**UNIVERSIDAD MAYOR DE SAN ANDRÉS**

**FACULTAD DE CIENCIAS PURAS Y NATURALES**

**CARRERA DE INFORMATICA**



**TESIS DE GRADO**

**"DIGITALIZACIÓN INTELIGENTE DE RECIBOS MANUSCRITOS PARA PEQUEÑOS COMERCIANTES MEDIANTE UNA APLICACIÓN MÓVIL BASADA EN DEEP LEARNING"**

Para optar al Título de Licenciatura en Informática

MENCIÓN: ingeniería de sistemas informáticos

**POR:** WHILLY EDGAR AMORAGA MAMANI

**TUTOR:** DR. JUAN MARCOS MIRANDA NINA PH.D.

**La Paz - Bolivia**

**2024**

# Índice

[Índice 3](#_Toc198574075)

[Tablas I](#_Toc198574076)

[CAPITULO I 1](#_Toc198574077)

[1 MARCO REFERENCIAL 1](#_Toc198574078)

[1.1 INTRODUCCIÓN 1](#_Toc198574079)

[1.2 ANTECEDENTES 3](#_Toc198574080)

[1.2.1 Antecedentes internacionales. 3](#_Toc198574081)

[1.1.1 Antecedentes nacionales. 3](#_Toc198574082)

[1.3 PLANTEAMIENTO DEL PROBLEMA 5](#_Toc198574083)

[1.3.1 Problema General. 5](#_Toc198574084)

[1.3.2 Problemas Específicos. 6](#_Toc198574085)

[1.4 OBJETIVOS. 7](#_Toc198574086)

[1.4.1 Objetivo General. 7](#_Toc198574087)

[1.4.2 Objetivo Específico. 7](#_Toc198574088)

[1.5 HIPOTESIS 7](#_Toc198574089)

[1.6 OPERACIÓN DE VARIABLES 8](#_Toc198574090)

[1.7 JUSTIFICACIÓN. 8](#_Toc198574091)

[1.7.1 Justificación Social. 8](#_Toc198574092)

[1.7.2 Justificación Económica. 8](#_Toc198574093)

[1.7.3 Justificación Académica. 9](#_Toc198574094)

[1.8 LIMITES Y ALCANCES 9](#_Toc198574095)

[1.8.1 Alcance. 9](#_Toc198574096)

[1.8.2 Limites. 10](#_Toc198574097)

[1.9 METODOLOGIAS 11](#_Toc198574098)

[1.9.1 Metodología De Desarrollo. 11](#_Toc198574099)

[1.9.2 Metodología De La Investigación Científica. 11](#_Toc198574100)

[CAPITULO II 13](#_Toc198574101)

[2 MARCO TEORICO 13](#_Toc198574102)

[2.1 INTRODUCCION 13](#_Toc198574103)

[2.2 REGIMEN TRIBUTARIO SIMPLIFICADO RTS 14](#_Toc198574104)

[2.3 INTELIGENCIA ARTIFICAL (IA) 14](#_Toc198574105)

[2.4 MACHINE LEARNING 16](#_Toc198574106)

[2.5 DEEP LEARNING 17](#_Toc198574107)

[2.5.1 Redes Neuronales. 18](#_Toc198574108)

[2.5.2 Funciones De Activación. 20](#_Toc198574109)

[2.5.3 Redes Neuronales Convolucionales (CNN). 26](#_Toc198574110)

[2.5.4 Redes Neuronales Recurrentes (RNN). 36](#_Toc198574111)

[2.5.5 Yolov5. 42](#_Toc198574112)

[2.5.6 Tesseract OCR. 44](#_Toc198574113)

[2.6 METODOLOGIA MOBILE-D 44](#_Toc198574114)

[CAPITULO III 49](#_Toc198574115)

[3 MARCO APLICATIVO 49](#_Toc198574116)

[3.1 INTRODUCCION 49](#_Toc198574117)

[3.2 FASE DE EXPLORACION 49](#_Toc198574118)

[3.2.1 Establecimiento De Grupo De Interés. 49](#_Toc198574119)

[3.2.2 Requisitos Iniciales. 50](#_Toc198574120)

[3.2.3 Requerimientos Funcionales. 51](#_Toc198574121)

[3.2.4 Requerimientos No Funcionales. 51](#_Toc198574122)

[3.2.5 Definición Del Alcance. 52](#_Toc198574123)

[3.2.6 Establecimiento De Tecnologías. 53](#_Toc198574124)

[3.3 FASE DE INICIACION 57](#_Toc198574125)

[3.3.1 Configuración Del Ambiente De Desarrollo. 57](#_Toc198574126)

[3.3.2 Planificación De Fases. 60](#_Toc198574127)

[3.3.3 Diseño De Aplicación. 62](#_Toc198574128)

[3.4 FASE DE PRODUCCION 63](#_Toc198574129)

[3.4.1 Iteración 1. 64](#_Toc198574130)

[3.4.2 Iteración 2 68](#_Toc198574131)

[3.4.3 Iteración 3 69](#_Toc198574132)

[3.4.4 Iteración 4. 72](#_Toc198574133)

[3.4.5 Iteración 5. 77](#_Toc198574134)

[3.5 FASE DE ESTABILIACION 79](#_Toc198574135)

[3.5.1 Iteración 6. 79](#_Toc198574136)

[3.5.2 Iteración 7. 83](#_Toc198574137)

[3.6 FASE DE FINALIZACION 85](#_Toc198574138)

[CAPITULO IV 86](#_Toc198574139)

[4 EVALUACION DE RESULTADOS 86](#_Toc198574140)

[4.1 METODOLOGIAS DE EVALUACION 86](#_Toc198574141)

[4.1.1 Selección De Estudio. 86](#_Toc198574142)

# Tablas

CAPITULO I

# MARCO REFERENCIAL

## INTRODUCCIÓN

En la actualidad, la transformación digital avanza rápidamente en todos los sectores, incluyendo el comercio. Sin embargo, en países como Bolivia, muchos pequeños comerciantes, especialmente aquellos que pertenecen al Régimen Tributario Simplificado (RTS), continúan utilizando métodos tradicionales para registrar sus ventas, como el uso de notas de venta manuscritos. Esta práctica limita su capacidad de organización, seguimiento de ingresos y acceso a servicios financieros o tecnológicos.

En Bolivia el Padrón Nacional de Contribuyentes Biométrico Digital (PNCBD) en la gestión 2021 reporta una cantidad final de 442.133 contribuyentes activos y habilitados, siendo el 24% perteneciente al Régimen Tributario Simplificado, correspondiente a comerciantes minoristas, artesanos y vivanderos. Boletín Económico Ingresos Tributarios, 2021, Bolivia, (economiayfinanzas.gob.bo portal).

Este régimen facilita las operaciones de pequeños negocios al eximirlos de la obligación de emisión de facturas fiscales. Sin embargo, también presenta limitaciones significativas en la gestión administrativa, el control y análisis de ventas. Muchos de estos contribuyentes suelen depender de registros manuales o notas de venta en talonarios, lo que dificulta la elaboración de balances contables, análisis de ventas y la planificación estratégica, limitando la transparencia operativa y dificultando el acceso a servicios financieros. Además, los clientes no cuentan con comprobantes digitales, lo que limita la transparencia en las transacciones.

Ante esta problemática, se propone el desarrollo de una aplicación móvil Android, capaz de digitalizar notas de venta manuscritas mediante el uso de visión por computadora y redes neuronales, específicamente una combinación de modelos YOLO[[1]](#footnote-1) y CRNN[[2]](#footnote-2). El objetivo principal es transformar automáticamente una imagen de una nota de venta en un recibo digital estructurado, que permita al comerciante registrar, consultar, extender sus ventas de manera rápida, sin necesidad de conocimientos técnicos avanzados ni conexión a internet. Este proyecto busca no solo modernizar procesos, sino también fomentar la inclusión tecnológica al promover el desarrollo en un sector esencial en la economía boliviana.

A lo largo de este documento, los términos “recibo”, “nota de venta” y “proforma” se emplean de forma intercambiable. En el contexto del RTS[[3]](#footnote-3) en Bolivia, estos documentos no siempre corresponden a facturas fiscales, pero cumplen la función de comprobante de venta, registro y respaldo de una transacción. Por razones de concisión, se optó por usar el término “recibo” en el título del trabajo, aunque la mayor parte de los datos y ejemplos provienen de notas de venta manuscritas utilizadas en comercios informales.

La primera parte del trabajo utilizará la metodología Mobile-D[[4]](#footnote-4) y consistirá en la recopilación de notas de venta de diferentes negocios para construir un dataset[[5]](#footnote-5) amplio. Luego, este dataset se usará para entrenar modelos de detección de áreas de interés en las notas manuscritas. Posteriormente, se entrenará una red CRNN para el reconocimiento de texto manuscrito. Finalmente, se procederá al diseño e implementación de la aplicación móvil que integrará ambos procesos.

## ANTECEDENTES

### Antecedentes internacionales.

**Análisis en modelos de atención, tesis de grado, Escuela Técnica Superior de Ingeniería Universidad de Sevilla.**

El proyecto presenta una solución basada en modelos de atención (Bahdanau, Loung, Monotonic). De esta manera se propone que la dependencia de la transcripción realizada de los caracteres anteriores y posteriores al que ocupa en cada momento puede aportar claridad y exactitud a la transcripción (García González, 2020).

**Aplicación de conversión de texto manuscrito. Tesis de maestría, Pontificia Universidad Católica Del Perú.**

Se diseña una aplicación que hace uso de HTR[[6]](#footnote-6) y redes neuronales convolucionales, para la detección y reconocimiento de texto manuscrito históricos peruanos utilizando modelos mixtos (Cruz, 2024) .

### Antecedentes nacionales.

**Aplicación de traducción chino a español, Tesis De Grado, Universidad Mayor De San Andrés.**

Se diseña una aplicación en ordenar que utiliza autómatas finitos determinísticos que hacen uso de OCR[[7]](#footnote-7) para la traducción de chino al español (Garcia, 2015).

El reconocimiento óptico de caracteres (OCR) una herramienta clave para la digitalización, según IBM[[8]](#footnote-8), “El software OCR puede aprovechar la inteligencia artificial (IA[[9]](#footnote-9)) para implementar métodos más avanzados de reconocimiento inteligente de caracteres (ICR[[10]](#footnote-10)) para identificar idiomas o escritura a mano” (IBM International Business Machines Corporation, s.f.).

En 1974, Ray Kurzweil fundó Kurzweil Computer Products, Inc., cuyo producto OCR omnicanal podía reconocer texto impreso en prácticamente cualquier fuente (IBM International Business Machines Corporation, s.f.). Con el tiempo, esta tecnología experimentó un avance exponencial, extendiéndose hacia nuevas áreas como el reconocimiento de texto manuscrito (HTR).

Utilizando HTR (handwritten text recogonize) reconocimiento de texto manuscrito, se aprovechó el uso de las redes neuronales convolucionales (CNN) para el reconocimiento de dígitos manuscritos. Desarrollando LeNet-5, un modelo que revolucionó el reconocimiento de caracteres. Utilizaron el conjunto de datos MNIST como base para sus investigaciones (LeCun, Bottou, Bengio, & Haffner, 1998).

Avances más posteriores realizados en su trabajo, Bluche demostró que las redes neuronales profundas podían superar significativamente los métodos tradicionales de reconocimiento de texto manuscrito, reduciendo las tasas de error de manera considerable (Bluche, 2015).

Estos avances no solo se han limitado a avances académicos, sino que también se han ampliado en diversas áreas, educación, logística, servicios administrativos, comercio, entre otros.

El reconocimiento de texto manuscrito, sigue presentando desafíos, la gran variedad de estilos de escritura, el grosor del trazo y el papel usado, así también el ruido en las imágenes, dificultando las tareas de reconocimiento. Sin embargo, modelos más recientes y el aprendizaje de transferencia, han abierto nuevas posibilidades antes estos desafíos, permitiendo entrenar sistemas más robustos y precisos, con la posibilidad de adaptarse a los diferentes tipos de escritura.

## PLANTEAMIENTO DEL PROBLEMA

### Problema General.

Los pequeños negocios acogidos al Régimen Tributario Simplificado (RTS) representan el 24% del total de contribuyentes activos en Bolivia, según el Padrón Nacional de Contribuyentes Biométrico Digital (PNCBD) del 2021 de un total de 442.133 contribuyentes activos. Aunque este régimen les otorga beneficios significativos, como estar exentos de la emisión de facturas, también representan limitaciones importantes que afectan su gestión administrativa y financiera. Al no contar con herramientas tecnológicas accesibles y adecuadas para sus necesidades, estos negocios se enfrentan a una dependencia casi total de registros manuales como recibos o notas de venta. Esta situación incrementa la probabilidad de errores contables, dificulta el análisis de ventas y limita la generación balances financieros precisos.

La ausencia de un sistema contable automatizado no solo afecta la eficiencia operativa y tiempo, sino también la transparencia de sus transacciones. Los clientes no reciben comprobantes digitales, lo que impacta negativamente en la confianza y profesionalización de estas pequeñas empresas. Además, esta carencia de registros formales dificulta el acceso a servicios financieros, ya que las entidades bancarias suelen requerir reportes de ventas confiables como parte de la evaluación para el crédito solicitado. Al no poder acceder a servicios financieros formales, muchos pequeños negocios se ven forzados a recurrir a prestamistas informales, exponiéndolos a tasas de interés elevadas y condiciones poco favorables. En consecuencia, los comerciantes del RTS, con recursos limitados, enfrentan barreras significativas para acceder a financiamiento, comprometiendo así su capacidad para crecer y desarrollarse en un entorno competitivo.

La falta de digitalización en sus operaciones no solo limita su sostenibilidad a largo plazo, sino que también los excluye de los beneficios que conlleva la adopción de tecnologías modernas, como el análisis de datos. Este problema no solo afecta a los comerciantes, sino que también tiene un impacto en la economía local, al frenar el desarrollo de un sector esencial en la actividad económica del país.

### Problemas Específicos.

* No se puede realizar un análisis de ventas.
* No es posible realizar balances contables del negocio
* No existe comprobantes digitales para el cliente.
* No existe comprobantes de ventas para solicitar financiamiento.
* Falta de indicadores de ventas por etapas o fechas.

## OBJETIVOS.

### Objetivo General.

Diseñar una aplicación móvil ligera, distribuida bajo una licencia GPL[[11]](#footnote-11), basada en Deep Learning[[12]](#footnote-12) con tecnología de reconocimiento de texto manuscrito (Handwritten Text Recognition, HTR), para los contribuyentes acogidos al régimen RTS, que permita la digitalización de recibos manuscritos, que sirva para un control de ventas y la generación de recibos digitales para el cliente. Lo que facilitaría un análisis en sus ventas y coadyuve en la toma de decisiones futuras.

### Objetivo Específico.

* Diseñar de un modelo de reconocimiento de texto manuscrito (HTR[[13]](#footnote-13)).
* Diseñar de funcionalidad para la generación de recibos digitales.
* Recolectar data set de texto manuscrito para el entrenamiento de la red neuronal.
* Transferir el entrenamiento de la red a una para ser utilizada en el móvil.
* Almacenar los datos en una base de datos para ser utilizadas en análisis de venta y control.
* Desarrollar de la aplicación móvil amigable e intuitiva.

## HIPOTESIS

Se plantea que el desarrollo e implementación de una aplicación móvil basada en técnicas de Deep Learning y reconocimiento de texto manuscrito (HTR), aplicada al contexto de los contribuyentes del Régimen Tributario Simplificado, será capaz de digitalizar notas de venta manuscritas con una precisión superior al 70%, generando recibos digitales estructurados que pueden ser extendidos a los clientes cuando se requiera, y permitiendo el almacenamiento de datos para facilitar un análisis básico de ventas.

## OPERACIÓN DE VARIABLES

**Variable dependiente:** aplicación móvil, resultante del proceso de diseño e implementación.

**Variable independiente:** Proceso de detección y digitalización de notas manuscritas.

**Variable interviniente:** Algoritmos de Deep Learning y técnicas de reconocimiento de texto manuscrito (HTR), que influyen en el rendimiento de la digitalización.

## JUSTIFICACIÓN.

### Justificación Social.

Con el desarrollo de la aplicación móvil con licencia GPL garantiza que se fomentará la inclusión tecnológica, promoviendo análisis en sus ventas y facilitando en el acceso a financiamiento. Al permitir el acceso abierto, pequeños comerciantes y comunidades con recursos limitados pueden beneficiarse de la herramienta sin preocuparse por costos de licencias privativas, fomentando una inclusión accesible a nuevas tecnologías.

### Justificación Económica.

La viabilidad económica se ha convertido en un factor predominante en el desarrollo de todo trabajo de investigación.

Al facilitar un control más eficiente de ventas y análisis, puede mejorar la gestión financiera de los pequeños negocios que pertenecen al régimen simplificado (RTS), esto incrementa la productividad, reduce las perdidas por errores contables, favoreciendo a una mejor toma de decisiones y estabilidad en estos negocios.

Esta solución permitirá reducir costos operativos al eliminar la necesidad de registros físicos, facilitando además el control de ventas, lo cual puede traducirse en una mejor planificación financiera, mayor acceso a créditos formales y una mejora en la sostenibilidad económica de los pequeños negocios acogidos al régimen RTS.

### Justificación Académica.

El proyecto contribuye al conocimiento en áreas como las redes neuronales, reconocimiento de texto manuscrito y aplicaciones móviles. Aportando una investigación aplicada. El uso de una licencia GPL en el desarrollo de esta aplicación tiene como objetico fomentar y permitir a otros desarrolladores e investigadores puedan utilizar, modificar y redistribuir el código de manera libre, esto permite a instituciones académicas, sectores públicos adoptar y adaptar esta tecnología sin costos adicionales.

## LIMITES Y ALCANCES

### Alcance.

La presente investigación contempla los siguientes alcances:

* Diseñar una aplicación móvil basado en Deep learning y HTR.
* Diseñar e implementar una tecnología innovadora para comerciantes minoristas que pertenecen al régimen simplificado de Bolivia.
* Utilizar los recursos en Deep Learning, CRNN, HTR, para la digitalización de recibos manuscritos.
* Generar recibos digitales.
* Almacenamiento de datos para análisis de ventas.
* Exportación de datos para uso en otras herramientas.
* Utilización de redes neuronales entrenadas previamente (transfer learning) para garantizar eficiencia en dispositivos móviles.

### Limites.

* La implementación está orientada únicamente al contexto boliviano, específicamente la paz Bolivia para negociantes en RTS.
* La aplicación está limitada a reconocer textos manuscritos hechos en recibos o notas de ventas.
* La aplicación se limita a escritura clara y legible. Escrituras excesivamente desordenadas o ilegibles pueden no ser reconocidas con precisión.
* La eficacia de la aplicación dependerá de la calidad de la cámara de celular.
* Aunque la aplicación funcione offline, la sincronización y exportación de datos podrían requerir acceso a internet.
* El almacenamiento de datos está limitado a la capacidad del dispositivo móvil.
* La eficiencia del reconocimiento de texto puede variar dependiendo de factores externos: ambiente, iluminación y el tamaño de trazo en la escritura.
* La implementación puede enfrentar resistencia por parte de comerciantes que prefieren métodos tradicionales.

## METODOLOGIAS

La metodología de esta tesis se enfoca en el diseño, desarrollo y validación técnica de una aplicación móvil para la digitalización de recibos escritos a mano, utilizando técnicas de Deep Learning y HTR. Se empleará el enfoque aplicado y experimental, acompañado de la metodología ágil Mobile-D para guiar el proceso iterativo de desarrollo de la aplicación.

### Metodología De Desarrollo.

Para el desarrollo del presente trabajo, se utilizará la metodología de desarrollo Mobile-D, es una metodología ágil, ya que es la más adecuada cuando se trata de plataformas móviles. Además, que permite responder rápidamente a los cambios que se puedan producir durante la etapa de desarrollo del proyecto, permitiendo la reducción de tiempos de producción (Amaya Balaguera, 2013).

El desarrollo de la aplicación seguirá principios de software libre, publicando todo el código fuente bajo una licencia GPL. Esto segura que cualquier persona pueda examinar, modificar y distribuir el software, promoviendo la transparencia y la posibilidad de mejoras continuas por parte de la comunidad.

### Metodología De La Investigación Científica.

La metodología científica, proporciona un conjunto de pasos y reglas lógicas por medio de las cuales es posible realizar el planteamiento de problemas y formular una hipótesis.

* **Observación**: Consiste en examinar atentamente los hechos y fenómenos que tiene lugar en la naturaleza y que pueden ser percibidos por los sentidos. La observación debe ser cuidadosa, exhaustiva y exacta.
* **Identificación de problema**: Después de la observación se plantea el cómo y porque del fenómeno observado, para la identificación del problema que se va estudiar.
* **Hipótesis**: Formular una hipótesis consiste en elaborar una explicación provisional de los hechos observados y de sus posibles causas. Es otras palabras es la solución preliminar ante el problema planteado, es una declaración que puede ser falsa o verdadera.
* **Experimentación**: Se realizan pruebas modificando la variable que intervienen en el proceso y comprobara si se cumple la hipótesis.
* **Resultados**: El análisis de los datos experimentales permite comprobar si la hipótesis era correcta y dar una explicación científica al hecho o fenómeno observado.

CAPITULO II

# MARCO TEORICO

## INTRODUCCION

En este capítulo presente se usarán los conceptos, métodos, herramientas que proporcionan las tecnologías actuales.

Las tecnologías móviles son consideradas un sector de crecimiento constante, lo que ha impulsado grandes innovaciones tecnológicas centradas en la telefonía móvil.

Hoy en día, El teléfono móvil, es un complemento cotidiano e indispensable por el amplio espectro de usos: escuchar música, enviar y recibir mensajes, llamadas, navegar en internet, entre otros.

A medida que avanza las tecnologías estas avanzan, estas se adaptan para cubrir muchos aspectos de la vida cotidiana, incluyendo diferentes áreas, tratando de integrar una parte de la sociedad que, por sus condiciones económicas, no siempre tiene tecnologías, un claro ejemplo de ellos es el RTS[[14]](#footnote-14), que podrían beneficiarse del uso de herramientas digitales adaptadas a sus necesidades.

En este contexto, el desarrollo de aplicaciones móviles accesibles y funcionales puede contribuir significativamente a la inclusión tecnológica, facilitando sus operaciones y promoviendo una gestión más eficiente de sus negocios.

## REGIMEN TRIBUTARIO SIMPLIFICADO RTS

El Régimen Tributario Simplificado (RTS), fue incorporado a la norma tributaria mediante el Decreto Supremo N°21521 de 13 de febrero de 1987, a la cual se le han realizado modificaciones en el transcurso de varias gestiones de gobierno. La última modificación realizada fue mediante el Decreto Supremo N°3698 en fecha 25 de octubre del 2018.

## INTELIGENCIA ARTIFICAL (IA)

La inteligencia artificial tiene por objeto que los ordenadores hagan la misma clase de cosas que puede hacer la mente (Boden, 2017).

Tipos de inteligencia artificial:

La inteligencia artificial se puede clasificar de varias maneras, las cuales se centran en una característica de la inteligencia, las más importantes de la IA serian:

**Según su funcionalidad o capacidad**

* Sistema de IA reactiva (Reactive AI): sistemas que reaccionan a estímulos del entorno, pero no son capaces de recordad ni usar experiencias pasadas para tomar decisiones, ejemplo: Deep Blue, computadora que gano al campeón de ajedrez Garry Kasparov 1997.
* IA con memoria limitada (Limited Memory AI): sistemas que usan datos históricos para tomar decisiones, ejemplo: vehículos autónomos.
* Teoría de la mente (Theory of mind AI): tipo de IA que busca comprender emociones, intenciones y pensamientos de los seres humanos, aun en desarrollo.
* IA autoconsciente (self-aware AI): tipo de IA hipotético que tendría el mismo novel de conciencia humana, con emociones y estados mentales.

**Según su nivel de autonomía**

* IA débil (Narrow AI o Weak AI): diseñada para realizar tareas específicas y no tiene conciencia ni capacidad para generalizar conocimientos. Ejemplo: asistentes virtuales como Siri, motores de recomendaciones como Netflix.
* IA fuerte (General AI): es una IA capaz de realizar tareas intelectuales que un ser humano puede hacer, aun no existente.
* Superinteligencia: IA que supera la inteligencia humana en todos los aspectos, incluyendo creatividad, resolución de problemas y toma de decisiones, aun siendo un concepto teórico.

**Según su enfoque técnico**

* Machine learning (ML): sistemas que aprenden de datos sin ser programados explícitamente, incluyen técnicas como el aprendizaje supervisado, no supervisado y por refuerzo.
* Deep learning: subconjunto del machine learning que utiliza redes neuronales artificiales con multicapa, llamada redes neuronales profundas.
* Natural language processing (NPL): enfocado en la interacción entre humanos y maquinas a través del lenguaje.
* Computer visión: permnite a las maquinas interpretar y entender imágenes y videos.
* Robótica inteligente: combinación de IA con robótica para crear maquinas que puedan realizar tareas físicas de manera autónoma.

## MACHINE LEARNING

Aprendizaje automatico (machine learning), la máquina se ve en los datos de entrada y las respuestas correspondientes, y descubra cuáles son las reglas debe ser. Un sistema de aprendizaje automático está capacitado en lugar de explícitamente programado. Se presenta muchos ejemplos relevantes para una tarea, y encuentra una estructura estadística en estos ejemplos que eventualmente permite que el sistema cree reglas para automatizar la tarea (Chollet, 2021).

El aprendizaje automático es una subparte de la inteligencia artificial que analizan grandes cantidades de datos estructurados e identifican patrones y correlacionan datos.

El aprendizaje automático se divide en tres categorías principales:

* Aprendizaje supervisado. - se define por su uso de conjuntos de datos etiquetados para entrenar algoritmos que clasifiquen datos o predigan resultados con precisión. A medida que se introducen datos de entrada en el modelo, éste ajusta sus ponderaciones hasta que se ha ajustado adecuadamente. Esto ocurre como parte del proceso de validación cruzada para garantizar que el modelo evite el sobreajuste o el infra ajuste (IBM International Business Machines Corporation, s.f.).
* Aprendizaje no supervisado. - también conocido como machine learning no supervisado, utiliza algoritmos de machine learning para analizar y agrupar conjuntos de datos no etiquetados (subconjuntos denominados clústeres). Estos algoritmos descubren patrones ocultos o agrupaciones de datos sin necesidad de intervención humana. La capacidad de este método para descubrir similitudes y diferencias en la información lo hace ideal para el análisis exploratorio de datos, las estrategias de venta cruzada, la segmentación de clientes y el reconocimiento de imágenes y patrones. También se utiliza para reducir el número de características de un modelo mediante el proceso de reducción de la dimensionalidad. El análisis de componentes principales (PCA) y la descomposición en valores singulares (DVE) son dos métodos habituales para ello. Otros algoritmos utilizados en el aprendizaje no supervisado son las redes neuronales, el k-medias y los métodos de agrupación probabilística (IBM International Business Machines Corporation, s.f.).
* Aprendizaje por refuerzo. - es un modelo de machine learning similar al aprendizaje supervisado, pero el algoritmo no se entrena con datos de ejemplo. Este modelo aprende sobre la marcha mediante el método de ensayo y error (IBM International Business Machines Corporation, s.f.).

## DEEP LEARNING

El deep learning es un subconjunto del machine learning que utiliza redes neuronales multicapa, llamadas redes neuronales profundas, para simular el complejo poder de toma de decisiones del cerebro humano (IBM International Business Machines Corporation, s.f.).

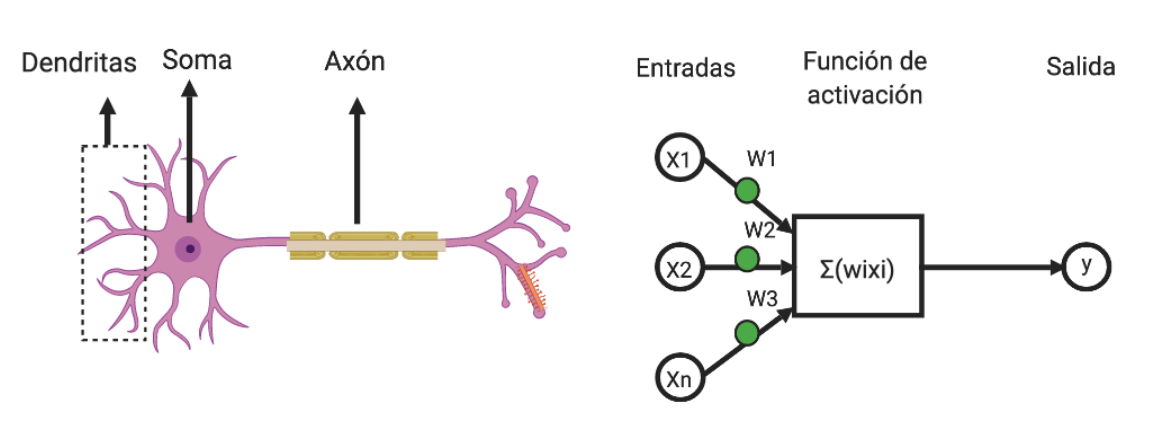
Para diferenciar entre el deep learning y el machine learning es la estructura de la arquitectura de red neuronal subyacente. Los modelos tradicionales de machine learning “no profundos (Deep learning)” utilizan redes neuronales simples con una o dos capas computacionales. Los modelos de deep learning utilizan tres o más capas, pero normalmente cientos o miles de capas, para entrenar los modelos.

Mientras que los modelos de aprendizaje supervisado requieren datos de entrada estructurados y etiquetados para obtener resultados precisos, los modelos de deep learning pueden utilizar el aprendizaje no supervisado. Con el aprendizaje no supervisado, los modelos de deep learning pueden extraer las características, los rasgos y las relaciones que necesitan para obtener resultados precisos a partir de datos brutos y no estructurados. Además, estos modelos pueden incluso evaluar y refinar sus resultados para aumentar la precisión.

### Redes Neuronales.

¿Qué es una neurona artificial? Dentro del campo de la Inteligencia Artificial son métodos de aprendizaje automático cuya finalidad es imitar los procesos biológicos de las redes neuronales de los organismos vivos (Hernández Orallo, Ramírez Quintana, & Ferri Ramírez, 2004).

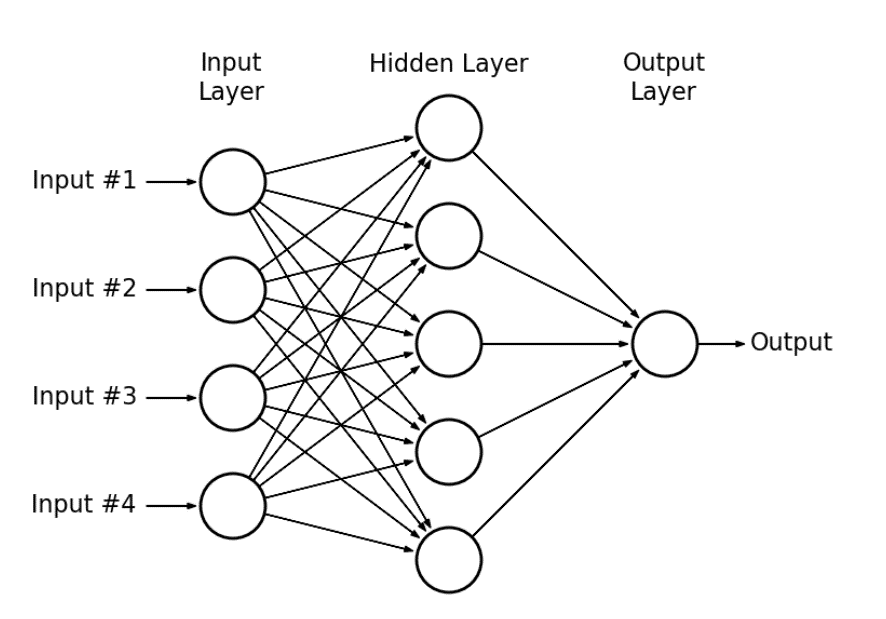
**Figura 1**  
*Introducción a redes neuronales*



*Nota*. – Comparación de como la base de una neurona biológica sirvió de base para la creación de una red neuronal, adaptado de Neurona vs. Perceptrón [fotografía], por Ulises García, 2019. Future Lab (<https://futurelab.mx/redes%20neuronales/inteligencia%20artificial/2019/06/25/intro-a-redes-neuronales-pt-1/>).

Hay tres partes normalmente en una red neuronal **Figura 1**: una capa de entrada, con unidades que representan los campos de entrada; una o varias capas ocultas; y una capa de salida, con una unidad o unidades que representa el campo o los campos de destino. Las unidades se conectan con fuerzas de conexión variables (o ponderaciones). Los datos de entrada se presentan en la primera capa, y los valores se propagan desde cada neurona hasta cada neurona de la capa siguiente. al final, se envía un resultado desde la capa de salida como se ve en Figura 2.

Figura   
Estructura simple de una red neuronal



*Nota*. – Perceptrón multicapa (MPL), las neuronas de cada capa se conectan con todas las neuronas de las siguientes capas, adaptado de CNN a Python [imagen], por Luciano Strika, 2019, kdnuggets (https://www.kdnuggets.com/2019/07/convolutional-neural-networks-python-tutorial-tensorflow-keras.html).

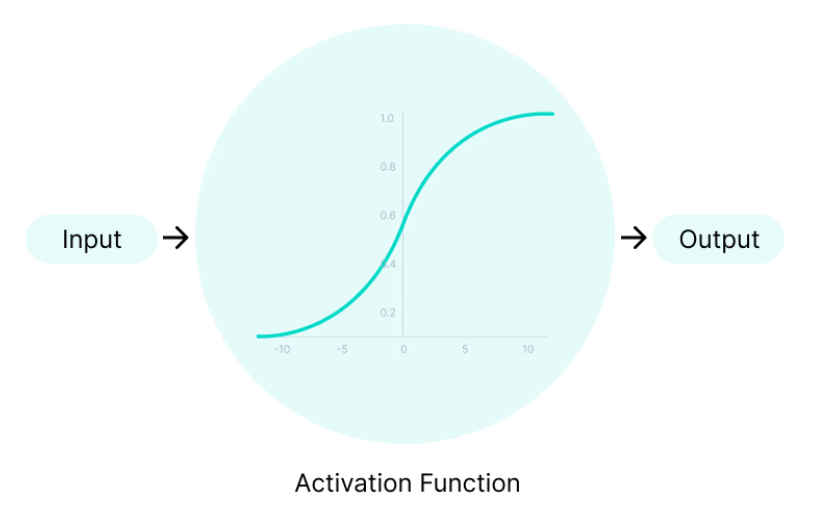
Como se ve en la **Figura 1**, puede formularse matemáticamente a la neurona básica de la siguiente manera: se cuenta con n entradas xi, en cada entrada es multiplicada por un peso wi, si existen n entradas se contará con n pesos, para finalmente calcular la suma total, como se muestra en la ecuación (1).

|  |  |  |
| --- | --- | --- |
|  |  | () |

### Funciones De Activación.

La función de activación es aquella que define la salida de una neurona como se ve en la **Figura 3**, si existe un estado de activación o no, es por ello que existen una variedad de funciones con distintas finalidades, a continuación, se presentan las funciones que son utilizadas con mayor frecuencia en los diversos problemas (Baheti, 2021).

**Figura 3**  
Función de activación



*Nota*. – Referencia de una función de activación. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Función de escalón o Función umbral de paso binario. - la función de escalón, es una de las funciones de activación más simpes utilizadas en redes neuronales. Su propósito principal es transformar la entrada de un nodo en una salida binaria, lo que permite tomar decisiones claras del tipo “si o no”, “activado o descativado”, o “verdadero o falso”.

Esta función compara el valor de entrada con un umbral previamente definido, si el valor de entrada es mayor o igual, la salida será 1, de lo contrario la salida será 0, como podemos observar en la **Figura 4** y matemáticamente expresarse como se muestra en la ecuación (2).

|  |  |  |
| --- | --- | --- |
|  |  | () |

**Figura 4**  
*Función de paso binario*



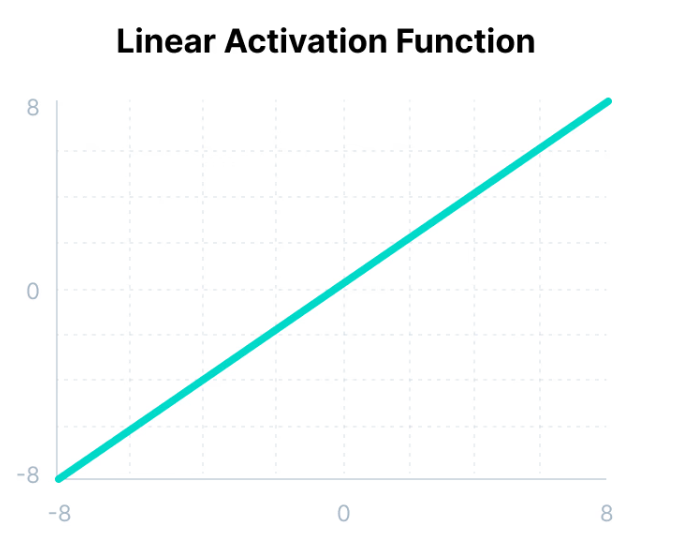
*Nota.* - Función de paso binario que depende de un valor umbral con la cual la neurona decide activarse o no. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Función lineal. – La función lineal es una de las funciones de activación más simples en redes neuronales. Su comportamiento consiste en devolver la misma salida que la entrada sin aplicar ningún tipo de transformación no lineal. En otras palabras, la activación neuronal es directamente proporcional al valor de entrada. Esto puede observarse en la **Figura 5** y se expresa matemáticamente en la Ecuación (3).

Esta función es útil en modelos donde se requiere una salida continua, como en problemas de regresión. Sin embargo, su limitación principal es que no permite introducir no linealidad en la red, por lo que no es adecuada para resolver problemas complejos que requieren aprendizaje de patrones no lineales.

|  |  |  |
| --- | --- | --- |
|  |  | () |

**Figura 5**  
*Función de activación lineal*

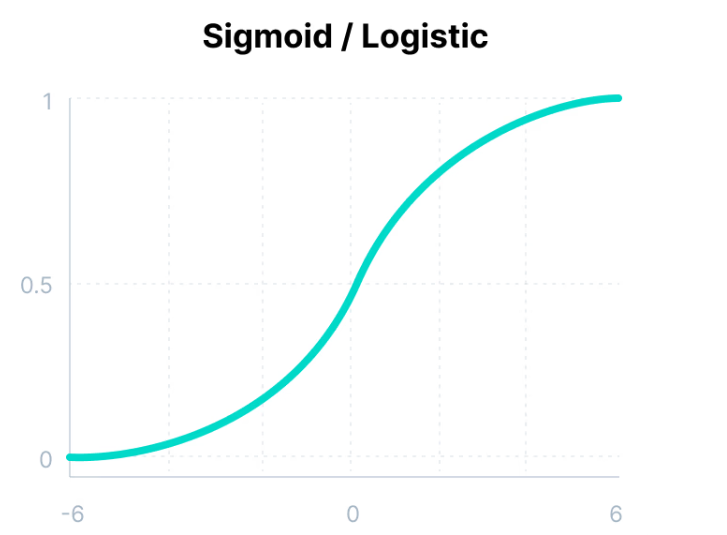


***Nota****.* – Función de activación también conocida como no activación o función de identidad. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Función sigmoidea o Logística. - La función es ampliamente utilizada en redes neuronales, especialmente en modelos donde se requiere una interpretación probabilística de la salida. Esta función presenta una curva en forma de "S" suave, como se muestra en la **Figura 6**, y su expresión matemática está representada en la Ecuación (4). Su valor de salida se encuentra en el rango de 0 a 1, aproximándose a 1 a medida que la entrada crece, y acercándose a 0 a medida que la entrada decrece. Esta propiedad la hace útil en tareas de clasificación binaria. Sin embargo, tiene desventajas como la saturación en sus extremos y el problema del desvanecimiento del gradiente, lo que puede dificultar el entrenamiento de redes profundas.

|  |  |  |
| --- | --- | --- |
|  |  | () |

**Figura 6**  
*Función de activación Sigmoid/Logistic*

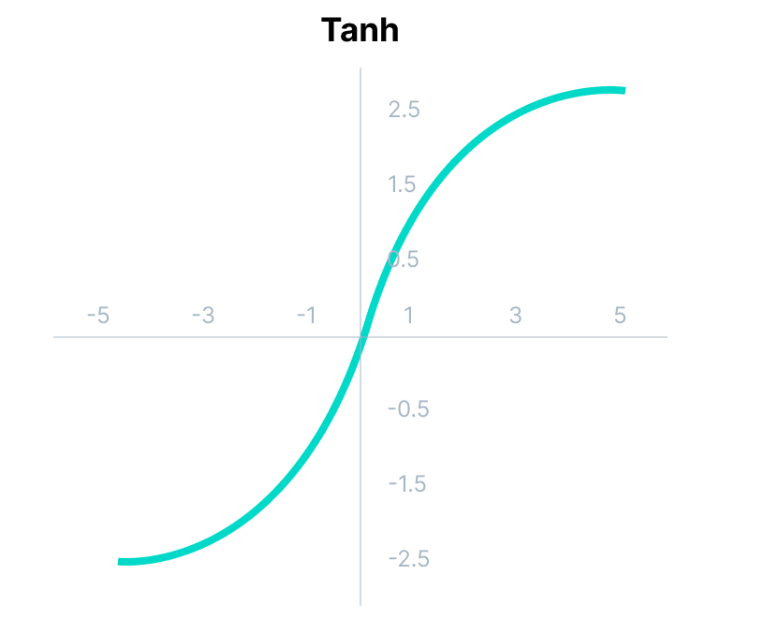


*Nota*. – Función de activación no lineal sigmoid/logistic toma cualquier valor real como valores de entrada y salida en el rango de 0 a 1. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Función de tangente hiperbólica (tanh). - La función de tangente hiperbólica, representada en la **Figura 7** y en la Ecuación (5), también tiene forma de “S”, pero a diferencia de la función sigmoide, su rango de salida esta va de -1 a 1. Esta característica le otorga una ventaja significativa al estar centrada en cero, lo que facilita el aprendizaje y la convergencia en modelos entrenados mediante retro propagación, mejorando así el desempeño en comparación con la función sigmoide en algunos escenarios de aprendizaje profundo.

|  |  |  |
| --- | --- | --- |
|  |  | () |

**Figura 7**  
*Función Tanh*

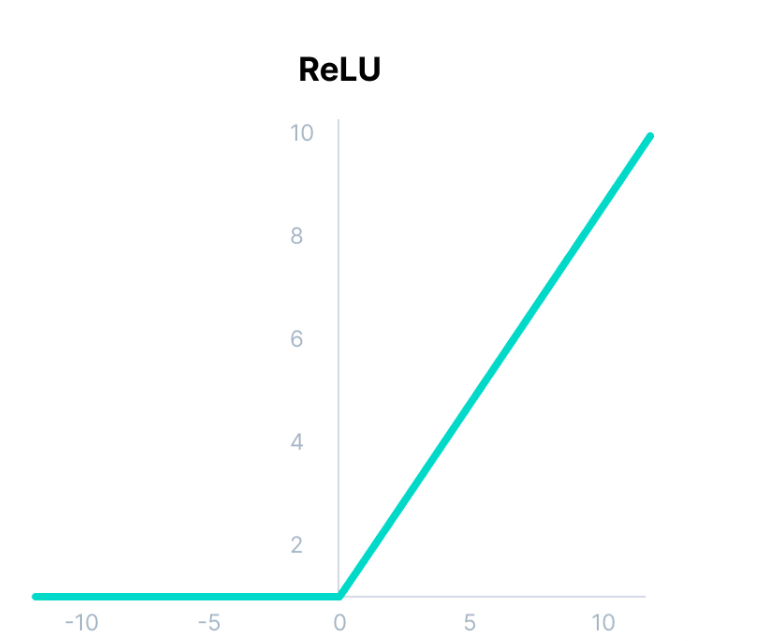


*Nota.* - Funcion Tanh similar a la función de actiivacion sigmoid/logistic con la diferencia en el rango de salida que admite -1 a 1. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Funcion ReLU (Rectified Linear Unit). – es una de las funciones de activación más utilizadas en redes neuronales, especialmente en redes convolucionales. Como se muestra en la **Figura 8** y en la Ecuación (6), esta función devuelve el mismo valor de entrada si este es positivo, y 0 si es negativo, introduce no linealidad sin saturación en el rango positivo y permite una retropropagación[[15]](#footnote-15) eficiente gracias a su derivada simple. Además, al no activar todas las neuronas simultáneamente, reduce el riesgo de co-adaptación entre nodos. Sin embargo, puede sufrir del problema de neuronas muertas[[16]](#footnote-16), por lo que se han desarrollado variantes como Leaky ReLU o ReLU Paramétrica para mitigar esta limitación.

|  |  |  |
| --- | --- | --- |
|  |  | () |

**Figura 8**  
*Función ReLu*



*Nota*. - Función ReLu que significa unidad lineal rectificada es una función derivada que permite retro propagación al mismo tiempo. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Función softmax. - La función softmax se utiliza principalmente en la capa de salida de redes neuronales destinadas a tareas de clasificación multiclase. Esta función transforma un vector de valores en un vector de probabilidades, donde cada elemento representa la probabilidad[[17]](#footnote-17) de pertenencia a una clase específica.

A diferencia de la función sigmoide, que retorna una única probabilidad, la función softmax devuelve una distribución de probabilidad sobre múltiples clases, haciendo que la suma total de todas las salidas sea igual a 1. Su expresión matemática puede observarse en la Ecuación (7). Gracias a esta propiedad, es ampliamente utilizada en combinación con funciones de pérdida como la entropía cruzada para problemas de clasificación supervisada.

|  |  |  |
| --- | --- | --- |
|  |  | () |
|  |  |  |

### Redes Neuronales Convolucionales (CNN).

Las redes neuronales convolucionales, conocidas como Convolutional Neural Networks (CNN o ConvNets), constituyen un tipo especializado de red neuronal artificial ampliamente utilizado en el campo del deep learning. Estas redes son particularmente efectivas para procesar datos con estructura espacial tridimensional, como imágenes (ancho, alto, canales), y han demostrado un rendimiento superior en tareas como clasificación de imágenes, reconocimiento de objetos, procesamiento de voz y análisis de audio (IBM International Business Machines Corporation, s.f.).

A diferencia de las redes neuronales tradicionales, las CNN se componen de una arquitectura jerárquica de capas que permite extraer características relevantes de manera automática y progresiva. Sus componentes principales son:

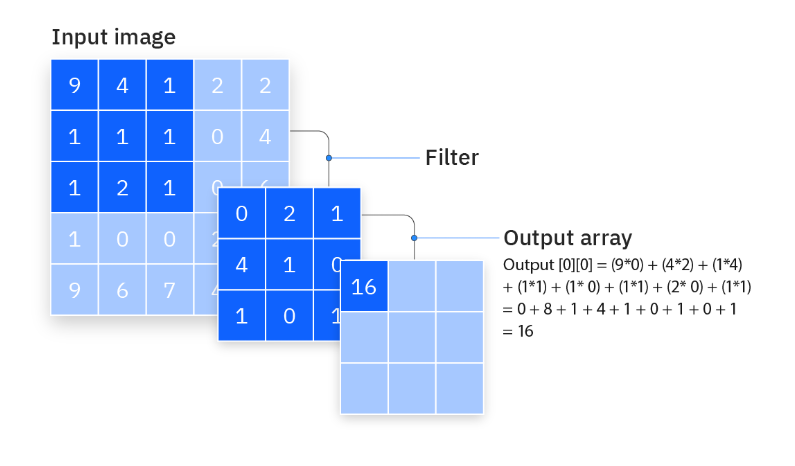
#### Capa Convolucional (Convolutional Layer)

La capa convolucional constituye el componente central de las redes neuronales convolucionales (CNN), ya que es en esta etapa donde se realiza la mayor parte del procesamiento computacional. Como se muestra en la **Figura 9**, esta capa emplea filtros o kernels, que son matrices bidimensionales de pesos entrenables, los cuales se deslizan sobre la imagen de entrada realizando una operación matemática denominada convolución.

Durante este proceso, el filtro examina pequeñas regiones locales de la imagen, permitiendo detectar patrones o características relevantes como bordes, esquinas, texturas o formas simples. Cada filtro puede aprender a detectar una característica distinta, y el resultado de aplicar dicho filtro genera un nuevo conjunto de datos denominado mapa de activación o feature map. Este mapa representa la activación de ciertas regiones de la imagen original, resaltando aquellas zonas donde la característica buscada está presente.

Después de aplicar cada convolución, es común que la red aplique una función de activación no lineal, como ReLU (Unidad Lineal Rectificada), sobre el mapa de características resultante. Esta operación no lineal permite que el modelo aprenda representaciones más complejas y profundas, contribuyendo a la capacidad de la red para modelar relaciones no lineales en los datos de entrada. La combinación de convolución y activación permite que las CNN extraigan progresivamente características de bajo nivel (como bordes) en las capas iniciales y características más abstractas (como formas o patrones complejos) en las capas posteriores.

**Figura 9**  
*Funcionamiento de una red neuronal convolucional*

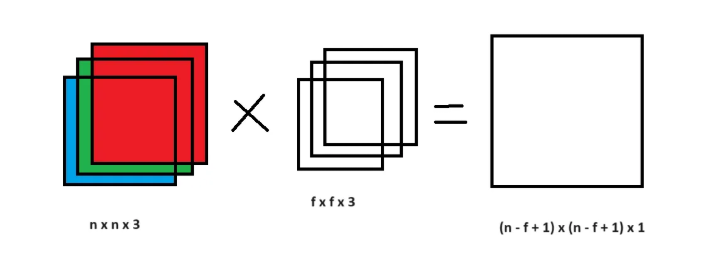


*Nota*. – Redes neuronales convolucionales se muestra el proceso de convolucion en las diferentes capas y filtros. Adaptado de Convolutional Neuronal Networs, por IBM International Business Machines, IBM (<https://www.ibm.com/think/topics/convolutional-neural-networks>). CC by 2.0

Es importante señalar que en una capa convolucional no se emplea un único kernel, sino múltiples kernels entrenables en paralelo. Como se muestra en la **Figura 10**, esto permite obtener múltiples mapas de características simultáneamente, cada uno capturando un aspecto distinto de la imagen. Es común que el número de estos filtros aumente a medida que se profundiza en la arquitectura de la red, siguiendo una progresión en potencias de dos (por ejemplo: 8, 16, 32, 64, 128, etc.), lo cual contribuye a la capacidad de la red para representar información compleja.

Además, para que la operación de convolución sea válida, el número de canales de entrada del filtro debe coincidir con el número de canales de la imagen de entrada. Por ejemplo, si la imagen de entrada tiene dimensiones 𝑊×𝐻×𝐷 (ancho, alto y profundidad o canales de color como RGB), entonces cada kernel deberá tener dimensiones 𝐹×𝐹×𝐷, donde 𝐹 representa el tamaño espacial (alto y ancho) del filtro. Esto asegura que la convolución se realice correctamente sobre toda la profundidad de la imagen, produciendo una salida bidimensional por cada filtro. En la **Figura 10** se muestra cómo se adapta esta operación en imágenes con múltiples canales, como las de color.

**Figura 10**  
*Operación de convolución*



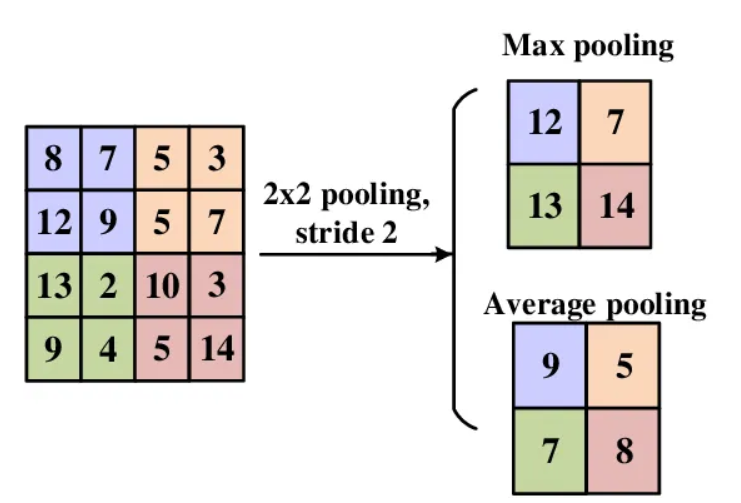
*Nota.* – Operación de convolución que ocurre en una imagen RBG, donde tanto la profundidad de la imagen de entrada y los kernel tienen que ser las mismas. Adaptación de Detailed understanding about convolution operation in a coloured image with the help of 3D kernels, por Abhishek Jain, 2024, Medium (https://medium.com/@abhishekjainindore24/detailed-understanding-about-convolution-operation-in-a-coloured-image-with-the-help-of-3d-kernels-16d698645988).

#### Capa De Agrupación O Pooling Layer.

También conocidas como capas de submuestreo o downsampling, permiten reducir progresivamente las dimensiones espaciales de los mapas de características (feature maps) generados por las capas convolucionales. Esta reducción tiene como principal objetivo disminuir el costo computacional, la cantidad de parámetros, y mitigar el riesgo de sobreajuste (overfitting), todo esto sin perder información clave de los datos de entrada.

A diferencia de las capas convolucionales, las capas de pooling no aprenden parámetros durante el entrenamiento, ya que su operación está determinada por funciones matemáticas fijas que se aplican de forma independiente en cada región de la entrada. La operación de pooling se realiza mediante un filtro que se desplaza sobre el mapa de características, dividiendo la imagen en pequeñas regiones y aplicando una función de agregación sobre cada una de ellas. Como se puede observar en la **Figura 11**.

**Figura 11**  
*Capa de agrupación*



*Nota.* – Adaptación de la capa de agrupación max y promedio, por Muhammad Shoaib Ali, 2022, Medium (https://medium.com/@muhammadshoaibali/flattening-cnn-layers-for-neural-network-694a232eda6a).

Los tipos de pooling más comunes son el **max pooling** y el **average pooling**:

En el **max pooling**, se selecciona el valor máximo dentro de cada región del filtro. Esta técnica es ampliamente utilizada por su capacidad para conservar las características más destacadas (como bordes o picos de activación), lo cual es especialmente útil para tareas de clasificación y detección.

En el **average pooling**, en cambio, se calcula el valor promedio de cada región. Aunque puede ser útil en ciertos contextos donde se busca suavizar las características, generalmente se considera menos efectivo que el max pooling en la práctica.

La principal ventaja del max pooling es que, además de reducir las dimensiones, permite conservar de manera efectiva las activaciones más importantes, lo que mejora la robustez del modelo frente a pequeñas variaciones o desplazamientos en la imagen. Esta propiedad se conoce como invarianza traslacional, y es clave en la capacidad de generalización de las CNN.

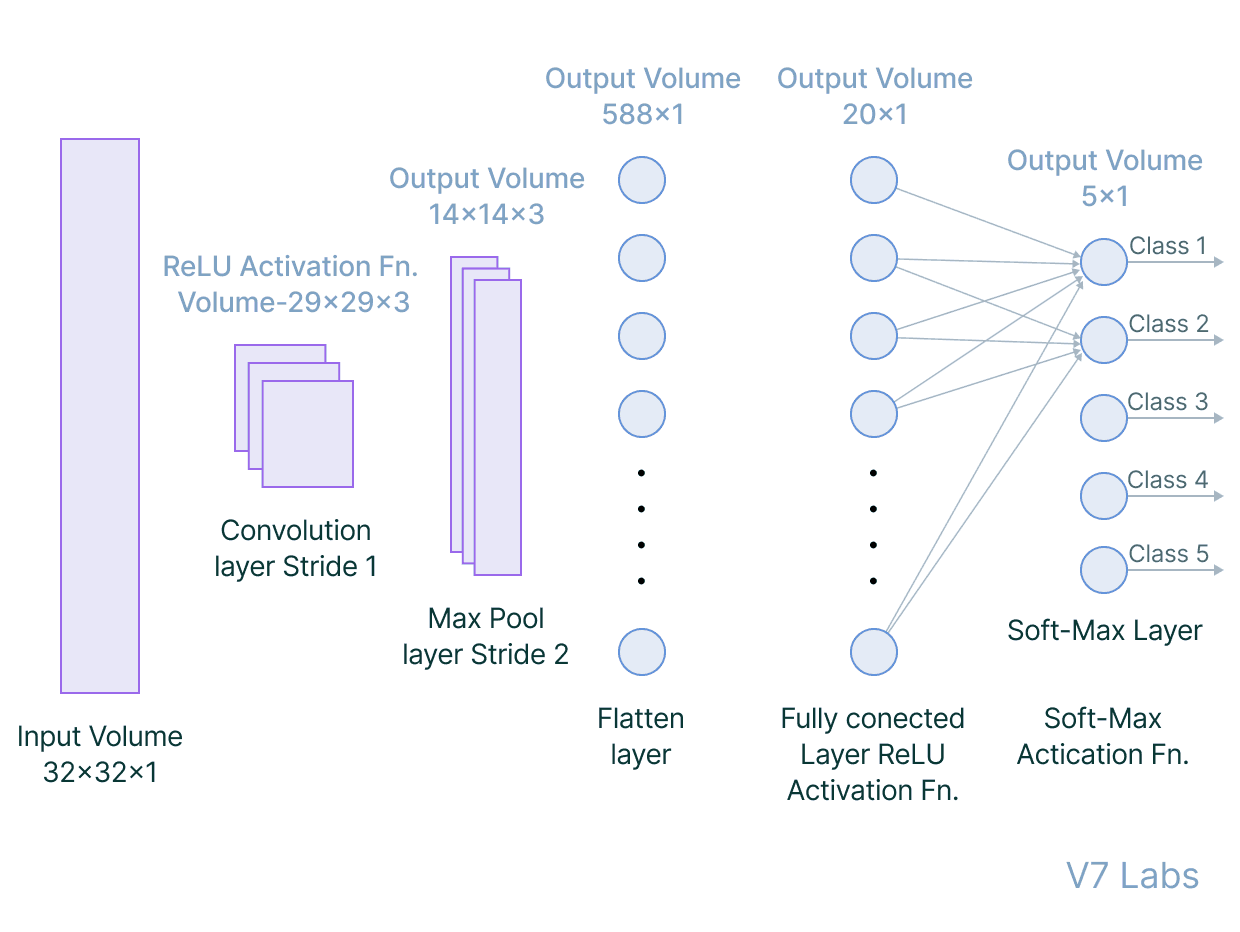
No obstante, se debe tener especial cuidado al seleccionar el tamaño del filtro de pooling y el stride (paso del filtro), ya que una configuración inadecuada puede eliminar demasiada información o no reducir suficientemente el tamaño de los datos, afectando negativamente el rendimiento del modelo. Un filtro de tamaño demasiado grande puede provocar una pérdida excesiva de información relevante, mientras que uno muy pequeño puede mantener un tamaño alto en los datos, generando mayor carga computacional y menor eficiencia.

#### Capa Totalmente Conectada o Fully-Connected Layer (FC).

La capa totalmente conectada, también conocida como fully-connected layer o simplemente FC, es una de las etapas finales en la arquitectura de una red neuronal convolucional (CNN). Su función principal es realizar la tarea de clasificación basándose en las características previamente extraídas por las capas convolucionales y de pooling.

En esta capa, cada nodo o neurona está conectado directamente con todos los nodos de la capa anterior como vemos en la **Figura 12**, lo que permite una integración completa de la información procesada hasta ese punto. A diferencia de las capas convolucionales, donde las conexiones son locales y limitadas a regiones específicas del mapa de activación, en la FC se utilizan conexiones densas que permiten combinar todas las características relevantes para generar una decisión final.

**Figura 12**  
*Capa totalmente conectada y procesos de capas anteriores*



Nota. – Adaptación de A Comprehensive Guide to Convolutional Neural Networks, por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/convolutional-neural-networks-guide).

La información que llega a esta capa suele estar ya transformada en un vector unidimensional (flattened vector), producto del proceso de aplanamiento (flattening) que convierte los mapas de activación bidimensionales en un solo arreglo de valores. Estos valores representan un resumen de alto nivel de las características más significativas presentes en la imagen original, como formas, texturas o patrones complejos.

En cuanto a la función de activación, mientras que en las capas anteriores se utiliza comúnmente ReLU (Rectified Linear Unit) para introducir no linealidades, en las capas fully-connected finales es habitual emplear la función softmax, especialmente en tareas de clasificación multiclase. Esta función transforma los valores de salida en probabilidades normalizadas entre 0 y 1, donde cada valor representa la probabilidad de que la entrada pertenezca a una determinada clase. La clase con la probabilidad más alta se considera la predicción final del modelo.

En redes neuronales profundas, puede haber una o varias capas FC entre las capas de extracción de características y la capa de salida. Estas capas adicionales permiten realizar combinaciones más complejas de las características aprendidas y pueden contribuir a mejorar la capacidad predictiva del modelo. Sin embargo, también incrementan el número de parámetros, lo que puede generar un mayor riesgo de sobreajuste si no se cuenta con un volumen de datos suficiente o si no se aplican técnicas de regularización como dropout.

En resumen, la capa fully-connected actúa como un clasificador final que interpreta y combina todas las características extraídas por las etapas previas de la red, y asigna una etiqueta o valor de salida al dato procesado. Es una parte fundamental del proceso de toma de decisiones en una CNN, y su correcta configuración influye directamente en el rendimiento global del modelo.

#### Capa Softmax o Softmax layer.

La capa Softmax es una capa de salida comúnmente utilizada en redes neuronales cuando se trata de resolver problemas de clasificación multiclase. Su principal función es transformar un vector de valores reales, que pueden estar en cualquier rango, en un vector de probabilidades normalizadas, en el que cada valor está comprendido entre 0 y 1, y la suma total de todos los valores es igual a 1. Esta propiedad permite interpretar la salida como la probabilidad de que la entrada pertenezca a una clase específica.

Matemáticamente, la función softmax aplica la siguiente operación a cada valor de entrada 𝑧𝑖, dentro de un vector 𝑧 como se ve en la ecuación (8):

|  |  |  |
| --- | --- | --- |
|  |  | () |

Donde 𝑛 es el número total de clases. Esta fórmula garantiza que los valores más altos en el vector original tendrán una mayor probabilidad, mientras que los valores menores recibirán proporciones correspondientemente más bajas, sin perder la relación relativa entre ellos.

La capa softmax suele colocarse al final de la red, justo después de una o varias capas totalmente conectadas (fully-connected layers). Es en esta capa donde se realiza la decisión final del modelo, ya que el resultado de la función softmax indica cuál es la clase más probable que corresponde a la entrada procesada. En tareas como reconocimiento de imágenes, clasificación de texto o detección de objetos, esta capa es crucial para traducir las representaciones abstractas internas de la red en una salida comprensible y accionable.

Una de las principales ventajas de utilizar softmax es que permite comparar directamente la confianza del modelo respecto a cada clase posible. Por ejemplo, si el modelo asigna una probabilidad del 0.95 a la clase "gato" y 0.05 a "perro", se puede inferir que el modelo está altamente seguro de su predicción. Esta propiedad también es útil en tareas donde se necesita establecer un umbral de decisión o interpretar el grado de certeza en entornos de clasificación con incertidumbre.

En resumen, la capa Softmax cumple un papel fundamental en redes neuronales para clasificación, ya que convierte la salida bruta de la red en una distribución de probabilidad interpretable, facilitando tanto la evaluación del rendimiento como la implementación de decisiones basadas en las predicciones del modelo.

#### Aplicación De La Redes Neuronales Convolucionales.

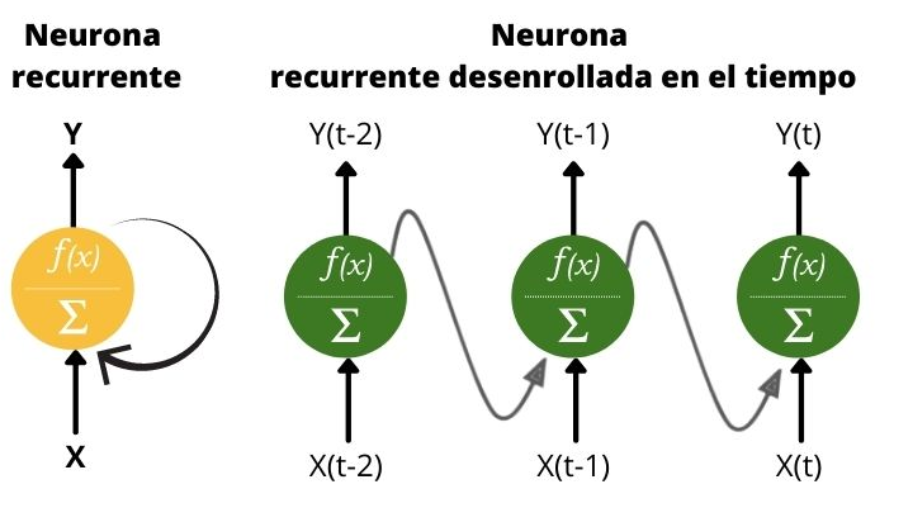
Las redes neuronales convolucionales son una parte muy importante en el Deep learning, ya que esta especialmente diseñada para procesar datos con estructura espacial, en imágenes, señales y secuencias, tiene un uso amplio en diversas áreas:

* Visión por computadora. - análisis de imágenes y videos, como detección medicas en radiografías como modelos de ResNet o DenseNet, reconocimiento de animales como iNaturalist, filtrado de contenido como en las redes sociales, detección de objetos como Yolo, Faster R-CNN son usados en Tesla, Waylo.
* Procesamiento de señales y series temporales, tenemos detecciones médicas, análisis de audio con Conv1D como aplicaciones Siri, Alexa, detección de audio como Shazam y identificación de patrones en series temporales de la bolsa de valores.
* Procesamiento de lenguaje natural (NPL) aunque los transformer dominan el NPL, las CNN aun como textCNN, en aplicaciones twiter X, fake news, traducción automática CNN + RNN ejemplo Google translate.
* Aplicaciones en dispositivos móviles y Edge computing. – modelos como MobileNet o EfficientNet usadas en apps fotográficas en sus mejoras de imagen y modo retrato usados ejemplo Google Pixel, o realidad aumentada como Snapchat filters, Pokemon Go y IOT internet de las cosas.

### Redes Neuronales Recurrentes (RNN).

La red neuronal recurrente o RNN se entrena con datos secuenciales o de series temporales para crear un modelo de machine learning (ML) y/o Deep learning (DL) como LSTM y GRU que puede hacer predicciones o conclusiones secuenciales basándose en entradas secuenciales como se ve en la **Figura 13**.

**Figura 13**  
*Neurona recurrente*



*Nota*. – adaptada de una red neuronal recurrente y como esta se desarrolla en ciclos, por Ruben Cañadas, 2021, abdatum (https://abdatum.com/tecnologia/redes-neuronales-recurrentes).

Al igual que las redes neuronales recurrentes utilizan datos de entrenamiento para aprender. Se distinguen por su memoria, ya que toman información de las entradas anteriores para que están influyan en la entrada y salida actuales.

Las salidas de las redes neuronales recurrentes como se mencionó dependen de los elementos anteriores dentro de la secuencia, tengamos en mente una frase y cada palabra forma una secuencia en la que el orden importa, se rastrea el contexto manteniendo un estado oculto en cada paso temporal. Se crea un bucle de feedback al pasar el estado oculto de un paso temporal al siguiente. El estado oculto actúa como una memoria que almacena la información sobre entradas anteriores, en cada paso temporal, la red neuronal recurrente procesa la entrada actual junto con es estado oculto del paso temporal anterior, permitiendo recordar puntos de datos anteriores y utiliza esa información para la salida.

Otra característica distintiva de las redes recurrentes es que comparten parámetros en cada capa de la red, mientras que las redes prealimentadas tienen diferentes pesos en cada nodo las redes neuronales recurrentes comparten el mismo parámetro de peso dentro de cada capa. Estos pesos todavía se ajustan a través de los procesos de retro propagación y descenso de gradiente facilitando el aprendizaje por refuerzo.

Las RNN utilizan algoritmos de propagación directa y retropropagacion en el tiempo (BPTT) para determinar los gradientes (o derivadas), lo que difiere ligeramente de la retropropagación tradicional, ya que es específico de los datos de secuencia. Los principios de la BPTT son los mismos que los de la retropropagación tradicional, donde el modelo se entrena a sí mismo calculando los errores de su capa de salida a su capa de entrada. Estos cálculos nos permiten ajustar y ajustar los parámetros del modelo adecuadamente. BPTT difiere del enfoque tradicional en que BPTT suma errores en cada paso de tiempo mientras que las redes prealimentadas no necesitan sumar errores, ya que no comparten parámetros a través de cada capa.

Una función de activación es una función matemática que se aplica a la salida de cada capa de neuronas de la red para introducir no linealidad y permitir que la red aprenda patrones más complejos en los datos. Sin funciones de activación, la RNN simplemente calcularía transformaciones lineales de la entrada, lo que la haría incapaz de manejar problemas no lineales. La no linealidad es crucial para aprender y modelar patrones complejos, particularmente en tareas como PLN, análisis de series temporales y predicción de datos secuenciales (IBM International Business Machines Corporation, s.f.).

#### Redes Neuronales Recurrentes Estándar.

Las Redes Neuronales Recurrentes estándar son una de las primeras arquitecturas diseñadas para el procesamiento de datos secuenciales. A diferencia de las redes neuronales tradicionales, que consideran cada entrada como independiente, las RNN estándar tienen conexiones que permiten que la información se propague de un paso de tiempo al siguiente. Esto significa que la salida en un instante determinado depende no solo de la entrada actual, sino también del estado oculto (hidden state) anterior, lo que introduce una forma de memoria temporal. Sin embargo, esta estructura básica tiene limitaciones importantes: cuando se trata de secuencias largas, estas redes tienen dificultades para aprender dependencias a largo plazo debido a problemas como el desvanecimiento o explosión del gradiente, lo que afecta negativamente el rendimiento en tareas complejas como la traducción o el modelado de lenguaje. (IBM International Business Machines Corporation, s.f.).

#### Redes Neuronales Recurrentes Bidireccionales (BRNN).

Las Redes Neuronales Recurrentes Bidireccionales (BRNN) son una extensión de las RNN estándar que abordan algunas de sus limitaciones. La idea principal es procesar la secuencia de datos en ambas direcciones, es decir, hacia adelante (del pasado al futuro) y hacia atrás (del futuro al pasado). Esto permite que la red acceda tanto al contexto anterior como al posterior en cada paso de tiempo, mejorando significativamente la comprensión del significado contextual. Este enfoque es especialmente útil en tareas como el reconocimiento de voz, análisis de sentimiento y procesamiento del lenguaje natural, donde la información futura puede ayudar a entender mejor una palabra o evento anterior. Las BRNN suelen utilizarse en combinación con LSTM o GRU para obtener un rendimiento aún mejor.

#### Memoria A Corto Plazo (LSTM).

Las LSTM (Long Short-Term Memory) son una variante avanzada de las redes neuronales recurrentes (RNN), introducidas por Sepp Hochreiter y Jürgen Schmidhuber en 1997, con el objetivo de superar una de las principales limitaciones de las RNN estándar: el problema del desvanecimiento o explosión del gradiente. Este fenómeno dificulta el aprendizaje de relaciones de dependencia a largo plazo en las secuencias, especialmente cuando estas son muy extensas, como suele ocurrir en el procesamiento de lenguaje natural, análisis de series temporales y datos biométricos.

La innovación clave de las LSTM radica en su estructura interna de celdas de memoria, diseñada para preservar y controlar el flujo de información a través del tiempo. Esta celda de memoria está compuesta por tres compuertas fundamentales:

* La compuerta de entrada regula qué nueva información será almacenada en la celda.
* La compuerta de olvido decide qué información de la celda anterior debe ser descartada.
* La compuerta de salida determina qué parte de la información almacenada será utilizada como salida en el paso de tiempo actual.

Estas compuertas funcionan mediante funciones sigmoides y operaciones de producto punto, permitiendo a la red mantener, actualizar o desechar información de forma selectiva en cada paso temporal. Esta capacidad de control fino es lo que permite a las LSTM modelar dependencias a largo plazo sin que los gradientes se desvanezcan durante el entrenamiento.

Las LSTM son especialmente eficaces en tareas donde el contexto previo tiene un impacto determinante en las predicciones futuras. Por ejemplo:

* En traducción automática, la red debe recordar el sujeto al inicio de una oración para traducir correctamente el verbo al final.
* En reconocimiento de voz, la pronunciación de una sílaba puede depender de sonidos anteriores.
* En análisis de sentimientos, una negación inicial (“no”) puede cambiar completamente el sentido del texto posterior.
* En predicción de series temporales financieras o meteorológicas, el comportamiento pasado es crucial para hacer pronósticos precisos.

Además, las LSTM han demostrado un rendimiento sobresaliente cuando se integran en arquitecturas más complejas, como los modelos Encoder–Decoder con atención, utilizados en traductores automáticos modernos o modelos de resumen automático de textos. También se emplean en la detección de anomalías, generación de texto y en modelos de predicción de eventos futuros en distintas industrias.

En resumen, las LSTM son un pilar fundamental en el desarrollo de redes neuronales que necesitan retener información contextual a través del tiempo, y su diseño estructural continúa siendo una de las soluciones más efectivas y versátiles para el modelado de secuencias en el campo de la inteligencia artificial.

#### Unidades Recurrentes Bloqueadas (GRU).

Las GRU (Gated Recurrent Units) son una versión simplificada y eficiente de las LSTM. Propuestas por Kyunghyun Cho y colegas en 2014, las GRU combinan las funcionalidades de las compuertas de entrada y olvido en una sola compuerta de actualización, lo que reduce el número de parámetros sin comprometer significativamente el rendimiento. Esta arquitectura mantiene muchas de las ventajas de las LSTM, como el manejo de dependencias a largo plazo, pero con menor complejidad computacional y un entrenamiento más rápido. Gracias a estas características, las GRU son una opción popular en aplicaciones con recursos limitados o donde se requiere una respuesta en tiempo real, como en sistemas de recomendación, asistentes virtuales, y análisis de sentimientos.

#### Codificador-Decodificador RNN.

El modelo Encoder-Decoder basado en RNN es una arquitectura central en tareas de secuencia a secuencia (seq2seq), como la traducción automática, resumen de texto, y chatbots. Este modelo está compuesto por dos partes principales:

* El codificador (encoder), que recibe una secuencia de entrada (por ejemplo, una oración en español) y la convierte en un vector de estado interno (también llamado vector de contexto) que resume toda la información relevante.
* El decodificador (decoder), que toma este vector de contexto y genera una secuencia de salida correspondiente (por ejemplo, la traducción en inglés).

Tanto el codificador como el decodificador suelen estar formados por RNN, LSTM o GRU, y a menudo se combinan con mecanismos adicionales como el mecanismo de atención (attention) para mejorar la capacidad de enfoque en partes específicas de la secuencia. Esta arquitectura es clave para aplicaciones donde se necesita transformar una entrada secuencial en otra secuencia de diferente longitud o estructura).

### Yolov5.

YOLO (You Only Look Once) es un algoritmo de detección de objetos en tiempo, es una de las herramientas más eficientes en el campo de la visión por computadora. Su principio fundamental es realizar la detección de objetos mediante una sola pasada a través de una red neuronal convolucional, lo que permite identificar múltiples objetos dentro de una imagen, asignándoles coordenadas de localización (bounding boxes) y un valor de probabilidad o confianza asociado a su clase. Esta característica lo hace sumamente rápido y adecuado para aplicaciones en tiempo real como videovigilancia, conducción autónoma, sistemas de asistencia y análisis de imágenes médicas, entre otros.

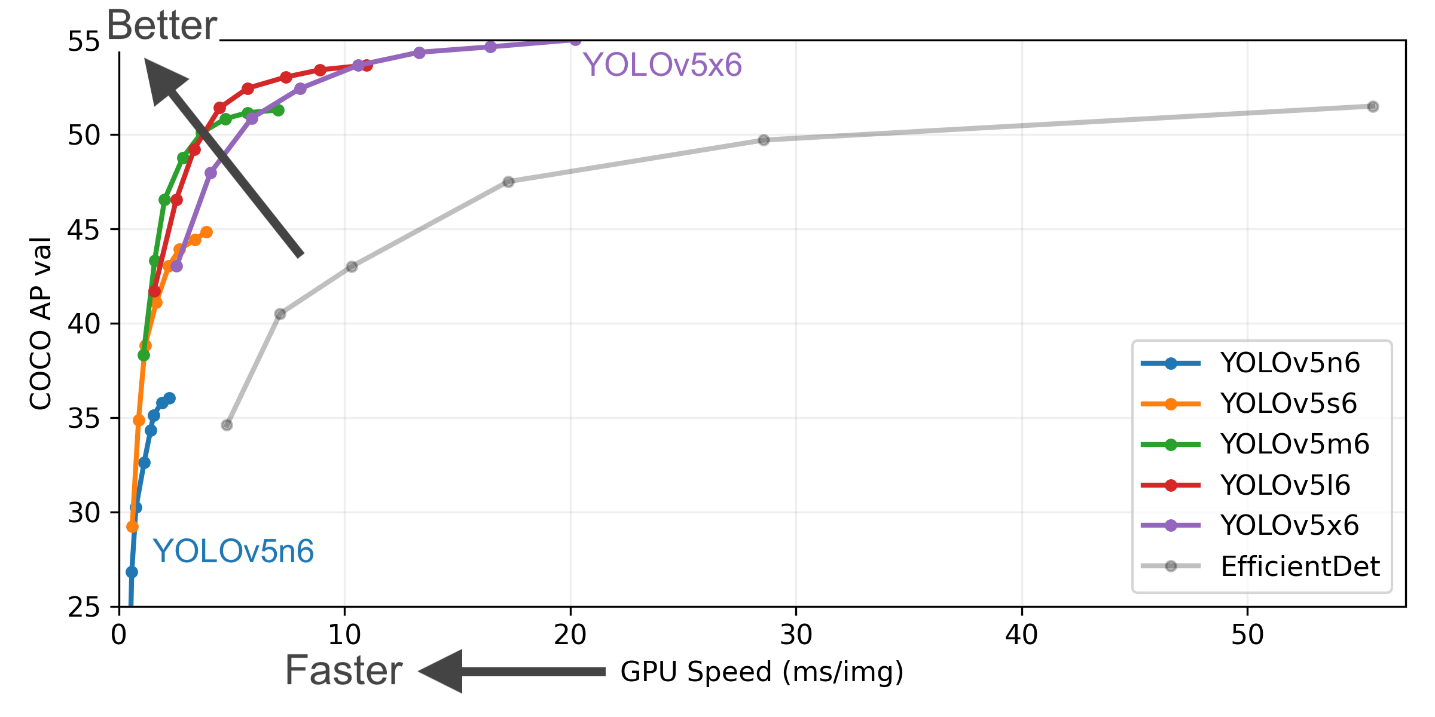
A lo largo de su evolución, YOLO ha tenido varias versiones significativas. En particular, YOLOv5, desarrollado por Glenn Jocher y lanzado en 2020 bajo la empresa Ultralytics, representa una de las versiones más populares debido a su facilidad de uso, eficiencia y precisión como observamos en la **Figura 14**  
*YoloV5 grafica de rendimiento***Figura 14**. A diferencia de sus predecesores, YOLOv5 fue implementado completamente en PyTorch, un framework de aprendizaje profundo ampliamente adoptado por la comunidad científica. El repositorio de YOLOv5 no es una continuación oficial de YOLOv4, pero ha ganado popularidad por su enfoque modular, soporte activo y rendimiento sobresaliente.

YOLOv5 cuenta con cuatro variantes principales:

* YOLOv5s (small): más rápido y ligero, ideal para dispositivos con recursos limitados.
* YOLOv5m (medium): un equilibrio entre velocidad y precisión.
* YOLOv5l (large): mejor precisión con un mayor coste computacional.
* YOLOv5x (extra-large): el más preciso, pero también el más demandante en términos de procesamiento y tiempo de entrenamiento.

Cada una de estas versiones está diseñada para adaptarse a distintos requerimientos, ofreciendo una progresión en el rendimiento y la precisión según la complejidad de la tarea y los recursos disponibles. En conjunto, estas características convierten a YOLOv5 en una herramienta versátil para la detección de objetos, tanto en imágenes estáticas como en secuencias de video.

**Figura 14**  
*YoloV5 grafica de rendimiento*



*Nota.* – Adaptacion de la gráfica de rendimiento de yoloV5 en sus diferentes versiones, donde COCO AP val denota la precisión media (mAP)sobre la intersección de unin (IoU) y la velocidad de GPU mide el tiempo medio de inferencia por imagen en el conjunto datos COCO, por Ultralitycs, 2020, Ultralytics (<https://github.com/ultralytics/yolov5>).

### Tesseract OCR.

Tesseract es un reconocido motor de reconocimiento óptico de caracteres (OCR) que permite la extracción de texto desde imágenes digitales, ya sean escaneos de documentos, fotografías de texto impreso o manuscrito, entre otros. Este sistema es capaz de identificar caracteres individuales, palabras y bloques de texto, facilitando la digitalización y estructuración de información contenida en documentos físicos.

Tesseract fue inicialmente desarrollado por los Hewlett-Packard Laboratories en Bristol, Reino Unido, y en Greeley, Colorado (EE. UU.) entre 1985 y 1994. Posteriormente, en 1996, se realizaron modificaciones para portarlo a sistemas Windows, y en 1998 se introdujeron cambios importantes en su estructura utilizando C++. En el año 2005, el proyecto fue liberado como código abierto por Hewlett-Packard, y desde 2006 hasta noviembre de 2018, Google asumió su desarrollo, mejorando significativamente su capacidad de reconocimiento y ampliando el soporte a múltiples idiomas y formatos.

Actualmente, Tesseract es considerado uno de los motores OCR de código abierto más robustos, y es ampliamente utilizado en tareas de extracción de datos, digitalización de documentos históricos, análisis de formularios, facturas, recibos y en proyectos que combinan visión por computadora con procesamiento de lenguaje natural (NLP).

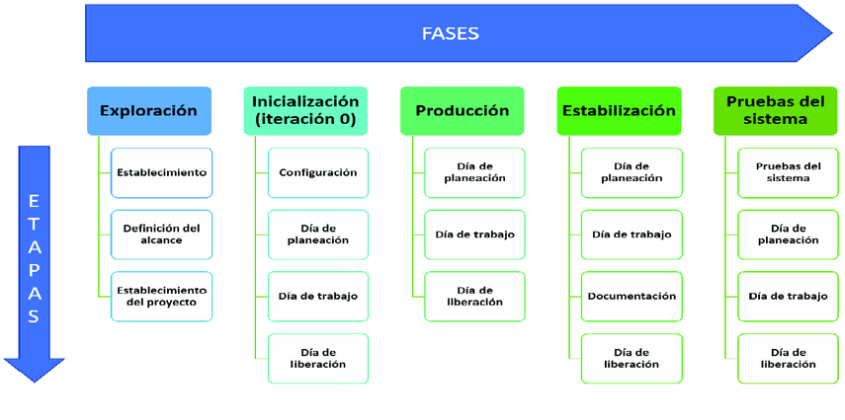
## METODOLOGIA MOBILE-D

El desarrollo de la aplicación se ha realizado utilizando la metodología Mobile-D, un enfoque ágil especialmente diseñado para la creación de aplicaciones móviles, donde los ciclos de desarrollo son rápidos y se requiere una alta adaptabilidad a los cambios. Esta metodología se basa en la combinación de principios y buenas prácticas provenientes de metodologías ampliamente reconocidas, como Extreme Programming (XP), Crystal Methodologies y Rational Unified Process (RUP).

Mobile-D está especialmente orientada a equipos pequeños y a entornos donde la entrega temprana y continua del producto es prioritaria. Además, promueve una estrecha colaboración entre los desarrolladores y los usuarios, permitiendo validar continuamente las funcionalidades y adaptarlas según las necesidades del proyecto.

Esta metodología se organiza en cinco fases clave, que permiten guiar de forma estructurada y ágil todo el proceso de desarrollo, desde la concepción de la idea hasta la entrega del producto final. Estas fases son: exploración, inicialización, producción, estabilización y pruebas, como se muestra en la **Figura 15**.

**Figura 15**  
*Metodología Mobile-D*



*Notas*. – Adaptación de la metodología Mobile-D, se muestra sus diferentes fases y etapas correspondiente para un mejor entendimiento, por Moramay Ramírez Hernández y Francisco León Pérez, 2023, Researchgate (<https://www.researchgate.net/figure/Figura-1-Metodologia-Mobile-D_fig1_368555118>).

Fases de la metodología Mobile-D:

* Exploración. - Esta fase inicial tiene como objetivo establecer una visión clara del proyecto. Se definen los conceptos fundamentales, el alcance general, los objetivos del sistema, las funcionalidades deseadas, y se identifican los principales riesgos. Además, se planifican las siguientes fases del proyecto. Esta etapa resulta clave para alinear al equipo de desarrollo con los requerimientos del cliente y los usuarios finales, asegurando que todos comprendan las expectativas del producto.
* Inicialización. - En esta etapa se identifican y organizan todos los recursos técnicos, humanos y de tiempo necesarios para el desarrollo de la aplicación. También se definen las herramientas, frameworks, bibliotecas y entornos que serán utilizados, asegurando una infraestructura sólida y adecuada. Se estructura el equipo de trabajo y se determinan los roles y responsabilidades, creando así un entorno preparado para comenzar el desarrollo iterativo.
* Producción. - Esta fase corresponde al núcleo del proceso de desarrollo, donde las funcionalidades del sistema se construyen de forma iterativa e incremental mediante ciclos de planificación-trabajo-liberación. Cada iteración comienza con la definición de requisitos específicos, seguida de la planificación de tareas concretas. Un principio clave en esta etapa es la práctica de Test-Driven Development (TDD), en la cual primero se escriben las pruebas automatizadas que verifican el comportamiento esperado de una funcionalidad, y luego se procede al desarrollo del código que debe cumplir dichas pruebas. Esta práctica mejora significativamente la calidad del software y permite detectar errores de forma temprana.
* Estabilización. - Una vez implementadas las funcionalidades principales, se inicia un proceso de integración y refinamiento del sistema. En esta fase se busca garantizar la coherencia y estabilidad del producto mediante pruebas de integración, validación del rendimiento y resolución de defectos. El objetivo es asegurar que todos los componentes funcionen correctamente en conjunto, y que el sistema como un todo cumpla con los estándares esperados de calidad.
* Pruebas. - Finalmente, se realiza una fase exhaustiva de pruebas donde el producto completamente integrado es evaluado frente a los requisitos funcionales y no funcionales definidos al inicio del proyecto. Esta etapa tiene como finalidad entregar una versión estable, robusta y funcional, eliminando todos los errores o inconsistencias detectadas. El sistema es validado por el cliente o los usuarios finales, asegurando que cumple con sus expectativas antes de su despliegue o lanzamiento.

La metodología Mobile-D, a través de estas fases bien definidas, permite manejar proyectos de desarrollo móvil de forma ordenada, flexible y centrada en la entrega continua de valor. La combinación de iteraciones ágiles, prácticas de calidad y validación constante con los usuarios convierte a Mobile-D en una alternativa eficaz para el desarrollo moderno de aplicaciones móviles.

CAPITULO III

# MARCO APLICATIVO

## INTRODUCCION

En el presente capítulo se describe el proceso de desarrollo de la aplicación móvil utilizando la metodología Mobile-D, la cual está específicamente orientada al desarrollo ágil de software para dispositivos móviles. Esta metodología propone una estructura organizada en fases que permiten abordar de forma iterativa e incremental el diseño, desarrollo y prueba de aplicaciones móviles. A lo largo de este capítulo se detallan cada una de las fases y etapas que han sido aplicadas en el desarrollo del sistema propuesto, orientado a facilitar el registro contable digital de los comerciantes minoristas a partir de notas de venta manuscritas.

## FASE DE EXPLORACION

En esta fase inicial se identificaron los requisitos preliminares necesarios para el desarrollo del sistema. Se estableció un plan general del proyecto, definiendo el alcance, los objetivos y las herramientas tecnológicas que se utilizarán. Asimismo, se identificaron los actores clave del sistema o grupo de interés (stakeholders), quienes cumplen roles esenciales tanto en la implementación como en el uso de la aplicación móvil.

### Establecimiento De Grupo De Interés.

* **Desarrollador**. - Es el responsable del diseño, implementación y mantenimiento del sistema. En este proyecto, el desarrollador cumple el rol de arquitecto de software, encargado de diseñar una aplicación móvil funcional, confiable y eficiente. Su labor implica la selección y configuración de las herramientas de desarrollo adecuadas (lenguajes de programación, frameworks, motores de inferencia, etc.), así como la integración de tecnologías de inteligencia artificial para el reconocimiento de texto manuscrito. Además, debe garantizar que el producto final cumpla con los requisitos y que se adapte al contexto y capacidades técnicas de los dispositivos móviles disponibles.
* **Comerciante minorista**. - Es el usuario final del sistema y representa al grupo de pequeños comerciantes que realizan ventas a través de notas manuscritas. Se caracteriza por operar con recursos limitados, tanto económicos como tecnológicos, lo que dificulta el acceso a herramientas digitales para la gestión contable. Estos comerciantes suelen registrar sus transacciones de manera manual, lo que conlleva errores, pérdida de información y dificultad para organizar sus finanzas o compartir datos de manera eficiente. El sistema propuesto busca atender esta problemática, facilitando la digitalización automática de sus notas de venta mediante una aplicación móvil sencilla, rápida y accesible.

### Requisitos Iniciales.

Diseñar una aplicación móvil ligera y distribuida bajo una licencia GPL[[18]](#footnote-18). Esta aplicación estará orientada a contribuir al proceso de digitalización contable de los comerciantes minoristas bolivianos acogidos al **RTS[[19]](#footnote-19)**, mediante el uso de técnicas avanzadas de Deep Learning y reconocimiento de texto manuscrito (Handwritten Text Recognition, HTR), para los contribuyentes acogidos al régimen simplificado (RTS), que permita la digitalización de recibos manuscritos, sirviendo para un control de ventas y ser extendidas como recibos digitales.

### Requerimientos Funcionales.

Los requerimientos funcionales establecen las funcionalidades esenciales que la aplicación debe cumplir para alcanzar sus objetivos. Estos son:

* Requerimiento funcional N°1.- La aplicación móvil debe permitir al usuario capturar una imagen de la nota de venta manuscrita, utilizando la cámara del dispositivo, para posteriormente iniciar el proceso de análisis y digitalización.
* Requerimiento funcional N°2.- La aplicación debe contar con un sistema de visión por computadora, que permita detectar automáticamente las áreas clave de la imagen (como fecha, nombre del cliente, productos, precios, totales), que serán procesadas por las redes neuronales para la lectura del texto.
* Requerimientos funcional N°3.- la aplicación móvil debe poder facilitar al usuario ser verificada antes de almacenarse.

### Requerimientos No Funcionales.

Los requerimientos no funcionales definen las características de calidad que debe poseer la aplicación para ser eficiente, accesible y útil en el entorno objetivo.

* Usabilidad: La aplicación debe tener una interfaz simple e intuitiva, adecuada para usuarios sin conocimientos técnicos. Debe permitir realizar el proceso completo (captura, verificación y almacenamiento) sin requerir capacitación especializada.
* Fiabilidad: La aplicación debe estar optimizada para funcionar en dispositivos Android de gama media, garantizando que el tiempo total de procesamiento de cada nota manuscrita no supere los 8 segundos, incluyendo análisis, reconocimiento y validación.
* Visualización: Durante el proceso de captura, se mostrará una previsualización en tiempo real, con herramientas básicas para ajustar brillo, contraste y enfoque. Los datos reconocidos (texto y números) serán presentados en una estructura tipo tabla, que refleje el formato original del recibo y que pueda exportarse a formatos compatibles con sistemas contables.
* Idioma: la aplicación móvil tendrá por defecto el idioma castellano.

### Definición Del Alcance.

Este proyecto posee alcances y limitaciones bien definidos, en función de los recursos disponibles, el contexto social del usuario objetivo y la viabilidad técnica del sistema.

#### Limitaciones.

* La implementación está orientada únicamente al contexto boliviano, específicamente la paz Bolivia para negociantes en RTS.
* La aplicación está limitada a reconocer textos manuscritos hechos en recibos o notas de ventas.
* La aplicación se limita a escritura clara y legible. Escrituras excesivamente desordenadas o ilegibles pueden no ser reconocidas con precisión.

#### Alcances.

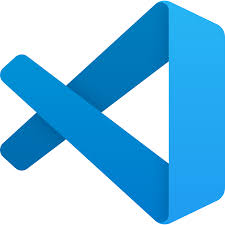
* Diseñar una aplicación móvil basado en Deep learning y HTR.
* Diseñar e implementar una tecnología innovadora para comerciantes minoristas que pertenecen al régimen simplificado de Bolivia.
* Utilizar los recursos en Deep learning, cnn, htr, para la digitalización de recibos manuscritos.
* Generar recibos digitales.
* Almacenamiento de datos para análisis de ventas.

### Establecimiento De Tecnologías.

A continuación, se describen las principales tecnologías y herramientas que se emplearán en el desarrollo de la aplicación móvil propuesta. Estas tecnologías han sido seleccionadas por su idoneidad para el procesamiento de imágenes, reconocimiento de texto manuscrito y desarrollo de aplicaciones móviles ligeras:

* VS Code (Visual Studio Code). - es un editor de código fuente desarrollado por Microsoft para diferentes sistemas operativos: Windows, Linux, MacOs, etc. Incluye soporte para depuración, control integrado de Git, respaldo de sintaxis, finalización inteligente de código, fragmentos y refactorización de código, siendo código fuente de licencia MIT, Freeware logo de la aplicación en la **Figura 16**.

**Figura 16**  
*Logo VsCode*



*Nota*. – Microsoft, 2025, VsCode (version 1.100.2) [Software de computadora], Visual Studio CODE (https://code.visualstudio.com/).

* Colab o Colaboraty. - es un servicio de Google que permite escribir y ejecutar código Python en un navegador, sin necesidad de ser instalado, podemos ver el logo en la **Figura 17**. Es una plataforma de aprendizaje automático, ciencia de datos y educación que ofrece acceso gratuito a recursos de computación con GPU´s y TPU´s.

**Figura 17**  
*Logo Colab*



*Nota*. – Google LLC, 2025, Collaboraty (version 2025-04-09) [Aplicación Web], Google Colab (https://colab.research.google.com/).

* Android Studio. – es el entorno de desarrollo integrado (IDE) oficial de Google para crear aplicaciones para Android podemos ver el logo en la **Figura 18**. Es una herramienta que se basa en intelliJ IDEA y ofrece funciones adicionales para mejorar la productividad en el desarrollo de aplicación Android.

**Figura 18**  
*Logo Android Studio*



*Nota*. – Google LLC, 2025, Android Studio (versión Merkat 2024.3.2) [Software de computadora], Android Studio (https://developer.android.com/studio?hl=es-419).

* Python. – es un lenguaje de programación de alto nivel, versátil y multiparadigma, interpretado y de código abierto, conocido por su sintaxis clara y legible podemos ver el logo en la **Figura 19**. Es ampliamente utilizado en diversas áreas como desarrollo web, ciencia de datos, aprendizaje automático y automatización.

**Figura 19**  
*Logo Python*



Nota. – Python Software Foundation, 2025, Python (version 8.3) [Software de computadora] (https://www.python.org/downloads/).

* YoloV5.- es un modelo de visión artificial para la detección de objetos, es una versión mejorada de los modelos Yolo anteriores y opera a alta velocidad de inferencia, lo que hace eficaz para aplicaciones en tiempo real podemos ver el logo en la **Figura 20**.

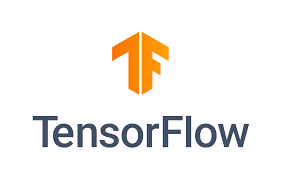
**Figura 20**  
*Logo YoloV5*



Nota. – Ultralitycs, 2025, YoloV5 (version s) [Modelo de visión artificial], Ultralitycs (https://github.com/ultralytics/yolov5/wiki).

* Tensor Flow. - es un sistema de computación numérica de código abierto de GoogleCLoud. Lo podemos encontrar en numerosos productos de Google Cloud ya que nos permite desarrollar algoritmos inteligentes podemos ver el logo en la **Figura 21**.

**Figura 21**  
*Logo TensorFLow*



Nota. – Google Brain, 2025, Tensorflow (version 2.12.0) [biblioteca de software], google (https://www.tensorflow.org/).

* Kotlin. – Es un lenguaje de programación moderno, multiplataforma y de código abierto, desarrollado por JetBrains **Figura 22**. Fue creado para ser interoperable con Java, y aunque inicialmente se enfocó en el desarrollo de aplicaciones de escritorio, ha ganado popularidad especialmente en el desarrollo de aplicaciones móviles Android.

**Figura 22**  
*Logo Kotlin*



Nota. – JetBrains, 2025, Kotlin (version 1.7.20) [lenguaje de programacion], Android Studio.

* Github. – es una plataforma en línea para alojamiento de código que permite a las desarrolladoras almacenar, compartir y colaborar en proyectos de software utilizado el sistema de control de versiones Git, podemos ver el logo en la **Figura 23**. Es una herramienta esencial para el desarrollo de software, ya que facilita la gestión de versiones, la colaboración entre equipos y la distribución de código fuente.

**Figura 23**  
*Logo GitHub*



Nota. – Awesome LLC, 2025, Github [Control de versiones colaborativo], (https://github.com/).

## FASE DE INICIACION

La fase de iniciación tiene como objetivo definir y establecer los elementos fundamentales que guiarán el desarrollo del proyecto. Durante esta etapa se detallará la configuración del ambiente de desarrollo, se seleccionan las versiones de las tecnologías a utilizar. Además, se elaboran el diseño de la aplicación.

Como resultado de esta fase, se obtiene una visión clara del proyecto.

### Configuración Del Ambiente De Desarrollo.

Esta actividad es específicamente para el desarrollador de la aplicación móvil ya que tiene como propósito el de instaurar un ambiente idóneo para llevar a cabo el desarrollo, en esta actividad además el desarrollador puede ejecutar pruebas de los requerimientos.

* Tipo de proyecto: aplicación móvil
* Tecnologías: YoloV5, CRNN, TFlite, Python, Kotlin

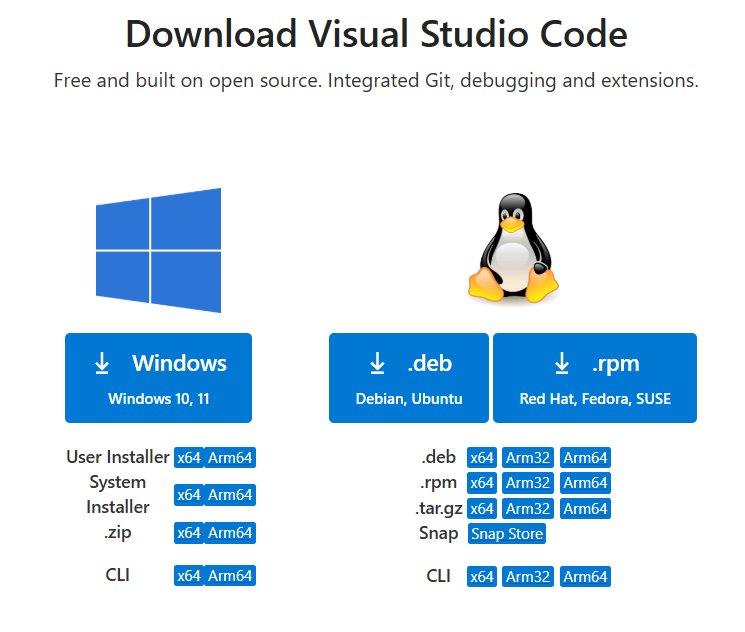
#### Preparación Del Ambiente.

La preparación del ambiente constituye un paso fundamental para garantizar que todas las herramientas y tecnologías necesarias para el desarrollo de la aplicación móvil funcionen de manera adecuada y sin conflictos. Esta etapa implica la instalación, configuración y verificación de compatibilidad de cada componente del ecosistema de desarrollo, con el objetivo de asegurar un entorno funcional, estable y coherente para la implementación del proyecto.

Para ello, se procede a la instalación y validación de las versiones correspondientes de las siguientes herramientas:

* Instalación de Visual Studio Code, descargar de la página oficial podemos ver una captura en **Figura 24**, a continuación, el link: <https://code.visualstudio.com/> para Windows 64 bits, instalar extensiones Python.

**Figura 24**  
*Imagen de descarga VsCode*



Nota. – Adaptación de la vista de la página de descarga Vscode editor de código, 2025, https://code.visualstudio.com/download

* Creación de una cuenta Github link a continuación <https://github.com>, se crea un repositorio, el cual se usará para publicar la aplicación y todo el código concerniente al proyecto.
* Para utilizar Google Colab en el proceso de entrenamiento de redes neuronales, es necesario contar con una cuenta de Google. Esta plataforma permite ejecutar código en la nube de manera gratuita, facilitando el desarrollo de modelos de aprendizaje automático sin requerir una infraestructura local. Sin embargo, debido a sus políticas de uso, es importante optimizar al máximo el tiempo disponible, ya que el uso prolongado y continuo puede conllevar a restricciones temporales. En este proyecto se empleará la configuración con GPU Tesla T4, la cual está disponible en la versión gratuita de Colab y ofrece un buen rendimiento para el entrenamiento de modelos de redes neuronales.
* Para el desarrollo del proyecto, es necesario instalar Python en el ordenador. La instalación puede realizarse desde el sitio web oficial: <https://www.python.org/>. En este caso, se utilizará la versión 3.8.0, seleccionada por su alta compatibilidad con diversos entornos de desarrollo, especialmente en proyectos vinculados al ecosistema Android y herramientas como TensorFlow, TensorFlow Lite y otras bibliotecas relacionadas con inteligencia artificial.
* Para el desarrollo y prueba de la aplicación móvil, se utilizará Android Studio, el entorno de desarrollo oficial proporcionado por Google. Puede descargarse desde el sitio oficial: https://developer.android.com/. Esta herramienta permitirá compilar, depurar y ejecutar la aplicación directamente en emuladores o dispositivos físicos. El lenguaje de programación utilizado será Kotlin, debido a su integración nativa con Android y su creciente adopción como estándar para el desarrollo de aplicaciones en esta plataforma.

Tecnologías base de la aplicación:

* Fronted: kotlin
* Backend Local: almacenamiento JSON, SqLite
* DL: Tensorflow lite para CRNN, YoloV5
* Librerías: OpenCv, TFLite, YoloV5, TesseractOcr, Numpy, Pandas, Matplotlib, TensorFlow.

Herramientas:

* Entrenamiento: Google Colab, requiere internet / PC local
* Dataset: 1500 imágenes etiquetadas primera detección, segunda detección 6000 imágenes áreas, tercera detección palabras 24000 + dataset externos.
* Exportación: TFLite para ejecución en dispositivos móviles.

### Planificación De Fases.

**Tabla 1**  
*Diseño Mobile-D*

|  |  |  |  |
| --- | --- | --- | --- |
| **Fase** | **Iteración** | **Descripción** | **Semana** |
| Exploración | Iteración 0 | * Establecimiento del proyecto. * Establecimiento de grupos de interés. * Limitaciones y alcances * Establecimiento de tecnologías. |  |
| Inicialización | Iteración 0 | * Análisis y requisitos iniciales. * Selección de herramientas * Planificación de dataset y pruebas. |  |
| Producción | Iteración 1 | * Captura de las notas de ventas, para crear los dataset. * Creación de códigos en Python para generar dataset. |  |
| Iteración 2 | * Creación de códigos en Python para tratar las imágenes. * Detección de áreas en las notas ventas con YoloV5, mediante colab. |  |
| Iteración 3 | * Incremento de captura de imágenes para incrementar la confianza en la detección de áreas. * Mejorando el código Python, para etiquetar datos en YoloV5. * Detección de áreas en subsección en nota de venta en el área detalle |  |
| Iteración 4 | * Códigos Python para generar el dataset para el CRNN * Integración de CRNN para detección y extracción de texto manuscrito por áreas. |  |
|  |
| Iteración 5 | * Corrección semántica haciendo uso de diccionario de palabras. * Estructura de notas digitales mediante la posición detectadas por yoloV5 en la iteración anterior. |  |
| Estabilización | Iteración 6 | * Conversión y optimización de modelos YoloV5 a TFLite. * Conversión y optimización del modelo CRNN a TFLite. |  |
| Iteración 7 | * Diseño de aplicación en Kotlin * Implementación de TFLite optimizados en código. * Manejo de datos tras captura. |  |
| Finalización | Iteración 8 | * Versión instalable APK. * Pruebas de usabilidad. |  |

### Diseño De Aplicación.

La aplicación está diseñada con un enfoque modular y basado en capas, integrando tecnologías de Deep Learning (DL) para el reconocimiento de escritura manuscrita y la digitalización de recibos. El sistema combina componentes de detección de regiones, reconocimiento de texto y almacenamiento estructurado de datos.

Podemos aprecias los diferentes procesos simplificados en la **Figura 25**. La imagen es capturada mediante el móvil Android, esta luego es procesada por la primera capa YoloV5 que detecta las áreas generales, posterior a esa esta áreas la área detalle es enviada a otroa capa YoloV5 donde esta es subdividida en subsecciones, cantidad, detalle, precioUnit y precioTotal, celda por celda, para ser enviadas a la CRNN para la detección del texto manuscrito en la celda, el resultado es enviado un diccionario de palabras para la corrección de las palabras, y usando las posiciones devueltas por la segunda capa YoloV5 se organiza en posición, para su posterior almacenamiento en SQLite.

**Figura 25**  
*Diseño de aplicación*

|  |
| --- |
| Redes neuronales o el arte de imitar el cerebro humano – magiquo creamos  inteligencia |

## FASE DE PRODUCCION

En esta fase se lleva a cabo el desarrollo funcional completo de la aplicación móvil, aplicando todos los requisitos definidos en las etapas previas. A través de un enfoque iterativo y ágil, propio de la metodología Mobile-D, se implementan las funcionalidades esenciales y complementarias, buscando asegurar que el producto cumpla con los criterios de calidad establecidos.

### Iteración 1.

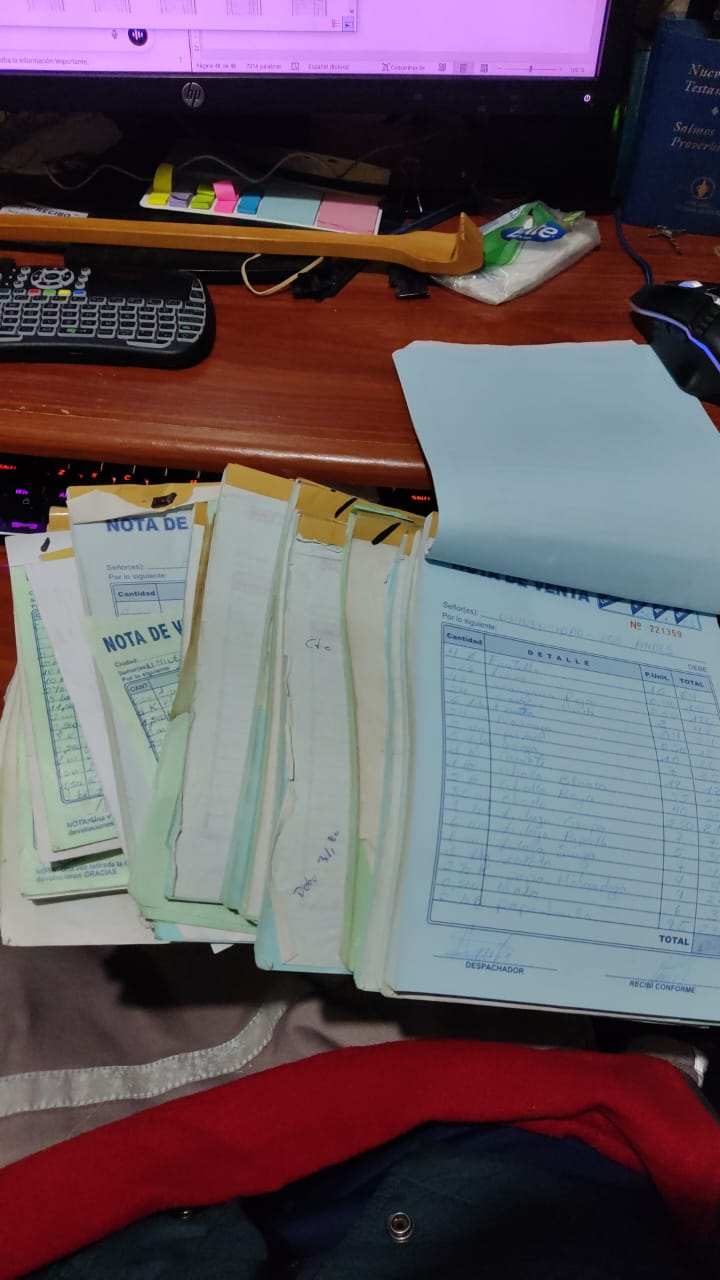
#### Captura De Notas De Venta.

En esta etapa se procede a la recolección de la mayor cantidad posible de notas de venta manuscritas, que servirán como base para el entrenamiento del modelo de reconocimiento de texto.

Las notas recolectadas serán digitalizadas y etiquetadas adecuadamente para su posterior uso en el proceso de entrenamiento. Este conjunto de datos representa una parte crucial del proyecto, ya que impacta directamente en la capacidad del sistema para generalizar ante nuevas entradas.

A continuación, en la **Figura 26**, se presentan algunos ejemplos de las notas de venta utilizadas en esta fase.

**Figura 26**  
*Notas de venta*

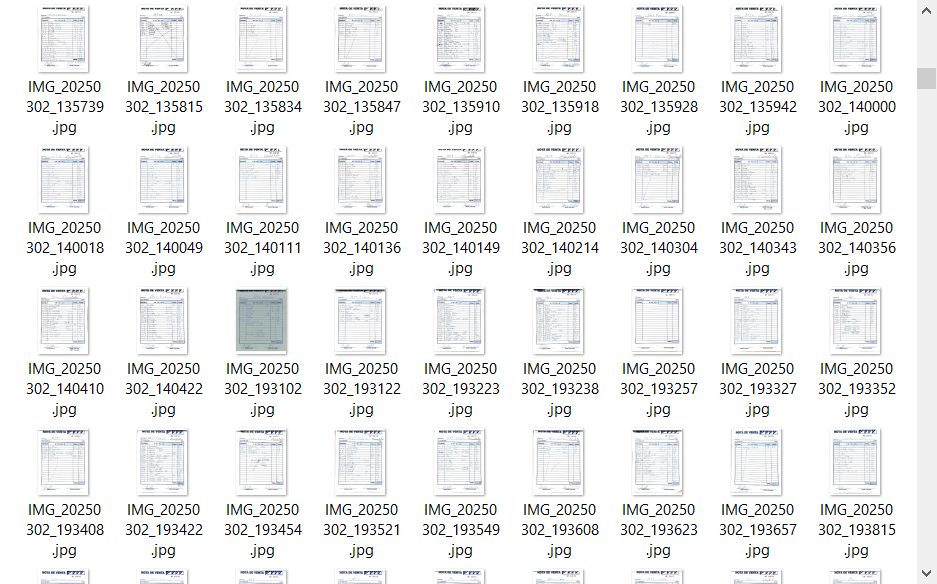


Utilizando la cámara de un teléfono celular, se capturaron fotografías de notas de venta manuscritas en diversos entornos y condiciones de iluminación, con el objetivo de obtener una amplia variedad de muestras representativas de situaciones reales. Esta diversidad en la calidad de las imágenes contribuye a que el modelo de reconocimiento de texto sea más robusto y adaptable a escenarios cotidianos.

Las imágenes capturadas fueron organizadas de manera sistemática en una carpeta específica, con el propósito de facilitar su procesamiento y posterior etiquetado. Esta organización permite una mejor gestión del dataset durante las fases de entrenamiento, validación y prueba del modelo de Deep Learning.

En la **Figura 27** se puede observar una vista general de la estructura de la carpeta que contiene las imágenes recopiladas.

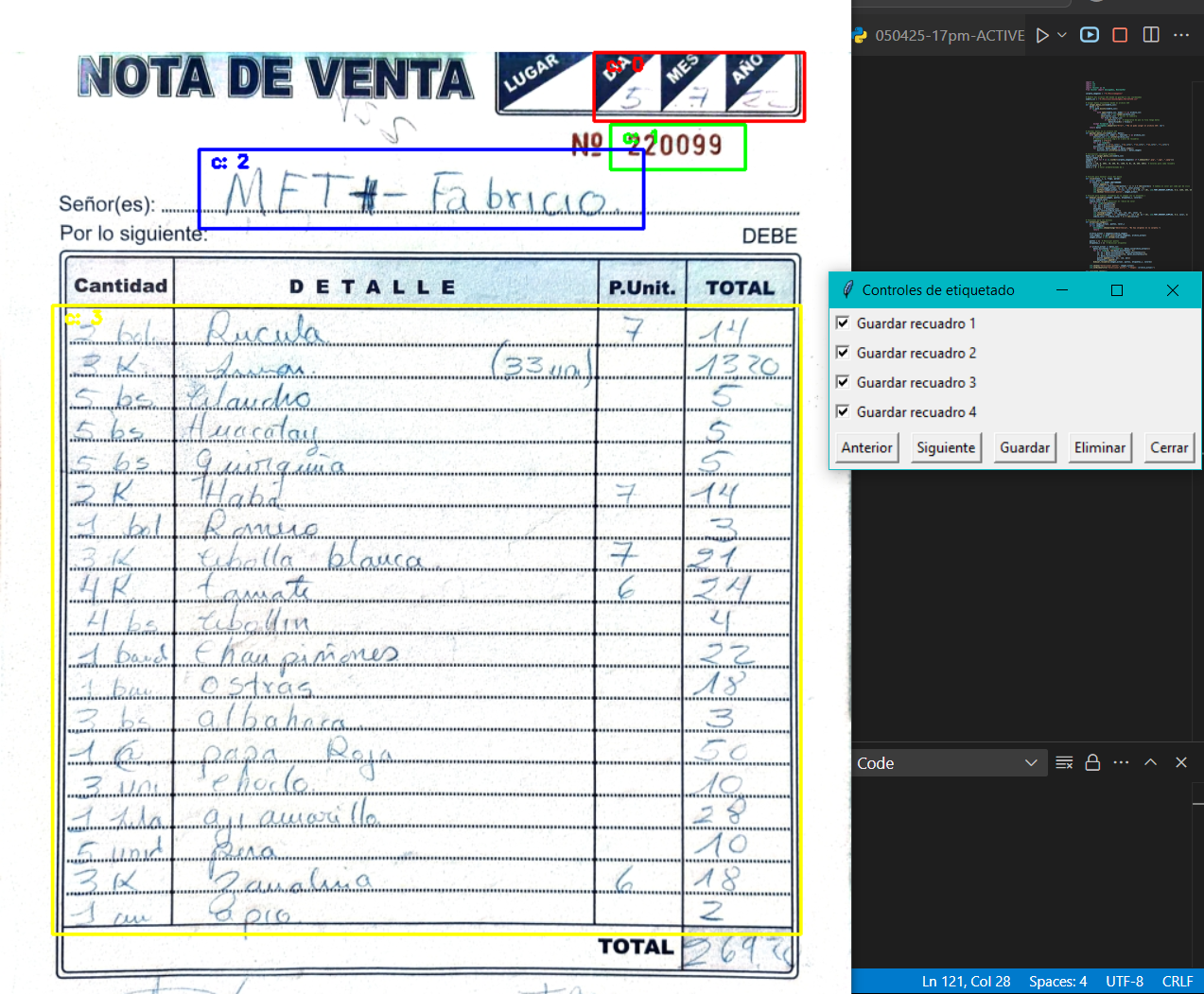
**Figura 27**  
*Notas de venta capturadas*



#### Códigos Python Para Tratar Las Capturas De Las Notas De Venta.

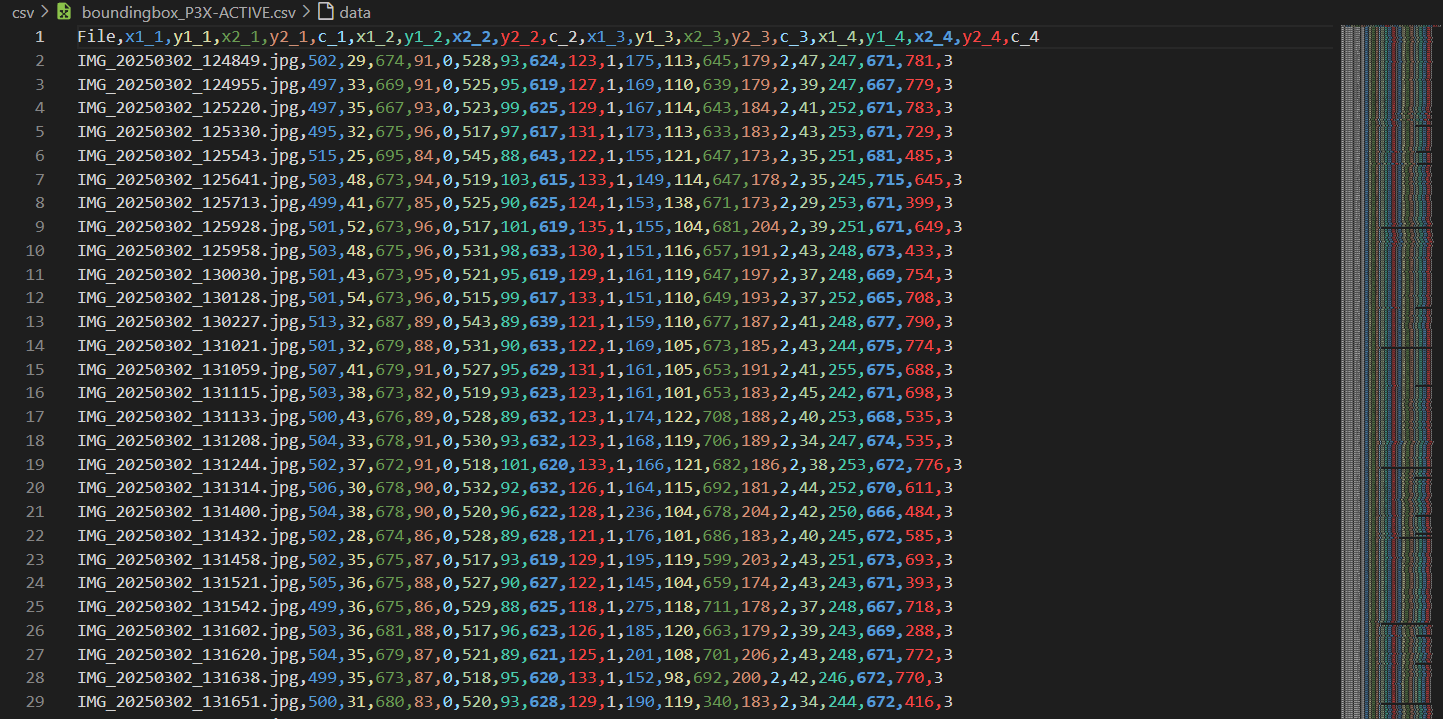
Se diseña un código que permita capturar las áreas de interes para el entrenamiento de YoloV5 para reducir consumo y potencia de computo como muestra en la **Figura 28**. Son cuatro áreas de interes, el de la fecha donde se hizo, el numero de serie que llevan impreso en la nota de venta, a quien se referia al nota de venta y el area mas importante en el cual se detalla la venta, la cantidad, nombre del producto, su precio unitario y su precio de cobro.

**Figura 28**  
*Notas de venta generación de bundingbox para el dataset*



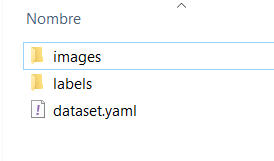
El código carga la carpeta de imágenes y crear un archivo csv con las coordenadas como se muestra en la **Figura 29**, que posteriormente se utilizara para generar áreas a detectar, para el entrenamiento en la deteccion de areas, con el respectivo nombre de la imagen.

**Figura 29**  
*boundingbox csv de áreas para el dataset*



Posteriormente a esto se diseño otro script[[20]](#footnote-20) en python para generar, organizar el dataset para el entrenamiento de la red neuronal como se aprecia en **Figura 30**.

**Figura 30**  
*formato de dataset de las notas de ventas*



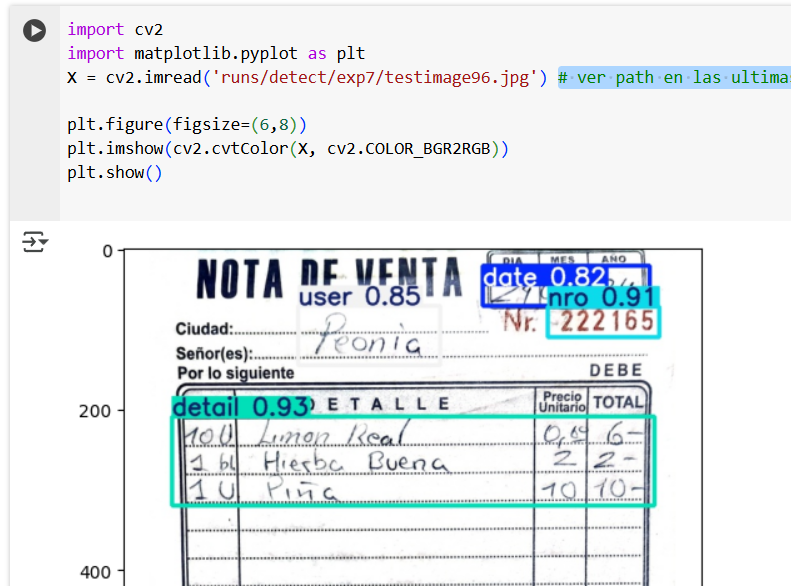
### Iteración 2.

#### Entrenamiento De Detección De Áreas Con Yolov5.

Para el entrenamiento del modelo de detección de áreas en notas de venta, se utilizó Google Colab como entorno de trabajo. Esta plataforma, ofrecida por Google, permite acceder gratuitamente a recursos computacionales avanzados, como GPUs Tesla T4, que superan significativamente la capacidad gráfica de un ordenador convencional. Gracias a estas prestaciones, es posible entrenar modelos de aprendizaje profundo con mayor eficiencia y en menor tiempo.

En esta etapa del proyecto se empleó YOLOv5. Como se muestra en la **Figura 31**, el modelo logra identificar las distintas áreas de interés en una nota de venta, resultado que se alcanza tras realizar un entrenamiento con una base de datos considerablemente amplia y bien etiquetada.

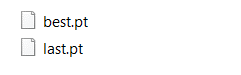
**Figura 31**  
*Detección con YoloV5 de nota de venta*



La calidad de las detecciones mejora directamente con el tamaño y la diversidad del conjunto de datos utilizado, así como con los ajustes realizados en los hiperparámetros del entrenamiento (como la tasa de aprendizaje, el número de épocas y el tamaño del lote). Si bien los resultados actuales son satisfactorios, se plantea la posibilidad de seguir ampliando el dataset o experimentar con diferentes configuraciones para incrementar aún más la precisión del modelo.

Una vez finalizado el entrenamiento, se exportan los pesos del modelo obtenidos. Estos archivos pueden utilizarse tanto para realizar pruebas locales como para continuar el proceso de refinamiento del modelo. En la **Figura 32** se ilustra cómo estos pesos se aplicarán para validar el desempeño del modelo fuera del entorno de entrenamiento.

**Figura 32**  
*Pesos de entrenamiento de YoloV5 áreas*

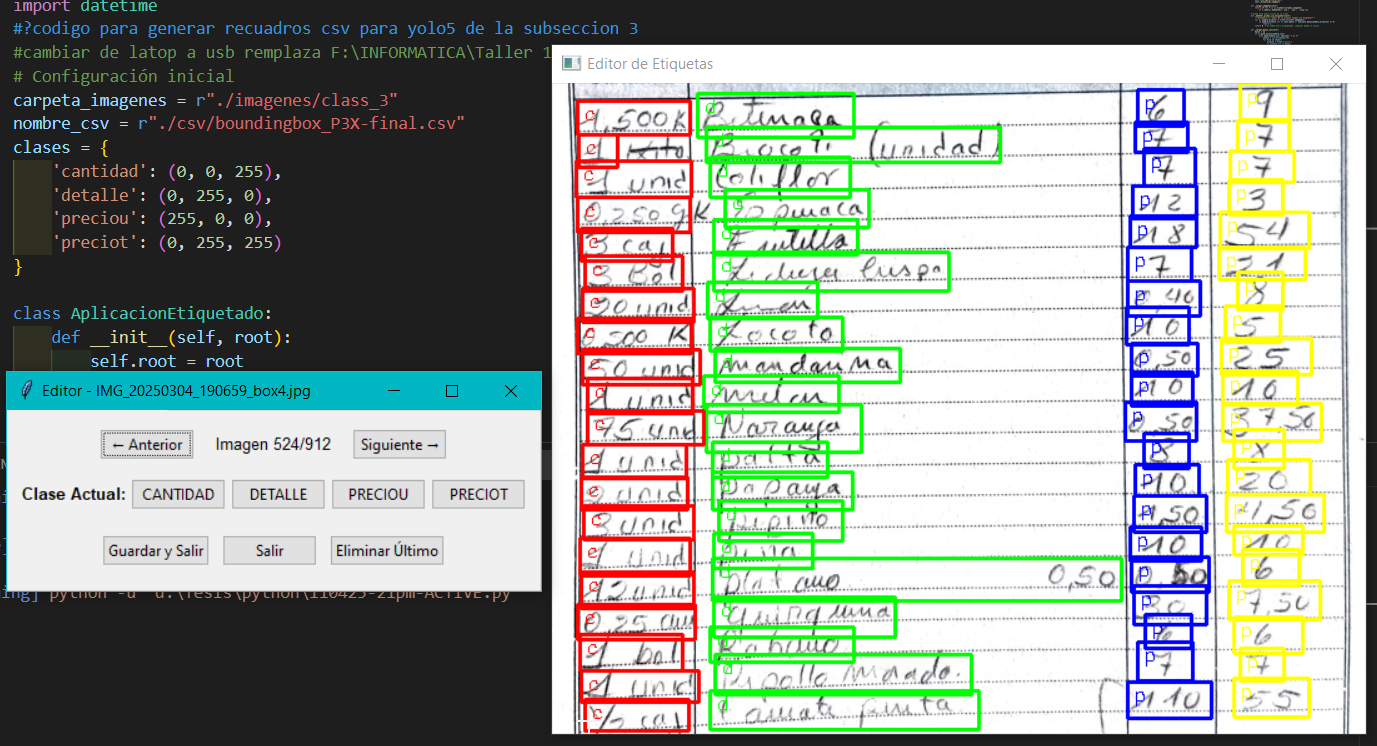


### Iteración 3.

#### Códigos Python Para Tratar Las Subsecciones Con Yolov5.

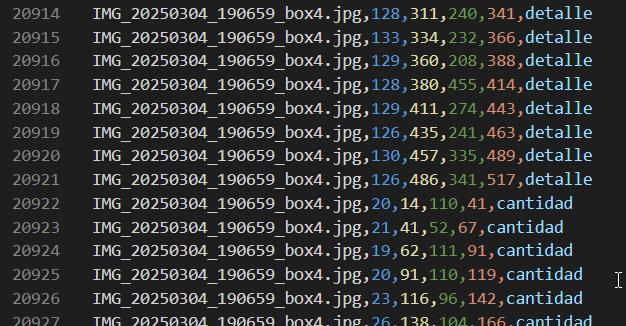
En esta seccion se diseñara un script, en el anterior iteracion se generaro boundiboxes para el entrenamiento de YoloV5, para esta operación se utilizaran la deteccion del area 3 que es la parte de detalle en la nota de venta manuscrita como se obserba en la **Figura 33**, a diferencia de la anterior script este tendra la diferencia tendra cuatro tipos de deteccion (cantitad, detalle, preciU y precioT) y tendra filas de una a veinte filas.

**Figura 33**  
*Notas de venta detección de celdas del área detalle*



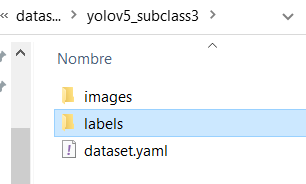
En la **Figura 34**, se observa las coordenadas que se capturaron para generar áreas especificas para el entrenamiento, con el respectivo nombre de la imagen.

**Figura 34**  
*Áreas de detalle en csv*



Una vez finalizado el conjunto de dataset, esta debe configurarse en un orden requerido para el entrenamiento como se ve en la **Figura 35** se ilustra cómo se organizara en carpetas separadas una donde tenga las imágenes, otra donde este los laves que son txt con las clases y coordenadas de coordenadas por imagen para el entrenamiento y el archivo \*.yaml que tiene las direcciones de la carpeta, clases a entrenar, tamaño de las imágenes que son necesarios para el entrenamiento.

**Figura 35**  
*formato de dataset de detalle*



#### Entrenamiento De Detección En Subáreas Con Yolov5.

Utilizando Colab para el entrenamiento en la detección de áreas con YoloV5 ya que al ser un servicio que permite Google con algunas configuraciones nos permite usar potencia grafica superior a la que contamos.

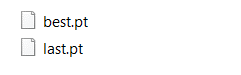
La detección es posible, pero tenemos una taza de confianza promedio aceptable como se ve en la **Figura 36**, debemos ampliar el dataset o cambiar la configuración en el aprendizaje, una vez ampliando y probando configuraciones se obtiene una confianza en la detección aceptable.

**Figura 36**  
*Detección de YoloV5 celdas detalle*



Una vez finalizado el entrenamiento, se exportan los pesos del modelo obtenidos. Estos archivos pueden utilizarse tanto para realizar pruebas locales como para continuar el proceso de entrenamiento del modelo. En la **Figura 37Figura 32**.

**Figura 37**  
*Pesos de entrenamiento de área detalle*



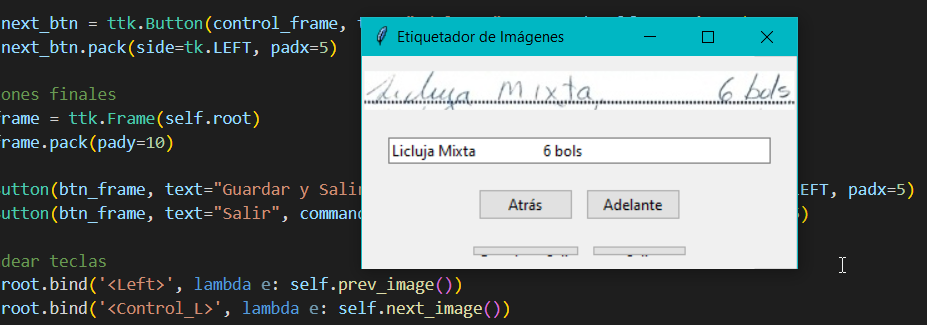
### Iteración 4.

#### Códigos En Python Para Generar El Dataset De Entrenamiento Para La CRNN.

En esta sección, utilizaremos la herramienta Vscode para desarrollar un script en Python que nos permitirá generar un Txt, a partir de las imágenes como se muestra en la (

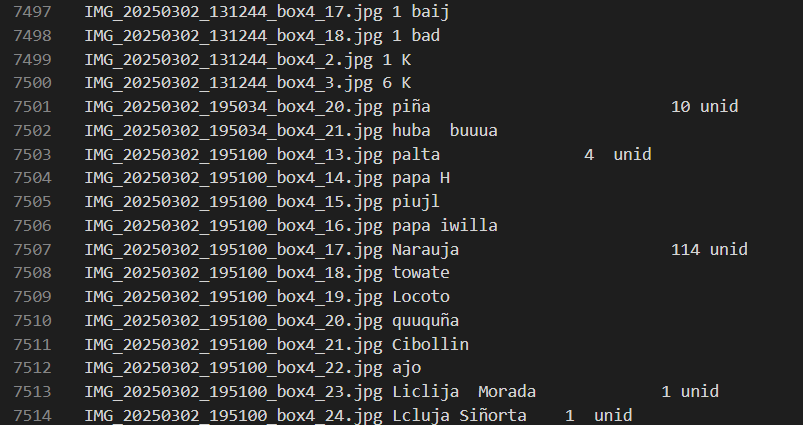
**Figura 38**), donde se identifica el valor que hay en las imagenes.

**Figura 38**  
*Captura de datos para el CRNN*



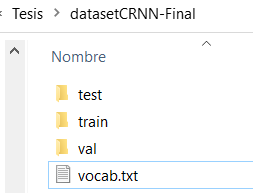
Los datos que se encuentran en el Txt como en la (**Figura 39**). Servirán para formar el dataset que se requiere para en entrenamiento del CRNN.

**Figura 39**  
*Archivo Txt con datos capturados*



Haciendo uso de VSCode diseñamos un script en Python para formar el dataset con las configuraciones necesarias para el entrenamiento como se ve en la **Figura 40**, se observa que consta de tres carpetas una de test, entrenamiento (train), val (validacion) y un archivo en txt que tendrá las letras, signos y símbolos que ayudará al entrenamiento de la CRNN.

**Figura 40**  
*Formato de dataset para entrenamiento*

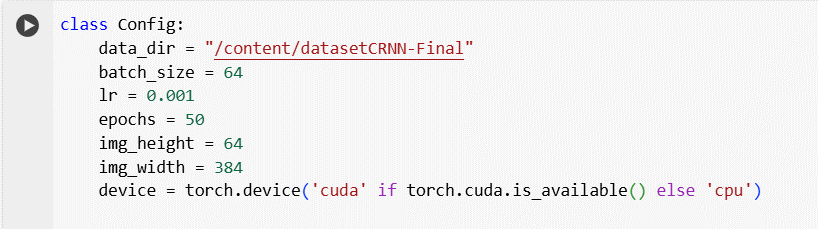


#### Entrenamiento De La CRNN ´Para La Detección De Letras Y Números Manuscritos.

En esta sección se realizará el entrenamiento de un modelo CRNN (Convolutional Recurrent Neural Network) para el reconocimiento de texto manuscrito, específicamente letras y números presentes en celdas que se detectaron previamente. Para ello, se define una clase de configuraciones con los parámetros para el funcionamiento optimo del modelo, como una ruta del dataset, tamaño de lote, tasa de aprendizaje, dimensiones de las imágenes y numero de épocas.

La parte inicial del entrenamiento son las configuraciones como vemos en la **Figura 41**, las partes más importantes de la configuración son: el batch\_size que es el tamaño de grupos de imágenes o lote de grupos de imágenes para el entrenamiento, lr (learning rate) que es la tasa de aprendizaje inicial, epochs las épocas que son las iteraciones de entrenamiento de lotes de imágenes son las configuraciones esenciales para el entrenamiento.

**Figura 41**  
*Configuración inicial para el entrenamiento CRNN*



El modelo CRNN posteriormente toma las imágenes las convierte en grises, recorta y ajusta para tener un tamaño unico, agregando un padding sin deformar la imagen y luego configurar la estructura del CRNN.

La clase CRNN representa el componente más crucial del proceso de entrenamiento del modelo podemos ver su estructura en la **Figura 42**, ya que define tanto la arquitectura como el flujo de información interno. En su constructor “\_\_init\_\_()”, se establece la secuencia de capas convolucionales para la extracción de características, una red recurrente LSTM bidireccional para el modelado secuencial, y una capa lineal final que proyecta las salidas a un espacio correspondiente al vocabulario definido.

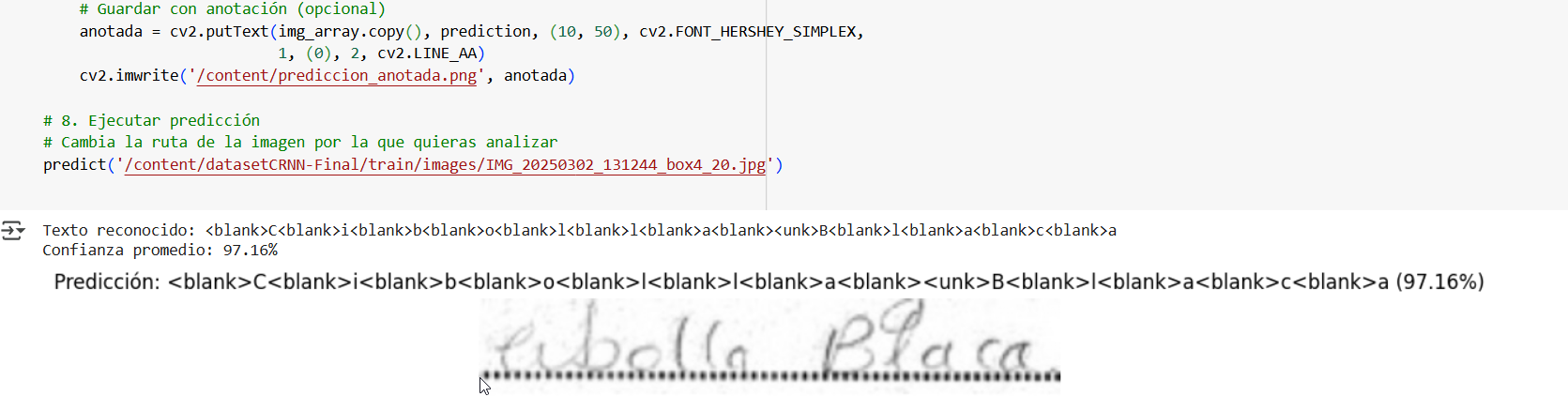
El método forward, por su parte, describe el recorrido que sigue una imagen de entrada a través de estas capas, transformándola progresivamente hasta obtener una secuencia de predicciones. Esta clase es ejecutada continuamente durante el entrenamiento, cada vez que se alimenta un lote de datos al modelo, y constituye la base sobre la cual se construye el proceso de aprendizaje supervisado mediante retropropagación.

**Figura 42**  
*Class CRNN arquitetura del modelo CRNN*



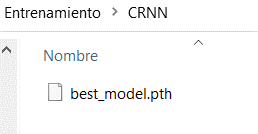
El modelo CRNN después de entrenarse se prueba y hace predicciones como se ve en la **Figura 43**. El nivel de confianza de las predicciones ronda entre 73 % a 97 % de confianza, utilizables para la aplicación móvil.

**Figura 43**  
*Prueba de detección de entrenamiento de palabras en celda*



Después del entrenamiento CRNN se extrae de colab los pesos como se ve en la **Figura 44** nos servirá para reentrenar o utilizar en la aplicación móvil.

**Figura 44**  
*Pesos de entrenamiento de la CRNN*



### Iteración 5.

#### Corrección Semántica Mediante Diccionario De Palabras En Python.

Dado que el modelo de reconocimiento de texto implementado (CRNN, Convolutional Recurrent Neural Network) puede producir resultados con errores ortográficos o semánticos debido a la calidad de las imágenes o la naturaleza confusa de la escritura manuscrita, se incorpora una etapa de corrección semántica posterior al reconocimiento.

Esta etapa consiste en la utilización de un diccionario de palabras previamente definido, implementado en Python, con el objetivo de validar y corregir los términos extraídos por el modelo. Para ello, se emplean algoritmos de similitud léxica, como la distancia de Levenshtein o el método get\_close\_matches de la biblioteca difflib, que permiten identificar la palabra más próxima en el diccionario a partir del texto detectado.

Cuando una palabra reconocida no se encuentra explícitamente en el diccionario, el sistema sugiere automáticamente una corrección basada en la palabra más similar y coherente desde el punto de vista semántico. Esta técnica permite no solo mejorar la precisión general del sistema de reconocimiento, sino también garantizar una mayor coherencia y legibilidad en los textos finales generados.

#### Estructuración De Notas Digitales Mediante Detección De Posiciones.

Una vez identificados y corregidos los textos mediante técnicas de OCR y corrección semántica, se procede a la estructuración lógica de la información contenida en la nota de venta manuscrita. Esta etapa tiene como objetivo organizar los elementos detectados en un formato estructurado tipo JSON, que permita representar digitalmente la nota de manera coherente y reutilizable.

Para ello, se emplea un modelo YOLOv5 previamente entrenado para detectar regiones clave del documento, tales como la fecha, número de serie, destinatario y el área de detalle de productos o servicios. Posteriormente, sobre la región del detalle detectado, se aplica un segundo modelo YOLOv5 especializado, encargado de identificar las celdas que conforman una tabla de cuatro columnas: cantidad, descripción del producto, precio unitario y precio total.

Una vez obtenidas las posiciones (coordenadas) de cada celda, se realiza un proceso de ordenamiento y asociación espacial, utilizando principalmente la coordenada vertical (eje Y) como criterio principal para agrupar elementos en filas, y la coordenada horizontal (eje X) para asignar los elementos a la columna correspondiente. De esta forma, se reconstruye lógicamente la tabla y se vinculan los textos reconocidos a su ubicación respectiva.

Finalmente, toda la información estructurada —incluyendo los datos generales y el contenido tabular— se convierte a un objeto JSON, lo cual permite su integración con otros sistemas, almacenamiento en bases de datos o visualización digital. Este formato garantiza una representación fiel y organizada de la nota de venta original.

## FASE DE ESTABILIACION

### Iteración 6.

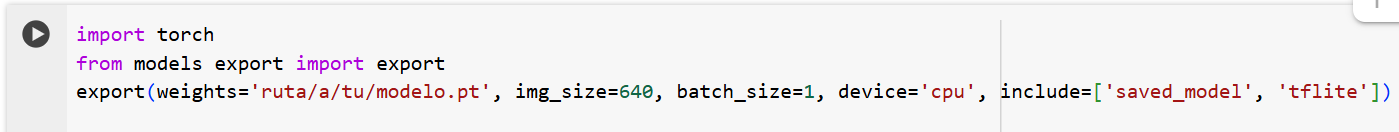
#### Conversión Y Optimización De Modelos Yolov5 Para Implementación En Dispositivos Móviles (Tflite + Kotlin).

Los modelos de detección y reconocimiento utilizados en este proyecto fueron entrenados utilizando **Google Colab**, ya que esta plataforma ofrece una mayor potencia computacional, especialmente útil para entrenamientos de modelos complejos como YOLOv5 y CRNN. Colab permite el uso de GPUs y TPUs, lo que optimiza significativamente los tiempos de entrenamiento y hace viable trabajar con grandes volúmenes de datos. Una vez completado el entrenamiento de los modelos —YOLOv5 para la detección de regiones clave y CRNN para el reconocimiento de texto, se procedió a su conversión y optimización a **TensorFlow Lite (.tflite)**, un formato compatible con aplicaciones móviles desarrolladas en **Android Studio/ Kotlin**.

El modelo YOLOv5, entrenado para detectar secciones específicas de la nota de venta (como la fecha, el número de serie, el cliente y la tabla de detalle), debe ser convertido a un formato compatible con dispositivos móviles para su implementación en la aplicación. Dado que los intentos iniciales de conversión mediante ONNX presentaron conflictos de versiones y otras incompatibilidades, se optó por un proceso alternativo directo desde PyTorch a TensorFlow Lite, lo que permitió superar estos inconvenientes de manera eficiente.

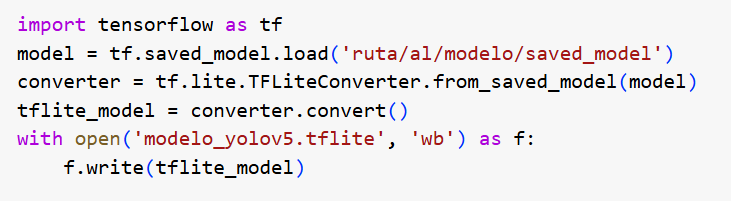
El proceso de conversión se llevó a cabo en varias etapas:

Exportación directa a TensorFlow: Para evitar los problemas derivados de la conversión a ONNX, se utilizó una función de exportación directa desde PyTorch a TensorFlow. Este enfoque optimiza el flujo de trabajo y garantiza que el modelo sea compatible con el formato TensorFlow. El código utilizado para este paso es el siguiente:



Este comando genera un modelo en formato saved\_model de TensorFlow, y simultáneamente crea una versión optimizada del modelo en formato TensorFlow Lite (.tflite), listo para su uso en dispositivos móviles.

Exportación a TFLite: Una vez generado el modelo en formato TensorFlow, se utiliza el conversor oficial de TensorFlow para obtener el archivo .tflite que es compatible con dispositivos móviles. El código para realizar esta conversión es el siguiente:



Optimización: Durante la conversión, se aplicaron técnicas de optimización, como la cuantización, utilizando configuraciones como float16 o int8, lo que permitió reducir el tamaño del modelo y acelerar el proceso de inferencia, sin que se viera afectada significativamente la precisión del modelo.

#### Conversión Y Optimización De Modelos CRNN Para Implementación En Dispositivos Móviles (Tflite + Kotlin).

El modelo CRNN (Convolutional Recurrent Neural Network), utilizado para el reconocimiento de texto en las notas de venta, fue entrenado originalmente en PyTorch. Al igual que con el modelo YOLOv5, la conversión del modelo CRNN al formato compatible con dispositivos móviles fue esencial para poder implementarlo en la aplicación móvil. Dado que el modelo estaba en formato .pth, se necesitaba convertirlo a TensorFlow Lite (.tflite) para su ejecución en dispositivos con recursos limitados.

El proceso de conversión se realizó en varias fases:

1. Adaptación del modelo CRNN a TFLITE

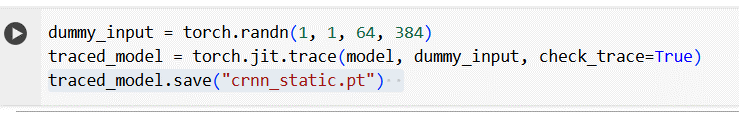
El proceso de adaptación se detectaron problemas en el modelo original que impedían una correcta conversión detallados en la **Tabla 2**.

**Tabla 2**  
*Problemas en adaptación CRNN a TFLITE*

|  |  |  |
| --- | --- | --- |
| Característica | Incompatibilidad | Solución |
| AdaptiveAvgPool2d  ((1, None)) | Dimensiones dinámicas en el pooling | Reemplazar por AvgPool2d(kernel\_size=(16, 1)) |
| LSTM bidireccional | Soporte limitado en TFLite | Usar proj\_size=0 y SELECT\_TF\_OPS |
| Normalización en preproceso | Inconsistencia entre entrenamiento/inferencia | Integrar normalización en el modelo ((x - 0.5) /0.5) |

1. Exportación TorchScript

Una vez determinado los problemas que impiden la conversión a TFLITE procedemos a la conversión fijando una entrada estatica de dimensiones ( [ 1 , 1 , 64 , 384 ] ) para tener una compatibilidad con TFLITE



1. Conversión a TFLITE con optimizaciones

Se empleó la herramienta TFLiteConverter para transformar el modelo CRNN previamente entrenado a su versión compatible con TensorFlow Lite. Durante esta fase, se especificó una forma de entrada fija ([1, 1, 64, 384]), la cual es representativa de las dimensiones que tendrá la entrada durante la inferencia en el dispositivo.

Debido a que el modelo CRNN incluye componentes como capas LSTM bidireccionales, fue necesario activar operadores no nativos de TFLite mediante la opción SELECT\_TF\_OPS. Esta configuración permite que la inferencia utilice ciertas operaciones avanzadas disponibles en la versión estándar de TensorFlow, asegurando así la compatibilidad completa con la arquitectura del modelo.

Con el fin de reducir el tamaño del modelo y optimizar su ejecución en dispositivos con limitaciones de hardware (memoria y procesamiento), se aplicó cuantización post-entrenamiento utilizando la estrategia tf.lite.Optimize.DEFAULT. Esta técnica convierte los pesos y activaciones del modelo de precisión flotante (float32) a formatos de menor tamaño como int8 o float16, lo que típicamente resulta en una disminución del tamaño del archivo hasta en un 75%, con un impacto mínimo en la precisión del reconocimiento.

Tras aplicar las configuraciones anteriores, el modelo resultante fue serializado y almacenado en un archivo .tflite, el cual puede ser embebido dentro de una aplicación móvil desarrollada en Kotlin. Este modelo está preparado para realizar inferencias eficientes directamente en el dispositivo, sin necesidad de conexión a internet ni recursos en la nube.

### Iteración 7.

En esta fase se desarrolla una aplicación móvil utilizando Android Studio con Kotlin como lenguaje principal, con el objetivo de integrar el flujo completo de procesamiento de notas de venta manuscritas mediante modelos previamente entrenados y convertidos a TensorFlow Lite. La aplicación se estructura en tres componentes funcionales: interfaz de usuario (frontend), procesamiento interno de datos tras la captura de la imagen, y presentación de estadísticas mediante una navegación intuitiva. El reconocimiento de texto y el procesamiento semántico se desarrollan completamente en Kotlin, utilizando estructuras de datos, algoritmos de corrección heurística y una base de datos local con SQLite para mejorar el rendimiento y la portabilidad en dispositivos de gama baja.

#### Diseño Fronted De La Aplicación.

La interfaz de la aplicación está diseñada con un enfoque minimalista y funcional, centrado en garantizar una experiencia fluida y adaptable a distintas resoluciones de pantalla. La pantalla principal presenta tres botones principales, dispuestos estratégicamente:

Botón central: ubicado en la parte inferior central, activa la cámara del dispositivo para capturar una imagen de la nota de venta manuscrita.

Botón izquierdo: accede a estadísticas de ventas según el rango de tiempo (por día, mes o año).

Botón derecho: muestra estadísticas de ventas agrupadas por producto.

La interfaz también incluye una previsualización de la imagen capturada, una barra de progreso o estado de análisis y mensajes emergentes que informan al usuario sobre los resultados o errores detectados.

Para su implementación se utilizaron herramientas como CameraX para la captura de imágenes, ConstraintLayout y RecyclerView para la disposición visual, asegurando una presentación clara y adaptativa.

#### Manejo De Datos Tras Captura.

Una vez capturada la imagen de una nota de venta manuscrita, esta es preprocesada y enviada a los modelos TFLite en la app, siguiendo el siguiente flujo de procesos:

* Redimensionamiento de la imagen: para asegurar compatibilidad con el modelo YOLOv5.
* Primera detección con YOLOv5: se identifican las áreas principales del documento, como la fecha, el número de serie, el destinatario y la sección de detalle.
* Segunda detección (detalle): si se identifica el área de detalle de productos, esta es recortada y analizada mediante un segundo modelo YOLOv5 entrenado para detectar celdas individuales correspondientes a cantidad, descripción, precio unitario y precio total.
* Reconocimiento de texto: cada celda (cantidad, descripción, precio unitario y total) es enviada al modelo CRNN (TFLite), que realiza el reconocimiento de texto manuscrito.
* Corrección semántica: se aplica un diccionario de palabras clave y reglas heurísticas para corregir errores comunes de escritura o interpretación.
* Organización por posición: las coordenadas X y Y proporcionadas por el modelo YOLO se utilizan para ordenar las celdas del detalle de producto de forma coherente, simulando una lectura de arriba hacia abajo y de izquierda a derecha.
* Generación y almacenamiento de datos estructurados: La información resultante se convierte en un objeto estructurado y se guarda localmente utilizando **SQLite**. Cada nota de venta se almacena con su respectivo metadato (fecha, cliente, productos, totales), permitiendo una recuperación eficiente y rápida.

Todos los procesos se ejecutan de forma local, sin necesidad de conexión a Internet, y están optimizados para funcionar en dispositivos de gama media y baja.

El uso exclusivo de Kotlin junto con TensorFlow Lite y SQLite reduce la complejidad del sistema, mejora la estabilidad, el rendimiento y la portabilidad de la aplicación.

## FASE DE FINALIZACION

**Figura 45**  
*Pesos de entrenamiento de área detalle*

Nota. – Elaboración propia, 2025.

CAPITULO IV

# EVALUACION DE RESULTADOS

En este capítulo se centra en validación de la hipótesis planteada en el CAPITULO I proponía que una aplicación móvil basada en Deep Learning y HTR podría permitir a los contribuyentes del RTS digitalizar notas de venta manuscritos con una precisión superior al 70%, generando versiones estructuradas de los mismos, listas para ser almacenadas y analizadas digitalmente. Para ello, se han desarrollado una aplicación que hace uso de la inteligencia artificial, utilizando Deep Learning, Yolov5, CRNN, y se ha aplicado en un conjunto de imágenes de notas de venta manuscritas para evaluar su rendimiento.

## METODOLOGIAS DE EVALUACION

### Selección De Estudio.

Se utilizo un conjunto de 50 notas de venta manuscritas, capturadas en distintos entornos, todas las notas presentan escritura legible (según criterio visual). Se usaron notas de venta con diferentes tipos de tinta azul, negro, para observar el contraste con el fondo del papel, se excluyeron notas de venta con letras tipo “doctor” ni escrituras con trazos ilegibles

1. Yolo es un modelo de visión artificial para la detección de objetos. [↑](#footnote-ref-1)
2. CRNN Convolutional Recurrent Neural Networks es la conbinacion de dos tipos de redes neuronales, CNN (convolutional neural network) y RNN (Recurrent neural networks). [↑](#footnote-ref-2)
3. RTS siglas de Régimen Tributario Simplificado [↑](#footnote-ref-3)
4. Mobile-D es una metodología de desarrollo de software ágil, específicamente diseñada para el desarrollo de aplicaciones móviles. [↑](#footnote-ref-4)
5. El dataset es un conjunto de datos estructurados que se utiliza para entrenar, validar y probar una red neuronal. [↑](#footnote-ref-5)
6. HTR siglas de Handwriting Text Recognition que significan reconocimiento de texto manuscrito. [↑](#footnote-ref-6)
7. OCR (Optical Character Recognition) es una tecnología que convierte imágenes de texto, como documentos escaneados o fotografías, en texto digital editable y que puede ser buscado. [↑](#footnote-ref-7)
8. IBM que significa International Business Machines, es una empresa multinacional de tecnología y consultoría. [↑](#footnote-ref-8)
9. IA siglas de inteligencia artificial [↑](#footnote-ref-9)
10. ICR (optical character recognition), reconocimiento inteligente de caracteres es una versión avanzada de reconocimiento de escritura a mano [↑](#footnote-ref-10)
11. GPL (GNU Licencia Pública General) es una licencia de software libre. [↑](#footnote-ref-11)
12. Deep Learning significa aprendizaje profundo, es un subconjunto del Machine Learning que utiliza redes neuronales artificiales con múltiples capas. [↑](#footnote-ref-12)
13. HTR siglas de Handwriting Text Recognition que significan reconocimiento de texto manuscrito. [↑](#footnote-ref-13)
14. RTS siglas de Régimen Tributario Simplificado. [↑](#footnote-ref-14)
15. La retropropagación es un método elegante para calcular cómo los cambios en cualquiera de las ponderaciones o sesgos de una red neuronal afectarán a la precisión de las predicciones del modelo. [↑](#footnote-ref-15)
16. Neuronas muertas es cuando muchas activaciones son cero [↑](#footnote-ref-16)
17. Probabilidad cálculo matemático que establece todas las posibilidades que existen de que ocurra un fenómeno en determinadas circunstancias de azar. [↑](#footnote-ref-17)
18. GPL licencia de uso libre (GNU licencia publica general) [↑](#footnote-ref-18)
19. RTS siglas de Régimen Tributario Simplificado [↑](#footnote-ref-19)
20. Script es un archivo que contiene instrucciones para ser ejecutadas por un intérprete o motor. [↑](#footnote-ref-20)