OPTIMIZATION TIER

MODULE 20

Capstone

Bringing it all together - your complete ML system

Module 20: Capstone
Benchmarking & Submission

From Framework to Reproducible Science

TinyTorch Optimization Tier

Claims Without Context Are Meaningless

The Marketing Claim

The Engineering Reality

Hardware Platform:	[_____]
Batch Size:	[_____]
Precision (FP32/INT8):	[_____]
Variance (Std Dev):	[_____]
System Load:	[_____]

Inter: A number is not a result unless it is reproducible.

The Three Pillars of Benchmarking

Reliable Benchmark

REPEATABILITY

Fixed seeds.

Consistent environments. Low **variance**. If I run it twice, I get the same number.

COMPARABILITY

Apples-to-apples.

Standardized **hardware** definitions. Explicit software versions.

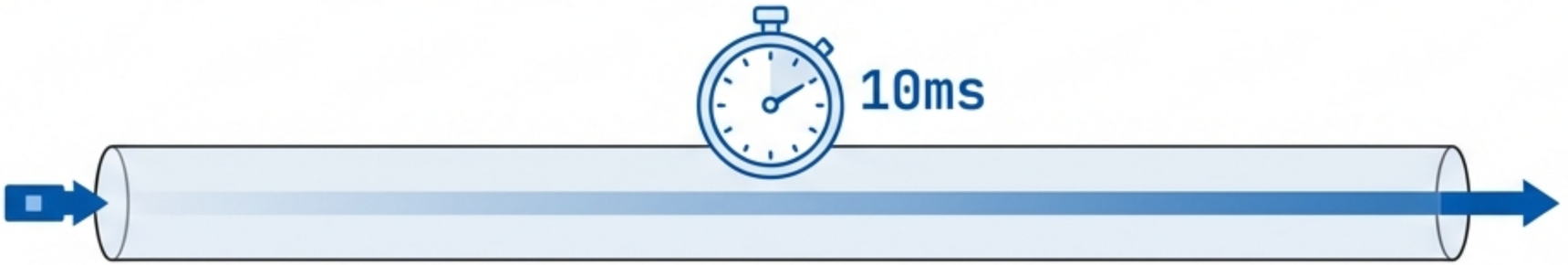
COMPLETENESS

No cherry-picking.

Measuring **Accuracy** + Latency + Memory together.

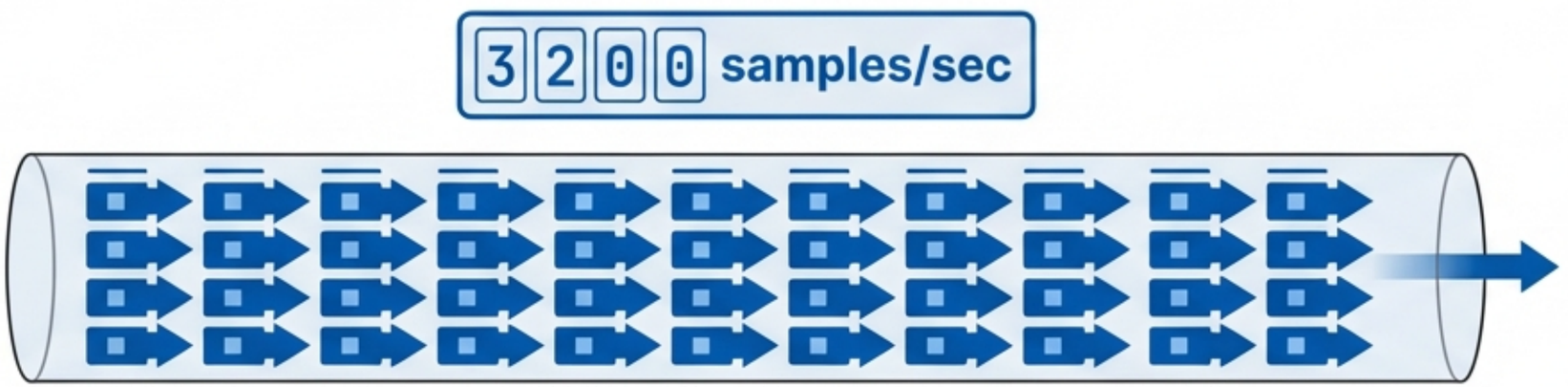
The Measurement Problem: Latency vs. Throughput

LATENCY (Real-time / UX)



Time to process ONE sample.

THROUGHPUT (Batch Processing)



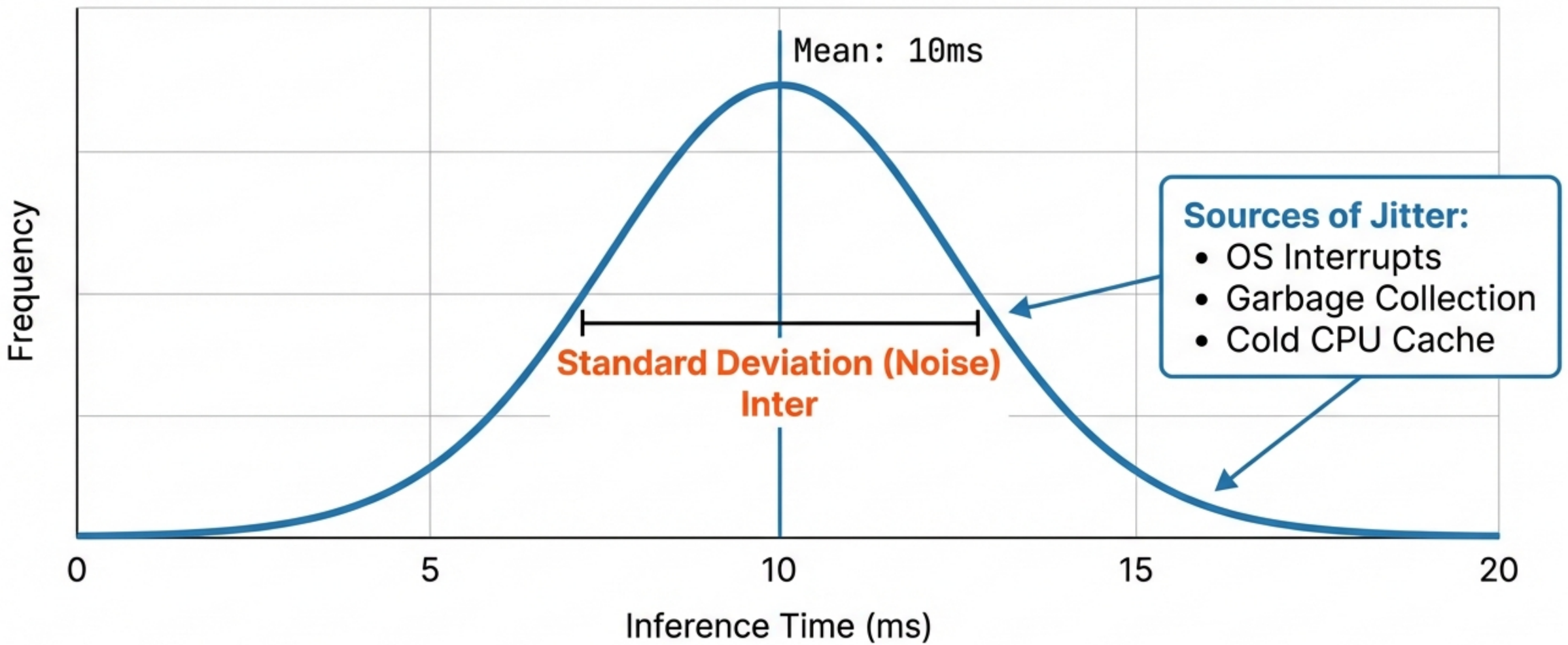
Total samples processed per second.

Throughput =

Batch Size / Total Time

Batching increases **Throughput** (efficiency) but degrades **Latency** **Latency** (wait time).

System Noise & Variance

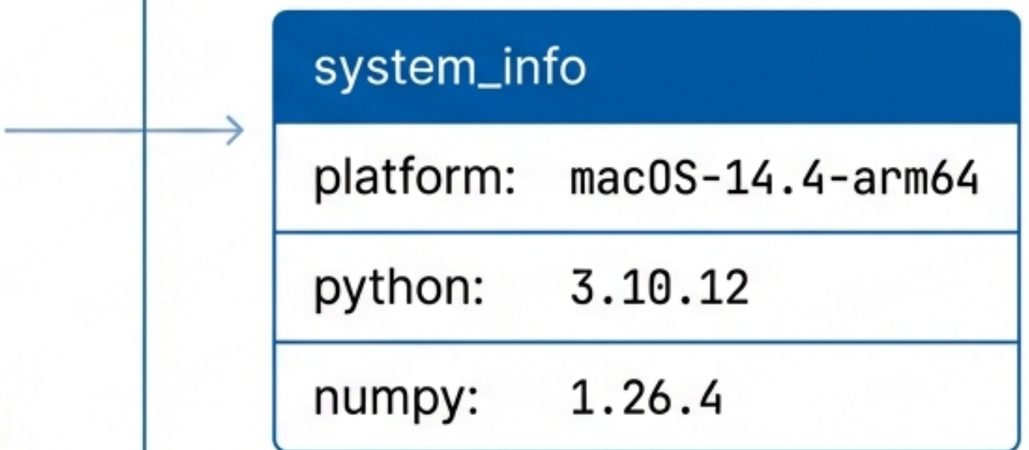


Single measurements capture noise. Distributions capture reality.

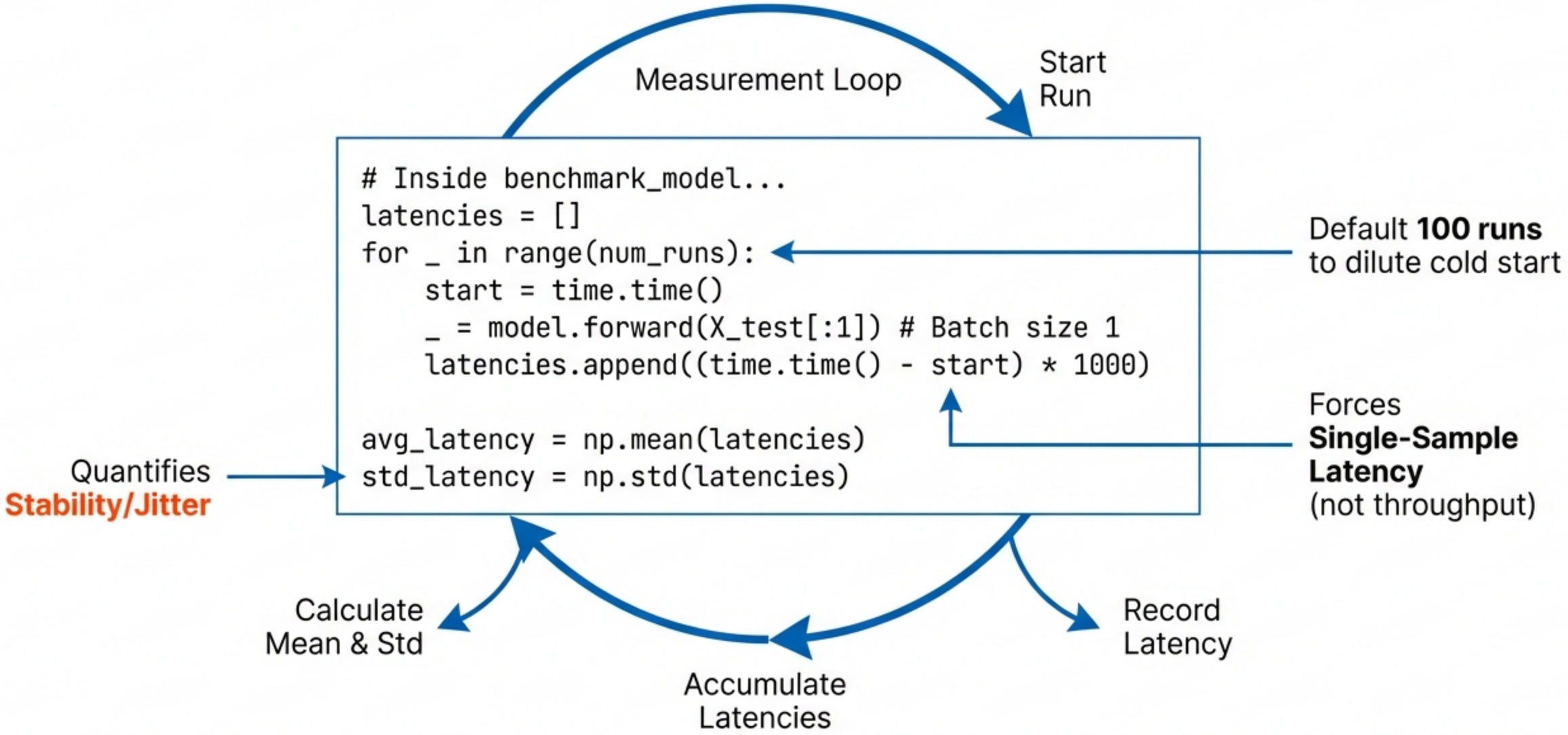
Capturing Context with BenchmarkReport

```
class BenchmarkReport:
    def __init__(self, model_name='model'):
        self.model_name = model_name
        self.metrics = {}
        # Capture Context immediately
        self.system_info = self._get_system_info()

    def _get_system_info(self):
        return {
            'platform': platform.platform(),
            'python_version': sys.version.split()[0],
            'numpy_version': np.__version__
        }
```



Implementing Latency Measurement



Measuring Cost (Model Size)

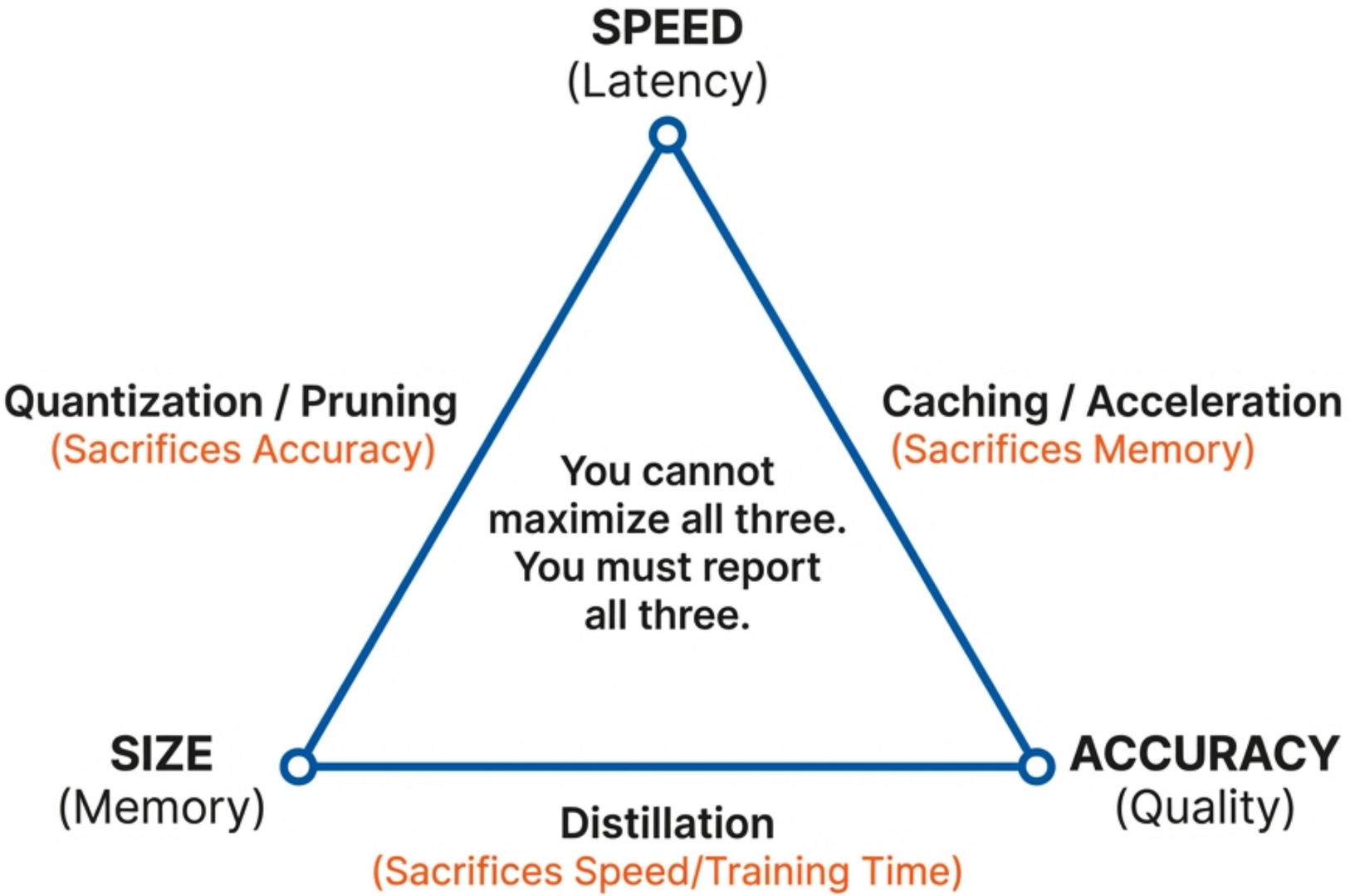
```
# Inside SimpleMLP...
def count_parameters(self):
    total = 0
    for param in self.parameters():
        total += param.data.size
    return total
```

Calculation Logic

$$\text{Model Size (MB)} = \frac{(\text{Parameter Count} \times \text{Bit Width})}{(1024^2)}$$

Parameters: 1,000,000	×	FP32 (4 Bytes)	=	4,000,000 Bytes	= →	3.81 MB

The Optimization Trade-off Triangle



Generating the Submission

```
submission['improvements'] = {
    'speedup': float(baseline / optimized),
    'compression': float(base_size / opt_size),
    'accuracy_delta': float(opt_acc - base_acc)
}
```

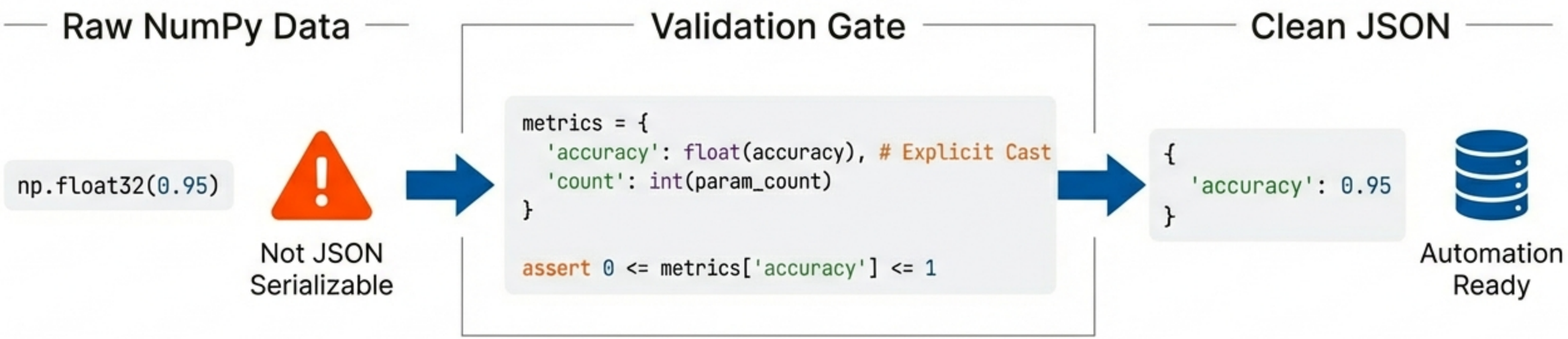
Success Criteria

✅ 1. Speedup > 1.0

✅ 2. Compression > 1.0

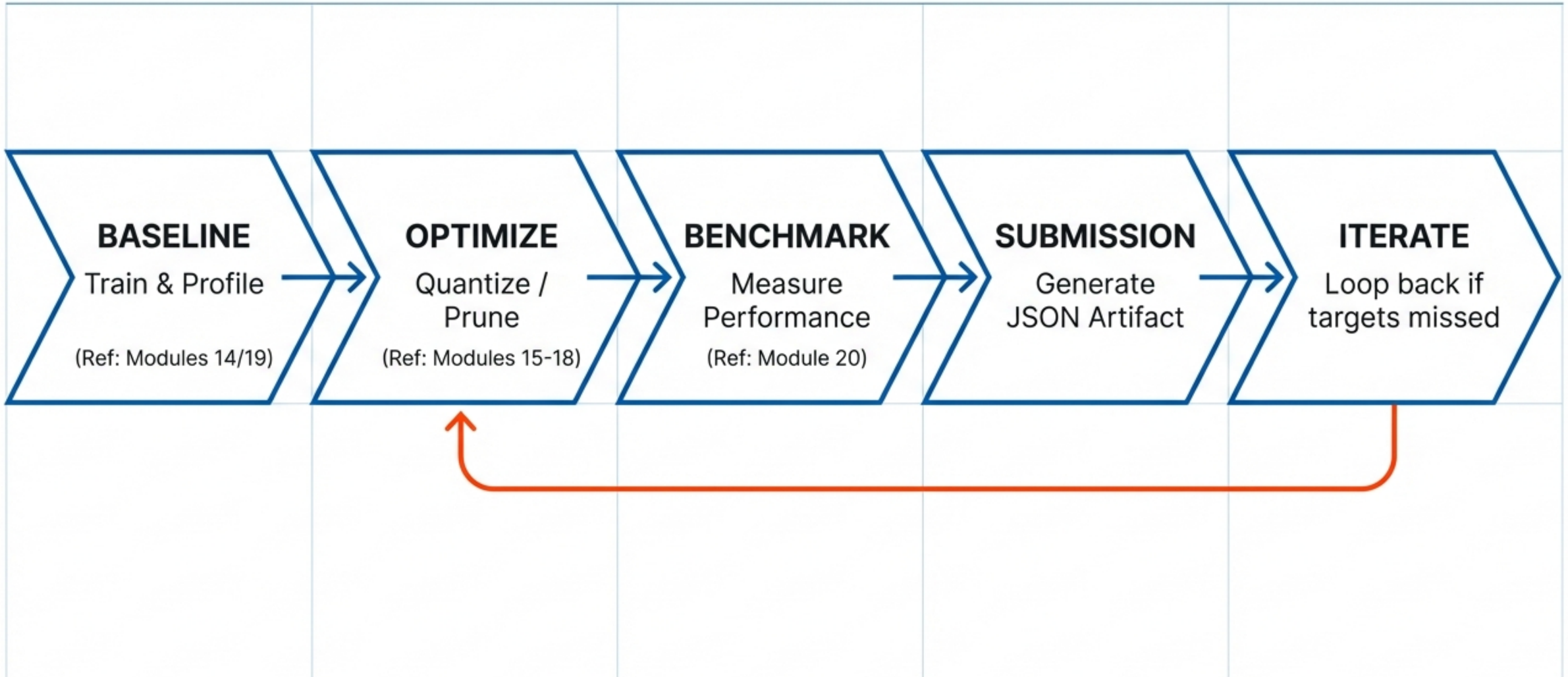
✅ 3. Accuracy Delta ≈ 0.0

Enforcing Standards via Types

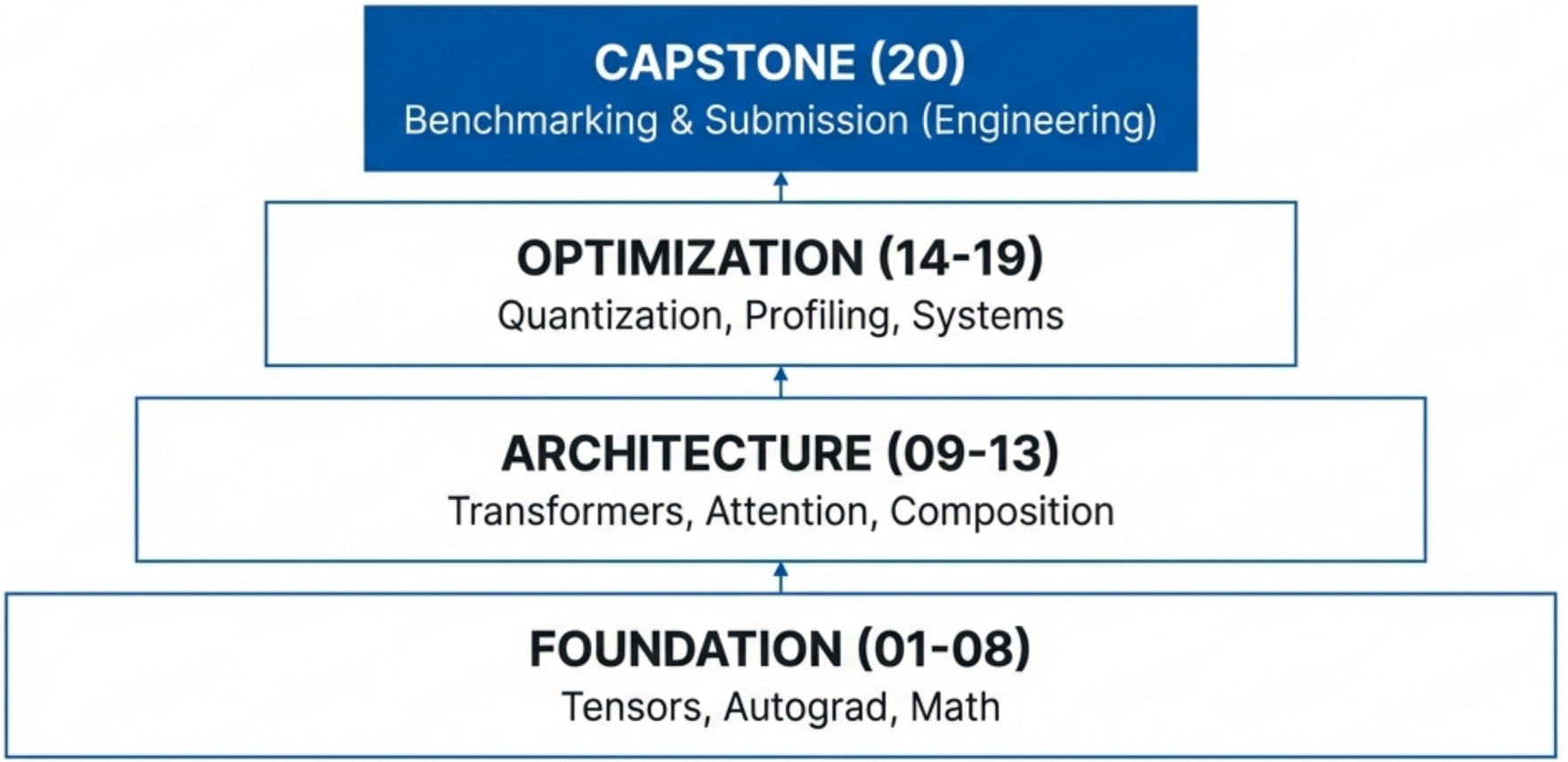


Strict types enable automated CI/CD pipelines and leaderboards.

The Complete ML Lifecycle



Capstone Retrospective



You have built a framework. Now go prove it works.