2015年度 卒業論文

# 機械学習を利用した<br>洋服管理アプリケーション開発

大阪産業大学 工学部 情報システム工学科
情報教育システム研究室

12H016 大沼里奈

# 目次

# 1　はじめに

洋服は種類と数か多く、更に流行か数年周期て変わるため新たに洋服を購入する機会か多ため、所持する洋服は増えていき管理か難しくなる。これら を解決するため、近年普及している iOS 端末を利することて、何時どこでも所持している洋服を確認てきるような iOS アプリケーションを開発する。更に洋服の種類を自動で分類し、どの種類の洋服かどれだけ所持しているかを一目で確認できるシステムの開発を行う。

第 2 章では本研究の目的について述べる。第 3 章では本研究で開発するために使用するシステムを述べる。第 5 章では本研究で開発したシステムの概要を述べる。第 6 章には研究の成果とともに今後の課題についてまとめる。

## 2 目的

本システムではディープラーニング [1] による画像認識モデルを簡単に作成する Web プラットホームの Labellio [2] を利用することで、簡単に且つ正確な画像認識機能の実装を行うことを目標にしたものである。

実装は普段持ち歩いている iOS でのアプリケーション開発を行い、どこでも手軽に自身が所持している洋服を確認することができるシステムを開発する。

# 3　本研究で利用するシステム

本研究で開発するシステムで利用するシステムの概要を述べる。

## 3.1　ディープラーニング

ディープラーニング [1] は、ニューラルネットワークの多層化、特に 3 層以上のものに対し、1990 年代に進められた脳、特に視覚野の研究や、One Learning Theory、ブルーノ・オルスホーゼンによるスパース・コンピューティング理論を基にしたアルゴリズムが実装されたものを指す。これに画像などのデータを入力すると、情報が第 1 層からより深くへ伝達されるうちに、各層で学習が繰り返される。この過程で、これまではデータ・サイエンティストと呼ばれる専門家が設定していた、概念を認識する特徴量と呼ばれる重要な変数を自動で発見できる。これは、人間の脳の構造をソフトウェア的に模倣し、人間が関与せずに学習を進める、いわゆる無教師学習の一つである。特徴量とは、問題の解決に必要な本質的な変数であったり、特定の概念を特徴づける変数である。この特徴量を発見できれば、あらゆる問題を解決につながったり、パターン認識精度の向上や、フレーム問題の解決につながると期待されている。この特徴量の自動的な学習は、ディープラーニングが従来の機械学習と決定的に異なる点であり、ピクセルから線、線からパーツ、パーツから全体の概念、というように、階層的に特徴が抽出される。しかし、ディープラーニング・システムが何故特定の特徴量を導くことができるのかは、解明されていない。

## 3.2　Labellio

Labellio とは株式会社 AlpacaDB [3] が開発したディープラーニングによる画像認識を可能にする Web プラットホームである。この Labellio を利用して Web 上で作成した学習モデルは Caffe を使って個人サーバで実際にモデル動かすことができる。図 1 に Labellio のログイン画面を示す。GitHub か Google のアカウントでログインが可能である。

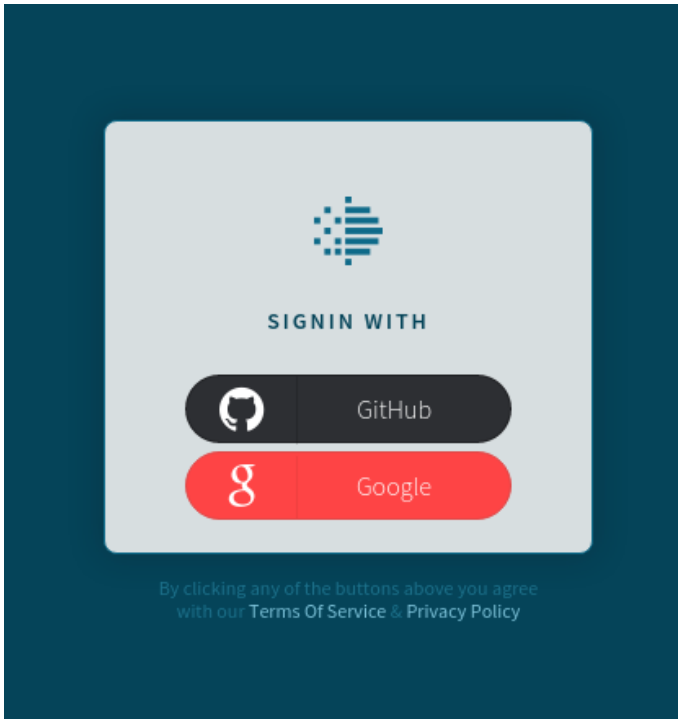


図 1　Labellio ログイン画面。GitHub のアカウントまたは Google のアカウントを用いてログインすることができる

### 3.2.1 モデルファイルの内容

Labellio で作成したモデルファイルの一覧を図 2 に示す。

| ファイル名 | 説明 |
|---|---|
| labellio.json | 設定ファイル |
| caffemodel.binaryproto | caffeモデルファイル |
| mean.binaryproto | meanファイル |
| deploy.prototxt | caffeネットワーク定義ファイル |
| labels.json | ラベル情報 |

図 2　モデルファイル一覧

## 3.3 Flask

Flask [4] とは Python 用の軽量な Web アプリケーションフレームワークである。Flask はどんな Web アプリケーションにも適合するよい基盤を作るための設計としてデータベース抽象化レイヤやフォーム値の検証などの機能を持たない。この Flask を利用し、画像認識と画像保持をする。

## 3.4 Caffe

Caffe [5] はディープラーニングのフレームワークの一つである。ディープラーニングのフレームワークは多数あるがその中でも画像処理に最適化されたフレームワークである。

## 3.5 動作環境

本システムでは動作確認端末として以下のものを使う

- iPod touch …… iOS 8.2

## 3.6 開発環境

開発環境を以下に示す

- Python(2.7.6)
  - Caffe
  - Flask
- Xcode(6.4)

# 4 類似 iOS アプリケーション

洋服の管理アプリケーションとして以下の類似アプリケーションが存在する。

- XZ(クローゼット) 着回し発見ファッションコーディネートアプリ [6]
  自分の所持している洋服を手動でカテゴリー分けし、それらを他のユーザーと共有するアプリケーション。
- my クローゼット/着回し・コーディネートのファッション管理 [7]
  洋服の画像に手動でカテゴリーを割り当て、管理する。ユーザー自身の情報記録機能がある。
- 自撮女子 [8]
  洋服の画像をカテゴリーごとに保存でき、コーディネートを日時で保存できる。更にコーディネートを SNS を利用して共有することが出来る。

# 5　システム

本研究では ZOZOTOWN [9] で画像収集を行い、それを用いて Labellio [2] でモデルを作成する。作成されたモデルは Flask [4] サーバーを用意し、iOS アプリケーション側から洋服の画像データを受け取り、モデルを利用してサーバー側でタグ付けを行う。タグごとに画像をサーバーに保存することで、iOS アプリケーション側でカテゴリー分けされた画像 URL を一覧で返し表示させる。

## 5.1　画像収集

参考画像の収集は ZOZOTOWN [9] のページにある画像を収集する。ZOZOTOWN は服をカテゴリーごと分けるだけでなく、袖の長さを指定できるため細かい指定をして画像を収集することができるため本研究での画像収集に役立つ。画像 1 枚 1 枚を保存していくことはかなりの時間を消費するため、Google ブラウザの拡張子 Google Image Downloder を利用した。これはページ内の画像を一括保存するもので、画像サイズを限定して保存することが出来る。ブラウザの設定でファイル保存するディレクトリを指定しておくことでに服のタグ別で保存すさせることができる。今回は 1 つのタグに 2000 2500 枚の画像を用意する。

## 5.2　カテゴリーの種類

洋服のカテゴリーを以下のよう分ける。

```
.
|-- outer
|   |-- longOuter
|   |-- mediumOutr
|   |-- onepiece
|   |-- overalls
|   `-- shortOuter
|-- tops
|   |-- 7PartsSleeve
|   |-- longSleeves
|   |-- shortSleeve
|   `-- sleeveless
`-- under
    |-- longSkirt
    |-- miniSkirt
    |-- pants
    |-- shorts
    `-- skirt
```

図 3　Labellio を用いてモデルを作成する際に使用する画像フォルダ一覧

## 5.3　画像フォルダ管理

集めた画像はフォルダごとに管理し、最終的に image フォルダにまとめ、その後 zip ファイルに圧縮する。Labellio は作成するモデル 1 つにつき zip ファイルのサイズは最大 512MB、ファイル数は最大 1 万個まで可能である。image ファイル直下のフォルダ名がそのままラベル名となる。画像のフォーマットは JPEG 形式及び PNG 形式がサポートされており、画像サイズは 227*227 にシステムの方でリサイズされる。

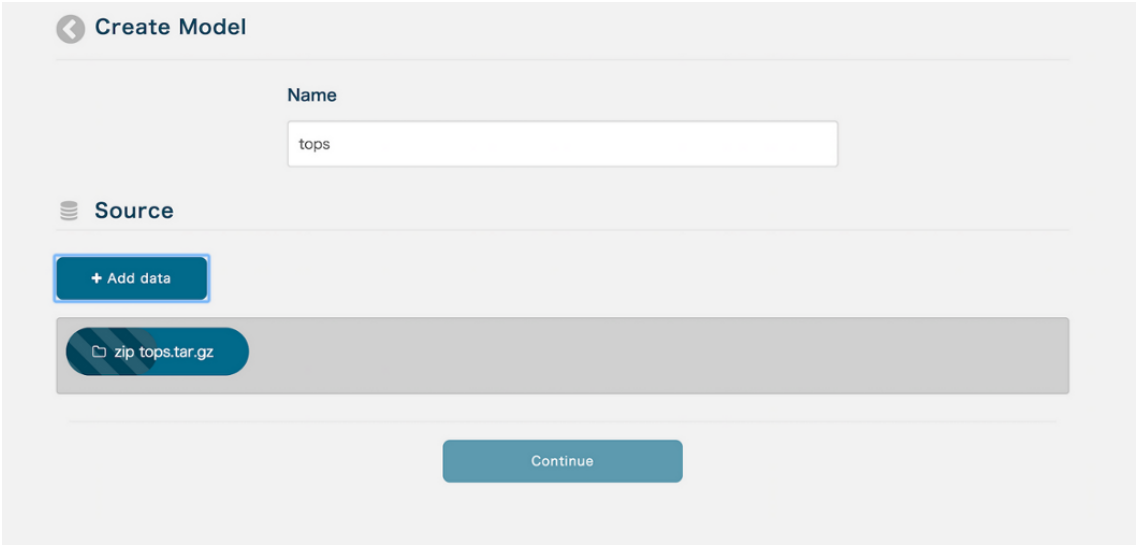


図 4　Labello でファイルをアップロード

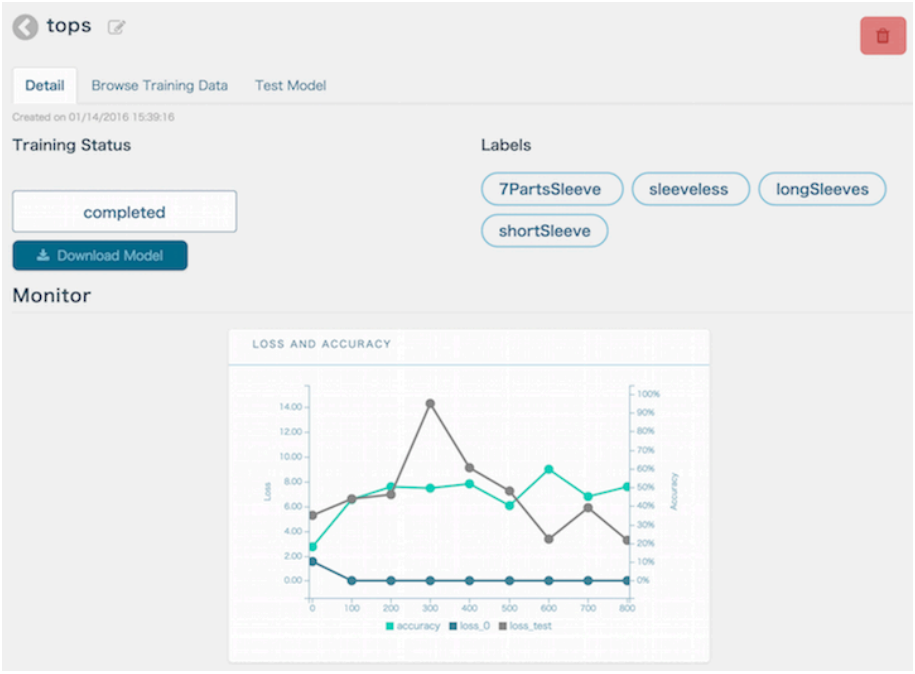


図 5　model を作成した画像

### 5.4 Flask を利用して学習モデルを動かす

作成したモデルを動かすために、トップスを topClothes.py、アンダーを underClothes.py、アウターを outerClothes.py とそれぞれファイルを用意する。ポートはそれぞれ 5000,4000,3000 とする。ファイルを 3 つに分ける理由として Labellio ではモデル 1 つに対して参照画像の数に制限があるため、すべてをまとめたモデルを作成した場合に、画像認識の性能に影響がでると考えたためである。それぞれのファイルに作成したモデルを python で読み込み、iOS 側から画像データが送られてきた場合モデルを動かし、タグ付けを行う。

### 5.5 タグ付けされた画像の保持

タグ付けされた画像は、サーバー側にそれぞれフォルダを作成し、そこに保持されるようにする。正常にタグ付けが行われ、画像の保持が出来た場合にのみタグ情報のみを iOS 側に返す。それ以外は NULL を返す。

### 5.6 サーバーに保持された画像の受け渡し

iOS 側からタグ名付きのリクエストがきた場合にそれぞれのフォルダに保持されている画像の URL を返す。フォルダに画像がなかった場合は NULL を返す。


図 6　iOS 側にタグ情報が返ってきた時の画面


図 7　iOS 側に結果が NULL で返ってきた時の画面

## 5.7　アプリ起動時の動作

　アプリを起動した後、カテゴリー分けする画像を選択するか、カテゴリー分けされた画像を表示するかを選択する。

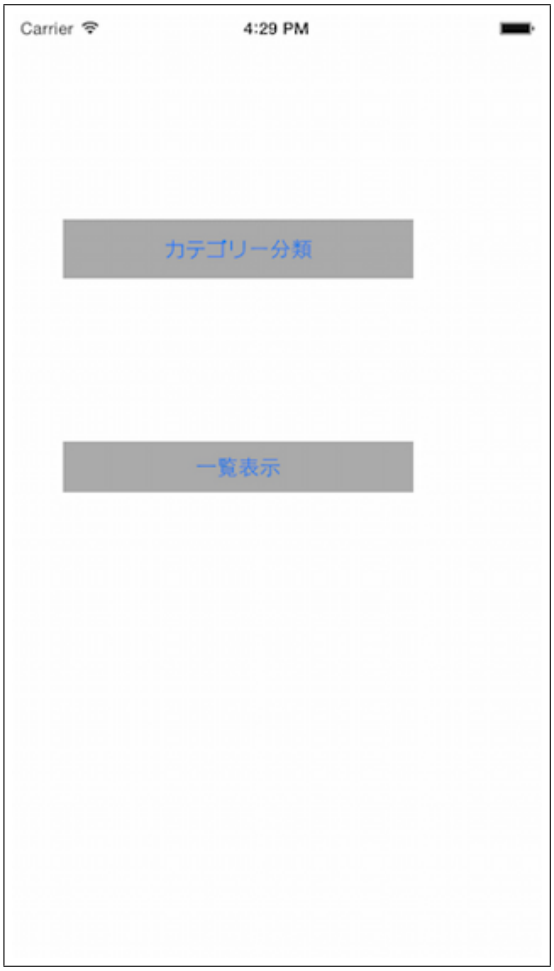


図 8　アプリ起動時の画面

## 5.8　タグ付けする画像を選択

ジャンルごとにボタンを配置し、ボタンが押されるとカメラロールを表示する。この時、選択されたボタンの名前を一時的に変数に保持させておく。選択された画像は png 形式に変換し、サーバー側になげる。この時、ボタンの名前別にアクセスするポートを変える。タグ情報が返ってきた場合アラートでどのタグにカテゴリー分けされたかを表示する。アラートを表示した後、アプリ起動時の画面へ自動遷移する。

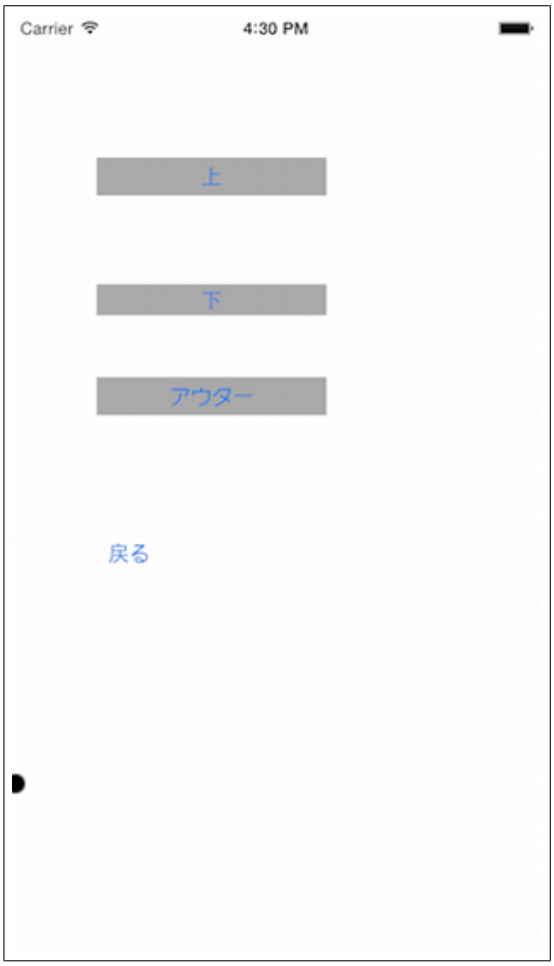


図 9　ジャンル選択画面


図 10　カメラロール画面

## 5.9　タグ付けされた画像の一覧表示

アプリ起動時にカテゴリー分けされた画像を表示するを選択した場合、タグ名を一覧でテーブルビュー表示する。その後、選択したカテゴリーとタグ名でサーバー側にリクエストを送る。画像 URL を取得出来た場合、画像を一覧で表示する。NULL で返ってきた場合はアラートを表示させる。


図 11　タグ別テーブルビュー画面

図 12　タグ別画像一覧画面

# 6 まとめ

本研究では所持している洋服を手軽に管理し、洋服のカテゴリー分けを自動化することで新しく洋服を購入する際に自身の記憶だけを頼りに今所持している洋服を確認するという不明確かつ無駄を無くすことを目的としたiOS アプリケーションの開発に取り組んだ。その結果は下記の通りである。

## 6.1 利点

本研究の利点をアプリ側、サーバー側毎に示す。

### 6.1.1 iOS アプリケーション側

アプリケーション側で画像データを渡す処理と、受け取った画像を表示する処理の 2 つにすることにより、アプリの操作を快適にする。

### 6.1.2 サーバー側

Labellio を使って画像認識をさせることにより高性能なサーバーを用意する必要がなく、さらに Flask を利用して軽量な Web サーバーが立てれるため、複雑な問題にぶつかることが少なかった。

## 6.2 問題点

### 6.2.1 画像認識の学習性能の評価

画像認識モデルを作成をする際の参考画像はカテゴリー毎に各 2000 2500 枚を収集したが、画像認識モデルの性能としてもっと良いものを作るためには参照画像の数を増やす必要がある。

### 6.2.2 カテゴリーの種類

カテゴリーを計 14 つ用意したが、カテゴリー分けを自動でされた場合にまったく別のカテゴリーに分類された場合に手動で変更が出来ない。

### 6.2.3 レスポンスの遅延

iOS アプリケーション側から画像データを投げた時のレスポンスが返ってくる時間にかなりの差が出る。

## 6.3 今後の改善点

### 6.3.1 カテゴリーの種類を個人で変更できる機能を追加

画像データをサーバー側に投げて返ってきたタグ情報を表示し、そのタグで登録していいかの選択をさせる。また、個人の判断で別のタグを付けたい場合にも対応できる状態を作る。

### 6.3.2 レスポンスの遅延改善

画像データをサーバーに送ったタイミングで画像のタグ付けと保持をすべてやるのではなく、段階を分けてサーバーで処理を行う。

# 7　謝辞

本研究を行っていく上で、大垣 斉准教授から御指導及びご協力を頂きました。また本研究のきっかけを与えてくださった卒業生の方、並びに情報教育システム研究室所属の学生に深く感謝致します。

## 参考文献


[1] そもそもディープラーニングとは何か. http://thinkit.co.jp/story/2015/08/31/6364, 8 2015.

[2] Labellio. https://www.labell.io/.

[3] 株式会社 alpacadb. http://www.alpaca.ai/.

[4] Flask. http://a2c.bitbucket.org/flask/.

[5] Caffe. http://wiki.ruka-f.net/index.php?Caffe.

[6] Xz(クローゼット) 着回し発見ファッションコーディネートアプリ. https://xz-closet.jp/#!/.

[7] my クローゼット/着回し・コーディネートのファッション管理. https://itunes.apple.com/jp/app/mykurozetto-zhe-huishi-kodinetonofasshon/id719829791?mt=8.

[8] 自撮女子. https://itunes.apple.com/jp/app/zi-cuo-nu-zi/id588931157?mt=8.

[9] Zozotown. http://zozo.jp/.

# 付録 A　付録 1

## A.1　付録 1.1

# 付録 B　ソースコード

## B.1　ソースコード.1

```
//
//  ImageViewController.h
//  Response
//
//  Created by OhnumaRina on 2016/01/24.
//  Copyright (c) 2016 年 OhnumaRina. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface ImageViewController : UIViewController

@end
```

## B.2　ソースコード.2

```
//
//  ImageViewController.m
//  Response
//
//  Created by OhnumaRina on 2016/01/24.
//  Copyright (c) 2016 年 OhnumaRina. All rights reserved.
//

#import "ImageViewController.h"

@interface ImageViewController ()

@end

//サムネイルの総数
NSInteger _thumbnailNum = 300;
//サムネイルのファイル名を入れる配列
NSMutableArray *_thumbnails;
//サムネイルのファイル名接頭詞
```

```
NSString *_thumbnailFilePreffix = @"D_";
//サムネイルのファイル名接尾詞
NSString *_thumbnailFileSuffix = @"_125";
//サムネイル間のマージン
NSInteger _thumbnailMargin = 35;
//サムネイルの外枠の寸法
NSInteger _thumbnailOutlineSize =100;
//サムネイルのボーダーを含まない寸法
NSInteger _thumbnailSize = 95;
//サムネイルの列の数
NSInteger _thumbnailColumnNum = 3;
//サムネイルの列数のカウント
NSInteger _thumbnailColumnCount = 0;
//サムネイルの行数のカウント
NSInteger _thumbnailRowCount = 0;
//サムネイルのボタンの寸法
NSInteger _buttonSize = 80;

@implementation ImageViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    //サムネイルのファイル名を配列に代入
    [self setThumbnailImageResources];
    //サムネイルリストを生成
    [self setThumbnailList];
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (void)setThumbnailImageResources
{
    //変更可能な配列を初期化
    _thumbnails = [NSMutableArray array];
    for (int i = 1; i <= _thumbnailNum; i++) {
        //拡張子より前のファイル名を保持
        [_thumbnails addObject:[NSString stringWithFormat:@"%@%@",  _thumbnailFilePreffix,[NSString str
    }
}

//サムネイルリストを生成
```

```
- (void)setThumbnailList
{
    //サムネイル群を入れるスクロールビューを初期化
    UIScrollView *scrollView = [[UIScrollView alloc] initWithFrame:CGRectMake(0, 0, 320, 568)];
    int i = 0;
    for (id thumbnailName in _thumbnails) {
        //サムネイルは 4 列で折り返す
        if (_thumbnailColumnCount == _thumbnailColumnNum) {
            _thumbnailColumnCount = 0;
            _thumbnailRowCount++;
        }
        //サムネイル表示
        NSString *imageFile = [NSString stringWithFormat:@"%@%@", thumbnailName, _thumbnailFileSuffix];
        //画像リソースの取得
        UIImage *image = [self getUIImageFromResources:imageFile ext:@"jpg"];
        UIImageView *thumbnailView = [[UIImageView alloc] initWithImage:image];
        //サムネイルの寸法指定
        CGRect rect = CGRectMake(_thumbnailMargin + (_thumbnailColumnCount * _thumbnailOutlineSize),
                                 _thumbnailMargin + (_thumbnailRowCount * _thumbnailOutlineSize), _thum
        [thumbnailView setFrame:rect];
        //サムネイルをタップ可能にする
        thumbnailView.userInteractionEnabled = YES;
        //ボタン追加
        UIButton *selectButton = [UIButton buttonWithType:UIButtonTypeCustom];
        selectButton.frame = CGRectMake(0, 0, _buttonSize, _buttonSize);
        //ボタンのタグを代入
        selectButton.tag = i;
        [selectButton addTarget:self action:@selector(selectImage:)
              forControlEvents:UIControlEventTouchUpInside];
        [thumbnailView addSubview:selectButton];
        //スクロールビューにサムネイルを追加
        [scrollView insertSubview:thumbnailView atIndex:[self.view.subviews count]];
        //サムネイルの列を一つ進める
        _thumbnailColumnCount++;
        i++;
    }
    //スクロール範囲の設定
    scrollView.contentSize = CGSizeMake(320, ((_buttonSize) * (_thumbnailRowCount - 1) - (_thumbnailMar
    //スクロールビューをステージに追加
    [self.view addSubview:scrollView];
}

//サムネイルがタップされた時のイベント
```

```
- (void)selectImage:(id)sender
{
    //予め代入しておいた tag を取得するためにセレクタを代入
    UIButton *button = (UIButton *)sender;
    //予め代入しておいた tag の数値を出力
    NSLog(@"button.tag: %ld", (long)button.tag);
}

//画像ファイルを取得
- (UIImage *)getUIImageFromResources:(NSString*)fileName ext:(NSString*)ext
{
    NSString *path = [[NSBundle mainBundle] pathForResource:fileName ofType:ext];
    UIImage *img = [[UIImage alloc] initWithContentsOfFile:path];
    return img;
}

@end
```

## B.3 ソースコード.3

```
//
//  ClothesTableViewController.h
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "ClothesTableViewCell.h"
#import "ClothesImgTableViewController.h"

#import "TableViewController.h"
#import "ImageViewController.h"

@interface ClothesTableViewController : UITableViewController

@property (strong, nonatomic) NSString *url;
@property (strong, nonatomic) NSArray *imageUrl;

@end
```

## B.4 ソースコード.4

```
//
//  ClothesTableViewController.m
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//

#import "ClothesTableViewController.h"
#import <AFNetworking/AFNetworking.h>

@interface ClothesTableViewController ()<UITableViewDelegate, UITableViewDataSource>
@property (weak, nonatomic) IBOutlet UITableView *tableView;

@property (nonatomic, strong) NSArray *dataSource;

@end

@implementation ClothesTableViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    // デリゲートメソッドをこのクラスで実装する
    self.tableView.delegate = self;
    self.tableView.dataSource = self;

    // テーブルに表示したいデータソースをセット
    self.dataSource = [NSArray arrayWithObjects:@"sleeveless",@"shortSleeve",@"longSleeves",@"7PartsSle
                       @"skirt",@"longSkirt",@"miniSkirt",@"pants",@"shorts",
                       @"outer",@"longOuter",@"shortOuter",@"onepiece",@"overalls",nil];

    // Uncomment the following line to preserve selection between presentations.
    // self.clearsSelectionOnViewWillAppear = NO;

    // Uncomment the following line to display an Edit button in the navigation bar for this view contr
    // self.navigationItem.rightBarButtonItem = self.editButtonItem;
}

-(void)backView:(UIButton *)button{
```

```
    [self dismissViewControllerAnimated:YES completion:NULL];
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
}

#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}



- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {

    NSInteger dataCount = 0;

    // テーブルに表示するデータ件数を返す
    switch (section) {
        case 0:
            dataCount = self.dataSource.count;
            break;
    }
    return dataCount;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
//    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:<#@"reuseIdentifier"#> forIn
    static NSString *CellIdentifier = @"Cell";
    // 再利用できるセルがあれば再利用する
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];

    if (!cell) {
        // 再利用できない場合は新規で作成
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
                                      reuseIdentifier:CellIdentifier];
    }

    switch (indexPath.section) {
        case 0:
```

```
            cell.textLabel.text = self.dataSource[indexPath.row];
            break;
        }

    return cell;

}


/*
// Override to support conditional editing of the table view.
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath {
    // Return NO if you do not want the specified item to be editable.
    return YES;
}
*/

/*
// Override to support editing the table view.
- (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        // Delete the row from the data source
        [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAnimationFade];
    } else if (editingStyle == UITableViewCellEditingStyleInsert) {
        // Create a new instance of the appropriate class, insert it into the array, and add a new row
    }
}
*/

/*
// Override to support rearranging the table view.
- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath *)fromIndexPath toIndexPath:
}
*/

/*
// Override to support conditional rearranging of the table view.
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:(NSIndexPath *)indexPath {
    // Return NO if you do not want the item to be re-orderable.
    return YES;
}
*/
```

```
#pragma mark - Table view delegate

// In a xib-based application, navigation from a table can be handled in -tableView:didSelectRowAtIndex
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    // Navigation logic may go here, for example:
    // Create the next view controller.
    //NSLog(@"%@", self.dataSource[indexPath.row]);
    NSString *clothesType = (NSString*)self.dataSource[indexPath.row];
    NSLog(@"%@:%ld", clothesType, (long)indexPath.row);

    if (indexPath.row == 0 || indexPath.row == 1 || indexPath.row == 2 || indexPath.row == 3){
        self.url = [NSString stringWithFormat:@"http://192.168.1.3:4000/image/tops/%@",clothesType];
    }else if (indexPath.row == 4 || indexPath.row == 5 || indexPath.row == 6 || indexPath.row == 7 || i
    {
        self.url = [NSString stringWithFormat:@"http://192.168.1.3:5000/image/under/%@",clothesType];
    }else if (indexPath.row == 9 || indexPath.row == 10 || indexPath.row == 11 || indexPath.row == 12 |
    {
        self.url = [NSString stringWithFormat:@"http://192.168.1.3:3000/image/outer/%@",clothesType];
    }else{

    }
    [self clothesTypeSave];
}

-(void)getImageUrl
{
    AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
    manager.responseSerializer = [AFXMLParserResponseSerializer serializer];
    manager.responseSerializer.acceptableContentTypes = [NSSet setWithObject:@"text/html"];
    NSString *urlU = [self.url stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncoding];
    NSString *str = [urlU stringByReplacingOccurrencesOfString:@"%13" withString:@""];

    [manager GET:str parameters:nil
         success:^(AFHTTPRequestOperation *operation, id responseObject) {
             NSLog(@"%@", responseObject);
             NSString *imagedata = operation.responseString;
             self.imageUrl =[imagedata componentsSeparatedByString:@","];
             NSLog(@"%@",self.imageUrl);
             [self clothesTypeSave];

         }
         failure:^(AFHTTPRequestOperation *operation, NSError *error){
```

```
            NSLog(@"%@", error);
        }];
}

-(void)clothesTypeSave
{
    //URL をユーザーデータに保持
    NSUserDefaults *userData = [NSUserDefaults standardUserDefaults];
    NSData *data = [NSKeyedArchiver archivedDataWithRootObject:self.imageUrl];
    [userData setObject:data forKey:@"IMAGE_URL"];
    [userData setObject:self.url forKey:@"URL"];

    //画像一覧表示画面に移動
    ImageViewController *civ = [[ImageViewController alloc] init];
    [self presentViewController:civ animated:YES completion:nil];
}

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    // Get the new view controller using [segue destinationViewController].
    // Pass the selected object to the new view controller.
}
*/

@end
```

## B.5 ソースコード.5

```
//
//  ClothesImgTableViewController.h
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "ClothesImgTableViewCell.h"

@interface ClothesImgTableViewController : UITableViewController
```

```
@property (nonatomic, strong) NSString *url;
@property (nonatomic, strong) NSString *imagedata;
@property (nonatomic, strong) NSMutableArray *imageUrlArray;

@end
```

## B.6 ソースコード.6

```
//
//  ClothesImgTableViewController.m
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//

#import "ClothesImgTableViewController.h"
#import <AFNetworking/AFNetworking.h>

@interface ClothesImgTableViewController ()<UITableViewDelegate, UITableViewDataSource>
@property (weak, nonatomic) IBOutlet UITableView *tableViewcontroller;
@end

@implementation ClothesImgTableViewController

- (void)viewDidLoad {
    [super viewDidLoad];
   // self.tableViewcontroller.delegate = self;
   // self.tableViewcontroller.dataSource = self;

//     [self.tableViewcontroller registerNib:[UINib nibWithNibName:@"ClothesImgTableViewCell" bundle:ni


    NSUserDefaults *userData = [NSUserDefaults standardUserDefaults];
    NSData *data = [userData objectForKey:@"IMAGE_URL"];
    self.imageUrlArray = [NSKeyedUnarchiver unarchiveObjectWithData:data];
    NSLog(@"%@",self.imageUrlArray);

    //[self getImageURL];
}

//-(void)getImageURL
```

```
//{
//
//    NSString *urlU = [self.url stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncoding];
//    NSString *str = [urlU stringByReplacingOccurrencesOfString:@"%13" withString:@""];
//
//    AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
//    manager.responseSerializer = [AFXMLParserResponseSerializer serializer];
//    manager.responseSerializer.acceptableContentTypes = [NSSet setWithObject:@"text/html"];
//
//    [manager GET:str parameters:nil
//         success:^(AFHTTPRequestOperation *operation, id responseObject) {
//             self.imagedata = operation.responseString;
//             self.imageUrlArray =[self.imagedata componentsSeparatedByString:@","];             NSLog
//         }
//         failure:^(AFHTTPRequestOperation *operation, NSError *error){
//             NSLog(@"%@", error);
//         }];
//}


- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    //Return the number of rows in the section.
    NSInteger dataCount = 0;
    // テーブルに表示するデータ件数を返す
    switch (section) {
        case 0:
            dataCount = self.imageUrlArray.count;
            break;
    }
    NSLog(@"%ld",dataCount);
    return dataCount;
}
```

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
//    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell" forIndexPath:indexPa
//    static NSString *CellIdentifier = @"Cell";
//    // 再利用できるセルがあれば再利用する
//    //UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
//
//    if (!cell) {
//        // 再利用できない場合は新規で作成
//        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
//                                      reuseIdentifier:CellIdentifier];
//    }
//
//    switch (indexPath.section) {
//        case 0:
//            cell.textLabel.text = self.imageUrlArray[indexPath.row];
//            NSLog(@"%@",self.imageUrlArray[indexPath.row]);
//            break;
//    }
//    return cell;
//    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];

    ClothesImgTableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];

    // Configure the cell...
//    if(cell == nil){
//        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:Cel
//    }

    //NSDictionary *status = [self.imageUrlArray objectAtIndex:indexPath.row];
    NSString *text = [self.imageUrlArray objectAtIndex:indexPath.row];
    NSLog(@"%@",[self.imageUrlArray objectAtIndex:indexPath.row]);
    //cell.textLabel.text = text;

    dispatch_queue_t q_global = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
    dispatch_queue_t q_main = dispatch_get_main_queue();
    cell.cloatheimage.image = nil;
    dispatch_async(q_global, ^{
        NSString *imageURL = text;
        UIImage *image = [UIImage imageWithData:[NSData dataWithContentsOfURL: [NSURL URLWithString: im

        dispatch_async(q_main, ^{
```

```
            cell.cloatheimage.image = image;
            [cell layoutSubviews];
        });
    });


    return cell;
}


/*
// Override to support conditional editing of the table view.
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath {
    // Return NO if you do not want the specified item to be editable.
    return YES;
}
*/

/*
// Override to support editing the table view.
- (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        // Delete the row from the data source
        [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAnimationFade];
    } else if (editingStyle == UITableViewCellEditingStyleInsert) {
        // Create a new instance of the appropriate class, insert it into the array, and add a new row
    }
}
*/

/*
// Override to support rearranging the table view.
- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath *)fromIndexPath toIndexPath:
}
*/

/*
// Override to support conditional rearranging of the table view.
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:(NSIndexPath *)indexPath {
    // Return NO if you do not want the item to be re-orderable.
    return YES;
}
*/
```

```
/*
#pragma mark - Table view delegate

// In a xib-based application, navigation from a table can be handled in -tableView:didSelectRowAtIndex
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    // Navigation logic may go here, for example:
    // Create the next view controller.
    <#DetailViewController#> *detailViewController = [[<#DetailViewController#> alloc] initWithNibName:

    // Pass the selected object to the new view controller.

    // Push the view controller.
    [self.navigationController pushViewController:detailViewController animated:YES];
}
*/

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    // Get the new view controller using [segue destinationViewController].
    // Pass the selected object to the new view controller.
}
*/

@end
```

## B.7 **ソースコード**.7

```
//
//  AppDelegate.m
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//

#import "AppDelegate.h"

@interface AppDelegate ()
```

```
@end

@implementation AppDelegate


- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOp
    // Override point for customization after application launch.
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application {
    // Sent when the application is about to move from active to inactive state. This can occur for cer
    // Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates.
}

- (void)applicationDidEnterBackground:(UIApplication *)application {
    // Use this method to release shared resources, save user data, invalidate timers, and store enough
    // If your application supports background execution, this method is called instead of applicationW
}

- (void)applicationWillEnterForeground:(UIApplication *)application {
    // Called as part of the transition from the background to the inactive state; here you can undo ma
}

- (void)applicationDidBecomeActive:(UIApplication *)application {
    // Restart any tasks that were paused (or not yet started) while the application was inactive. If t
}

- (void)applicationWillTerminate:(UIApplication *)application {
    // Called when the application is about to terminate. Save data if appropriate. See also applicatio
}

@end
```

## B.8 ソースコード.8

```
//
//  AppDelegate.h
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//
```

```
#import <UIKit/UIKit.h>

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;


@end
```

## B.9 ソースコード.9

```
//
//  ViewController.h
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "SelectViewController.h"
#import "ClothesTableViewController.h"

@interface ViewController : UIViewController

@property (nonatomic, strong) IBOutlet UIButton *categoryBtn;
@property (nonatomic, strong) IBOutlet UIButton *clothesListBtn;

@end
```

## B.10 ソースコード.10

```
//
//  ViewController.m
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//
```

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    [self.categoryBtn addTarget:self
                         action:@selector(tapCategoryBtn:)
               forControlEvents:UIControlEventTouchUpInside];

    [self.clothesListBtn addTarget:self
                         action:@selector(tapClothesListBtn:)
               forControlEvents:UIControlEventTouchUpInside];



    // Do any additional setup after loading the view, typically from a nib.
}

-(void)tapCategoryBtn:(UIButton *)button{
    SelectViewController *sv = [[SelectViewController alloc]init];
    [self presentViewController:sv animated:YES completion:nil];
}

-(void)tapClothesListBtn:(UIButton *)button{
    ClothesTableViewController *cv = [[ClothesTableViewController alloc]init];
    [self presentViewController:cv animated:YES completion:nil];
}


- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

## B.11 ソースコード.11

```
//
//  SelectViewController.h
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "ViewController.h"

@interface SelectViewController : UIViewController
<UINavigationControllerDelegate,
UIImagePickerControllerDelegate>

@property (weak, nonatomic) IBOutlet UIButton *topsSelectBtn;
@property (weak, nonatomic) IBOutlet UIButton *underSelectBtn;
@property (weak, nonatomic) IBOutlet UIButton *outerSelectBtn;
@property (weak, nonatomic) IBOutlet UIButton *backBtn;

@property (strong, nonatomic) NSString *url;
@property (strong, nonatomic) UIImage *selectImage;

@end
```

## B.12 ソースコード.12

```
//
//  SelectViewController.m
//  Response
//
//  Created by OhnumaRina on 2015/10/23.
//  Copyright (c) 2015年 OhnumaRina. All rights reserved.
//

#import "SelectViewController.h"
#import <AFNetworking/AFNetworking.h>

@interface SelectViewController ()
```

```
@end

@implementation SelectViewController

- (void)viewDidLoad {

    [super viewDidLoad];

    [self.backBtn addTarget:self
                     action:@selector(tapBackBtn:)
           forControlEvents:UIControlEventTouchUpInside];

    [self.topsSelectBtn addTarget:self
                           action:@selector(tapTopsBtn:)
                 forControlEvents:UIControlEventTouchUpInside];
    [self.underSelectBtn addTarget:self
                            action:@selector(tapUnderBtn:)
                  forControlEvents:UIControlEventTouchUpInside];
    [self.outerSelectBtn addTarget:self
                            action:@selector(tapOuterBtn:)
                  forControlEvents:UIControlEventTouchUpInside];
    // Do any additional setup after loading the view from its nib.
}

-(void)tapBackBtn:(UIButton *)button{
    [self dismissViewControllerAnimated:YES completion:NULL];

}
-(void)tapTopsBtn:(UIButton *)button{

    //self.url = [NSURL URLWithString:@"http://192.168.1.3:4000/classify"];
    self.url = @"http://192.168.1.3:4000/classify";


    UIImagePickerController *imgPic = [[UIImagePickerController alloc]init];
    imgPic.delegate = self;
    [imgPic setSourceType:UIImagePickerControllerSourceTypePhotoLibrary];
    [self presentViewController: imgPic animated:YES completion:nil];

}

-(void)tapUnderBtn:(UIButton *)button{
```

```
    //self.url = [NSURL URLWithString:@"http://192.168.1.3:5000/classify"];
    self.url = @"http://192.168.1.3:5000/classify";
    UIImagePickerController *imgPic = [[UIImagePickerController alloc]init];
    imgPic.delegate = self;
    [imgPic setSourceType:UIImagePickerControllerSourceTypePhotoLibrary];
    [self presentViewController: imgPic animated:YES completion:nil];
}

-(void)tapOuterBtn:(UIButton *)button{
    self.url = @"http://192.168.1.3:3000/classify";
    UIImagePickerController *imgPic = [[UIImagePickerController alloc]init];
    imgPic.delegate = self;
    [imgPic setSourceType:UIImagePickerControllerSourceTypePhotoLibrary];
    [self presentViewController: imgPic animated:YES completion:nil];

}



- (void)imagePickerController :(UIImagePickerController *)picker
        didFinishPickingImage :(UIImage *)image editingInfo :(NSDictionary *)editingInfo {

    //画像が選択されたとき。オリジナル画像を UIImageView に突っ込む
    self.selectImage = image;

    // 読み込んだ画像表示
    NSLog(@"selected");



    //UIImageView *iv = [[UIImageView alloc] initWithImage:image];
    //[self.view addSubview:iv];

    [self dismissViewControllerAnimated:YES completion:nil];
    //[self imageSelect];
    [self uploadButtonTouched];
}

-(void)imageSelect
{
    NSString* boundary = @"MyBoundaryString";
    NSURLSessionConfiguration* config = [NSURLSessionConfiguration defaultSessionConfiguration];
    config.HTTPAdditionalHeaders =
```

```
    @{
      @"Content-Type" : [NSString stringWithFormat:@"multipart/form-data; boundary=%@", boundary]
      };
    NSMutableURLRequest* request = [NSMutableURLRequest requestWithURL:self.url];
    NSURLSession* session = [NSURLSession sessionWithConfiguration:config];

    // アップロードする画像
    NSData* jpgData = [[NSData alloc] initWithData:UIImageJPEGRepresentation(self.selectImage, 0.5f)];
    NSString* jpg64Str = [jpgData base64EncodedStringWithOptions:NSDataBase64Encoding76CharacterLineLen

//    NSString* path = [[NSBundle mainBundle] pathForResource:self.selectImage ofType:@"png"];
//    NSData* imageData = [NSData dataWithContentsOfFile:path];
     NSData* imageData = [NSData dataWithContentsOfFile:jpg64Str];
    NSLog(@"%@",imageData);
    // post データの作成
    NSMutableData* data = [NSMutableData data];

    // 画像の設定
    [data appendData:[[NSString stringWithFormat:@"--%@\r\n", boundary] dataUsingEncoding:NSUTF8StringE
    [data appendData:[[NSString stringWithFormat:@"Content-Disposition: form-data;"] dataUsingEncoding:
    [data appendData:[[NSString stringWithFormat:@"name=\"%@\";", @"image"] dataUsingEncoding:NSUTF8Str
    [data appendData:[[NSString stringWithFormat:@"filename=\"%@\"\r\n", @"vlcsnap-2015-05-31-00h33m14s
    [data appendData:[[NSString stringWithFormat:@"Content-Type: image/jpeg\r\n\r\n"] dataUsingEncoding
    [data appendData:jpgData];
    [data appendData:[[NSString stringWithFormat:@"\r\n"] dataUsingEncoding:NSUTF8StringEncoding]];

    // 最後にバウンダリを付ける
    [data appendData:[[NSString stringWithFormat:@"--%@--\r\n", boundary] dataUsingEncoding:NSUTF8Strin

    request.HTTPMethod = @"POST";
    request.HTTPBody = data;
    NSURLSessionDataTask* task = [session dataTaskWithRequest:request
                                            completionHandler:^(NSData *data, NSURLResponse *response,
                                                // 完了時の処理
                                                NSArray *array = [NSJSONSerialization JSONObjectWithDat
                                                NSLog(@"%@",array);
                                                NSLog(@"%@,%@", [array valueForKeyPath:@"label"], [arra
                                                if([array valueForKeyPath:@"label"] == nil) {
                                                    //[self alertView];
                                                }
                                            }];
    [task resume];
```

```
}

-(void)uploadButtonTouched
{
    NSData *imageData = [[NSData alloc] initWithData:UIImageJPEGRepresentation(self.selectImage, 0.1)];
   // NSLog(@"%@", imageData);

    AFHTTPRequestOperationManager *manager = [[AFHTTPRequestOperationManager alloc] initWithBaseURL:[NS
    NSDictionary *parameters = nil;

    manager.requestSerializer= [AFHTTPRequestSerializer serializer];

    AFHTTPRequestOperation *op = [manager POST:self.url parameters:parameters constructingBodyWithBlock
        //do not put image inside parameters dictionary as I did, but append it!
        [formData appendPartWithFileData:imageData name:@"image" fileName:@"image.jpg" mimeType:@"image
    } success:^(AFHTTPRequestOperation *operation, id responseObject) {
        NSLog(@"Success: %@", responseObject);
        NSLog(@"%@,%@", [responseObject valueForKeyPath:@"label"],[responseObject valueForKeyPath:@"lab
        [self sucessView];
    } failure:^(AFHTTPRequestOperation *operation, NSError *error) {
        NSLog(@"Error: %@ ***** %@", operation.responseString, error);
        [self alertView];
    }];

    [op start];


}


-(void)sucessView{
    UIAlertView *alert = [[UIAlertView alloc]
                          initWithTitle:@"完了"
                          message:@"カテゴリー分けしました。"
                          delegate:self
                          cancelButtonTitle:nil
                          otherButtonTitles:@"Ok", nil];

    // アラートビューを表示
    [alert show];
}
```

```
-(void)alertView
{
    UIAlertView *alert = [[UIAlertView alloc]
                          initWithTitle:@"失敗"
                          message:@"カテゴリー分けに失敗しました。"
                          delegate:self
                          cancelButtonTitle:nil
                          otherButtonTitles:@"Ok", nil];

    // アラートビューを表示
    [alert show];
}



- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}



@end
```

## B.13 ソースコード.13

```
#!/usr/bin/env python2
import json
import os
import tempfile
import commands
from werkzeug.utils import secure_filename
import datetime
import shutil

# pip install flask
from flask import Flask, request, make_response, abort, redirect, url_for, send_from_directory
from labellio import Classifier, Config, ImageLoader, Label

# Flask Settings
DEBUG = True
UPLOAD_FOLDER = "/tmp"
MODEL_FOLDER = "tops_model"
SUPPORT_EXTENSIONS = set([".jpg", ".jpeg", ".png"])
app = Flask(__name__)
```

```
app.config.from_object(__name__)

# Labellio Settings
labellio_config = Config(app.config['MODEL_FOLDER'])
labellio_label = Label(labellio_config)
labellio_image_loader = ImageLoader(labellio_config)
labellio_classifier = Classifier(labellio_config)
with open(labellio_config.label_file) as fp:
    labellio_label_dict = json.load(fp)


def images(image_dir):
    for base, _, files in os.walk(image_dir):
        for f in files:
            yield os.path.join(base, f)


def exec_batch(batch, classifier, label):
    paths, data = zip(*batch)
    result = {"label_name": labellio_label_dict}
    for i, output in enumerate(classifier.forward_iter(data)):
        result[i] = {
            "label": label.label(output.best), "score": output.values.tolist()}
#    return image_upload(labellio_label_dict)
    return result


def labellio_exec(image_dir):
    global labellio_config, labellio_label, labellio_image_loader, labellio_classifier
    batch = []
    for image in images(image_dir):
        batch.append((image, labellio_image_loader.load(image)))
    return exec_batch(batch, labellio_classifier, labellio_label)

# URL


@app.route('/', methods=['GET'])
def help():
    return """POST your image as "image" parameter to /classify.
(ex. curl -F "image=@test.png" http://{0}/classify)
""".format(request.host)
```

```
@app.route('/classify', methods=['POST'])
def classify():
    uploaded_file = request.files['image']
    root, ext = os.path.splitext(uploaded_file.filename)
    temp_dir = ""

    if ext in app.config['SUPPORT_EXTENSIONS']:
        temp_dir = tempfile.mkdtemp(dir=app.config['UPLOAD_FOLDER'])
        uploaded_file.save(os.path.join(temp_dir, uploaded_file.filename))
        src = os.path.join(temp_dir, uploaded_file.filename)
    else:
        abort(400)

    result = labellio_exec(temp_dir)
    label_name = result[0]["label"]
    dist = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'static/image/tops', label_name,str
    #src = os.path.join(temp_dir, str(datetime.datetime.now()) + uploaded_file.filename)
 #os.rename(src, os.path.join(temp_dir, str(datetime.datetime.now()),uploaded_file.filename))
   # uploaded_file.name = str(datetime.datetime.now())+secure_filename(uploaded_file.filename)
   # uploaded_file.save(os.path.join(dist, label_name, uploaded_file.name))
    print src
    print dist

    shutil.move(src, dist)

    # return redirect(url_for(view_upload(filepath=filepath,filename=uploaded_file.name,label_name=labe
   # return redirect(url_for(filepath=filepath, filename=uploaded_file.name, label_name))

    response = make_response()
    response.data = json.dumps(result)
    response.headers["Content-Type"] = "application/json"
    return response

def view_upload(filepath,filename,label_name):
     return send_from_directory(os.path.join(filepath,filename),filename)

#image

@app.route('/image', methods=['POST'])
#def image_upload():
 #   file = request.files['imgUpload'];
  #  file.name = secure_filename(file.filename)
```

```
   # file.save(os.path.join('/image/tops',result))
    #return redirect(url_for("view_upload,filename=file.name))

@app.route('/image', methods=['GET'])
def image_post():
    return """/image"""



@app.route('/static/image/tops/sleeveless', methods=['GET'])
def sleeveless():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/tops/sleeveless'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))

@app.route('/static/image/tops/shortSleeve', methods=['GET'])
def short():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/tops/shortSleeve'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))

@app.route('/static/image/tops/longSleeves', methods=['GET'])
def long():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/tops/longSleeves'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))

@app.route('/static/image/tops/7PartsSleeve', methods=['GET'])
def image_7part():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/tops/7PartsSleeve'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))

if __name__ == '__main__':
    app.run(host='192.168.1.3', port=4000)
```

## B.14 **ソースコード**.14

```
#!/usr/bin/env python2
import json
import os
import tempfile
import commands
from werkzeug.utils import secure_filename
import datetime
import shutil
```

```
# pip install flask
from flask import Flask, request, make_response, abort, redirect, url_for, send_from_directory
from labellio import Classifier, Config, ImageLoader, Label

# Flask Settings
DEBUG = True
UPLOAD_FOLDER = "tmp"
MODEL_FOLDER = "under_model"
SUPPORT_EXTENSIONS = set([".jpg", ".jpeg", ".png"])
app = Flask(__name__)
app.config.from_object(__name__)

# Labellio Settings
labellio_config = Config(app.config['MODEL_FOLDER'])
labellio_label = Label(labellio_config)
labellio_image_loader = ImageLoader(labellio_config)
labellio_classifier = Classifier(labellio_config)
with open(labellio_config.label_file) as fp:
    labellio_label_dict = json.load(fp)


def images(image_dir):
    for base, _, files in os.walk(image_dir):
        for f in files:
            yield os.path.join(base, f)


def exec_batch(batch, classifier, label):
    paths, data = zip(*batch)
    result = {"label_name": labellio_label_dict}
    for i, output in enumerate(classifier.forward_iter(data)):
        result[i] = {
            "label": label.label(output.best), "score": output.values.tolist()}
    return result


def labellio_exec(image_dir):
    global labellio_config, labellio_label, labellio_image_loader, labellio_classifier
    batch = []
    for image in images(image_dir):
        batch.append((image, labellio_image_loader.load(image)))
    return exec_batch(batch, labellio_classifier, labellio_label)
```

```
# URL


@app.route('/', methods=['GET'])
def help():
    return """POST your image as "image" parameter to /classify.
(ex. curl -F "image=@test.png" http://{0}/classify)
""".format(request.host)


@app.route('/classify', methods=['POST'])
def classify():
    uploaded_file = request.files['image']
    root, ext = os.path.splitext(uploaded_file.filename)
    temp_dir = ""
    if ext in app.config['SUPPORT_EXTENSIONS']:
        temp_dir = tempfile.mkdtemp(dir=app.config['UPLOAD_FOLDER'])
        uploaded_file.save(os.path.join(temp_dir, uploaded_file.filename))
        src = os.path.join(temp_dir, uploaded_file.filename)
    else:
        abort(400)

    result = labellio_exec(temp_dir)
    label_name = result[0]["label"]
    dist = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'static/image/under', label_name,st

    print src
    print dist

    shutil.move(src, dist)

    response = make_response()
    response.data = json.dumps(result)
    response.headers["Content-Type"] = "application/json"
    return response


@app.route('/static/image/under/longSkirt', methods=['GET'])
def longSkirt():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/under/longSkirt'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))
```

```
@app.route('/static/image/under/miniSkirt', methods=['GET'])
def miniSkirt():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/under/miniSkirt'
    return '', ''.join(commands.getoutput("ls %s" % (path)).split('\n'))

@app.route('/static/image/under/pants', methods=['GET'])
def pants():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/under/pants'
    return '', ''.join(commands.getoutput("ls %s" % (path)).split('\n'))

@app.route('/static/image/under/shorts', methods=['GET'])
def shorts():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/under/shorts'
    return '', ''.join(commands.getoutput("ls %s" % (path)).split('\n'))

@app.route('/static/image/under/skirt', methods=['GET'])
def skirt():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/under/skirt'
    return '', ''.join(commands.getoutput("ls %s" % (path)).split('\n'))

if __name__ == '__main__':
    app.run(host='192.168.1.3', port=5000)
```

## B.15 ソースコード.15

```
#!/usr/bin/env python2
import json
import os
import tempfile
import commands
from werkzeug.utils import secure_filename
import datetime
import shutil

# pip install flask
from flask import Flask, request, make_response, abort, redirect, url_for, send_from_directory
from labellio import Classifier, Config, ImageLoader, Label

# Flask Settings
DEBUG = True
UPLOAD_FOLDER = "/tmp"
MODEL_FOLDER = "outer_model"
SUPPORT_EXTENSIONS = set([".jpg", ".jpeg", ".png"])
```

```
app = Flask(__name__)
app.config.from_object(__name__)

# Labellio Settings
labellio_config = Config(app.config['MODEL_FOLDER'])
labellio_label = Label(labellio_config)
labellio_image_loader = ImageLoader(labellio_config)
labellio_classifier = Classifier(labellio_config)
with open(labellio_config.label_file) as fp:
    labellio_label_dict = json.load(fp)


def images(image_dir):
    for base, _, files in os.walk(image_dir):
        for f in files:
            yield os.path.join(base, f)


def exec_batch(batch, classifier, label):
    paths, data = zip(*batch)
    result = {"label_name": labellio_label_dict}
    for i, output in enumerate(classifier.forward_iter(data)):
        result[i] = {
            "label": label.label(output.best), "score": output.values.tolist()}
#    return image_upload(labellio_label_dict)
    return result


def labellio_exec(image_dir):
    global labellio_config, labellio_label, labellio_image_loader, labellio_classifier
    batch = []
    for image in images(image_dir):
        batch.append((image, labellio_image_loader.load(image)))
    return exec_batch(batch, labellio_classifier, labellio_label)

# URL


@app.route('/', methods=['GET'])
def help():
    return """POST your image as "image" parameter to /classify.
(ex. curl -F "image=@test.png" http://{0}/classify)
""".format(request.host)
```

```
@app.route('/classify', methods=['POST'])
def classify():
    uploaded_file = request.files['image']
    root, ext = os.path.splitext(uploaded_file.filename)
    temp_dir = ""

    if ext in app.config['SUPPORT_EXTENSIONS']:
        temp_dir = tempfile.mkdtemp(dir=app.config['UPLOAD_FOLDER'])
        uploaded_file.save(os.path.join(temp_dir, uploaded_file.filename))
        src = os.path.join(temp_dir, uploaded_file.filename)
    else:
        abort(400)

    result = labellio_exec(temp_dir)
    label_name = result[0]["label"]
    dist = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'static/image/outer', label_name,st

    print src
    print dist

    shutil.move(src, dist)

    response = make_response()
    response.data = json.dumps(result)
    response.headers["Content-Type"] = "application/json"
    return response

def view_upload(filepath,filename,label_name):
     return send_from_directory(os.path.join(filepath,filename),filename)

#image

@app.route('/static/image/outer/longOuter', methods=['GET'])
def longOuter():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/outer/longOuter'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))

@app.route('/static/image/outer/mediumOuter', methods=['GET'])
def mediumOuter():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/outer/mediumOuter'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))
```

```
@app.route('/static/image/outer/onepiece', methods=['GET'])
def onepiece():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/outer/onepiece'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))

@app.route('/static/image/outer/overalls', methods=['GET'])
def overalls():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/outer/overalls'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))

@app.route('/static/image/outer/shortOuter', methods=['GET'])
def shortOuter():
    path = os.path.dirname(os.path.abspath(__file__)) + '/static/image/outer/shortOuter'
    return ", ".join(commands.getoutput("ls %s" % (path)).split('\n'))

if __name__ == '__main__':
    app.run(host='192.168.1.3', port=3000)
```