INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Trabajo TerminaI

“Prototipo de aplicación para la detección de deficiencia de nutrientes en cultivos de hidroponía”

*2017-A054*

Presenta:

Edgar Rodrigo Arredondo Basurto

Directores:

Ing. Eduardo Gutiérrez Aldana Dr. José Félix Serrano Talamantes



*Mayo de 2018*

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADÉMICA

No de TT:2017-A054 de mayo de 2017

Documento técnico

“Prototipo de aplicación para la detección de deficiencia de nutrientes en cultivos de hidroponía”

*2017-A054*

Presenta:

Edgar Rodrigo Arredondo Basurto1

Directores:

Ing. Eduardo Gutiérrez Aldana Dr. José Félix Serrano Talamantes

RESUMEN

El objetivo de este proyecto es desarrollar un prototipo de aplicación de visión por computadora que tome como entrada imágenes de hojas de cultivos con una enfermedad visible y como resultado se obtiene un probable diagnóstico de la enfermedad de la planta, bajo ciertas restricciones de plantas y enfermedades. Es así que en este documento se describen el análisis, diseño y construcción del sistema que cumple con dicho objetivo.

Palabras clave: Clasificador, descriptor, machine learning, deep learning, red neuronal artificial.

I. Resumen

El objetivo de este proyecto es desarrollar un prototipo de aplicación de visión por computadora que analice imágenes de hojas de cultivos que presenten síntomas visibles de alguna enfermedad. Como resultado se obtiene un probable diagnóstico de cuál es la enfermedad que presenta la planta.

Es así como en este documento se definen y diseñan las diferentes etapas que requiere este prototipo para su implementación, así como detalles y evidencias del desarrollo y puesta a punto del mismo.

En el capítulo 1 se presenta una breve introducción en la que se explica el contexto de este proyecto y la motivación que dio origen a su desarrollo. Además, se definen los objetivos para la etapa de diseño del prototipo.

En el capítulo 2 se exponen algunos fundamentos teóricos útiles para entender de mejor forma el funcionamiento de una de las etapas fundamentales del prototipo, que es la detección de objetos durante la etapa de segmentación. En esta etapa se aíslan las hojas del resto de la imagen, ya que son los objetos que contienen la información útil para esta aplicación.

Inmediatamente después, en el capítulo 3 se abordan las etapas de captura de la imagen y segmentación. Aquí se definen el tipo de objetos a capturar y ciertas restricciones que debe tener la imagen que el usuario proporciona, como la especie de planta, el contenido de la imagen y la posición de las hojas dentro de la imagen. En la segmentación se presenta el método de detección de objetos seleccionado.

En el capítulo 4 se presenta lo relacionado a la clasificación de los objetos, incluyendo la selección del descriptor y el algoritmo de clasificación. En este capítulo se delimita cuáles son los nutrientes por identificar por este prototipo, siendo estos el potasio, nitrógeno, azufre y magnesio.

Finalmente, en el capítulo 5 se aborda el tema de las herramientas y lenguaje de programación para implementar el prototipo. Todas las etapas relacionadas a la visión por computadora se desarrollarán con el apoyo de la biblioteca de OpenCV la cual ofrece soporte para C++, java y python.

II. Abstract

The purpose of this project is the developing of a computer vision application that analyze plant leaves images with a visible disease. As the result, the application gives a probably diagnostic of the plant disease.

In this document are defined and designed the different phases that this prototype needs for its implementation.

In the chapter 1 is presented a brief introduction where is explained the context of this project and the motivation behind the its development. Also, the objectives for the design phase are presented.

In the chapter 2 is found the theoretical framework the object detection topic is discussed. The object detection is required in the segmentation step, where the leaves are separated from the background of the image, because the leaves contains the useful information for the next steps of the application.

In the chapter 3 are discussed the image capture and segmentation steps. In this chapter are defined the object type to capture and some restrictions about the captured image, such as the plant species, the image’s contents and the position of the leaves inside the image. In the segmentation section is presented the object detection method chosen.

In the chapter 4 is presented the feature descriptor and the image classifier. Also, in this chapter is defined the deficiencies to detect in the images, which are the potassium, nitrogen, sulfur and magnesium deficiencies, by reasons that are discussed in that section.

Finally, in the chapter 5 are discussed the developing tools and the programming language for the implementation. All the steps related to computer vision are going to be developed with the support of the OpenCV library, which support the C++, java and python languages.

III. Índice general

[I. Resumen 3](#_Toc497097033)

[II. Abstract 4](#_Toc497097034)

[III. Índice general 5](#_Toc497097035)

[Capítulo 1. Introducción 6](#_Toc497097036)

[1.1 Objetivos. 9](#_Toc497097037)

[1.1.2 Objetivo general. 9](#_Toc497097038)

[1.1.3 Objetivos particulares. 9](#_Toc497097039)

[1.2 Justificación. 9](#_Toc497097040)

[1.3 Trabajo propuesto. 10](#_Toc497097041)

[Capítulo 2. Marco teórico 12](#_Toc497097042)

[2.1 Detección de objetos mediante visión por computadora. 12](#_Toc497097043)

[2.1.1 Detección de objetos Viola-Jones 12](#_Toc497097044)

[2.1.2 Clasificación SVM con descriptores de histogramas de gradientes orientados (HOG) 14](#_Toc497097045)

[Capítulo 3. Captura y segmentación de la imagen. 15](#_Toc497097046)

[3.1 Imagen adquirida 16](#_Toc497097047)

[3.2 Segmentación 18](#_Toc497097048)

[Capítulo 4. Extracción de descriptores y clasificación. 21](#_Toc497097049)

[4.1 Selección de descriptores 22](#_Toc497097050)

[4.2 Clasificador 23](#_Toc497097051)

[Capítulo 5. Implementación. 25](#_Toc497097052)

[5.1 Lenguaje de programación 25](#_Toc497097053)

[5.2 Arquitectura del sistema 26](#_Toc497097054)

[5.3 Diseño de la aplicación. 28](#_Toc497097055)

[Referencias. 30](#_Toc497097056)

Capítulo 1. Introducción

Introducción.

En las siguientes secciones se proporciona información general acerca del contexto de este proyecto; la situación que dio origen a su desarrollo, el alcance y objetivos del mismo, la metodología de desarrollo y el marco teórico relacionado con la propuesta solución planteada para el problema de la clasificación de enfermedades de plantas con base en el análisis de imágenes.

* 1. Planteamiento del problema.

La hidroponía es una rama de la agricultura. Es una técnica de producción intensiva de plantas, que se caracteriza por abastecer el agua y los nutrientes de manera controlada y de proporcionar a las plantas los elementos nutritivos en las concentraciones y proporciones más adecuadas, a través de una solución de nutrientes minerales. Para su aplicación se utilizan sustratos inertes diferentes al suelo agrícola a los que se les adiciona en forma constante una solución nutritiva, preparada a partir de fertilizantes comerciales; con esto se logra un medio que proporciona las condiciones físicas, químicas y sanitarias más adecuadas para el desarrollo de los cultivos.

La hidroponía ha sido utilizada en forma comercial desde hace 50 años y se ha adaptado a diferentes situaciones, tanto con cultivos al aire libre como bajo condiciones de invernadero. Este sistema de producción es importante porque permite cultivar especies para el consumo humano en regiones donde no existen tierras de cultivo o donde el clima no es favorable para el cultivo tradicional de ciertas especies [1].

Las ventajas anteriores han favorecido el mercado global de la hidroponía, el cual fue valuado en 411.88 millones de dólares (USD) en el año 2017 y se espera que crezca a una tasa anual de crecimiento compuesta (CARG) de 12.81% para alcanzar un mercado de 752.57 millones de dólares en el año 2022 [2].

Sin embargo, en México el 60% de los invernaderos de hidroponía que se han instalado han fracasado ante el desconocimiento de productores, la falta de capacitación de técnicos y de mercado [3]. Parte de este problema es el mantener saludables las plantas, ya que la deficiencia de algún nutriente en la solución puede causar que los frutos no sean aptos para consumo humano, o en casos extremos, que la planta muera. Además existen otro tipo de enfermedades derivadas de bacterias o insectos que una planta puede padecer. Identificar la enfermedad tomando como referencia los síntomas visibles que presenta la planta no es una tarea sencilla, ya que requiere de conocimientos y experiencia para realizar un diagnóstico preciso.

El prototipo propuesto tiene como finalidad ayudar al agricultor a diagnosticar posibles enfermedades de plantas con base en los síntomas visibles que se presenten. Como se verá más adelante, la solución propuesta permite clasificar diferentes tipos de enfermedades y plantas, siempre y cuando se tenga una cantidad suficiente de imágenes y una computadora con los recursos suficientes para procesarlas. En dicha solución se considera un conjunto de datos de 14,828 imágenes clasificadas en nueve enfermedades que presenta el tomate.

1.2 Objetivos.

Los objetivos de este proyecto están clasificados en un objetivo general y en diversos objetivos para metas particulares. Dichos objetivos se describen a continuación.

1.2.1 Objetivo general.

Diseñar y desarrollar el prototipo de una aplicación de visión por computadora que analiza imágenes de hojas de tomate con una anormalidad visible y realiza un diagnóstico de una posible enfermedad, bajo un subconjunto predefinido de enfermedades del tomate.

1.2.2 Objetivos particulares.

* Entrenar el modelo de clasificación de imágenes con el conjunto de datos predefinido.
* Realizar pruebas de eficiencia del clasificador, obteniendo un resultado superior al 90%.
* Implementar un sistema web en el que se aloje el clasificador y permita realizar identificaciones de enfermedades a los usuarios.

1.3 Justificación.

La arquitectura general de sistemas de clasificación de enfermedades de plantas a través del análisis imágenes consiste de tres etapas: pre-procesamiento, extracción de descriptores y clasificación [3].

En el pre-procesamiento las imágenes son preparadas usando algunas operaciones, tales como conversión de espacio de color de RGB a algún otro. También suele incluirse la eliminación el fondo de la imagen tratando de concentrar el análisis en el objeto de interés. Desafortunadamente, este tipo de operación es complicada y algunas veces requiere de la intervención del usuario, lo cual decrece la automatización del sistema.

Los descriptores propuestos por expertos son extraídos de la imagen para formar un vector descriptor. Ejemplos de descriptores son los momentos de color o de textura obtenidos a partir de la matriz de co-ocurrencia. El objetivo de un descriptor es, a partir de su valor, poder clasificar un objeto entre distintas clases.

La última etapa determina a que clase pertenece el objeto no identificado a través de un modelo o algoritmo de clasificación. Este modelo debe ser entrenado usando algoritmos de aprendizaje y muestras de imágenes pre-clasificadas (muestras etiquetadas). Ejemplos de este tipo de algoritmos son la Máquina de Vectores de Soporte (SVM), los k-vecinos más cercanos (KNN) y la Red Neuronal Artificial (ANN).

Las etapas de pre-procesamiento y extracción de descriptores implican operaciones que suelen ser complejas y consumir tiempo considerable. Además, debido a esto es común solicitar asistencia del usuario lo que hace que el sistema no esté completamente automatizado.

En este proyecto se plantea el uso de un modelo de aprendizaje profundo (en adelante llamado deep learning), específicamente de una Red Neuronal Convolucional (CNN) como una alternativa a la clasificación de enfermedades en plantas. Este modelo permite que los descriptores sean definidos y obtenidos de una forma completamente automática. Además permite realizar la etapa de entrenamiento y clasificación directamente sobre las imágenes, sin incluir operaciones de pre-procesamiento. Las características anteriores permitirán desarrollar un sistema completamente automatizado en el que la única acción que el usuario tendrá que realizar es proporcionar la imagen de la hoja enferma.

1.4 Metodología.

El objetivo de este proyecto y los métodos para alcanzarlo están bien definidos, tal como se describió en las secciones anteriores. Teniendo esto en cuenta y considerando que hay dos etapas fácilmente identificables en el desarrollo del proyecto (el clasificador y el desarrollo del sistema web) se plante un modelo incremental de desarrollo de software.

El primer incremento consiste en el análisis, diseño y construcción del clasificador haciendo uso de un algoritmo CNN. El segundo incremento consiste en el análisis, diseño y construcción del sistema web en el que será integrado el clasificador para que esté disponible a los usuarios. Lo anterior se encuentra representado en la figura 1.1.



Figura 1.1 Metodología de desarrollo propuesta.

En los capítulos siguientes se desarrolla cada una de las etapas del modelo incremental propuesto.

Capítulo 2. Marco teórico

Marco teórico.

El deep learning es una tendencia reciente en al aprendizaje automático (machine learning), el cual ha tenido éxito en áreas como la visión por computadora, el reconocimiento automático de voz y el procesamiento de lenguaje natural.

2.1 Comparación de clasificación de imágenes usando modelos tradicionales de Machine Leaning y Deep Learning.

La clasificación de imágenes usando algoritmos de machine learning está compuesta de dos fases:

* Fase de entrenamiento. En esta fase se entrena el algoritmo usando un conjunto de datos pre-clasificados (muestras etiquetadas).
* Fase de predicción. En esta fase se utiliza el algoritmo entrenado para predecir la etiqueta de imágenes fuera del conjunto de entrenamiento.

La fase de entrenamiento para el problema de clasificación de imágenes tiene dos etapas principales:

1. Extracción de descriptores. En esta etapa se hace uso del conocimiento general en el área para seleccionar y extraer descriptores que serán usados por el algoritmo de machine learning, de acuerdo con el tipo de imágenes a clasificar. HOG y SIFT (Histograma de Gradientes Orientados y Transformación de Características Invariante a la Escala respectivamente) son ejemplos de descriptores usados en la clasificación de imágenes.
2. Entrenamiento del modelo. En esta etapa se hace uso de un conjunto de entrenamiento compuesto por descriptores de imágenes y sus etiquetas correspondientes para entrenar el modelo de machine learning.

En la fase de predicción, se aplica el mismo proceso de extracción de descriptores a imágenes nuevas y los descriptores obtenidos se pasan al algoritmo de machine learning para predecir su etiqueta o clase a la que pertenece. Ambas etapas, entrenamiento y predicción se muestran de forma esquemática en la figura 2.1.

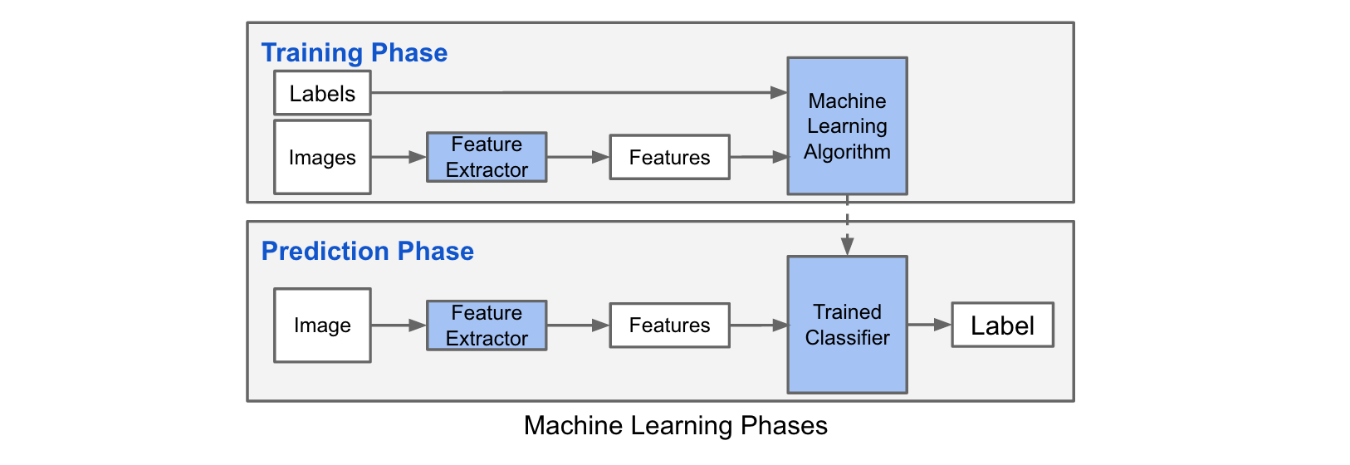


Figura 2.1 Clasificación con algoritmos de machine learning.

Ahora, la principal diferencia de los algoritmos tradicionales de machine learning y los algoritmos de deep learning radica en la ingeniería de descriptores. En los algoritmos tradicionales de machine learning este proceso es elaborado manualmente, es decir, la selección y extracción debe ser diseñada e implementada por el programador. En cambio, en los algoritmos de deep learining la ingeniería de descriptores es realizada automáticamente por el algoritmo. La ingeniería de descriptores es costosa, consume tiempo importante y requiere de cierta experiencia. Un descriptor mal seleccionado condena al fracaso el resto del clasificador. Es así como los algoritmos de machine learning prometen resultados más precisos comparados con los algoritmos tradicionales de machine learning, con menos o incluso sin ingeniería de descriptores. Esta diferencia se muestra en la figura 2.2.

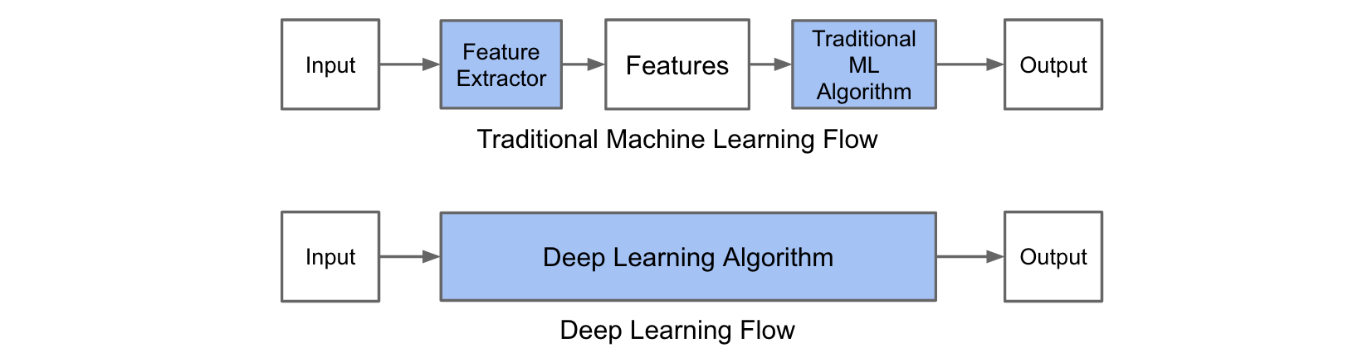


Figura 2.2 Diferencias entre la clasificación con modelos de machine learning y deep learning.

2.2 Redes Neuronales Artificiales (ANN).

Deep learning se refiere a una clase de redes neuronales artificiales compuestas de muchas capas de procesamiento. Las redes neuronales tienen décadas de existencia pero el reciente éxito y popularidad del deep learning puede rastrearse a la publicación “ImageNet Classification with Deep Convolutional Networks” realizada por Krizhevsky, Sutskever, and Hinton en el año 2012. En dicha publicación demostraron por primera vez como una CNN puede superar en desempeño, de forma contundente, a otros métodos tradicionales de clasificación de imágenes. Además, factores como el incremento del poder computacional de las computadoras, el uso de GPUs y la disponibilidad de grandes conjuntos de datos han favorecido el desarrollo de los modelos de deep learning.

Las neuronas artificiales están inspiradas en las neuronas biológicas. Una neurona artificial tiene un número finito de entradas con pesos asociadas a ellas, y una función de activación. La salida de la neurona es el resultado de la función de activación aplicada a la suma de las entradas con los pesos correspondientes (figura 2.3). Las neuronas artificiales se conectan entre sí para formar redes neuronales artificiales.

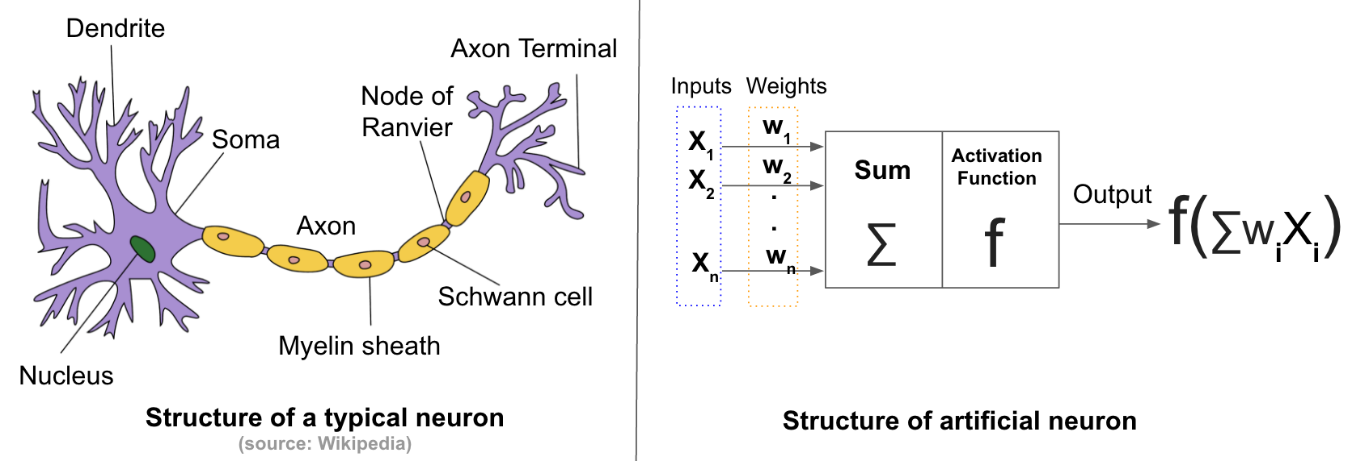


Figura 2.3 Estructura de una neurona artificial.

2.2.1 Redes neuronales unidireccionales.

Las redes neuronales unidireccionales (feedforward) son las más sencillas de las redes neuronales artificiales. Estas redes tienen tres tipos de capas, capa de entrada, capa oculta y capa de salida. Los datos se mueven de la capa de entrada hacia las neuronas ocultas y desde estas hacia las neuronas de salida. En la figura 2.4 se muestra un ejemplo de una red unidireccional en la que cada nodo o neurona está conectada a todas las de la siguiente capa. Este tipo de conexión es llamado “Completamente conectado” (fully-connected). El número de capas ocultas y su tamaño es variable. Mientras más grandes y de mayor profundidad sean las capas ocultas, pueden modelarse patrones más complejos.



Figura 2.4 Red neuronal unidireccional con dos capas ocultas.

2.2.2 Funciones de activación.

Las funciones de activación transforman la suma de las entradas con los pesos aplicados. Estas funciones deben ser no lineales para permitir codificar patrones de complejos de datos. Las funciones de activación más comunes son las funciones Sigmoidal, tangente hiperbólica y ReLU (figura 2.5). En las redes neuronales profundas (con muchas capas ocultas) la función más usada es la ReLU.



Figura 2.5 Funciones de activación Sigmoidal, tanh y ReLU.

2.2.3 Entrenamiento de una red neuronal artificial.

El objetivo de la fase de entrenamiento es determinar los valores de los pesos de la red neuronal. Para alcanzar este objetivo son necesarios dos elementos:

* El conjunto de datos de entrenamiento. En el caso de clasificación de imágenes el conjunto de datos está compuesto de imágenes y sus correspondientes etiquetas.
* Una función de pérdida. Está función mide la imprecisión de las predicciones.

Con estos dos elementos se entrena la red neuronal con un algoritmo llamado backpropagation [4].

2.3 Redes Neuronales Convolucionales (CNN).

Las redes neuronales convolucionales son un tipo especial de redes unidireccionales. Estos modelos están diseñados para emular el comportamiento de la corteza visual, por lo que asumen que la entrada de la red son imágenes. Las CNNs tienen un muy buen desempeño en tareas de reconocimiento visual. La arquitectura de una CNNs está compuesta por capas tres capas: Capa convolucional (CONV layer), capa de agrupamiento (pooling layer) y capa conectada completamente (Fully-connected layer). Estas capas permiten a la red codificar determinadas propiedades de la imagen. En la figura 2.6 se muestra la estructura de la CNN LeNet.



Figura 2.6 CNN LeNet.

2.3.1 Capa convolucional.

La capa convolucional es el bloque fundamental con que se construye una red convolucional y donde se realiza la mayor parte del trabajo computacional de la red. Los parámetros de la red convolucional consisten de un conjunto de filtros aprendibles. Cada filtro es pequeño espacialmente, pero es desplazado por toda la imagen calculando la convolución entre el filtro y la imagen de entrada. Conforme el filtro se desplaza por el alto y ancho de la imagen, se producen mapas de activación de dos dimensiones que indican la respuesta de dicho filtro en cada posición espacial. De forma intuitiva la red aprenderá los filtros que se activan cuando se encuentran algún tipo de característica visual, tales como bordes, orientaciones o regiones de algún color. Lo anterior en las primeras capas de la red. En capas superiores pueden detectarse características más complejas como ruedas, señales de tránsito, etc.

Cada neurona en la capa convolucional está conectada a una región local espacial de la imagen de entrada, pero considera todos los canales de color. En la figura 2.7 se tienen cinco neuronas de la capa convolucional, todas ellas conectadas a la misma región de la imagen de entrada.



Figura 2.7 Conectividad local de las neuronas de la capa convolucional.

2.3.2 Capa de agrupamiento.

Es común en una arquitectura CNN insertar periódicamente capas de agrupamiento entre capas convolucionales sucesivas, con la intención de progresivamente reducir el tamaño espacial de la representación para reducir la cantidad de parámetros y el cómputo en la red. De esta forma se controla el sobreajuste, para evitar entrenar de más la red.

La capa de agrupamiento ajusta el tamaño espacial empleando la operación MAX. La forma más común es tomando un filtro de 2x2 aplicado a toda la entrada espacialmente. De esta forma se reduce en un 75% las activaciones (figura 2.8).



Figura 2.8 Reducción del volumen en la capa de agrupamiento.

2.4 Arquitectura de las redes neuronales convolucionales.

La arquitectura más sencilla de una red neuronal convolucional comienza con una capa de entrada (imágenes) seguida de una secuencia de capas convolucionales y de agrupamiento, terminando con capas completamente conectadas. Las capas convolucionales usualmente están seguidas de una capa de funciones de activación ReLU (figura 2.9).

Las capas convolucionales, de agrupamiento y ReLU actúan como extractores de descriptores aprendibles, mientras que las capas completamente conectadas actúan como un clasificador de machine learning. Además, las primeras capas de la red codifican patrones genéricos de las imágenes, mientras que las capas posteriores codifican patrones detallados de las imágenes.

También debe mencionarse que solo las capas convolucionales y completamente conectadas tienen pesos, los cuales son aprendidos en la etapa de entrenamiento [5].



Figura 2.9 Ejemplo de arquitectura CNN.

Existen distintas arquitecturas con un nombre asignado. Las más comunes son las siguientes:

* LeNet. Desarrollada por Yann LeCun en la década de 1990. Con esta arquitectura se desarrollaron las primeras aplicaciones exitosas de redes convolucionales.
* AlexNet. Es el primer trabajo que popularizo las redes convolucionales en la visión por computadora, desarrollado por Alex Krizhevsky, Ilya Sutskever y Geoff Hinton. La AlexNet fue inscrita en el reto ImageNet ILSVRC challenge en 2012 y superó de forma significativa al segundo clasificado.
* ZF Net. Una mejora de la AlexNet que resultó ganadora del ILSVRC 2013, desarrollada por Matthew Zeiler y Rob Fergus.
* GoogLeNet. La ganadora del ILSVRC 2014, desarrollada por Szegedy et al [6] de Google.

Capítulo 3. Análisis del sistema.

Análisis del sistema.

En esta sección se describen los requisitos de hardware y software, se presenta un modelos de caso de usos y la especificación de la interfaz gráfica.

Esta sección se realizó teniendo en consideración el conjunto de datos que se logró recopilar. Este conjunto de datos es un subconjunto del repositorio de acceso abierto publicado en [www.PlantVillage.org](http://www.PlantVillage.org). En el momento del desarrollo de este proyecto, dicho repositorio se encontraba temporalmente inhabilitado, por lo que las imágenes usadas en este proyecto fueron facilitadas por Brahimi M. [7].

El conjunto de datos usados en este sistema está formado por 14,828 imágenes divididas en nueve enfermedades del tomate, tal como se describe en la tabla 3.1.

|  |  |
| --- | --- |
| Enfermedad | Número de imágenes |
| Virus del rizado amarillo del tomate (Tomato yellow leaf curl virus) | 4032 |
| Virus del mosaico del tomate (Tomato mosaic virus) | 325 |
| Corynespora cassiicola. Mancha en forma de blanco. (Target spot) | 1,356 |
| Araña roja (Spider mites) | 1,628 |
| Septoriosis (Septoria spot) | 1,723 |
| Passalora fulva. Moho en la hoja (Leaf mold) | 904 |
| Tizón tardío (Lateblight) | 1,781 |
| Tizón temprano (Earlyblight) | 952 |
| Mancha bacteriana (Bacterial spot) | 2,127 |
| Total | 14,828 |

3.1 Requisitos del sistema.

Este proyecto nació de la necesidad de poder identificar de forma automática enfermedades en plantas, a través de síntomas visuales. Y además, teniendo en consideración los objetivos y el alcance de este proyecto, los requisitos de software de este prototipo se describen a continuación.

3.1.1 Requisitos de software.

Requisitos funcionales.

**RF01.** Selección de imagen.

**Nivel de madurez:** Alta.

**Prioridad:** Media.

**Descripción:** El sistema permitirá seleccionar una imagen del sistema de archivos local del dispositivo del usuario.

**RF02.** Clasificación.

**Nivel de madurez:** Alta.

**Prioridad:** Alta.

**Descripción:** El sistema clasificará la imagen en alguna de las nueve enfermedades descritas en la tabla 3.1, indicando la probabilidad de que esa sea la enfermedad correcta.

Requisitos no funcionales.

**RNF01**. La eficiencia de clasificación deberá ser superior al 90%.

**RNF02**. El clasificador identificará la clase de la imagen de entrada en un tiempo no mayor a cinco segundos.

**RNF05**. El sistema será desarrollado para un ambiente Web.

**RNF06**. El sistema podrá usarse a través del navegador Chrome v.64.

3.1.2 Requisitos de hardware.

El mayor trabajo computacional se realiza durante la etapa de entrenamiento del clasificador, donde dependiendo del cpu, gpu y de la cantidad de memoria, el proceso puede extenderse por horas, o incluso días. Sin embargo, una vez entrenado el clasificador, clasificar una imagen es una tarea muy sencilla que se lleva a cabo en cuestión de algunos segundos. Por lo anterior, las características del servidor de producción no se ven restringidas en este sentido.

La etapa de entrenamiento será realizado con el framework Caffe. Caffe es un framework para deep learning desarrollado por Berkeley Al Research (BAIR). Los requisitos del sistema para este framework realmente dependen de las acciones que se vayan a realizar. Para este proyecto se realizará un fine-tuning (red pre-entrenada), lo que reduce el cómputo. Esto se detalla en el capítulo de diseño. Por la razón anterior, una computadora con los requisitos siguientes es suficiente:

* CPU: Intel core i3.
* RAM: 4 GB.
* GPU\*: Se require CUDA para el modo GPU, versión 6 o superior.

La GPU es opcional pero su uso puede reducir el tiempo de cómputo a unas pocas horas. De la misma forma, un mejor CPU o más RAM ayudan en el mismo sentido.

3.2 Modelo de casos de uso.

Desde el punto de vista del usuario, la aplicación le permitirá hacer las siguientes acciones. Seleccionar una imagen desde su dispositivo y obtener el diagnóstico de enfermedad de la planta capturada en la imagen. Esto se ilustra en el diagrama de casos de uso de la figura 3.1.



Figura 3.1 Diagrama de casos de uso del prototipo.

3.2.1 CU1. Seleccionar imagen.

Resumen.

El usuario accederá a la interfaz del sistema y seleccionará la imagen de la hoja con la enfermedad que quiere identificar.

Descripción.

|  |  |
| --- | --- |
| Caso de uso: CU1. Seleccionar imagen | |
| Versión | 1.0 |
| Actor | Usuario. |
| Propósito | Seleccionar la imagen con la hoja enferma. |
| Entradas | Ninguna. |
| Origen | IU1.0 Pantalla de inicio. |
| Salidas | Imagen seleccionada |
| Destino | IU1.0 Pantalla de inicio. |
| Precondiciones | Dar click en el botón “Seleccionar imagen”. |
| Postcondiciones | Imagen seleccionada y lista para enviar. |
| Errores | Extensión incorrecta del archivo seleccionado. |
| Reglas de negocio | Solo son admitidas las extensiones jpg, png. |
| Tipo | Caso de uso primario. |
| Observaciones | Solo se selecciona la imagen, aún no se envía para su clasificación. |
| Autor | Edgar Rodrigo Arredondo Basurto. |
| Estatus | Versión 1.0 Revisada. |

Trayectorias del caso de uso.

Trayectoria principal.

1 Da click en el botón “Seleccionar imagen”.

2 Selecciona un archivo en la ventana de dialogo. [Trayectoria A].

- - - - - - - *Fin del caso de uso.*

Trayectoria alternativa A.

Condición: El usuario selecciono un archivo con una extensión distinta a jpg o png.

1  Muestra el mensaje “Seleccione un archivo válido (\*.jpg, \*.png)”.

2  Termina el caso de uso.

- - - - - - - *Fin de la trayectoria.*

3.2.1 CU2. Obtener diagnóstico.

Resumen.

El usuario enviará la imagen preseleccionada y obtendrá como resultado la clasificación de la misma en alguna de las nueve categorías mostradas en la tabla 3.1.

Descripción.

|  |  |
| --- | --- |
| Caso de uso: CU2. Obtener diagnostico | |
| Versión | 1.0 |
| Actor | Usuario. |
| Propósito | Obtener la clasificación de la imagen con la hoja enferma. |
| Entradas | La imagen a clasificar. |
| Origen | IU1.0 Pantalla de inicio. |
| Salidas | Diagnóstico y su probabilidad. |
| Destino | IU1.0 Pantalla de inicio. |
| Precondiciones | Dar click en el botón “Enviar imagen”. |
| Postcondiciones | Ninguna. |
| Errores | Ninguno. |
| Reglas de negocio | Solo se puede clasificar en alguna de las nueve categorías listadas en la tabla 3.1. |
| Tipo | Caso de uso primario. |
| Observaciones | Ninguna. |
| Autor | Edgar Rodrigo Arredondo Basurto. |
| Estatus | Versión 1.0 Revisada. |

Trayectorias del caso de uso.

Trayectoria principal.

1 Da click en el botón “Enviar imagen”.

2 Muestra el resultado de la clasificación y su probabilidad.

- - - - - - - *Fin del caso de uso.*

3.3 Interfaz gráfica.

En el sistema solo es necesaria una pantalla, desde la cual el usuario selecciona la imagen la envía y recibe el diagnóstico.

3.3.1 IU1.0 Pantalla de inicio.

Objetivo.

Desde esta pantalla el usuario puede seleccionar la imagen desde su dispositivo y enviarla para obtener el diagnóstico. El resultado le será informado con un mensaje en esta misma pantalla.

Diseño.



Figura 3.2 IU1.0 Pantalla de inicio.

Capítulo 4. Diseño del sistema.

Diseño del sistema.

La extracción de descriptores y clasificación corresponden a las etapas funcionales del prototipo dos propuesto (ver figura 4.1), el cual tiene por objetivo clasificar los objetos segmentados del prototipo uno en clases que permitan identificar la deficiencia de nutriente en la planta.

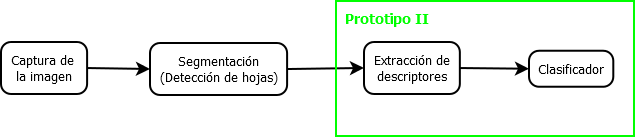


Figura 4.1 Etapas del segundo prototipo.

Este prototipo ayuda a diagnosticar deficiencias de un subconjunto definido de nutrientes, siendo estos potasio, nitrógeno, azufre y magnesio, ya que son las deficiencias que presentan síntomas diferenciables a partir del análisis visual. El resto de deficiencias de nutrientes suele ser difícil de diagnosticar mediante técnicas visuales debido a la gran similitud de sus síntomas. Incluso en el caso del nitrógeno y el azufre los síntomas visuales, son bastante similares. La forma de diferenciarlos es considerando el tipo de hoja que presenta la deficiencia, es decir, si es una hoja joven (de la parte alta de la planta) o es una hoja vieja (de la parte inferior de la planta). En este último caso, el usuario deberá complementar el diagnóstico indicando el tipo de hoja que presenta los síntomas.

Es así como las categorías o clases posibles de clasificación son hoja sana, hoja con deficiencia de potasio, hoja con deficiencia de magnesio, hoja con deficiencia de nitrógeno/azufre y hoja con una deficiencia de nutriente no identificada. En el caso de que una hoja sea clasificada en la clase nitrógeno/azufre, para discernir entre ambos nutrientes, el resultado de la clasificación será complementado con el dato sobre el tipo de hoja (hoja joven u hoja vieja).

Para poder realizar la etapa de clasificación, es necesario que una vez segmentada la hoja, se extraigan ciertas características de la hoja que permitan clasificarla en alguna de las categorías mencionadas.

4.1 Selección de descriptores

Un descriptor es un valor numérico que representa una característica particular del objeto, y que ayuda a diferenciar un objeto de otro. En este proyecto se requiere uno o más descriptores que permitan diferenciar una hoja sana de una hoja con alguna de las deficiencias del subconjunto de nutrientes seleccionado.

Los descriptores propuestos en este proyecto para diferenciar entre objetos son los descriptores definidos por Haralick [9]. Estos descriptores son un conjunto de medidas de textura basadas en la matriz de co-ocurrencia Tienen su fundamento en las relaciones espaciales que se dan entre los distintos niveles de gris de un objeto en una imagen. Las catorce medidas estadísticas que Haralick describió son:

1. Segundo momento angular
2. Contraste
3. Correlación
4. Varianza
5. Momento de diferencias inverso
6. Suma de promedios
7. Suma de varianzas
8. Suma de entropías
9. Entropía
10. Diferencia de varianzas
11. Diferencia de entropías
12. Medida de correlación 1
13. Medida de correlación 2
14. Máximo coeficiente de correlación

Sin embargo, dependiendo de la aplicación, las catorce medidas no son absolutamente necesarias, sino únicamente aquellas que aporten un carácter discriminador entre los objetos. De hecho, las tres más ampliamente utilizadas son el segundo momento angular, el contraste y la correlación.

Con el fin de identificar los descriptores de Haralick que ofrecen mayor capacidad de discriminante entre las deficiencias de nutrientes, se realizará un análisis estadístico de los descriptores mediante la técnica de la matriz de correlación. De esta forma se conocerán los descriptores que ayudan a diferenciar entre deficiencias de nutrientes.

4.2 Clasificador

Una vez definido el descriptor y conociendo la forma de calcularlo, puede usarse para clasificar el objeto en las categorías predefinidas. El algoritmo que realiza tal tarea es llamado clasificador.

Antes de usar un clasificador es necesario contar con un conjunto predefinido de objetos de cada clase con su respectivo descriptor. De esta forma, mediante alguna técnica el clasificador decide cual es la clase a la que pertenece el objeto a clasificar. Los siguientes son ejemplos de clasificadores y se incluye una pequeña descripción de su método de decisión.

1. Clasificador de la distancia mínima. Calcula el centroide de los representantes de cada clase, posteriormente calcula la distancia del objeto a clasificar al centroide de cada clase y lo clasifica en aquella clase con la distancia mínima.

2. Clasificador de la máxima probabilidad. Emplea una función de densidad de probabilidad para calcular la probabilidad de que el objeto pertenezca a cada clase y lo clasifica en aquella con la máxima probabilidad. Requiere el cálculo de parámetros como la media y la varianza de cada clase.

3. Clasificador knn. Considera los k-vecinos más cercanos al objeto. Aquella clase con más representantes dentro de esos k-vecinos es a la que es asignado el objeto a clasificar.

La elección del clasificador depende de la aplicación. La forma en que se tomará esta decisión será midiendo la eficiencia de cada uno de ellos. Este procedimiento consiste en considerar como el objeto a clasificar a cada uno de los representantes de las clases y verificando si el clasificador lo asigna a la clase a la que en realidad pertenece. Un clasificador con 100% de eficiencia es aquel que clasifique de forma correcta todos los representantes en sus clases correspondientes. El clasificador con mayor eficiencia será el que será usado en la aplicación final.

Capítulo 5. Implementación.

Implementación.

El prototipo tres es el último de este proyecto, por lo cual su objetivo es presentar la funcionalidad de los dos prototipos anteriores en un contexto que resulte adecuado para el usuario (ver figura 5.1).

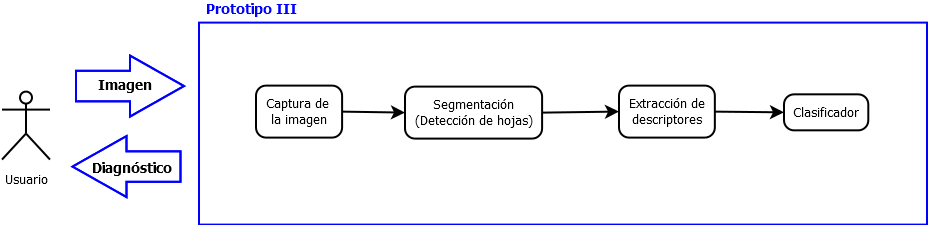


Figura 5.1 Prototipo tres.

5.1 Lenguaje de programación

La herramienta de apoyo para la implementación del sistema de visión por computadora, incluyendo los algoritmos antes mencionados, es la biblioteca de OpenCV. . Sin embargo, el sistema de visión por computadora debe complementarse con una interfaz para el usuario en la que pueda realizar pruebas con el prototipo.

OpenCV ofrece soporte para Java, Python y C++. En aplicaciones de alto desempeño, como puede ser este caso, no hay duda que C++ es la mejor opción. Sin embargo, en cuanto a facilidad de implementación C++ resulta frecuentemente más complicado.

Para decidir en qué lenguaje de programación se va a desarrollar la aplicación es necesario considerar el tiempo de respuesta de las implementaciones de los algoritmos de OpenCV en los tres lenguajes de programación mencionados. Con estos datos puede analizarse la diferencia de desempeño y considerando un tiempo de respuesta total razonable, decidir si es factible una implementación en Java o Python, o en cambio, debe realizarse en C++.

OpenCV cuenta con una implementación de un clasificador SVM que es entrenado a partir de descriptores HOG. A su vez también cuenta con los descriptores de cientos de rostros concentrados en una archivo xml. El tiempo de ejecución de esta implementación resulta un parámetro útil para la selección del lenguaje de programación, ya que el algoritmo y el descriptor son los mismo que van a emplearse en la detección de hojas, la diferencia radica en los datos numéricos de entrada, y en consecuencia, el tipo de objeto detectado. En la tabla 5.1 se muestra una comparativa del tiempo promedio de ejecución del programa de identificación de personas ya implementado en OpenCV en los tres lenguajes de programación mencionados. Las tres implementaciones fueron ejecutadas en la misma computadora con la misma imagen.

|  |  |  |  |
| --- | --- | --- | --- |
| Lenguaje de programación | Java | Python | C++ |
| Tiempo de respuesta [ms] | 188.8 | 195.58 | 112.48 |

Tabla 5.1 Comparativa en el desempeño de un mismo detector de objetos implementado en diferentes lenguajes de programación.

De la tabla 5.1 se puede concluir que la implementación en C++ es la que ofrece un mejor desempeño y en comparación con ella, las implementaciones de java y python tardan un setenta por ciento más aproximadamente.

De esta forma, se selecciona C++ como lenguaje de desarrollo, buscando el mejor rendimiento posible de la aplicación y teniendo en cuenta la nueva funcionalidad que pueda introducirse al sistema y que pueda impactar en el rendimiento.

5.2 Arquitectura del sistema

Para la implementación y presentación del prototipo propuesto se tienen las siguientes alternativas.

* Aplicación de escritorio.
* Aplicación móvil.
* Aplicación web.

La tendencia actual va hacía abandonar las aplicaciones de escritorio, ya que resultan más accesibles a los usuarios las aplicaciones web o móviles. La selección entre una aplicación web o móvil, entre otros factores, está determinada por los recursos que el dispositivo necesite para ejecutar la aplicación correctamente; un dispositivo móvil generalmente tiene recursos más limitados que un servidor web.

Únicamente la funcionalidad propuesta en este proyecto puede considerarse ser implementada dentro de una aplicación móvil; sin embargo, tendiendo en mente que el diseño debe estar enfocado para agregar más funcionalidad con el objetivo final de contar con un sistema robusto que no tenga restricciones de especies de plantas o de deficiencias de nutrientes a identificar, resulta más conveniente implementar el sistema de visión por computadora en un servidor web. De esta forma no se excluye a los dispositivos móviles, ya que pueden acceder a la interfaz web desde su navegador, o incluso, en un trabajo a futuro, desarrollar una aplicación móvil la cual se conecte al servidor web para obtener los diagnósticos.

Es así como la presentación de este prototipo será un servidor que se instalará en una máquina en una red lan, de forma tal que equipos dentro de la red puedan conectarse al servidor a través de una interfaz web. A través de está interfaz el usuario puede enviar la imagen de la planta enferma y obtener el diagnóstico correspondiente. Lo anterior se muestra de forma esquemática en la figura 5.2. Las peticiones entre cliente y servidor serán peticiones Ajax.

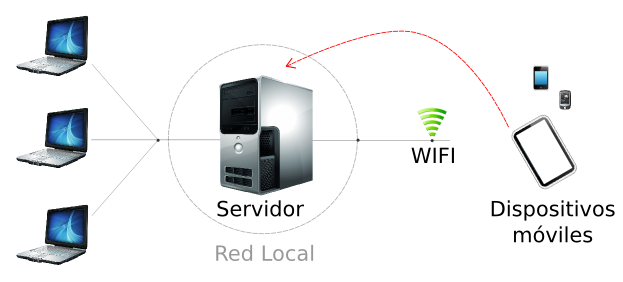


Figura 5.1 Arquitectura del sistema.

5.3 Diseño de la aplicación.

Desde el punto de vista del usuario, la aplicación le permitirá hacer las siguientes acciones. Seleccionar una imagen desde su dispositivo y obtener el diagnóstico de enfermedad de la planta capturada en la imagen. Esto se ilustra en el diagrama de casos de uso de la figura 5.2.



Figura 5.2 Casos de uso.

Las acciones previas al inicio de la obtención del diagnóstico las realiza el usuario en la interfaz web. Estas acciones consisten básicamente de dos actividades, seleccionar una imagen y presionar un botón para enviar la imagen al servidor. Una vez recibida la imagen en el servidor, se inicia el proceso descrito en los dos primeros prototipos para la obtención del diagnóstico. De esta forma, en el servidor se cuenta con una clase *Controlador* que se encarga de recibir la imagen, instanciar un objeto de la clase *Clasificador* y llamar los métodos necesarios de dicha clase para obtener el diagnóstico. Una vez obtenido el resultado se envía al cliente y se muestra en la interfaz web. En la figura 5.3 se muestra la interacción entre las clases *Controlador* y *Clasificador*, y en la figura 5.4 el diseño preliminar de la interfaz web.

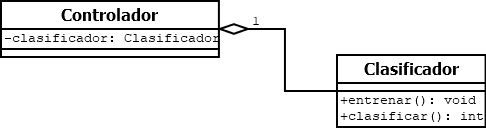


Figura 5.3 Diagrama de clases.



Figura 5.4 Propuesta de interfaz web.

De esta forma se concluye el diseño del tercer y último prototipo de la metodología de desarrollo y en una etapa posterior se complementará este documento con los detalles acerca de la implementación y puesta a punto del proyecto.

Referencias.

1. SAGARPA. Hidroponia rústica. [En línea] Disponible en: http://www.sagarpa.gob.mx.
2. Research and Markets. Global Hydroponics Market - Forecasts from 2017 to 2022. [En línea] Disponible en: https://www.researchandmarkets.com.
3. Mohanty S., Hughes D., Salathé M. Using Deep Learning for Image-Based Plant Disease Detection. 2016.
4. Moujahid A. A practical Introduction to Deep Learning with Caffe and Python. [En línea] Disponible en http://adilmoujahid.com
5. CS231n. Convolutional Neural Networks for Visual Recognition. [En linea] Disponible en http://cs231n.github.io/convolutional-networks/
6. Szegedy et al. Going Deeper with Convolutions. [En línea] Disponible en: https://arxiv.org/abs/1409.4842
7. Mohamed Brahimi. Computer Science Department. Mohamed El Bachir El Ibrahimi University.
8. Rosenbrock A. Histogram of Oriented Gradients and Object Detection. Pyimagesearch. [En línea] Disponible en http://www.pyimagesearch.com/
9. [Daren Mueller, Iowa State University, Bugwood.org](https://www.ipmimages.org/browse/detail.cfm?imgnum=5465949#collapseseven)
10. Haralick, R. M., Shanmugan, K., and Dinstein, I. Textural features for image classification. 1973.