Japanese Car Number Plate Recognition using KNN

In this project I use two models:

- Wpod-net model to detect license plate
- KNN model to recognize character in plate

Libraries and packages:

- `keras`
- `tensorflow`
- `opencv-python==3.4.2.16`
- `pip>=19.2.3`
- `setuptools>=41.2.0`
- `django>=3.0.0`
- `psycopg2`
- `pillow`
- `matplotlib`

Database: postgresql(to store uploaded images)

.

At first detect the License plate from the car and cropped it.

In this step, use Wpod-Net to detect license plate.

Required libraries and packages

- cv2
- numpy
- matplotlib
- os.path/globz
- keras.models

After detecting and extracting number plate with wpod-net

All the extracted images are resized

```
img = cv2.imread(image_name, cv2.IMREAD_UNCHANGED)
image_bicubic = cv2.resize(img, (1280,720), cv2.INTER_CUBIC)
```

Resized images are converted to grayscale and processed by binary thresholding to get white background and black foreground and crop it.

```
grayImage = cv2.cvtColor(image_bicubic, cv2.COLOR_BGR2GRAY)
(thresh, blackAndWhiteImage) = cv2.threshold(grayImage, 117, 255, cv2.THRESH_BINARY)
cv2.imwrite(image_name, blackAndWhiteImage)
img = Image.open(image_name)
border = (80, 47, 58, 40) # left, up, right, bottom
ImageOps.crop(img, border).save(image_name)
```

```
img = Image.open(image_name)
border = (80, 47, 58, 40) # left, up, right, bottom
ImageOps.crop(img, border).save(image_name)
```

Cropped images (two line) are converted into one line image.

```
# Read the pixel-by-pixel color of the original image and convert from BGR to RGB
            pix_img = cv2.imread(image_name)
            pix_img = cv2.cvtColor(pix_img, cv2.COLOR_BGR2RGB)

# Extract all white columns from between place names and numbers
            all_white = np.full((width ,3), 255)
            white_index = []
            for i in range(int(height*0.2), int(height*0.8)):
                if (pix_img[i]==all_white).all(axis=0).any() == True:
                    white_index.append(i)

# Stores the top and bottom columns of the white columns
            try:
                upper_white_index = min(white_index)
                lower_white_index = max(white_index)
# If all white columns do not exist → Suspended as unprocessable
            except ValueError:
                print(image_name + " Cannot be processed.")
                sys.exit()

# Top: Trimming place names
            box = (int(width*0.2), 1, int(width*0.8), upper_white_index)
            place_image = original_image.crop(box)
            place_image = delete_margin(place_image)

# Bottom: Hiragana (small), number trimming
            box = (1, lower_white_index, width, height)
            num_hiragana_image = original_image.crop(box)

# Trimming only hiragana
            bottom_height = height - lower_white_index
            box = (1, int(bottom_height*0.2), int(width*0.2), int(bottom_height*0.8))
            hiragana_image = num_hiragana_image.crop(box)
            hiragana_image = delete_margin(hiragana_image)

# Trim only the number
            box = (int(width*0.2), 1, width, bottom_height)
            number_image = num_hiragana_image.crop(box)
            number_image = delete_margin(number_image)

# Joining and saving images
            bottom_image = resize_connect(hiragana_image, number_image)
```

```
# Joining and saving images
            bottom_image = resize_connect(hiragana_image, number_image)
            resize_connect(place_image, bottom_image).save(image_name, quality=95)
#print(image_name + 'Image processing is finished.')

            img = cv2.imread(image_name, cv2.IMREAD_UNCHANGED)
            scale_percent = 40 # percent of original size
            width = int(img.shape[1] * scale_percent / 100)
            height = int(img.shape[0] * scale_percent / 100)
            dim = (width, height)
# resize image
            resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
            cv2.imwrite(image_name, resized)
```

The result is :

The one line image is trained by KNN character recognition model.

```
kNearest.train(npaFlattenedImages, cv2.ml.ROW_SAMPLE, npaClassifications)
imgTestingNumbers = cv2.imread(image_name)          # read in testing numbers image
```

Recognizing number plate with KNN character recognition model

Trained KNN model with necessary Kanji, hiragana and numbers.

When you run the train program, you have to hit the correspondent char in your keyboard while the chars appears on screen. when I enter from the keyboard ,Kanjis and hiragans character are not match. So I configure as below:

```
intValidChars = [ord('0'), ord('1'), ord('2'), ord('3'), ord('4'), ord('5'), ord('6'), ord('7'), ord('8'), ord('9'),
                 ord('a'), ord('b'), ord('c'), ord('d'), ord('e'), ord('f'), ord('g'), ord('h'), ord('i'), ord('j'),
                 ord('k'), ord('l'), ord('m'), ord('n'), ord('o'), ord('p'), ord('q'), ord('r'), ord('s'), ord('t'),ord('u'),ord('v'),ord('w'),ord('x'),ord('y'),
                 ord('A'), ord('B'), ord('C'), ord('D'), ord('E'), ord('F'),ord('G'),ord('H'),ord('I'),ord('J'),ord('K'),ord('L'),ord('M'),ord('N'),ord('z')]
```

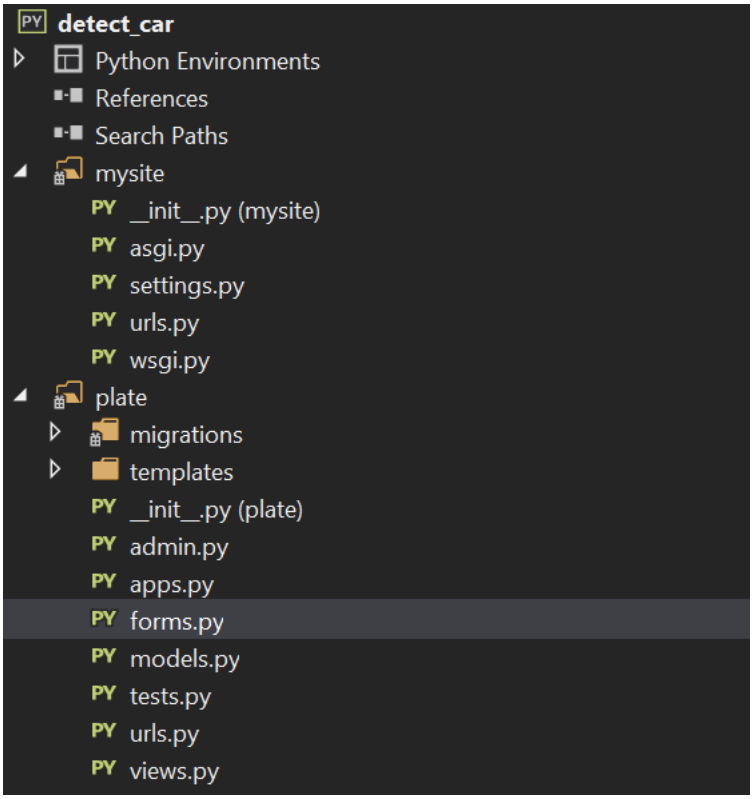
```
    elif intChar in intValidChars:
        if intChar == ord('a'):
            intChar = ord('横')

        if intChar == ord('b'):
            intChar = ord('浜')

        if intChar == ord('c'):
            intChar = ord('練')
```

After training kNN model, classification file and flattened_images file are generated.

Loading these files, number plates can be read.

Upload image in Django

Project's structure



Add MEDIA_ROOT and MEDIA_URL to the urlpatterns of urls.py files of Django project and Django app.

```
urlpatterns = [
    path('plate/',include("plate.urls")),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

```
urlpatterns = [
    path('', upload_view, name='image_upload'),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

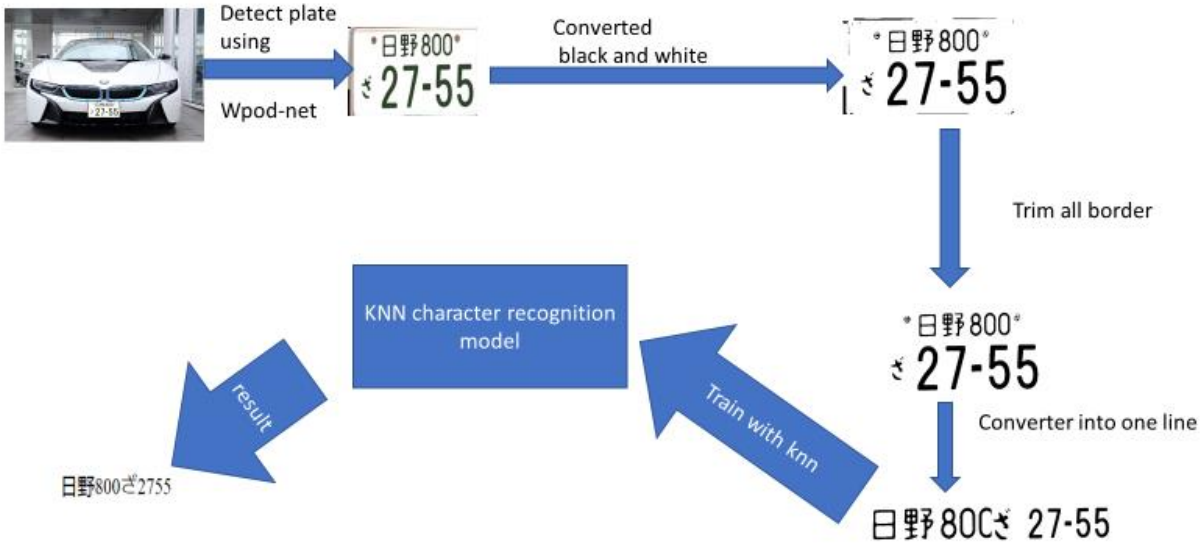
Models.py

```
# Create your models here.
class uploadimage(models.Model):
    upload_Img = models.ImageField(upload_to='uploads/')
```

Forms.py

```
from django import forms
from .models import *

class UploadForm(forms.ModelForm):

    class Meta:
        model = uploadimage
        fields = ['upload_Img']
```

Flow of Number Plate Recognition



After running :python manage.py runserver”