15. 組み込み

組み込み（２）

目的と目標

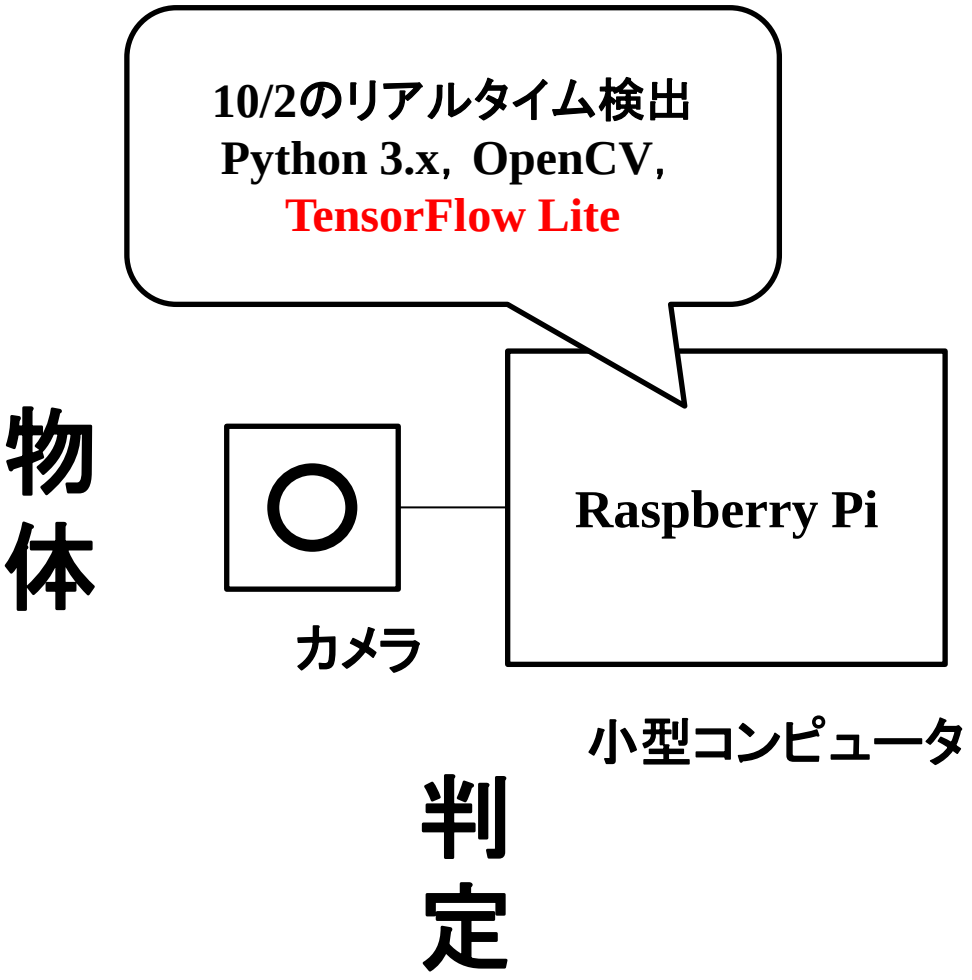
目的：機械学習を搭載するエッジコンピューティングへの理解（２）

- Raspberry Pi上にTensorFlow Lite環境を構築できる
- Google colabで作成した学習結果（重みファイル）をTensorFlow Liteに変換できる
- TensorFlow Lite モデルメーカーを使ってモデルを作成できる

目標：Raspberry Pi上でのリアルタイム検出の実現（Tensorflow Lite版）

- 演習：10/2に実施したリアルタイム検出をTensorflow Liteを使ってRaspberry Pi上で実現

カメラ＋小型コンピュータ



演習：10/2に実施したリアルタイム検出をTensorFlow Liteを使ってRaspberry Pi上で実現

詳細は別紙の演習資料

TensorFlow Lite

モバイル デバイス、組み込みデバイス、IoT デバイスで TensorFlow モデルを実行できるようにするツールセット

TensorFlow Lite インタープリタ:

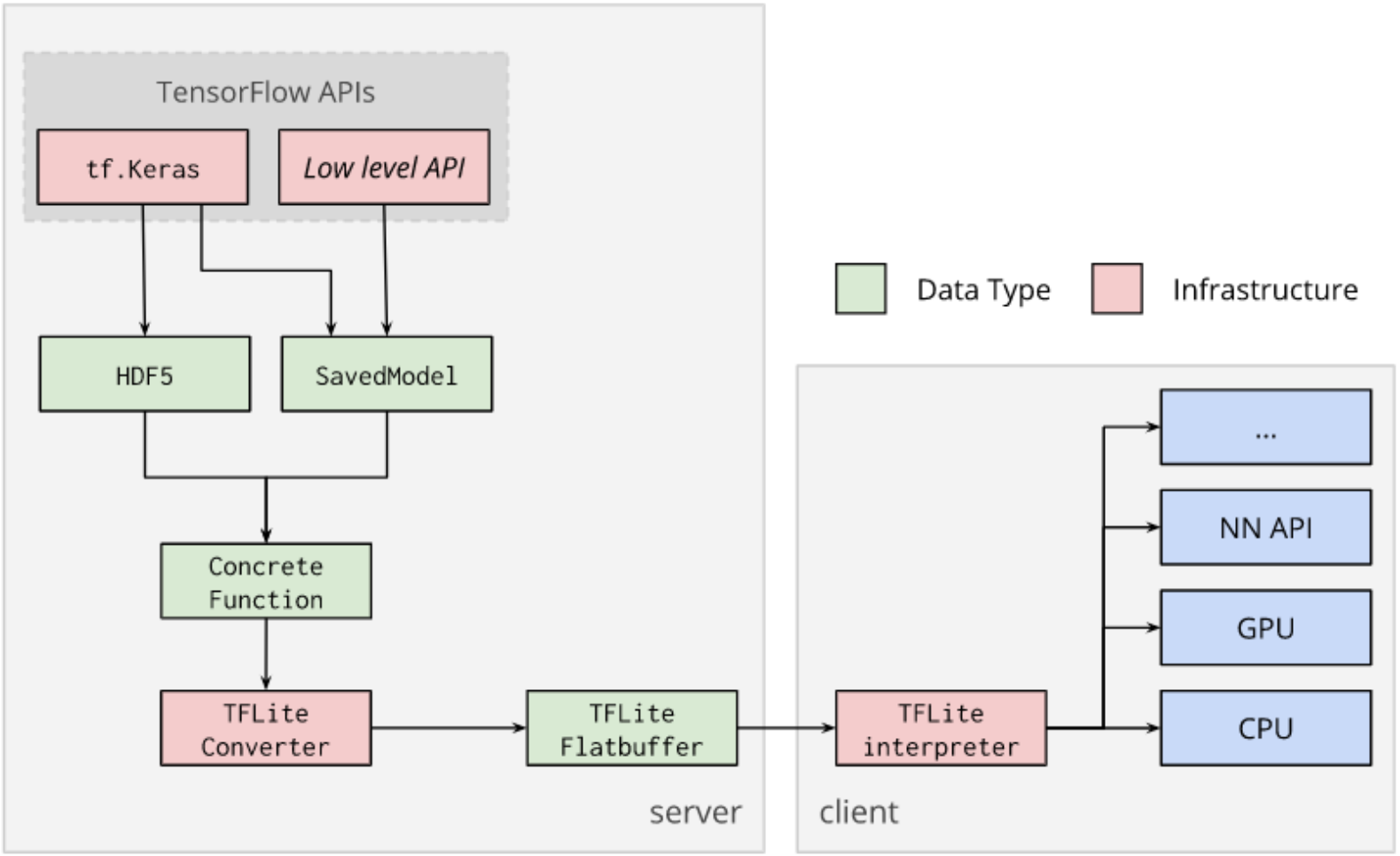
スマートフォン、組み込み Linux デバイス、マイクロコントローラなどのさまざまなハードウェア上で最適化されたモデルを実行できるようになるインタープリタ

TensorFlow Lite コンバータ:

TensorFlow モデルをインタープリタで使用するための効率的な形式に変換し、最適化を行うことができるコンバータ

https://www.tensorflow.org/lite?hl=ja

TensorFlow Lite



https://www.tensorflow.org/lite/convert?hl=ja

TensorFlow Liteモデルメーカー

概観

TensorFlow Liteモデルメーカーライブラリは、カスタムデータセットを使用してTensorFlow Liteモデルをトレーニングするプロセスを簡素化します。転移学習を使用して、必要なトレーニングデータの量を減らし、トレーニング時間を短縮します。

サポートされているタスク

モデルメーカーライブラリは現在、次のMLタスクをサポートしています。

サポートされているタスク	タスクユーティリティ
画像分類ガイド	画像を事前定義されたカテゴリに分類します。
テキスト分類ガイド	テキストを事前定義されたカテゴリに分類します。
質問回答ガイド	与えられた質問に対する特定のコンテキストで答えを見つけます

https://www.tensorflow.org/lite/guide/model_maker?hl=ja

転移学習

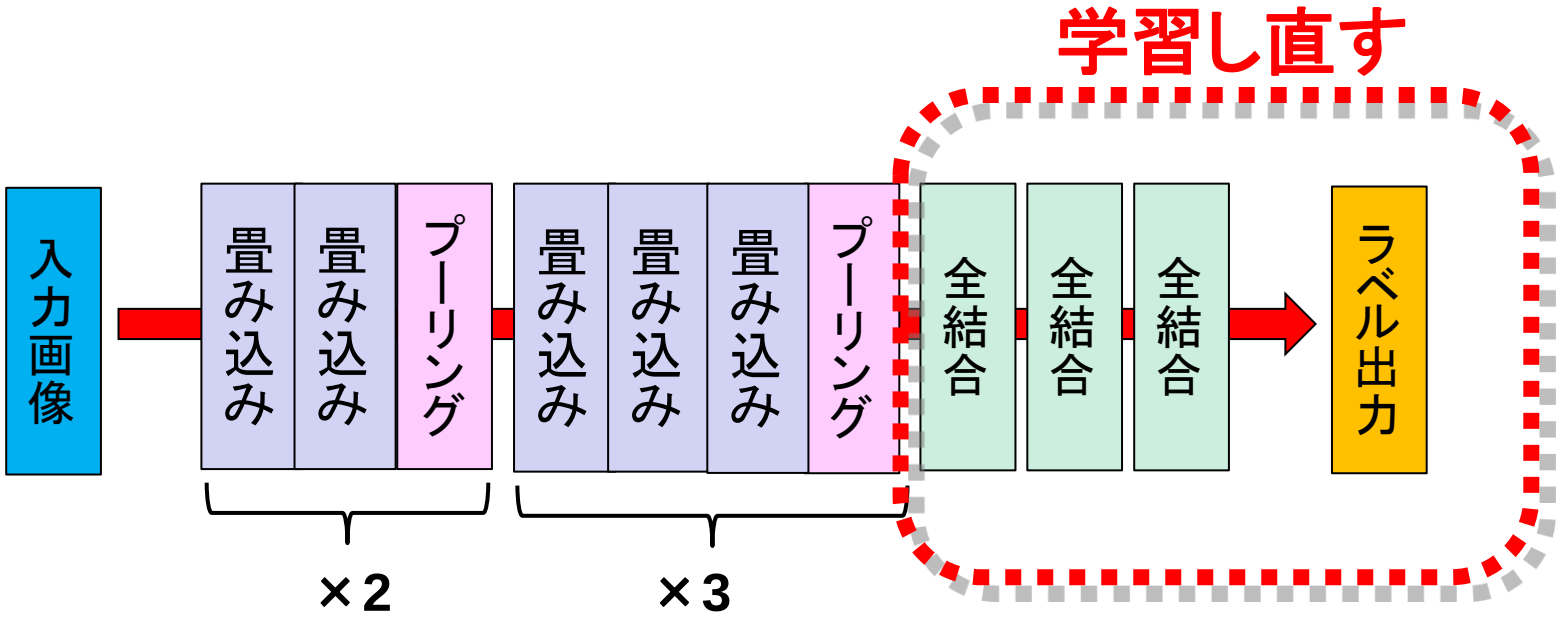
高度な深層学習モデルには大量のパラメータ（重み）があり、それらを何もないところから訓練するには大量の計算リソースのデータが必要

転移学習は関連するタスクについてすでに学習済みのモデルの一部を取り出し、新しいモデルの中で再利用することで、そのようなリソースの大部分を省略するテクニック

https://www.tensorflow.org/js/tutorials/transfer/what_is_transfer_learning?hl=ja

転移学習

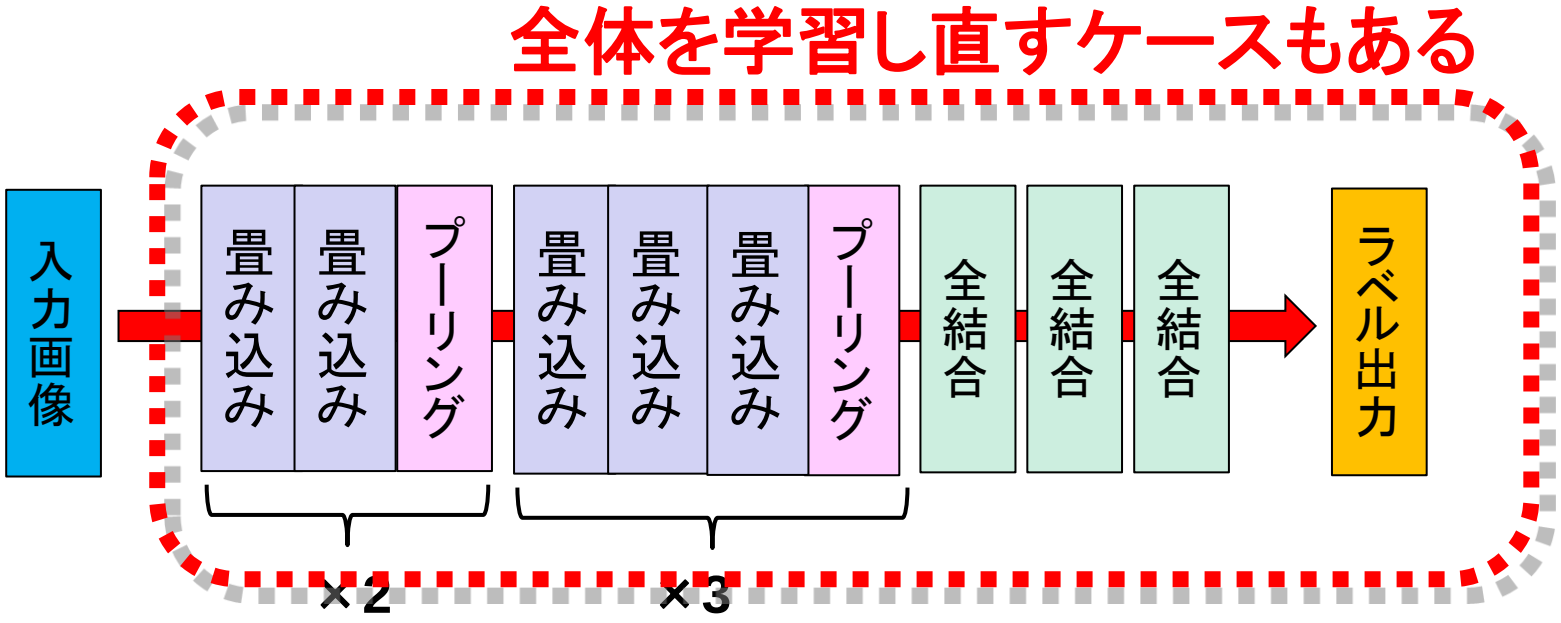
VGG16



Karen Simonyan, Andrew Zisserman, "**Very Deep Convolutional Networks for Large-Scale Image Recognition**", https://arxiv.org/abs/1409.1556, (2014)

転移学習

VGG16



Karen Simonyan, Andrew Zisserman, “**Very Deep Convolutional Networks for Large-Scale Image Recognition**”, https://arxiv.org/abs/1409.1556, (2014)

CS231n Convolutional Neural Networks for Visual Recognition

When and how to fine-tune? How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset (small or big), and its similarity to the original dataset (e.g. ImageNet-like in terms of the content of images and the classes, or very different, such as microscope images). Keeping in mind that ConvNet features are more generic in early layers and more original-dataset-specific in later layers, here are some common rules of thumb for navigating the 4 major scenarios:

1. *New dataset is small and similar to original dataset.* Since the data is small, it is not a good idea to fine-tune the ConvNet due to overfitting concerns. Since the data is similar to the original data, we expect higher-level features in the ConvNet to be relevant to this dataset as well. Hence, the best idea might be to train a linear classifier on the CNN codes.
2. *New dataset is large and similar to the original dataset.* Since we have more data, we can have more confidence that we won't overfit if we were to try to fine-tune through the full network.
3. *New dataset is small but very different from the original dataset.* Since the data is small, it is likely best to only train a linear classifier. Since the dataset is very different, it might not be best to train the classifier form the top of the network, which contains more dataset-specific features. Instead, it might work better to train the SVM classifier from activations somewhere earlier in the network.
4. *New dataset is large and very different from the original dataset.* Since the dataset is very large, we may expect that we can afford to train a ConvNet from scratch. However, in practice it is very often still beneficial to initialize with weights from a pretrained model. In this case, we would have enough data and confidence to fine-tune through the entire network.

https://cs231n.github.io/transfer-learning/

いつどのようにFine-Turningするか？

新しいデータセットの「大きさ」と「類似度」が重要

１）新しいデータセットが小さい・元のデータセットに似ている
過学習が懸念される。最後の層の分類器を訓練するのがお勧め。

２）新しいデータセットが大きい・元のデータセットに似ている
全体をFine-Turningしても過学習しない可能性が高い。

３）新しいデータセットが小さい・元のデータセットに似ていない
分類器を訓練するのが良い。ネットワークの出力側よりも入力側のどこかの特徴量を使って訓練するのが良い。

４）新しいデータセットが大きい・元のデータセットに似ていない
ゼロから訓練しても大丈夫。訓練済みモデルの重みでネットワークを初期化すると有効であることが多い。

演習：targetを複数にしたモデルの作成

プログラム15d_TFL_ModelMaker.ipynbをもとに、targetを複数にして検出するモデルを各自で作成し、15e_detecRoi_mMaker.pyで動作するか確認してみましょう。

なお、フォルダが異なっていれば画像の名前は同じものがあっても特に問題ありません。

例えば、target1/target.1.jpgとtarget2/target.1.jpgは画像名が同じですが、フォルダが異なるので特に問題ありません。

（前回）Extra演習

extraA）TensorFlowを使った物体認証の復習1

各自でリアルタイム検出を行うことができるか、新しい物体を選んで各自で実施してみましょう

extraB）TensorFlowを使った物体認証の復習2

microSDカードにraspberry pi OSのイメージを書き込んで、各自でインストールの最初からセットアップができるか実施してみましょう

extraC）TensorFlowを使った物体認証（VGG-16版1）

10/02に実施した「VGG-16演習」を各自でラズベリーパイ上に再現してみましょう

extraD）TensorFlowを使った物体認証（VGG-16版2）

10/02に実施した「VGG-16演習」の機能を今回実施したプログラムに組み込んでみましょう。※リアルタイム検出の中身をVGG-16に設定

最終課題に向けて

これまでの内容を活かした、各自のアイディアによる自由製作です

例）

- 画像判定、動画判定器（非リアルタイムでもよい）
- 組み込み機を利用した何かリアルタイム判定機、あるいはWebサービス
- 自然言語処理を応用したネガティブ、ポジティブ判断、bot、類似判定
  （BERTは学習時間が大変かも。Word2vecによるコサイン類似など）

また、最終課題の制作物に対して、サービスの概要・仕様書を書いて下さい。
外部仕様、内部仕様、REST APIなどのしっかりとしたものではなく、他者がさっと目を通して概略が理解できる範囲の物でかまいません