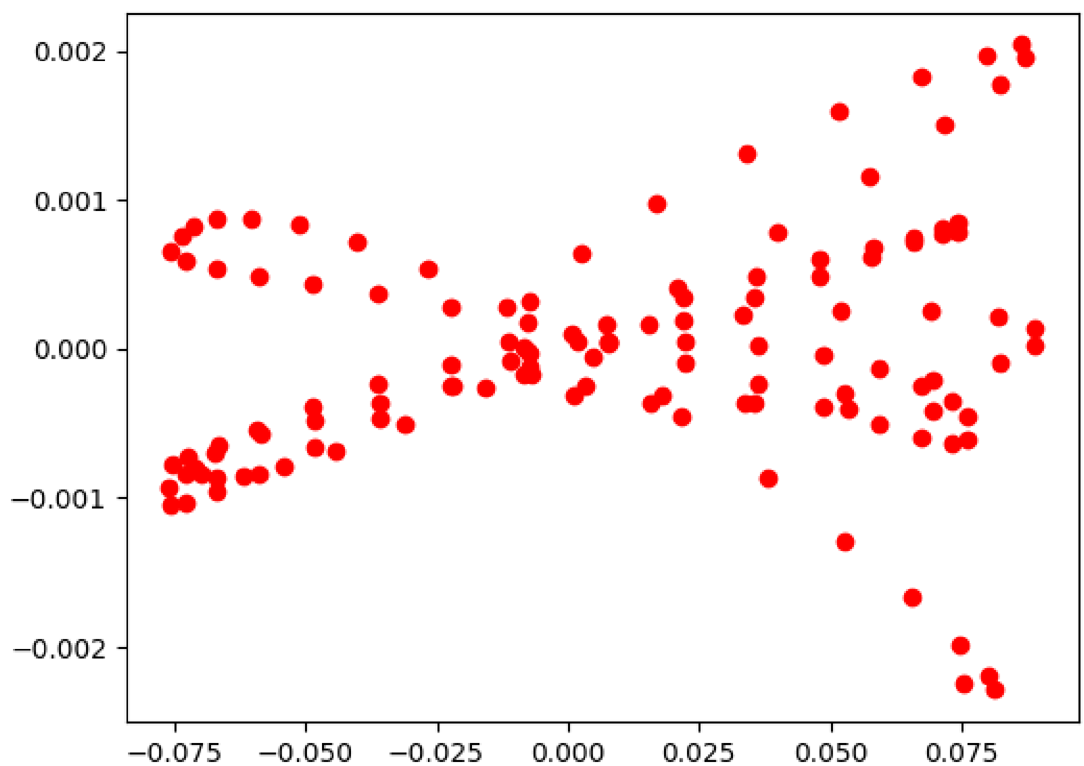




[31]:
```
bpsk = data['bpsk']
plt.figure(1)
plt.plot(bpsk[0].T)
plt.figure(2)
plt.scatter(bpsk[0][0], bpsk[0][1], c='r')
```

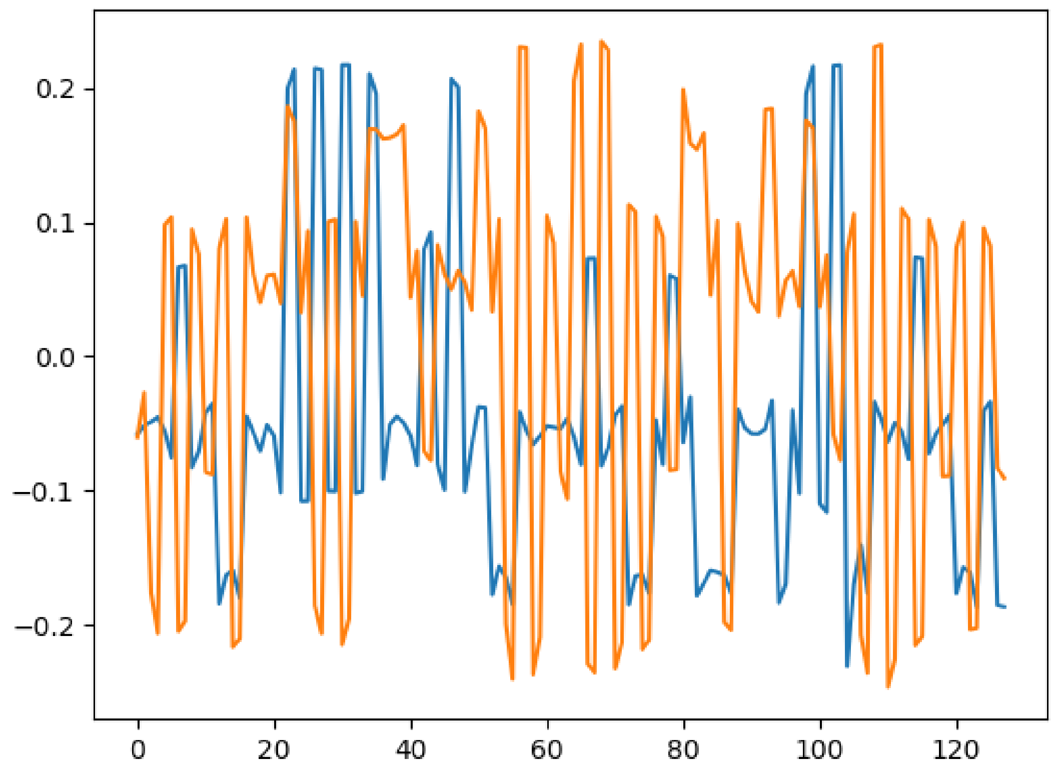
[31]: <matplotlib.collections.PathCollection at 0x7ad9a58d4a60>

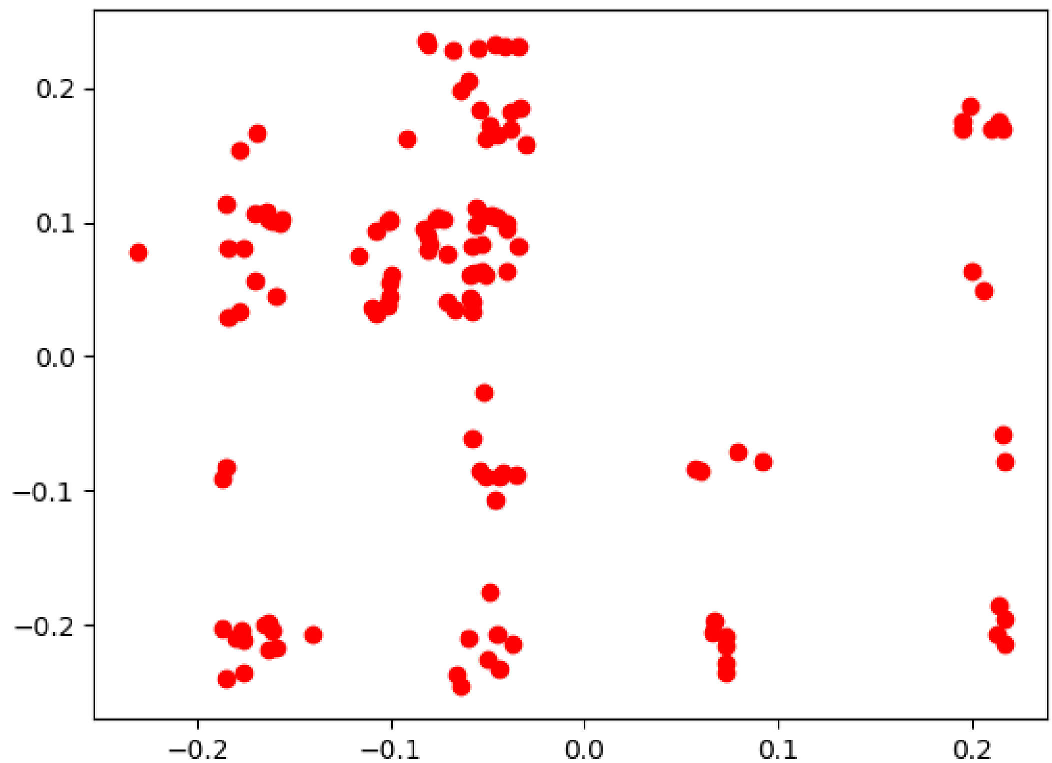

0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
0
20
40
60
80
100
120


0.002
0.001
0.000
−0.001
−0.002
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075

[32]:
```
qam16 = data['qam16']
plt.figure(1)
plt.plot(qam16[0].T)
plt.figure(2)
plt.scatter(qam16[0][0], qam16[0][1], c='r')
```

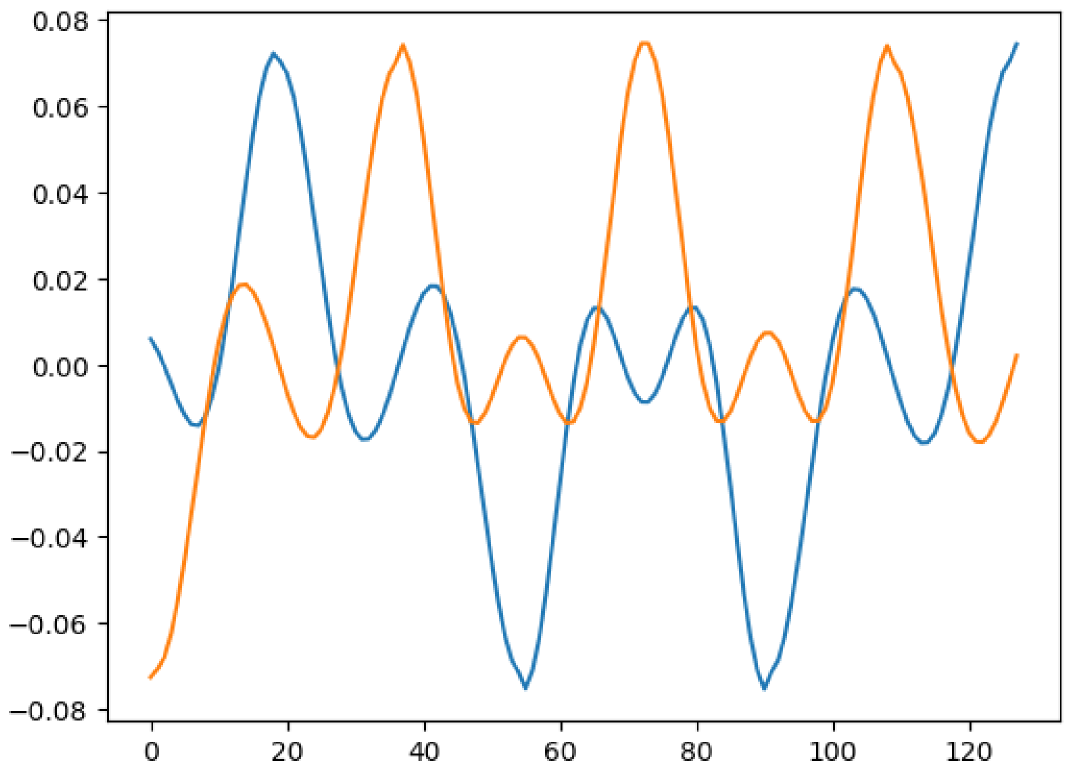
[32]: <matplotlib.collections.PathCollection at 0x7ad9a56e4220>

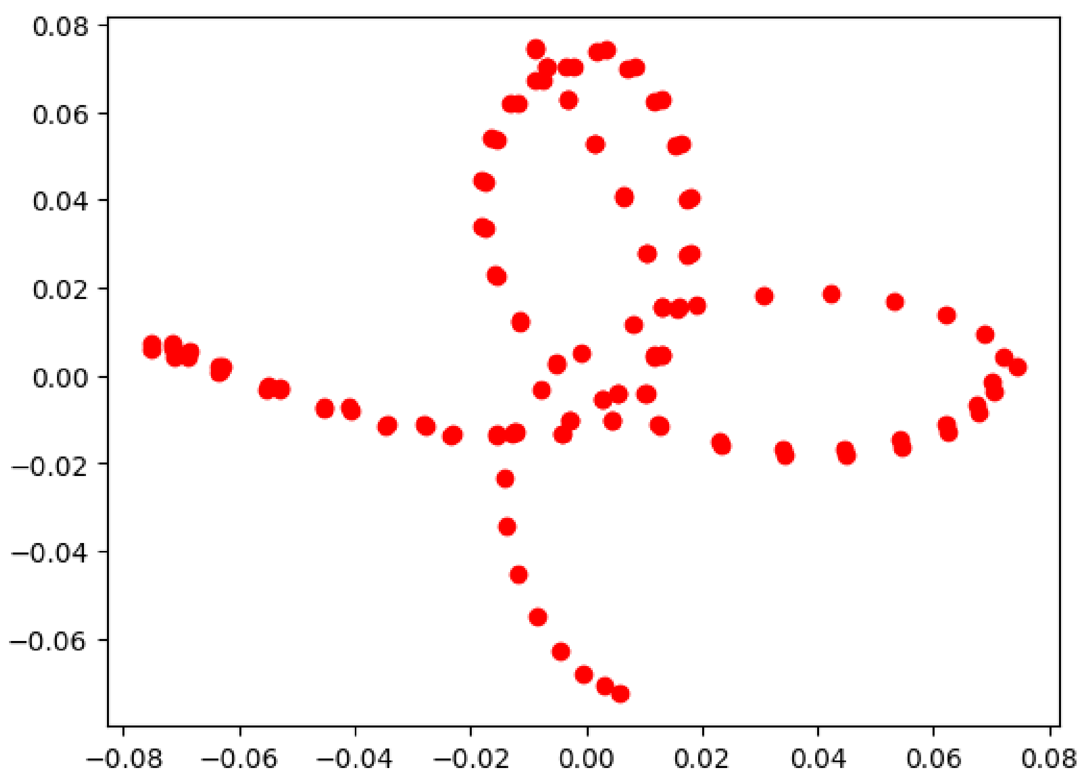




[33]:
```
fsk4 = data['4fsk']
plt.figure(1)
plt.plot(fsk4[0].T)
plt.figure(2)
plt.scatter(fsk4[0][0], fsk4[0][1], c='r')
```

[33]: <matplotlib.collections.PathCollection at 0x7ad99af94130>


0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
20
40
60
80
100
120


0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08

[34]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader, random_split

from torchvision.transforms import Compose, Lambda

import numpy as np
from sklearn.model_selection import StratifiedShuffleSplit
```

[35]:
```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

[36]:
```
class IQDataset(Dataset):
    def __init__(self, data, labels, data_transform=None):
        self.data = data
        self.labels = labels
        self.data_transform = data_transform

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        data = self.data[idx]
        labels = self.labels[idx]
        if self.data_transform:
            data = self.data_transform(data)
        elif isinstance(data, np.ndarray):
            data = torch.from_numpy(data).float()
        return data, labels
```

[37]:
```
labels = data.keys()
X = np.concatenate([data[label] for label in labels])
y = np.concatenate([np.ones(data[label].shape[0]) *
                    i for i, label in enumerate(labels)])
X.shape, y.shape
```

[37]:
```
((219912, 2, 128), (219912,))
```

[38]:
```
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=0)
train_idx, test_idx = next(sss.split(X, y))
```

[39]:
```
MLP_txfm = Compose([
    Lambda(lambda x: torch.tensor(x).float()),
    Lambda(lambda x: x.view(-1))
])
```

```
dataset = IQDataset(X[train_idx], y[train_idx], data_transform=MLP_txfm)

train_dataset, val_dataset = random_split(
    dataset, [int(0.8 * len(dataset)), len(dataset) - int(0.8 * len(dataset))])

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)
```

[40]:
```
class MLP(nn.Module):
    def __init__(self, in_features, hidden_features, out_features):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(in_features, hidden_features)
        self.fc2 = nn.Linear(hidden_features, hidden_features)
        self.fc3 = nn.Linear(hidden_features, out_features)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

[41]:
```
class ConvNet(nn.Module):
    def __init__(self, in_channels, num_classes):
        super(ConvNet, self).__init__()
        self.conv1 = nn.Conv1d(in_channels, 16, 5, padding='same')
        self.conv2 = nn.Conv1d(16, 32, 5, padding='same')
        self.maxpool = nn.MaxPool1d(2)  # 2x downsampling
        self.fc1 = nn.Linear(1024, num_classes)  # 128 / 4 * 32 = 1024

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.maxpool(x)
        x = F.relu(self.conv2(x))
        x = self.maxpool(x)
        x = x.view(x.size(0), -1)
        x = self.fc1(x)
        return x
```

[42]:
```
def train_one_epoch(model, optimizer, criterion, train_loader):
    model.train()
    avg_loss = 0.
    for data, labels in train_loader:
        data, labels = data.to(device), labels.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, labels.long())
```

```python
        loss.backward()
        optimizer.step()
        avg_loss += loss.item()
    return avg_loss / len(train_loader)


def validate(model, criterion, val_loader):
    model.eval()
    avg_loss = 0.
    correct = 0
    with torch.no_grad():
        for data, labels in val_loader:
            data, labels = data.to(device), labels.to(device)
            output = model(data)
            avg_loss += criterion(output, labels.long()).item()
            pred = F.softmax(output, dim=-1).argmax(dim=1, keepdim=True)
            correct += pred.eq(labels.view_as(pred)).sum().item()
    return avg_loss / len(val_loader), correct / len(val_loader.dataset)
```

[43]:
```python
class Tracker:
    def __init__(self, metric, mode='auto'):
        self.metric = metric
        self.mode = mode
        self.mode_dict = {
            'auto': np.less if 'loss' in metric else np.greater,
            'min': np.less,
            'max': np.greater
        }
        self.operator = self.mode_dict[mode]

        self._best = np.inf if 'loss' in metric else -np.inf

    @property
    def best(self):
        return self._best

    @best.setter
    def best(self, value):
        self._best = value
```

[44]:
```python
NUM_EPOCHS = 50

model = MLP(in_features=2*128, hidden_features=256,
            out_features=len(labels)).to(device)
optimizer = torch.optim.Adam(
    model.parameters(), lr=1e-3, weight_decay=5e-4)  # L2 regularization
criterion = nn.CrossEntropyLoss()
```

```python
tracker = Tracker('val_loss')

print(f'Model has {sum(p.numel() for p in model.parameters())} parameters')


history = {
    'train_loss': [],
    'val_loss': [],
    'val_acc': []
}
for epoch in range(NUM_EPOCHS):
    train_loss = train_one_epoch(model, optimizer, criterion, train_loader)
    val_loss, val_acc = validate(model, criterion, val_loader)
    print(
        f'Epoch {epoch}, Train Loss: {train_loss}, Val Loss: {val_loss}, Val␣
 ↪Acc: {val_acc}')

    history['train_loss'].append(train_loss)
    history['val_loss'].append(val_loss)
    history['val_acc'].append(val_acc)

    if tracker.operator(val_loss, tracker.best):
        tracker.best = val_loss
        torch.save(model.state_dict(), './Models/best_mlp.pth')
        print('Model saved with val loss:',
              val_loss, 'at ./Models/best_mlp.pth')
```

```
Model has 137238 parameters
Epoch 0, Train Loss: 1.5374527312031223, Val Loss: 1.174192016741197, Val Acc:
0.605073405222814
Model saved with val loss: 1.174192016741197 at ./Models/best_mlp.pth
Epoch 1, Train Loss: 1.0565946272408135, Val Loss: 0.9838634078871548, Val Acc:
0.6595426789658309
Model saved with val loss: 0.9838634078871548 at ./Models/best_mlp.pth
Epoch 2, Train Loss: 0.9144472695536724, Val Loss: 0.8978868533827187, Val Acc:
0.6837404183448097
Model saved with val loss: 0.8978868533827187 at ./Models/best_mlp.pth
Epoch 3, Train Loss: 0.8443075458348092, Val Loss: 0.8517410799102744, Val Acc:
0.6997531505781474
Model saved with val loss: 0.8517410799102744 at ./Models/best_mlp.pth
Epoch 4, Train Loss: 0.8018891648925351, Val Loss: 0.8265247595953916, Val Acc:
0.7076133558529297
Model saved with val loss: 0.8265247595953916 at ./Models/best_mlp.pth
Epoch 5, Train Loss: 0.7734160682974806, Val Loss: 0.8191302797195201, Val Acc:
0.7081005586592178
Model saved with val loss: 0.8191302797195201 at ./Models/best_mlp.pth
Epoch 6, Train Loss: 0.749570601875053, Val Loss: 0.7864358304445618, Val Acc:
```

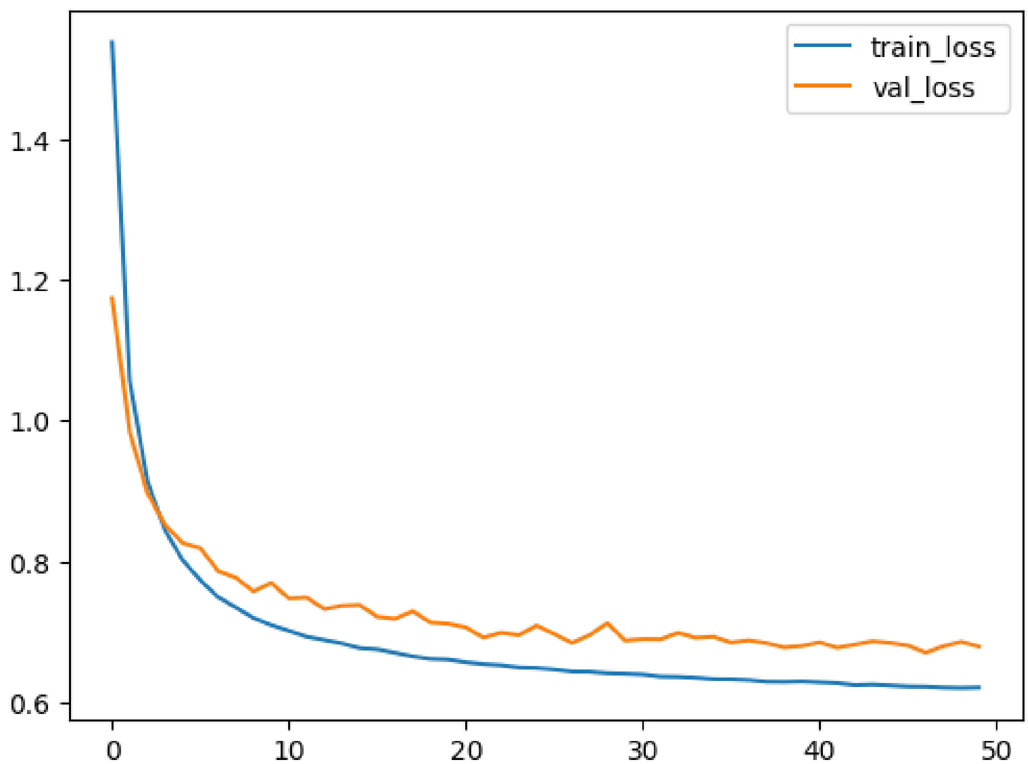
```
0.7182343770300117
Model saved with val loss: 0.7864358304445618 at ./Models/best_mlp.pth
Epoch 7, Train Loss: 0.7348068544433531, Val Loss: 0.7771800289458575, Val Acc:
0.7250876965051318
Model saved with val loss: 0.7771800289458575 at ./Models/best_mlp.pth
Epoch 8, Train Loss: 0.7200027033628567, Val Loss: 0.7579432146880981, Val Acc:
0.7314862933610498
Model saved with val loss: 0.7579432146880981 at ./Models/best_mlp.pth
Epoch 9, Train Loss: 0.7098306185024129, Val Loss: 0.7698805538602831, Val Acc:
0.721612316486943
Epoch 10, Train Loss: 0.7017935700433909, Val Loss: 0.7478195880370472, Val Acc:
0.7324606989736261
Model saved with val loss: 0.7478195880370472 at ./Models/best_mlp.pth
Epoch 11, Train Loss: 0.6933927211042317, Val Loss: 0.7490214257503967, Val Acc:
0.7349941535663246
Epoch 12, Train Loss: 0.688711042031278, Val Loss: 0.7326348251890418, Val Acc:
0.740158503312979
Model saved with val loss: 0.7326348251890418 at ./Models/best_mlp.pth
Epoch 13, Train Loss: 0.6838089217842942, Val Loss: 0.7372655619713376, Val Acc:
0.7328829414057425
Epoch 14, Train Loss: 0.6771660595203134, Val Loss: 0.7382753528229049, Val Acc:
0.7376575289073665
Epoch 15, Train Loss: 0.675403771593404, Val Loss: 0.7216245077728358, Val Acc:
0.7433090814603092
Model saved with val loss: 0.7216245077728358 at ./Models/best_mlp.pth
Epoch 16, Train Loss: 0.6704666859400864, Val Loss: 0.7189368535363166, Val Acc:
0.744153566324542
Model saved with val loss: 0.7189368535363166 at ./Models/best_mlp.pth
Epoch 17, Train Loss: 0.6654367825061385, Val Loss: 0.7294945531328395, Val Acc:
0.7342795894504353
Epoch 18, Train Loss: 0.6619099988259721, Val Loss: 0.7136381606756094, Val Acc:
0.7425295569702481
Model saved with val loss: 0.7136381606756094 at ./Models/best_mlp.pth
Epoch 19, Train Loss: 0.6609841239788403, Val Loss: 0.7117987552257342, Val Acc:
0.7461023775496947
Model saved with val loss: 0.7117987552257342 at ./Models/best_mlp.pth
Epoch 20, Train Loss: 0.6570521717250858, Val Loss: 0.7065884137636405, Val Acc:
0.7463946992334676
Model saved with val loss: 0.7065884137636405 at ./Models/best_mlp.pth
Epoch 21, Train Loss: 0.654396837761601, Val Loss: 0.6921094492897933, Val Acc:
0.751039365986748
Model saved with val loss: 0.6921094492897933 at ./Models/best_mlp.pth
Epoch 22, Train Loss: 0.6528296658484215, Val Loss: 0.6991421507897778, Val Acc:
0.7500324801870859
Epoch 23, Train Loss: 0.6496664499634116, Val Loss: 0.695703366154327, Val Acc:
0.7506496037417175
Epoch 24, Train Loss: 0.6488743516788076, Val Loss: 0.7090873296448994, Val Acc:
0.7465246199818111
```

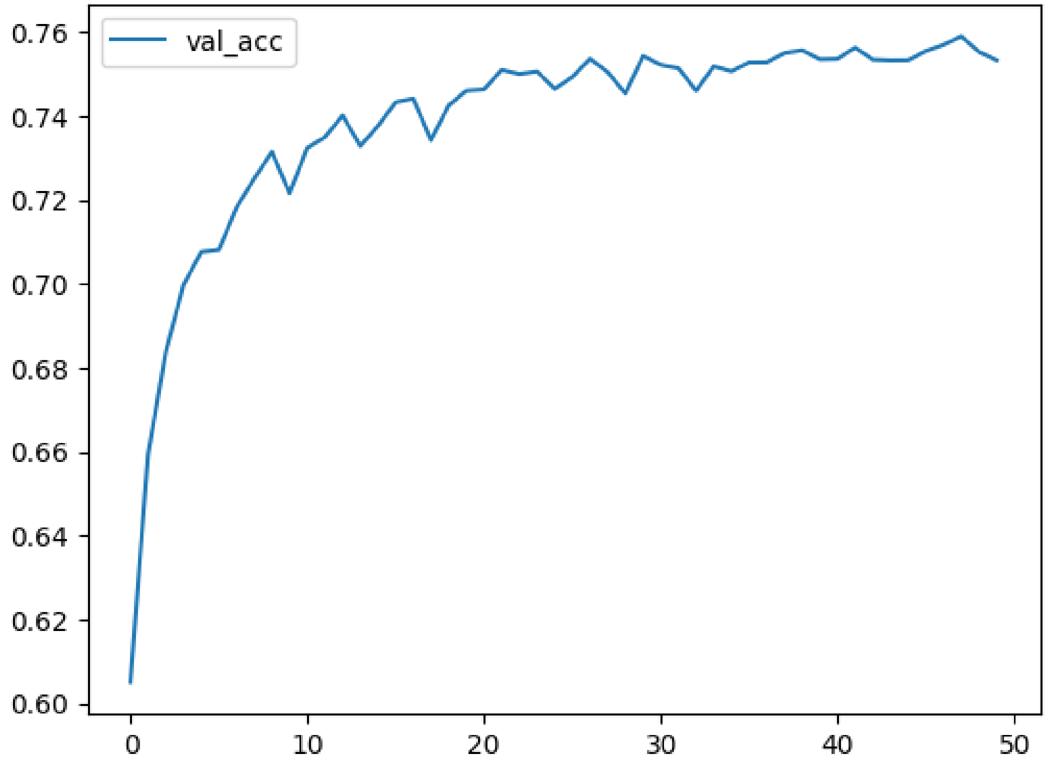
```
Epoch 25, Train Loss: 0.6469121919913426, Val Loss: 0.6973380799853046, Val Acc:
0.7493503962582825
Epoch 26, Train Loss: 0.6439503157106304, Val Loss: 0.6845766814698312, Val Acc:
0.75370274132779
Model saved with val loss: 0.6845766814698312 at ./Models/best_mlp.pth
Epoch 27, Train Loss: 0.6435700504965335, Val Loss: 0.6956812850473331, Val Acc:
0.7504872028062881
Epoch 28, Train Loss: 0.6414367669966352, Val Loss: 0.712629210726123, Val Acc:
0.7454202936208912
Epoch 29, Train Loss: 0.6407441921700562, Val Loss: 0.6880066721610191, Val Acc:
0.7543523450695075
Epoch 30, Train Loss: 0.6397259319043834, Val Loss: 0.6899142217660866, Val Acc:
0.7522411329089256
Epoch 31, Train Loss: 0.6367497627465376, Val Loss: 0.6896868752294364, Val Acc:
0.7514616084188644
Epoch 32, Train Loss: 0.6363536527897669, Val Loss: 0.6989277293625156, Val Acc:
0.7460698973626088
Epoch 33, Train Loss: 0.6350606190566739, Val Loss: 0.6919524150969207, Val Acc:
0.7518838508509809
Epoch 34, Train Loss: 0.6331031938928169, Val Loss: 0.6933719735217366, Val Acc:
0.7507145641158893
Epoch 35, Train Loss: 0.6328044036869322, Val Loss: 0.685052427779477, Val Acc:
0.7528257762764714
Epoch 36, Train Loss: 0.6317589022248217, Val Loss: 0.6881586862254118, Val Acc:
0.7528582564635572
Epoch 37, Train Loss: 0.6295262773947828, Val Loss: 0.6842190963085567, Val Acc:
0.7550344289983111
Model saved with val loss: 0.6842190963085567 at ./Models/best_mlp.pth
Epoch 38, Train Loss: 0.6292281368005489, Val Loss: 0.6781534688433135, Val Acc:
0.7556840327400286
Model saved with val loss: 0.6781534688433135 at ./Models/best_mlp.pth
Epoch 39, Train Loss: 0.629854089348432, Val Loss: 0.6802175554407225, Val Acc:
0.7535728205794465
Epoch 40, Train Loss: 0.6286289407529283, Val Loss: 0.6853073626478142, Val Acc:
0.7536702611407041
Epoch 41, Train Loss: 0.6274779112842617, Val Loss: 0.6777665680380749, Val Acc:
0.7563336364817461
Model saved with val loss: 0.6777665680380749 at ./Models/best_mlp.pth
Epoch 42, Train Loss: 0.6245518333534353, Val Loss: 0.682236362871475, Val Acc:
0.7534753800181889
Epoch 43, Train Loss: 0.6253285195710412, Val Loss: 0.6868558871721305, Val Acc:
0.7533129790827595
Epoch 44, Train Loss: 0.6240996338928418, Val Loss: 0.6844908537523026, Val Acc:
0.7533454592698454
Epoch 45, Train Loss: 0.6226349736644435, Val Loss: 0.6808468983552166, Val Acc:
0.7555216318045992
Epoch 46, Train Loss: 0.6223895612839879, Val Loss: 0.6706797781701153, Val Acc:
0.7570157204105495
```

```
Model saved with val loss: 0.6706797781701153 at ./Models/best_mlp.pth
Epoch 47, Train Loss: 0.6211793030483688, Val Loss: 0.6798021647603341, Val Acc:
0.759029492009874
Epoch 48, Train Loss: 0.6206433982020944, Val Loss: 0.6860780209488586, Val Acc:
0.7553267506820839
Epoch 49, Train Loss: 0.6212244527189413, Val Loss: 0.6794575133891864, Val Acc:
0.7533454592698454
```

[45]:
```
plt.figure(1)
plt.plot(history['train_loss'], label='train_loss')
plt.plot(history['val_loss'], label='val_loss')
plt.legend()

plt.figure(2)
plt.plot(history['val_acc'], label='val_acc')
plt.legend()
```

[45]:
```
<matplotlib.legend.Legend at 0x7ad9c7dcb430>
```





[46]:
```
dataset = IQDataset(X[train_idx], y[train_idx])

train_dataset, val_dataset = random_split(
    dataset, [int(0.8 * len(dataset)), len(dataset) - int(0.8 * len(dataset))])

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)
```

[47]:
```
model = ConvNet(in_channels=2, num_classes=len(labels)).to(device)
optimizer = torch.optim.Adam(
    model.parameters(), lr=1e-3, weight_decay=5e-4)  # L2 regularization
criterion = nn.CrossEntropyLoss()
tracker = Tracker('val_loss')

print(f'Model has {sum(p.numel() for p in model.parameters())} parameters')


history = {
    'train_loss': [],
    'val_loss': [],
    'val_acc': []
}
```

```
for epoch in range(NUM_EPOCHS):
    train_loss = train_one_epoch(model, optimizer, criterion, train_loader)
    val_loss, val_acc = validate(model, criterion, val_loader)
    print(
        f'Epoch {epoch}, Train Loss: {train_loss}, Val Loss: {val_loss}, Val␣
 ↪Acc: {val_acc}')

    history['train_loss'].append(train_loss)
    history['val_loss'].append(val_loss)
    history['val_acc'].append(val_acc)

    if tracker.operator(val_loss, tracker.best):
        tracker.best = val_loss
        torch.save(model.state_dict(), './Models/best_convnet.pth')
        print('Model saved with val loss:',
              val_loss, 'at ./Models/best_convnet.pth')
```

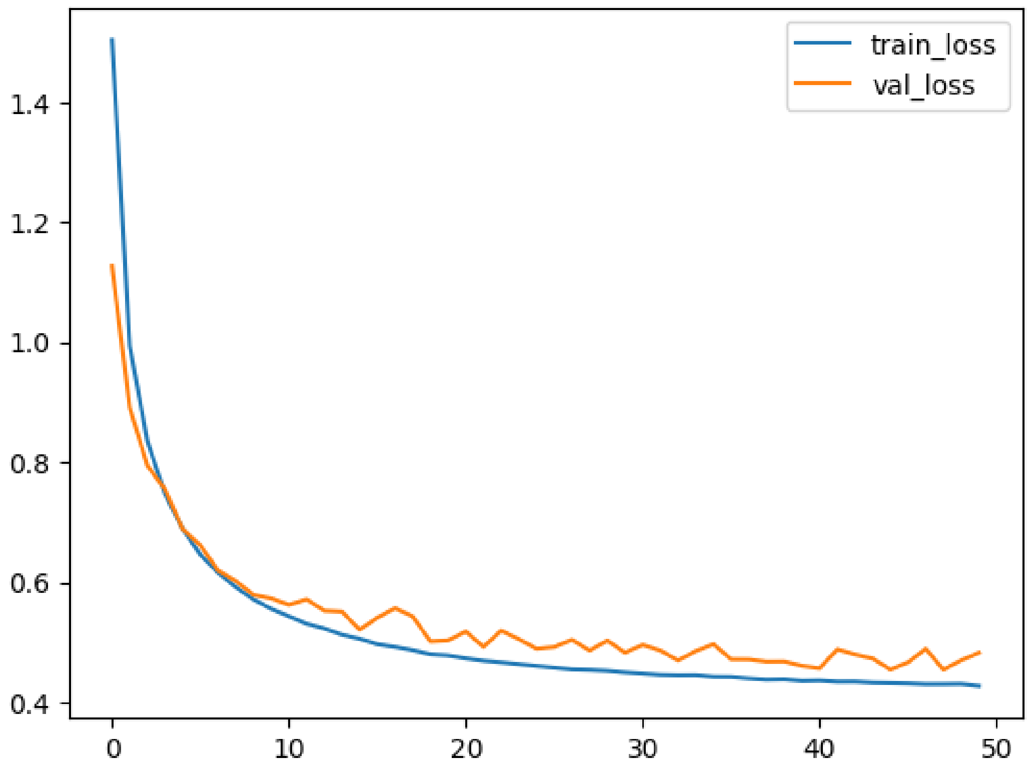
```
Model has 25318 parameters
Epoch 0, Train Loss: 1.5035233854101553, Val Loss: 1.1265492247396294, Val Acc:
0.6386579186696115
Model saved with val loss: 1.1265492247396294 at ./Models/best_convnet.pth
Epoch 1, Train Loss: 0.9939782235042619, Val Loss: 0.8908661828915031, Val Acc:
0.7023840457321034
Model saved with val loss: 0.8908661828915031 at ./Models/best_convnet.pth
Epoch 2, Train Loss: 0.8353282560672535, Val Loss: 0.7940981756984754, Val Acc:
0.7276536312849162
Model saved with val loss: 0.7940981756984754 at ./Models/best_convnet.pth
Epoch 3, Train Loss: 0.7462510570949565, Val Loss: 0.7543433858226765, Val Acc:
0.7408730674288684
Model saved with val loss: 0.7543433858226765 at ./Models/best_convnet.pth
Epoch 4, Train Loss: 0.6884977312506437, Val Loss: 0.6883596751920159, Val Acc:
0.7612056645446278
Model saved with val loss: 0.6883596751920159 at ./Models/best_convnet.pth
Epoch 5, Train Loss: 0.6455298900859763, Val Loss: 0.6610975813271472, Val Acc:
0.7760491100428738
Model saved with val loss: 0.6610975813271472 at ./Models/best_convnet.pth
Epoch 6, Train Loss: 0.6157149611881536, Val Loss: 0.6192450842598641, Val Acc:
0.7757892685461868
Model saved with val loss: 0.6192450842598641 at ./Models/best_convnet.pth
Epoch 7, Train Loss: 0.5916923259134013, Val Loss: 0.6018127820387807, Val Acc:
0.7879043783292192
Model saved with val loss: 0.6018127820387807 at ./Models/best_convnet.pth
Epoch 8, Train Loss: 0.5710766438184759, Val Loss: 0.5788216089533869, Val Acc:
0.7935559308821619
Model saved with val loss: 0.5788216089533869 at ./Models/best_convnet.pth
Epoch 9, Train Loss: 0.5553759632597778, Val Loss: 0.5728239509299289, Val Acc:
0.793783292191763
```

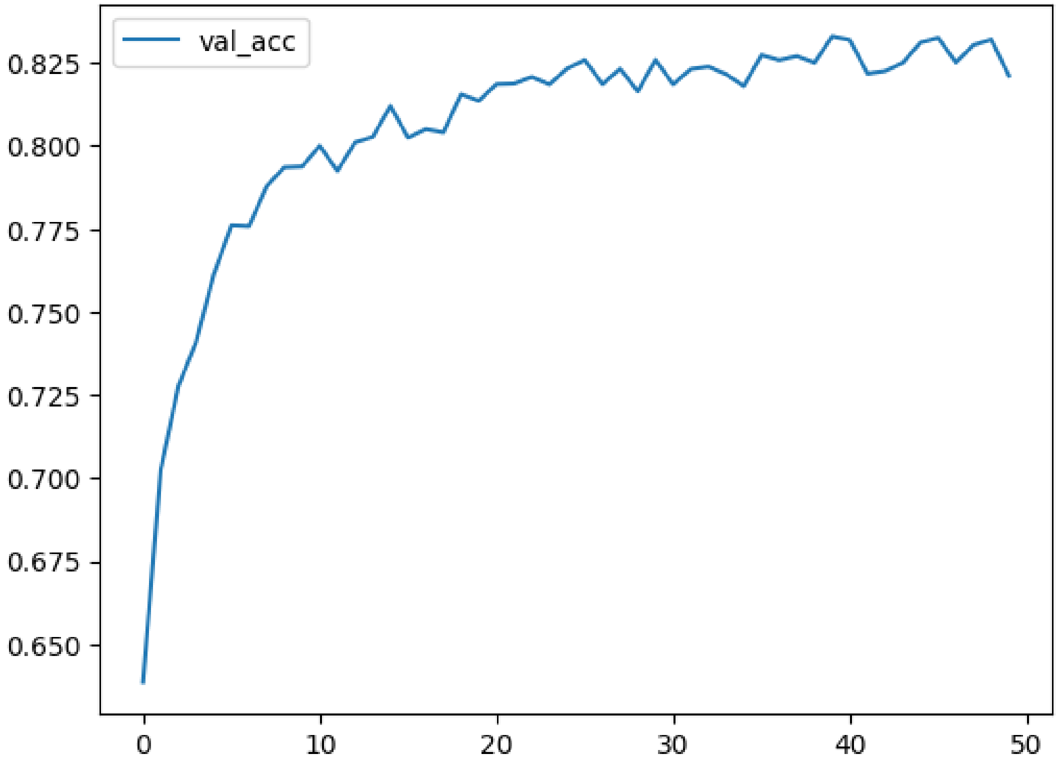
```
Model saved with val loss: 0.5728239509299289 at ./Models/best_convnet.pth
Epoch 10, Train Loss: 0.5430314768077678, Val Loss: 0.5620531294328526, Val Acc:
0.8000194881122515
Model saved with val loss: 0.5620531294328526 at ./Models/best_convnet.pth
Epoch 11, Train Loss: 0.5306052828948509, Val Loss: 0.5711346974563499, Val Acc:
0.7924840847083279
Epoch 12, Train Loss: 0.5227330668262458, Val Loss: 0.5523390135227829, Val Acc:
0.8010263739119138
Model saved with val loss: 0.5523390135227829 at ./Models/best_convnet.pth
Epoch 13, Train Loss: 0.5124426236065, Val Loss: 0.5507673701486112, Val Acc:
0.8026828634532935
Model saved with val loss: 0.5507673701486112 at ./Models/best_convnet.pth
Epoch 14, Train Loss: 0.5054614976896314, Val Loss: 0.5207146908449607, Val Acc:
0.8120371573340263
Model saved with val loss: 0.5207146908449607 at ./Models/best_convnet.pth
Epoch 15, Train Loss: 0.496533748702233, Val Loss: 0.54070393326436, Val Acc:
0.8024555021436923
Epoch 16, Train Loss: 0.4920414529293659б, Val Loss: 0.5565261555330776, Val
Acc: 0.8050539171105625
Epoch 17, Train Loss: 0.4866102635531371, Val Loss: 0.5428063363796206, Val Acc:
0.8040795114979862
Epoch 18, Train Loss: 0.4797559358270548, Val Loss: 0.5011656375781769, Val Acc:
0.815545017539301
Model saved with val loss: 0.5011656375781769 at ./Models/best_convnet.pth
Epoch 19, Train Loss: 0.4778808984981942, Val Loss: 0.5026976375348348, Val Acc:
0.8135312459399766
Epoch 20, Train Loss: 0.47302365438152705, Val Loss: 0.5177243951386999, Val
Acc: 0.8185981551253735
Epoch 21, Train Loss: 0.4690126609216885, Val Loss: 0.49199092440763614, Val
Acc: 0.8187605560608029
Model saved with val loss: 0.49199092440763614 at ./Models/best_convnet.pth
Epoch 22, Train Loss: 0.46614980461474176, Val Loss: 0.5191466875320159, Val
Acc: 0.8207418474730415
Epoch 23, Train Loss: 0.4631803642893556, Val Loss: 0.5048709178298928, Val Acc:
0.8185331947512018
Epoch 24, Train Loss: 0.46017893265171095, Val Loss: 0.4889899065035278, Val
Acc: 0.8232753020657398
Model saved with val loss: 0.4889899065035278 at ./Models/best_convnet.pth
Epoch 25, Train Loss: 0.457473272635321, Val Loss: 0.4919557370717404, Val Acc:
0.8258412368455242
Epoch 26, Train Loss: 0.45458063929227766, Val Loss: 0.503893121891304, Val Acc:
0.8185656749382877
Epoch 27, Train Loss: 0.45373159656279366, Val Loss: 0.4860133882902122, Val
Acc: 0.8232103416915681
Model saved with val loss: 0.4860133882902122 at ./Models/best_convnet.pth
Epoch 28, Train Loss: 0.45223908964133197, Val Loss: 0.5026050437462293, Val
Acc: 0.8164219825906197
Epoch 29, Train Loss: 0.4495996057960113, Val Loss: 0.4824586473366677, Val Acc:
```

```
0.8259061972196959
Model saved with val loss: 0.4824586473366677 at ./Models/best_convnet.pth
Epoch 30, Train Loss: 0.44761250872105923, Val Loss: 0.4955121452687066, Val
Acc: 0.8185331947512018
Epoch 31, Train Loss: 0.44547789775572616, Val Loss: 0.4859658282865245, Val
Acc: 0.8231129011303105
Epoch 32, Train Loss: 0.4448572159022534, Val Loss: 0.46956808222912305, Val
Acc: 0.8238599454332857
Model saved with val loss: 0.46956808222912305 at ./Models/best_convnet.pth
Epoch 33, Train Loss: 0.4450043572848975, Val Loss: 0.48506994873501924, Val
Acc: 0.8215213719631025
Epoch 34, Train Loss: 0.4422958985002018, Val Loss: 0.49659111958117, Val Acc:
0.8180135117578278
Epoch 35, Train Loss: 0.4420570718445354, Val Loss: 0.4715094081090371, Val Acc:
0.8273353254514746
Epoch 36, Train Loss: 0.4394407582363546, Val Loss: 0.4713386546197586, Val Acc:
0.8258087566584383
Epoch 37, Train Loss: 0.43755882730280216, Val Loss: 0.46746430115103105, Val
Acc: 0.8270105235806158
Model saved with val loss: 0.46746430115103105 at ./Models/best_convnet.pth
Epoch 38, Train Loss: 0.4381258869947753, Val Loss: 0.46767747270664817, Val
Acc: 0.8250292321683773
Epoch 39, Train Loss: 0.43545659640588585, Val Loss: 0.4604470732381039, Val
Acc: 0.8329543978173314
Model saved with val loss: 0.4604470732381039 at ./Models/best_convnet.pth
Epoch 40, Train Loss: 0.4360098505591566, Val Loss: 0.4564339689763412, Val Acc:
0.8318825516434974
Model saved with val loss: 0.4564339689763412 at ./Models/best_convnet.pth
Epoch 41, Train Loss: 0.43412529729610605, Val Loss: 0.4879051185950931, Val
Acc: 0.8217162530856178
Epoch 42, Train Loss: 0.43421189478753636, Val Loss: 0.4794824834707989, Val
Acc: 0.8224957775756788
Epoch 43, Train Loss: 0.43249417088439973, Val Loss: 0.4727948047568865, Val
Acc: 0.8250292321683773
Epoch 44, Train Loss: 0.4319671853838663, Val Loss: 0.454218094330958, Val Acc:
0.8311355073405223
Model saved with val loss: 0.454218094330958 at ./Models/best_convnet.pth
Epoch 45, Train Loss: 0.43131264328964347, Val Loss: 0.46629678288538506, Val
Acc: 0.8324996751981292
Epoch 46, Train Loss: 0.43004299860344086, Val Loss: 0.48895213916232405, Val
Acc: 0.825094192542549
Epoch 47, Train Loss: 0.43010535615354306, Val Loss: 0.45414820405522116, Val
Acc: 0.8303559828504612
Model saved with val loss: 0.45414820405522116 at ./Models/best_convnet.pth
Epoch 48, Train Loss: 0.4304160912293495, Val Loss: 0.469713594296392, Val Acc:
0.832012472391841
Epoch 49, Train Loss: 0.42711283526317767, Val Loss: 0.48224779251703714, Val
Acc: 0.8210991295309861
```

[48]:
```
plt.figure(1)
plt.plot(history['train_loss'], label='train_loss')
plt.plot(history['val_loss'], label='val_loss')
plt.legend()

plt.figure(2)
plt.plot(history['val_acc'], label='val_acc')
plt.legend()
```

[48]: <matplotlib.legend.Legend at 0x7ad9b4b233d0>





0.2 Testing

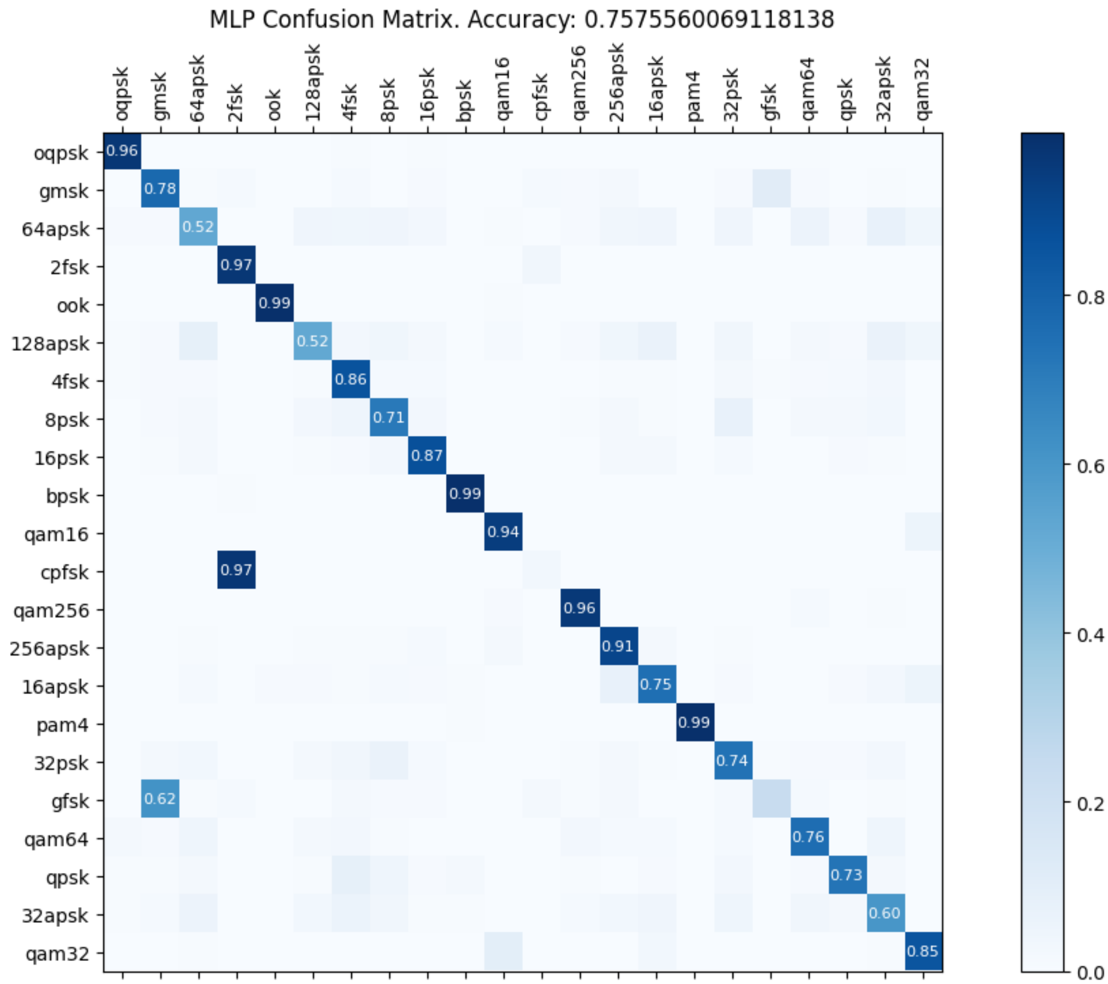
[62]:
```
results = {
    "mlp": {
        'preds': [],
        'labels': []
    },
    "convnet": {
        'preds': [],
        'labels': []
    }
}
test_dataset = IQDataset(X[test_idx], y[test_idx], data_transform=MLP_txfm)
test_dataloader = DataLoader(test_dataset, batch_size=32, shuffle=False)


model = MLP(in_features=2*128, hidden_features=256,
            out_features=22)
model.load_state_dict(torch.load('./Models/best_mlp.pth', weights_only=True))

model = model.to(device)
```

```
model.eval()
with torch.no_grad():
    for data, labels in test_dataloader:
        data, labels = data.to(device), labels.to(device)
        output = model(data)
        pred = F.softmax(output, dim=-1).argmax(dim=1, keepdim=True)
        results['mlp']['preds'].extend(pred.cpu().numpy())
        results['mlp']['labels'].extend(labels.cpu().numpy())

test_dataset = IQDataset(X[test_idx], y[test_idx])
test_dataloader = DataLoader(test_dataset, batch_size=32, shuffle=False)

model = ConvNet(in_channels=2, num_classes=22).to(device)
model.load_state_dict(
    torch.load('./Models/best_convnet.pth', weights_only=True))

model.eval()
with torch.no_grad():
    for data, labels in test_dataloader:
        data, labels = data.to(device), labels.to(device)
        output = model(data)
        pred = F.softmax(output, dim=-1).argmax(dim=1, keepdim=True)
        results['convnet']['preds'].extend(pred.cpu().numpy())
        results['convnet']['labels'].extend(labels.cpu().numpy())
```

[78]:
```
# visualize the results
# compute the accuracy, precision, recall, f1-score
# and the confusion matrix for each model
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score

plt.rcParams['figure.figsize'] = [15, 8]

labels = ['oqpsk', 'gmsk', '64apsk', '2fsk', 'ook', '128apsk', '4fsk', '8psk',␣
 ↪'16psk', 'bpsk', 'qam16',
          'cpfsk', 'qam256', '256apsk', '16apsk', 'pam4', '32psk', 'gfsk',␣
 ↪'qam64', 'qpsk', '32apsk', 'qam32']
for model_name, result in results.items():
    print(f'Performance of {model_name}')
    print(classification_report(
        result['labels'], result['preds'], target_names=labels))
    acc = accuracy_score(result['labels'], result['preds'])
    print('Accuracy:', acc)
    conf_mat = confusion_matrix(
        result['labels'], result['preds'], normalize='true')
    ax = plt.subplot()
    im = ax.matshow(conf_mat, cmap='Blues')
```

```python
    ax.set_xticks(range(len(labels)))
    ax.set_yticks(range(len(labels)))
    ax.set_xticklabels(labels, rotation=90)
    ax.set_yticklabels(labels)
    ax.title.set_text(
        f'{model_name.upper()} Confusion Matrix. Accuracy: {acc}')
    plt.colorbar(im)

    for (i, j), val in np.ndenumerate(conf_mat):
        if val < 0.5:
            continue
        ax.text(j, i, f'{val:.2f}', ha='center', va='center',
                color='white', fontsize=8)

    plt.show()
```

```
Performance of mlp
              precision    recall  f1-score   support

       oqpsk       0.94      0.96      0.95      2999
        gmsk       0.52      0.78      0.62      2999
      64apsk       0.62      0.52      0.57      2999
        2fsk       0.49      0.97      0.65      2999
         ook       0.98      0.99      0.99      2999
     128apsk       0.75      0.52      0.62      2999
        4fsk       0.70      0.86      0.77      2999
        8psk       0.70      0.71      0.71      2999
       16psk       0.84      0.87      0.86      2999
        bpsk       0.96      0.99      0.97      2999
       qam16       0.85      0.94      0.89      2998
       cpfsk       0.33      0.03      0.06      2999
      qam256       0.90      0.96      0.93      2999
     256apsk       0.76      0.91      0.83      2998
      16apsk       0.75      0.75      0.75      2998
        pam4       1.00      0.99      1.00      2999
       32psk       0.70      0.74      0.72      2999
        gfsk       0.63      0.24      0.35      2999
       qam64       0.79      0.76      0.77      2999
        qpsk       0.87      0.73      0.79      2999
      32apsk       0.64      0.60      0.62      2998
       qam32       0.82      0.85      0.84      2999

    accuracy                           0.76     65974
   macro avg       0.75      0.76      0.74     65974
weighted avg       0.75      0.76      0.74     65974

Accuracy: 0.7575560069118138
```

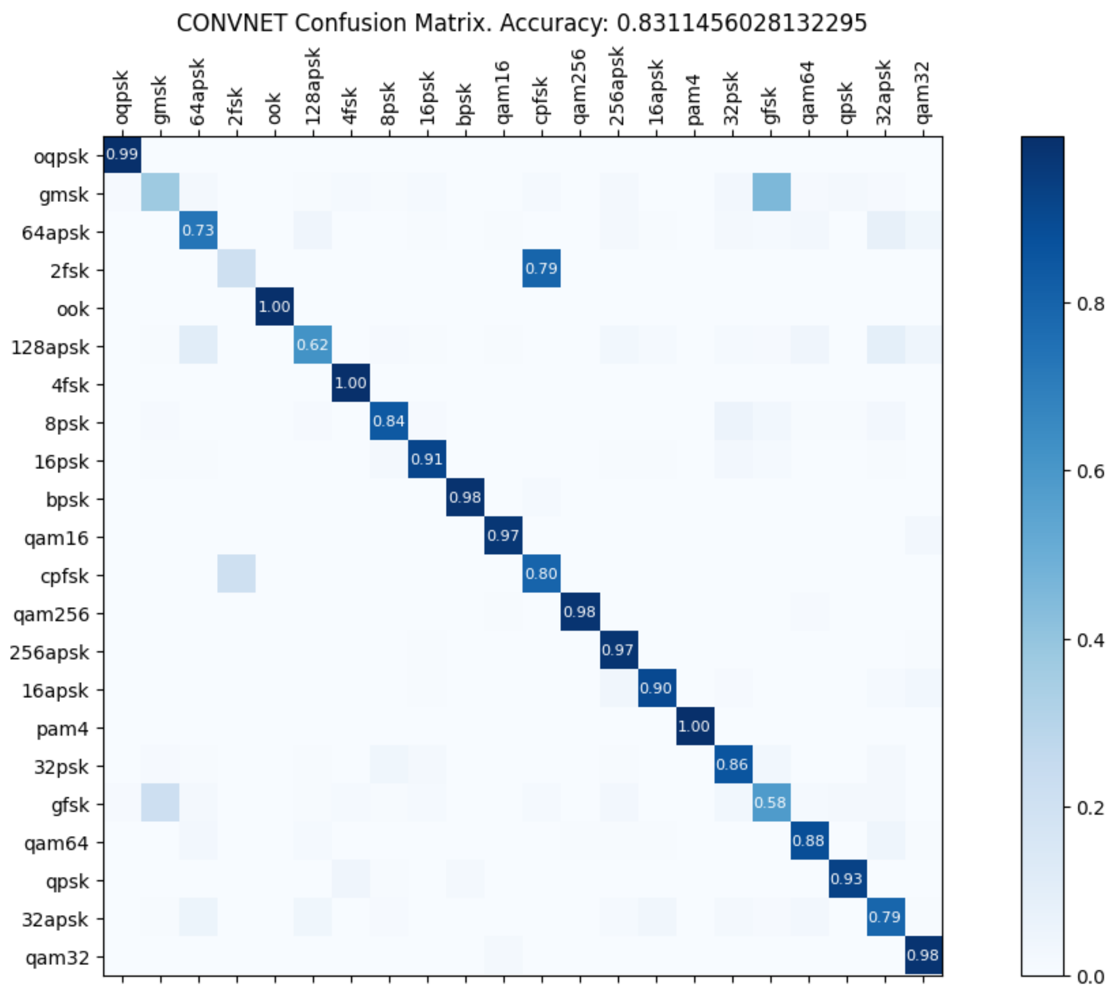


Performance of convnet

	precision	recall	f1-score	support
oqpsk	0.98	0.99	0.99	2999
gmsk	0.59	0.37	0.46	2999
64apsk	0.75	0.73	0.74	2999
2fsk	0.50	0.21	0.29	2999
ook	1.00	1.00	1.00	2999
128apsk	0.83	0.62	0.71	2999
4fsk	0.93	1.00	0.96	2999
8psk	0.90	0.84	0.87	2999
16psk	0.91	0.91	0.91	2999
bpsk	0.97	0.98	0.98	2999
qam16	0.96	0.97	0.96	2998
cpfsk	0.49	0.80	0.61	2999
qam256	0.98	0.98	0.98	2999
256apsk	0.86	0.97	0.91	2998

16apsk	0.92	0.90	0.91	2998
pam4	1.00	1.00	1.00	2999
32psk	0.81	0.86	0.83	2999
gfsk	0.51	0.58	0.54	2999
qam64	0.87	0.88	0.88	2999
qpsk	0.94	0.93	0.93	2999
32apsk	0.71	0.79	0.75	2998
qam32	0.87	0.98	0.92	2999
accuracy			0.83	65974
macro avg	0.83	0.83	0.82	65974
weighted avg	0.83	0.83	0.82	65974

```
Accuracy: 0.8311456028132295
```



Observations from the experiment include the following:

1. The performance of the ConvNet is not comparable to that of the MLP. There is significant gains with the ConvNet architecture
2. Both architectures struggle to classify class with similar signal characteristics eg. gmsk and gfsk, cpfsk and 2fsk