模型编译优化指标速查表

🎯 快速决策

我应该关注哪些指标?

> 问：我的场景是什么?
> └ 选择对应的优先级

场景类型	核心指标（必看）	次要指标	可忽略
手机APP	模型大小、功耗、延迟	准确率	吞吐量
实时视频	延迟、FPS	准确率、功耗	模型大小
云端服务	吞吐量、成本	准确率、延迟	功耗、大小
边缘设备	功耗、内存、延迟	准确率	成本
IoT/嵌入式	功耗、模型大小、内存	延迟	吞吐量

📊 六大指标类别

1. 性能指标 ⚡

指标	定义	单位	典型目标	测量工具
延迟	单次推理时间	ms	< 50ms	`time.perf_counter()`
吞吐量	每秒处理量	QPS/FPS	> 100 QPS	`samples/second`
FLOPs	浮点运算数	GFLOPs	越低越好	`thop.profile()`

2. 资源指标 💾

指标	定义	单位	典型目标	测量工具
模型大小	磁盘占用	MB	< 50MB	`os.path.getsize()`
内存占用	运行时内存	MB	< 500MB	`torch.cuda.max_memory_allocated()`
参数量	可学习参数数	M	< 100M	`sum(p.numel())`

3. 精度指标 🎯

指标	定义	范围	可接受损失	应用场景
准确率	预测正确率	0–100%	< 2% drop	分类
mAP	平均精度	0–1	< 5% drop	检测
BLEU	翻译质量	0–100	< 3 drop	NLP

4. 能耗指标 🔋

指标	定义	单位	典型目标	重要场景
功耗	推理时功率	W	< 5W	移动设备
能效比	推理/瓦特	inf/W	越高越好	所有场景
单次能耗	一次推理能量	mJ	< 100mJ	IoT

5. 可靠性指标 ✅

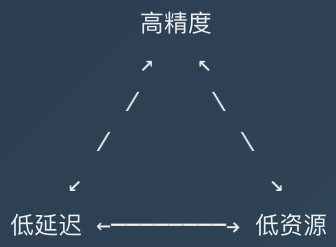
指标	定义	目标	检查方法
稳定性	结果一致性	标准差 < 0.01	多次运行
数值精度	无NaN/Inf	0错误	`torch.isnan()`
错误率	崩溃率	< 0.01%	压力测试

6. 成本指标 💰

指标	定义	考虑因素
硬件成本	设备价格	芯片、内存、存储
云服务费	推理费用	实例类型、使用时长
研发成本	开发时间	工程师、调试、训练

⚖️ 不可能三角

你只能选两个！



典型权衡：

- **高精度 + 低延迟** = 需要更多资源（大GPU）
- **高精度 + 低资源** = 延迟会增加（慢）
- **低延迟 + 低资源** = 精度会下降

🔧 优化技术对比

技术	速度提升	大小压缩	精度影响	难度	推荐度
INT8量化	2–4x	4x	–1~2%	⭐⭐	⭐⭐⭐⭐⭐
FP16量化	1.5–2x	2x	0%	⭐	⭐⭐⭐⭐⭐
剪枝50%	1.5–2x	2x	–2~5%	⭐⭐⭐	⭐⭐⭐⭐
知识蒸馏	5–10x	10x	–3~5%	⭐⭐⭐⭐	⭐⭐⭐⭐
TensorRT	2–5x	1x	0%	⭐⭐	⭐⭐⭐⭐⭐
轻量架构	10x+	10x+	–5~10%	⭐⭐⭐⭐⭐	⭐⭐⭐

📱 场景速查

场景1：手机APP

```
约束：
- 模型大小：< 10MB
- 延迟：< 100ms
- 功耗：< 2W

推荐方案：
1. MobileNetV3 / EfficientNet-Lite
2. INT8量化
3. TFLite / PyTorch Mobile

预期指标：
- 准确率：85-90%
- 延迟：30-80ms
- 模型大小：3-8MB
```

场景2：实时监控

```
约束：
- FPS：> 15
- 延迟：< 66ms
- mAP：> 0.4

推荐方案：
1. YOLO-Nano / MobileNet-SSD
2. TensorRT + FP16
3. 批处理优化

预期指标：
- FPS：20-30
- mAP：0.42-0.48
- 延迟：35-50ms
```

场景3：云端批处理

```
约束：
- 吞吐量：> 500 QPS
- 准确率：> 95%
- 成本：< $200/day

推荐方案：
1. ResNet / EfficientNet
2. FP16 + 大batch
3. 动态batching

预期指标：
- 吞吐量：600-1000 QPS
- 准确率：95-97%
- 延迟：50-200ms
```

场景4：边缘设备

```
约束：
- 功耗：< 10W
- 内存：< 2GB
- 延迟：< 100ms

推荐方案：
1. Jetson系列专用模型
2. TensorRT + INT8
3. 内存池优化

预期指标：
- 功耗：5-8W
- 内存：0.8-1.5GB
- 延迟：50-80ms
```

场景5：IoT设备

```
约束：
- 功耗：< 1W
- 模型大小：< 1MB
- 内存：< 100MB

推荐方案：
1. TinyML模型
2. 定点运算
3. 算子裁剪

预期指标：
- 功耗：0.3-0.8W
- 模型大小：0.3-0.8MB
- 准确率：70-80%
```

🎓 优化步骤

Step 1: 明确约束

```
constraints = {
    'latency_ms': 50,        # 必须满足
    'model_size_mb': 20,     # 必须满足
    'accuracy_min': 0.90,    # 尽量满足
}
```

Step 2: 基准测试

```
baseline = {
    'accuracy': 0.945,
    'latency': 120,
    'model_size': 98
}
```

Step 3: 选择优化技术

```
if latency > 2x constraint:
    → 用TensorRT
elif model_size > 2x constraint:
    → 用INT8量化
elif accuracy_drop > 3%:
    → 用知识蒸馏恢复
```

Step 4: 逐步优化

```
原始模型
  ↓
量化 (INT8) → 测试指标
  ↓
满足约束?
  ├─ 是 → 完成 ✓
  └─ 否 → 继续剪枝 → 测试指标
            ↓
          满足约束?
            ├─ 是 → 完成 ✓
            └─ 否 → 蒸馏恢复精度
```

💡 常见问题

Q1: 延迟高怎么办?

1. 使用TensorRT/ONNX Runtime
2. 减少输入尺寸
3. 使用INT8量化
4. 换更轻量的模型架构

Q2: 模型太大怎么办?

1. INT8量化（4x压缩）
2. 剪枝（2-3x压缩）
3. 蒸馏到小模型（10x+）
4. 权重共享/霍夫曼编码

Q3: 精度掉太多怎么办?

1. QAT（量化感知训练）
2. 知识蒸馏
3. 只用FP16而不是INT8
4. 增加校准数据

Q4: 功耗太高怎么办?

1. 降低推理频率
2. INT8量化
3. 用更小的模型
4. 硬件加速器（NPU/DSP）

📏 测量代码模板

延迟测量

```
import time
import numpy as np

latencies = []
for _ in range(100):
    start = time.perf_counter()
    output = model(input)
    latencies.append((time.perf_counter() - start) * 1000)

print(f"P50: {np.percentile(latencies, 50):.2f} ms")
print(f"P99: {np.percentile(latencies, 99):.2f} ms")
```

吞吐量测量

```
start = time.time()
count = 0
while time.time() - start < 10:  # 10秒测试
    output = model(input)
    count += batch_size

qps = count / 10
print(f"吞吐量: {qps:.2f} QPS")
```

模型大小

```
import os
size_mb = os.path.getsize("model.pth") / 1024 / 1024
print(f"模型大小: {size_mb:.2f} MB")
```

内存占用

```
import torch
torch.cuda.reset_peak_memory_stats()
output = model(input)
peak = torch.cuda.max_memory_allocated() / 1024 / 1024
print(f"峰值内存: {peak:.2f} MB")
```

功耗测量

```
import subprocess

result = subprocess.run(
    ['nvidia-smi', '--query-gpu=power.draw',
     '--format=csv,noheader,nounits'],
    capture_output=True, text=True
)
power = float(result.stdout.strip())
print(f"功耗: {power:.2f} W")
```

🎯 记忆口诀

选指标，记住这几点：

1. 手机APP看大小和功耗
2. 实时系统看延迟FPS
3. 云端服务看吞吐成本
4. 边缘设备看功耗内存
5. 准确率永远要考虑
6. 权衡取舍是关键

📚 进一步学习

推荐工具

- TensorRT – NVIDIA GPU优化
- ONNX Runtime – 跨平台推理
- TFLite – 移动端部署
- OpenVINO – Intel优化

推荐阅读

- 《Efficient Deep Learning》
- 模型压缩综述论文
- TensorRT Best Practices

快速查找你关心的指标，找到最佳优化方案！