Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Redes neuronales adversarias para el reconocimiento de malezas

Tesina presentada por Baruffaldi Juan Manuel
para obtener el grado de Licenciado en Ciencias de la Computación
Dirigida por Uzal Lucas

27/02/2018

Resumen

Se aborda el problema de reconocimiento de malezas en video para poder realizar una aplicación de herbicidas selectiva de la maleza sobre campo con cultivo de soja. El sistema de reconocimiento propuesto es compatible además con la implementación de técnicas de robótica para remover la maleza con actuadores mecánicos sin el uso de agroquímicos. El problema es abordado con técnicas de Deep Learning, donde los datos de entrenamiento son filmaciones del campo con la presencia de cultivo y maleza. El sistema de visión propuesto está basado en *Convolutional Neural Networks* (CNN). Se utilizó la técnica de *Generative Adversarial Networks* (GAN) para hacer un pre-entrenamiento no supervisado del modelo de modo de explotar la gran cantidad de imágenes que se obtienen a partir de secuencias de video. Luego se entrena en forma supervisada con una mínima cantidad de datos etiquetados para especializar el modelo. Se analizan y comparan resultados de distintos métodos utilizados y su aporte en el reconocimiento. Se combinan dos redes discriminantes de *Deep Convolutional Generative Adversarial Networks* (DCGAN) y se utiliza una *Support Vector Machines* (SVM) en la última capa de la red entrenada sobre datos etiquetados utilizando *Data Augmentation* para lograr mayor robustez. Como resultado final se obtuvo un precisión sobre el conjunto de test de 91 %.

Agradecimientos

Esta tesina representa el trabajo y acompañamiento de mucha gente a lo largo de estos años. Una de esas personas es Lucas, que como director y mentor en esta etapa, ha sido un gran maestro. Así también agradezco a todo el grupo de investigación de CIFASIS-CONICET por la predisposición y buena voluntad durante todo el proceso. Una parte fundamental han sido los docentes y autoridades de la carrera que han hecho una gran familia LCCiana. Gracias a todos mis compañeros de estudio por compartir este viaje. Agradezco enormemente a mi familia, tanto mis viejos como excelentes ejemplos de vida, mi hermano y compañero de aventura; como así también mis tíos que me alojaron humildemente en su hogar y mis primos que siempre estuvieron presentes. Y a toda mi familia que siempre de alguna forma a ha estado acompañando. A mi amor Rocío que me ha brindado el apoyo y contención durante toda la carrera. Gracias a mis amigos de toda la vida y compañeros de distintos espacios que me han apoyado incondicionalmente.

"Si he logrado ver más lejos, ha sido porque he subido a hombros de gigantes..."
Isaac Newton

"Las redes GAN son la idea más cool de los últimos 20 años del Aprendizaje Profundo..."
Yann LeCun, AI research at Facebook

Índice general

Resumen **2**

Agradecimientos **3**

1. Introducción **7**

1.1. Problemática 7
1.2. Antecedentes 9
1.3. Objetivos 13
1.4. Desarrollo de software 13
1.5. Organización de la tesina 14

2. Marco teórico **15**

2.1. Aprendizaje automático 15
2.2. Aprendizaje supervisado, no supervisado y semi supervisado . . . 16
2.3. Redes neuronales artificiales 17
2.4. Redes neuronales convolucionales 21
2.5. Redes adversarias generativas 24
2.5.1. Entrenamiento adversario 26
2.6. Redes convolucionales profundas generativas adversarias 29
2.7. Máquinas de vectores soporte 31

3. Metodología **33**

3.1. Conjunto de datos 34
3.2. Métodos 36
3.3. Aumentación de datos 46
3.4. Herramientas de trabajo 47

4. Resultados **48**
4.1. Preliminares 48
4.2. DCGAN 51
4.3. DCGANCLASS 53
4.4. Evaluación de rendimientos 54
4.5. Reconocimientos 56

5. Conclusiones y trabajo a futuro **62**
5.1. Conclusión 62
5.2. Trabajo futuro 63

A. Ejemplos **65**

Capítulo 1

Introducción

En este trabajo de tesina se aborda el problema de reconocimiento de malezas en video utilizando técnicas de Deep Learning. Esto se realiza en el contexto de campo sembrado con soja y presencia de malezas en la misma altura de crecimiento, con la finalidad de distinguir la maleza del cultivo para poder eliminarla. Dentro de la rama de Deep Learning se utilizaron Redes Neuronales Convolucionales (Convolutional Neural Networks), y en particular Redes Adversarias Generativas (Generative Adversarial Networks) para hacer un pre-entrenamiento no supervisado del modelo. Luego se entrena en forma supervisada con una mínima cantidad de datos etiquetados para especializar el modelo en el problema de detección de malezas entre cultivo de soja. Los objetivos de esta tesina apuntan a reducir la utilización de los mismos y reemplazarlos por métodos menos invasivos.

1.1. Problemática

El uso constante y creciente de agroquímicos para controlar malezas en los cultivos genera un círculo vicioso: a mayores dosis de herbicidas mayor resistencia se desarrolla al producto [13]. En nuestro país se aplican agroquímicos tradicionalmente de manera uniforme en la totalidad del lote, independientemente de la existencia o no de malezas, por lo que su uso es cada vez más ineficiente, costoso y perjudicial para el medio-ambiente y la salud humana [13].

Según datos recientes de la Cámara de Sanidad Agropecuaria y Fertilizantes (CASAFE) [58], en los últimos 22 años el consumo de herbicidas aumentó un

858 %, mientras que la superficie cultivada se incrementó en un 50 % y el rendimiento de los cultivos, sólo un 30 % [13]. Esto muestra una necesidad de encontrar una solución al incremento del uso de agroquímicos con el fin de reducir costos y cuidar el medio ambiente.

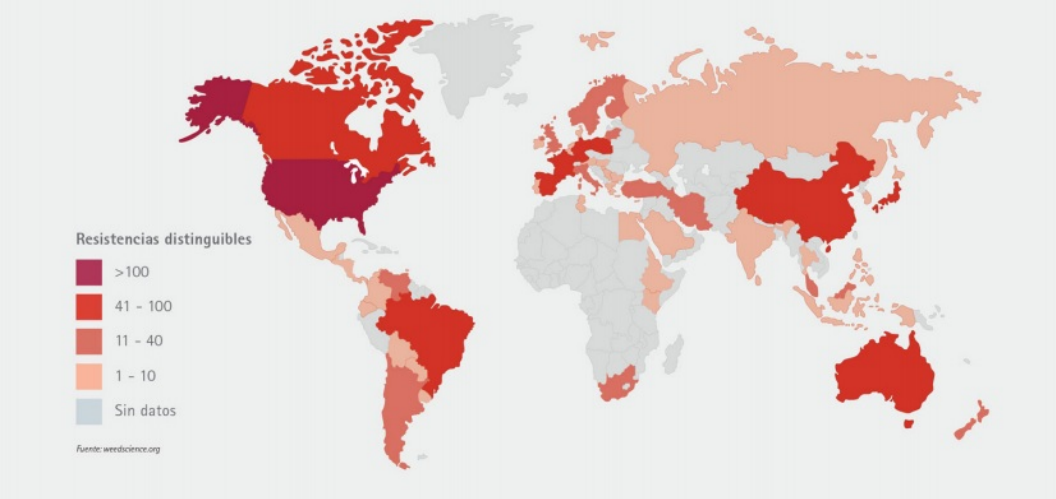

Figura 1.1: Mapa que visualiza la problemática de malezas resistentes a nivel mundial [42], evaluadas en cantidad de poblaciones de malezas resistentes distinguibles.

El uso indiscriminado de herbicidas ha logrado que las malezas sean cada vez más resistentes, teniendo que aumentar la dosis para lograr eliminarlas (En la Figura 1.1 se muestra cómo este problema se presenta a nivel global). Un estudio realizado por docentes de la Facultad de Agronomía de la Universidad de Buenos Aires (FAUBA), estima que por año se destinan unos 1300 millones de dólares para combatir las malezas que afectan a la zona núcleo [18]. Podemos observar la evolución de este dato en la Figura 1.2.

El mercado actual necesita poder realizar una pulverización inteligente, diferenciando la maleza del cultivo en las distintas etapas de crecimiento y a una velocidad de respuesta aceptable dada la velocidad de trabajo del pulverizador. Los beneficios radican en el claro ahorro del uso del agua, cuidado del medio ambiente y reducción de costos para el productor agropecuario.

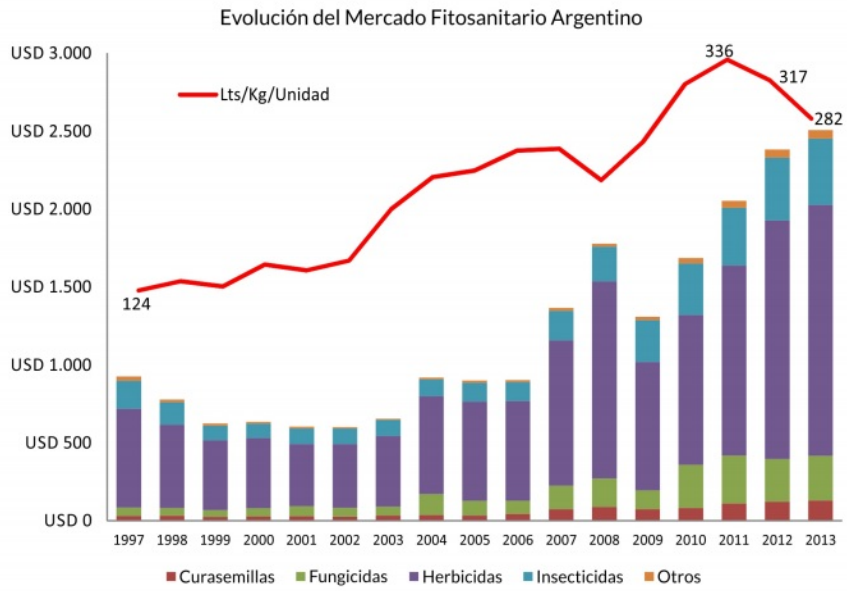

Figura 1.2: Se observa la evolución de los gastos en agroquímicos comparada con la cantidad de litros/Kg (línea roja) por hectáreas aplicados en los campos, según Mercado Argentino de Fitosanitarios [43]

1.2. Antecedentes

Dentro del Grupo de Aprendizaje Automático y Aplicaciones del CIFASIS existen antecedentes de la aplicación de Deep Learning al reconocimiento de especies vegetales. Se ha trabajado en el diseño de un método automático para la clasificación de especies y variedades vegetales agrícolas en Argentina, analizando exclusivamente la información contenida en imágenes de las nervaduras de las hojas. Los primeros resultados se obtuvieron mediante técnicas tradicionales de Visión por Computador (Computer Vision) y Aprendizaje Automático (Machine Learning) [23–25]. Este problema se abordó luego con Aprendizaje Profundo (Deep Learning), mejorando los resultados previamente obtenidos [26]. Sin embargo estas técnicas no son tan efectivas para la clasificación masiva de especies vegetales en tiempo real ya que requieren una adquisición cuidadosa por medio de scanners de la imagen de la hoja.

Investigadores del Instituto Nacional de Tecnología Agropecuaria (INTA) de Castelar, desarrollaron un detector de malezas mediante sensores ópticos para la aplicación precisa de los herbicidas. Utilizaron 2 sensores, uno que detecta la pre-

sencia de color verde y otro por reflejo de luz infrarroja. Con este último se han logrado reducciones en el uso de herbicidas de entre el 30 y el 70 % con niveles de precisión en la detección del 95 %, dependiendo esto último del tipo cultivo y de la maleza [15]. La desventaja de estos métodos es la de no poder distinguir la maleza del cultivo. Sin embargo, permiten realizar reducciones importantes en los volúmenes de herbicidas aplicados al control de malezas cuando se realiza la práctica del barbecho químico (es el período entre el primer tratamiento herbicida que se realiza después de la cosecha y la siembra directa del cultivo siguiente)[14].

Las soluciones existentes en el mercado, en su mayoría, radican en la detección del pigmento verde o la cantidad de luz por medios de sensores ópticos. Estos detectores (técnicamente llamados weedsekers) funcionan en tiempo real, instalado en un equipo pulverizador autopropulsado o de arrastre. Entran en acción en la instancia de barbecho químico, previa a la siembra del cultivo, cuando se controlan las malezas con agroquímicos pre-emergentes. No está indicado para utilizar con los cultivos, ya que no diferencia las malezas del cultivo en cuestión. Por su configuración modular, también puede servir para aplicaciones en los entre-surcos de plantaciones de maíz y de caña de azúcar [14, 16].

Se definen a continuación las 2 técnicas más relevantes para abordar el problema de reconocimiento de malezas en el campo:

Detección de color: Una aproximación muy sencilla para lograr una pulverización selectiva radica en detectar la presencia del color verde. Con esta simple operación estaríamos en condiciones de distinguir las malezas verdes del suelo y, de esta manera, poder pulverizar solamente los sectores que poseen malezas. Este método resultaría muy beneficioso para aquellos campos que implementan la siembra directa, minimizando el volumen de herbicidas aplicado al control de malezas en el barbecho. Para la implementación de este método se debe colocar un sensor que realice una exploración del suelo con el objetivo de detectar las malezas antes que las mismas lleguen a la pastilla pulverizadora. Una unidad central de proceso (CPU) se encarga de procesar los datos enviados por el sensor y de determinar la existencia o no de malezas, actuando sobre una válvula electrónica que restringe el paso el herbicida a la pastilla pulverizadora (ver figura 1.3). Con el método de detección de color se consigue una clara distinción entre la maleza

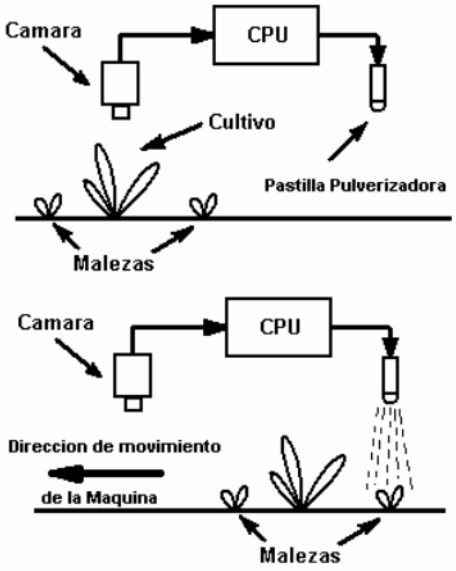

Figura 1.3: Esquema de un sistema de pulverización selectiva por sensores ópticos [14]

y el suelo. Sin embargo, por las características propias del sensor no es posible distinguir entre el cultivo y la maleza. Esto último sería realmente útil en la aplicación de los herbicidas post-emergentes. Si bien actualmente esta selectividad se logra con el uso de variedades de cultivos resistentes al herbicida, se está realizando un uso ineficiente del mismo al pulverizar todo el lote para tratar las malezas que no se encuentran distribuidas uniformemente [14].

Existen varios trabajos enfocados en la detección de malezas mediante la cantidad de luz reflejada por las mismas al ser iluminadas. Fundamentalmente se han realizado estudios para determinar la cantidad de luz reflejada cerca del infrarrojo (NIR) y en el rango del rojo (RR) [17]. Dado que el suelo y las plantas reflejan cantidades distintas de ambos espectros, es posible utilizar esta característica para realizar una clasificación. Las plantas debido a su clorofila, absorben gran cantidad de luz roja, en el rango de longitudes de onda que van de 630 a 660 nm (RR), mientras que la luz cercana al infrarrojo, que se encuentra entre 750 y 1200 nm (NIR) es ampliamente reflejada. Según trabajos realizados, la relación de NIR/RR se encuentra entre 1.1 y 1.5 para el suelo y entre 6 y 15 para las plantas con hojas verdes, dependiendo del tipo de suelo y de maleza [15]. En lo que al procedimiento respecta, la idea es similar al caso anterior. Lo único que se

modifica es la forma de detectar las malezas, o sea el sensor. En este caso está conformado por dos fotodiodos, uno dedicado a detectar la cantidad de luz roja reflejada por la planta y otro dedicado a detectar la cantidad de luz cercana al infrarrojo reflejada. Esto es posible mediante el uso de filtros ópticos de la longitud de onda deseada (RR o NIR) para cada sensor. El procesamiento de los datos tomados por el sensor y la actuación sobre la electroválvula, también puede ser llevado a cabo por un microcontrolador como se especificó en el método anterior, debido a que el algoritmo de discriminación a aplicar tampoco posee una gran complejidad. Las conclusiones para este método son idénticas a las del método anterior, haciendo hincapié en su simple diseño y la desventaja de no poder distinguir la maleza del cultivo [14].

Detección mediante visión artificial: Los sistemas de visión artificial están basados en el uso de cámaras de video o fotográficas. Dentro de éstos podemos distinguir básicamente dos posibilidades: Procesamiento en tiempo real y offline. En el caso offline consistiría en la toma de fotografías del campo georefernciadas antes de realizar la pulverización, para luego tratar las imágenes mediante el algoritmo de detección deseado con una PC de escritorio. Una vez que se obtiene un mapa de la ubicación de las malezas, estos datos son cargados en la pulverizadora que cuenta con un GPS. De esta manera es posible lograr la aplicación solo en aquellos sitios indicados en el mapa como afectados por las malezas. Una de las ventajas de este método es que el procesamiento grueso de información no se realiza en tiempo real, pudiéndose hacer uso de una PC de escritorio. El otro caso es la adquisición y procesamiento de las imágenes en tiempo real. Esto permite tener una pulverizadora selectiva completamente independiente y automatizada. Trabajar en tiempo real implica tomar las imágenes, procesarlas, detectar las malezas y actuar consecuentemente mientras el equipo de pulverización se encuentra en movimiento. Estos equipos cuentan con el uso de cámaras digitales (CCD) y de una o varias CPU capaces de ejecutar los algoritmos de detección para lograr una actuación en el momento que se precisa [14].

1.3. Objetivos

En este proyecto se intentará resolver el problema utilizando técnicas de Machine Learning identificando las malezas por métodos de visión artificial. Dadas las características del hardware actual, utilizando software embebidos con GPUs, se logra con modelos de redes neuronales obtener tiempos de respuestas muy eficaces; y da lugar a desarrollar el reconocedor en tiempo real solucionando el problema de la velocidad y de la identificación de malezas luego del nacimiento del cultivo.

Se trabajará como proyecto de tesina y se utilizará el trabajo iniciado por Martín Escarrá en el grupo de investigación de CIFASIS-CONICET Rosario. Este fue un trabajo preliminar basado en registros fílmicos donde el cultivo estaba en su máximo crecimiento, que eran los que se disponían al momento pero que no coinciden con la instancia en la que se desea actuar sobre la maleza. En nuestro caso se realizará la detección en crecimientos iguales tanto de la maleza como del cultivo; donde el problema es mucho más interesante.

Esta tesina se encuentra en el marco del desarrollo de un robot desmalezador por parte del CIFASIS-CONICET y existen líneas de investigación que continuaran con el proyecto luego de finalizada la misma.

El objetivo del trabajo es desarrollar un prototipo escalable del problema, obteniendo resultados de reconocimientos aceptables. Se busca lograr la reducción del uso de herbicidas respecto de su aplicación sobre la totalidad del lote con el fin de reducir el impacto ambiental.

1.4. Desarrollo de software

La tecnología que se utilizó son redes neuronales profundas con aprendizaje semi supervisado utilizando bases de datos obtenidas de forma artesanal. El lenguaje de programación fue Python utilizando el entorno Theano [64] para el manejo de los algoritmos de Deep Learning y una librería de abstracción Lasagne [63] que facilita la creación y mantención de las redes utilizadas.

El entorno de desarrollo que se utilizó fue Linux y todas las herramientas de programación y desarrollo de software que este Sistema Operativo ofrece. Además, se usaron los recursos de hardware que la Facultad de Ciencias Exactas, Ingeniería y Agrimensura (perteneciente a la Universidad Nacional de Rosario) pone a disposición de sus alumnos. Particularmente para el proyecto fue beneficioso poder hacer uso de una placa gráfica de última generación, modelo NVIDIA K40, ya que permitió hacer análisis en video en tiempo real.

1.5. Organización de la tesina

Esta tesina está organizada de la siguiente manera:

- Capítulo 1: Introducción a la problemática abordada para entender la relevancia del trabajo, antecedentes del mismo y objetivos de la tesina.
- Capítulo 2: Explica los conceptos utilizados a lo largo del trabajo propios del Aprendizaje Automático.
- Capítulo 3: Metodología del trabajo, cómo se desarrolló la solución y las dificultades encontradas en el camino.
- Capítulo 4: Se muestran los resultados obtenidos con los distintos métodos trabajados.
- Capítulo 5: Conclusiones obtenidas y trabajo a futuro.
- Apéndice: Se utiliza este capítulo para mostrar imágenes de ejemplos, otros resultados y contenido adicional al trabajo de tesina.

Parte de los resultados obtenidos para esta tesina fueron presentados en las 46º Jornadas Argentinas de Informática (JAIIO[46]), simposio estudiantil EST2017 (20º Concurso de Trabajos Estudiantiles), donde se obtuvo el primer premio entre los trabajos presentados en la categoría “Trabajos Finales de Carreras de Grado”.

Capítulo 2

Marco teórico

El trabajo se enmarca en el área de Machine Learning [1] y, en particular, dentro de redes neuronales convolucionales [38] y sus distintas formas de aprendizaje (supervisado, no supervisado y semi supervisado [27–29]). Utilizaremos el reciente algoritmo de redes adversarias generativas [8, 10, 11] por sus interesantes resultados en problemas no supervisados. Este algoritmo hace uso de redes neuronales profundas [7] que introduciremos en esta sección.

2.1. Aprendizaje automático

El aprendizaje automático es una consecuencia natural de la intersección de la informática y la estadística. Se centra en la cuestión de cómo hacer que las programas evolucionen por si mismos, a partir de la experiencia de los datos. Es un subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender.

Una de las definiciones más citadas es la de Tom M. Mitchell: *"Se dice que un programa aprende de la experiencia E con respecto a cierto tipo de tarea T y una medida de performance P, si el desempeño en tareas de T según la medida P, mejora con la experiencia E"*[1]. Otro autor conocido es Peter Flach que define al Aprendizaje Automático como: *"El estudio sistemático de algoritmos y sistemas que mejoran su conocimiento o desempeño con la experiencia"* y también agrega *"El aprendizaje automático puede ser visto como un intento de automatizar*

algunas partes del método científico mediante métodos matemáticos"[2].

2.2. Aprendizaje supervisado, no supervisado y semi supervisado

La forma más común de aprendizaje automático, profundo o no, es el aprendizaje supervisado. Imagínese que queremos construir un sistema que pueda clasificar las imágenes según su contenido, por ejemplo una casa, un automóvil, una persona o una mascota. Primero recopilamos un gran conjunto de datos de imágenes de casas, autos, personas y mascotas, cada uno etiquetado con su categoría. Durante el entrenamiento, la máquina muestra una imagen y produce una salida en forma de un vector de puntajes, uno para cada categoría. Queremos que la categoría deseada tenga la puntuación más alta de todas las categorías, pero es poco probable que suceda antes del entrenamiento [21].

Se calcula una función objetivo que mide el error (o distancia) entre las puntuaciones de salida y el patrón de puntuaciones deseado. La máquina modifica entonces sus parámetros ajustables internos para reducir este error. Estos parámetros ajustables, a menudo llamados pesos, son números reales que pueden ser vistos como "perillas" que definen la función de entrada-salida de la máquina. En un sistema de aprendizaje profundo típico, puede haber millones de estos pesos ajustables, y millones de ejemplos etiquetados con los cuales entrenar la máquina.

Por su parte el aprendizaje no supervisado (aprender de datos no etiquetados) tuvo un efecto catalítico al revivir el interés por el aprendizaje profundo, pero desde entonces ha sido ensombrecido por los éxitos del aprendizaje puramente supervisado. Se espera que el aprendizaje sin supervisión sea mucho más importante a largo plazo, al igual que la combinación de los mismos (es decir, aprendizaje semi-supervisado). El aprendizaje humano y animal es en gran medida sin supervisión: descubrimos la estructura del mundo observándola, no por el nombre de cada objeto [28].

El aprendizaje semi-supervisado es una clase de tareas y técnicas de aprendizaje supervisadas que también hacen uso de datos no etiquetados para el entrenamiento, típicamente una pequeña cantidad de datos etiquetados con una gran

cantidad de datos no etiquetados [29].

Se ha descubierto que con el uso de datos no etiquetados junto con una pequeña cantidad de datos etiquetados se puede obtener una muy buena precisión. La recopilación de datos no etiquetados es poco costosa en relación con los datos etiquetados. A menudo los datos etiquetados son escasos y los datos no etiquetados son abundantes. En tales situaciones, se puede usar el aprendizaje semi-supervisado. Además, gran parte del aprendizaje humano implica una pequeña cantidad de instrucción directa (datos etiquetados) combinada con grandes cantidades de observación (datos no etiquetados).

2.3. Redes neuronales artificiales

Las redes neuronales artificiales (ANN, por sus siglas en inglés) proporcionan un método general y práctico para el aprendizaje de funciones basadas en valores reales, discretos y vectoriales. El aprendizaje ANN es robusto a los errores en los datos de entrenamiento y se ha aplicado con éxito a problemas tales como la interpretación de escenas visuales, el reconocimiento de voz y el aprendizaje de estrategias de control de robots [1].

Según Rumelhart y McClelland [4] un modelo estándar (ver Fig. 2.1) de red neuronal artificial para la i-ésima neurona consiste en:

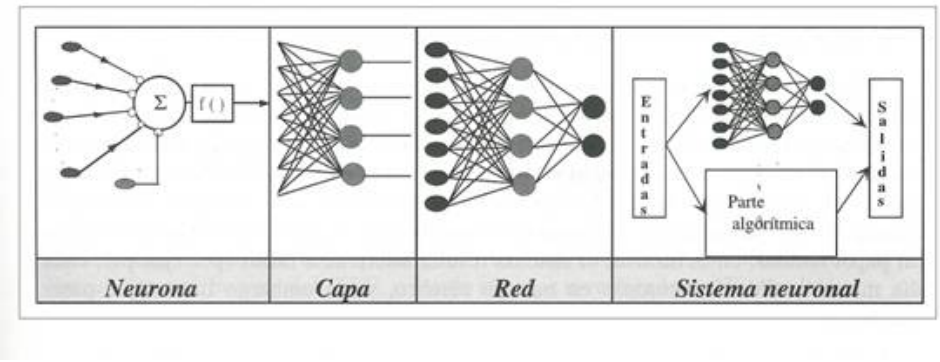

Figura 2.1: Sistema global de proceso de una red neuronal [5]

- Un *conjunto de entradas* x_j y unos pesos sinápticos w_{ij}, con $j = 1, ..., n$

- Una *regla de propagación* h_i definida a partir del conjunto de entradas y los pesos sinápticos. Es decir:

$$h_i(x_1, ..., x_n, w_{i_1}, ..., w_{i_n})$$

  La regla de propagación más comúnmente utilizada consiste en combinar linealmente las entradas y los pesos sinápticos, obteniéndose:

$$h_i(x_1, ..., x_n, w_{i_1}, ..., w_{i_n}) = \sum_{i=1}^{n} w_{ij} x_j$$

  Suele ser habitual añadir al conjunto de pesos de la neurona un parámetro adicional θ_i, que se denomina umbral, el cual se acostumbra a restar al potencial pos-sináptico. Es decir:

$$h_i(x_1, ..., x_n, w_{i_1}, ..., w_{i_n}) = \sum_{i=1}^{n} w_{ij} x_j - \theta_i$$

  Si hacemos que los índices i y j comiencen en 0, y denotamos por $w_{i0} = \theta_i$ y $x_0 = -1$, podemos expresar la regla de propagación como:

$$h_i(x_1, ..., x_n, w_{i_1}, ..., w_{i_n}) = \sum_{j=0}^{n} w_{ij} x_j = \sum_{i=1}^{n} w_{ij} x_j - \theta_i$$

- Una *función de activación*, la cual representa simultáneamente la salida de la neurona y su estado de activación. Si denotamos por y_i dicha función de activación, se tiene:

$$y_i = f_i(h_i) = f_i(\sum_{j=0}^{n} w_{ij} x_j)$$

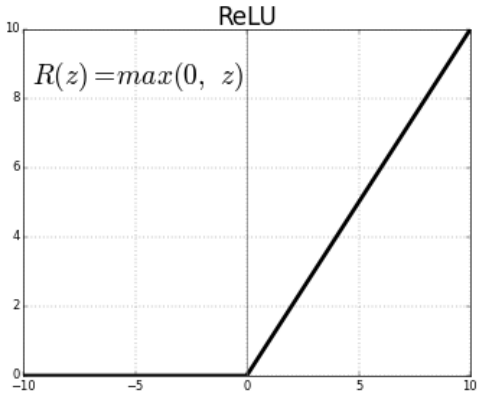

Figura 2.2: Rectified Linear Units (ReLU)[48]

Dentro de las funciones de activación más conocidas se encuentran la función *sigmoide*[49] y la función *ReLU (Rectified Linear Unit)*[48]. ReLU es una de las funciones más utilizada en Deep Learning y se define como $f(x) = \text{máx}(0, x)$, ver figura 2.2.

Para entender mejor cómo aplica la función de activación ver la Figura 2.3.

La función de activación, cualquiera sea, es no lineal lo cual es muy importante, porque si la función de activación que elegimos es lineal, la red estará limitada a resolver problemas lineales.

Una forma sencilla de implementar redes de neuronas consiste en almacenar los pesos en matrices. Es fácil ver que si ahora guardamos los valores de todas las neuronas de una capa en un vector, el producto del vector y la matriz de pesos de salida, nos da los valores de entrada de cada neurona en la siguiente capa 2.4. Ahora sólo falta aplicar la función de activación que hayamos elegido a cada elemento de ese segundo vector, y repetir el proceso. Justo antes de aplicar la función de activación, cada neurona añade a la suma de productos un nuevo término constante, llamado habitualmente *bias* [39].

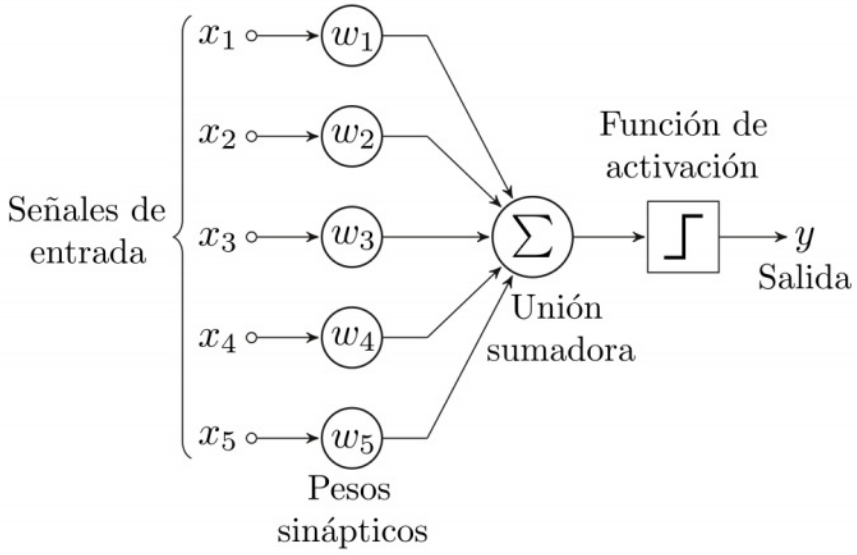

Figura 2.3: Diagrama de una neurona (también llamada perceptrón) con cinco señales de entrada [6]

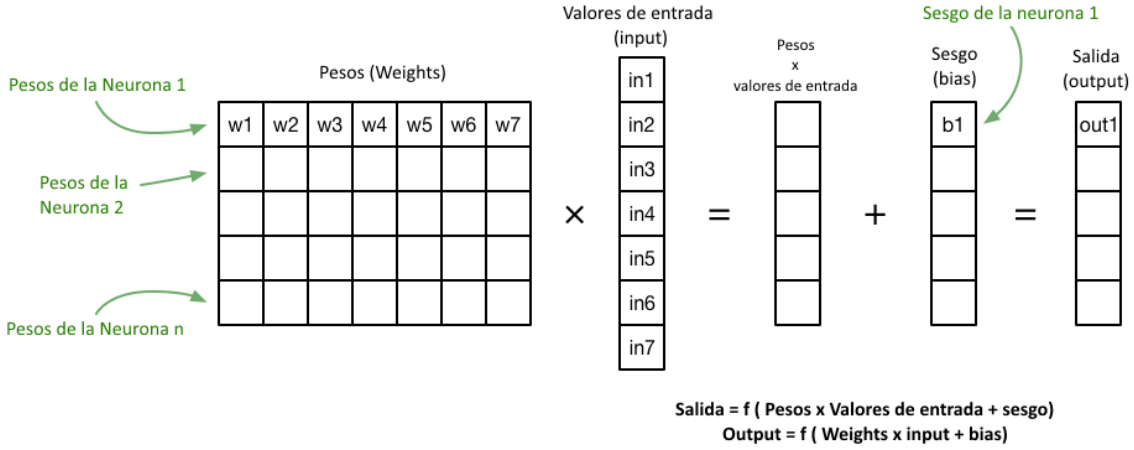

Figura 2.4: Diagrama del almacenamiento de pesos en una matriz y los productos con vectores [69].

2.4. Redes neuronales convolucionales

Las redes neuronales convolucionales (conocidas como CNN [38]) están diseñadas para procesar datos que vienen en forma de múltiples matrices, por ejemplo una imagen en color compuesta de tres matrices 2D que contienen intensidades de pixeles en los tres canales de color. Muchas modalidades de datos están en forma de múltiples matrices: 1D para señales y secuencias, incluyendo lenguaje; 2D para imágenes o espectrogramas de audio; Y 3D para video o imágenes volumétricas. Existen cuatro ideas clave detrás de CNN que aprovechan las propiedades de las señales naturales: conexiones locales, pesos compartidos, uso y agrupación de muchas capas.

La arquitectura de un CNN típica se estructura como una serie de etapas. Las primeras se componen de dos tipos de capas: capas convolucionales y capas de agrupación 2.5. Las unidades en una capa convolucional están organizadas en mapas de características, dentro de los cuales cada unidad está conectada a parches locales en los mapas de características de la capa anterior a través de un conjunto de pesos denominado banco de filtros. El resultado de esta suma ponderada local se pasa entonces a través de una no linealidad, típicamente una ReLU, para conocer más en detalle ver [7]. Todas las unidades de un mapa de funciones comparten el mismo filtro. Diferentes mapas de características en una capa utilizan filtros diferentes.

La razón de esta arquitectura es doble. En primer lugar, en los datos de la matriz, como en las imágenes, los valores cercanos suelen estar altamente correlacionados, formando características locales distintivas que se detectan fácilmente. En segundo lugar, las estadísticas locales de imágenes y otras señales son invariantes traslacionales. En otras palabras, si un patrón puede aparecer en una parte de la imagen, podría aparecer en cualquier lugar, de ahí la idea de unidades en diferentes lugares compartiendo los mismos pesos y detectando el mismo patrón en diferentes partes de la matriz. Matemáticamente, la operación de filtrado realizada por un mapa de características es una convolución discreta, de ahí el nombre (ver figura 2.6).

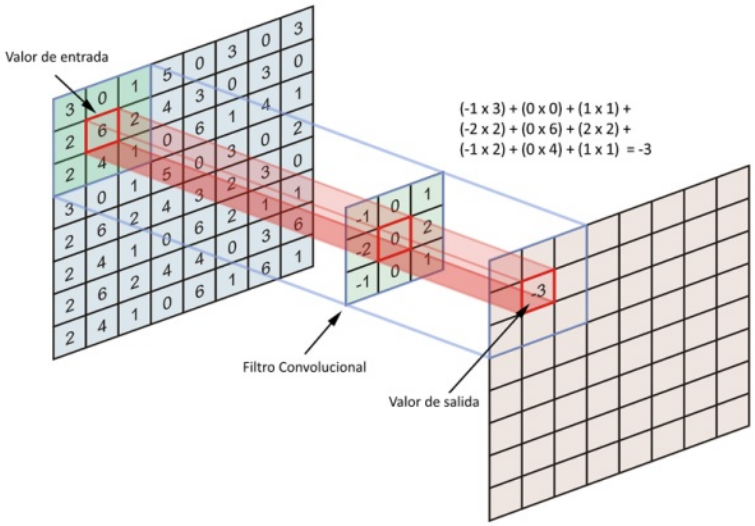

Figura 2.5: Diagrama de la capa convolucional aplicada sobre los valores de la imagen [70].

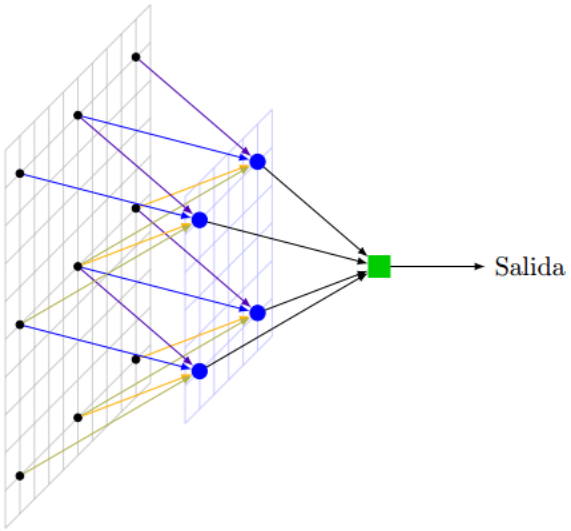

Figura 2.6: Arquitectura simplificada de una Red Convolucional [71].

Cuando a estas redes neuronales se le incrementa la cantidad de capas ocultas[1] se lo llama aprendizaje profundo (Deep Learning). La idea es que la capa más cercana a los vectores de datos aprenda características simples, mientras que las capas profundas deben aprender características de nivel más alto [32].

Una arquitectura de aprendizaje profundo es una pila multicapa de módulos simples, todos (o la mayoría) de los cuales están sujetos al aprendizaje, y muchos computan asignaciones de entrada/salida no lineales. Cada módulo en la pila transforma su entrada para aumentar tanto la selectividad como la invariancia de la representación. Con múltiples capas no lineales, por ejemplo, una profundidad de 5 a 20, un sistema puede implementar funciones extremadamente complejas que son simultáneamente sensibles a los detalles mínimos [21].

El algoritmo más conocido para actualizar los pesos de la red es *backpropagation*. El procedimiento de backpropagation se realiza para calcular el gradiente de una función objetivo con respecto a los pesos de una pila de módulos multicapa, y no es más que una aplicación práctica de la regla de la cadena para derivadas. La idea clave es que la derivada (o gradiente) del objetivo con respecto a la entrada de un módulo se puede calcular trabajando hacia atrás desde el gradiente con respecto a la salida de ese módulo (o la entrada del módulo posterior). La ecuación de backpropagation puede aplicarse repetidamente para propagar gradientes a través de todos los módulos, comenzando desde la salida en la parte superior (donde la red produce su predicción) hasta la parte inferior (donde se alimenta la entrada externa). Una vez que se han calculado estos gradientes, es sencillo calcular los gradientes con respecto a los pesos de cada módulo [21].

Cuando utilizamos una red neuronal para aceptar una entrada x y producir una salida y, la información fluye hacia adelante a través de la red. La entrada x proporciona la información inicial que luego se propaga a las unidades ocultas en cada capa y finalmente produce la salida y. Esto se llama propagación hacia adelante. Durante el entrenamiento, la propagación hacia adelante puede continuar hasta que produce un costo escalar en $J(\theta)$. El algoritmo de backpropagation, permite que la información del costo fluya hacia atrás a través de la red para calcular el gradiente.

[1]Se llama capa oculta a las capas intermedias de una red neuronal

2.5. Redes adversarias generativas

Los modelos generativos son uno de los enfoques más prometedores para trabajar en problemas donde no se poseen suficientes datos etiquetados. Para formar un modelo generativo primero recolectamos una gran cantidad de datos en algún dominio y luego formamos un modelo para generar datos sintéticos que resultan indistinguibles de los datos reales. Estos modelos, en el largo plazo, tienen el potencial de aprender automáticamente la distribución de probabilidad de un conjunto de datos [7]. Los métodos de aprendizaje profundo han presentado grandes avances en la generación de imágenes.

Las redes adversarias generativas (conocidas como GAN[8] por sus siglas en inglés) plantean el proceso de entrenamiento como un juego entre dos redes separadas: una red generadora de muestras de imágenes, y una segunda red discriminadora que trata de clasificar las muestras como procedentes de la distribución verdadera, es decir, ejemplos tomados del dataset de entrenamiento; o la distribución del modelo, es decir, ejemplos sintéticos producidos por la red generadora. Ambas redes son entrenadas a la par y cada vez que el discriminador nota una diferencia entre las dos distribuciones, el generador ajusta ligeramente sus parámetros para hacerla disminuir, hasta que al final (en teoría) el generador reproduce exactamente la distribución de datos verdadera y el discriminador está adivinando al azar, incapaz de encontrar una diferencia.

En términos matemáticos el discriminador es una función D que toma x como entrada y usa $\theta(D)$ como parámetros. El generador está definido por una función G que toma z como entrada y usa $\theta(G)$ como parámetros.

Ambos jugadores tienen funciones de costo que son definidas en términos de sus parámetros de juego. El discriminante desea minimizar una función de costo de clasificación binaria $J(D)(\theta(D), \theta(G))$ y debe hacerlo controlando solamente $\theta(D)$. De forma similar el generador desea minimizar $J(G)(\theta(D), \theta(G))$ controlando solamente $\theta(G)$. Debido a que el costo de cada jugador depende de los parámetros del otro jugador, y uno no puede controlar los parámetros del otro,

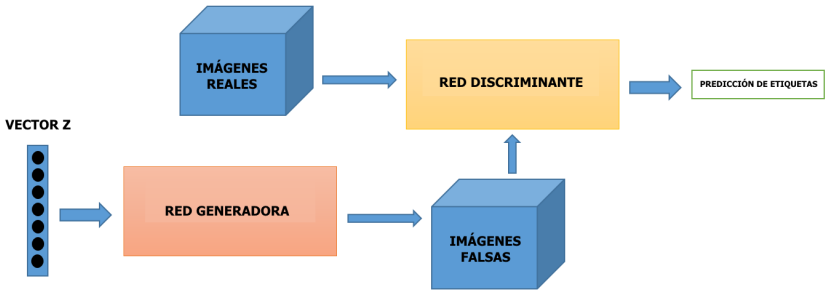

Figura 2.7: Estructura del modelo de redes adversarias [31]

este escenario es más sencillo de describir como un juego que como un problema de optimización. La solución a un problema de optimización es un mínimo (local), un punto en el espacio de parámetros donde todos los puntos vecinos tienen un costo mayor o igual. La solución a un juego es un *Equilibrio de Nash*[47]. En este contexto, un equilibrio de Nash es una tupla $(\theta(D), \theta(G))$ que es un mínimo local de $J(D)$ con respecto a $\theta(D)$ y un mínimo local de $J(G)$ con respecto a $\theta(G)$ [9].

Debido a que las redes GAN naturalmente se las enmarca con teoría de juegos, se llaman "adversarias". Pero también podemos considerarlas cooperativas, en el sentido de que el discriminador estima una proporción de densidades y luego comparte libremente esta información con el generador. Desde este punto de vista, el discriminador se parece más a un maestro, que instruye al generador sobre cómo mejorar, que a un adversario. Hasta ahora, esta visión cooperativa no ha llevado a ningún cambio particular en el desarrollo de las matemáticas [9].

La red discriminador es un clasificador que determina si una imagen de entrada se parece a una imagen real del conjunto de datos o es una imagen creada artificialmente. Esto es básicamente un clasificador binario que toma la forma de una red neuronal convolucional normal (CNN). La red generador toma valores de entrada aleatorios y los transforma en imágenes a través de una red neuronal deconvolucionales (reverso de capas convolucionales). Se puede visualizar el proceso en la figura 2.7.

El mayor problema que se enfrentan los investigadores de redes GAN es la no convergencia. La mayoría de los modelos profundos se entrena utilizando un

algoritmo de optimización que busca un valor bajo de una función de costo. Si se lo piensa en términos de teoría de juegos, si cada jugador disminuye con éxito su función de costo, la misma actualización puede aumentar la del otro jugador evitando el equilibrio. Este es un problema general con juegos no exclusivos de GAN, por lo que una solución general a este problema tendría aplicaciones de gran alcance [9].

Otro problema de las redes GAN es el colapso que se suele obtener en las imágenes generadas, también conocido como escenario de Helvetica[9]. Este es un problema que ocurre cuando el generador aprende a mapear varios valores z de entradas diferentes en el mismo punto de salida. En la práctica, el colapso completo es raro, pero el colapso parcial es común (se refiere a los escenarios en los que el generador crea múltiples imágenes que contienen el mismo color o temas de textura, o múltiples imágenes que contienen diferentes vistas de la misma imagen)[9].

2.5.1. Entrenamiento adversario

El proceso de entrenamiento consiste en un descenso de gradiente estocástico (Stochastic Gradient Descent) simultáneo [51]. En cada paso, se muestrean dos minibatches (pequeña porción de datos de entrada): un minibatch de x valores del conjunto de datos y un minibatch de valores z extraídos de las variables anteriores del modelo. Luego, se realizan dos pasos de gradiente simultáneamente: uno que actualiza $\theta(D)$ para reducir $J(D)$ y uno que actualiza $\theta(G)$ para reducir $J(G)$. En esta tesina se utilizó en ambos casos el algoritmo de optimización basado en descenso por gradiente denominado Adam [50]. Muchos autores recomiendan ejecutar más pasos de un jugador que el otro, pero a finales de 2016, la opinión del autor es que el protocolo que funciona mejor en la práctica es el descenso simultáneo de gradiente, con un paso para cada jugador [9].

Las dos redes (discriminador y generador), como se dijo en la sección 2.5, están compitiendo en un juego: el discriminador está tratando de distinguir las imágenes reales de las imágenes falsas y el generador está tratando de crear imágenes que engañen al discriminador en esta tarea. Idealmente, la red generadora estará emitiendo imágenes que son indistinguibles de imágenes reales para el discriminador. Las muestras del generador comienzan ruidosas y caóticas, y con el tiempo

Figura 2.8: Ejemplos de imágenes generadas con Deep Convolutional Generative Adversarial Networks [11]. Dataset de entrenamiento: CIFAR-10 [62]

convergen para tener estadísticas de imágenes más plausibles (ver figura 2.8).

El entrenamiento adversario es complicado ya que se tiene que optimizar para un generador de imágenes y un discriminador al mismo tiempo. Este tipo de optimización es difícil de resolver, y si el entrenamiento no fuera estable, no se encontraría este punto de equilibrio entre modelos.

Para clarificar un poco más pensemos un ejemplo concreto. Supongamos que una red recién inicializada genera 200 imágenes, donde en la primer época del modelo generativo se recibe una secuencia de números aleatorios como entrada. La pregunta es: ¿cómo debemos ajustar los parámetros de la red para producir muestras un poco más realistas?

Notar que no estamos en un entorno supervisado simple y no tenemos ningún objetivo explícito deseado para nuestras 200 imágenes generadas; simplemente queremos que parezcan reales. Un enfoque inteligente en torno a este problema es seguir el enfoque de la red adversaria generativa (GAN). Aquí se introduce una segunda red discriminadora (normalmente una red neuronal convolucional estándar) que trata de clasificar si una imagen de entrada es real o generada. Por ejemplo, podríamos alimentar las 200 imágenes generadas y 200 imágenes reales en el discriminador y entrenarlo como un clasificador estándar para distinguir

entre las dos fuentes. Pero además de eso se aplica un algoritmo de backpropagation (Rumelhart et al., 1986a) [30] a través del discriminador y del generador para encontrar cómo debemos cambiar los parámetros del generador para hacer que sus 200 muestras sean un poco más similares a las reales para el discriminador.

Además de la finalidad de generar imágenes, se introduce un enfoque para el aprendizaje semi-supervisado con GAN que implica que el discriminador produzca una salida adicional orientada a clasificar los datos de entrada. Este enfoque permite obtener resultados de última generación [20] en un dataset como MNIST [60], SVHN [61] y CIFAR-10 [62] en entornos con muy pocos ejemplos etiquetados. En el MNIST, por ejemplo, se obtiene una precisión del 99.14 % con sólo 10 ejemplos etiquetados por clase con una red neuronal completamente conectada [20]. Un resultado que está muy cerca de los resultados más conocidos con enfoques completamente supervisados usando los 60.000 ejemplos etiquetados disponibles. Esto es muy prometedor porque la clasificación manual de ejemplos para entrenar puede ser bastante costosa en la práctica [20].

La Figura 2.9 muestra un ejemplo de la aplicación de redes GAN para la generación de rostros.

Figura 2.9: Ejemplo aplicado a la generación de rostros (Deep Convolutional Generative Adversarial Networks) [11]

2.6. Redes convolucionales profundas generativas adversarias

Uno de estos modelos recientes es la red DCGAN (Deep Convolutional Generative Adversarial Networks) de Radford [10]. Este modelo introduce una clase de CNNs llamadas redes convolucionales generativas adversarias y demuestra que son un fuerte candidato para el aprendizaje sin supervisión sobre el datasets de imágenes.

Radford propone y evalúa un conjunto de restricciones sobre la topología de la arquitectura convolucional de GANs que los hacen estables para entrenar en la mayoría de los escenarios. Utiliza a los discriminadores capacitados para tareas de clasificación de imágenes, con otros algoritmos no supervisados. Visualiza los filtros aprendidos por GANs y muestra empíricamente que los filtros específicos han aprendido a dibujar objetos específicos. Prueba también que los generadores tienen interesantes propiedades aritméticas vectoriales que permiten una fácil manipulación de muchas cualidades semánticas de las muestras generadas [10]. Y por último muestra que en la última capa del discriminante se obtienen en forma no supervisada una representación de los datos que extrae características de alto nivel y que puede ser reutilizada por ejemplo para un problema de clasificación.

Esta red toma como entrada 100 números aleatorios muestreados a partir de una distribución uniforme (nos referimos a ellos como un código, o variables latentes, en rojo) y emite una imagen (un arreglo de píxeles de 64x64x3, en verde). Cuando la entrada al generador se cambia de forma incremental, las imágenes generadas también lo hacen; esto muestra que el modelo ha aprendido características para describir el dataset, en lugar de simplemente memorizar algunos ejemplos. La red (en amarillo Fig. 2.11) está formada por componentes de red neuronal convolucional estándar, tales como capas deconvolucionales, capas totalmente conectadas, etc. Se recomienda ver en detalle la red generadora para comprender su arquitectura (ver figura 2.11). Para tener una imagen global del sistema ver las figuras 2.12 y 2.10.

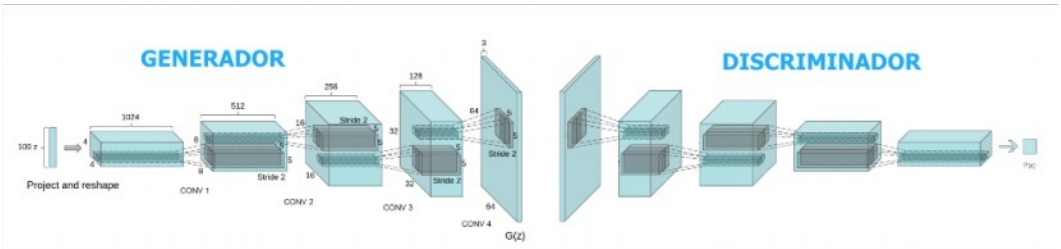

Figura 2.10: Aquitectura de Modelos Generativos (DCGAN) [10].

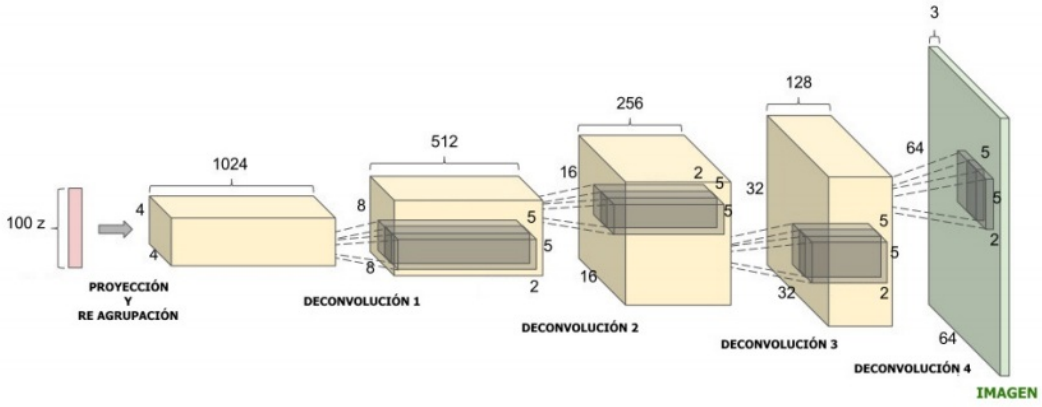

Figura 2.11: Red Generador en detalle (DCGAN) [72].

DCGAN, como la mayoría de los modelos generativos, se inicializa con pesos aleatorios, por lo que un código aleatorio conectado a la red generaría una imagen completamente aleatoria. Sin embargo, la red tiene alrededor de 35 millones de parámetros que se pueden ajustar, y el objetivo es encontrar una configuración de estos parámetros que hace que las muestras generadas a partir de códigos aleatorios se parezcan a los datos de entrenamiento. O para decirlo de otra manera, se quiere que la distribución del modelo coincida con la distribución de datos verdaderos en el espacio de las imágenes. Nótese que el generador no está expuesto a los datos de forma directa, sino que aprende de la señal que llega del discriminador. Si uno observa que el generador esta capturando la distribución de los datos, es decir, genera imágenes símil a las reales, entonces necesariamente el discriminador tiene que haber extraído características relevantes del dataset.

DCGAN usa *batch normalization* (Ioffe and Szegedy, 2015 [33]) en la mayoría

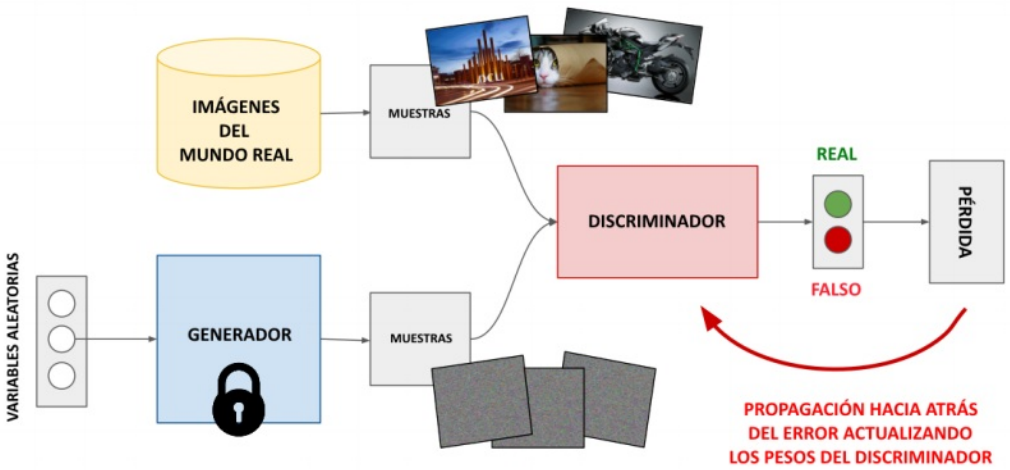

Figura 2.12: Estructura completa de la red (DCGAN) [36]

de las capas tanto del discriminador como del generador, con los dos minibatches para el discriminador normalizados por separado. La última capa del generador y la primera capa del discriminador no están normalizadas por esta técnica, por lo que el modelo puede aprender la media y la escala correctas de la distribución de datos.

Cabe mencionar que DCGAN también sufre del problema de colapsos mencionado en la sección 2.5.

2.7. Máquinas de vectores soporte

Las máquinas de vectores de soporte (SVM) son un conjunto de métodos de aprendizaje supervisados utilizadas para la detección de clasificación, regresión y valores atípicos [54].

Formalmente, una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta [3].

Las ventajas de las máquinas de vectores de soporte son[54]:

- Efectivo en espacios de alta dimensión.
- Sigue siendo efectivo en casos donde el número de dimensiones es mayor que el número de muestras.
- Utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamados vectores de soporte), por lo que también es eficiente desde el punto de vista de la memoria.
- Versátil: se pueden especificar diferentes funciones del Kernel para la función de decisión. Se proporcionan núcleos comunes, pero también es posible especificar kernels personalizados [56].

Las desventajas de las máquinas de vectores de soporte incluyen[54]:

- Si la cantidad de funciones es mucho mayor que la cantidad de muestras, evite el ajuste excesivo al elegir las funciones Kernel y el término de regularización es crucial.
- Las SVM no proporcionan estimaciones de probabilidad directamente, estas se calculan utilizando una costosa validación cruzada.
- No escala bien en la cantidad de datos.

Se introduce este concepto aquí porque en este tesina realizaremos una combinación de la red DCGAN con una SVM, siguiendo (y motivados) por el trabajo *"Deep Learning using Linear Support Vector Machines"* [45]. En el mismo los autores comparan los desempeños de utilizar softmax en la última capa de la red profunda con utilizar una SVM con kernel linear; aplicado al reconocimiento de expresiones faciales.

Capítulo 3

Metodología

La construcción del dataset de entrenamiento y la elección de los métodos de aprendizaje automáticos considerados están orientados a obtener un método tal que dada una imagen tomada sobre el campo sembrado pueda detectar automáticamente las regiones de la imagen donde hay presencia de maleza. Es decir que el objetivo es poder etiquetar por regiones la imagen completa. Sin embargo trabajamos con parches de 64x64 píxeles, sobre una resolución total de 1920x1080 px para la imagen completa, con los cuales se define el problema de clasificación (presencia o no de maleza o soja). Para clasificar se selecciona una región de interés dentro de la imagen, se la divide en recortes de 64x64 px y estos son ingresados a la red para su clasificación.

Se decidió utilizar tamaños de 64x64 px para los parches dado que es la resolución considerada en Ref.[10] donde se propone DCGAN y se inició la investigación con ese valor. Luego al visualizar los resultados del generador se determinó que este tamaño era suficientemente local para aislar la maleza del entorno.

Se propone evaluar además el aporte del aprendizaje no supervisado sobre parches sin etiquetar utilizando distintas estrategias de aprendizaje semi-supervisado. Adicionalmente se desea evaluar si la clasificación de la región de 64x64 px puede ser mejorada incorporando información de contexto, es decir información de la vecindad del parche a clasificar.

La construcción del dataset y los métodos propuestos que se detallan a continuación están motivados por estas consideraciones.

3.1. Conjunto de datos

Al no conseguirse bases de datos para atacar esta problemática, se avanzó en la construcción de un dataset de filmaciones de campo sembrado con soja en distintas etapas de crecimiento (desde fase V6 a R2 [34]) siempre dentro de la etapa de crecimiento que nos interesa, es decir, cuando el cultivo está fuera de la tierra y antes de que cierre el surco. Para la misma se realizaron distintas grabaciones con distintos dispositivos en distintos momentos del día para obtener variaciones en la luminosidad. Las grabaciones se realizaron a distintas velocidades y en distintos lotes, con mayor o menor presencia de maleza y con y sin presencia de rastrojo.

Generar la base de datos de entrenamiento implicó ir al terreno a capturar con videos los posibles escenarios a reconocer y desarrollar una herramienta que simplifique el etiquetado de malezas sobre las imágenes adquiridas. Aún así la cantidad de datos etiquetados, dada las características del problema, no son suficientes para entrenar el modelo profundo en forma completamente supervisada. Por esta razón se decidió utilizar un esquema de aprendizaje semi-supervisado.

Se trabajó con videos de 1920x1080 a 30fps. Para cada frame se realiza el reconocimiento seleccionando una región de interés de 576 píxeles de alto y 1472 de ancho centrado en la imagen; es decir, la región de interés de cada frame tiene un tamaño de 1472x576 (ver figura 3.2). Se determinó de esta forma ya que el foco de la cámara mantiene muy buena nitidez en esta región y desenfoca hacia el horizonte. En total, en cada región de interés hay 207 parches de 64x64 a analizar (23x9).

Para la etapa supervisada, se consideraron los parches de 64x64px, dentro del área de interés, de cada frame, con sus respectivos entornos. Un entorno es la región superior consecutiva de pixeles que rodea al parche de 512x512px como muestra la figura 3.3. Se tomaron frames aleatorios no consecutivos para lograr una buena variedad. Hay que destacar que se separaron videos completos para la etapa de test y que no fueron utilizados en ninguna etapa de entrenamiento.

La forma de etiquetado fue manual por medio de una plataforma web de eti-

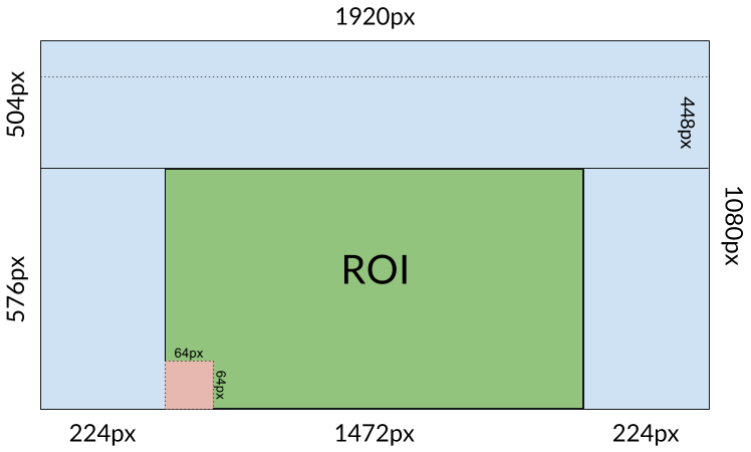

Figura 3.1: Región de interés sobre cada frame

Figura 3.2: Ejemplos de las 3 clases a etiquetadas.

quetado colectivo creada exclusivamente para este proyecto (ver figura 3.4). Las personas que realizaron el etiquetado están relacionadas de alguna forma con el sector agropecuario, como por ejemplo productores, pero no necesariamente ingenieros agrónomos. Participaron en total 23 personas y se etiquetó en cada frame las siguientes 3 clases: soja, maleza y entre-surco.

En esta web de etiquetado se visualizaron frames al azar donde se pedía etiquetar de forma aleatoria maleza, soja o entre-surco; para generar un etiquetado aleatorio del dataset pero mostrando el frame completo con el fin de que la persona tenga información de contexto para tomar la decisión. El tamaño de cada sección a seleccionar es de 64x64px. De forma automática los entornos correspon-

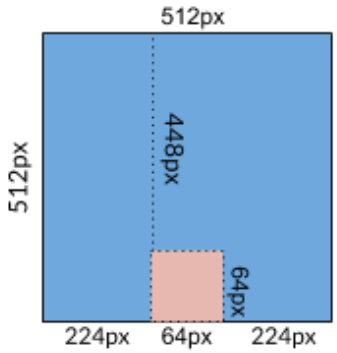

Figura 3.3: Entornos de trabajo de 512x512px

dientes a cada parche son etiquetados con la misma etiqueta que el parche.

Se etiquetaron 200 frames, generando 5102 parches de 64x64 etiquetados de entrenamiento y 4075 de testeo. Esto representa el 5 % de los datos no supervisados. Se espera que esta cantidad de datos sea suficiente ya que la etapa supervisada se utiliza para refinar sólo la última capa de la red, mientras que el aprendizaje intensivo se da en la etapa no supervisada.

Para la etapa no supervisada se seleccionaron, por cada frame, 25 parches de 64x64 pixeles de ubicaciones aleatorias dentro de la región de interés. Sumado a esto para cada frame se seleccionaron, de forma aleatoria, 5 recortes de 512x512 pixeles para generar datos no supervisados asociados a los entornos de la región a clasificar (ver figura 3.3), que son escalados a 64x64 pixeles.

3.2. Métodos

Se utilizaron redes neuronales convolucionales con aprendizaje semi supervisado utilizando GAN en la etapa no supervisada. En una primer etapa se hicieron pruebas con algoritmos DCGAN, K-means[65] y PCA[66] sobre datos sin etiquetar para tener una primer aproximación del problema visualizando cómo se ven y agrupan los datos en el espacio de representación correspondiente. Se buscó verificar que la representación de los datos obtenida con DCGAN representa una mejora respecto del espacio original (pixeles crudos) y de una extracción básica

Figura 3.4: Web para etiquetado colaborativo de parches en frames aleatorios.

de características.

Luego se entrenó una red DCGAN sobre los parches de 64x64px recortados de cada frame de los videos de entrenamiento y se tomó hasta la penúltima capa de la red discriminante. Sobre esta representación se entrena una *Support Vector Machine* (SVM) [54] utilizando datos etiquetados (ver figura 3.5). También se tomó la penúltima capa de la red discriminante de aplicar DCGAN a los entornos y se realizó una concatenación con la penúltima capa de la red discriminante de aplicar DCGAN a los parches. Nuevamente, con estas dos redes sin sus últimas capas y concatenadas, se entrenó una SVM.

Para los distintos algoritmos se realizó validación cruzada [40] para seleccionar los mejores parámetros, usando kfolds [55], sobre los kernels Linear, RBF, Poly (ver en [56]) de la SVM.

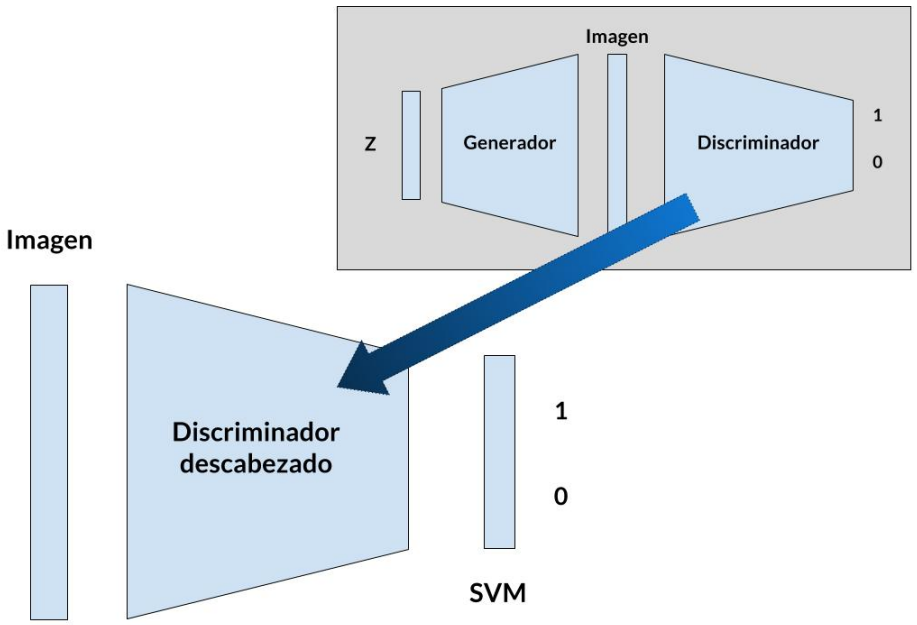

Figura 3.5: Resumen de la arquitectura desarrollada.

En detalle se realizaron los siguientes experimentos para DCGAN:

- Aplicar SVM sobre los datos etiquetados crudos (ver gráficamente Fig. 3.6).
- Aplicar SVM + DA (Aumentación de datos, ver Sec. 3.3) sobre los datos etiquetados crudos (ver gráficamente Fig. 3.7).
- Utilizar la red discriminante de DCGAN entrenada sobre parches no etiquetados y aplicarla sobre parches etiquetados aumentados con DA. Sobre la representación de la penúltima capa del discriminante entrenar una SVM (ver gráficamente Fig. 3.8).
- Concatenar las representaciones de las redes discriminantes de DCGAN entrenadas sobre entornos y sobre parches y sobre este espacio entrenar una SVM con los datos etiquetados (ver gráficamente Fig. 3.9).
- Agregar al ítem anterior DA sobre los datos etiquetados de parches (ver gráficamente Fig. 3.10).

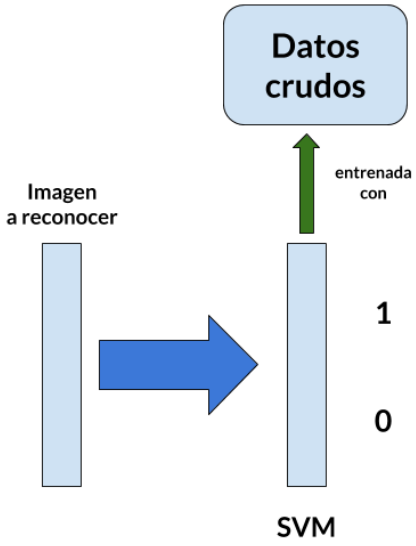

Figura 3.6: Aplicar SVM sobre los datos etiquetados crudos.

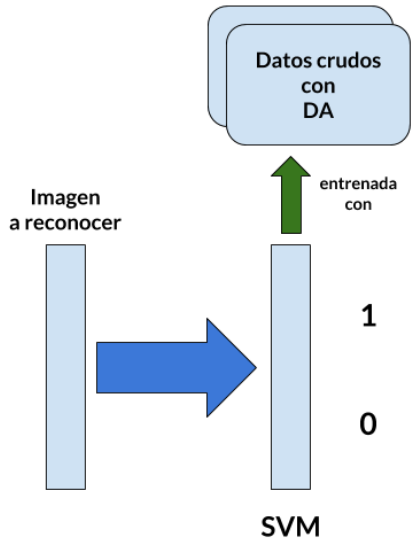

Figura 3.7: Aplicar SVM + DA.

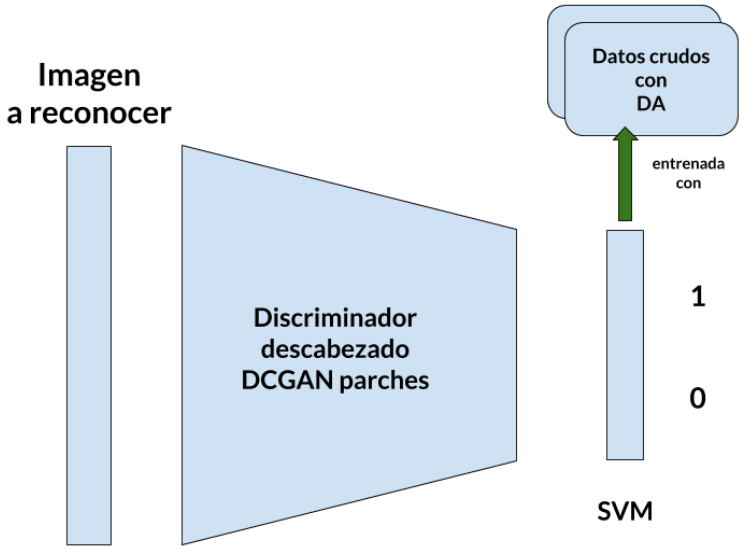

Figura 3.8: Aplicar DCGAN parches + SVM + DA.

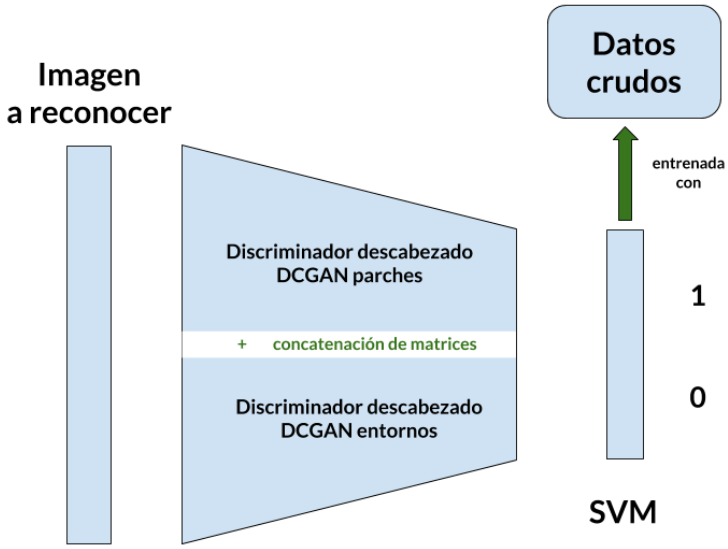

Figura 3.9: Aplicar DCGAN parches + DCGAN entornos + SVM.

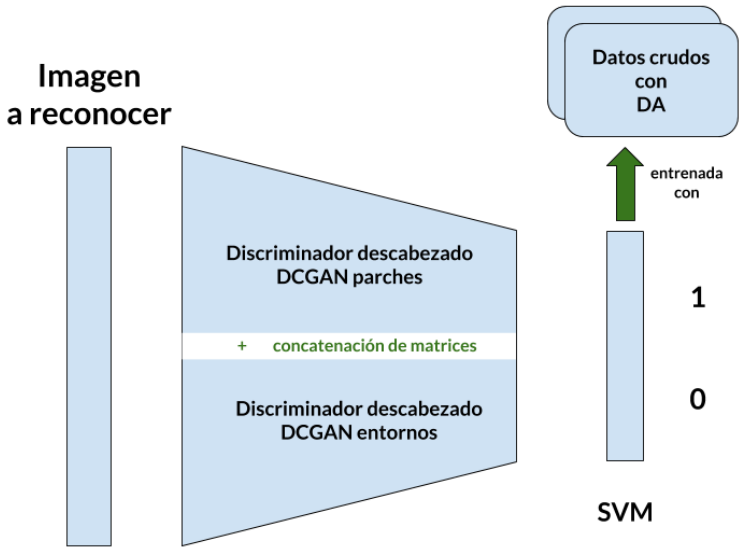

Figura 3.10: Aplicar DCGAN parches + DCGAN entornos + SVM + DA.

De esta forma se entrenaron dos redes DCGAN en 50000 iteraciones (22 épocas), una con los parches y otra con los entornos correspondientes (ver figura 3.11).

Luego de esto se decidió experimentar, de forma preliminar, con un aprendizaje semi supervisado de DCGAN. En este caso al discriminante se le agrega una capa de salida extra que realiza la clasificación. El clasificador así obtenido comparte con el discriminante todas las capas excepto la última. A cada iteración de DCGAN se le intercala una iteración de entrenamiento del clasificador sobre los datos etiquetados. De esta forma se entrena en forma conjunta DCGAN y el clasificador, evitando el sobre ajuste de este último sobre la pequeña fracción de datos etiquetados. Utilizando una fracción de datos de validación se puede determinar en qué iteración se obtiene el mejor modelo (ver Fig. 3.12) y no necesariamente retener el correspondiente a la última iteración como en los casos anteriores. Nos referiremos a este método como DCGANCLASS.

En DCGANCLASS se entrenó de forma semi-supervisada utilizando una porción del conjunto de train como validación y manteniendo invariable el conjunto de test como en todos los otros casos, es decir, no se utilizó para nada el conjunto de test durante el entrenamiento.

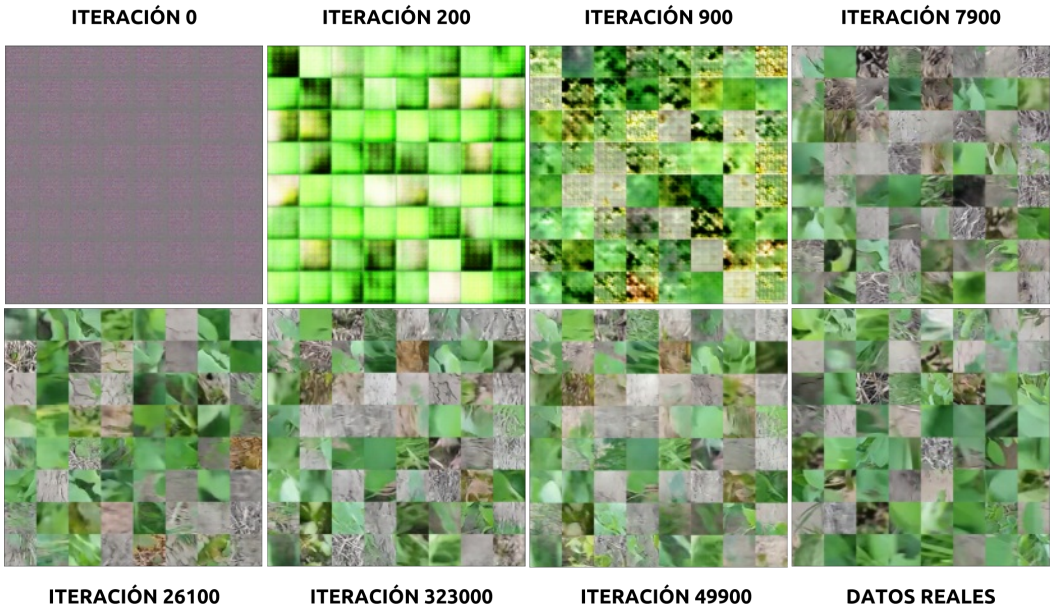

Figura 3.11: Evolución de la red DCGAN sobre parches a lo largo de las iteraciones. Se tomaron 64 muestras al azar generadas por la red y se la comparó con una muestra de datos reales.

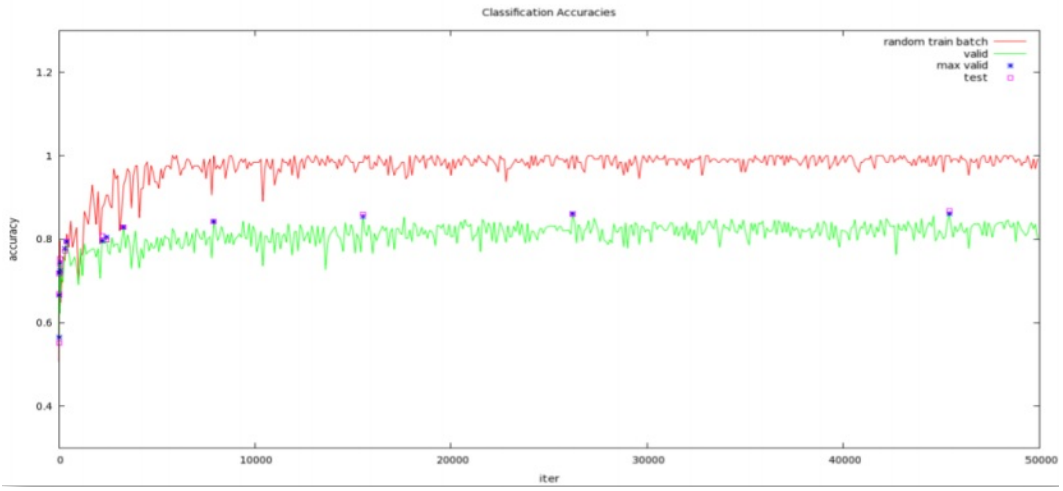

Figura 3.12: Accuracy del clasificador a lo largo de las 50000 iteraciones evaluado en el conjunto de validación (línea verde) y en el conjunto de entrenamiento (línea roja). Los puntos azules marcan los máximos de accuracy sobre el conjunto de validación.

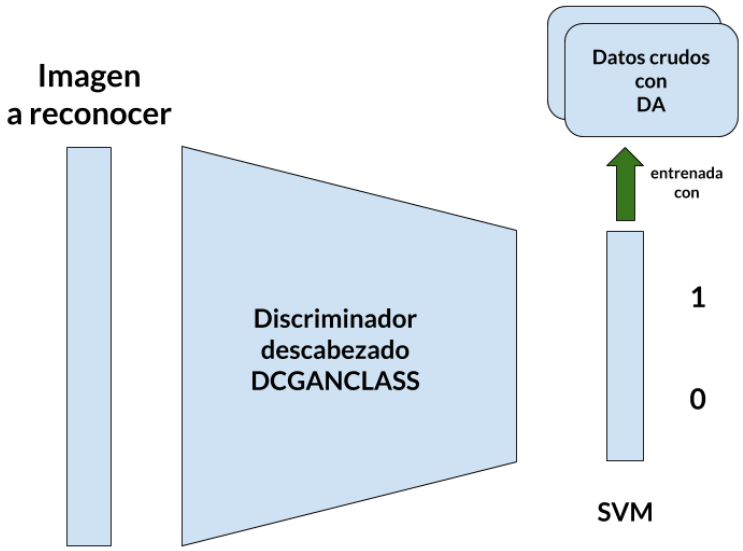

Figura 3.13: Aplicar DCGANCLASS + SVM + DA.

En detalle se realizaron los siguientes experimentos para DCGANCLASS:

- Utilizar la red discriminante de DCGANCLASS y en la penúltima capa aplicar SVM junto con data augmentation de los datos (ver gráficamente Fig. 3.13).
- Combinar la red discriminante de DCGANCLASS sobre parches con el discriminante obtenido con DCGAN sobre entornos. Entrenar SVM sobre las representaciones concatenadas (ver gráficamente Fig. 3.14).
- Ídem anterior, con DA (Aumentación de datos) (ver gráficamente Fig. 3.15).

Dadas las características de la arquitectura DCGAN es interesante evaluar la capacidad de aprendizaje del algoritmo utilizando distintos tamaños para los mapas de características que modifican los tamaños de todas las capas, tanto del generador y como del discriminante, disminuyendo la capacidad de ambos modelos. Podemos visualizar en la imagen 3.16, como ya vimos anteriormente, la arquitectura con tamaño de mapa 128. Las variaciones de este parámetro experimentadas fueron 32, 64 y 128.

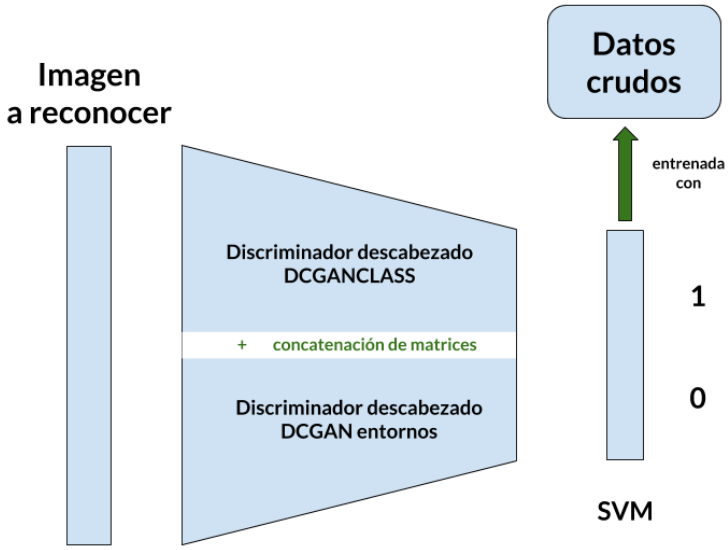

Figura 3.14: Aplicar DCGANCLASS + DCGAN entornos + SVM.

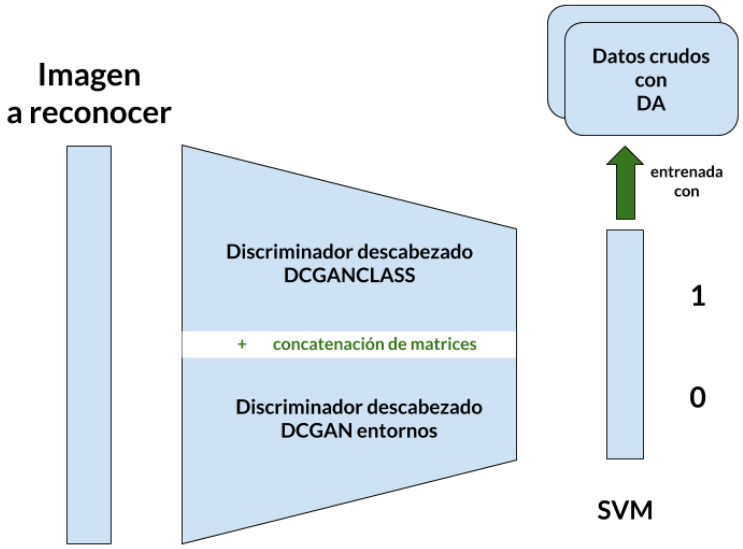

Figura 3.15: Aplicar DCGANCLASS + DCGAN entornos + SVM + DA.

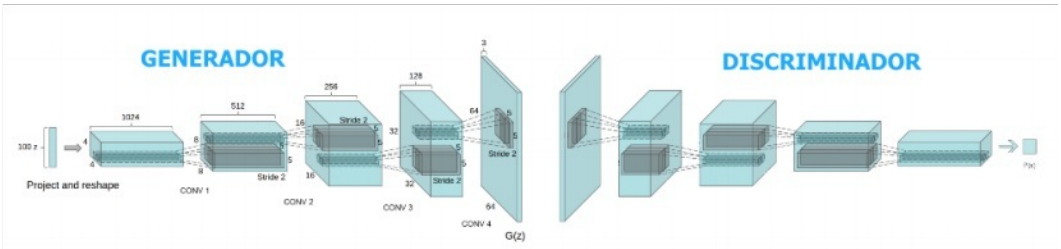

Figura 3.16: Arquitectura de Modelos Generativos (DCGAN) [10].

Utilizando modelos de los distintos tamaños recomendados, las capas intermedias nos quedan de los siguientes tamaños:

	Generador		
Tamaño de modelos	**32**	**64**	**128**
Input	(100)	(100)	(100)
CONV1	(256, 4, 4)	(512, 4, 4)	(1024, 4, 4)
CONV2	(128, 8, 8)	(256, 8, 8)	(512, 8, 8)
CONV3	(64, 16, 16)	(128, 16, 16)	(256, 16, 16)
CONV4	(32, 32, 32)	(64, 32, 32)	(128, 32, 32)
Output	(3, 64, 64)	(3, 64, 64)	(3, 64, 64)

Cuadro 3.1: Red Generador

	Discriminante		
Tamaño de modelos	**128**	**64**	**32**
Input	(3, 64, 64)	(3, 64, 64)	(3, 64, 64)
CONV1	(128, 32, 32)	(64, 32, 32)	(32, 32, 32)
CONV2	(256, 16, 16)	(128, 16, 16)	(64, 16, 16)
CONV3	(512, 8, 8)	(256, 8, 8)	(128, 8, 8)
CONV4	(1024, 4, 4)	(512, 4, 4)	(256, 4, 4)
Output	(1)	(1)	(1)

Cuadro 3.2: Red Discriminante

En ambos casos se hicieron pruebas con distintos tamaños de arquitectura del modelo DCGAN: 32, 64 y 128; y con distintas semillas de números aleatorios, con 5 corridas cada modelo y sus 50000 iteraciones.

Figura 3.17: Ejemplo de data augmentation sobre los datos

3.3. Aumentación de datos

Para reforzar el aprendizaje supervisado, dado que adquirir datos etiquetados es muy costoso, se decidió utilizar la técnica de *data augmentation* (DA) [41, 44] para mejorar el tamaño del dataset. Esta técnica consiste en tomar imágenes que ya están en el conjunto de datos de entrenamiento y aplicarle transformaciones simples para crear muchas versiones alteradas de las mismas. Esto proporciona más imágenes para entrenar, pero también puede ayudar a exponer al clasificador a una variedad más amplia de situaciones de iluminación y coloración para que sea más robusto.

Esta técnica se obtuvo del módulo de la librería Keras [53] que realiza pequeñas perturbaciones aleatorias sobre las imágenes basadas en rotaciones, cambios de coloración, espejados, entre otras transformaciones (ver figura 3.17). Se sabe que esta estrategia produce en general una mejora del desempeño en clasificación de imágenes [44].

Para este caso se volvió a generar el dataset de imágenes etiquetadas recortando ahora parches de 96x96 pixeles (en vez de 64x64 pixeles como fue en la etapa anterior), ya que la función de la librería genera efectos indeseados en los bordes de las imágenes (por ejemplo en las rotaciones). Recortando el parche central de 64x64 luego de aplicar DA sobre parches de 96x96 se descartan estos efectos de borde.

3.4. Herramientas de trabajo

Se trabajó en un cluster del CIFASIS con nodos de 8 núcleos Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, 16gb de RAM, GeForce GTX 970, 4gb de memoria. Un entrenamiento completo de DCGAN en estas arquitecturas ronda las 24 horas.

Se dispuso además de una máquina virtual de Microsoft Azure [57] de 6 núcleos Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz, 56gb de RAM, Tesla K80, 12gb de memoria; y del servidor de cómputo del Laboratorio de Ciencias de la Computación de la Facultad de Ciencias Exactas, Ingeniería y Agrimensura de la Universidad Nacional de Rosario que dispone de una placa Tesla K40 con 12gb de memoria.

Se utilizó Python con Lasagne [63] basado en Theano [64] para la implementación de los modelos profundos y ScikitLearn [65] para la implementación de SVM.

Capítulo 4

Resultados

Para evaluar lo trabajado hasta el momento y el potencial de las redes DCGAN realizamos distintos experimentos comparando cuánto aporta cada técnica sobre el problema, basados en el porcentaje de aciertos sobre el conjunto de testeo etiquetado previamente seleccionado. Se evalúan los resultados sobre los métodos mencionados en la sección 3.2.

4.1. Preliminares

A modo de motivación se decidió explorar el problema en primera instancia con el algoritmo k-means para evaluar la representación obtenida con DCGAN. Esto se realizó utilizando únicamente los datos no etiquetados, es decir que esta primer aproximación está basada únicamente en aprendizaje no supervisado.

En particular con el algoritmo K-means [65] buscando 3 clusters se espera que estos tres clusters capturen las tres categorías más evidentes en las imágenes: soja, maleza y surco. Para esto se tomó la penúltima capa de DCGAN, aplicada sobre los parches de 64x64 sacados de los frames, y se la utilizó como entrada del algoritmo K-means buscando los 3 clusters. Esto se contrastó con el resultado de aplicar K-means por un lado, sobre la representación de pixeles crudos directamente y por otro lado sobre el resultado de aplicar PCA a los parches, para obtener la misma dimensionalidad que la penúltima capa de DCGAN [1].

Las Figuras A, B y C en 4.1 muestran los resultados obtenidos con k-means

[1]La dimensionalidad de DCGAN en su penúltima capa es de 1024 características

($k = 3$) aplicado sobre los datos crudos, sobre PCA y sobre la representación obtenida con DCGAN, respectivamente.

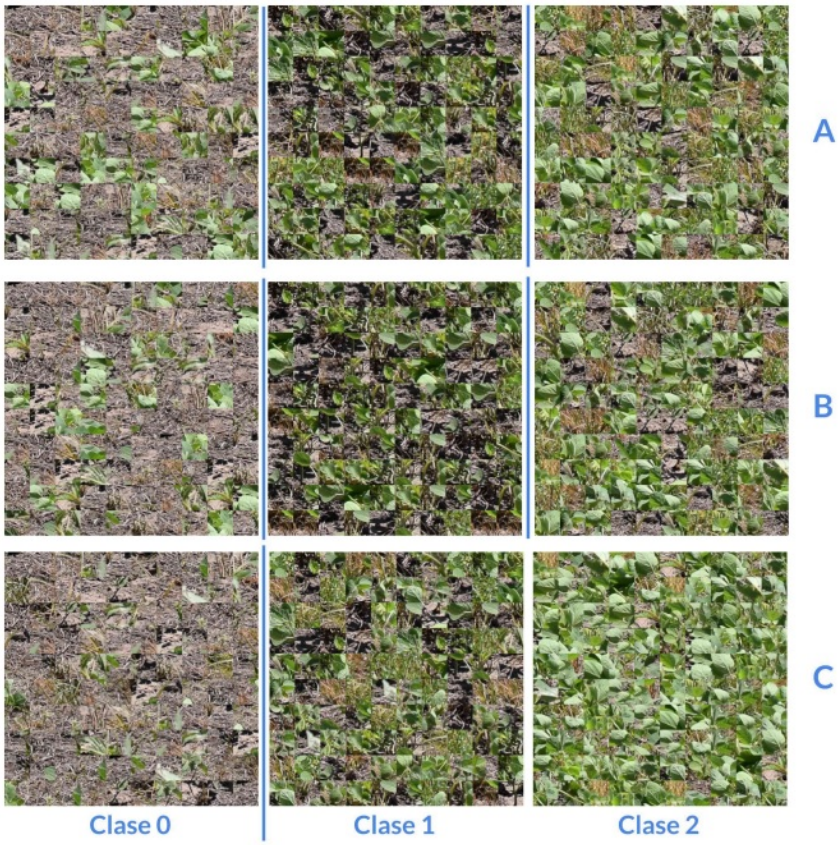

Figura 4.1: Kmeans aplicado sobre datos crudos (A), con reducción de dimensionalidad (B) y con DCGAN (C)

Si se analizan visualmente estas imágenes se puede notar que existe una mejora del caso C (usando DCGAN) en la homogeneidad de las imágenes agrupadas, comparado con los casos A (Kmean sobre datos crudos) y B (Kmean sobre PCA); destacando una coincidencia entre los clusters 0, 1 y 2 respecto a las tres clases esperadas: entre-surco, maleza y soja. Se puede ver que las Clases 0 y las Clases 2 de los distintos algoritmos (A, B y C) poseen imágenes similares entre sí. La Clase 1 se dificulta más pero también esta bien formada. Se puede visualizar en la siguiente imagen (Fig. 4.2) la composición de los ejemplos en cada espacio separados por la formación de clusters de cada algoritmo.

También se analizaron varias ejecuciones distintas y se detectó que la for-

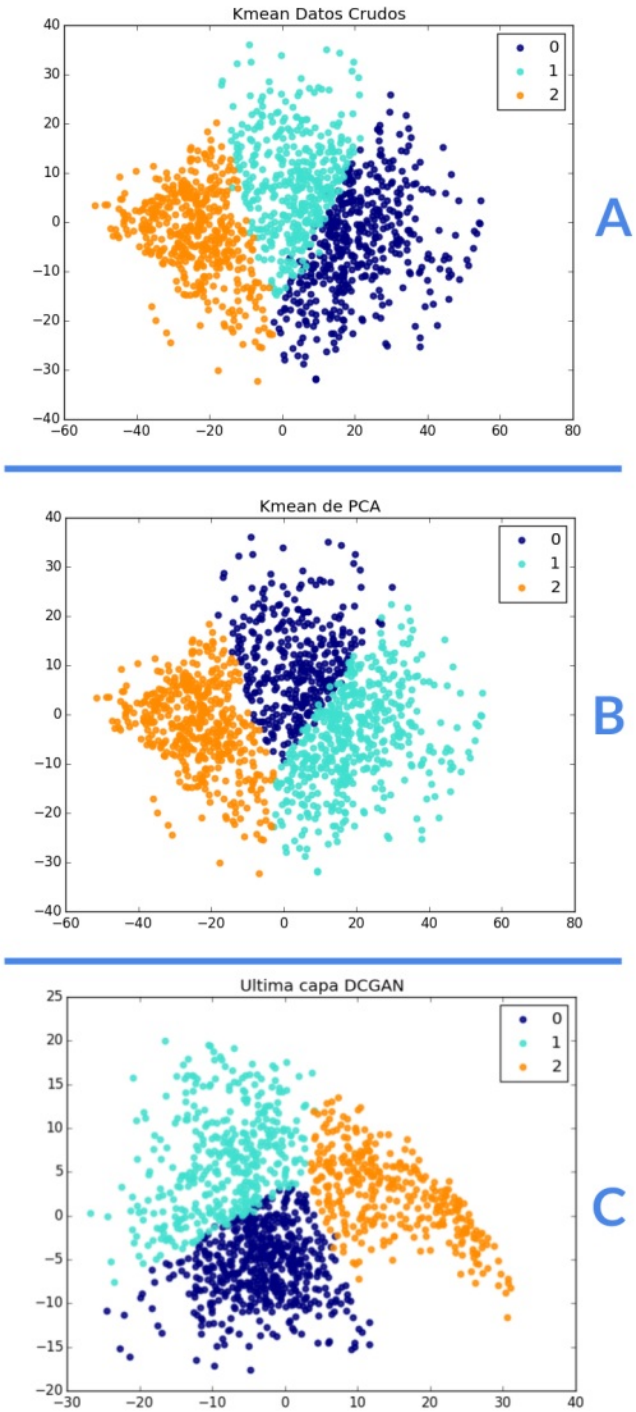

Figura 4.2: Kmeans aplicado sobre datos crudos (A), con reducción de dimensionalidad (B) y con DCGAN (C)

mación de clusters en el caso A y B de la figura 4.1, varía entre un 15 % y un 32 % (en cantidad de elementos); comparada con el agrupamiento de clusters en la representación obtenida con de DCGAN que se mantiene estable en todas las ejecuciones. Esto es una motivación que nos da indicio de que es una buena herramienta para avanzar hacia el problema de clasificación utilizando esta representación basada en entrenamiento no supervisado.

4.2. DCGAN

En esta etapa es interesante comparar cuál es el aporte de cada una de las herramientas mencionadas anteriormente en la sección 3.2: DCGAN sobre los parches, DCGAN sobre los entornos, DCGAN + SVM con data augmentation (DA). Para este primer trabajo se decidió reducir el espacio de clases de 3 (entre-surco, soja y maleza) a 2 (maleza y no maleza), ya que lo interesante es diferenciar la maleza de su entorno; quedando a futuro analizar las herramientas con 3 o más clases. Para ver el detalle de los métodos experimentados ver sección 3.2.

Se puede visualizar las imágenes generadas por la red comparadas con las reales en la figura 4.3.

Figura 4.3: Resultados de DCGAN para: (A) parches de 64x64 y (B) para entornos de trabajo

En la Tabla 4.1 se presentan los resultados basados en el porcentaje de acierto sobre el mismo conjunto de testeo (sin aplicar data augmentation sobre este

conjunto).

Cuadro 4.1: Resultados de evaluación

Accuracy	**Train**	**Test**
KMEANS	55 %	54 %
SVM	99 %	67 %
SVM + DA (Data Augmentation)	100 %	54 %
DCGAN parches + SVM + DA	88 %	83 %
DCGAN parches + DCGAN entornos + SVM	93 %	87 %
DCGAN parches + DCGAN entornos + SVM + DA	91 %	86 %

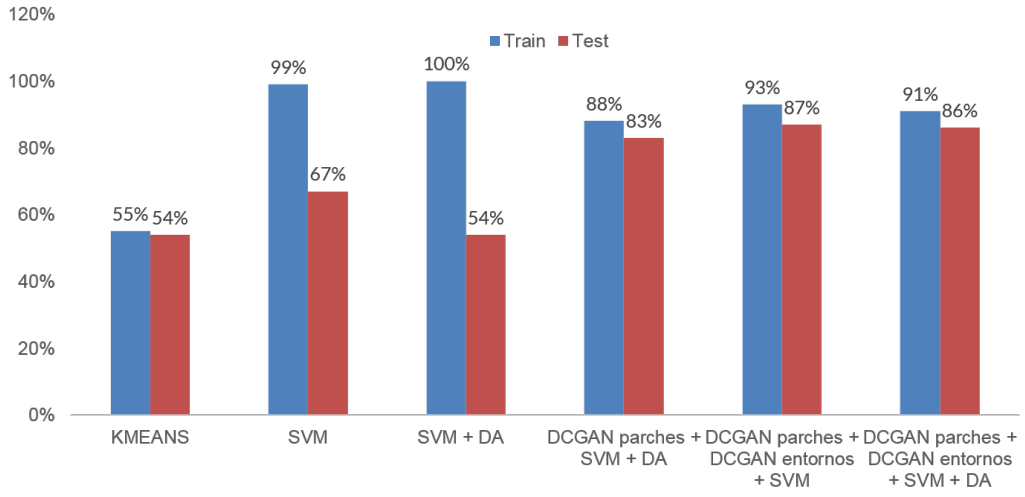

Figura 4.4: Mostramos los resultados de forma gráfica 4.1

Se puede observar que el método que anduvo mejor, basado en el accuracy sobre el conjunto de test, fue el método combinado de *DCGAN parches + DCGAN entornos + SVM*. Esto muestra una mejora en el aporte de la red de entornos como se esperaba. Sin embargo la técnica de DA no tuvo el efecto esperado, lo cual puede deberse a un sobreajuste sobre los datos basados en los altos porcentajes de accuracy sobre el conjunto de entrenamiento, pero si logró reducir ese sobreajuste.

4.3. DCGANCLASS

Se siguió trabajando de forma preliminar en el entrenamiento del método DCGANCLASS (ver Sec. 3.2). Dados los tiempos solo fue posible aplicar este aprendizaje sobre los parches, quedando a futuro aplicar está técnica sobre los entornos y sobre ambos al mismo tiempo. El mejor modelo fue obtenido a las 28000 iteraciones obteniendo 96 % de acierto sobre el conjunto de entrenamiento, 88 % sobre el conjunto de validación y 87 % sobre test.

En la Tabla 4.2 se muestran los resultados basados en el porcentaje de acierto siguiendo el procedimiento para los resultados anteriores.

Cuadro 4.2: Resultados Preliminares

Accuracy	**Train**	**Test**
DCGANCLASS + SVM + DA	94 %	91 %
DCGANCLASS + DCGAN entornos + SVM	98 %	84 %
DCGANCLASS + DCGAN entornos + SVM + DA	97 %	84 %

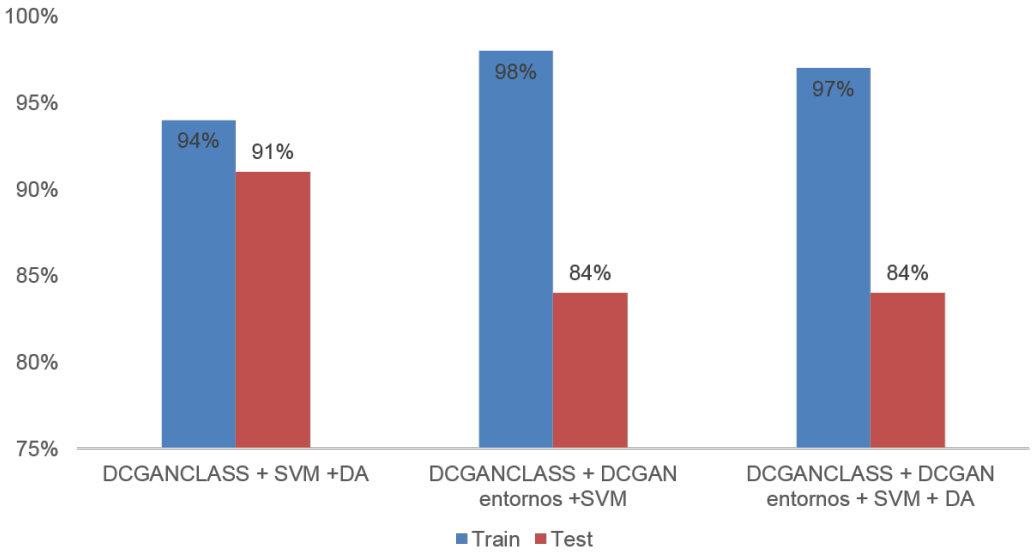

Figura 4.5: Mostramos los resultados de forma gráfica 4.2

Se puede observar una mejora comparada con los resultados obtenidos en la sección DCGAN 4.2, obtenida con el método *DCGANCLASS + SVM + DA*. En este caso la red de entornos no aportó al modelo como se esperaba lo cual puede deberse a no haber entrenado de forma semi-supervisada la red de entornos en conjunto con la red de parches. Tampoco notamos mejora con la técnica DA sobre el conjunto de testeo y no logra reducir significativamente el sobreajuste de los datos.

4.4. Evaluación de rendimientos

Hasta el momento todo lo que se analizó fue para una arquitectura de tamaño 128 que es el tamaño por defecto en DCGAN de Radford. Ahora se analizan los resultados al generar imágenes con los distintos tamaños de modelos (ver Sec. 3.2). En primer lugar visualizamos las imágenes generadas por la red de entornos (Fig. 4.6) y luego por la red DCGANCLASS con parches (Fig. 4.7).

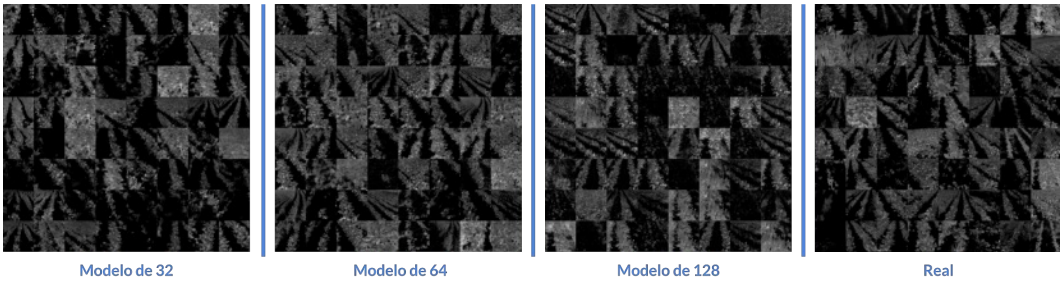

Figura 4.6: Modelos para entornos

Figura 4.7: Modelos DCGANCLASS parches

Se puede visualizar en la Figura 4.6 algunos colapsos en la generación de las imágenes, es decir, imágenes generadas que deberían ser distintas pero colapsaron a la misma imagen. Se habla de esto en el marco teórico haciendo referencia a la inestabilidad del método GAN como una de sus desventajas. Si bien en la actualidad existe un método nuevo WGAN [12] que corrige esto, fue publicado durante el periodo de esta tesina. Aún así vale destacar que este generador produce imágenes bastante realistas. Sin embargo en la Figura 4.7 se visualiza que para el caso de los parches no se dan estos colapsos y que claramente genera imágenes muy reales y que visualmente no se nota la diferencia entre los distintos modelos, lo cual da un indicio de que pueden clasificar de formas similares.

Se puede observar las tasas de reconocimientos sobre el conjunto de test en Sec. 4.3 basados en los métodos de trabajo descriptos en la Sección 3.2 (Metodología).

Tamaño de modelos	**32**		**64**		**128**	
Accuracy	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*
DCGANCLASS	**99 %**	**85 %**	**100 %**	**85 %**	**100 %**	**87 %**
DCGANCLASS+ SVM + DA	**92 %**	**90 %**	**93 %**	**90 %**	**94 %**	**91 %**
DCGANCLASS +DCGAN entornos + SVM + DA	**95 %**	**82 %**	**96 %**	**82 %**	**97 %**	**84 %**

Cuadro 4.3: Resultados con distintos tamaños de modelos

Esto muestra que es posible utilizar un tamaño menor del modelo para mejores tiempos de respuestas del algoritmo perdiendo muy pocos puntos de reconocimiento. Dependerá del problema si es necesario y conveniente hacerlo.

En el hardware utilizado, el tiempo de procesamiento fue de 14 segundos por frame (utilizando un tamaño de 128 para los modelos de DCGAN), esto incluye el recorte de los parches y el reconocimiento de los mismos. Se debe tener en cuenta que es una aproximación al problema y que se pueden realizar varias optimizaciones (como por ejemplo evaluar este tiempo con tamaño de modelos de 32 y 64). En este tesis no se priorizó la velocidad de respuesta sino el reconocimiento.

4.5. Reconocimientos

Para evaluar la clasificación de forma visual mostrando los reconocimientos sobre algunos ejemplos se utilizó el caso con mejor resultado de reconocimiento (accuracy). Este fue el caso DCGANCLASS+ SVM + DA con arquitectura de mapa 128 (ver 4.4).

En base al etiquetado del conjunto de test podemos visualizar los siguientes resultados:

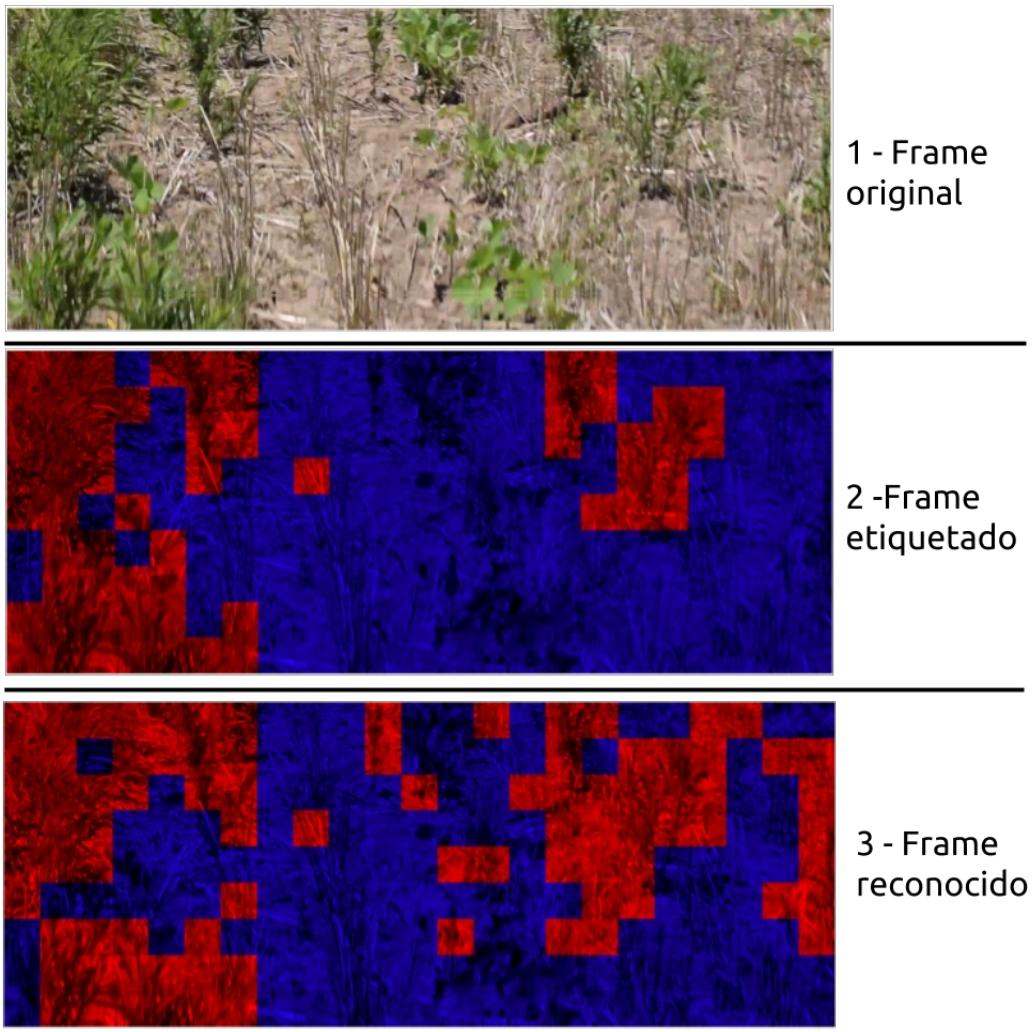

Figura 4.8: Frame de conjunto de etiquetado de test, con su etiquetado y el reconocimiento del algoritmo

Ahora, si visualizamos los resultados sobre un frame de los videos que no fueron utilizados en ninguna etapa del algoritmo y tampoco fueron etiquetados, podemos observar los siguientes resultados:

Figura 4.9: Frame de ejemplo sobre video de test A

Figura 4.10: Mapa de reconocimiento basado en probabilidades de malezas según el algoritmo del frame 4.9

Figura 4.11: Resultado de aplicar el algoritmo de reconocimiento sobre el frame 4.9

Lo que hay que destacar hasta el momento es que el mejor modelo fue el que no utiliza información de los entornos, pero aun así podemos visualizar que en la Figura 4.11 la clasificación local por parches permite recuperar completamente la estructura de surcos, con lo cual los entornos no estarían aportando información adicional.

Otro ejemplo en un lote con condiciones distintas es la Fig. 4.12, donde también se visualiza su mapa de reconocimiento por probabilidades en la figura 4.13 y la combinación de ambas figuras en una para apreciar el resultado de reconocimiento del algoritmo se visualiza en la figura 4.14.

Figura 4.12: Frame de ejemplo sobre video de test B

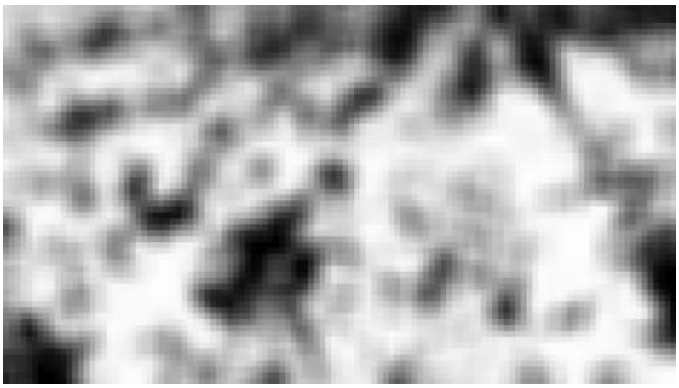

Figura 4.13: Mapa de reconocimiento basado en probabilidades de malezas según el algoritmo, del frame 4.12

Figura 4.14: Resultado de aplicar el algoritmo de reconocimiento sobre el frame 4.12

En ambos casos se aprecia visualmente el reconocimiento de la maleza presente en el video.

Compartimos un link a un video de ejemplo:

Figura 4.15: Video de ejemplo aplicando el reconocimiento: https://youtu.be/iKe6AEnFZzk

Capítulo 5

Conclusiones y trabajo a futuro

5.1. Conclusión

Los resultados obtenidos muestran que es posible alcanzar altos niveles de detección de malezas en imágenes de cultivos de soja sin requerir de un proceso intensivo de etiquetado manual de datos para entrenamiento. Esto está fundamentalmente basado en el entrenamiento no supervisado de redes neuronales, que permite alcanzar un 91 % de aciertos utilizando sólo un 5 % de datos etiquetados. Concretamente, se observó que la técnica de entrenamiento no supervisado considerada basada en Redes Neuronales Adversarias captura correctamente la distribución de los datos y permite entrenar una representación de los mismos que simplifica el proceso de clasificación.

Dado que los métodos considerados son de propósito general y resultan agnósticos de la aplicación concreta, resulta interesante de plantearse abordar otros problemas similares donde es muy costosa la adquisición de datos etiquetados, utilizando estas técnicas. Las pruebas con la técnica de Data Augmentation no arrojaron mejoras en la clasificación con SVM. Esto se debe a que SVM tiene una complejidad cúbica respecto la cantidad de datos y limita severamente la cantidad de datos que se pueden generar con esta técnica.

En cuanto a la incorporación de información de contexto, es decir utilizar conjuntamente un campo visual más amplio para obtener la clasificación local, no se pudo determinar si esta información mejora el desempeño; ya que basado en los resultados se observa que la red sin información de contexto logro aprender esta

característica del conjunto de datos. Se requieren más experimentos para poder caracterizar más precisamente su incidencia, realizando aprendizaje semi supervisado con entornos y expandiendo el conjunto de datos para tener una mayor variedad de escenarios de trabajo.

Analizando los últimos resultados preliminares con DCGANCLASS se puede observar que el clasificador auxiliar dentro del aprendizaje semi-supervisado está ayudando a obtener una mejor representación de los datos internamente. La posibilidad de acoplar el entrenamiento de un clasificador al entrenamiento con DCGAN es muy prometedor. Se obtuvieron mejores resultados que los alcanzados anteriormente, lo cual da un indicio de que es el siguiente camino a seguir. Si bien los entornos empeoraron el resultado en este caso, se estima que se debe a que la red no se entrenó en simultáneo con la red de parches.

Otro punto importante es que al reducir los tamaños de la arquitectura de los modelos se observa una leve disminución en las tasas de reconocimiento, lo cual puede ser conveniente para aplicaciones en tiempo real. Una reducción en estos tamaños nos da menor consumo de calculo que se traduce en más cantidad de frames procesados por segundo, permitiendo procesar los videos a una mayor velocidad de captura, es decir, mayor velocidad de trabajo para el equipo pulverizador.

5.2. Trabajo futuro

El trabajo, enmarcado en una tesis de grado, concluye con la finalización de una red neuronal entrenada en forma semi supervisada que incluye, aparte de todo lo analizado, la posibilidad de realizar el aprendizaje supervisado mientras se entrena DCGAN, que en este trabajo se incorpora una versión preliminar del mismo. Sumado a esto a futuro queda hacer un análisis sobre distintos videos de distintos campos, logrando mayor variabilidad en los datos.

Recordar que este trabajo se abordo sobre 2 clases (maleza o no maleza) pero a futuro es interesante investigar el problema con 3 clases (maleza, soja y entresurco). También otro agregado interesante es el de llevar los falsos negativos a

cero para evitar dejar maleza sin reconocer, esto es útil a la hora de pulverizar ya que la soja es resistente a los herbicidas utilizados. Se puede lograr implementando una Curva ROC [52]. Se puede combinar esta técnica aplicando algoritmos de decisión sobre vecinos para determinar la presencia de maleza y técnicas de seguimiento de flujo óptico en video para una vez detectada la maleza seguirla en los siguientes frames del video.

Desde el punto de vista de la problemática se genera una solución muy prometedora a la hora de aplicar herbicidas. Pero es el primer paso hacia una fumigación inteligente, o mejor dicho, hacia un control de malezas inteligente, que permita combatirlas sin aplicar producto (por ejemplo con actuadores robóticos).
Se espera que al incorporar la visión de cámaras se creen redes neuronales profundas donde no solo se pueda diferenciar la maleza del cultivo sino también diferenciar variedades de malezas entre sí para poder aplicar distintas técnicas sobre la misma. Otra alternativa es la complementación con sensores infrarrojos para separar lo verde de las imágenes y reconocer sobre eso las plantas de soja, en vez de las de malezas.

Apéndice A

Ejemplos

La sofisticada máquina de adquisición de datos para generar la base de datos inicial con la que luego se entrenaron los modelos. Este equipo tiene una cámara digital con la que se realizaron las filmaciones (ver Fig. A.1).

Figura A.1: Equipo de adquisición de imágenes

Ejemplos de datos de entrada de las redes:

La Fig. A.2 muestra un ejemplo de parche de trabajo de 64x64 pixeles. La Fig. A.3 muestra un ejemplo de parche de trabajo de 96x96 pixeles usado con data augmentation. La Fig. A.4 muestra un ejemplo de entornos de trabajo de 512x512 pixeles antes de reducir su tamaño para ingresarlo a le red y A.5 muestra su posterior escalamiento a tamaño de 64x64 pixeles. La Fig. A.6 muestra un ejemplo del frame completo.

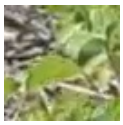

Figura A.2: Ejemplo de parches de trabajo de tamaño 64x64 pixeles

Figura A.3: Ejemplo de parches de trabajo de tamaño 96x96 pixeles usado con data augmentation

Figura A.4: Ejemplo de entorno de trabajo de tamaño 512x512 pixeles

Figura A.5: Ejemplo de entorno de trabajo escaleado a tamaño 64x64 pixeles

Figura A.6: Ejemplo de frame

Ahora vemos otro ejemplo de reconocimiento del algoritmo sobre un lote con rastrojo de trigo:

Figura A.7: Frame de ejemplo sobre video de test C

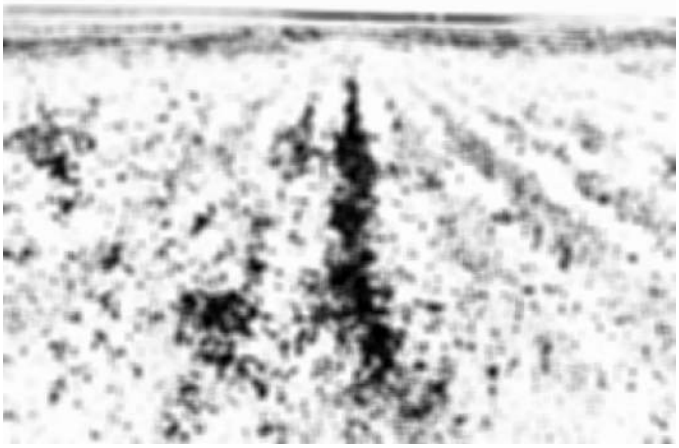

Figura A.8: Mapa de reconocimiento basado en probabilidades de malezas según el algoritmo del frame A.7

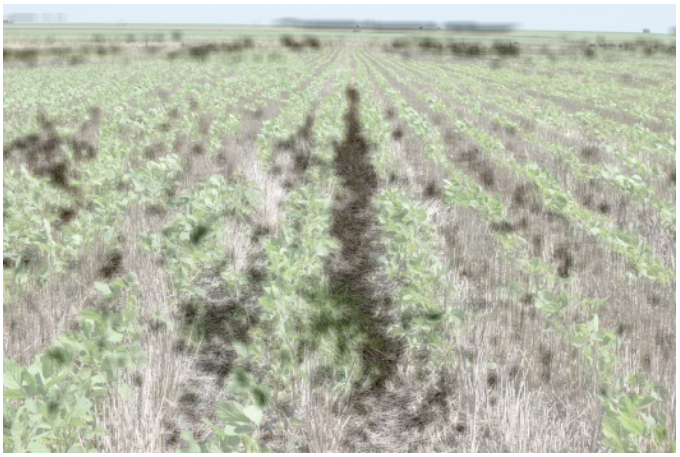

Figura A.9: Resultado de aplicar el algoritmo de reconocimiento sobre el frame A.7

Se analizaron distintos filtros para aplicar sobre los entornos de trabajo basados en la técnica *RG Chromaticity* [68] y su posterior conversión a imagen blanco y negro; con el objetivo de simplificar la información aumentando la intensidad del canal verde y facilitar la identificación de los surcos en la imagen.

La técnica consiste en que dado un color (R, G, B) donde R, G, B = intensidad de rojo, verde y azul; esto se puede convertir en color (r, g, b) dónde r, g, b implica la proporción de rojo, verde y azul en el color original. Matemáticamente tenemos:

$$
\begin{aligned}
r &= \frac{R}{R+G+B} \\
g &= \frac{G}{R+G+B} \\
b &= \frac{B}{R+G+B} \\
r + g + b &= 1
\end{aligned}
$$

Y luego:

$$
\begin{aligned}
R &= \frac{rG}{g} \\
G &= G \\
B &= \frac{(1-r-g)G}{g}
\end{aligned}
$$

Se puede ver ejemplos del mismo en la figura A.10, donde la utilizada como entrada de la red de entornos es la que da resultado de aplicar la técnica RG

Chromaticity convertida a imagen blanco y negro.

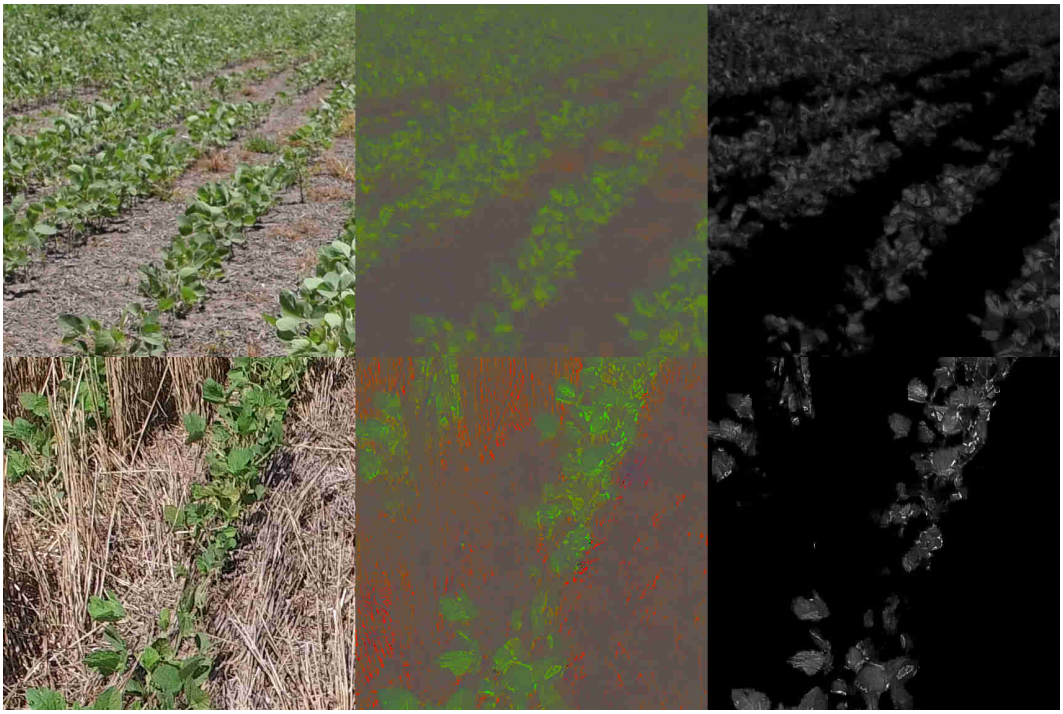

Figura A.10: Ejemplo de entornos de trabajos, los filtros aplicados y su posterior conversión a imagen blanco y negro.

Bibliografía

[1] Machine Learning, Tom Mitchell, McGraw Hill, 1997

[2] Flach, Peter (2012) Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press.

[3] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., Scholkopf, B. (1998). Support vector machines. IEEE Intelligent Systems and their applications, 13(4), 18-28.

[4] J.L. McClelland, E. Rumelhart (eds.) (1986). Parallel Distributed Processing. Vol 1 y Vol 2. Psychological and Biological Models, MIT Press

[5] Martín del Brío, B. and A. Sanz. 2001. Redes neuronales y sistemas Borrosos. Editorial Rama, España

[6] https://es.wikipedia.org/wiki/Perceptron

[7] Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville. MIT Press 2016. http://www.deeplearningbook.org

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. *Generative adversarial nets.* NIPS, 2014.

[9] Ian Goodfellow, NIPS 2016 Tutorial, Generative adversarial nets.

[10] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised representation learning with deep convolutional generative adversarial networks.* arXiv, 2015. https://github.com/Newmu/dcgan_code

[11] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, *Improved Techniques for Training GANs.* Goodfellow, Jun 2016.

[12] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. arXiv preprint arXiv:1701.07875.

[13] Andrés Moltoni, Gerardo Masiá, *Detector de malezas para la aplicación sitio específica de herbicidas*, Innovar 2011. Laboratorio de Electrónica de INTA Castelar, Nov 2011.

[14] Moltoni, A. F., Moltoni, L. A. (2006). Pulverización selectiva de herbicidas. implicancias tecnológicas y económicas de su implementación en la Argentina. Curso Internacional de Agricultura de Precisión. 6. Expo de Máquinas Precisas. 1. 2006 07 25-27, 25 al 27 de julio de 2006. Manfredi, Córdoba. AR.

[15] Biller R. H. 1998. Reduced input of herbicides by use of optoelectronic sensors. Journal of Agricultural and Engineering Research 71 4 , 357-362.

[16] Site-specific weed control: desempeño de un detector de maleza diseñado y construido en el instituto de ingeniería rural de inta castelar. Andrés Moltoni, Luciana Moltoni, Leonardo Venturelli, Adriana Fuica, Gerardo Masiá. Xxxv Congresso Brasileiro De Engenharia Agrícola, Agosto 2006

[17] Shropshire, G. J., K. Von Bargen, and D. A. Mortensen. 1990. Optical reflectance sensor for detecting plants. In Proceedings os SPIE (the International Society for Optical Engineering): Optics in Agriculture, 7-8 November, Boston, Mass., Vol 1379:222-235. Bellinham, Wash.:SPIE Press.

[18] "Malezas: se pierden USD 1300 millones", LA NACION, Sábado 07 de marzo de 2015. http://www.lanacion.com.ar/1773794-malezas-se-pierden-us-1300-millones

[19] "Desarrollaron un sensor para detectar malezas en los cultivos", LA NACION, Viernes 22 de mayo de 2009, http://www.lanacion.com.ar/1130830-desarrollaron-un-sensor-para-detectar-malezas-en-los-cultivos

[20] Andrej Karpathy, Pieter Abbeel, Greg Brockman, Peter Chen, Vicki Cheung, Rocky Duan, Ian Goodfellow, Durk Kingma, Jonathan Ho, Rein Houthooft, Tim Salimans, John Schulman, Ilya Sutskever, and Wojciech Zaremba. *Generative Models*. Jun 2016.

[21] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". Nature 521.7553 (2015): 436-444.

[22] Rasmus, Antti, et al. *Rasmus, Antti, et al.*. Advances in Neural Information Processing Systems. 2015.

[23] Mónica G. Larese, Rafael Namías, Roque M. Craviotto, Miriam R Arango, Carina Gallo, and Pablo M Granitto. Automatic classification of legumes using leaf vein image features. Pattern Recognition, 47(1):158–168, 2014.

[24] Larese, Mónica G., Bayá A. et al. Multiscale recognition of legume varieties based on leaf venation images. Expert Systems with Applications 41.10 (2014): 4638-4647.

[25] Larese, Mónica G., and Pablo M. Granitto. Finding local leaf vein patterns for legume characterization and classification. Machine Vision and Applications (2015): 1-12.

[26] Grinblat, G.L., Uzal, L.C., Larese, M.G., Granitto, P.M., Deep learning for plant identification using vein morphological patterns, Computers and Electronics in Agriculture 127 (2016): 418-424.

[27] Kotsiantis, S. B., Zaharakis, I., Pintelas, P. (2007). Supervised machine learning: A review of classification techniques.

[28] Coates, A., Ng, A., Lee, H. (2011, June). An analysis of single-layer networks in unsupervised feature learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics (pp. 215-223).

[29] Zhu, X., Goldberg, A. B. 2009. Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning, 3 1, 1 130.

[30] Isasi Viñuela, P. E. D. R. O., Galván León, I. M. (2004). Redes de neuronas artificiales. Un Enfoque Práctico, Editorial Pearson Educación SA Madrid España.

[31] https://www.oreilly.com/learning/generative-adversarial-networks-for-beginners

[32] Unsupervised Deep Learning: A Short Review. Juha Karhunen, Tapani Raiko and KyungHyun Cho. Advances in Independent Component Analysis and Learning Machines. ed. S.Kaski Ella Bingham. The Netherlands, 2015. p. Chapter 7, pp. 125-142.

[33] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Sergey Ioffe, Christian Szegedy. arXiv:1502.03167. 2015

[34] Andrade, F. H., Aguirrezábal, L. A. N., Rizzalli, R. H. (2000). Crecimiento y rendimiento comparados. Bases para el manejo del maíz, el girasol y la soja. Buenos Aires: Editorial Médica Panamericana, 61-96.

[35] https://rubenlopezg.wordpress.com/2014/05/07/que-es-y-como-funciona-deep-learning/

[36] https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

[37] Utgoff, P. E., Stracuzzi, D. J. (2002). Many-layered learning. Neural Computation, 14(10), 2497-2529.

[38] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks.

[39] https://en.wikipedia.org/wiki/Convolutional_neural_network

[40] Refaeilzadeh, P., Tang, L., Liu, H. (2009). Cross-validation. In Encyclopedia of database systems (pp. 532-538). Springer US.

[41] https://keras.io/preprocessing/image/

[42] http://www.weedscience.org/

[43] Observatorio socio ambiental de la Soja. http://observatoriosoja.org/nota-breve/uso-de-agroquimicos-en-soja/

[44] CS231n: Convolutional Neural Networks for Visual Recognition Spring 2017, Lecture 7. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

[45] Tang, Yichuan. Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, 2013.

[46] http://www.clei2017-46jaiio.sadio.org.ar/sites/default/files/Mem/EST/est-05.pdf

[47] https://es.wikipedia.org/wiki/Equilibrio_de_Nash

[48] http://csci431.artifice.cc/notes/deep-learning.html

[49] https://es.wikipedia.org/wiki/Funci%C3%B3n_sigmoide

[50] Adam: A Method for Stochastic Optimization, Kingma y Ba, 2014. arXiv:1412.6980

[51] On the Convergence of SGD Training of Neural Networks, Thomas M. Breuel. 12 Aug 2015, arXiv:1508.02790.

[52] Signal detection theory and ROC analysis in psychology and diagnostics: collected papers; Swets, 1996

[53] https://keras.io/

[54] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[55] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

[56] http://scikit-learn.org/stable/modules/svm.html#svm-kernels

[57] https://azure.microsoft.com/es-es/

[58] http://www.casafe.org/

[59] http://inta.gob.ar/

[60] http://yann.lecun.com/exdb/mnist/

[61] http://ufldl.stanford.edu/housenumbers/

[62] http://www.cs.toronto.edu/~kriz/cifar.html

[63] https://github.com/Lasagne/Lasagne

[64] http://www.deeplearning.net/software/theano/

[65] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[66] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[67] https://es.wikipedia.org/wiki/NP-hard

[68] https://en.wikipedia.org/wiki/Rg_chromaticity

[69] http://machinethink.net/blog/convolutional-neural-networks-on-the-iphone-with-vggnet/

[70] https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

[71] http://www.analiticaweb.es/la-prediccion-del-dato-redes-neuronales-artificiales/

[72] https://blog.openai.com/generative-models/