visualize

December 6, 2025

1 Visuals of Torch and Custom Model

[1]:

```python
import train as custom_model
import CNN_torch as torch_model
import visuals
import glob

# load data
# 1. Train Custom Model
print(">>> Running Custom Framework...")
custom_log_path = custom_model.train(
    optimizer_type="Adam",
    log_name="benchmark_run",
    num_epochs=5,
    batch_size=64,
    lr=0.001
)

# 2. Train Torch Model
print(">>> Running PyTorch Framework...")
torch_log_path = torch_model.train_torch(
    optimizer_type="Adam",
    log_name="benchmark_run",
    num_epochs=5,
    batch_size=64,
    lr=0.001
)
```

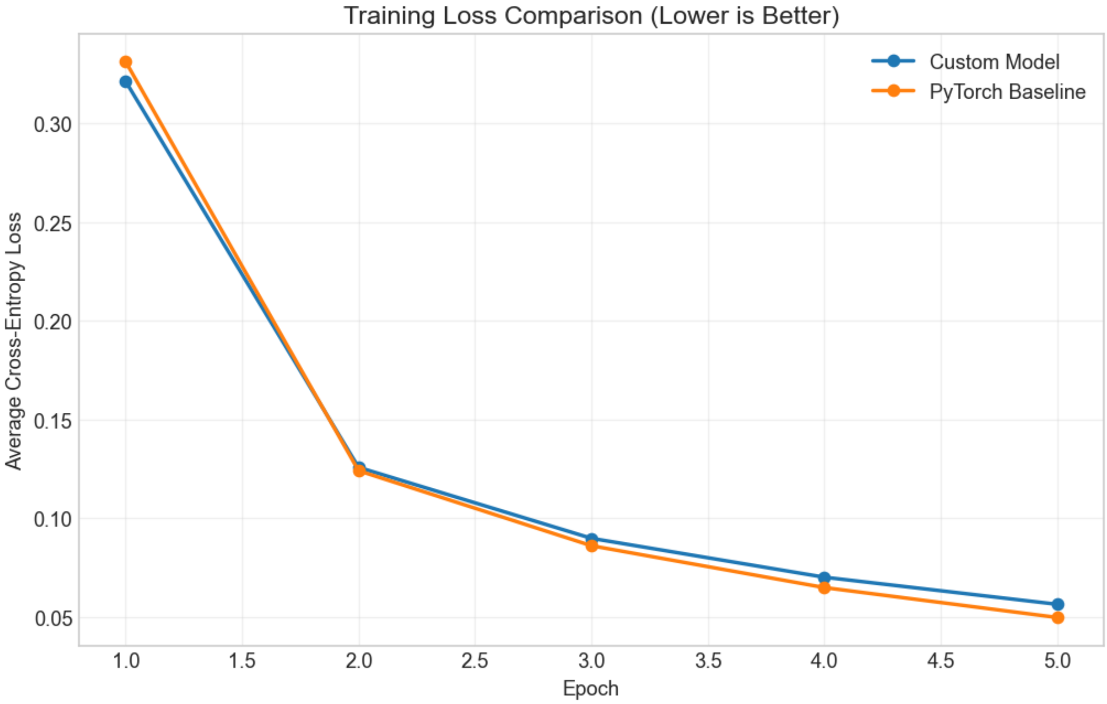
```
>>> Running Custom Framework…
[INFO] Training Start | Optimizer: Adam | Epochs: 5
[INFO] Training log saved to:
./logs/benchmark_run_k3_c8_h128_Adam_lr0.001_20251206_214003.json
>>> Running PyTorch Framework…
[INFO-TORCH] Training Start | Opt: Adam | Device: cpu
Training log saved to:
./logs/benchmark_run_k3_c8_h128_Adam_lr0.001_20251206_214111.json
```

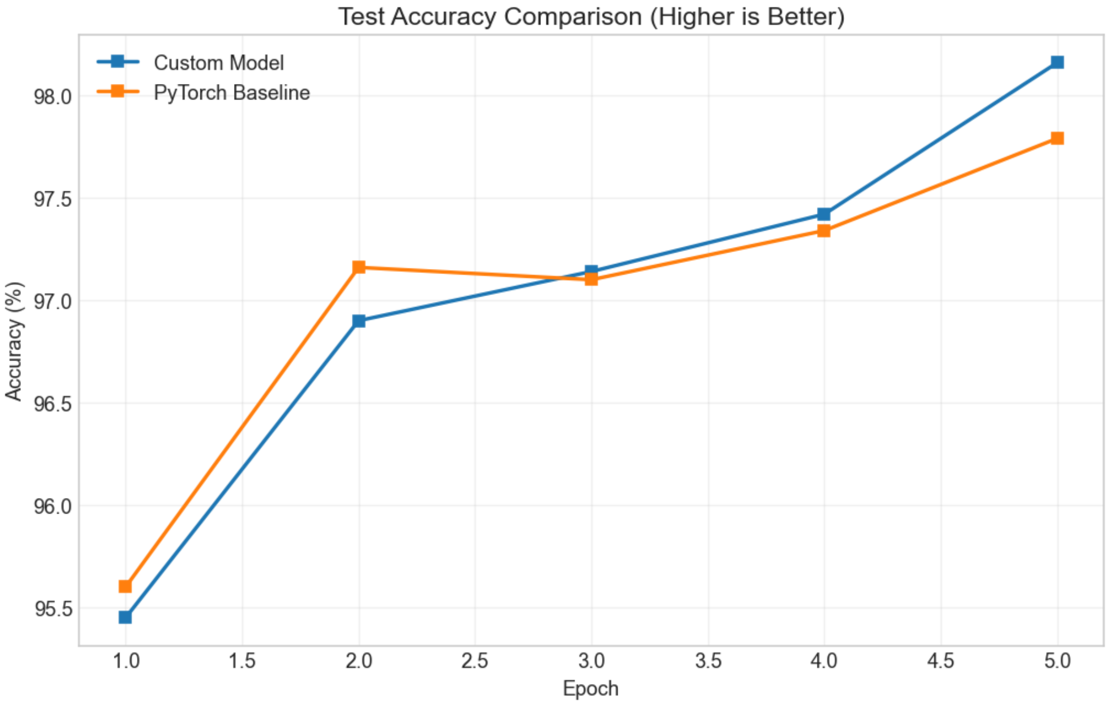
1.1 1. Basic Comparison: Loss & Accuracy Curves

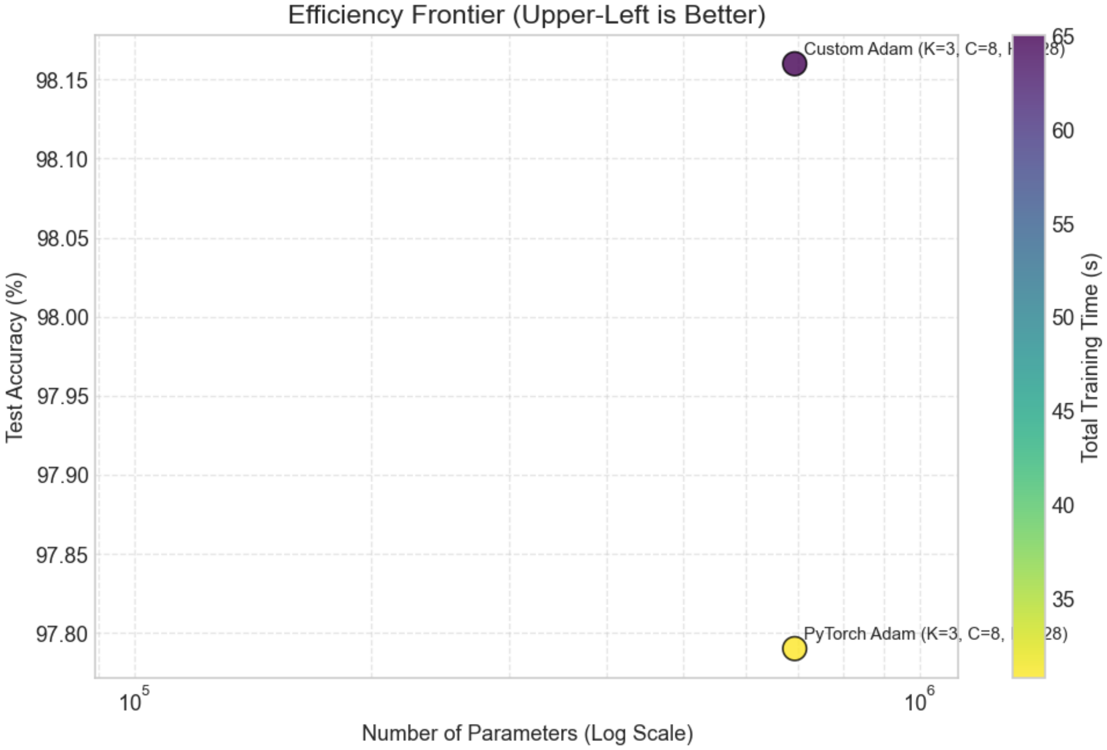
[2]:

```
print("\n>>> Generating Comparison Plots...")
visuals.plot_loss_comparison(custom=custom_log_path, torch=torch_log_path)
visuals.plot_accuracy_comparison(custom=custom_log_path, torch=torch_log_path)
visuals.plot_efficiency_frontier([custom_log_path, torch_log_path])
```

```
>>> Generating Comparison Plots…
```



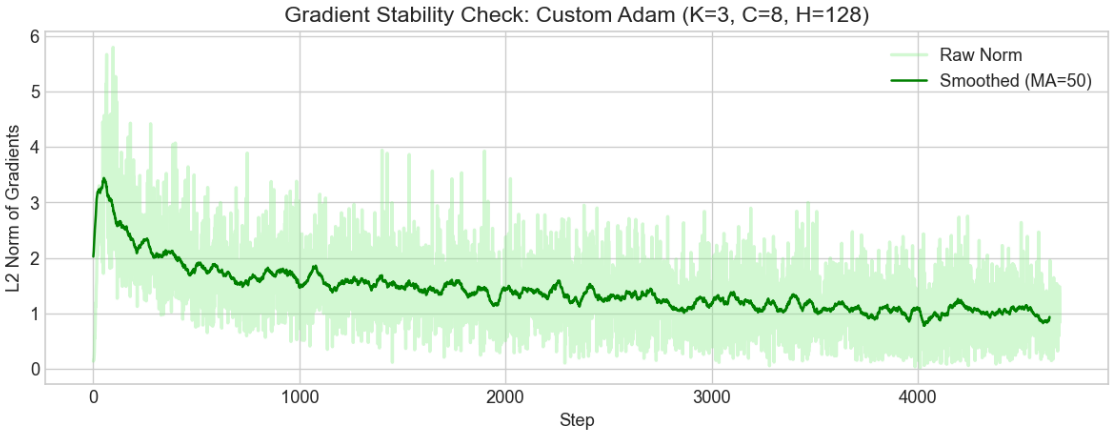

Test Accuracy Comparison (Higher is Better)
Custom Model
PyTorch Baseline
Accuracy (%)
98.0
97.5
97.0
96.5
96.0
95.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Epoch

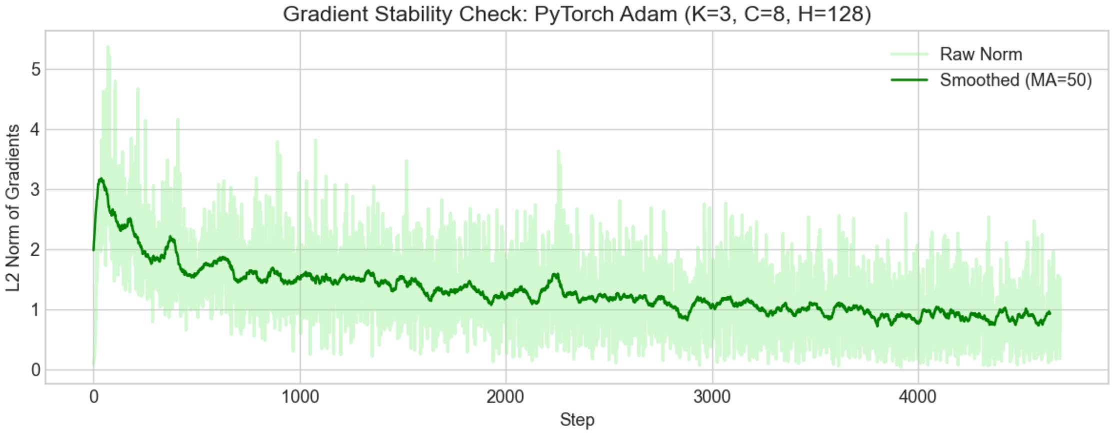

Efficiency Frontier (Upper-Left is Better)
Custom Adam (K=3, C=8, H 28)
PyTorch Adam (K=3, C=8, 28)
Test Accuracy (%)
98.15
98.10
98.05
98.00
97.95
97.90
97.85
97.80
10^5
10^6
Number of Parameters (Log Scale)
Total Training Time (s)
65
60
55
50
45
40
35

1.2 2. Gradient Norms Comparison

```
[3]: print("\n>>> Deep Diagnostics: Gradient Norm (Stability Check)")
     visuals.plot_gradient_norm(custom_log_path)
     visuals.plot_gradient_norm(torch_log_path)
```

```
>>> Deep Diagnostics: Gradient Norm (Stability Check)
```

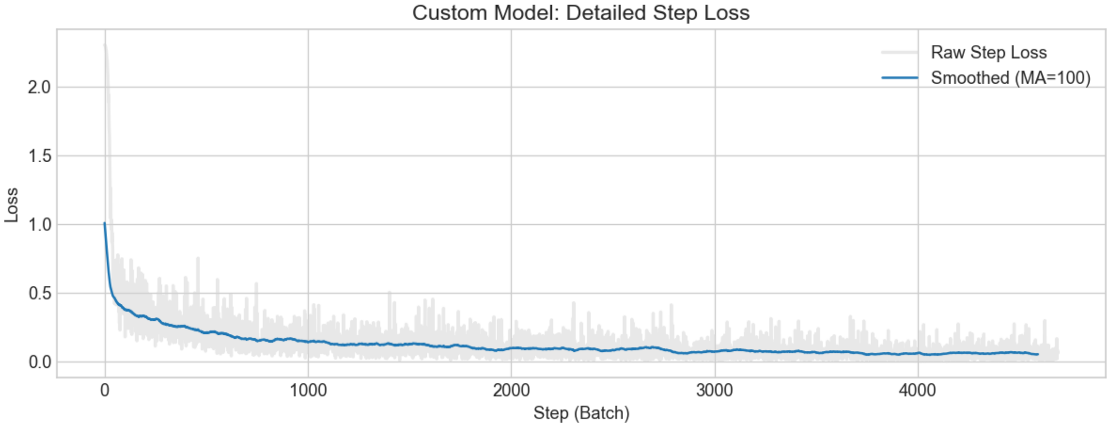


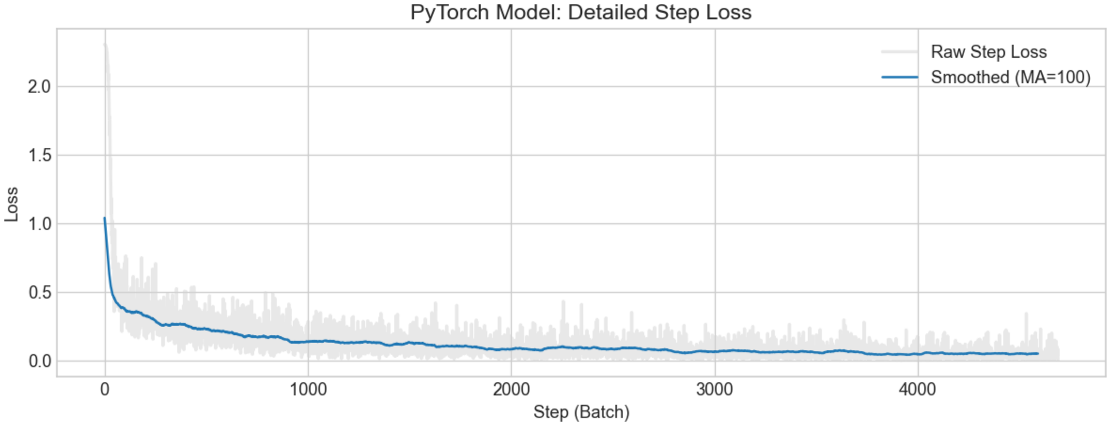


1.3 3. Detailed Training Dynamics Analysis

```
[4]: print("\n>>> 2. Detailed Training Dynamics (Step-level Loss)")
     visuals.plot_detailed_loss(custom_log_path, window=100, title="Custom Model:␣
      ↪Detailed Step Loss")
     visuals.plot_detailed_loss(torch_log_path, window=100, title="PyTorch Model:␣
      ↪Detailed Step Loss")
```

```
>>> 2. Detailed Training Dynamics (Step-level Loss)
```



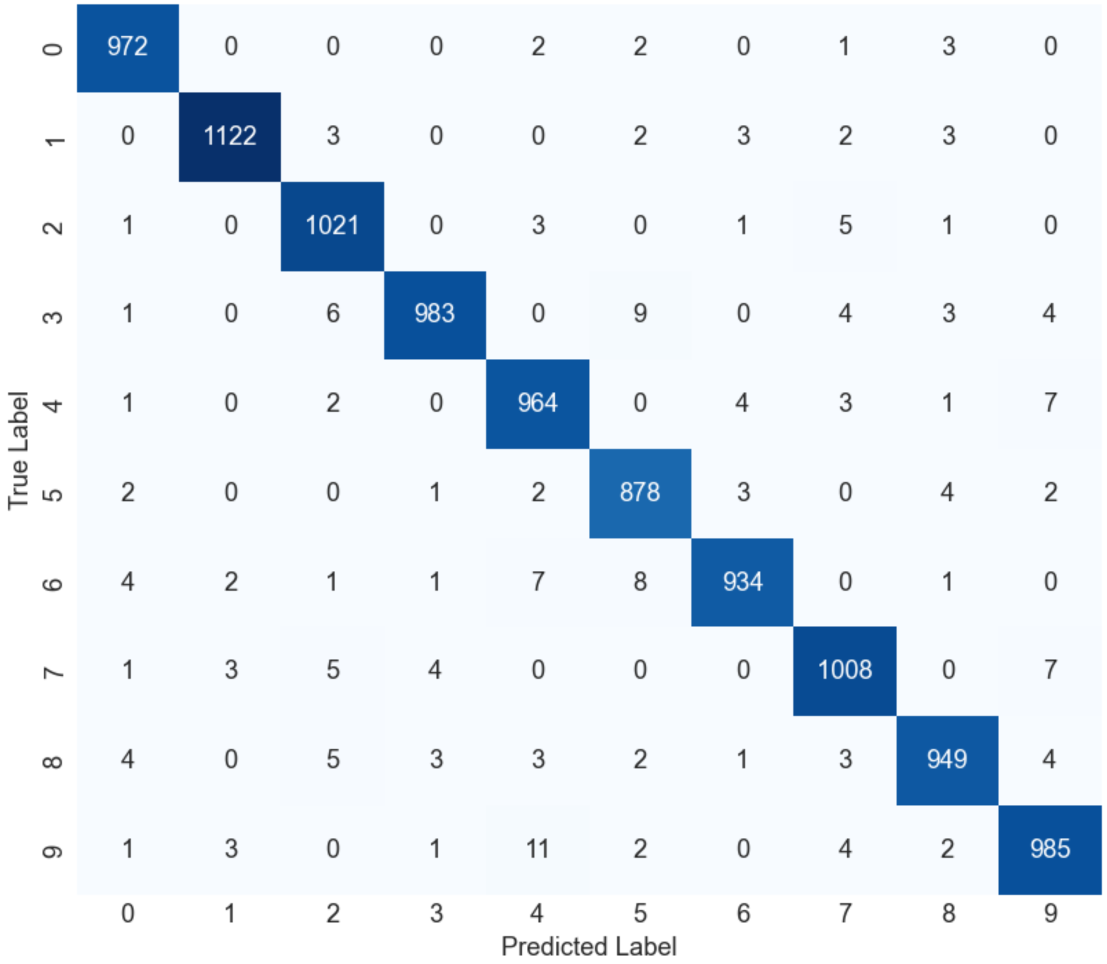


1.4 4. Conclusion Matrix Visualization

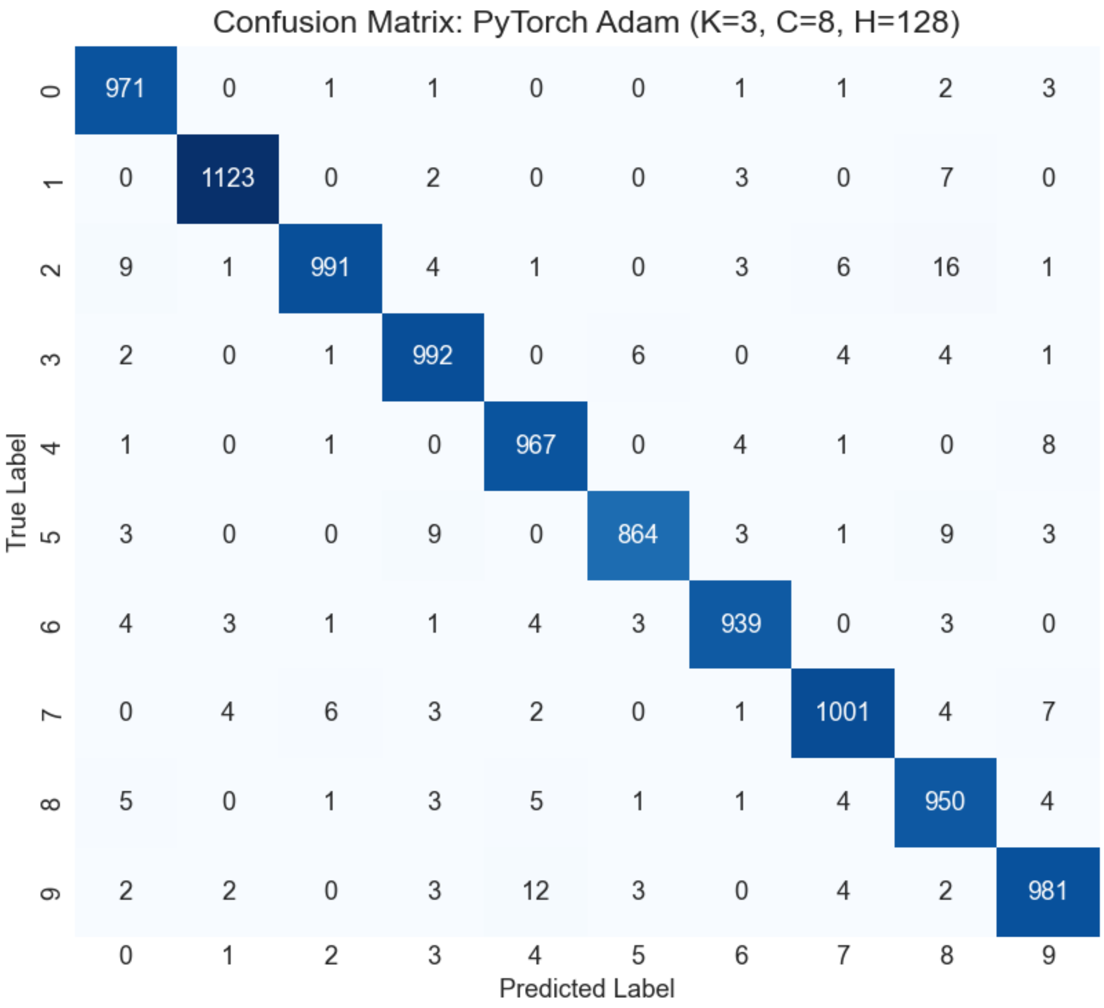
```
[5]: print("\n>>> 3. Error Analysis: Confusion Matrix")
     visuals.plot_confusion_matrix(custom_log_path)
     visuals.plot_confusion_matrix(torch_log_path)
```

```
>>> 3. Error Analysis: Confusion Matrix
```

Confusion Matrix: Custom Adam (K=3, C=8, H=128)



	0	1	2	3	4	5	6	7	8	9
0	972	0	0	0	2	2	0	1	3	0
1	0	1122	3	0	0	2	3	2	3	0
2	1	0	1021	0	3	0	1	5	1	0
3	1	0	6	983	0	9	0	4	3	4
4	1	0	2	0	964	0	4	3	1	7
5	2	0	0	1	2	878	3	0	4	2
6	4	2	1	1	7	8	934	0	1	0
7	1	3	5	4	0	0	0	1008	0	7
8	4	0	5	3	3	2	1	3	949	4
9	1	3	0	1	11	2	0	4	2	985

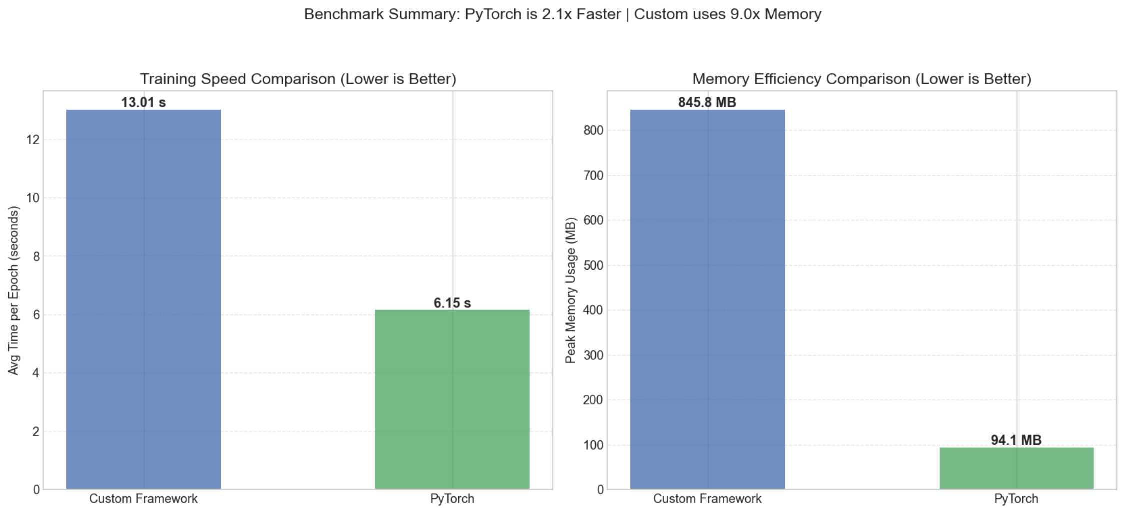


Confusion Matrix: PyTorch Adam (K=3, C=8, H=128)

	0	1	2	3	4	5	6	7	8	9
0	971	0	1	1	0	0	1	1	2	3
1	0	1123	0	2	0	0	3	0	7	0
2	9	1	991	4	1	0	3	6	16	1
3	2	0	1	992	0	6	0	4	4	1
4	1	0	1	0	967	0	4	1	0	8
5	3	0	0	9	0	864	3	1	9	3
6	4	3	1	1	4	3	939	0	3	0
7	0	4	6	3	2	0	1	1001	4	7
8	5	0	1	3	5	1	1	4	950	4
9	2	2	0	3	12	3	0	4	2	981

1.5 5. Time & Memory Efficiency Comparison

[6]:
```
visuals.plot_system_benchmark(custom_log_path, torch_log_path)
```


Benchmark Summary: PyTorch is 2.1x Faster | Custom uses 9.0x Memory
Training Speed Comparison (Lower is Better)
Avg Time per Epoch (seconds)
0
2
4
6
8
10
12
13.01 s
6.15 s
Custom Framework
PyTorch
Memory Efficiency Comparison (Lower is Better)
Peak Memory Usage (MB)
0
100
200
300
400
500
600
700
800
845.8 MB
94.1 MB
Custom Framework
PyTorch

[ ]: