AIVA-Reconocimiento de crotales.

Entregable 2

2020

MARZO 2020

MUVA
Creado por:
Roberto Gallardo Cava
César González Fernández

Universidad Rey Juan Carlos

Contents

Introducción. 3

Funcionamiento del sistema. 4

Diseño del sistema. 5

- App 6
- Identifier 6
- BaseReader 6
- PretrainedReader 6
- BaseDigitExtractor 6
- SplitDigitExtractor 6
- BaseDigitRecognition 7
- TesseractDigitRecognition 7
- PretrainedModelDigitRecognition 7
- Excepciones 7
- InvalidFile 7
- ExceptionIntCastFailure 7
- NotDigitRecognized 7

Secuencia de funcionamiento del sistema. 8

Actividad del sistema. 9

Introducción.

La aplicación desarrollada será capaz de extraer el identificador del animal a partir de una imagen del cotral de dicho animal. En este cotral, deberá aparecer un identificador numérico diferenciable de otros dígitos que puedan aparecer en este elemento. El tipo de crotal y su formato no es estándar, por lo que la aplicación debe ser capaz de solventar este problema. Por tanto, para ser capaz de superar esta problemática, se acepta suponer que el número de identificación es el número de mayor tamaño.

La propuesta desarrollada ofrece las herramientas necesarias para, dada la imagen de un crotal, sea capa de extraer el identificador. Esta herramienta podrá ser utilizada a través de línea de comandos, o ser usada como módulo de Python en otras aplicaciones como puede ser una aplicación web.

Para poder reconocer un dígito a partir de una imagen de este, se han planteado dos alternativas diferentes:

1. **Utilizar el motor de reconocimiento óptico *Tesseract*.**
2. **Utilizar un modelo *Deep Leerning* preentrenado.**

En esta primera versión, se ha implementado la primera de estas alternativas, dejando el sistema preparado para añadir la segunda de ellas en una futurible versión.

Funcionamiento del sistema.

Para poder procesar la imagen, la herramienta desarrollada sigue una serie de pasos:

La herramienta lee la imagen a procesar.

Se aplica un umbralizado a la imagen, y se aplican diversas operaciones morfológicas para poder hacer los caracteres más identificables.

1. **A partir de la imagen tratada, se aplican técnicas de procesamiento de imagen, detecta los *bounding boxes*(BBs) susceptibles de contener el identificador.**

2. **De entre todos estos BBs, se selecciona aquel más probable de contener el identificador a partir de su posición y tamaño.**

3. **Los dígitos de esta zona son recortados en imágenes individuales.**

4. **Estos dígitos, ya recortados, son reconocidos extrayendo su valor numérico y construyendo la solución.**

Diseño del sistema.

El software se ha desarrollado siguiendo los principios **SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion),** con la idea de facilitar futuribles extensiones y mejoras del mismo. Para ello, se han definido distintas clases Base desde las cuales heredan las clases que implementan las distintas acciones. Además, estas clases han sido diseñadas teniendo un propósito único y bien definido.

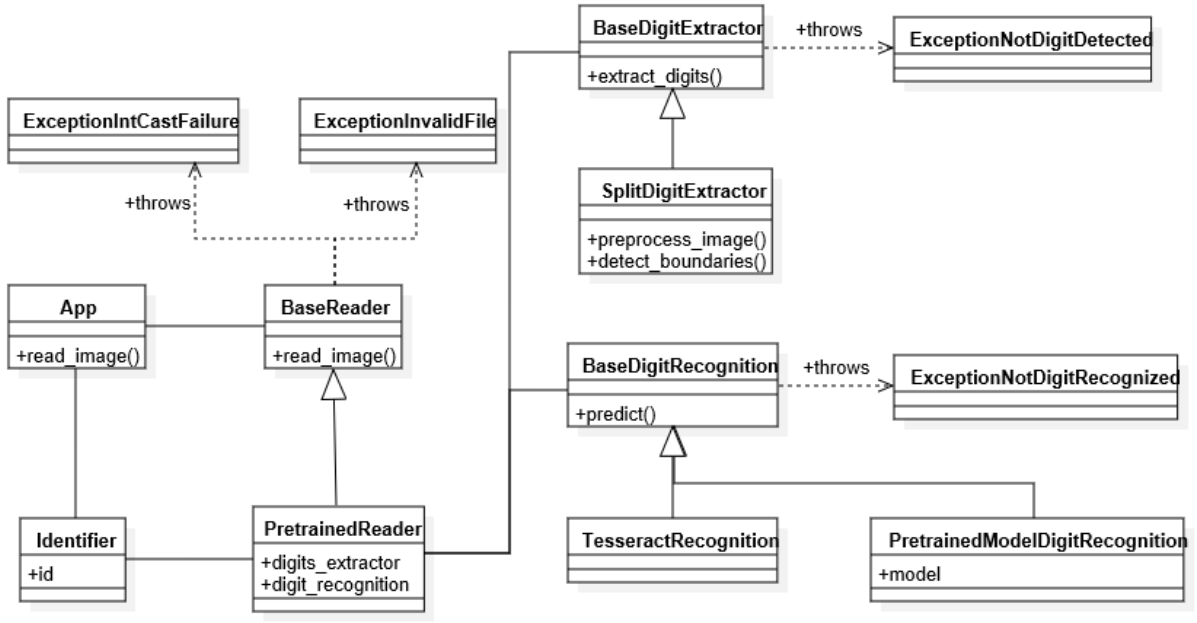

Figura 1. Diagrama de clases.

En la Figura 1 se representan las clases definidas en este proyecto. A continuación, se detalla cada una de estas clases.

App

Clase a través de la cual el usuario se conecta al sistema para poder ser usado. Esta clase implementará la interfaz que el sistema ofrece al usuario para ser usado. Esta clase hace uso de un Reader, a la cual hará peticiones para extraer la identificación a partir de la imagen que le llega desde el usuario. La respuesta a esta petición será un objeto de la clase *Identifier*, que, tras procesarla, devolverá este identificador en un formato que pueda entender el usuario.

Identifier

Clase que representa el identificador extraído desde la imagen de un crotal. En esta primera versión, este identificador consta del número que identifica al animal, pudiendo ser extendida en futuras versiones para almacenar más datos extraídos del crotal.

BaseReader

Clase de la cual tendrán que heredar las distintas clases que implementen la lógica capaz de, a partir de la imagen de un crotal, extraiga el identificador del animal. Consta de una función, *read_image*, la cual espera como entrada una imagen, y devuelve un objeto *Identifier*.

PretrainedReader

Clase que hereda de la clase Reader y que implementa la lógica implementada en esta primera versión. Tiene como atributos un objeto de la clase *SplitDigitExtractor* (*digit_extractor*) y un objeto de la clase *PretrainedModelDigitRecognition (digit_recognition)*. Esta clase implementa la función *read_image*, la cual utiliza *digits_extractor* para extraer los dígitos del identificador de una detectados en una imagen, y el objeto *digits_recognition*, para identificar el valor numérico correspondiente a cada uno de estos dígitos.

BaseDigitExtractor

Base de las clases que se cuyo objetivo sea reconocer, en la imagen de un crotal, aquellas secciones de esta que representen los dígitos que forman el identificador. Cuenta con una función, *extract_digits*, la cual espera como entrada la imagen del crotal y devuelve una lista de las coordenadas de los BBs de los dígitos detectados.

SplitDigitExtractor

Clase que hereda de *BaseDigitExtractor*. Implementa algoritmo de búsqueda del mejor BBs en *extract_digits*, que hará uso de *preprocess_image* para el umbralizado de la misma y facilitar la detección con el uso de operaciones morfológicas. Se hará uso de *detect_boundaries* para localizar los dígitos por separado.

BaseDigitRecognition

Clase base de la que heredan aquellas clases que implementan la lógica con la cual, a partir de la imagen de un dígito, es capaz de inferir el valor de este dígito. Define la función *predict*, la cual, a partir de la imagen de un dígito, es capaz de devolver el valor numérico de este.

TesseractDigitRecognition

Clase que hereda de la clase *BaseDigitRecognition* y que utiliza la herramienta *Tesseract* para identificar dígitos de una imagen. Esta clase implementa en la función *predict* el código necesario para interactuar con la herramienta *Tesseract*, la cual predice el dígito a partir de la imagen pasada.

PretrainedModelDigitRecognition

Clase que extiende la clase *BaseDigitRecognition* y que utiliza un modelo de *Deep Learning preentrenado* para inferir el dígito que aparece en una imagen. Este modelo es guardado en el atributo model de esta clase. En la función *predict* se encuentra la lógica necesaria para ejecutar este modelo.

Excepciones

Se han definido una serie de excepciones que son lanzadas por las distintas clases en caso de producirse un error. La definición de distintas clases de excepciones facilita la detección y tratamiento de excepcione.

InvalidFile

Excepción lanzada por el Reader si el fichero a procesar no es válido.

ExceptionIntCastFailure

Error que se produce en caso de que alguno de los dígitos reconocidos no pueda ser transformado en número.

NotDigitDetected

Es lanzada por el *DigitExtractor* en caso de no ser posible detectar dígitos en la imagen.

NotDigitRecognized

Lanzada por el *DigitRecognition* en no es capaz de identificar el digito en la imagen pasada.

Secuencia de funcionamiento del sistema.

La Figura 2 presenta un diagrama de secuencia que muestra el comportamiento del sistema. Esta muestra la interacción entre las distintas clases que conforman el sistema, desde el momento en el que el usuario realiza una petición a este, hasta que el resultado es devuelto.

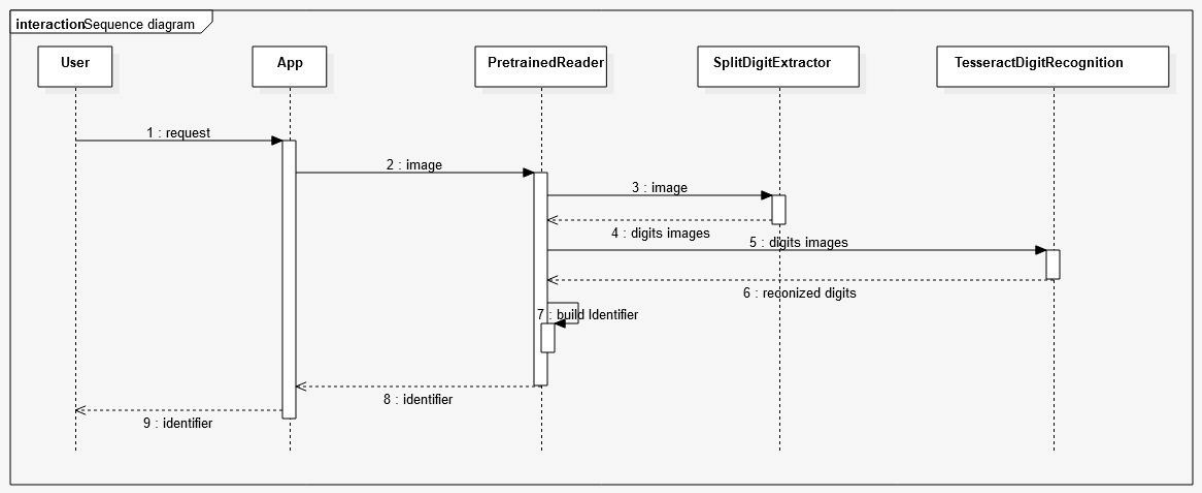

Figura 2. Diagrama de secuencia.

En primera instancia, el usuario realiza una petición a *App* para que evalúe una (o varias) imágenes. *App* dispone de un *Reader,* en esta primera implementación, *PretrainedReader* para que, a partir de la imagen del crotal, le devuelva un identificador. Este *Reader*, utiliza la clase *SplitDigitExtractor* para extraer una lista de dígitos identificados en este crotal. Esta lista de imágenes de dígitos es enviada al objeto de la clase *TesseractDigitRecognition* para que identifique el valor numérico de estos dígitos.
Una vez que el *Reader* dispone de los valores numéricos, construye un objeto que de la clase *Identifier* la cual devuelve como respuesta a la *App.* Por último, la *App* devuelve este identificador de forma que pueda ser interpretable por el usuario.

Actividad del sistema.

El diagrama que muestra la Figura 3 el diagrama de actividad del sistema.

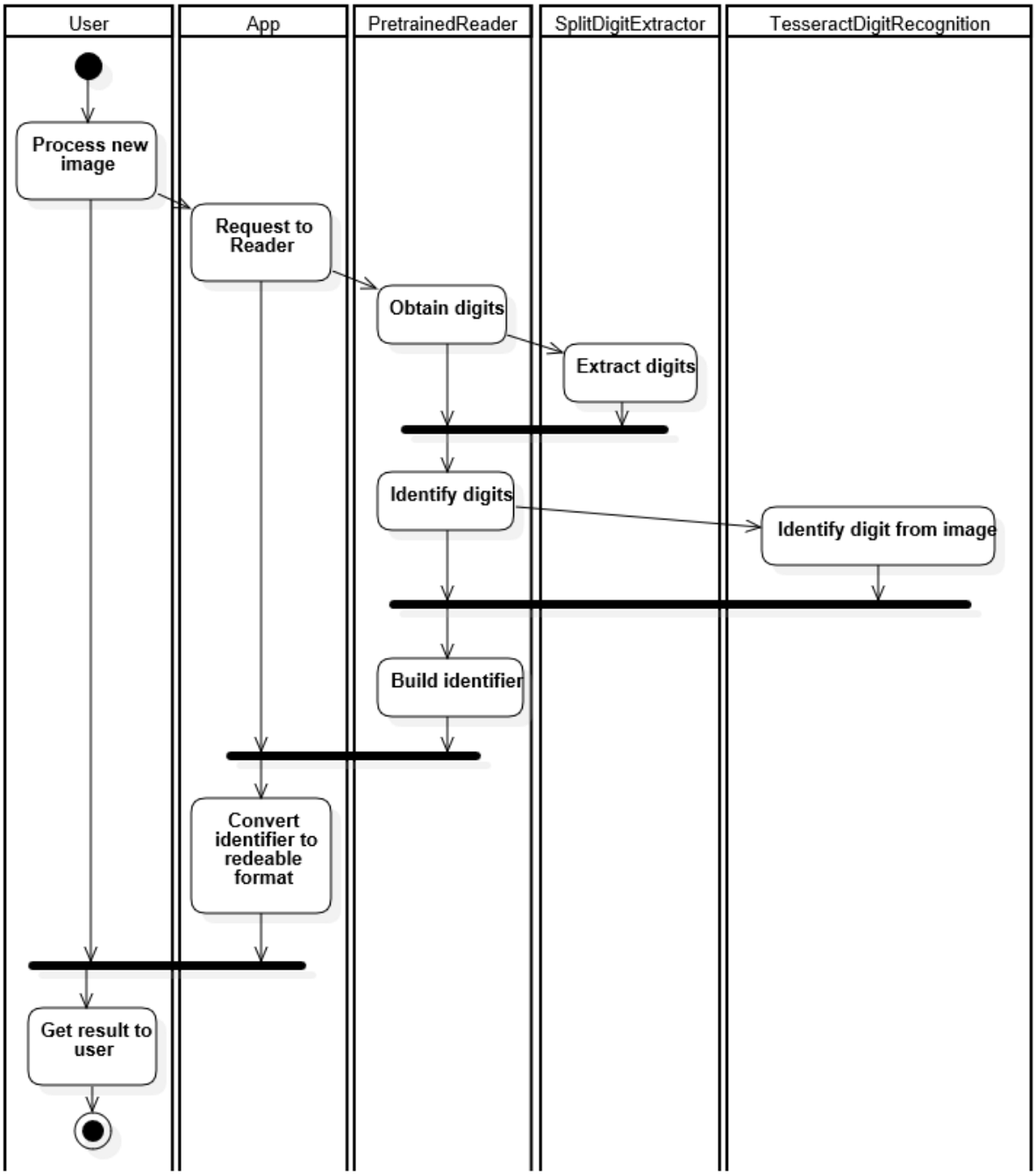

Figura 3. Diagrama de actividad.

Este diagrama refleja el flujo de trabajo que sigue el sistema y en qué momento, cada una de las clases involucradas realiza su actividad.

Dado que el sistema ha sido diseñado para que cada clase desarrolle una única función, siendo el *Reader* el encargado de utilizar al *DigitExtractor* y al *DigitRecognition* para construir el identificador. El

Reader es invocado desde la *App*, la cual espera hasta que el *Reader* tenga listo el *Identifier.* Una vez esté disponible, el *User* que es el iniciador del todo el proceso, y acaba recibiendo, desde la *App,* el resultado del procesamiento de la imagen.

Gracias a este diseño, cada clase solo se comunica y tiene relación con un subconjunto acotado del resto de clases, sin tener que realizar tareas para las cuales no han sido ideadas. Así por ejemplo, la clase *App* en ningún momento interacciona con el *DigitExtractor* o con el *DigitRecognition,* siendo estos últimos transparentes para la primera.