

Google Colab Pro+を活用した 商用レベルの物体検出モデルの構築

LibreYOLO (RF-DETR) を用いたセットアップ・
運用手順とGCP本番移行へのロードマップ

コア・エンジンの選定：商用化の障壁と精度の壁を突破する

LibreYOLO（商用利用の完全な自由）



完全なMITライセンス（AGPLの制約を完全に排除）

閉鎖的ソースや組み込み商用製品への安全な統合が可能

YOLOv9, YOLOX, D-FINE等を単一APIで扱う統合環境

RF-DETR（次世代のエンドツーエンド精度）



NMS（非最大値抑制）のポストプロセスが不要なTransformerアーキテクチャ

グローバルな受容野により、密集した状況や重なり合い（オクルージョン）に極めて強い

L4 GPU上で約20 FPS（50msレイテンシ）の実用リアルタイム性能

「商用化の障壁」と「精度の壁」を同時に突破するアーキテクチャ。

低コスト環境の構築: GCPからColab Pro+へのマッピング

GCP本番環境		Colab Pro+環境
Storage (データ・ログ保存): Cloud Storage (GCS)	→	Google Drive + ローカル (/content/)
Compute (演算リソース): Spot VM / A100 GPU	→	割り当て型ハイエンドGPU (A100)
Environment (環境依存性): カスタムコンテナ (ゴールデンイメージ)	→	セル冒頭の動的セットアップスクリプト

プロトタイプ開発を最速で回すため、GCPの概念をColabの機能で代替・再現する。

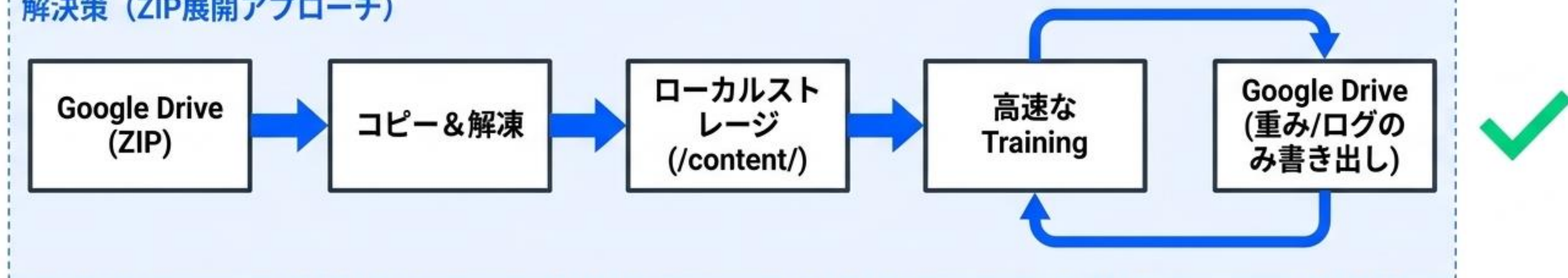
ストレージ設計とI/Oボトルネック対策

課題: Drive上の数万枚の画像を直接読み込みながら学習すると、I/O速度がボトルネックとなり学習時間が大幅に遅延。



OK Path

解決策 (ZIP展開アプローチ)



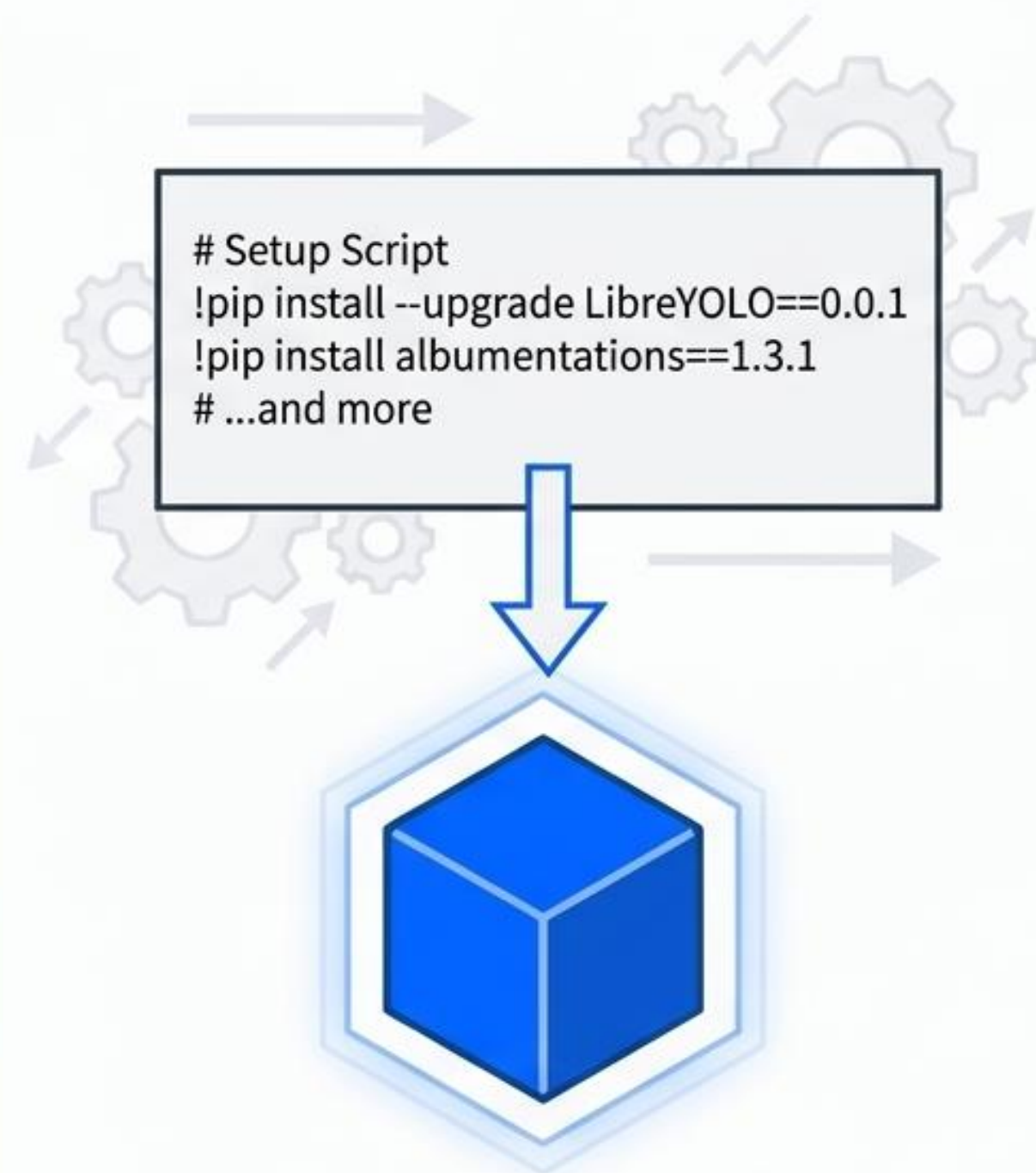
環境構築：動的セットアップによる「ゴールデンイメージ」の代替

Colabの現実:

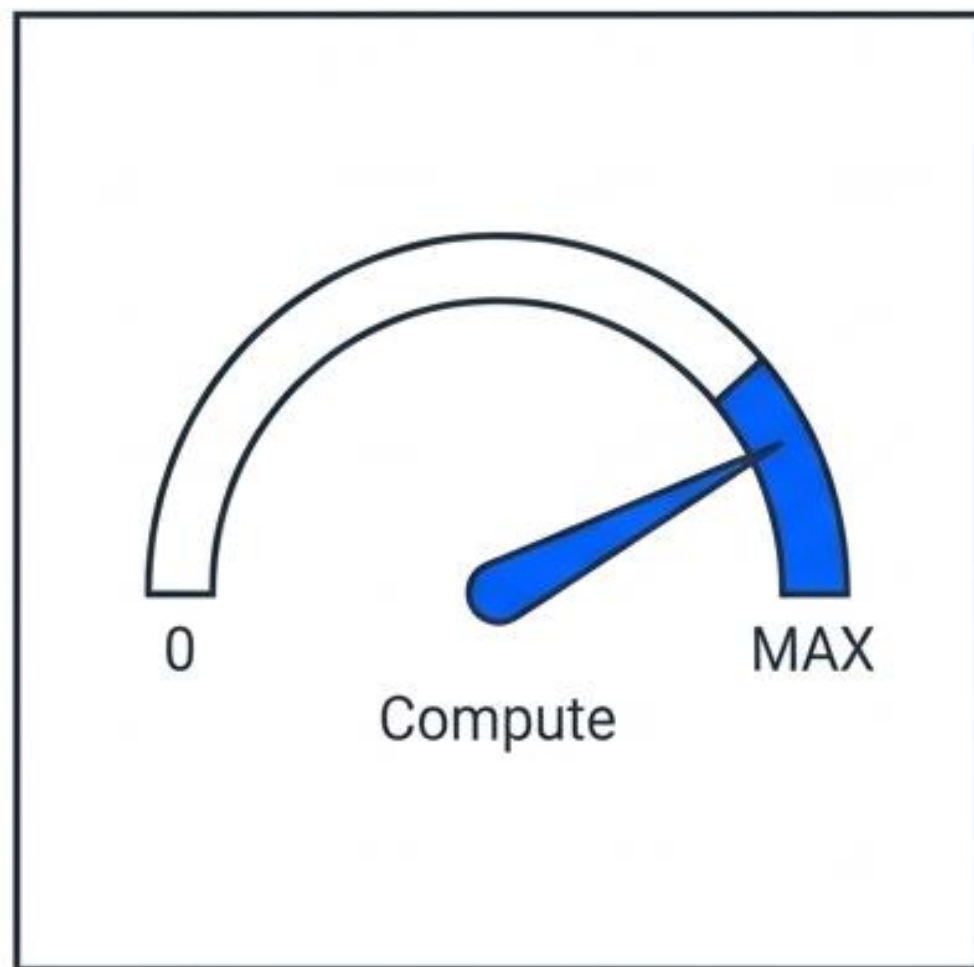
Googleによる裏側でのランタイム（CUDAや標準ライブラリ）の定期アップデートにより、昨日稼働したコードが突然エラーを吐くリスクが常在する。

必須の対策:

- ノートブックのセル冒頭に、毎回完全な環境を構築するセットアップスクリプトを記述（!pip installの明示的実行）。
- LibreYOLO、Albumentations（データ拡張）などの依存関係バージョンを厳密に指定し、環境を動的に「固定化」する。

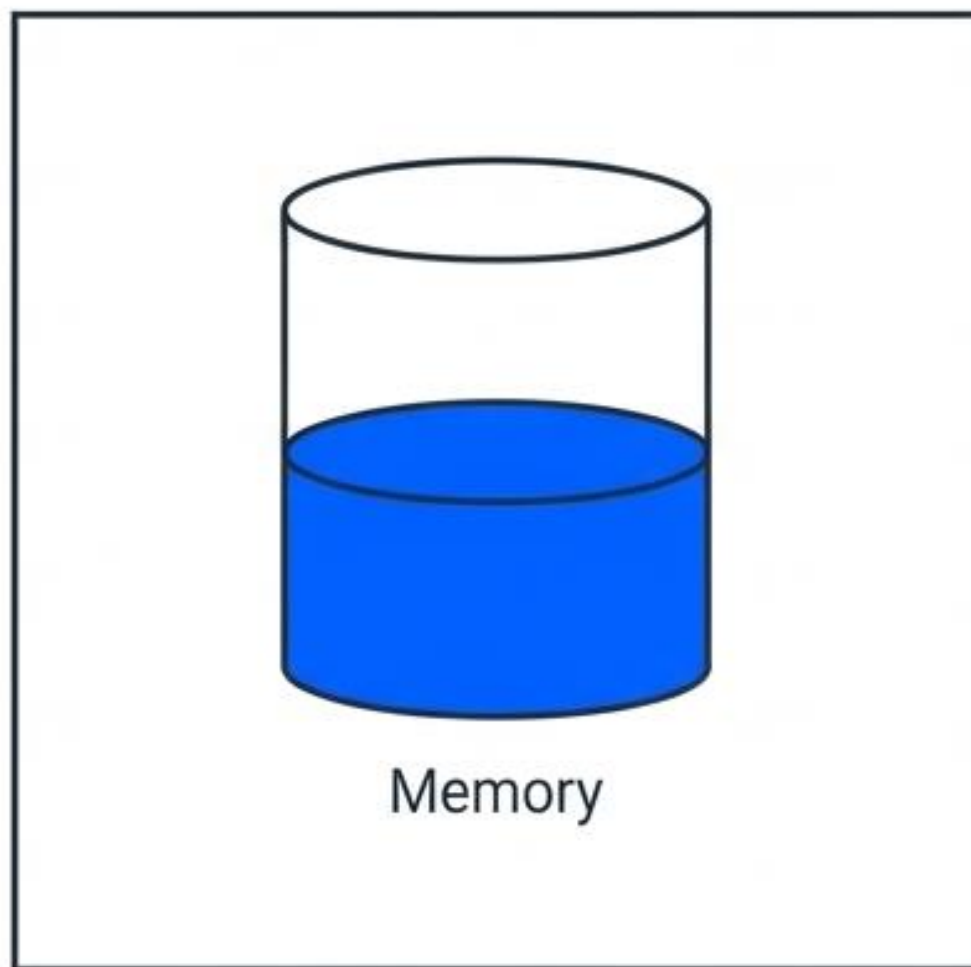


GPUリソースの極大化と学習効率化



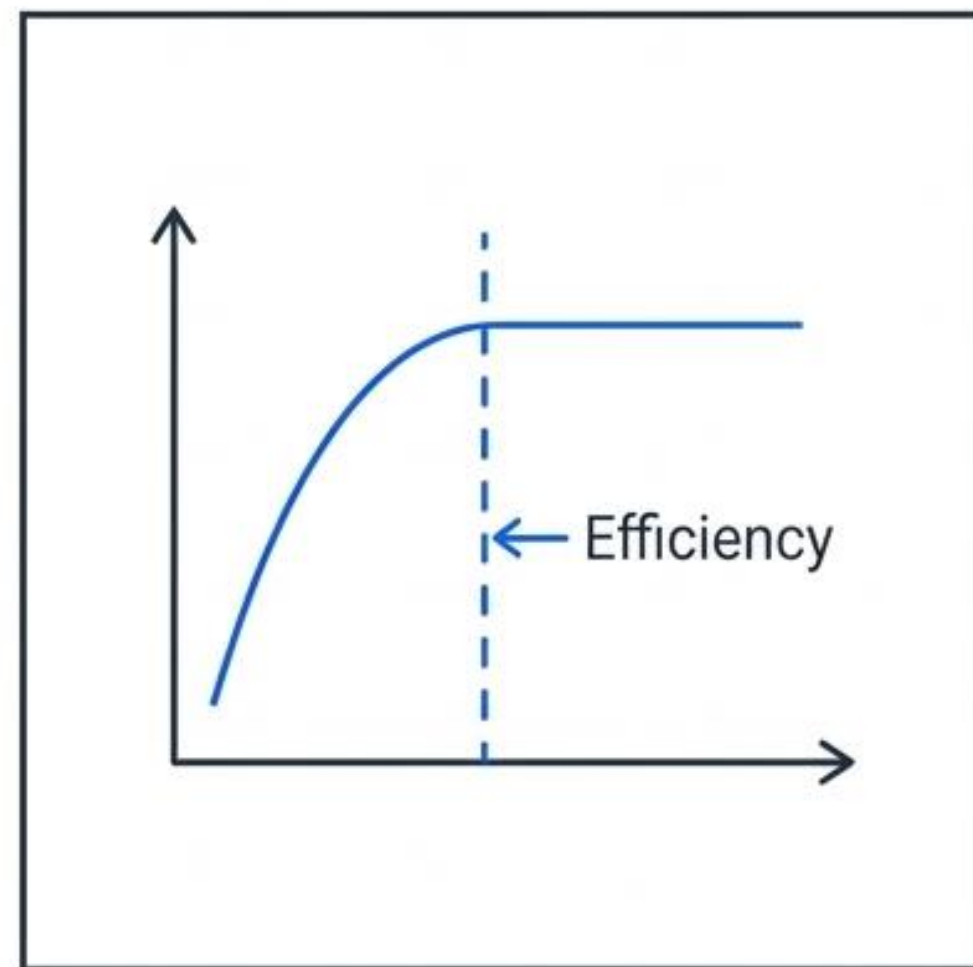
1. A100 GPUの積極活用

Colab Pro+のハイエンドGPU割り当てを最大限に活用し、大規模Transformerモデルの計算を処理。



2. FP16混合精度学習 (AMP)

LibreYOLO組み込みのAMPを適用し、GPUメモリ消費を半減。これによりバッチサイズを最大化し、ハードウェアのコスト効率を限界まで高める。



3. Early Stopping (早期終了) の徹底

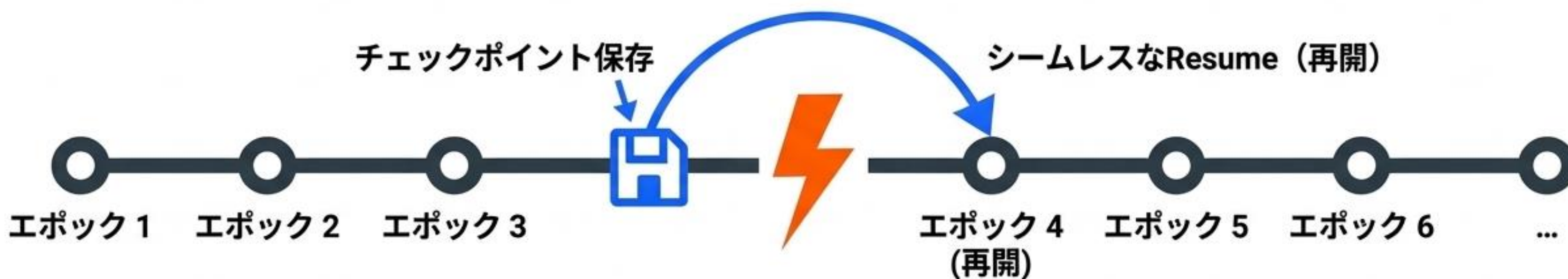
検証メトリクスを常時監視。20~30エポック連続で精度向上が停滞した場合、自動的に学習を打ち切り、無駄なリソース消費をカット。

セッション切断対策：耐障害性の組み込み

リスク: Colab特有の「90分ルール（非アクティブ切断）」や最大稼働時間制限による強制終了。

システム設計（中断を前提とするアーキテクチャ）：

- **細かなチェックポイント保存:** 学習ステップ（エポック）ごとに、Driveの永続ディレクトリへ重みの状態を退避。GCPのSpot VM運用と同等の耐障害性を確保。
- **Resume（再開）ロジックの実装:** 実行開始時に最新の重みファイルを自動検知し、中断地点からロスなく学習をシームレスに再開。



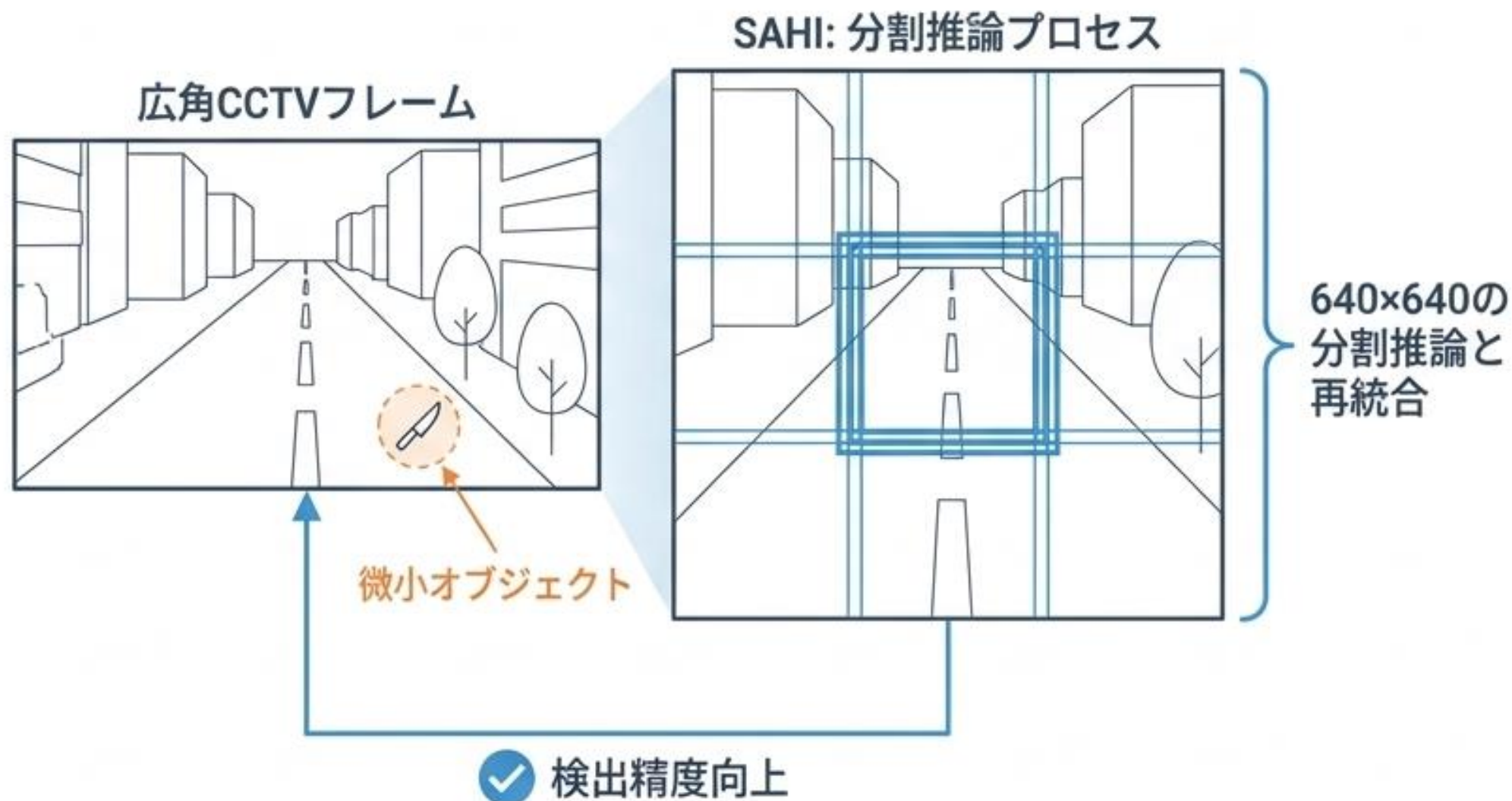
現場の精度課題を突破する：微小オブジェクトの検出（SAHI）

課題:

広角カメラ映像では、刃物やスマートフォンなどの重要オブジェクトが極小となり、通常の1パス推論では見落としが発生する。

SAHI (Slicing Aided Hyper Inference) の導入:

- ベースモデル（RF-DETR）のアーキテクチャは一切変更しない。
- 高解像度の入力フレームを、オーバーラップ（例：640×640タイル、20%の重なり）させながら分割して推論。
- 推論結果を再統合することで、微小オブジェクトの検出率を劇的に向上させる。



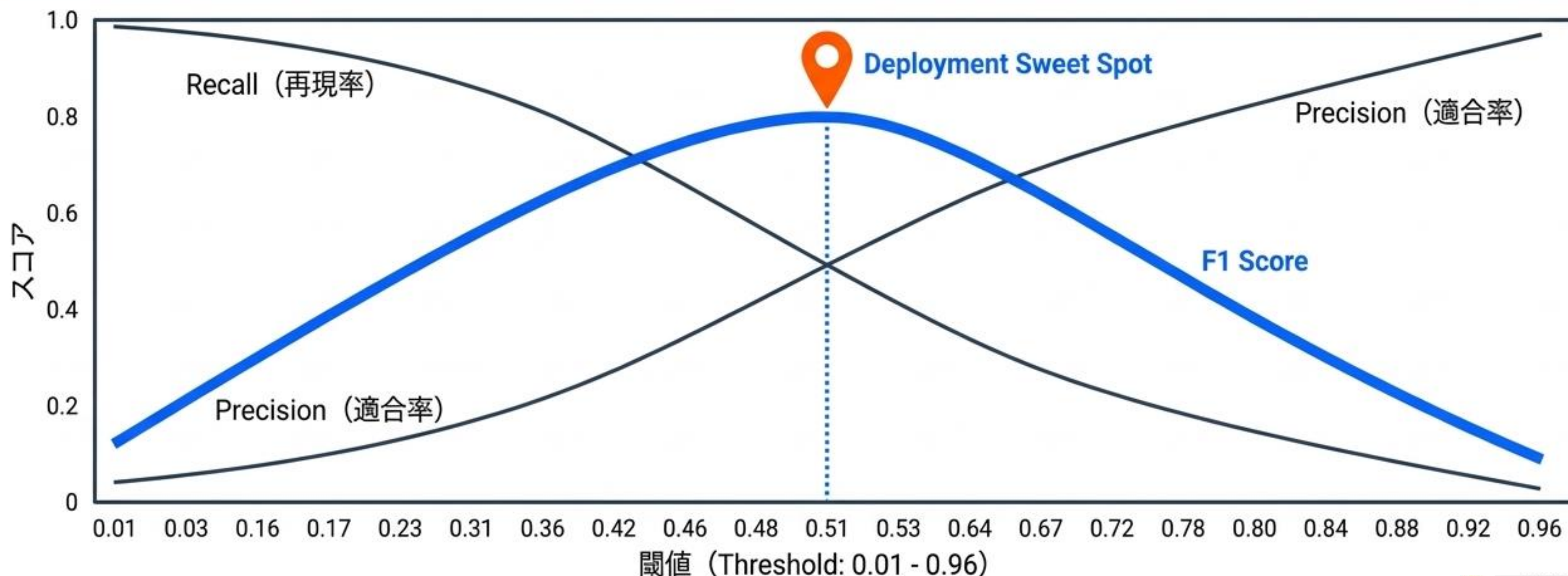
閾値の最適化：F1スコアによる自動キャリブレーション

固定閾値の罠:

クラスごとに検出難易度が異なるため、静的な信頼度（Confidence）閾値では過検知と見落としのバランスが崩壊する。

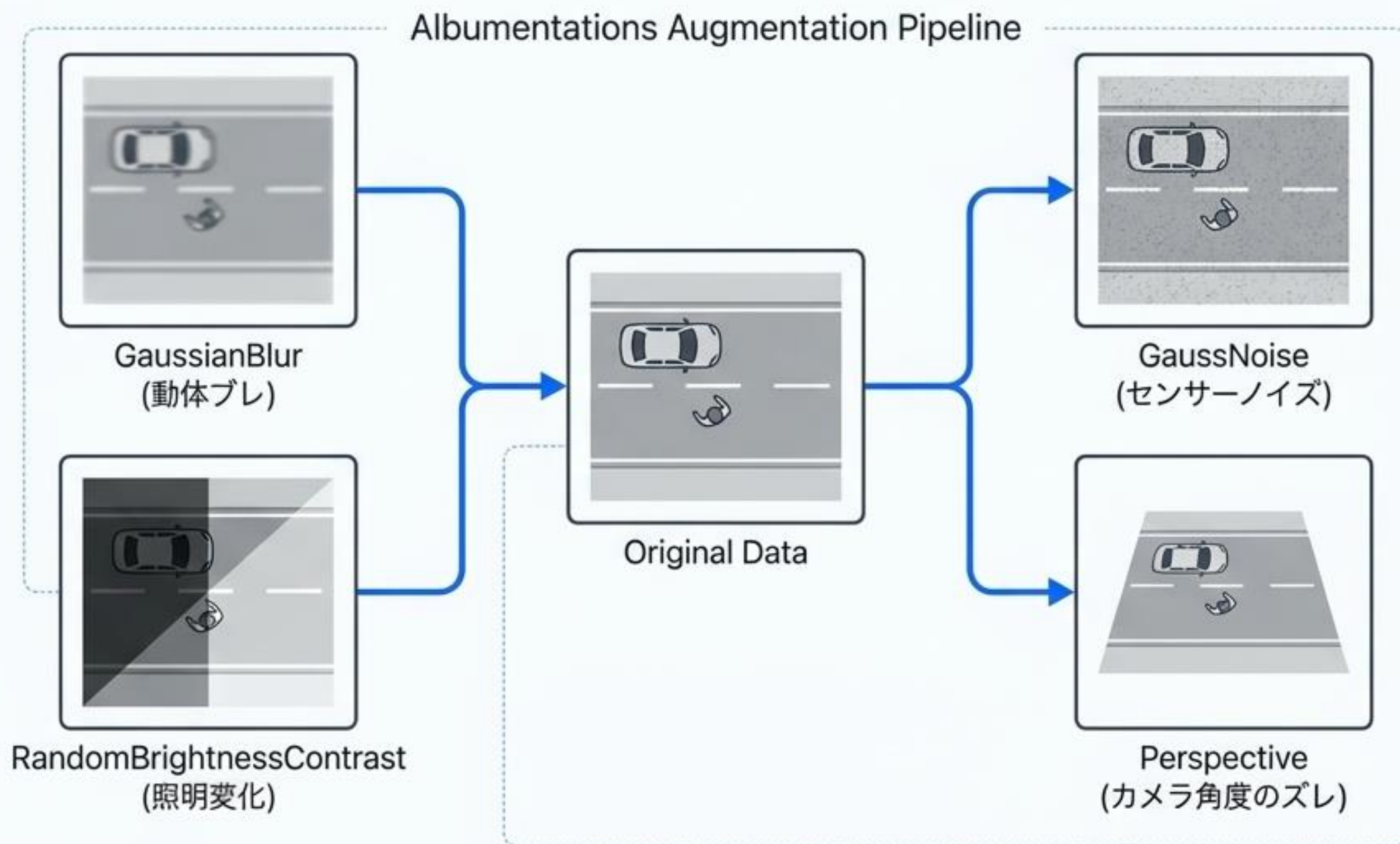
最適解の自動算出:

0.01から0.96の範囲で閾値をスweepし、PrecisionとRecallの調和平均である「F1スコア」が最大となる閾値を数学的に特定しデプロイする。



データセットの壁：AI生成より「強靱な拡張」

俯瞰視点（CCTV角度）のデータは極端に少なく、AIによる合成データ生成はスケールコストが高く非現実的。
Albumentationsを用いた物理ベースのパイプライン内データ拡張が必須。



Colab Pro+運用の限界とリスク分析

1. GPU割り当ての不確実性

A100が常に確保できる保証はなく、混雑状況により下位GPUへダウングレードされるリスク。

2. I/Oオーバーヘッドの蓄積

セッション開始のたびに発生する「ZIP解凍・ローカルコピー」の待機時間が、反復試行時のタイムロスとなる。

3. 環境のサイレントな変化

Googleのバックグラウンドアップデートによる、予期せぬ依存関係エラーの発生。完全なコンテナ固定が不可。

4. スケールアップ・自動化の限界

ノートブック前提のため、複数モデルの同時並行学習や完全なバッチ処理パイプラインの構築に不向き。

本番環境への移行ロードマップと判定基準

Phase 2: 本格的な精度向上 & スケール

環境: GCP + GCS

移行後: GCSへの直接ルーティング、カスタムコンテナによる完全な環境固定、Spot VMによるコスト最適化と自動化ジョブの実行。

[移行のトリガー条件]: データセットが数万枚規模に到達した時。 / 100%のUptime、または複数モデルの同時並行学習が必要になった時。

Phase 1: 概念実証 & 初期学習

環境: Google Colab Pro+

対象: コードの動作確認、小規模データでのテスト、ハイパーパラメータの初期探索。

まとめ：商用レベルモデル構築へのステップ

1. エンジンの選定 (LibreYOLO × RF-DETR):

MITライセンスによる商用化の自由と、Transformerによる最高水準の推論精度を両立。

2. プロトタイプ環境の極大化 (Colab Pro+):

I/Oの最適化（ローカル解凍）と耐障害性（Resumeロジック）を実装することで、強力な初期開発エンジンへと昇華。

3. シームレスな本番移行:

環境の構造的な限界を理解し、適切なデータ規模の閾値でGCPカスタムトレーニング環境へスケールアップする。

