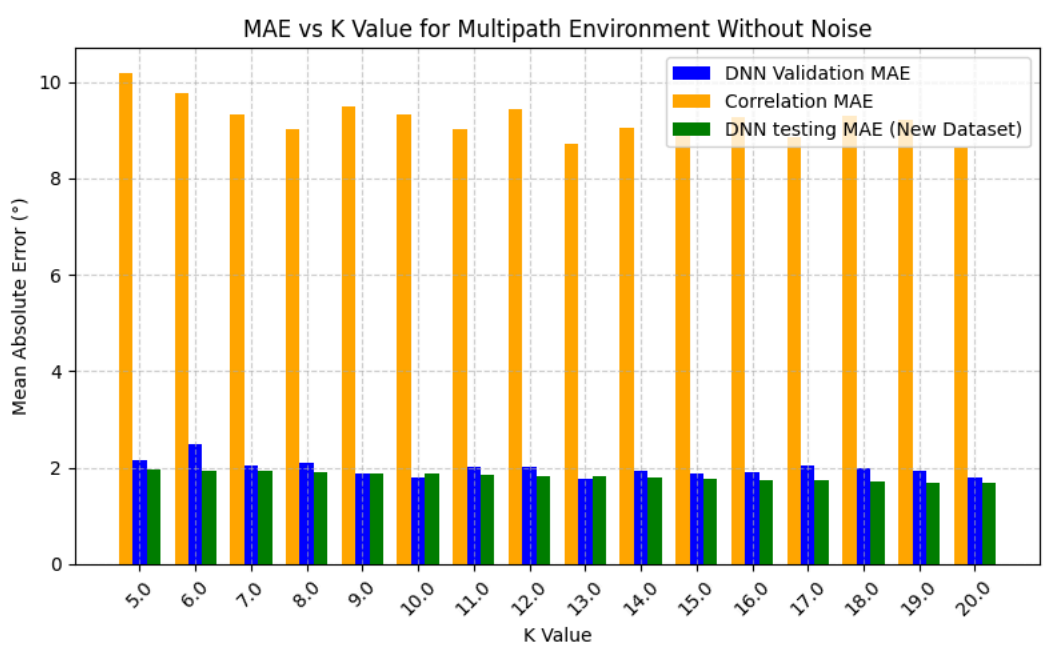
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam

# === Load Original Dataset ===
df = pd.read_csv("ris_highK_dataset_KAngleSweep_ITUrician_N16_iter20.csv")
X = df.drop(columns=["angle"]).values
y = df["angle"].values
K_values = df["K"].values

# === Normalize and Split ===
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test, _, K_test = train_test_split(
    X_scaled, y, K_values, test_size=0.2, stratify=y, random_state=42
)

# === Define and Train the Model ===
model = Sequential([
    Dense(50, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.2),
    Dense(40, activation='relu'),
    Dropout(0.2),
    Dense(30, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='linear')
])
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='mean_squared_error',
              metrics=['mae'])
model.fit(X_train, y_train, epochs=150, batch_size=32, validation_split=0.1, verbose=1)

# === Predict and Compute MAE per K (Original) ===
y_pred = model.predict(X_test).flatten()
abs_err = np.abs(y_pred - y_test)
unique_k = np.sort(np.unique(K_test))
mae_per_k = [np.mean(abs_err[K_test == k]) for k in unique_k]

# === Load External Reference CSV ===
ref_df = pd.read_csv("MAE_vs_K_20iter.csv")
ref_k = ref_df["K_Factor"]
ref_mae = ref_df["Mean_Absolute_Error"]

# === Load and Evaluate on New Dataset ===
new_df = pd.read_csv("ris_highK_test_dataset_KAngleSweep_ITUrician_N16_iter20.csv")
X_new = new_df.drop(columns=["angle"]).values
y_new = new_df["angle"].values
K_new = new_df["K"].values

X_new_scaled = scaler.transform(X_new)
y_new_pred = model.predict(X_new_scaled).flatten()
abs_err_new = np.abs(y_new_pred - y_new)
unique_k_new = np.sort(np.unique(K_new))
mae_per_k_new = [np.mean(abs_err_new[K_new == k]) for k in unique_k_new]

# === Align K values for bar plot ===
all_k = sorted(set(unique_k).union(set(ref_k)).union(set(unique_k_new)))
x = np.arange(len(all_k))

# Map MAEs to aligned K axis (fill with NaNs for missing)
def map_mae(all_k, source_k, source_mae):
    k_mae_map = dict(zip(source_k, source_mae))
    return [k_mae_map.get(k, np.nan) for k in all_k]

mae_orig = map_mae(all_k, unique_k, mae_per_k)
mae_ref = map_mae(all_k, ref_k, ref_mae)
mae_new = map_mae(all_k, unique_k_new, mae_per_k_new)
```

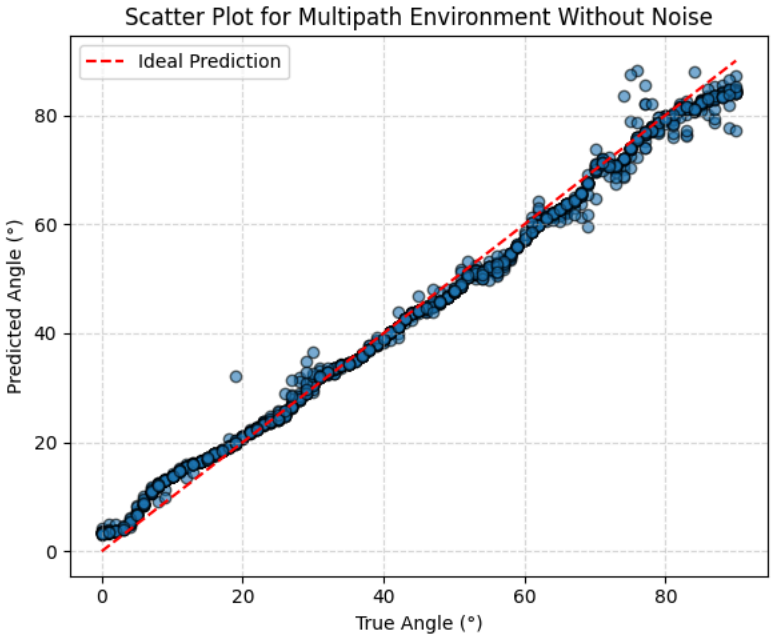
Show hidden output

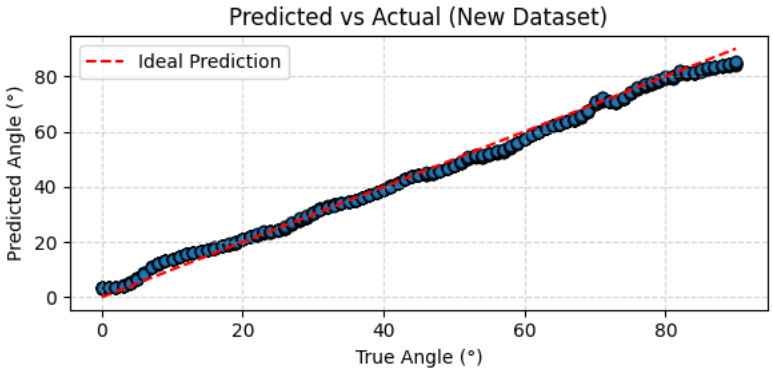
```
# === Bar Plot Settings ===
width = 0.25

plt.figure(figsize=(8, 5))
```

```
plt.bar(x , mae_orig, width, label="DNN Validation MAE", color='blue')
plt.bar(x - width,         mae_ref,  width, label="Correlation MAE", color='orange')
plt.bar(x + width, mae_new,  width, label="DNN testing MAE (New Dataset)", color='green')

plt.xticks(x, [f"{k:.1f}" for k in all_k], rotation=45)
plt.xlabel("K Value")
plt.ylabel("Mean Absolute Error (°)")
plt.title("MAE vs K Value for Multipath Environment Without Noise")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```




```
# === Scatter Plot: Original Dataset ===
plt.figure(figsize=(6, 5))
plt.scatter(y_test, y_pred, alpha=0.6, edgecolor='k')
plt.plot([0, 90], [0, 90], 'r--', label="Ideal Prediction")
plt.xlabel("True Angle (°)")
plt.ylabel("Predicted Angle (°)")
plt.title("Scatter Plot for Multipath Environment Without Noise")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()


# === Scatter Plot: New Dataset ===
plt.figure(figsize=(6, 3))
plt.scatter(y_new, y_new_pred, alpha=0.6, edgecolor='k')
plt.plot([0, 90], [0, 90], 'r--', label="Ideal Prediction")
plt.xlabel("True Angle (°)")
plt.ylabel("Predicted Angle (°)")
plt.title("Predicted vs Actual (New Dataset)")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```


Scatter Plot for Multipath Environment Without Noise
Ideal Prediction
Predicted Angle (°)
True Angle (°)
0
20
40
60
80


Predicted vs Actual (New Dataset)
Ideal Prediction
Predicted Angle (°)
True Angle (°)
0
20
40
60
80