演示应用软件 matlab 代码：

```
classdef wave_pre < matlab.apps.AppBase
% Properties that correspond to app components
properties (Access = public)
UIFigure matlab.ui.Figure
UIAxes matlab.ui.control.UIAxes
Button matlab.ui.control.StateButton
DATASlider matlab.ui.control.Slider
Label matlab.ui.control.Label
Button_2 matlab.ui.control.StateButton
Label_2 matlab.ui.control.Label
YearsLabel matlab.ui.control.Label
YearsSpinner matlab.ui.control.Spinner
end
% Callbacks that handle component events
methods (Access = private)
% Value changed function: Button
function ButtonValueChanged(app, event)
value = app.Button.Value;
years=2013:2020;
for year=years
h=load([num2str(year) 'pred.mat']);
hp=h.y_pred;
hp=hp';
filename=['D:\Test_counter\wave_0001' num2str(year) '.nc'];
ncdisp(filename,'/','full')
partition=ncread(filename,'partition');
time=ncread(filename,'time');
lon=ncread(filename,'lon');
lat=ncread(filename,'lat');
hs=ncread(filename,'hs');
hs1=hs;
indices = 1:6:size(hs1, 2);
result_hs= hs1(:, indices);
cha=result_hs-hp;
meanmlp=mean(hp,2);
meanfvcom=mean(result_hs,2);
meancha=meanfvcom-meanmlp;
xmd=500;
ymd=500;
d=load('D:\Test_counter\bigmap.dat');
xoutline=[d(:,1);d(1,1)];
youtline=[d(:,2);d(1,2)];
[X,Y]=meshgrid(linspace(min(xoutline),max(xoutline),xmd),linspace(min(youtline),max(youtli
ne),ymd));
Inwater=double(inpolygon(X,Y,xoutline,youtline));
Inwater(Inwater==0)=NaN;
d=load('D:\Test_counter\hainandao.dat');
```

```
x1outline=[d(:,1);d(1,1)];
y1outline=[d(:,2);d(1,2)];
q=lon(:,1);
w=lat(:,1);
z=double(meanmlp);
Zh=griddata(double(q),double(w),double(z),X,Y,'V4');
ZhPlot=Zh.*Inwater;
f=contourf(app.UIAxes,X,Y,ZhPlot);
shading interp;
hold(app.UIAxes,"on");
plot(app.UIAxes,xoutline, youtline, 'LineWidth', 0.1, 'Color', 'k');
colormap (app.UIAxes,jet);
colormap(app.UIAxes,slanCM('coolwarm'));
hold(app.UIAxes,"on");
caxis(app.UIAxes,[0 2]);
axis(app.UIAxes,[98 123 0 25]);
box on
grid on;
set(gca,'GridLineStyle',':');
contourLevels = 1:1:8;
fill(app.UIAxes,x1outline,y1outline,[1 1 1]*0.7);
plot(app.UIAxes,x1outline, y1outline, 'LineWidth', 0.1, 'Color', 'k');
set(app.UIAxes,'color',[1 1 1]*0.7,'FontSize',15);
hc=colorbar(app.UIAxes,'EastOutside','FontSize',14);
set(get(hc,'title'),'string','Hs/m','FontName','Times New Roman','FontSize',15);
title(app.UIAxes,[num2str(year), 'yr'],'Fontname','Times New Roman')
set(gca,'xtick',[98:5:123],'xticklabel',{'98^oE','103^oE','108^oE','113^oE','118^oE','123^oE'},'lin
ewidth',1,'FontSize',21,'Fontname','Times New Roman');
set(gca,'ytick',[0:5:25],'yticklabel',{'0^oN','5^oN','10^oN','15^oN','20^oN','25^oN'},'linewidth',1,
'FontSize',21,'Fontname','Times New Roman');
set(app.UIAxes,'Layer','top')
hold(app.UIAxes,"off")
xmd=500;
ymd=500;
figure('Units','pixels','Position',[100,100,700,500]);
d=load('D:\Test_counter\bigmap.dat');
xoutline=[d(:,1);d(1,1)];
youtline=[d(:,2);d(1,2)];
[X,Y]=meshgrid(linspace(min(xoutline),max(xoutline),xmd),linspace(min(youtline),max(youtli
ne),ymd));
Inwater=double(inpolygon(X,Y,xoutline,youtline));
Inwater(Inwater==0)=NaN;
d=load('D:\Test_counter\hainandao.dat');
x1outline=[d(:,1);d(1,1)];
y1outline=[d(:,2);d(1,2)];
q=lon(:,1);
w=lat(:,1);
z=double(meanmlp);
```

```
Zh=griddata(double(q),double(w),double(z),X,Y,'V4');
ZhPlot=Zh.*Inwater;
f=contourf(X,Y,ZhPlot);
shading interp;
hold on;
plot(xoutline, youtline, 'LineWidth', 0.1, 'Color', 'k');
colormap(slanCM('coolwarm'));
hold on
caxis([0 2]);
axis([98 123 0 25]);
box on
grid on;
set(gca,'GridLineStyle',':');
contourLevels = 1:1:8;
fill(x1outline,y1outline,[1 1 1]*0.7);
plot(x1outline, y1outline, 'LineWidth', 0.1, 'Color', 'k');
set(gca,'color',[1 1 1]*0.7,'FontSize',15);
hc=colorbar('EastOutside','FontSize',14);
set(get(hc,'title'),'string','Hs/m','FontName','Times New Roman','FontSize',15);
title([num2str(year), 'yr'],'Fontname','Times New Roman')
set(gca,'xtick',[98:5:123],'xticklabel',{'98^oE','103^oE','108^oE','113^oE','118^oE','123^oE'},'lin
ewidth',1,'FontSize',21,'Fontname','Times New Roman');
set(gca,'ytick',[0:5:25],'yticklabel',{'0^oN','5^oN','10^oN','15^oN','20^oN','25^oN'},'linewidth',1,
'FontSize',21,'Fontname','Times New Roman');
hold off
set(gca,'Layer','top')
hold on
saveas(gcf,num2str(year),'jpg');
close
end
end
% Value changed function: YearsSpinner
function YearsSpinnerValueChanged(app, event)
app.DATASlider.Value = app.YearsSpinner.Value;
end
% Value changed function: Button_2
function Button_2ValueChanged(app, event)
value = app.Button_2.Value;
year = app.YearsSpinner.Value;
figname = [num2str(year), '.jpg'];
A = imread(figname);
imshow(A);
end
% Callback function
function DATASliderValueChanging(app, event)
changingValue = event.Value;
app.Gauge.Value = changingValue;
figname = [num2str(changingValue), '.jpg'];
```

```
A = imread(figname);
imshow(A);
end
% Callback function
function YearsSpinnerValueChanged2(app, event)
%value = app.YearsSpinner.Value;
end
% Callback function
function YearsSpinnerValueChanged3(app, event)
%value = app.YearsSpinner.Value;
end
end
% Component initialization
methods (Access = private)
% Create UIFigure and components
function createComponents(app)
% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible', 'off');
app.UIFigure.Position = [100 100 640 480];
app.UIFigure.Name = 'UI Figure';
% Create UIAxes
app.UIAxes = uiaxes(app.UIFigure);
title(app.UIAxes, '')
xlabel(app.UIAxes, 'Longitude（°E）')
ylabel(app.UIAxes, 'Latitude（°N）')
app.UIAxes.FontName = 'Times New Roman';
app.UIAxes.FontSize = 14;
app.UIAxes.TitleFontWeight = 'bold';
app.UIAxes.Position = [22 96 578 332];
% Create Button
app.Button = uibutton(app.UIFigure, 'state');
app.Button.ValueChangedFcn = createCallbackFcn(app, @ButtonValueChanged, true);
app.Button.Text = '预测绘图';
app.Button.BackgroundColor = [0 0.4471 0.7412];
app.Button.FontName = '微软雅黑';
app.Button.FontSize = 16;
app.Button.FontColor = [1 1 1];
app.Button.Position = [51 9 97 41];
% Create DATASlider
app.DATASlider = uislider(app.UIFigure);
app.DATASlider.Limits = [2013 2020];
app.DATASlider.MinorTicks = [];
app.DATASlider.FontName = 'Times New Roman';
app.DATASlider.FontSize = 16;
app.DATASlider.Position = [295 47 305 3];
app.DATASlider.Value = 2013;
% Create Label
app.Label = uilabel(app.UIFigure);
```

```
app.Label.BackgroundColor = [0 0.4471 0.7412];
app.Label.HorizontalAlignment = 'center';
app.Label.FontName = '微软雅黑';
app.Label.FontSize = 20;
app.Label.FontColor = [1 1 1];
app.Label.Position = [1 439 640 42];
app.Label.Text = '海浪智能快速预报演示软件';
% Create Button_2
app.Button_2 = uibutton(app.UIFigure, 'state');
app.Button_2.ValueChangedFcn = createCallbackFcn(app, @Button_2ValueChanged, true);
app.Button_2.Text = '数据查看';
app.Button_2.BackgroundColor = [0.9294 0.6941 0.1255];
app.Button_2.FontName = '微软雅黑';
app.Button_2.FontSize = 16;
app.Button_2.FontColor = [1 1 1];
app.Button_2.Position = [163 8 98 42];
% Create Label_2
app.Label_2 = uilabel(app.UIFigure);
app.Label_2.FontName = '微软雅黑';
app.Label_2.FontSize = 16;
app.Label_2.Position = [418 60 60 23];
app.Label_2.Text = '时间/年';
% Create YearsLabel
app.YearsLabel = uilabel(app.UIFigure);
app.YearsLabel.HorizontalAlignment = 'right';
app.YearsLabel.FontName = 'Times New Roman';
app.YearsLabel.FontSize = 18;
app.YearsLabel.Position = [81 67 46 22];
app.YearsLabel.Text = 'Years';
% Create YearsSpinner
app.YearsSpinner = uispinner(app.UIFigure);
app.YearsSpinner.Limits = [2013 2020];
app.YearsSpinner.ValueChangedFcn = createCallbackFcn(app, @YearsSpinnerValueChanged,
true);
app.YearsSpinner.FontName = 'Times New Roman';
app.YearsSpinner.FontSize = 18;
app.YearsSpinner.Position = [142 66 82 22.8000011444092];
app.YearsSpinner.Value = 2013;
% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end
% App creation and deletion
methods (Access = public)
% Construct app
function app = wave_pre
% Create UIFigure and components
createComponents(app)
```

```
1

2       % Register the app with App Designer
3       registerApp(app, app.UIFigure)
4       if nargout == 0
5       clear app
6       end
7       end
8       % Code that executes before app deletion
9       function delete(app)
10      % Delete UIFigure when app is deleted
11      delete(app.UIFigure)
12      end
13      end
14      end
15
```

附录 1：训练模型 Python 代码

```
#这里是调用 Scikit-Learn
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense, Dropout
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.wrappers.scikit_learn import KerasRegressor
from keras import regularizers
import matplotlib.pyplot as plt
import matplotlib
from sklearn.model_selection import train_test_split
from sklearn.metrics import make_scorer
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from tensorflow.keras.optimizers import Adam
from scipy.io import loadmat
#%% 导入训练数据
mat_data = loadmat('D:\python\CNN-LSTM\风场数据提取/uv2002-20110.5.mat')
inputuv= mat_data['uv']
h_data=loadmat('D:\python\CNN-LSTM\波浪数据提取/oh2002-2011.mat')
ouths=h_data['all_hs']
#%% 导入测试数据
x_data= loadmat('D:\python\CNN-LSTM\风场数据提取/uv20120.5')
xinput=x_data['uv']
#%% 数据归一化处理
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(inputuv)
X_test_scaled = scaler.transform(xinput)
X_test=X_test_scaled
#%%
X_train1=X_train_scaled
y_train1=ouths
X_train, X_val, y_train, y_val = train_test_split(X_train1, y_train1, test_size=0.2,
random_state=42)
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import LeaveOneOut
# 定义 MLPRegressor 模型
mlp = MLPRegressor()
# 定义参数网格
param_grid = {
    'solver': ['adam'],
    #'hidden_layer_sizes': [(256,256)],
```

```
        'hidden_layer_sizes': [(32), (64), (128), (256), (32, 32), (64, 64), (128, 128),(256, 256), (32, 32, 32), (64, 64, 64), (128, 128, 128), (256, 256, 256)],
        'activation': ['relu'],
        'alpha': [0.0001],
        'beta_1': [0.9],
        'beta_2': [0.999],
        'batch_size': [200],
        'early_stopping': [False],
        'validation_fraction': [0.1],
        'learning_rate': ['adaptive'],
        'max_iter': [200],
        'learning_rate_init': [0.001],
        'epsilon': [1e-08],
        'shuffle': [True],
        'random_state': [None],
        'warm_start': [False],
        'n_iter_no_change': [10],
        'verbose': [True],
        'tol': [0.0001]}
    # 初始化 GridSearchCV
    grid_search = GridSearchCV(estimator=mlp, param_grid=param_grid, scoring='r2', cv=5, verbose=2)
    # 在训练数据上直接拟合 GridSearchCV neg_mean_absolute_error   r2
    grid_result = grid_search.fit(X_train, y_train)
    print("Best Parameters:", grid_result.best_params_)
    # 输出最佳模型的测试分数
    print("Best Score:", grid_result.best_score_)
    #%% 获取网格搜索结果
    from matplotlib.font_manager import FontProperties
    font = FontProperties(fname=r"C:\WINDOWS\Fonts\simsun.TTC",size=16.5)
    results = grid_search.cv_results_
    import matplotlib.pyplot as plt
    # 获取每次迭代的损失值和 MAE
    loss_values = grid_result.best_estimator_.loss_curve_
    mae_values = -grid_result.cv_results_['mean_test_score']
    # 创建一个新的图形
    plt.figure(figsize=(10, 5))
    # 绘制损失值和 MAE 随迭代次数的变化图
    plt.plot(loss_values, color='royalblue')
    #plt.plot(mae_values, label='MAE', color='red')
    # 添加图例和标题
    plt.legend()
    plt.xlabel('Epochs',fontproperties=font)
    plt.ylabel('Loss',fontproperties=font)
    # 调整坐标轴粗细
    plt.gca().spines['bottom'].set_linewidth(1.5)
    plt.gca().spines['left'].set_linewidth(1.5)
    plt.gca().spines['top'].set_linewidth(1.5)
```

```
plt.gca().spines['right'].set_linewidth(1.5)
# 调整坐标轴刻度文字大小
plt.tick_params(axis='both', which='major', labelsize=14)
# 调整坐标轴刻度向内移动
plt.tick_params(axis='both', which='major', direction='in', labelsize=14)
# 显示图形
#plt.show()
#g.ax_joint.legend()
# 显示图表
#plt.show()
# 调整布局以防止标签被裁剪
plt.tight_layout()
plt.savefig('损失系数 5.28.png', dpi=300, bbox_inches='tight')  # 保存为 PNG 格式，分辨
率为 300 dpi
#%%
import joblib
# 保存最佳模型
best_model = grid_result.best_estimator_
joblib.dump(best_model, 'best_model256321.pkl')
#%% 加载最佳模型
import joblib
loaded_model = joblib.load('best_model256321.pkl')
#%%在验证集上评估最佳模型性能
best_model = grid_search.best_estimator_
validation_score = best_model.score(X_val, y_val)
print("Validation R^2 Score:", validation_score)
#%%
# 使用最佳模型 best_model 对验证集 X_val 进行预测
# 使用加载的模型进行预测
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score
val_predict = loaded_model.predict(X_val)
# 评估预测性能
validation_score1 = r2_score(y_val, val_predict)
#%% 绘制模型训练的误差图
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score
# 假设这里 y_true 和 y_pred 是您的实际数据
y_pred = val_predict
y_true = y_val
#y_true_flat = y_true.flatten()
#y_pred_flat = y_pred.flatten()
#r2_total = r2_score(y_true_flat, y_pred_flat )
#rmse_total = np.sqrt(mean_squared_error(y_true_flat, y_pred_flat ))
```

```
#bias_total = np.mean(y_pred_flat - y_true_flat)
#%%
from matplotlib.font_manager import FontProperties
font = FontProperties(fname=r"C:\WINDOWS\Fonts\simsun.TTC",size=16.5)
mean_y_pred = np.mean(y_pred, axis=1)
mean_y_true = np.mean(y_true, axis=1)
# 计算 R^2, RMSE 和 Bias
r2 = r2_score(mean_y_true,mean_y_pred)
rmse = np.sqrt(mean_squared_error(mean_y_true,mean_y_pred))
bias = np.mean(mean_y_pred - mean_y_true)
# 创建 JointGrid 对象
g = sns.JointGrid(x=mean_y_true, y=mean_y_pred, space=0, ratio=4)
# 绘制散点图和直方图
g = g.plot_joint(plt.scatter, color="royalblue", edgecolor="white", s=50)
# 绘制直方图
g = g.plot_marginals(sns.histplot, kde=True, color="royalblue")
# 计算最佳拟合线的斜率和截距
slope, intercept = np.polyfit(mean_y_true, mean_y_pred, 1)
# 创建 x 轴的值范围
line = np.linspace(min(np.min(mean_y_true), np.min(mean_y_pred)),
max(np.max(mean_y_true), np.max(mean_y_pred)), 30000)
# 计算最佳拟合线的 y 值
best_fit_line = slope * line + intercept
# 绘制最佳拟合线
g.ax_joint.plot(line, best_fit_line, color='royalblue')
# 绘制 1:1 线
g.ax_joint.plot(line, line, color='black', linestyle='--')
# 添加注释
g.ax_joint.text(0.05, 0.95, f'$R^2$={r2:.2f}\nRMSE={rmse:.2f}m\nBias={bias:.2f}m',
                        verticalalignment='top', horizontalalignment='left',
transform=g.ax_joint.transAxes, fontsize=12)
# 设置轴标签
g.ax_joint.set_xlabel('FVCOM-SWAVE 空间平均有效波高（m）',fontproperties=font)
g.ax_joint.set_ylabel('MLP 空间平均有效波高（m）',fontproperties=font)
# 设置 x 轴和 y 轴的相同范围
max_limit = max(np.max(mean_y_true), np.max(mean_y_pred))
min_limit = -0.5;
g.ax_joint.set_xlim(min_limit, max_limit+0.5)
g.ax_joint.set_ylim(min_limit, max_limit+0.5)
# 设置轴的比例为 1:1
g.ax_joint.set_aspect('equal')
# 添加图例
g.ax_joint.legend()
# 调整坐标轴刻度向内移动
plt.tick_params(axis='both', which='major', direction='in', labelsize=14)
# 显示图表
#plt.show()
# 调整布局以防止标签被裁剪
```

```
plt.tight_layout()
# 保存图片
plt.savefig('验证 0604.png', dpi=300, bbox_inches='tight')  # 保存为 PNG 格式，分辨率为 300 dpi
plt.savefig('验证 0604.jpg', dpi=300, bbox_inches='tight')  # 保存为 JPG 格式，分辨率为 300 dpi
#%%
# 使用最佳模型进行预测
y_pred_normalized = best_model.predict(X_test)
#%%
import matplotlib.pyplot as plt
import matplotlib
original=testh
df = pd.DataFrame(original)
yoriginal = df.values
y1_pred = y_pred_normalized[:,500]
y1_original=testh[:, 500]
# 设置全局字体为中文字体，这里以 SimHei 为例
matplotlib.rcParams['font.family'] = 'SimHei'
# 设置图形的长宽和分辨率
plt.figure(figsize=(15, 7), dpi=300)
# 绘制曲线，增加线的粗细
plt.plot(y1_original, color='red', label='模拟值', linewidth=2)
plt.plot(y1_pred, color='blue', label='预测值', linewidth=2)
# 添加图例，修改字体大小
plt.legend(fontsize=14)
# 添加标题，修改字体大小
plt.title('预测与模拟波高对比', fontsize=16, pad=10)
# 添加坐标轴标签，修改字体大小
plt.xlabel('时间', fontsize=16)
plt.ylabel('波高/m', fontsize=16)
# 修改坐标轴刻度的字体大小
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
# 修改坐标轴粗细
plt.gca().spines['bottom'].set_linewidth(2)
plt.gca().spines['left'].set_linewidth(2)
plt.gca().spines['right'].set_linewidth(2)
plt.gca().spines['top'].set_linewidth(2)
#%%  保存为 mat
import scipy.io
# 假设您有一些数据
data = {'y_pred': y_pred}
# 保存为 .mat 文件
scipy.io.savemat('predmin.mat', data)
```

附录 2：预测模型 Python 代码

```
#这里是调用 Scikit-Learn
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense, Dropout
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.wrappers.scikit_learn import KerasRegressor
from keras import regularizers
import matplotlib.pyplot as plt
import matplotlib
from sklearn.model_selection import train_test_split
from sklearn.metrics import make_scorer
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from tensorflow.keras.optimizers import Adam
from scipy.io import loadmat
#%% 导入应用数据
x_data= loadmat('D:\python\CNN-LSTM\风场数据提取/uv20120.5')
xinput=x_data['uv']
#%% 数据归一化处理
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_test_scaled  = scaler.fit_transform(xinput)
X_test=X_test_scaled
#%% 加载最佳模型
import joblib
loaded_model = joblib.load('best_model256321.pkl')
# 使用最佳模型进行预测
import time
start_time = time.time()
y_pred= loaded_model.predict(X_test)
prediction_time = time.time() - start_time
print("模型预测所花费的时间：", prediction_time, "秒")
#%%
# 记录预测开始时间
# 使用最佳模型进行预测
y_pred = loaded_model.predict(X_test)# 计算预测所花费的时间
#%% 绘制模型应用的误差图
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score
# 假设这里 y_true 和 y_pred 是您的实际数据
h_data=loadmat('D:\python\CNN-LSTM\波浪数据提取/TQh2012.mat')
hs2012=h_data['hs']
```

```
y_true = hs2012
from matplotlib.font_manager import FontProperties
font = FontProperties(fname=r"C:\WINDOWS\Fonts\simsun.TTC",size=16.5)
mean_y_pred = np.mean(y_pred, axis=1)
mean_y_true = np.mean(y_true, axis=1)
# 计算 R^2, RMSE 和 Bias
r2 = r2_score(mean_y_true,mean_y_pred)
rmse = np.sqrt(mean_squared_error(mean_y_true,mean_y_pred))
bias = np.mean(mean_y_pred - mean_y_true)
# 创建 JointGrid 对象
g = sns.JointGrid(x=mean_y_true, y=mean_y_pred, space=0, ratio=4)
# 绘制散点图和直方图
g = g.plot_joint(plt.scatter, color="royalblue", edgecolor="white", s=50)
# 绘制直方图
g = g.plot_marginals(sns.histplot, kde=True, color="royalblue")
# 计算最佳拟合线的斜率和截距
slope, intercept = np.polyfit(mean_y_true, mean_y_pred, 1)
# 创建 x 轴的值范围
line = np.linspace(min(np.min(mean_y_true), np.min(mean_y_pred)),
max(np.max(mean_y_true), np.max(mean_y_pred)), 30000)
# 计算最佳拟合线的 y 值
best_fit_line = slope * line + intercept
# 绘制最佳拟合线
g.ax_joint.plot(line, best_fit_line, color='royalblue')
# 绘制 1:1 线
g.ax_joint.plot(line, line, color='black', linestyle='--')
# 添加注释
g.ax_joint.text(0.05, 0.95, f'$R^2$={r2:.2f}\nRMSE={rmse:.2f}m',
                        verticalalignment='top', horizontalalignment='left',
transform=g.ax_joint.transAxes, fontsize=12)
# 设置轴标签
g.ax_joint.set_xlabel('FVCOM-SWAVE 空间平均有效波高（m）',fontproperties=font)
g.ax_joint.set_ylabel('MLP 空间平均有效波高（m）',fontproperties=font)
# 设置 x 轴和 y 轴的相同范围
max_limit = max(np.max(mean_y_true), np.max(mean_y_pred))
min_limit = -0.5;
g.ax_joint.set_xlim(min_limit, max_limit+0.5)
g.ax_joint.set_ylim(min_limit, max_limit+0.5)
# 设置轴的比例为 1:1
g.ax_joint.set_aspect('equal')
# 添加图例
g.ax_joint.legend()
# 调整坐标轴刻度向内移动
plt.tick_params(axis='both', which='major', direction='in', labelsize=14)
# 显示图表
#plt.show()
# 调整布局以防止标签被裁剪
plt.tight_layout()
```

```
# 保存图片
plt.savefig('2012 应用.png', dpi=300, bbox_inches='tight')  # 保存为 PNG 格式，分辨率为
300 dpi
plt.savefig('2012 应用.jpg', dpi=300, bbox_inches='tight')  # 保存为 JPG 格式，分辨率为
300 dpi
#%% 保存为 mat
import scipy.io
# 假设您有一些数据
data = {'y_pred': y_pred}
# 保存为 .mat 文件
scipy.io.savemat('2012pred.mat', data)
```