In [1]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from tensorflow import keras
from sklearn.metrics import mean_squared_error, mean_absolute_error
from math import ceil
from scipy.stats import pearsonr
```

```
C:\Users\18072\anaconda3\envs\moiome\lib\site-packages\requests\__init__.py:86: RequestsDependencyWarning: Unable to find accep
table character detection dependency (chardet or charset_normalizer).
  warnings.warn(
```

This notebooks creates the visualisations of a part of the time series of CPU usage (y-axis) across time (x-axis) for job 113, as shown in the Figure 2 of the paper, that allows to better understand the ML-based predictions.The first figure compares the predicted time series to the ground truth. Each box depicts different parts of the time series, where the predicted one is derived following the three different experiments described in the section 2.1 of the paper, using the trained model of job 113 for experiments A, B and job 917 for experiment C. The number on the upper right of each box shows the Pearson correlation coefficient value between the two depicted time series.

In [3]:
```
job_labels = ['113', '917']
job_ids = ['113812204462', '91724979887']
```

In [5]:
```
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence) - 1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X), np.array(y)
```

In [7]:
```
left = [] # 113 0
center = [] # 113 3
right = [] # 917 0
```

In [103…
```
for ind, job_id in enumerate(job_ids):
    job_label = job_labels[ind]
    columns_to_use = ['avg_cpu_usage', 'start_time']
    time_col = 'start_time'
    col = 'avg_cpu_usage'
    time_unit = 'us'
    freq = '5min'

    # folder path
    dir_path = r'input-large/' + job_id + '/'

    # list to store files
    list_of_files = []

    # Iterate directoryf
    for path in os.listdir(dir_path):
        # check if current path is a file
        if os.path.isfile(os.path.join(dir_path, path)):
            list_of_files.append(path)

    csv_data_list = []
    for i in list_of_files:
        type(i)
        dir = 'input-large/' +job_id + '/' + i
        csv_data_list.append(pd.read_csv(filepath_or_buffer=dir, usecols=columns_to_use))

    data = []
    for df in csv_data_list:
        df[time_col] = pd.to_datetime(
        df[time_col], unit=time_unit)
        df.set_index(time_col, inplace=True)
        df = df[~df.index.duplicated(keep="first")]
        df = df.resample(freq).mean()
        df = df.asfreq(freq=freq, method="ffill")
        df = df.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
        data.append(df)
```

```
for mod_index, model_id in enumerate(job_ids):
    model_label = job_labels[mod_index]
    number_of_tasks = len(list_of_files)
    for i in range (0, number_of_tasks):
        x_test = data[i][:800]['avg_cpu_usage']
        n_features = 1
        n_steps = 30

        a = min(x_test)
        b = max(x_test)
        x_test_normalised = []
        for x in x_test:
            x_test_normalised.append(((x-a)/(b-a)))

        # split into samples
        X_test, y_test = split_sequence(x_test_normalised, n_steps)
        X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], n_features))

        model = keras.models.load_model('pretrained_lstm_models/job' + model_label + '/newbaseline_model/model') #LSTM moa
        m = model
        yh = m.predict(X_test, verbose=0)

        y_denorm = []
        for y in yh:
            y_denorm.append(y*((b)-(a)) + (a))

        if i == 0 and model_id == '113812204462' and job_id == '113812204462':
            print("LSTM & Non-ML correlation")
            job5 = np.concatenate(y_denorm[0:100], axis=0)
            cor1prev = pearsonr(job5,x_test[29:129])[0]
            print(pearsonr(job5,x_test[29:129])[0])
            print("LSTM & ground truth correlation")
            cor1gt = pearsonr(job5,x_test[30:130])[0]
            print(pearsonr(job5,x_test[30:130])[0])
            left.append(y_denorm[0:100])
            left.append(x_test[29:129])
            left.append(x_test[30:130])
        if i == 0 and model_id == '91724979887' and job_id == '113812204462':
            print("LSTM & Non-ML correlation")
            job5 = np.concatenate(y_denorm[0:100], axis=0)
```

```
                cor2prev = pearsonr(job5,x_test[29:129])[0]
                print(pearsonr(job5,x_test[29:129])[0])
                print("LSTM & ground truth correlation")
                cor2gt = pearsonr(job5,x_test[30:130])[0]
                print(pearsonr(job5,x_test[30:130])[0])
                right.append(y_denorm[0:100])
                right.append(x_test[29:129])
                right.append(x_test[30:130])
```

```
LSTM & Non-ML correlation
0.997338713422377
LSTM & ground truth correlation
0.9792816610190345
LSTM & Non-ML correlation
0.9959973874542003
LSTM & ground truth correlation
0.978783424235379
```

In [97]:
```
left[2]
```

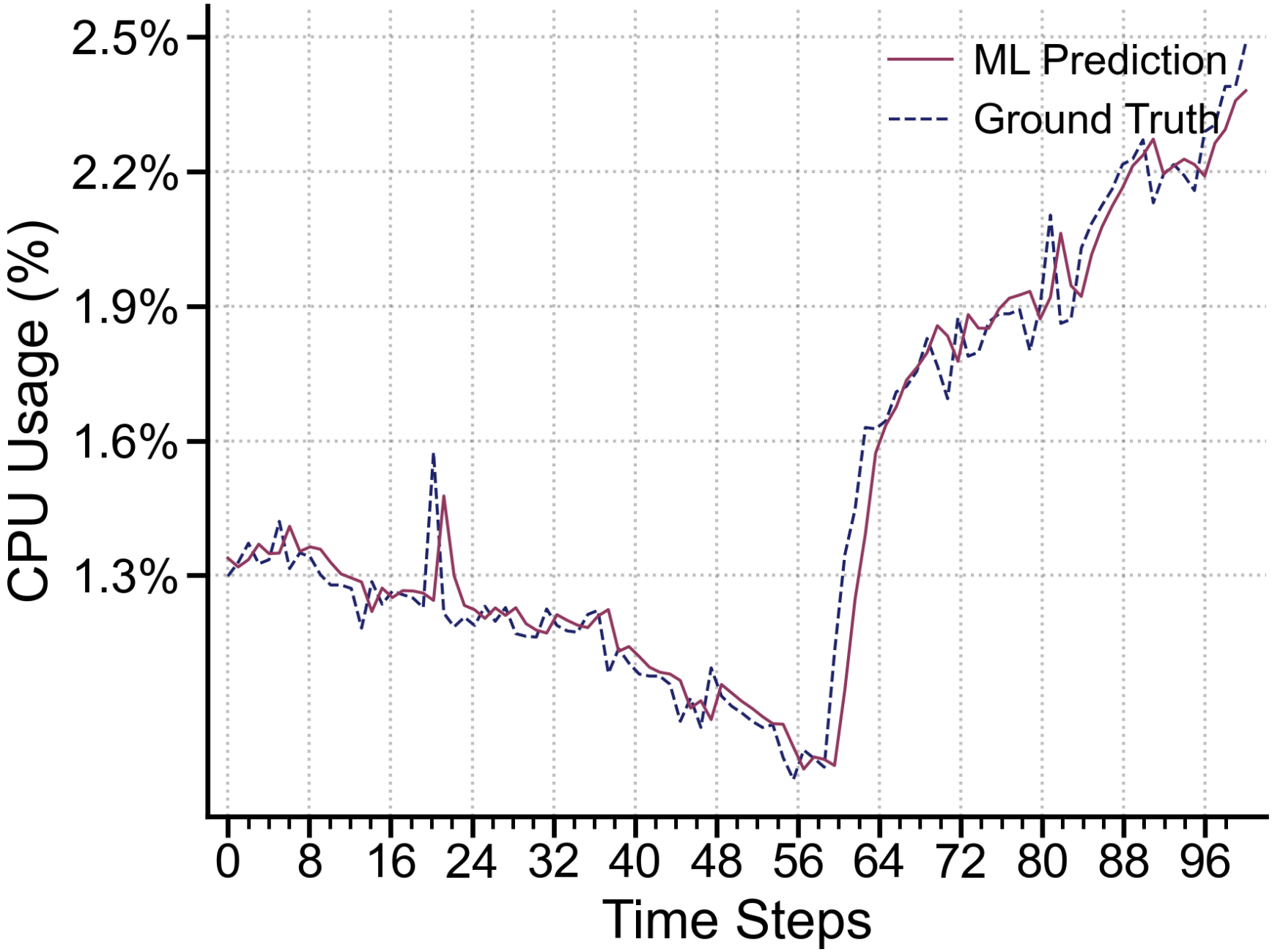
Out[97]:
```
30     0.012985
31     0.013290
32     0.013718
33     0.013260
34     0.013351
          ...
125    0.022888
126    0.023041
127    0.023895
128    0.023895
129    0.024872
Name: avg_cpu_usage, Length: 100, dtype: float64
```

Figure 2a.

In [121…
```
import matplotlib.pyplot as plt
import numpy as np

# 基础参数设置
lw = 0.5
plt.rcParams['font.family'] = 'Arial'  # 设置全局字体
```

```
# 创建图形对象
fig, ax = plt.subplots(figsize=(3.5, 2.5), dpi=600)
fig.subplots_adjust(left=0.15, right=0.85, top=0.9, bottom=0.15)

# 生成坐标轴数据
bins = np.linspace(0, 100, 100)

# 绘制双曲线（假设left数据结构不变）
ax.plot(bins, left[0][:100],
        alpha=1,
        color="#923b62",
        label='ML Prediction',
        linewidth=lw,
        zorder=3)

ax.plot(bins, left[2].iloc[:100],
        alpha=1,
        color="#1e256c",
        label='Ground Truth',
        linewidth=lw,
        linestyle='--',
        zorder=2)

# ========== 关键修改部分 ==========
# 原始数值到百分比的转换（0.013 → 1.3%）
y_ticks = [0.013, 0.016, 0.019, 0.022, 0.025]
y_tick_labels = [f"{y*100:.1f}%" for y in y_ticks]  # 正确转换公式

ax.set_ylabel("CPU Usage (%)", size=9, labelpad=2)  # 添加百分比单位
ax.set_yticks(y_ticks)
ax.set_yticklabels(y_tick_labels)
# ================================

# X轴设置
ax.set_xlabel('Time Steps', fontsize=9, labelpad=2)
ax.set_xticks(np.arange(0, 100, 8))
ax.set_xticks(np.arange(0, 100, 2), minor=True)

# 样式优化
ax.tick_params(axis='both', which='major', labelsize=8, pad=1)
```

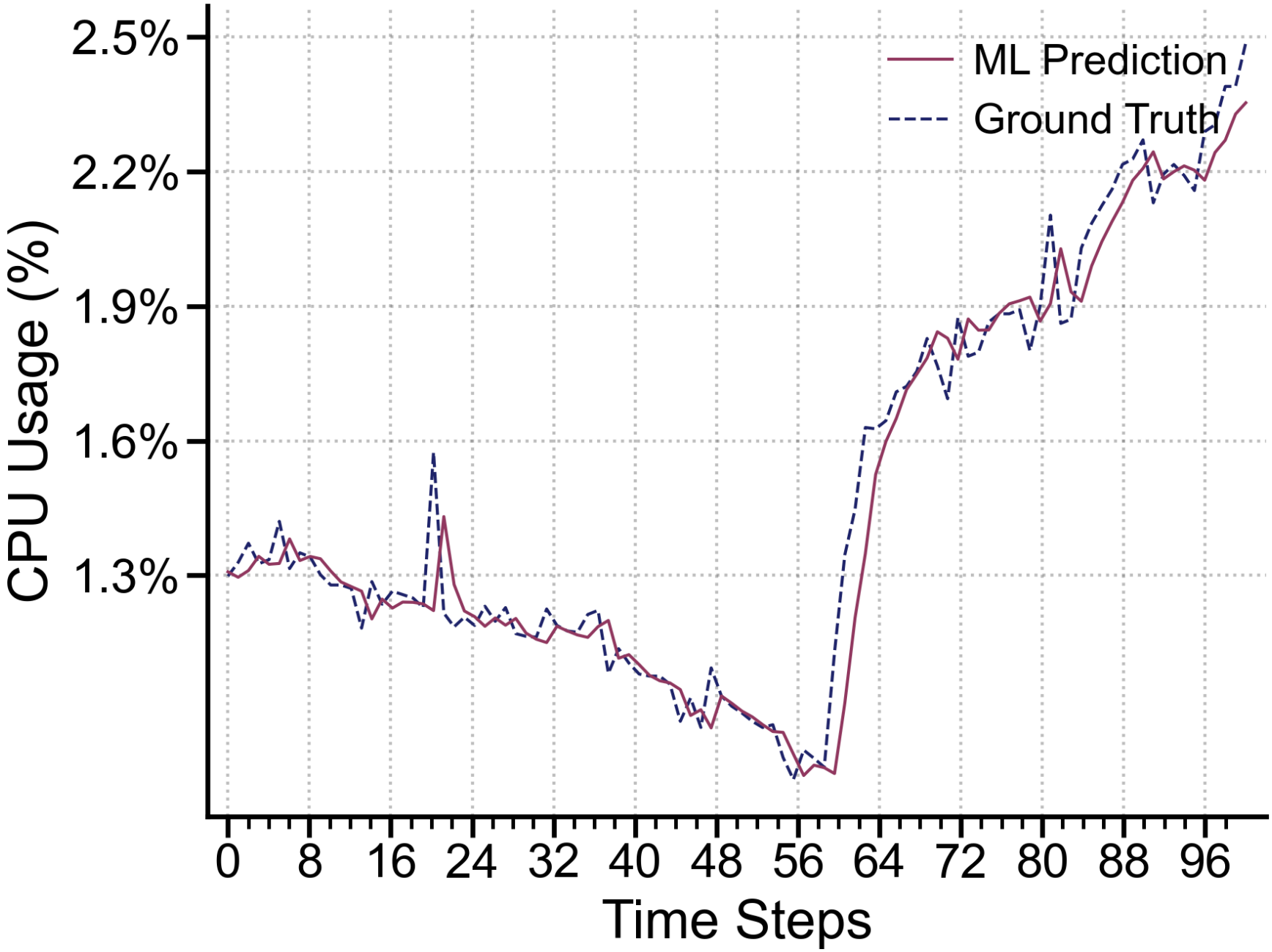
```
ax.grid(True, which='major', linestyle=':', linewidth=0.5, color='gray', alpha=0.5)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# 图例设置
ax.legend(loc='upper right',
          frameon=False,
          fontsize=7,
          handlelength=1.5,
          handletextpad=0.5)

# 边界控制
ax.margins(x=0.02, y=0.05)

plt.show()
```


2.5%
2.2%
1.9%
1.6%
1.3%
CPU Usage (%)
ML Prediction
Ground Truth
0
8
16
24
32
40
48
56
64
72
80
88
96
Time Steps

In [129…

```
# 基础参数设置
lw = 0.5
plt.rcParams['font.family'] = 'Arial'  # 设置全局字体

# 创建图形对象
fig, ax = plt.subplots(figsize=(3.5, 2.5), dpi=600)
fig.subplots_adjust(left=0.15, right=0.85, top=0.9, bottom=0.15)

# 生成坐标轴数据
bins = np.linspace(0, 100, 100)

# 绘制双曲线（假设left数据结构不变）
ax.plot(bins, right[0][:100],
        alpha=1,
        color="#923b62",
        label='ML Prediction',
        linewidth=lw,
        zorder=3)

ax.plot(bins, right[2].iloc[:100],
        alpha=1,
        color="#1e256c",
        label='Ground Truth',
        linewidth=lw,
        linestyle='--',
        zorder=2)

# ========== 关键修改部分 ==========
# 原始数值到百分比的转换（0.013 → 1.3%）
y_ticks = [0.013, 0.016, 0.019, 0.022, 0.025]
y_tick_labels = [f"{y*100:.1f}%" for y in y_ticks]  # 正确转换公式

ax.set_ylabel("CPU Usage (%)", size=9, labelpad=2)  # 添加百分比单位
ax.set_yticks(y_ticks)
ax.set_yticklabels(y_tick_labels)
# ==============================

# X轴设置
ax.set_xlabel('Time Steps', fontsize=9, labelpad=2)
```

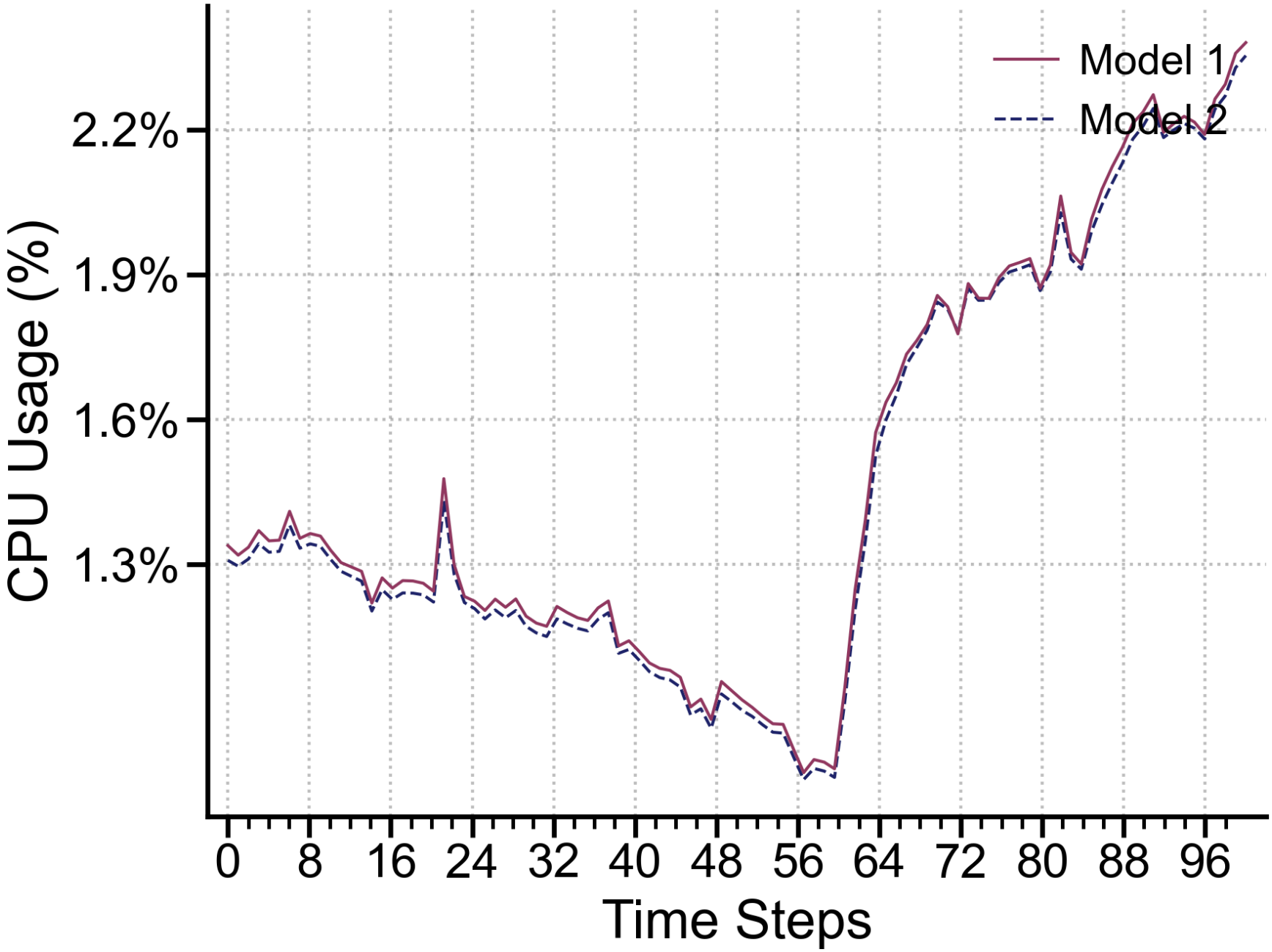
```
ax.set_xticks(np.arange(0, 100, 8))
ax.set_xticks(np.arange(0, 100, 2), minor=True)

# 样式优化
ax.tick_params(axis='both', which='major', labelsize=8, pad=1)
ax.grid(True, which='major', linestyle=':', linewidth=0.5, color='gray', alpha=0.5)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# 图例设置
ax.legend(loc='upper right',
          frameon=False,
          fontsize=7,
          handlelength=1.5,
          handletextpad=0.5)

# 边界控制
ax.margins(x=0.02, y=0.05)

plt.show()
```


2.5%
2.2%
1.9%
1.6%
1.3%
CPU Usage (%)
ML Prediction
Ground Truth
0
8
16
24
32
40
48
56
64
72
80
88
96
Time Steps

In [131…

```
# 基础参数设置
lw = 0.5
plt.rcParams['font.family'] = 'Arial'  # 设置全局字体

# 创建图形对象
fig, ax = plt.subplots(figsize=(3.5, 2.5), dpi=600)
fig.subplots_adjust(left=0.15, right=0.85, top=0.9, bottom=0.15)

# 生成坐标轴数据
bins = np.linspace(0, 100, 100)

# 绘制双曲线（假设left数据结构不变）
ax.plot(bins, left[0][:100],
        alpha=1,
        color="#923b62",
        label='Model 1',
        linewidth=lw,
        zorder=3)

ax.plot(bins, right[0][:100],
        alpha=1,
        color="#1e256c",
        label='Model 2',
        linewidth=lw,
        linestyle='--',
        zorder=2)

# ========== 关键修改部分 ==========
# 原始数值到百分比的转换（0.013 → 1.3%）
y_ticks = [0.013, 0.016, 0.019, 0.022, 0.025]
y_tick_labels = [f"{y*100:.1f}%" for y in y_ticks]  # 正确转换公式

ax.set_ylabel("CPU Usage (%)", size=9, labelpad=2)  # 添加百分比单位
ax.set_yticks(y_ticks)
ax.set_yticklabels(y_tick_labels)
# ==============================

# X轴设置
ax.set_xlabel('Time Steps', fontsize=9, labelpad=2)
```

```
ax.set_xticks(np.arange(0, 100, 8))
ax.set_xticks(np.arange(0, 100, 2), minor=True)

# 样式优化
ax.tick_params(axis='both', which='major', labelsize=8, pad=1)
ax.grid(True, which='major', linestyle=':', linewidth=0.5, color='gray', alpha=0.5)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# 图例设置
ax.legend(loc='upper right',
          frameon=False,
          fontsize=7,
          handlelength=1.5,
          handletextpad=0.5)

# 边界控制
ax.margins(x=0.02, y=0.05)

plt.show()
```


Model 1
Model 2
CPU Usage (%)
2.2%
1.9%
1.6%
1.3%
0
8
16
24
32
40
48
56
64
72
80
88
96
Time Steps

In [ ]:

In [ ]: