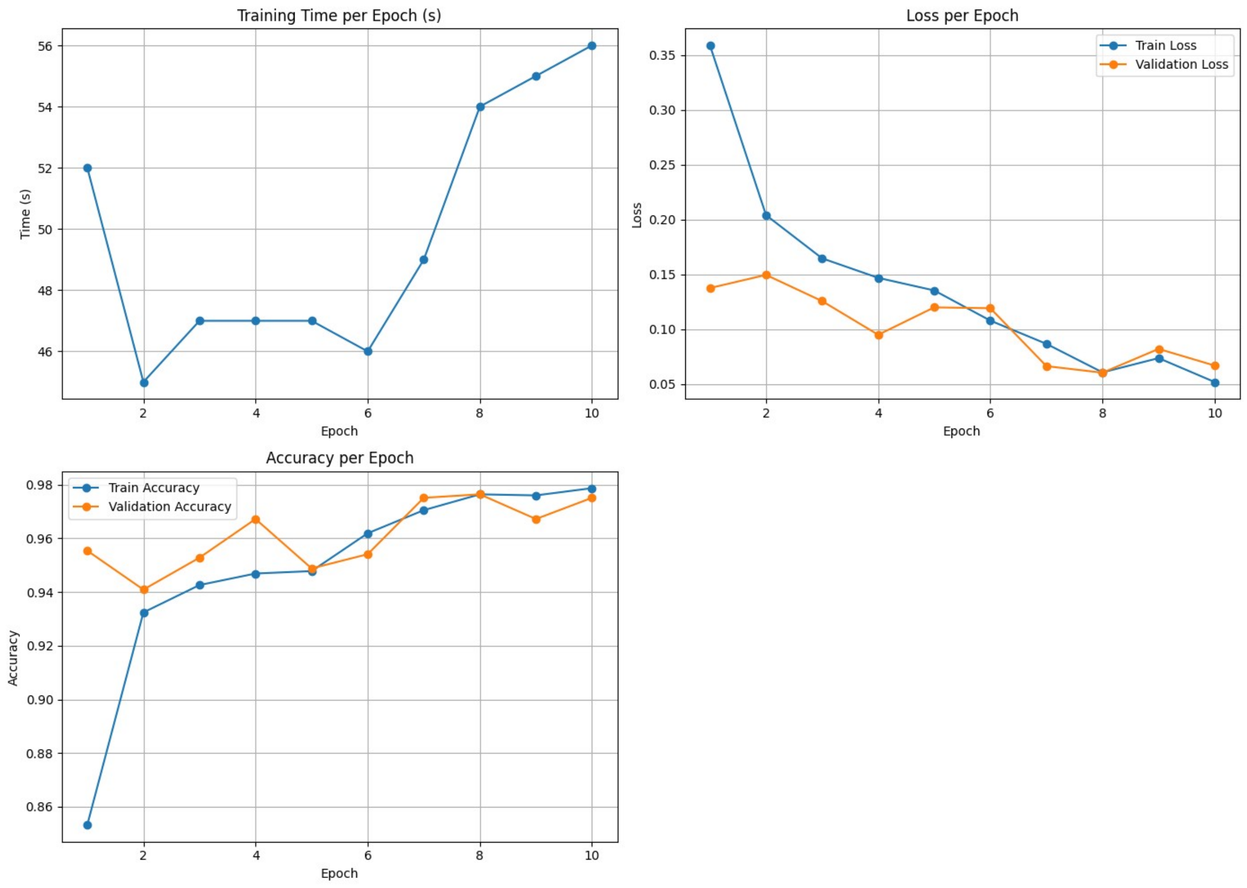
Without Multiprocessing

```
import matplotlib.pyplot as plt

# Metrics data from the user's training logs
epochs = list(range(1, 11))
train_time = [52, 45, 47, 47, 47, 46, 49, 54, 55, 56]
train_loss = [0.3588, 0.2040, 0.1646, 0.1468, 0.1353, 0.1078, 0.0866,
0.0606, 0.0736, 0.0518]
val_loss = [0.1375, 0.1495, 0.1256, 0.0949, 0.1199, 0.1191, 0.0663,
0.0603, 0.0820, 0.0668]
train_acc = [0.8533, 0.9324, 0.9426, 0.9469, 0.9478, 0.9619, 0.9705,
0.9764, 0.9760, 0.9787]
val_acc = [0.9554, 0.9409, 0.9528, 0.9672, 0.9488, 0.9541, 0.9751,
0.9764, 0.9672, 0.9751]

# Create subplots
fig, axs = plt.subplots(2, 2, figsize=(14, 10))

# Plot training time per epoch
axs[0, 0].plot(epochs, train_time, marker='o', color='tab:blue')
axs[0, 0].set_title("Training Time per Epoch (s)")
axs[0, 0].set_xlabel("Epoch")
axs[0, 0].set_ylabel("Time (s)")
axs[0, 0].grid(True)

# Plot loss
axs[0, 1].plot(epochs, train_loss, label='Train Loss', marker='o')
axs[0, 1].plot(epochs, val_loss, label='Validation Loss', marker='o')
axs[0, 1].set_title("Loss per Epoch")
axs[0, 1].set_xlabel("Epoch")
axs[0, 1].set_ylabel("Loss")
axs[0, 1].legend()
axs[0, 1].grid(True)

# Plot accuracy
axs[1, 0].plot(epochs, train_acc, label='Train Accuracy', marker='o')
axs[1, 0].plot(epochs, val_acc, label='Validation Accuracy',
marker='o')
axs[1, 0].set_title("Accuracy per Epoch")
axs[1, 0].set_xlabel("Epoch")
axs[1, 0].set_ylabel("Accuracy")
axs[1, 0].legend()
axs[1, 0].grid(True)

# Hide the unused subplot
axs[1, 1].axis('off')
```

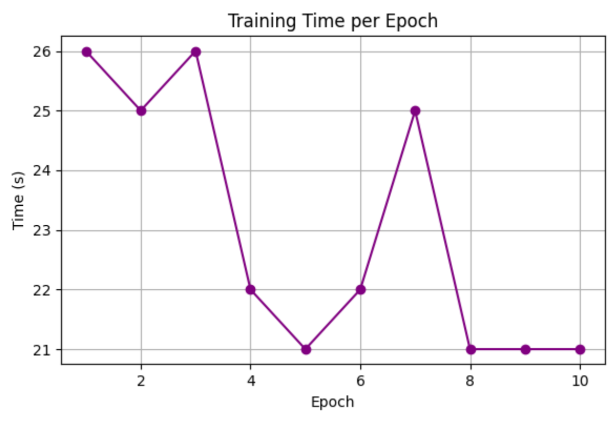
```
plt.tight_layout()
plt.show()
```

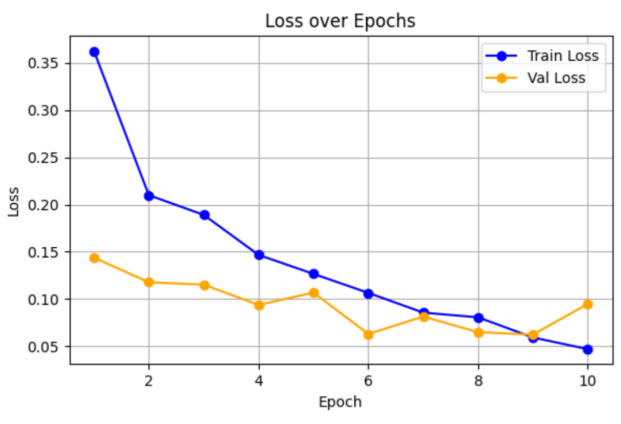


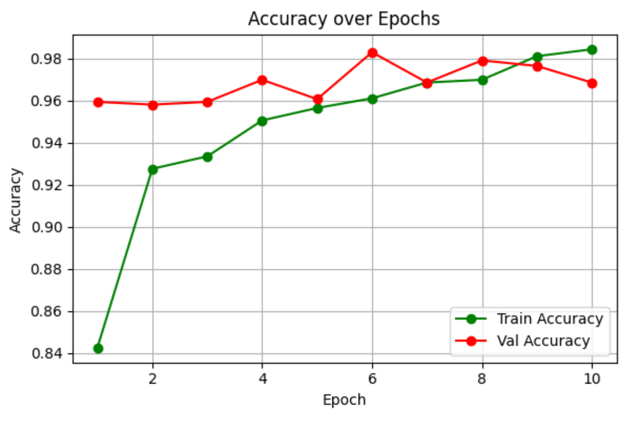
With Multiprocessing

```
import matplotlib.pyplot as plt

# New training log
epochs = list(range(1, 11))
train_time = [26, 25, 26, 22, 21, 22, 25, 21, 21, 21]
train_loss = [0.3627, 0.2100, 0.1891, 0.1467, 0.1266, 0.1065, 0.0856,
0.0806, 0.0593, 0.0469]
val_loss =   [0.1438, 0.1177, 0.1152, 0.0937, 0.1068, 0.0630, 0.0814,
0.0649, 0.0624, 0.0948]
train_acc = [0.8425, 0.9275, 0.9334, 0.9505, 0.9564, 0.9610, 0.9685,
0.9698, 0.9810, 0.9843]
val_acc =   [0.9593, 0.9580, 0.9593, 0.9698, 0.9606, 0.9829, 0.9685,
0.9790, 0.9764, 0.9685]

# Plotting
plt.figure(figsize=(12, 8))
```

```python
plt.subplot(2, 2, 1)
plt.plot(epochs, train_time, marker='o', label='Training Time (s)',
color='purple')
plt.title("Training Time per Epoch")
plt.xlabel("Epoch")
plt.ylabel("Time (s)")
plt.grid(True)

plt.subplot(2, 2, 2)
plt.plot(epochs, train_loss, marker='o', label='Train Loss',
color='blue')
plt.plot(epochs, val_loss, marker='o', label='Val Loss',
color='orange')
plt.title("Loss over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 3)
plt.plot(epochs, train_acc, marker='o', label='Train Accuracy',
color='green')
plt.plot(epochs, val_acc, marker='o', label='Val Accuracy',
color='red')
plt.title("Accuracy over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```



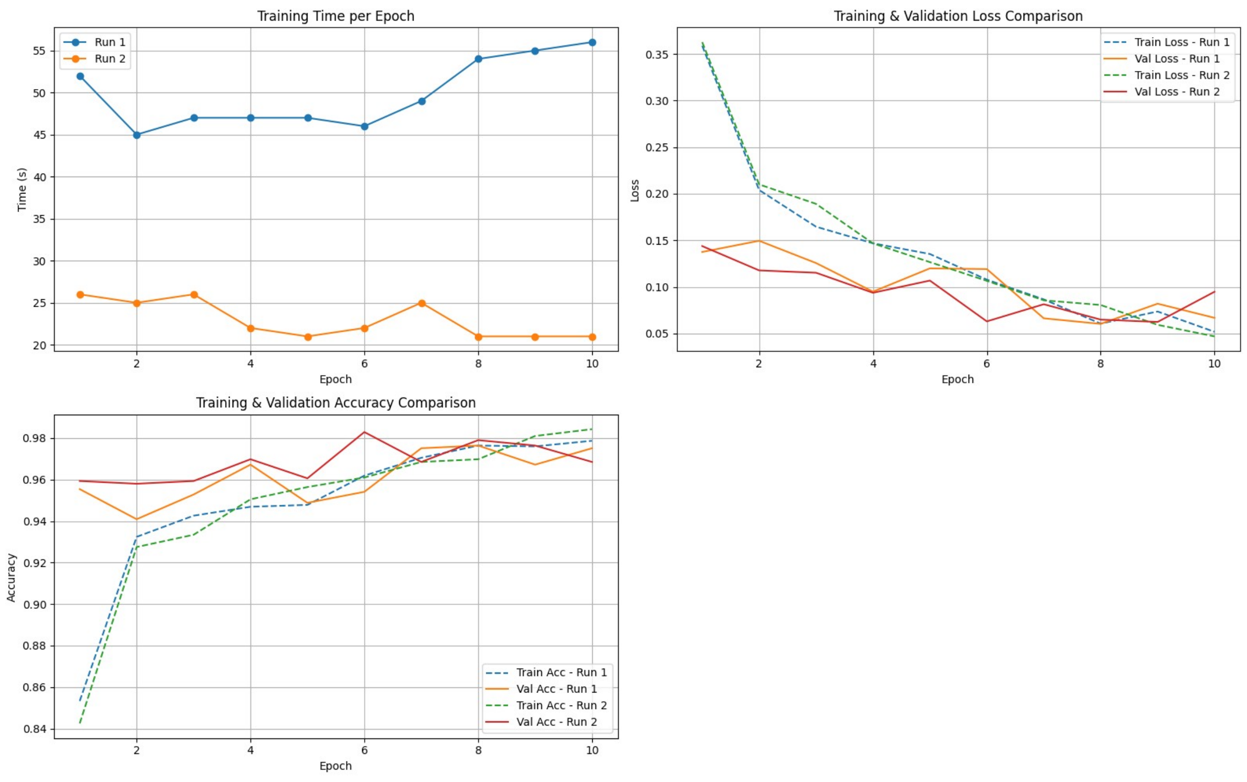




Comparision - With and without Multiprocessing

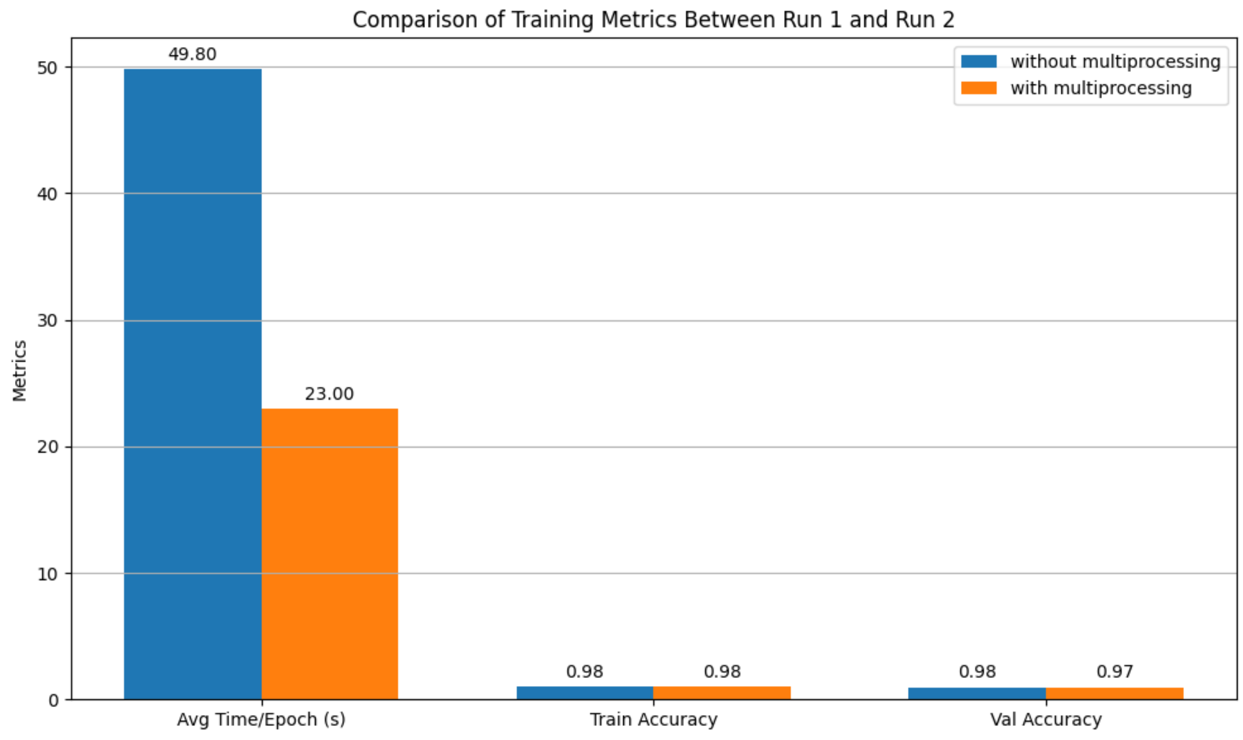
```python
import matplotlib.pyplot as plt

# First training run
epoch_1 = list(range(1, 11))
time_1 = [52, 45, 47, 47, 47, 46, 49, 54, 55, 56]
train_loss_1 = [0.3588, 0.2040, 0.1646, 0.1468, 0.1353, 0.1078,
0.0866, 0.0606, 0.0736, 0.0518]
train_acc_1 = [0.8533, 0.9324, 0.9426, 0.9469, 0.9478, 0.9619, 0.9705,
0.9764, 0.9760, 0.9787]
val_loss_1 = [0.1375, 0.1495, 0.1256, 0.0949, 0.1199, 0.1191, 0.0663,
0.0603, 0.0820, 0.0668]
val_acc_1 = [0.9554, 0.9409, 0.9528, 0.9672, 0.9488, 0.9541, 0.9751,
0.9764, 0.9672, 0.9751]

# Second training run
epoch_2 = list(range(1, 11))
time_2 = [26, 25, 26, 22, 21, 22, 25, 21, 21, 21]
train_loss_2 = [0.3627, 0.2100, 0.1891, 0.1467, 0.1266, 0.1065,
0.0856, 0.0806, 0.0593, 0.0469]
```

```python
train_acc_2 = [0.8425, 0.9275, 0.9334, 0.9505, 0.9564, 0.9610, 0.9685,
0.9698, 0.9810, 0.9843]
val_loss_2 = [0.1438, 0.1177, 0.1152, 0.0937, 0.1068, 0.0630, 0.0814,
0.0649, 0.0624, 0.0948]
val_acc_2 = [0.9593, 0.9580, 0.9593, 0.9698, 0.9606, 0.9829, 0.9685,
0.9790, 0.9764, 0.9685]

# Plotting
plt.figure(figsize=(16, 10))

plt.subplot(2, 2, 1)
plt.plot(epoch_1, time_1, marker='o', label="Run 1")
plt.plot(epoch_2, time_2, marker='o', label="Run 2")
plt.title("Training Time per Epoch")
plt.xlabel("Epoch")
plt.ylabel("Time (s)")
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 2)
plt.plot(epoch_1, train_loss_1, label="Train Loss - Run 1",
linestyle='--')
plt.plot(epoch_1, val_loss_1, label="Val Loss - Run 1", linestyle='-')
plt.plot(epoch_2, train_loss_2, label="Train Loss - Run 2",
linestyle='--')
plt.plot(epoch_2, val_loss_2, label="Val Loss - Run 2", linestyle='-')
plt.title("Training & Validation Loss Comparison")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 3)
plt.plot(epoch_1, train_acc_1, label="Train Acc - Run 1",
linestyle='--')
plt.plot(epoch_1, val_acc_1, label="Val Acc - Run 1", linestyle='-')
plt.plot(epoch_2, train_acc_2, label="Train Acc - Run 2",
linestyle='--')
plt.plot(epoch_2, val_acc_2, label="Val Acc - Run 2", linestyle='-')
plt.title("Training & Validation Accuracy Comparison")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```



```python
import matplotlib.pyplot as plt
import numpy as np

# Average training times
avg_time_run1 = np.mean([52, 45, 47, 47, 47, 46, 49, 54, 55, 56])
avg_time_run2 = np.mean([26, 25, 26, 22, 21, 22, 25, 21, 21, 21])

# Final accuracies and losses
final_acc_run1 = 0.9787
final_acc_run2 = 0.9843

final_val_acc_run1 = 0.9751
final_val_acc_run2 = 0.9685

final_loss_run1 = 0.0518
final_loss_run2 = 0.0469

final_val_loss_run1 = 0.0668
final_val_loss_run2 = 0.0948

# Bar chart
metrics = ['Avg Time/Epoch (s)', 'Train Accuracy', 'Val Accuracy']
run1_values = [avg_time_run1, final_acc_run1, final_val_acc_run1]
run2_values = [avg_time_run2, final_acc_run2, final_val_acc_run2]

x = np.arange(len(metrics))
```

```python
width = 0.35

fig, ax = plt.subplots(figsize=(10, 6))
bars1 = ax.bar(x - width/2, run1_values, width, label='without
multiprocessing')
bars2 = ax.bar(x + width/2, run2_values, width, label='with
multiprocessing')

# Annotate bars
for bar in bars1 + bars2:
    height = bar.get_height()
    ax.annotate(f'{height:.2f}',
                xy=(bar.get_x() + bar.get_width() / 2, height),
                xytext=(0, 3),
                textcoords="offset points",
                ha='center', va='bottom')

ax.set_ylabel('Metrics')
ax.set_title('Comparison of Training Metrics Between Run 1 and Run 2')
ax.set_xticks(x)
ax.set_xticklabels(metrics)
ax.legend()
plt.tight_layout()
plt.grid(axis='y')
plt.show()
```



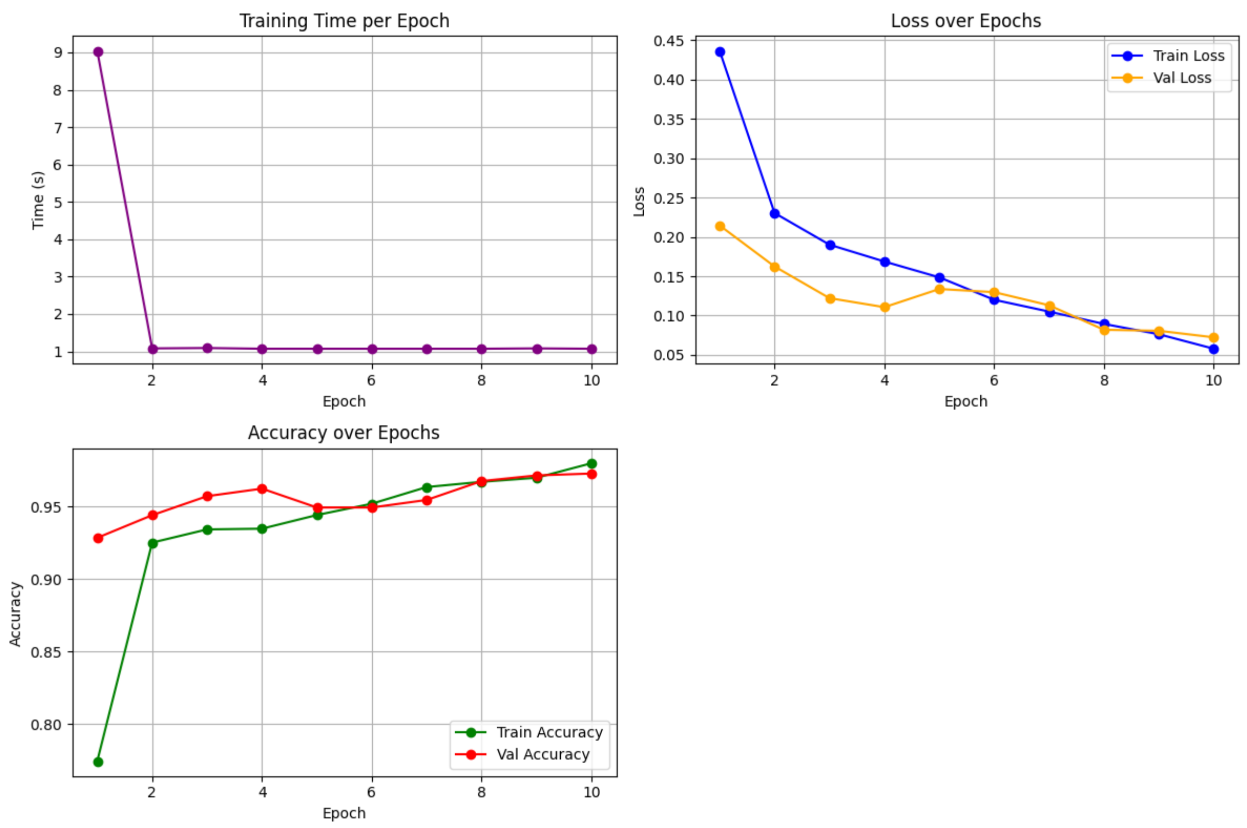
```
import os

# Epoch times in seconds
serial_times = [52, 45, 47, 47, 47, 46, 49, 54, 55, 56]
parallel_times = [26, 25, 26, 22, 21, 22, 25, 21, 21, 21]

# Calculate total times
T_serial = sum(serial_times)
T_parallel = sum(parallel_times)

# Get number of logical cores
p = os.cpu_count()

# Calculate speedup and efficiency
speedup = T_serial / T_parallel
efficiency = speedup / p

# Print results
print(f"CPU Cores Available         : {p}")
print(f"Total Time (Serial)         : {T_serial:.2f} seconds")
print(f"Total Time (Parallel)       : {T_parallel:.2f} seconds")
print(f"Speedup                     : {speedup:.2f}×")
print(f"Efficiency                  : {efficiency * 100:.2f}%")
```

```
CPU Cores Available         : 4
Total Time (Serial)         : 498.00 seconds
Total Time (Parallel)       : 230.00 seconds
Speedup                     : 2.17×
Efficiency                  : 54.13%
```

GPU (Google Collab)

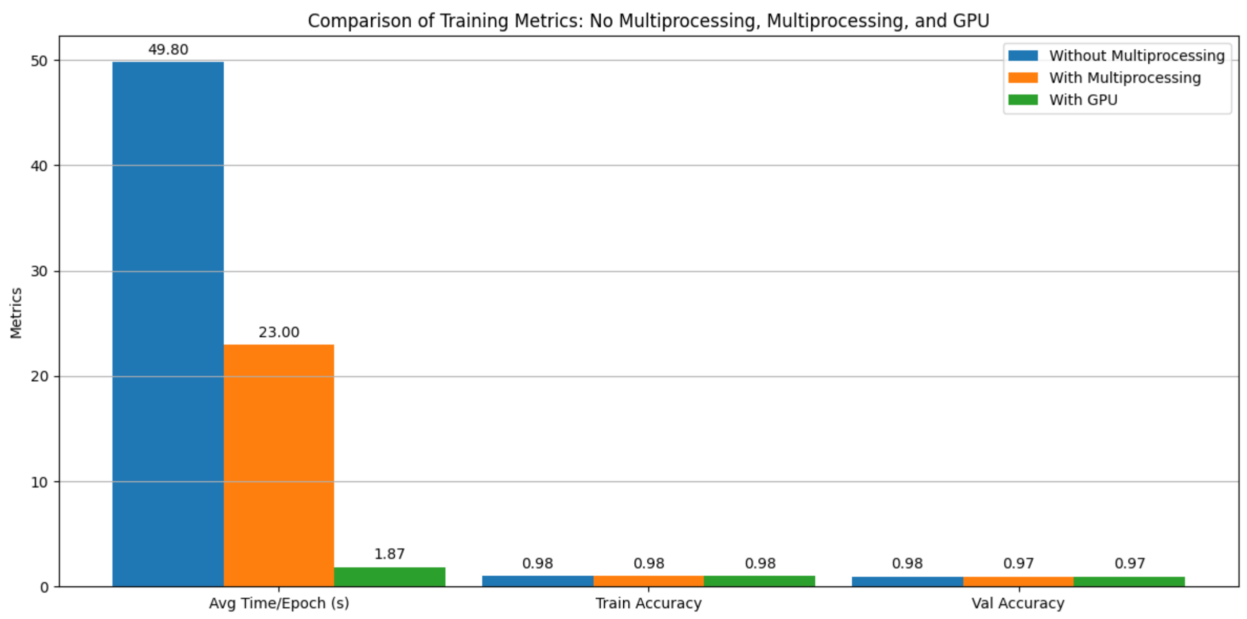
```
import matplotlib.pyplot as plt

# Training log data
epochs = list(range(1, 11))
train_time = [9.04, 1.08, 1.09, 1.07, 1.07, 1.07, 1.07, 1.07, 1.08,
1.07]  # in seconds
train_loss = [0.4363, 0.2303, 0.1901, 0.1687, 0.1486, 0.1201, 0.1051,
0.0894, 0.0763, 0.0579]
val_loss = [0.2145, 0.1624, 0.1222, 0.1107, 0.1338, 0.1298, 0.1131,
0.0820, 0.0807, 0.0724]
train_acc = [0.7744, 0.9250, 0.9341, 0.9346, 0.9440, 0.9518, 0.9633,
0.9669, 0.9697, 0.9797]
val_acc = [0.9283, 0.9439, 0.9570, 0.9622, 0.9492, 0.9492, 0.9544,
0.9674, 0.9713, 0.9726]

# Plotting
```

```
plt.figure(figsize=(12, 8))

# Plotting training time per epoch
plt.subplot(2, 2, 1)
plt.plot(epochs, train_time, marker='o', label='Training Time (s)',
color='purple')
plt.title("Training Time per Epoch")
plt.xlabel("Epoch")
plt.ylabel("Time (s)")
plt.grid(True)

# Plotting loss over epochs
plt.subplot(2, 2, 2)
plt.plot(epochs, train_loss, marker='o', label='Train Loss',
color='blue')
plt.plot(epochs, val_loss, marker='o', label='Val Loss',
color='orange')
plt.title("Loss over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)

# Plotting accuracy over epochs
plt.subplot(2, 2, 3)
plt.plot(epochs, train_acc, marker='o', label='Train Accuracy',
color='green')
plt.plot(epochs, val_acc, marker='o', label='Val Accuracy',
color='red')
plt.title("Accuracy over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)

# Adjust layout and display
plt.tight_layout()
plt.show()
```



```python
import matplotlib.pyplot as plt
import numpy as np

# Average training times
avg_time_run1 = np.mean([52, 45, 47, 47, 47, 46, 49, 54, 55, 56])  # No multiprocessing
avg_time_run2 = np.mean([26, 25, 26, 22, 21, 22, 25, 21, 21, 21])  # With multiprocessing

# GPU training times (in seconds per epoch)
gpu_times = [9.04, 1.08, 1.09, 1.07, 1.07, 1.07, 1.07, 1.07, 1.08, 1.07]
avg_time_gpu = np.mean(gpu_times)  # Average time for GPU run

# Final accuracies and losses
final_acc_run1 = 0.9787  # Without multiprocessing
final_acc_run2 = 0.9843  # With multiprocessing

final_val_acc_run1 = 0.9751  # Without multiprocessing
final_val_acc_run2 = 0.9685  # With multiprocessing

final_loss_run1 = 0.0518  # Without multiprocessing
final_loss_run2 = 0.0469  # With multiprocessing

final_val_loss_run1 = 0.0668  # Without multiprocessing
```

```
final_val_loss_run2 = 0.0948  # With multiprocessing

# GPU final metrics (accuracy and loss from the provided data)
final_acc_gpu = 0.9797  # From the last epoch accuracy on GPU
final_val_acc_gpu = 0.9726  # From the last epoch validation accuracy
on GPU
final_loss_gpu = 0.0579  # From the last epoch loss on GPU
final_val_loss_gpu = 0.0833  # From the last epoch validation loss on
GPU

# Bar chart
metrics = ['Avg Time/Epoch (s)', 'Train Accuracy', 'Val Accuracy']
run1_values = [avg_time_run1, final_acc_run1, final_val_acc_run1]
run2_values = [avg_time_run2, final_acc_run2, final_val_acc_run2]
gpu_values = [avg_time_gpu, final_acc_gpu, final_val_acc_gpu]

x = np.arange(len(metrics))
width = 0.3  # Adjusted width for three sets of bars

fig, ax = plt.subplots(figsize=(12, 6))
bars1 = ax.bar(x - width, run1_values, width, label='Without
Multiprocessing')
bars2 = ax.bar(x, run2_values, width, label='With Multiprocessing')
bars3 = ax.bar(x + width, gpu_values, width, label='With GPU')

# Annotate bars
for bar in bars1 + bars2 + bars3:
    height = bar.get_height()
    ax.annotate(f'{height:.2f}',
                xy=(bar.get_x() + bar.get_width() / 2, height),
                xytext=(0, 3),
                textcoords="offset points",
                ha='center', va='bottom')

ax.set_ylabel('Metrics')
ax.set_title('Comparison of Training Metrics: No Multiprocessing,
Multiprocessing, and GPU')
ax.set_xticks(x)
ax.set_xticklabels(metrics)
ax.legend()
plt.tight_layout()
plt.grid(axis='y')
plt.show()
```

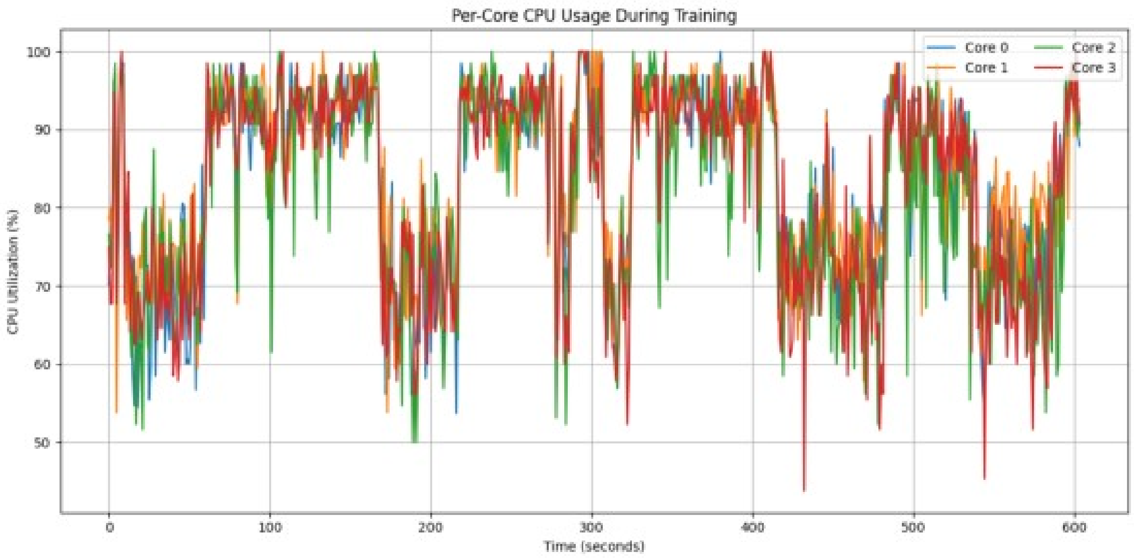


Final CPU & GPU usages

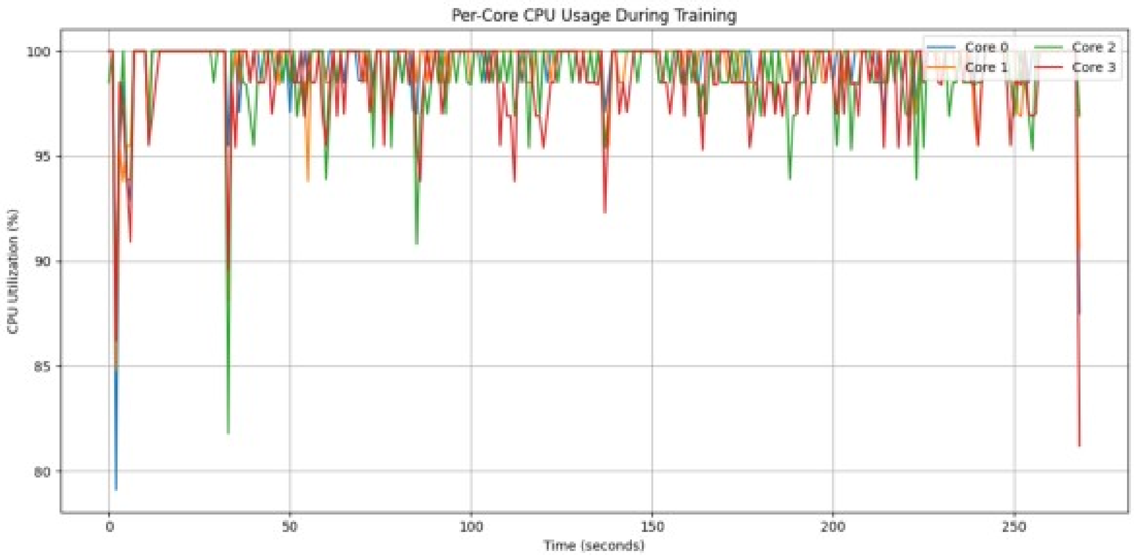
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Load images (ensure these image files are in the same directory as
your script)
img2 = mpimg.imread('cpu-with.png')
img1 = mpimg.imread('cpu-without.png')
img3 = mpimg.imread('gpu.png')

# Create subplot
fig, axs = plt.subplots(3, 1, figsize=(30, 10))  # 1 row, 2 columns

# Plot first image
axs[0].imshow(img1)
axs[0].axis('off')
axs[0].set_title('Without Multiprocessing')

# Plot second image
axs[1].imshow(img2)
axs[1].axis('off')
axs[1].set_title('With Multiprocessing')


# Plot third image
axs[2].imshow(img3)
axs[2].axis('off')
axs[2].set_title('With GPU')
```

```
plt.tight_layout()
plt.show()
```

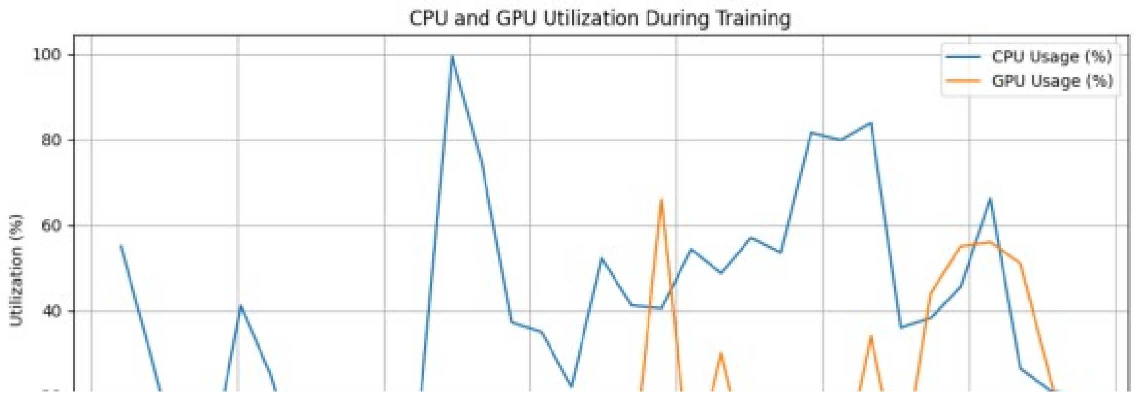
Without Multiprocessing



With Multiprocessing



With GPU


CPU and GPU Utilization During Training
CPU Usage (%)
GPU Usage (%)
Utilization (%)