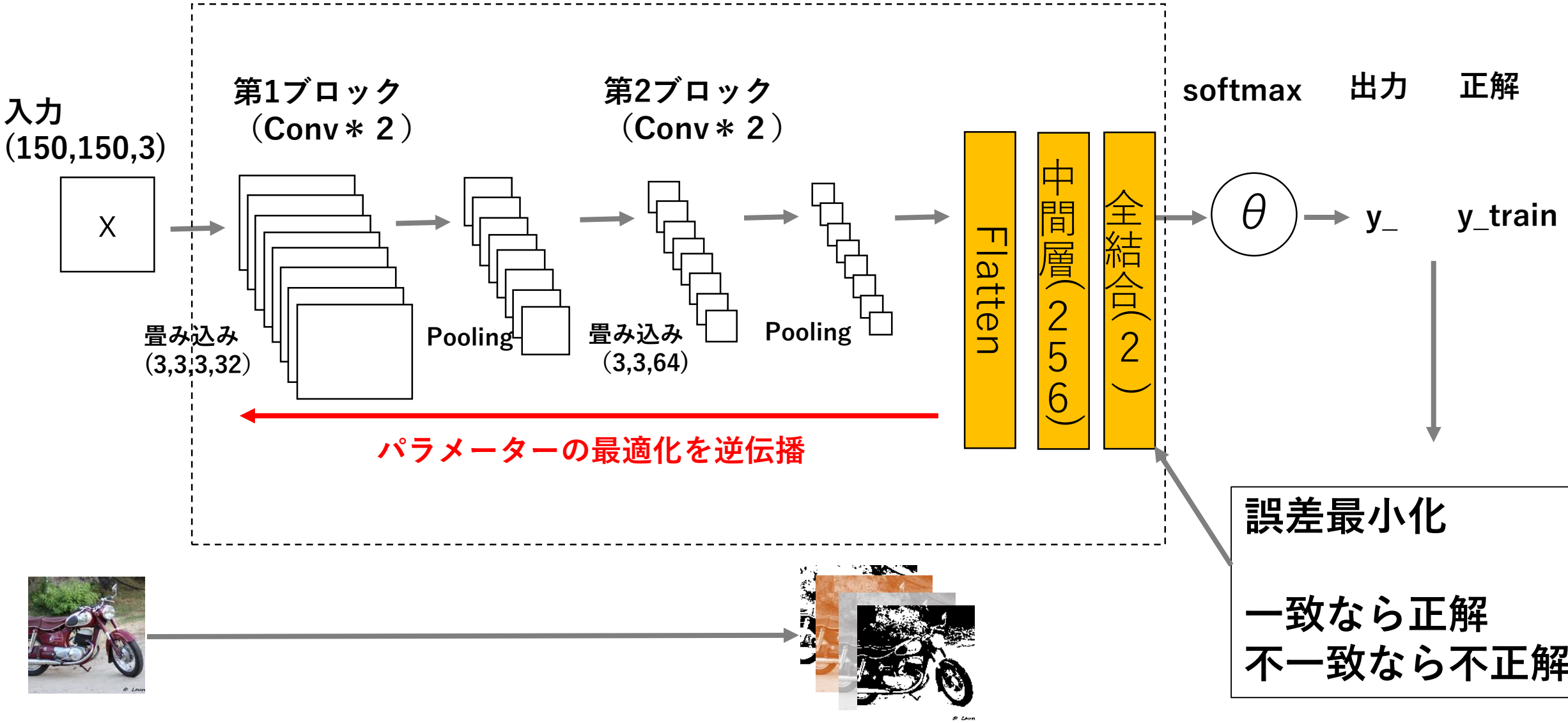
DjangoでAIアプリ開発

転移学習・ファインチューニングで高精度アプリを作ろう

畳み込みニューラルネットワーク



ファインチューニング

この部分を学習済みモデルで代用

入力
(150,150,3)

X

第1ブロック
（Conv＊２）

畳み込み
(3,3,3,32)

Pooling

第2ブロック
（Conv＊２）

畳み込み
(3,3,64)

Pooling

Flatten

中間層(２５６)

全結合(２)

softmax

θ

出力

y_

正解

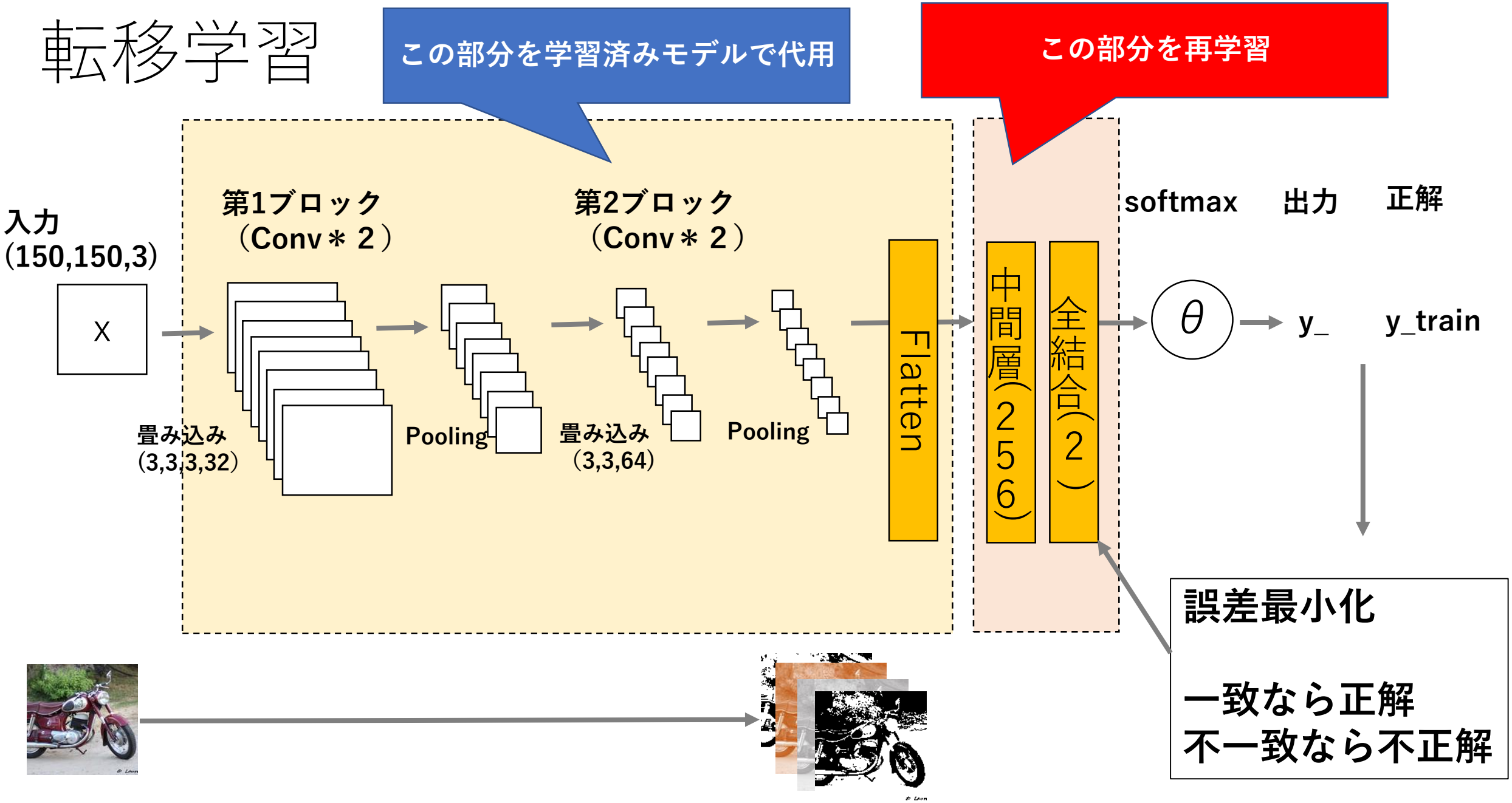
y_train

誤差最小化

ら正解
なら不正解

畳み込みフィルターも再学習

転移学習


この部分を学習済みモデルで代用
この部分を再学習
入力
(150,150,3)
X
第1ブロック
（Conv＊２）
第2ブロック
（Conv＊２）
畳み込み
(3,3,3,32)
Pooling
畳み込み
(3,3,64)
Pooling
Flatten
中間層(256)
全結合(2)
softmax
θ
出力
y_
正解
y_train
誤差最小化
一致なら正解
不一致なら不正解

転移学習のセクションの流れ

1. データ準備
    1. NumPy形式で保存

2. トレーニング
    1. データをロードする
    2. モデルを定義する
    3. 最適化手法を定義する
    4. トレーニング（fit）を実行する
    5. 精度評価を行う

3. モデルを保存する（h5形式）

4. モデルをロードして推定を行う

モデルの詳細：

https://arxiv.org/pdf/1409.1556.pdf

出典

- ConvNet（畳み込みニューラルネットワーク）
- 入力は224x224x3

Djangoアプリの構成

/carbike

CarBikeメニュー

ファイルの選択

ファイルアップロード

ファイルのパース

NumPy配列に変換

predictを実行

/predict

推定結果を表示

分類ラベル

推定精度

Bootstrapで見た目を整えたい

アップロード画像を表示したい

Djangoのテンプレート機能

- テンプレートのロード
  - loader.get_template(“アプリ名/xxx.html”)
  - templates/アプリ名　以下にテンプレートファイルを作成しておく
  - Bootstrap4モジュールを使うと便利
  - 共通部分はbase.htmlにまとめておく
  - DTL（Django Template Language）

- context
  - 変数と値の関連付けを行う
  - テンプレートに変数と値のセットを渡して参照する

テンプレートファイル

base.html

```
<!DOCTYPE html>
<html lang=“ja”>
<head>
    <meta charset="UTF-8">
    {% load static %}
    {% load bootstrap4 %}
    {% bootstrap_css %}
    <link rel="stylesheet" type="text/css" href="{%
‘carbike/css/style.css' %}">
    {% bootstrap_javascript jquery='full’ %}
    <title>サイト名｜{% block title %}{% endblock %}</title>
</head>
<body>
    <nav  ナビゲーションバー>
    </nav>
    <div class=“container”>
        {% block content %}{% endblock %}
    </div>
</html>
```

上書き

上書き

index.html

```
{% extends ‘carbike/base.html' %}

{% block title %}ページタイトル{% endblock %}

{% block content %}
<div>
    <form>
        # 投稿フォーム(ファイルアップロード)
    <button type="submit" class="btn btn-
primary">判定</button>
  </form>
</div>
{% endblock %}
```

DjangoでFormを使おう

- HTMLにフォームを埋め込む方法
  1. テンプレート内に直接HTMLタグで記述する
  2. Djangoのフォーム機能を使う（form.py）
     1. クラスを作成しておいてインポート
     2. モデルとの連携がしやすい

- carbike/templates/carbike/ 以下にテンプレート
  - base.html
  - index.html　←　ここに追加
  - predict.html

```
1.  {% extends ‘carbike/base.html' %}
2.  {% block title %}車・バイク推定アプリ{% endblock %}
3.  {% block content %}
4.  <div>
5.    <h4 class=“mt-4 mb-5 border-bottom”>車・バイク推定ア
6.    <p>画像ファイルを選択して推定ボタンを押してください。
7.    <form action="{% url 'carbike:predict' %}" method="post" class="form" enctype="multipart/form-data">
8.      {% csrf_token %} // クロスサイトリクエストフォージェリ
9.      <div class="form-group">
10.       <div class="custom-file">
11.         {{ form.image }}
12.         <label class="custom-file-label" for="customFile">ファイルを選択してください</label>
13.       </div>
14.     </div>
15.     <button type=“submit” class=“btn btn-primary”>推定する！</button>
16.   </form>
17. </div>
18. {% endblock %}
```

アクション（遷移先）指定

CSRF対策

定義済みフォームの読み込み

Formタグ

- <form action="{% url 'carbike:predict' %}" method="post" class="form" enctype="multipart/form-data">
  - carbikeのpredict関数にデータを送る
- POSTとGET（できればPOSTで渡したい）
  - POST: セッション内にデータを入れる（外部から見えない）
  - GET：　URLに変数を含めて渡す（外部から見えるため要注意）

{{ form.image }}

forms.py

```
1. from django import forms
2.
3. class PhotoForm(forms.Form):
4.     image = forms.ImageField(widget=forms.FileInput(attrs={'class': 'custom-file-input'}))
```

views.py

```
1. from django.shortcuts import redirect
2. from django.http import HttpResponse
3. from django.template import loader
4. from .forms import PhotoForm    # forms.pyをインポート
5. from .models import Photo

6. def index(request):
7.     template = loader.get_template('carbike/index.html')
8.     context = { 'form': PhotoForm() }

9.     return HttpResponse(template.render(context, request))
```

```
def predict(request):
    if not request.method == 'POST':
        return redirect('carbike:index')
    form = PhotoForm(request.POST, request.FILES)
    if not form.is_valid():
        raise ValueError('invalid form')
    photo = Photo(image=form.cleaned_data['image'])
    predict_label, percentage = photo.predict()
    template = loader.get_template('carbike/show_prediction.html')
    context = {
        'image_name': photo.image.name,
        'image_data': photo.image_src(),
        'class_label': predict_label,
        'score': percentage,
    }
    return HttpResponse(template.render(context, request))
```