Task1 Report

Description and analysis of the problem definition

功率放大器（PA）行为建模的主要目的是通过理解其输入信号与输出信号之间的关系来实现更好的数字预失真。PA 的输入信号和输出信号分别用复数数组表示，并且PA的非线性和记忆效应可以通过公式表示为：

$$y_n = f(x_n, x_{n-1}, \ldots, x_{n-m})$$

其中，y_n 是输出信号，$x_n, x_{n-1}, \ldots, x_{n-m}$ 是输入信号，m 是PA的记忆深度。为了评估模型的性能，我们使用归一化均方误差（NMSE）作为指标，具体公式为：

$$NMSE = 10 \log \left(\frac{\sum_{n=1}^{N} (I_{out}(n) - \hat{I}_{out}(n))^2 + (Q_{out}(n) - \hat{Q}_{out}(n))^2}{\sum_{n=1}^{N} I_{out}(n)^2 + Q_{out}(n)^2} \right)$$

其中，$\hat{I}_{out}(n)$ 和 $\hat{Q}_{out}(n)$ 分别表示预测值的实部和虚部，$I_{out}(n)$ 和 $Q_{out}(n)$ 分别表示实际值的实部和虚部。

Introduction to the data preprocessing process

在数据预处理中，我们的主要目标是将复数信号的输入和输出分解为实部和虚部，然后将它们组合成一个特征向量。具体步骤如下：

1. **加载数据**：从`.mat`文件中加载训练集和测试集的数据，分别得到`paInput`和`paOutput`，这些数据是复数数组。
2. **数据分解与重构**：将每个复数数组分解为实部和虚部，并将它们组合在一起。为了能够捕捉PA的记忆效应，我们采用长度为10的序列来构建特征和标签。
3. **构建特征矩阵和标签矩阵**：通过遍历输入数据和输出数据，根据序列长度生成特征矩阵X和标签矩阵Y。

具体的代码实现如下：

```
def preprocess_data(pa_input, pa_output, sequence_length=10):
    X = []
    Y = []

    for i in range(len(pa_input) - sequence_length):
        X.append(np.hstack((np.real(pa_input[i:i + sequence_length]), np.imag(pa_input[i:i + sequence_length]))))
        Y.append(np.hstack((np.real(pa_output[i:i + sequence_length]), np.imag(pa_output[i:i + sequence_length]))))

    return np.array(X), np.array(Y)
```

Detailed explanation of the methods used and implementation process

为了实现这个任务，我们采用了一个简单的神经网络模型。具体步骤如下：

1. **构建神经网络模型**：
   我们采用了一个包含两个隐藏层的全连接神经网络，每个隐藏层包含64个神经元，激活函数为ReLU。最后一层输出与输入形状相同的线性层。模型编译时使用均方误差（MSE）作为损失函数，优化器采用Adam。

```
def build_model(input_shape):
    model = Sequential()
    model.add(Input(shape=(input_shape,)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(input_shape, activation='linear'))
    model.compile(optimizer='adam', loss='mse')
    return model
```

2. **模型训练**：
   使用预处理后的训练数据对模型进行训练，设置训练轮数为100，批次大小为32。

```
model.fit(X_train, Y_train, epochs=100, batch_size=32, verbose=0)
```

3. **模型预测与评估**：
   通过训练好的模型对测试数据进行预测，并使用NMSE评估模型的性能。

```
Y_pred = model.predict(X_test)
nmse = evaluate_model(Y_pred, Y_test)
```

   其中，评估函数`evaluate_model`计算预测值和实际值的NMSE。

```
def evaluate_model(Y_pred, Y_test):
    Iout_pred = Y_pred[:, :Y_pred.shape[1]//2]
    Iout_test = Y_test[:, :Y_test.shape[1]//2]
    Qout_pred = Y_pred[:, Y_pred.shape[1]//2:]
    Qout_test = Y_test[:, Y_test.shape[1]//2:]

    nmse = 10 * np.log10(
        np.sum((Iout_test - Iout_pred)**2 + (Qout_test - Qout_pred)**2)
/
        np.sum(Iout_test**2 + Qout_test**2)
    )

    return nmse
```

Test results


```
Task 1
Normalized Mean Square Error (NMSE): -32.5756
```