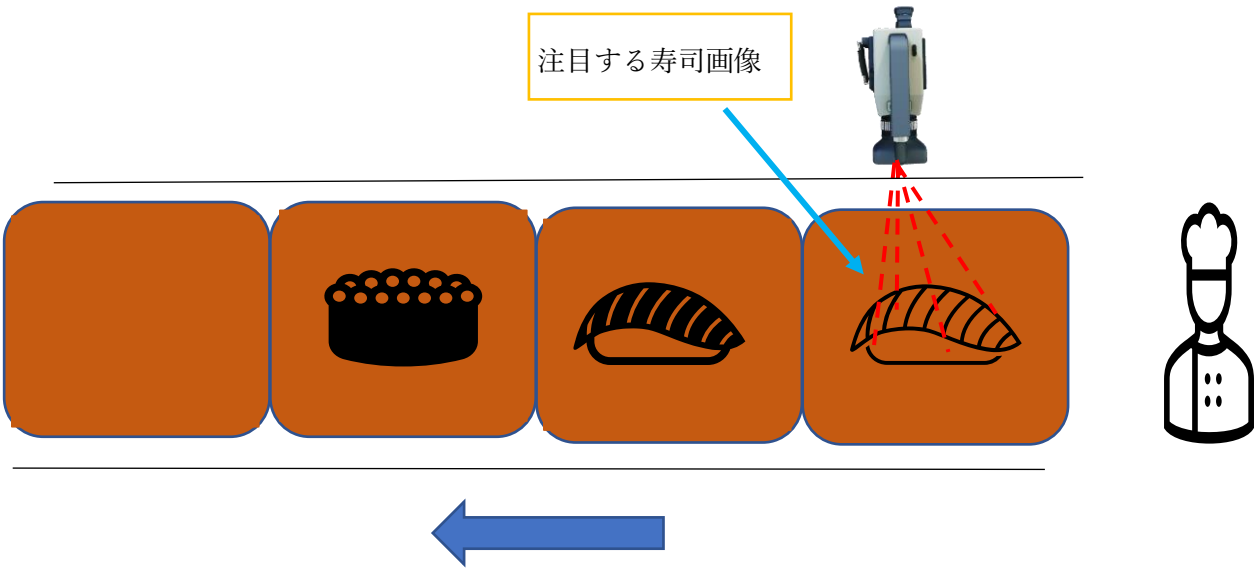
1. テーマ
飲食店におけるメニューと品質の最終確認

2. 画像認識システムの概要
a. ユーザ＆利用シーン
・ユーザ: 飲食店の調理者(今回は回転寿司店)
・利用シーン: 料理(寿司)をつくり、客に出す前の商品チェック.

b. 課題解決イメージ
・現状の問題: 特に回転寿司などの注文回転が速い飲食店では、商品の個数を間違えたり、異なる商品を出してしまったり、形が崩れた状態で出してしまったりという人為的ミスが頻繁に起こっている. また、これらのことを防ぐために、店員が入念にチェックすることで、作業効率が下がってしまっている.
・課題解決: メニューが間違っていないか、形が崩れていないかを判断するシステムによる課題解決を目指す. このシステムを導入することによって、人為的ミスがゼロとなり、品質が向上し、厨房の店員はメニューを作ることのみに専念でき、作業効率を上げることができると考えられる.

3. システムの詳細
a. システムの全体像
[実際のシステム]
客から受けた注文と料理人がレーンに乗せた寿司を照合し、全ての商品が合っているかつ品質が保てていたらそのまま商品を出し、どちらかが満たされていなかったら調理者へ警告を出し、調理場へ戻す.
[今回実装するシステム]
今回は組み込みシステムではないため、商品画像と注文リストを入力とし、OK か NG の出力のみ行う.(OK: 「そのまま商品を出す」に該当. NG: 「調理者に警告を出し、調理場へ戻す」に該当.)
[システムの流れ]
① 客からの注文の入力を受け入れ状態にする.(標準入力)
注文が入力されたら、注文リストに格納する.(注文は順不同)
② 料理人に注文に応じた寿司をレールの上に乗せてもらう. 今回は寿司の画像を標準入力で入力してもらう.(寿司画像は順不同)
注文リストの数と入力画像の数が等しくないとき、エラーを出力し、再び画像を入力させる.
③ 注目する寿司画像を、寿司が崩れているか否かの 2 つのラベルに分けて機械学習で判定する. 崩れている(×)と判定されたら NG を出力し、全ての入力画像を初期化して再度 ②

に戻る．崩れていない(〇)と判定が出たら④に進む．
④注目する寿司画像を、どの種類に該当するか機械学習で判定する．判定結果に該当する注文を配列から取り除いていく．配列にない判定結果が現れたら NG を出力し、さらに入力画像を初期化して再度②に戻る．
⑤ ③,④を注文数だけ繰り返す．無事全てマッチし終わったら⑥へ進む．
⑥ OK を出力する．注文リスト、入力画像を初期化して、①に戻る．



b. 実現したい機能
・このシステムの大本となる寿司の種類を認識する機能（機械学習を用いる.）
寿司画像を入力として、寿司の種類を出力とする．

マグロ

イカ

・崩れていないか確認する機能（機械学習を用いる.）

形が崩れているか否かの 2 種類のラベルに分け、学習を行う.
崩れていない場合は〇を、崩れている場合は×を出力する.

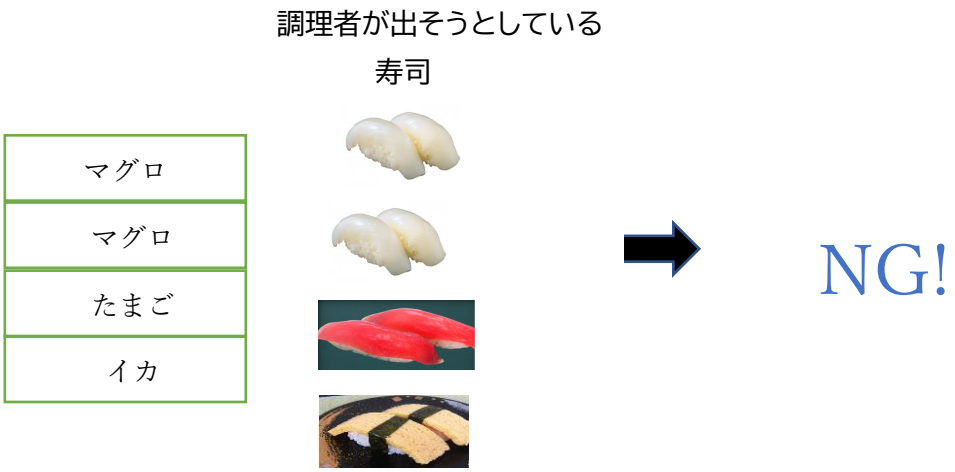
・複数の注文を照合できるように、注文内容と調理者が出す商品の認識結果をマッチできる機能
[システムの流れ]④~⑥に該当.

注文リスト

マグロ
マグロ
たまご
イカ

調理者が出そうとしている
寿司

OK!



c. 想定される技術的課題

[解決済み]

・インターネットの寿司画像もしくは、回転寿司店の寿司を撮った画像を用いようと思っているが、法律的に使用していいのか不安.

→自分で撮影した画像や著作権フリー画像に置き換えた.

・(3.b)における形崩れラベルについて、「形崩れ」と一括りにするか、「マグロ形崩れ」などとそれぞれの商品の形崩れとするか迷っている.

→最初に形崩れか判定し、その後寿司の種類を判定するという 2 段階に分けることとした.

[未解決・新たな課題]

・ノイズを許してしまうと、異物がついているときに OK としてしまう恐れがあるので、水増しの際にノイズが使えない?

・崩れている度合いによって、正規のものとするか、崩れているとするか判断するため、訓練データが大量に必要?

・寿司画像を標準入力で入力してもらう予定であるが、どのように実装してよいか不明.

4. 開発イメージ

a. 入力画像、ラベル数

・入力画像: 商品として出す前の寿司画像(今回はテストデータ)

・ラベル数: 崩れているか否かの 2 種類、寿司を判定するために寿司の種類数 N だけラベルが必要.　計 N+2.

b. 開発手順

①各商品に対する見本をそれぞれ 30 枚程度撮影し、各商品のフォルダに入れ、正解ラベ

ルを付ける.
あまりにも判定が厳しいと、逆に店舗の効率が落ちるため、客に出せる程度に崩れた画像も用意し、「形崩れ」でないと判定する.
②この 30 枚をテストデータ:訓練データ=1:4 に分け、各商品のテストデータ、訓練データのフォルダに入れておく.
③訓練データの水増しを行う.
ImageDataGenerator による自動水増しを目指し、できなければ imageJ で行う.
拡大、縮小、回転、などによる水増しをする.
回転について、寿司屋の検知の場所は定位置であると考えられるため、鉛直方向の回転はせず、水平方向の回転のみ行う.
④訓練データを過学習しない程度に学習させ、テストデータで、精度=1.0 となるような CNN を構築する.
⑤複数の注文が来た時に対応できるように、3.b(実現したい機能)の 3 つ目を実装する.
注文配列を用意し、複数の注文を配列に格納する. CNN による寿司画像の認識結果に該当する注文を配列から取り除いていく. 途中で配列にない認識結果が現れたり、最終的に注文リストに注文が残っていたりしたら NG、無事全てマッチし終わったら OK を出す.
⑥注文と客に出す前の画像(順不同)を入力とし、CNN で 1 枚ずつ順に判定し、最終的に OK か NG を出力させるようにする. 回転寿司店では、一度の注文上限を 5 としている店舗が多いため、注文入力数の上限を 5 とする.