```
import os
import numpy as np
import matplotlib.pyplot as plt
import keras
from keras import layers

# Set random seed for reproducibility
np.random.seed(42)
```

Data Loading and Preprocessing

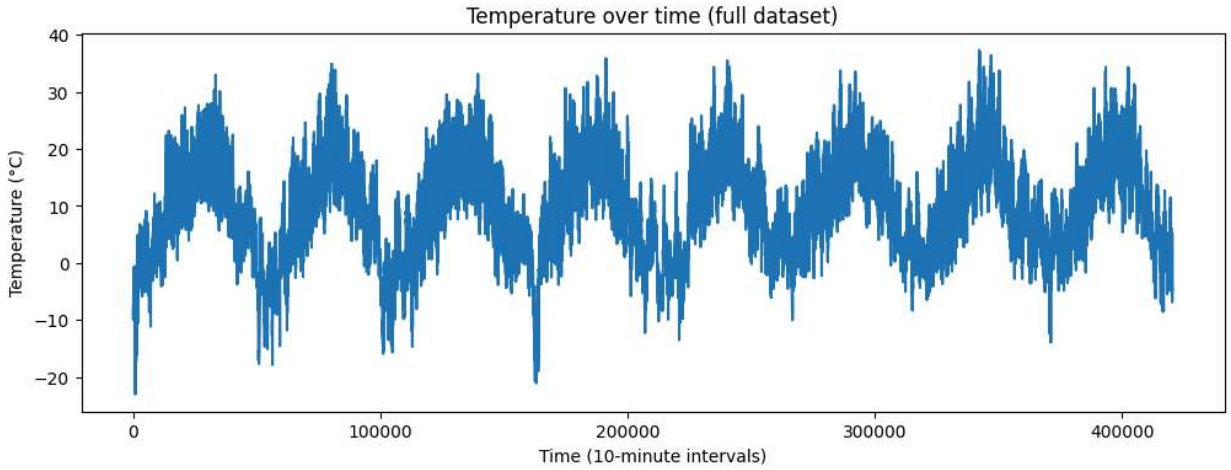
The Jena Climate dataset contains weather observations from the Max Planck Institute

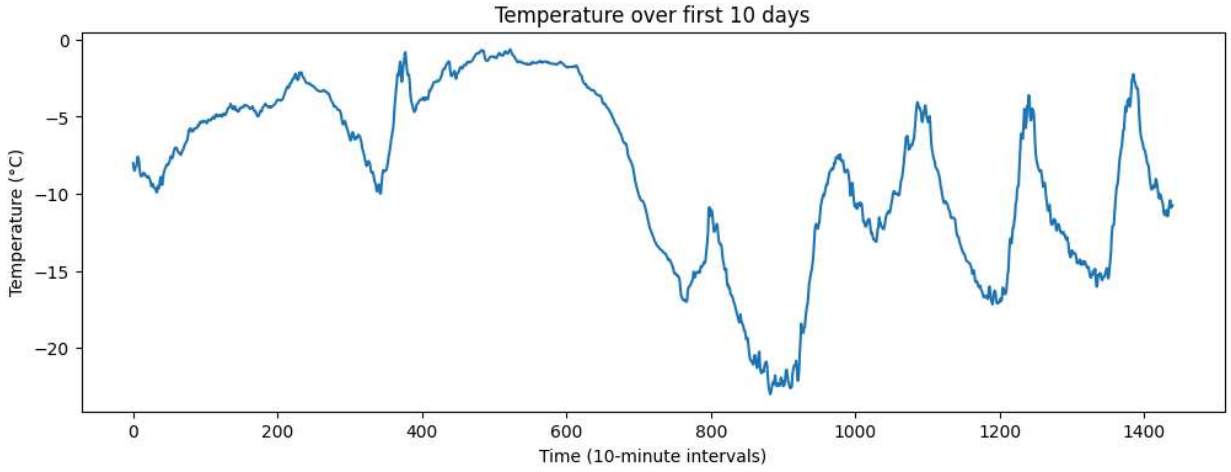
```
fname = "jena_climate_2009_2016.csv"
import urllib.request
url = "https://s3.amazonaws.com/keras-datasets/jena_climate_2009_2016.csv.zip"
urllib.request.urlretrieve(url, "jena_climate_2009_2016.csv.zip")
import zipfile
with zipfile.ZipFile("jena_climate_2009_2016.csv.zip", 'r') as zip_ref:
  zip_ref.extractall(".")

with open(fname) as f:
    data = f.read()

lines = data.split("\n")
header = lines[0].split(",")
lines = lines[1:]
print("Header:", header)
print("Number of lines:", len(lines))


temperature = np.zeros((len(lines),))
raw_data = np.zeros((len(lines), len(header) - 1))

for i, line in enumerate(lines):
    values = [float(x) for x in line.split(",")[1:]]
    temperature[i] = values[1]
    raw_data[i, :] = values[:]
```

```
Header: ['"Date Time"', '"p (mbar)"', '"T (degC)"', '"Tpot (K)"', '"Tdew (degC)"', '"rh (%)"', '"VPmax
Number of lines: 420451
```

VISUALIZING THE DATA

```
plt.figure(figsize=(12, 4))
plt.plot(range(len(temperature)), temperature)
plt.title("Temperature over time (full dataset)")
plt.xlabel("Time (10-minute intervals)")
plt.ylabel("Temperature (°C)")
plt.savefig("temperature_full.png", dpi=150, bbox_inches='tight')
plt.show()

# Plot first 10 days (1440 timesteps)
plt.figure(figsize=(12, 4))
plt.plot(range(1440), temperature[:1440])
plt.title("Temperature over first 10 days")
plt.xlabel("Time (10-minute intervals)")
plt.ylabel("Temperature (°C)")
plt.savefig("temperature_10days.png", dpi=150, bbox_inches='tight')
plt.show()
```





Preparing Data Splits

```
num_train_samples = int(0.5 * len(raw_data))
num_val_samples = int(0.25 * len(raw_data))
num_test_samples = len(raw_data) - num_train_samples - num_val_samples
```

```
print("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)


mean = raw_data[:num_train_samples].mean(axis=0)
raw_data -= mean
std = raw_data[:num_train_samples].std(axis=0)
raw_data /= std
```

```
num_train_samples: 210225
num_val_samples: 105112
num_test_samples: 105114
```

To create the Timeseries Dataset

```
sampling_rate = 6          # Sample data every hour (6 * 10 minutes)
sequence_length = 120      # 120 hours of history
delay = sampling_rate * (sequence_length + 24 - 1)  # 24 hours ahead
batch_size = 256

print(f"\nDataset parameters:")
print(f"  Sampling rate: {sampling_rate} (1 hour intervals)")
print(f"  Sequence length: {sequence_length} timesteps (5 days)")
print(f"  Delay: {delay} timesteps (24 hours ahead)")
print(f"  Batch size: {batch_size}")

# Create training dataset
train_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-delay],
    targets=temperature[delay:],
    sampling_rate=sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=batch_size,
    start_index=0,
    end_index=num_train_samples,
)

# Create validation dataset
val_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-delay],
    targets=temperature[delay:],
    sampling_rate=sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=batch_size,
    start_index=num_train_samples,
    end_index=num_train_samples + num_val_samples,
)

# Create test dataset
test_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-delay],
    targets=temperature[delay:],
    sampling_rate=sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=batch_size,
    start_index=num_train_samples + num_val_samples,
)


for samples, targets in train_dataset:
    print("\nData shapes:")
    print("  samples shape:", samples.shape)
```

```
        print("  targets shape:", targets.shape)
        break
```

```
Dataset parameters:
  Sampling rate: 6 (1 hour intervals)
  Sequence length: 120 timesteps (5 days)
  Delay: 858 timesteps (24 hours ahead)
  Batch size: 256

Data shapes:
  samples shape: (256, 120, 14)
  targets shape: (256,)
```

BASELINE: NAIVE METHOD

```
print("BASELINE: NAIVE METHOD")


def evaluate_naive_method(dataset):

    total_abs_err = 0.0
    samples_seen = 0
    for samples, targets in dataset:

        preds = samples[:, -1, 1] * std[1] + mean[1]
        total_abs_err += np.sum(np.abs(preds - targets))
        samples_seen += samples.shape[0]
    return total_abs_err / samples_seen

val_mae_naive = evaluate_naive_method(val_dataset)
test_mae_naive = evaluate_naive_method(test_dataset)

print(f"Validation MAE: {val_mae_naive:.2f}")
print(f"Test MAE: {test_mae_naive:.2f}")
```

```
BASELINE: NAIVE METHOD
Validation MAE: 2.44
Test MAE: 2.62
```

MODEL 1: BASELINE GRU

```
print("MODEL 1: BASELINE GRU (32 UNITS)")


inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.GRU(32)(inputs)
outputs = layers.Dense(1)(x)
model_gru_baseline = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_gru_baseline.keras", save_best_only=True)
]

model_gru_baseline.compile(optimizer="adam", loss="mse", metrics=["mae"])
history_gru_baseline = model_gru_baseline.fit(
    train_dataset,
    epochs=20,
    validation_data=val_dataset,
    callbacks=callbacks,
)

model_gru_baseline = keras.models.load_model("jena_gru_baseline.keras")
test_loss, test_mae = model_gru_baseline.evaluate(test_dataset)
```

```
print(f"Test MAE: {test_mae:.2f}")
```

```
MODEL 1: BASELINE GRU (32 UNITS)
Epoch 1/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 12s 13ms/step - loss: 53.1812 - mae: 5.3907 - val_loss: 10.8446 - val_mae
Epoch 2/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 10.6486 - mae: 2.5362 - val_loss: 9.8863 - val_mae:
Epoch 3/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 9.3828 - mae: 2.3864 - val_loss: 10.0397 - val_mae:
Epoch 4/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - loss: 8.6929 - mae: 2.2964 - val_loss: 9.7609 - val_mae:
Epoch 5/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - loss: 7.9343 - mae: 2.2048 - val_loss: 9.5078 - val_mae:
Epoch 6/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 7.3741 - mae: 2.1325 - val_loss: 9.8718 - val_mae:
Epoch 7/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 6.9378 - mae: 2.0696 - val_loss: 11.2727 - val_mae:
Epoch 8/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 6.6491 - mae: 2.0251 - val_loss: 10.0260 - val_mae:
Epoch 9/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 6.3247 - mae: 1.9734 - val_loss: 10.9048 - val_mae:
Epoch 10/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 6.0624 - mae: 1.9317 - val_loss: 10.5019 - val_mae:
Epoch 11/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 5.7792 - mae: 1.8847 - val_loss: 11.3407 - val_mae:
Epoch 12/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 5.5489 - mae: 1.8467 - val_loss: 11.5537 - val_mae:
Epoch 13/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 5.3443 - mae: 1.8143 - val_loss: 11.6120 - val_mae:
Epoch 14/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 5.1511 - mae: 1.7792 - val_loss: 11.2269 - val_mae:
Epoch 15/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.9266 - mae: 1.7422 - val_loss: 11.2962 - val_mae:
Epoch 16/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.7498 - mae: 1.7116 - val_loss: 12.1305 - val_mae:
Epoch 17/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.6321 - mae: 1.6881 - val_loss: 11.6064 - val_mae:
Epoch 18/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.4701 - mae: 1.6569 - val_loss: 12.3567 - val_mae:
Epoch 19/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.3254 - mae: 1.6306 - val_loss: 12.0616 - val_mae:
Epoch 20/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.2135 - mae: 1.6090 - val_loss: 12.3747 - val_mae:
405/405 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 9.9837 - mae: 2.4802
Test MAE: 2.49
```

MODEL 2: STACKED GRU (64-32)

```
print("MODEL 2: STACKED GRU (64-32 UNITS)")


inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.GRU(64, return_sequences=True)(inputs)
x = layers.Dropout(0.2)(x)
x = layers.GRU(32)(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(1)(x)
model_gru_stacked = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_gru_stacked.keras", save_best_only=True)
]

model_gru_stacked.compile(optimizer="adam", loss="mse", metrics=["mae"])
history_gru_stacked = model_gru_stacked.fit(
    train_dataset,
    epochs=20,
    validation_data=val_dataset,
```

```
    callbacks=callbacks,
)

model_gru_stacked = keras.models.load_model("jena_gru_stacked.keras")
test_loss, test_mae = model_gru_stacked.evaluate(test_dataset)
print(f"Test MAE: {test_mae:.2f}")
```

```
MODEL 2: STACKED GRU (64-32 UNITS)
Epoch 1/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 17s 17ms/step - loss: 46.0416 - mae: 4.9803 - val_loss: 10.6388 - val_mae
Epoch 2/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 11.0886 - mae: 2.5823 - val_loss: 9.5055 - val_mae:
Epoch 3/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 9.0602 - mae: 2.3502 - val_loss: 10.3672 - val_mae:
Epoch 4/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 7.3451 - mae: 2.1191 - val_loss: 11.2673 - val_mae:
Epoch 5/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 5.9870 - mae: 1.9082 - val_loss: 12.1180 - val_mae:
Epoch 6/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 5.0591 - mae: 1.7533 - val_loss: 12.6028 - val_mae:
Epoch 7/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 4.4897 - mae: 1.6527 - val_loss: 12.7488 - val_mae:
Epoch 8/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 4.0747 - mae: 1.5668 - val_loss: 12.8461 - val_mae:
Epoch 9/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 3.7770 - mae: 1.5041 - val_loss: 12.8189 - val_mae:
Epoch 10/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 3.4942 - mae: 1.4464 - val_loss: 12.9782 - val_mae:
Epoch 11/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 3.2883 - mae: 1.4009 - val_loss: 13.0224 - val_mae:
Epoch 12/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 3.1502 - mae: 1.3731 - val_loss: 13.1167 - val_mae:
Epoch 13/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 2.9988 - mae: 1.3396 - val_loss: 13.1167 - val_mae:
Epoch 14/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 2.8626 - mae: 1.3069 - val_loss: 13.3138 - val_mae:
Epoch 15/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 2.7371 - mae: 1.2750 - val_loss: 13.0226 - val_mae:
Epoch 16/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 2.6166 - mae: 1.2496 - val_loss: 12.7297 - val_mae:
Epoch 17/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 2.5613 - mae: 1.2323 - val_loss: 12.8649 - val_mae:
Epoch 18/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 2.4807 - mae: 1.2135 - val_loss: 12.7057 - val_mae:
Epoch 19/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 2.4211 - mae: 1.1974 - val_loss: 12.9111 - val_mae:
Epoch 20/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 14s 17ms/step - loss: 2.3489 - mae: 1.1780 - val_loss: 13.1402 - val_mae:
405/405 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 10.5172 - mae: 2.5283
Test MAE: 2.54
```

MODEL 3: DEEPER STACKED GRU (128-64-32)

```
print("MODEL 3: DEEPER STACKED GRU (128-64-32 UNITS)")


inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.GRU(128, return_sequences=True)(inputs)
x = layers.Dropout(0.2)(x)
x = layers.GRU(64, return_sequences=True)(x)
x = layers.Dropout(0.2)(x)
x = layers.GRU(32)(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(1)(x)
model_gru_deep = keras.Model(inputs, outputs)

callbacks = [
```

```
    keras.callbacks.ModelCheckpoint("jena_gru_deep.keras", save_best_only=True)
]

model_gru_deep.compile(optimizer="adam", loss="mse", metrics=["mae"])
history_gru_deep = model_gru_deep.fit(
    train_dataset,
    epochs=20,
    validation_data=val_dataset,
    callbacks=callbacks,
)

model_gru_deep = keras.models.load_model("jena_gru_deep.keras")
test_loss, test_mae = model_gru_deep.evaluate(test_dataset)
print(f"Test MAE: {test_mae:.2f}")
```

```
MODEL 3: DEEPER STACKED GRU (128-64-32 UNITS)
Epoch 1/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 21s 23ms/step - loss: 38.8986 - mae: 4.5357 - val_loss: 10.9411 - val_mae
Epoch 2/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 9.6863 - mae: 2.4041 - val_loss: 11.7215 - val_mae:
Epoch 3/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 6.1552 - mae: 1.9131 - val_loss: 12.8140 - val_mae:
Epoch 4/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 4.3705 - mae: 1.6057 - val_loss: 12.7170 - val_mae:
Epoch 5/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 3.5086 - mae: 1.4312 - val_loss: 12.5007 - val_mae:
Epoch 6/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 3.0865 - mae: 1.3447 - val_loss: 12.5921 - val_mae:
Epoch 7/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 2.7080 - mae: 1.2566 - val_loss: 12.7005 - val_mae:
Epoch 8/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 2.5359 - mae: 1.2120 - val_loss: 13.1175 - val_mae:
Epoch 9/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 2.3802 - mae: 1.1696 - val_loss: 12.4816 - val_mae:
Epoch 10/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 2.2165 - mae: 1.1265 - val_loss: 12.4130 - val_mae:
Epoch 11/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 2.1424 - mae: 1.1019 - val_loss: 12.5404 - val_mae:
Epoch 12/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 2.0240 - mae: 1.0716 - val_loss: 12.6523 - val_mae:
Epoch 13/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 1.9216 - mae: 1.0435 - val_loss: 12.5026 - val_mae:
Epoch 14/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 1.8606 - mae: 1.0262 - val_loss: 12.3585 - val_mae:
Epoch 15/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 1.8043 - mae: 1.0087 - val_loss: 12.3507 - val_mae:
Epoch 16/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 1.7344 - mae: 0.9861 - val_loss: 12.3163 - val_mae:
Epoch 17/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 1.6767 - mae: 0.9670 - val_loss: 12.4080 - val_mae:
Epoch 18/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 1.6189 - mae: 0.9516 - val_loss: 12.0700 - val_mae:
Epoch 19/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 19s 23ms/step - loss: 1.5979 - mae: 0.9447 - val_loss: 12.1361 - val_mae:
Epoch 20/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 18s 22ms/step - loss: 1.5299 - mae: 0.9270 - val_loss: 12.1817 - val_mae:
405/405 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 12.6850 - mae: 2.7212
Test MAE: 2.73
```

MODEL 4: LSTM (64 UNITS)

```
print("MODEL 4: LSTM (64 UNITS)")


inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(64)(inputs)
outputs = layers.Dense(1)(x)
model_lstm = keras.Model(inputs, outputs)
```

```
callbacks = [
    keras.callbacks.ModelCheckpoint("jena_lstm.keras", save_best_only=True)
]

model_lstm.compile(optimizer="adam", loss="mse", metrics=["mae"])
history_lstm = model_lstm.fit(
    train_dataset,
    epochs=20,
    validation_data=val_dataset,
    callbacks=callbacks,
)

model_lstm = keras.models.load_model("jena_lstm.keras")
test_loss, test_mae = model_lstm.evaluate(test_dataset)
print(f"Test MAE: {test_mae:.2f}")
```

```
MODEL 4: LSTM (64 UNITS)
Epoch 1/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 12s 13ms/step - loss: 34.9382 - mae: 4.2277 - val_loss: 9.8526 - val_mae:
Epoch 2/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 12s 14ms/step - loss: 8.8692 - mae: 2.3230 - val_loss: 10.5475 - val_mae:
Epoch 3/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 7.0824 - mae: 2.0687 - val_loss: 11.2091 - val_mae:
Epoch 4/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 5.7141 - mae: 1.8511 - val_loss: 11.2757 - val_mae:
Epoch 5/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 4.9251 - mae: 1.7127 - val_loss: 11.9178 - val_mae:
Epoch 6/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 4.1885 - mae: 1.5781 - val_loss: 12.2260 - val_mae:
Epoch 7/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 3.5245 - mae: 1.4491 - val_loss: 12.8809 - val_mae:
Epoch 8/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 3.1629 - mae: 1.3700 - val_loss: 13.1057 - val_mae:
Epoch 9/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 2.6482 - mae: 1.2611 - val_loss: 13.4510 - val_mae:
Epoch 10/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 2.4135 - mae: 1.2040 - val_loss: 13.7082 - val_mae:
Epoch 11/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 2.5080 - mae: 1.2244 - val_loss: 13.6770 - val_mae:
Epoch 12/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 2.0750 - mae: 1.1160 - val_loss: 13.6920 - val_mae:
Epoch 13/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 1.9254 - mae: 1.0757 - val_loss: 14.0526 - val_mae:
Epoch 14/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 1.8251 - mae: 1.0473 - val_loss: 14.1841 - val_mae:
Epoch 15/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 1.8787 - mae: 1.0481 - val_loss: 14.0399 - val_mae:
Epoch 16/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 1.5981 - mae: 0.9825 - val_loss: 14.1689 - val_mae:
Epoch 17/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 1.5523 - mae: 0.9617 - val_loss: 13.9940 - val_mae:
Epoch 18/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 1.4841 - mae: 0.9426 - val_loss: 14.3124 - val_mae:
Epoch 19/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 1.4973 - mae: 0.9465 - val_loss: 14.2956 - val_mae:
Epoch 20/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 1.2942 - mae: 0.8795 - val_loss: 14.3146 - val_mae:
405/405 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - loss: 11.0449 - mae: 2.6089
Test MAE: 2.62
```

MODEL 5: STACKED LSTM (64-32)

```
print("MODEL 5: STACKED LSTM (64-32 UNITS)")


inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.LSTM(64, return_sequences=True)(inputs)
x = layers.Dropout(0.2)(x)
x = layers.LSTM(32)(x)
```

```
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(1)(x)
model_lstm_stacked = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_lstm_stacked.keras", save_best_only=True)
]

model_lstm_stacked.compile(optimizer="adam", loss="mse", metrics=["mae"])
history_lstm_stacked = model_lstm_stacked.fit(
    train_dataset,
    epochs=20,
    validation_data=val_dataset,
    callbacks=callbacks,
)

model_lstm_stacked = keras.models.load_model("jena_lstm_stacked.keras")
test_loss, test_mae = model_lstm_stacked.evaluate(test_dataset)
print(f"Test MAE: {test_mae:.2f}")
```

```
MODEL 5: STACKED LSTM (64-32 UNITS)
Epoch 1/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 17s 18ms/step - loss: 44.1197 - mae: 4.8623 - val_loss: 11.6450 - val_mae
Epoch 2/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 10.2705 - mae: 2.4704 - val_loss: 11.5143 - val_mae
Epoch 3/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 7.3923 - mae: 2.0881 - val_loss: 13.0455 - val_mae:
Epoch 4/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 5.7564 - mae: 1.8440 - val_loss: 13.6415 - val_mae:
Epoch 5/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 4.8919 - mae: 1.6956 - val_loss: 13.4537 - val_mae:
Epoch 6/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 4.2586 - mae: 1.5851 - val_loss: 13.8223 - val_mae:
Epoch 7/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 3.8622 - mae: 1.5063 - val_loss: 14.1842 - val_mae:
Epoch 8/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 3.5827 - mae: 1.4515 - val_loss: 14.1621 - val_mae:
Epoch 9/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 3.2571 - mae: 1.3833 - val_loss: 13.9022 - val_mae:
Epoch 10/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 3.0499 - mae: 1.3386 - val_loss: 14.5295 - val_mae:
Epoch 11/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.9623 - mae: 1.3166 - val_loss: 14.6542 - val_mae:
Epoch 12/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.7656 - mae: 1.2688 - val_loss: 13.8066 - val_mae:
Epoch 13/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.6389 - mae: 1.2406 - val_loss: 14.3625 - val_mae:
Epoch 14/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.5437 - mae: 1.2160 - val_loss: 14.2687 - val_mae:
Epoch 15/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.4567 - mae: 1.1933 - val_loss: 15.1857 - val_mae:
Epoch 16/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.4009 - mae: 1.1796 - val_loss: 15.7827 - val_mae:
Epoch 17/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.2988 - mae: 1.1546 - val_loss: 14.4754 - val_mae:
Epoch 18/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.2537 - mae: 1.1414 - val_loss: 14.9156 - val_mae:
Epoch 19/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.1519 - mae: 1.1171 - val_loss: 16.3433 - val_mae:
Epoch 20/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 15s 18ms/step - loss: 2.1275 - mae: 1.1076 - val_loss: 14.8573 - val_mae:
405/405 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 13.1293 - mae: 2.8339
Test MAE: 2.84
```

MODEL 6: CONV1D + GRU

```
print("MODEL 6: CONV1D + GRU")
```

```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.Conv1D(32, 5, activation="relu")(inputs)
x = layers.MaxPooling1D(3)(x)
x = layers.Conv1D(32, 5, activation="relu")(x)
x = layers.GRU(32)(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(1)(x)
model_conv_gru = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_conv_gru.keras", save_best_only=True)
]

model_conv_gru.compile(optimizer="adam", loss="mse", metrics=["mae"])
history_conv_gru = model_conv_gru.fit(
    train_dataset,
    epochs=20,
    validation_data=val_dataset,
    callbacks=callbacks,
)

model_conv_gru = keras.models.load_model("jena_conv_gru.keras")
test_loss, test_mae = model_conv_gru.evaluate(test_dataset)
print(f"Test MAE: {test_mae:.2f}")
```

```
MODEL 6: CONV1D + GRU
Epoch 1/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 12s 13ms/step - loss: 43.9523 - mae: 4.8849 - val_loss: 12.1774 - val_mae
Epoch 2/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 12.2778 - mae: 2.7272 - val_loss: 11.1131 - val_mae
Epoch 3/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 9.9505 - mae: 2.4599 - val_loss: 11.9965 - val_mae:
Epoch 4/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 8.3156 - mae: 2.2404 - val_loss: 13.4547 - val_mae:
Epoch 5/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 7.0968 - mae: 2.0617 - val_loss: 14.1758 - val_mae:
Epoch 6/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 6.2985 - mae: 1.9419 - val_loss: 14.7671 - val_mae:
Epoch 7/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 5.8117 - mae: 1.8636 - val_loss: 13.9424 - val_mae:
Epoch 8/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 5.3409 - mae: 1.7863 - val_loss: 14.6413 - val_mae:
Epoch 9/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.9585 - mae: 1.7221 - val_loss: 14.3239 - val_mae:
Epoch 10/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.7182 - mae: 1.6837 - val_loss: 14.7982 - val_mae:
Epoch 11/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.4190 - mae: 1.6288 - val_loss: 14.8779 - val_mae:
Epoch 12/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.2234 - mae: 1.5908 - val_loss: 14.3715 - val_mae:
Epoch 13/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.0625 - mae: 1.5600 - val_loss: 14.5975 - val_mae:
Epoch 14/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.9327 - mae: 1.5333 - val_loss: 15.2901 - val_mae:
Epoch 15/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.8030 - mae: 1.5087 - val_loss: 15.6749 - val_mae:
Epoch 16/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.6658 - mae: 1.4832 - val_loss: 14.9470 - val_mae:
Epoch 17/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.6011 - mae: 1.4676 - val_loss: 14.9738 - val_mae:
Epoch 18/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.4698 - mae: 1.4416 - val_loss: 15.5804 - val_mae:
Epoch 19/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.4109 - mae: 1.4264 - val_loss: 14.9298 - val_mae:
Epoch 20/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.3223 - mae: 1.4110 - val_loss: 15.5077 - val_mae:
405/405 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - loss: 12.2075 - mae: 2.7374
Test MAE: 2.75
```

MODEL 7: CONV1D + LSTM

```
print("MODEL 7: CONV1D + LSTM")


inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.Conv1D(32, 5, activation="relu")(inputs)
x = layers.MaxPooling1D(3)(x)
x = layers.Conv1D(32, 5, activation="relu")(x)
x = layers.LSTM(32)(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(1)(x)
model_conv_lstm = keras.Model(inputs, outputs)

callbacks = [
    keras.callbacks.ModelCheckpoint("jena_conv_lstm.keras", save_best_only=True)
]

model_conv_lstm.compile(optimizer="adam", loss="mse", metrics=["mae"])
history_conv_lstm = model_conv_lstm.fit(
    train_dataset,
    epochs=20,
    validation_data=val_dataset,
    callbacks=callbacks,
)

model_conv_lstm = keras.models.load_model("jena_conv_lstm.keras")
test_loss, test_mae = model_conv_lstm.evaluate(test_dataset)
print(f"Test MAE: {test_mae:.2f}")
```

```
MODEL 7: CONV1D + LSTM
Epoch 1/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 12s 12ms/step - loss: 48.0173 - mae: 5.1112 - val_loss: 12.5664 - val_mae
Epoch 2/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 12.1286 - mae: 2.7056 - val_loss: 11.3850 - val_mae
Epoch 3/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 9.8785 - mae: 2.4456 - val_loss: 12.6658 - val_mae:
Epoch 4/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 8.2587 - mae: 2.2253 - val_loss: 13.2254 - val_mae:
Epoch 5/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 7.2065 - mae: 2.0652 - val_loss: 13.9235 - val_mae:
Epoch 6/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 6.3537 - mae: 1.9397 - val_loss: 13.6243 - val_mae:
Epoch 7/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 5.7478 - mae: 1.8435 - val_loss: 14.3850 - val_mae:
Epoch 8/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 5.2665 - mae: 1.7627 - val_loss: 14.9610 - val_mae:
Epoch 9/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.9251 - mae: 1.7064 - val_loss: 14.5101 - val_mae:
Epoch 10/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.5873 - mae: 1.6461 - val_loss: 15.4485 - val_mae:
Epoch 11/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.3822 - mae: 1.6070 - val_loss: 15.0914 - val_mae:
Epoch 12/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.2007 - mae: 1.5769 - val_loss: 15.3940 - val_mae:
Epoch 13/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 4.0006 - mae: 1.5389 - val_loss: 15.2776 - val_mae:
Epoch 14/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.8436 - mae: 1.5051 - val_loss: 15.2259 - val_mae:
Epoch 15/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.6778 - mae: 1.4728 - val_loss: 15.7400 - val_mae:
Epoch 16/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.6265 - mae: 1.4618 - val_loss: 15.8320 - val_mae:
Epoch 17/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.5305 - mae: 1.4409 - val_loss: 15.9071 - val_mae:
Epoch 18/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.4062 - mae: 1.4179 - val_loss: 15.6281 - val_mae:
Epoch 19/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.3713 - mae: 1.4068 - val_loss: 15.8984 - val_mae:
Epoch 20/20
819/819 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - loss: 3.2386 - mae: 1.3812 - val_loss: 16.0527 - val_mae:
405/405 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - loss: 12.8277 - mae: 2.8154
Test MAE: 2.83
```

Collecting all Results in one cell

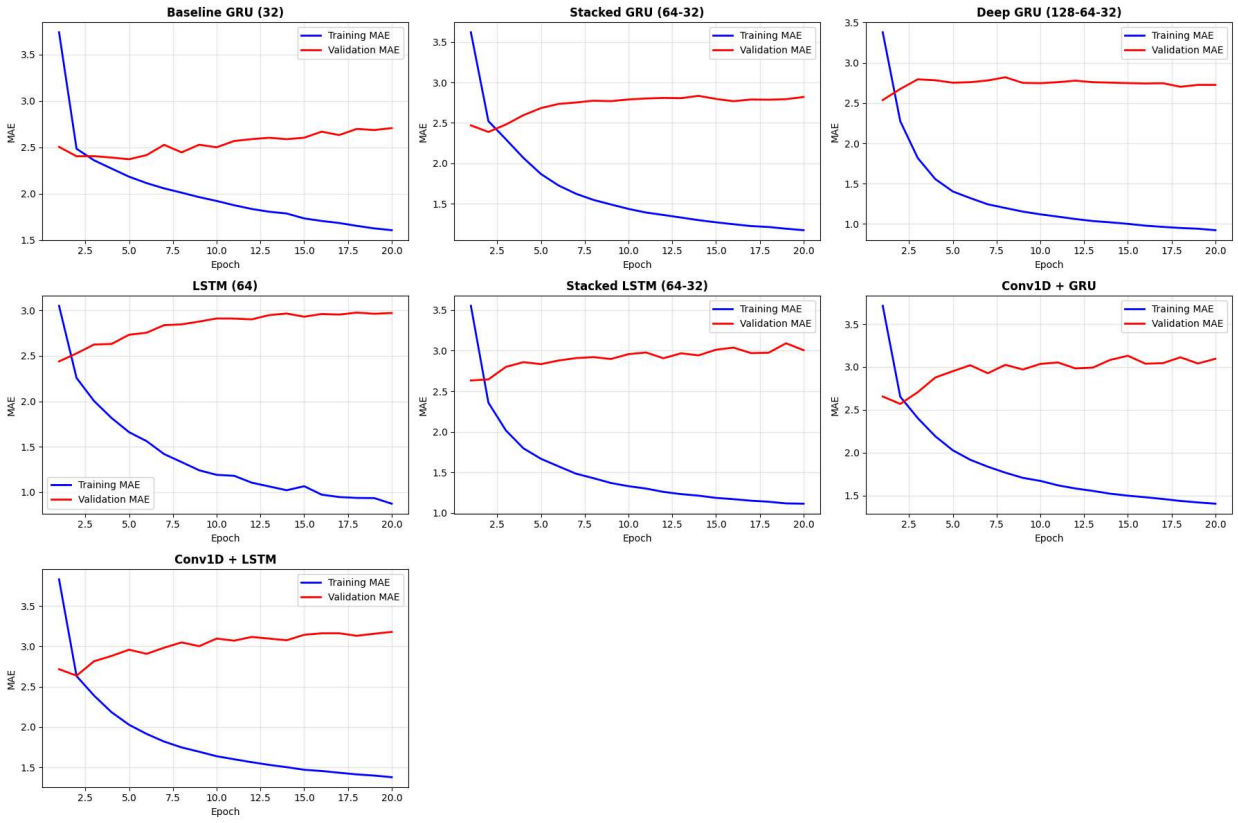
```
print("FINAL RESULTS SUMMARY")



results = {
    "Naive Baseline": test_mae_naive,
    "Baseline GRU (32)": model_gru_baseline.evaluate(test_dataset)[1],
    "Stacked GRU (64-32)": model_gru_stacked.evaluate(test_dataset)[1],
    "Deep GRU (128-64-32)": model_gru_deep.evaluate(test_dataset)[1],
    "LSTM (64)": model_lstm.evaluate(test_dataset)[1],
    "Stacked LSTM (64-32)": model_lstm_stacked.evaluate(test_dataset)[1],
    "Conv1D + GRU": model_conv_gru.evaluate(test_dataset)[1],
    "Conv1D + LSTM": model_conv_lstm.evaluate(test_dataset)[1],
}


print("\nModel Performance (Test MAE in °C):")
print("-" * 50)
for model_name, mae in sorted(results.items(), key=lambda x: x[1]):
    print(f"{model_name:30s}: {mae:6.2f}°C")

best_model_name = min(results, key=results.get)
best_mae = results[best_model_name]
print("\n" + "="*60)
print(f"BEST MODEL: {best_model_name}")
print(f"Test MAE: {best_mae:.2f}°C")
print("="*60)
```

```
FINAL RESULTS SUMMARY
405/405 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - loss: 10.0012 - mae: 2.4834
405/405 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 10.5056 - mae: 2.5275
405/405 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 12.6843 - mae: 2.7208
405/405 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 11.0633 - mae: 2.6118
405/405 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - loss: 13.1388 - mae: 2.8353
405/405 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - loss: 12.2243 - mae: 2.7402
405/405 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - loss: 12.8422 - mae: 2.8166

Model Performance (Test MAE in °C):
--------------------------------------------------
Baseline GRU (32)             :   2.49°C
Stacked GRU (64-32)           :   2.54°C
LSTM (64)                     :   2.62°C
Naive Baseline                :   2.62°C
Deep GRU (128-64-32)          :   2.73°C
Conv1D + GRU                  :   2.75°C
Conv1D + LSTM                 :   2.83°C
Stacked LSTM (64-32)          :   2.84°C

============================================================
BEST MODEL: Baseline GRU (32)
Test MAE: 2.49°C
============================================================
```

Plotting

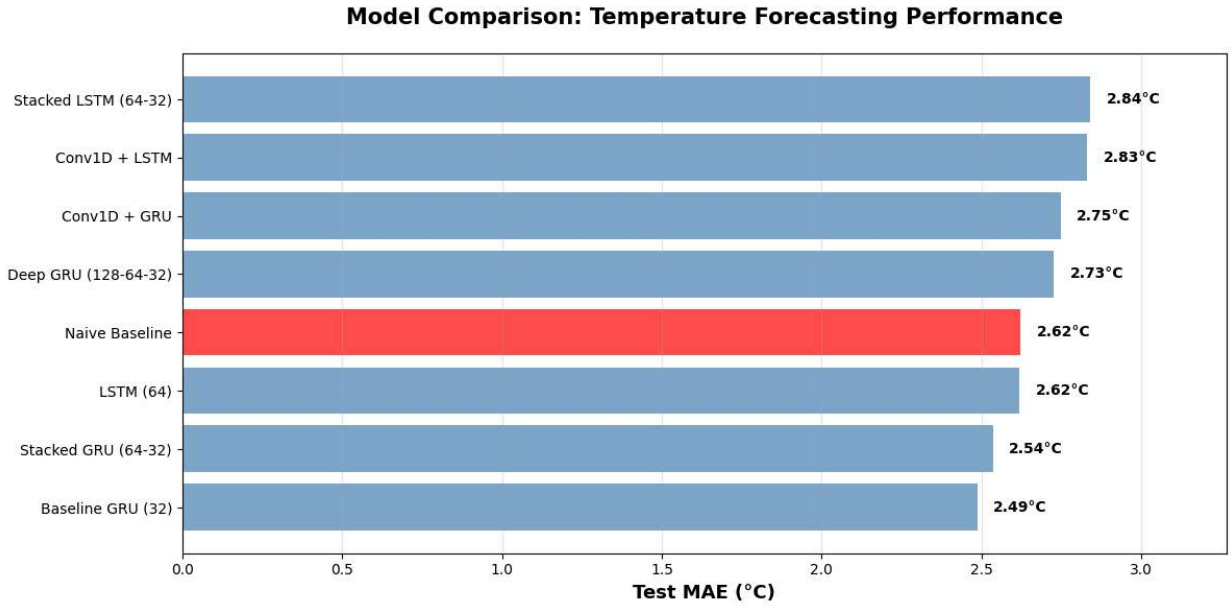
```
histories = [
    (history_gru_baseline, "Baseline GRU (32)"),
    (history_gru_stacked, "Stacked GRU (64-32)"),
    (history_gru_deep, "Deep GRU (128-64-32)"),
    (history_lstm, "LSTM (64)"),
    (history_lstm_stacked, "Stacked LSTM (64-32)"),
```

```
    (history_conv_gru, "Conv1D + GRU"),
    (history_conv_lstm, "Conv1D + LSTM"),
]

fig, axes = plt.subplots(3, 3, figsize=(18, 12))
axes = axes.flatten()

for idx, (history, name) in enumerate(histories):
    ax = axes[idx]

    loss = history.history["mae"]
    val_loss = history.history["val_mae"]
    epochs = range(1, len(loss) + 1)

    ax.plot(epochs, loss, "b-", label="Training MAE", linewidth=2)
    ax.plot(epochs, val_loss, "r-", label="Validation MAE", linewidth=2)
    ax.set_title(name, fontsize=12, fontweight='bold')
    ax.set_xlabel("Epoch")
    ax.set_ylabel("MAE")
    ax.legend()
    ax.grid(True, alpha=0.3)


for idx in range(len(histories), len(axes)):
    axes[idx].axis('off')

plt.tight_layout()
plt.savefig("training_histories.png", dpi=200, bbox_inches='tight')
plt.show()
```



Result Comparison

```
fig, ax = plt.subplots(figsize=(12, 6))

models = list(results.keys())
maes = list(results.values())

# Sort by MAE
sorted_pairs = sorted(zip(models, maes), key=lambda x: x[1])
models_sorted = [x[0] for x in sorted_pairs]
maes_sorted = [x[1] for x in sorted_pairs]

colors = ['red' if 'Naive' in m else 'steelblue' for m in models_sorted]
bars = ax.barh(models_sorted, maes_sorted, color=colors, alpha=0.7)

# Add value labels
for i, (bar, mae) in enumerate(zip(bars, maes_sorted)):
    ax.text(mae + 0.05, i, f'{mae:.2f}°C', va='center', fontsize=10, fontweight='bold')

ax.set_xlabel("Test MAE (°C)", fontsize=13, fontweight='bold')
ax.set_title("Model Comparison: Temperature Forecasting Performance",
             fontsize=15, fontweight='bold', pad=20)
ax.grid(axis='x', alpha=0.3)
ax.set_xlim(0, max(maes_sorted) * 1.15)

plt.tight_layout()
plt.savefig("results_comparison.png", dpi=200, bbox_inches='tight')
plt.show()
```




```
import pandas as pd

results_df = pd.DataFrame([
    {"Model": name, "Test_MAE": mae}
```

```
        for name, mae in results.items()
    ]).sort_values("Test_MAE")

    results_df.to_csv("model_results.csv", index=False)
```