**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](#_Toc198408527)

[Tablas I](#_Toc198408528)

[CAPITULO I 1](#_Toc198408529)

[MARCO REFERENCIAL 1](#_Toc198408530)

[1.1 INTRODUCCIÓN 1](#_Toc198408531)

[1.2 ANTECEDENTES 3](#_Toc198408532)

[1.2.1 Antecedentes internacionales. 3](#_Toc198408533)

[1.2.2 Antecedentes nacionales. 3](#_Toc198408534)

[1.3 PLANTEAMIENTO DEL PROBLEMA 5](#_Toc198408535)

[1.3.1 Problema General. 5](#_Toc198408536)

[1.3.2 Problemas Específicos. 6](#_Toc198408537)

[1.4 OBJETIVOS. 6](#_Toc198408538)

[1.4.1 Objetivo General. 6](#_Toc198408539)

[1.4.2 Objetivo Específico. 6](#_Toc198408540)

[1.5 HIPOTESIS 7](#_Toc198408541)

[1.6 OPERACIÓN DE VARIABLES 7](#_Toc198408542)

[1.7 JUSTIFICACIÓN. 7](#_Toc198408543)

[1.7.1 Justificación Social. 7](#_Toc198408544)

[1.7.2 Justificación Económica. 8](#_Toc198408545)

[1.7.3 Justificación Académica. 8](#_Toc198408546)

[1.8 LIMITES Y ALCANCES 8](#_Toc198408547)

[1.8.1 Alcance. 8](#_Toc198408548)

[1.8.2 Limites. 9](#_Toc198408549)

[1.9 METODOLOGIAS 10](#_Toc198408550)

[1.9.1 Metodología De Desarrollo. 10](#_Toc198408551)

[1.9.2 Metodología De La Investigación Científica. 10](#_Toc198408552)

[CAPITULO II 12](#_Toc198408553)

[MARCO TEORICO 12](#_Toc198408554)

[1.1 INTRODUCCION 12](#_Toc198408555)

[1.2 REGIMEN TRIBUTARIO SIMPLIFICADO RTS 12](#_Toc198408556)

[1.3 INTELIGENCIA ARTIFICAL (IA) 13](#_Toc198408557)

[1.4 MACHINE LEARNING 15](#_Toc198408558)

[1.5 DEEP LEARNING 16](#_Toc198408559)

[1.5.1 Redes Neuronales. 17](#_Toc198408560)

[1.5.2 Funciones De Activación. 19](#_Toc198408561)

[1.5.3 Redes Neuronales Convolucionales (RNN). 25](#_Toc198408562)

[1.5.4 Redes Neuronales Recurrentes (RNN). 30](#_Toc198408563)

[1.5.5 Yolov5. 33](#_Toc198408564)

[1.5.6 Tesseract OCR. 34](#_Toc198408565)

[1.6 METODOLOGIA MOBILE-D 35](#_Toc198408566)

[CAPITULO III 37](#_Toc198408567)

[MARCO APLICATIVO 37](#_Toc198408568)

[1.7 INTRODUCCION 37](#_Toc198408569)

[1.8 FASE DE EXPLORACION 37](#_Toc198408570)

[1.8.1 Establecimiento De Grupo De Interés. 37](#_Toc198408571)

[1.8.2 Requisitos Iniciales. 37](#_Toc198408572)

[1.8.3 Requerimientos Funcionales. 38](#_Toc198408573)

[1.8.4 Requerimientos No Funcionales. 38](#_Toc198408574)

[1.8.5 Definicion Del Alcance. 39](#_Toc198408575)

[1.8.6 Establecimiento De Tecnologías. 39](#_Toc198408576)

[1.9 FASE DE INICIACION 44](#_Toc198408577)

[1.9.1 Configuración Del Ambiente De Desarrollo. 45](#_Toc198408578)

[1.9.2 Planificación De Fases. 47](#_Toc198408579)

[1.9.3 Diseño De Aplicación. 49](#_Toc198408580)

[1.10 FASE DE PRODUCCION 50](#_Toc198408581)

[1.10.1 Iteración 1. 50](#_Toc198408582)

[1.10.2 Iteración 2 54](#_Toc198408583)

[1.10.3 Iteración 3 55](#_Toc198408584)

[1.10.4 Iteración 4. 58](#_Toc198408585)

[1.10.5 Iteración 5. 61](#_Toc198408586)

[1.11 FASE DE ESTABILIACION 63](#_Toc198408587)

[1.11.1 Iteración 6. 63](#_Toc198408588)

[1.11.2 Iteración 7. 67](#_Toc198408589)

[1.12 FASE DE FINALIZACION 70](#_Toc198408590)

[CAPITULO IV 71](#_Toc198408591)

[EVALUACION DE RESULTADOS 71](#_Toc198408592)

[1.13 METODOLOGIAS DE EVALUACION 71](#_Toc198408593)

[1.13.1 Selección De Estudio. 71](#_Toc198408594)

# 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 para comerciantes minoristas, artesanos y vivanderos (economiayfinanzas.gob.bo portal, Boletín Económico Ingresos Tributarios 2021).

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 generación de balances contables, análisis de ventas y la planificación estratégica, limitando la transparencia de sus operaciones 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 y CRNN. El objetivo principal es transformar automáticamente una imagen de la nota 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 “pro-forma” se emplean de forma intercambiable. En el contexto del Régimen Tributario Simplificado (RTS) 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 y consistirá en la recopilación de notas de venta de diferentes negocios para construir un dataset 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 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 para la traducción de chino al español (Garcia, 2015).

El reconocimiento óptico de caracteres (OCR) una herramienta clave para la digitalización, según IBM, “El software OCR puede aprovechar la inteligencia artificial (IA) para implementar métodos más avanzados de reconocimiento inteligente de caracteres (ICR) 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 su capacidad para generar 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 obligados a recurrir a prestamistas informales, lo que los expone a tasas de interés elevadas y condiciones desfavorables. 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 y distribuida bajo una licencia GPL (Licencia Pública General), basada en Deep Learning con tecnología de 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 para el cliente. Lo que facilitaría un análisis en sus ventas y coadyuve en la toma de decisiones futuras.

### Objetivo Específico.

* Diseño de un modelo de reconocimiento de texto manuscrito (HTR).
* Diseño 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.
* Desarrollo 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 (RTS), permite 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.

**Variable independiente:** detección y digitalización.

**Variable interviniente:** Deep Learning y HTR.

## JUSTIFICACIÓN.

### Justificación Social.

Con el desarrollo de la aplicación móvil con licencia GLP garantiza 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.

### 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 GLP 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, cnn, htr, para la digitalización de recibos manuscritos.
* Generar recibos digitales.
* Almacenamiento de datos para análisis de ventas.
* Exportación de datos .csv 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 Handwritten Text Recognition (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 GLP. 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 (Régimen Tributario Simplificado) 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 (IA) 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 (IA) 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.).

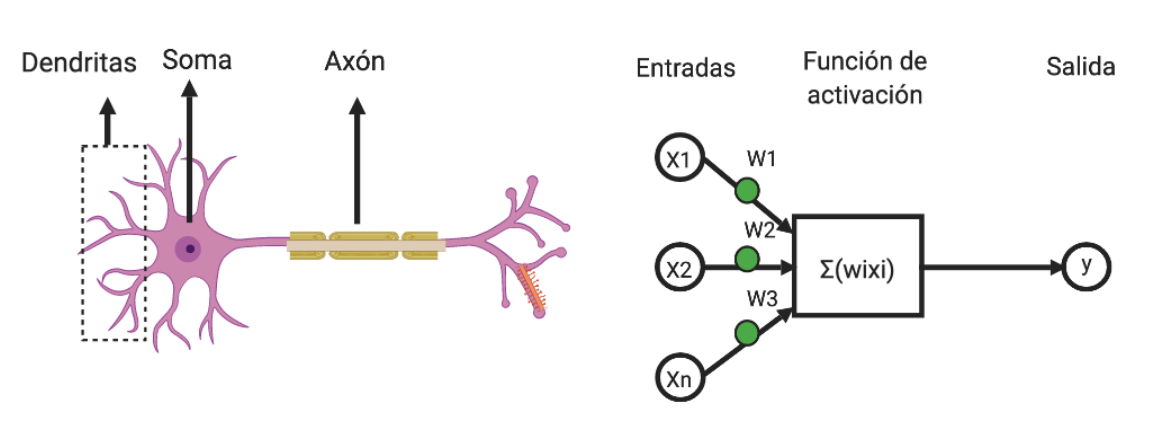
Para diferenciar entre el deep learning y el machine learning es la estructura de la arquitectura de red neuronal subyacente. Los modelos tradicionales de machine learning “no profundos (Deep learning)” utilizan redes neuronales simples con una o dos capas computacionales. Los modelos de deep learning 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).

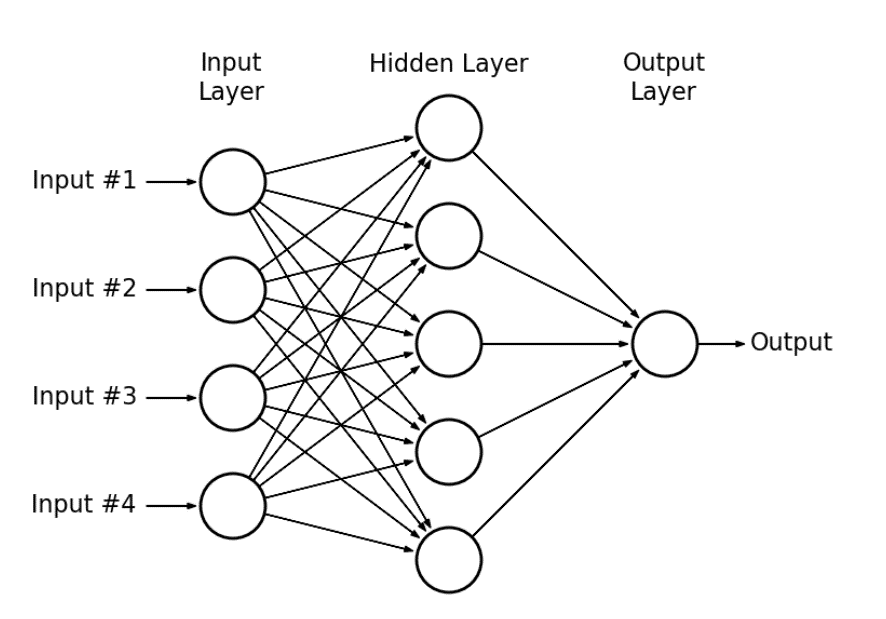
**Figura 1**  
Introducción a redes neuronales



*Nota*. – Comparación de como la base de una neurona biológica sirvió de base para la creación de una red neuronal, adaptado de Neurona vs. Perceptron [fotagrafia], 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 2   
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).

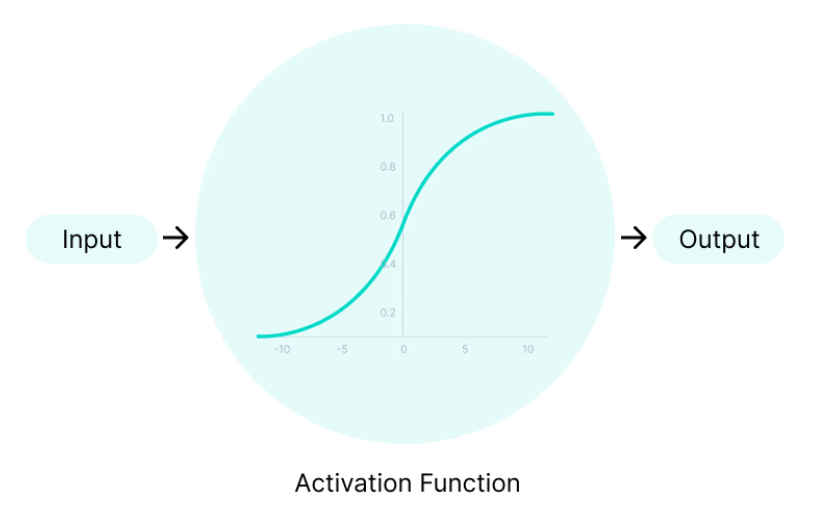
Como se ve en la **Figura 1**, puede formularse matemáticamente a la neurona básica de la siguiente manera: se cuenta con n entradas xi, en cada entrada es multiplicada por un peso wi, si existen n entradas se contará con n pesos, para finalmente calcular la suma total, como se muestra en la ecuación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

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

### Funciones De Activación.

La función de activación es aquella que define la salida de una neurona como se ve **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 umbral de paso binario. - esta función, mapea los datos de entrada en salidas binarias en función a un umbral definido, con el cual se comparan las entradas como se ve **Figura 4** y su ecuación (2).

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

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

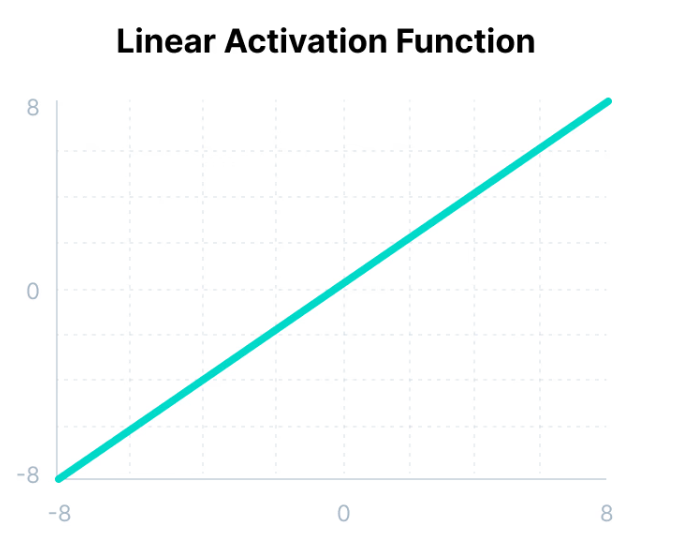


*Nota.* - Función de paso binario que depende de un valor umbral con la cual la neurona decide activarse o no. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Función lineal. – la activación neuronal es proporcional a la entrada, por tanto, la salida es exactamente igual a la entrada sin realizar ninguna clase de procesamiento como se ve en **Figura 5** y su ecuación (3).

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

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

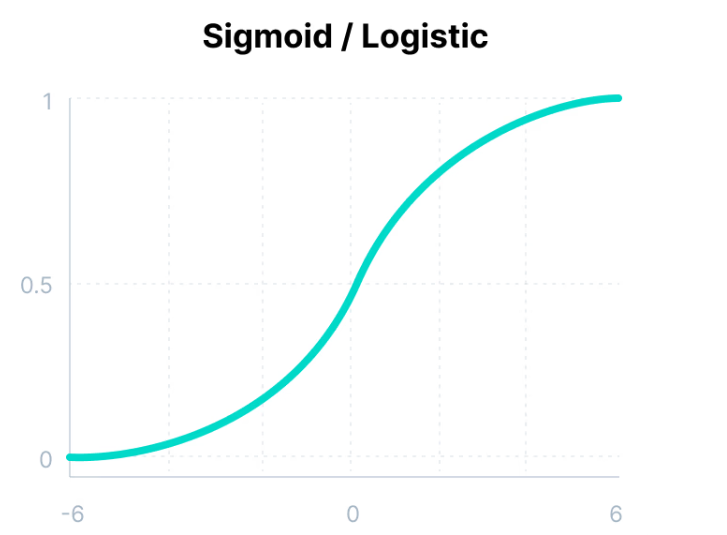


***Nota****.* – Función de activación también conocida como no activación o función de identidad. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Función sigmoidea. - tiene la particularidad de tener la forma de una S como se ve en la **Figura 6**, la salida producida por la función se encuentra en el rango de 0 a 1, acercándose a cada cota mientras el valor sea mayor o menor respectivamente como se ve en la ecuación (4). Esta es la más utilizada en modelos que usan la probabilidad como salida.

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

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

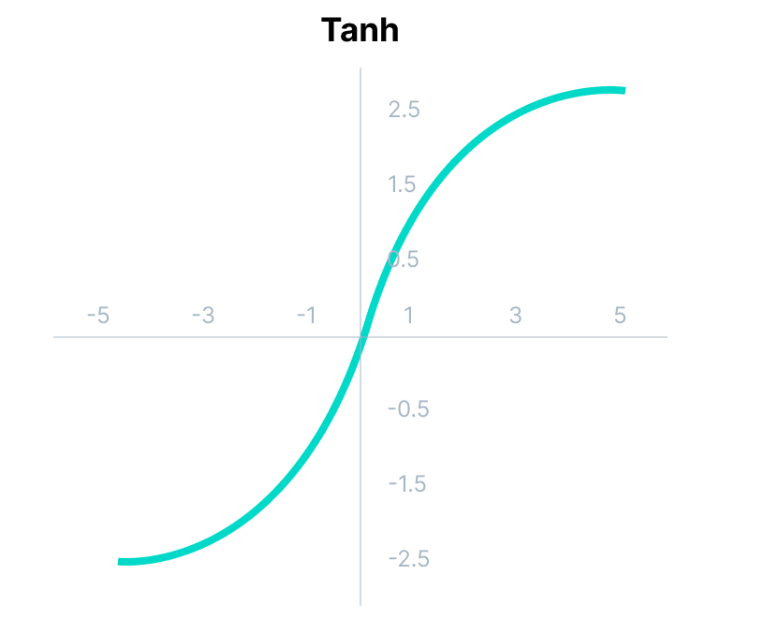


*Nota*. – Función de activación no lineal sigmoid/logistic toma cualquier valor real como valores de entrada y salida en el rango de 0 a 1. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Función de tangente hiperbólica. - comparte similitudes con la función sigmoide puede verse en la **Figura 7**, incluso tiene la misma forma S con la diferencia de que el rango va de -1 a 1 como se aprecia en la ecuación (5). Una de las ventajas y cualidades de esta función que está centrada en cero, por los que los valores pueden identificarse como fuertemente negativos, neutrales o fuertemente positivos.

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

**Figura 7**  
*Función Tanh*

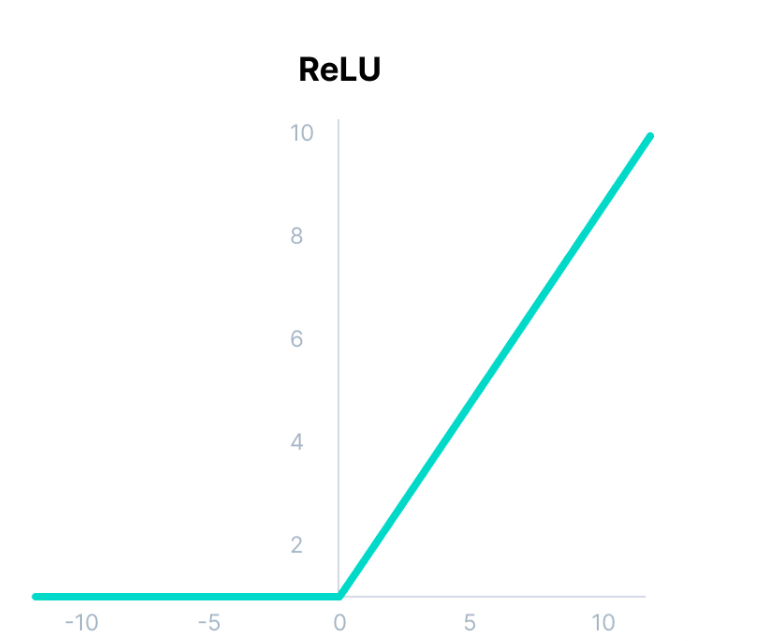


*Nota.* - Funcion Tanh similar a la función de actiivacion sigmoid/logistic con la diferencia en el rango de salida que admite -1 a 1. Adaptado de Activation Functions in Neural Networks [Imagen], por Pragati Baheti, 2021, v7 labs (https://www.v7labs.com/blog/neural-networks-activation-functions).

* Funcion ReLU. – ReLu son siglas de Unidad Lineal Rectificada tiene una función derivada que permite la retro propagación al mismo tiempo hace que esto sea computacionalmente eficiente como se ve en la **Figura 8**, en este la función mapea las entradas positivas, caso contrario la salida será 0 como se ve en la ecuación (6), esta característica es útil al no activar al mismo tiempo todas las neuronas, siendo ampliamente usadas en arquitecturas de redes neuronales convolucionales. Sin embargo, puede llegar a ser problemática durante el uso de blackpropagation, por lo que existen variantes como leaky ReLU o ReLU paramétrico.

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

**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. - comúnmente utilizada en problemas de clasificación de múltiples clases, también se puede ver como la combinación de varias funciones sigmoides, con la diferencia que esta función retorna una probabilidad por cada clase como se ve en la ecuación (7).

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

### Redes Neuronales Convolucionales (RNN).

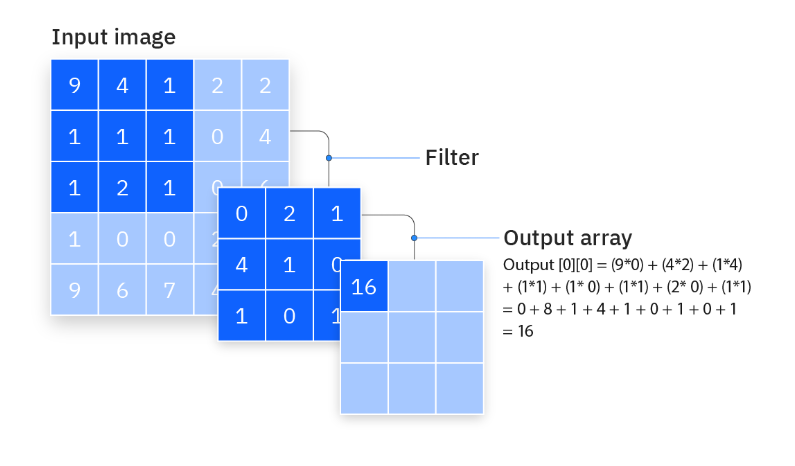
Las redes neuronales convolucionales o también neuronal network convolution, con siglas RNN o ConvNets. Es un subconjunto de Machine learning y es base principal del Deep learning.

Las redes neuronales convolucionales utilizan datos tridimensionales para tareas de clasificación de imágenes y reconocimiento de objetos (IBM International Business Machines Corporation, s.f.).

Las redes neuronales convolucionales se diferencian de otras redes por tener un rendimiento superior en sus entradas en imagen, voz y audio. Tienen tres capas principales, las cuales son:

* Capa convolucional o colvolutional layer. – es la parte central de las redes neuronales de convolución y es donde ocurre la mayoría de los cálculos como se ve en la **Figura 9**. Se crea un Kernel o filtro el cual es una matriz bidimensional, que se mueve a través de la imagen verificando si existe alguna característica relevante, generando una salida también conocida como mapa de activación o capa de convolución. Después de cada operación de convolución, una CNN aplica una transformación de Unidad Lineal Rectificada (ReLU) al mapa de características, introduciendo la no linealidad al modelo.

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

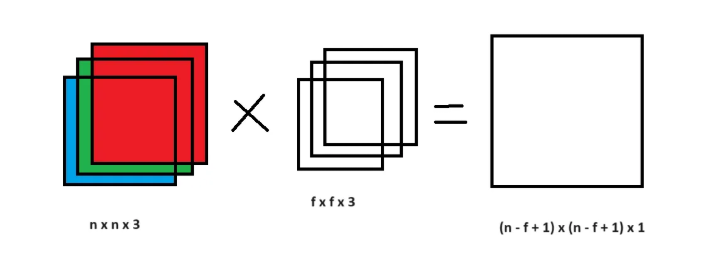


*Nota*. – Redes neuronales convolucionales se muestra el proceso de convolucion en las diferentes capas y filtros. Adaptado de Convolutional Neuronal Networs, por IBM International Business Machines, IBM (<https://www.ibm.com/think/topics/convolutional-neural-networks>). CC by 2.0

En las capas convolucionales se usan múltiples Kernels, y no solo un kernel por capa, en lugar de ello se entrenan multitud de ellos de forma paralela como se aprecia en la **Figura 10**, es usual encontrar un número elevado de kernels, suelen ir en potencia de dos, elevando su número a medida que se va más profundo en la estructura de la red.

El número de canales debe seri igual, tanto para la imagen de entrada como el kernel, es decir si tenemos una imagen en RGB que componen la forma W x H x D, y un kernek de dimensiones F x F, debe cumplirse la cantidad del kernel coincida con los de la imagen de entrada, como se muestra en la imagen.

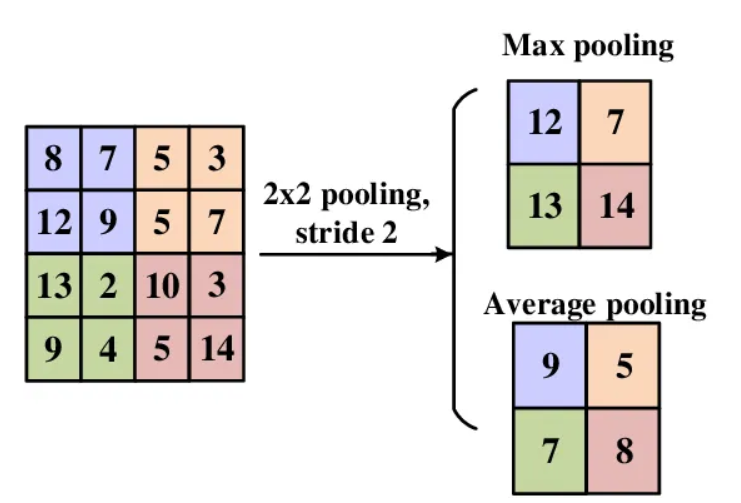
**Figura 10**  
*Operación de convolución*



*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 downsampling, similar a la capa de convolución, la operación de agrupación utiliza un filtro en toda la entrada como se ve en la **Figura 11**. Su principal objetivo es el reducir de manera progresiva el tamaño, para reducir el coste computacional y complejidad del modelo.

**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).

Las capas de pooling operan directamente e independientemente, de manera similar a las capas convolucionales sobre su entrada, al tratarse de operaciones fijas, no existen parámetros que necesitan ser aprendidos. Existen varios tipos de pooling de los cuales los más usados y conocidos son: average pooling y max pooling, siendo este último el más usado.

La gran ventaja en el uso de capas max pooling, que sumado a la gran reducción de dimensiones que experimentan los datos, estas son altamente representativos. Sin embargo, debe tenerse especial cuidado en el tamaño de los filtros de pooling, ya que un tamaño muy pequeño puede resultar bastante perjudicial en las dimensiones de los datos y puede recaer en un decremento en el rendimiento de la red.

* Capa totalmente conectada o fully-connected layer o FC. – en la capa totalmente conectada, cada nodo de la capa de salida se conecta directamente a un nodo de la capa anterior. Esta capa realiza la tarea de clasificación en función de las características extraídas a través de las capas anteriores y sus diferentes filtros. en esta capa se realiza la tarea de clasificación en función de las características extraídas a través de las capas anteriores y sus diferentes filtros. mientras que en las capas anteriores tienden a usar las funciones ReLU, las capas FC generalmente aprovechan una función de activación softmax para clasificar las entradas de manera adecuada, produciendo una probabilidad de 0 a 1.

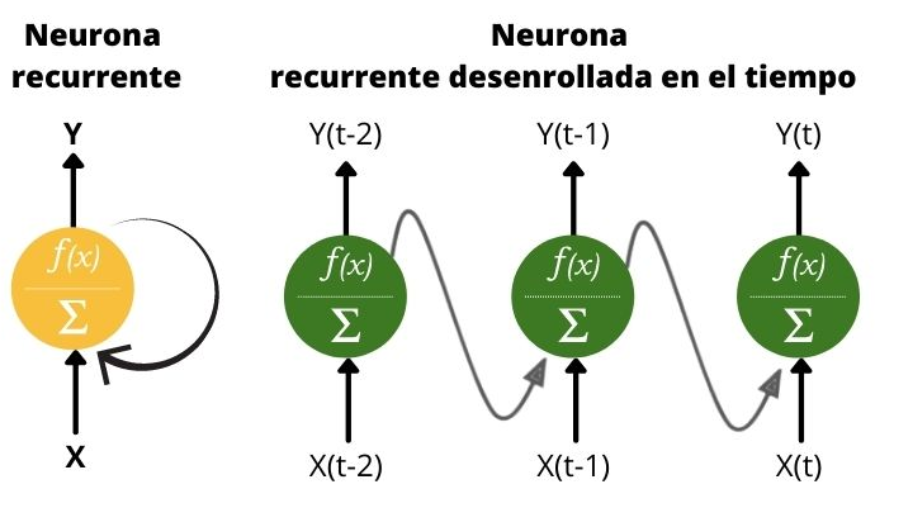
Las redes neuronales convolucionales son una parte muy importante en el Deep learning, ya que esta especialmente diseñada para procesar datos con estructura espacial, en imágenes, señales y secuencias, tiene un uso amplio en diversas áreas:

* 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 12**.

**Figura 12**  
*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.).

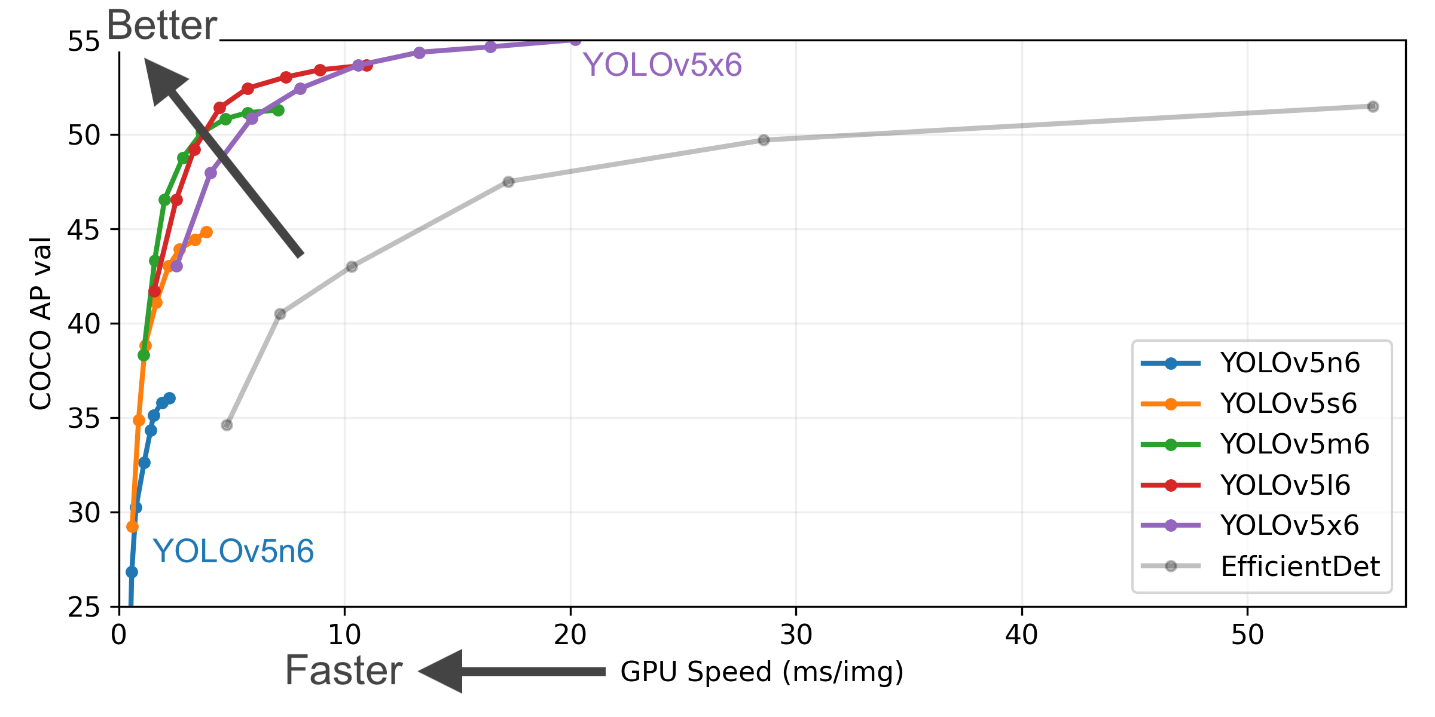
* + - Redes neuronales recurrentes estándar. - Es la versión más básica de RNN, en la que la salida en cada paso temporal depende tanto de la entrada actual como del estado oculto del paso temporal anterior, lo que dificulta el aprendizaje de dependencias a corto plazo (IBM International Business Machines Corporation, s.f.).
    - Redes neuronales recurrentes bidireccionales (BRNN). – son una variante avanzada de las RNN estándar que mejoran la capacidad de procesamiento de secuencias al analizar los datos en ambas direcciones.
    - Memoria a corto plazo (LSTM). – es una arquitectura RNN popular, que fue introducida por Sepp Hochreiter y Juergen Schmidhuber como solución al problema del vanishing gradient o desvanecimiento del gradiente. Su trabajo abordo el problema de las dependencias a largo plazo. Son ampliamente utilizadas en tareas donde el contexto histórico es crucial, como traducción automática, predicción de series temporales y procesamiento de lenguaje natural (NPL).
    - Unidades recurrentes bloqueadas (GRU). – Las GRU son una variante mas eficiente y simplificada de las LSTM (Long Short-Term Memory). Diseñadas para resolver el problema de vanishing gradient en redes recurrentes, pero con menos parámetros y mayor velocidad de entrenamiento, Son ampliamente utilizadas en procesamiento de secuencias.
    - Codificador-decodificador RNN. – el modelo codificador-decodificador (Encoder-Decoder) basado en RNN es una arquitectura fundamental para tareas de procesamiento de secuencia a secuencia (seq2seq), como traducción automática, resumen de texto, chatbots, combina dos redes neuronales recurrentes (RNN, LSTM o GRU).

### Yolov5.

Yolo es un algoritmo de detección de objetos en tiempo real que puede identificar objetos en una imagen y proporcionar las áreas y la probabilidad de confianza.

Yolov5 es un repositorio que surgio como una evolución de yolov3 una implementación de PyTorch desarrollado por Glenn Jocher en 2020, lanzado por ultralytics. Yolov5 tiene cuatro versiones principales: yolov5s, yolov5m, yolov5l, yolov5x cada uno ofrece tasas de precisión progresivamente más altas, cada variante también toma una cantidad diferente de tiempo para entrenar como en la **Figura 13**.

**Figura 13**  
*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 motor de reconocimiento óptico de caracteres para varios sistemas operativos, permite el reconocimiento de caracteres dentro una imagen digital.

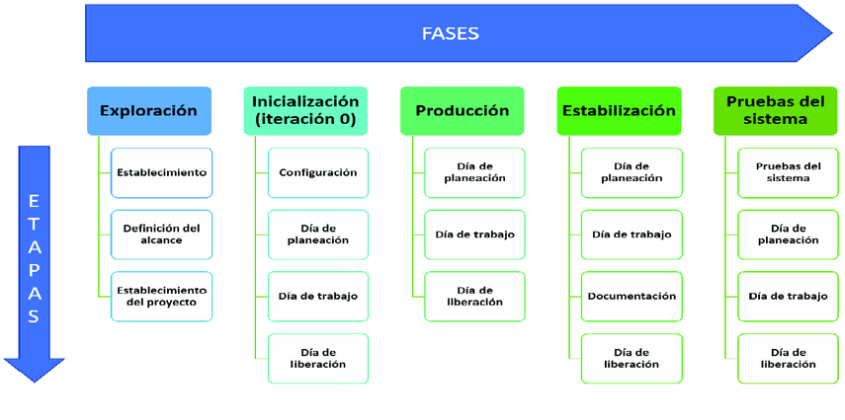
Tesseract se desarrolló originalmente en Hewlett-Packard Laboratories Bristol UK y en Hewlett-Packard Co, Greeley colorado USA entre 1985 y 1994, con algunos cambios mas realizados en 1966 para portar a Windows, y algunos C++ en 1998. En 2005 Tesseract fue código abierto por HP, desde el 2006 hasta noviembre del 2018 fue desarrollado por Google.

## METODOLOGIA MOBILE-D

El desarrollo de la aplicación esta realizado a través de la metodología Mobile-D, esta metodología está orientada al desarrollo de aplicaciones móviles, Mobile-D se basa en metodologías conocidas pero aplicadas de estricta como: Extreme Programming (XP), Crystal Methodologies y Rational Unified Process. XP para las prácticas de desarrollo, Cristal para escalar los métodos y RUP como base de ciclo de vida (Leyva, Carreño, Estrada, Sandoval, & Ezpinoza, 2016).

La metodología Mobile-D se divide en cinco fases siendo estas: exploración, inicialización, producción, estabilización y pruebas como se ve en la **Figura 14**

**Figura 14**  
*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>).

las fases de la metodología Mobile-D:

* Exploración. - permite planificar y definir los conceptos básicos del proyecto, su alcance y el establecimiento con las funcionalidades a donde se desea llegar, además de la planificación de las demás fases.
* Inicialización. - identifica los recursos necesarios para desarrollar la aplicación.
* Producción. - se repite iterativamente (Planificación-Trabajo-Liberación) hasta implementar todas las funcionalidades. Primero se planifica la iteración de trabajo en términos de requisitos y tareas a realizar. Se preparan las pruebas (Test-Driven-Development, TDD), antes de iniciar al desarrollo de una funcionalidad debe existir la prueba que verifique su funcionamiento.
* Estabilización. - se llevan a cabo las últimas acciones de integración para asegurar que el sistema funcione correctamente.
* Pruebas. - tiene como meta la disponibilidad de una versión estable y plenamente funcional del sistema, el producto terminado e integrado se prueba con los requisitos del cliente y se eliminan todos los defectos encontrados.

CAPITULO III

# MARCO APLICATIVO

## INTRODUCCION

En el presente capítulo se desarrolla la aplicación esta realizado a través de la metodología Mobile-D, esta metodología está orientada al desarrollo de aplicaciones de gama media, en este capítulo se describe el desarrollo de la aplicación móvil en cada una de sus fases.

## FASE DE EXPLORACION

En esta fase se determinó los requisitos iniciales para el desarrollo de la aplicación, se estableció un plan de proyecto y los conceptos, además se definió a los involucrados del proyecto.

### Establecimiento De Grupo De Interés.

* Desarrollador. - es un arquitecto de software cuya función específica es la de elaborar sistemas y tratar de que estas se ejecuten de una manera correcta sin tener ningún margen de error en su programación.
* Comerciante minorista. - podemos describirlo como una persona que hacen movimientos económicos pequeños de un capital reducido, utilizan notas de ventas cuales llenan a mano, el cual les dificulta un control contable y envió de las mismas digitalmente ya sea por falta de conocimiento o capital reducido que impide el uso de las mismas.

### Requisitos Iniciales.

Diseñar una aplicación móvil ligera y distribuida bajo una licencia GPL (Licencia Pública General), basada en Deep Learning con tecnología de 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 para el cliente.

### Requerimientos Funcionales.

* Requerimiento funcional N°1.- la aplicación móvil debe permitir tomar captura para el análisis de la nota de venta manuscrita.
* Requerimiento funcional N°2.- la aplicación móvil debe poder detectar las áreas de interés en la imagen a ser tratadas con las redes neuronales.
* Requerimientos funcional N°3.- la aplicación móvil debe poder facilitar al usuario ser verificada antes de almacenarse.

### Requerimientos No Funcionales.

* Usabilidad: la aplicación debe contar con una interfaz intuitiva que permita al usuario final, realizar la captura de las notas manuscritas a su conversión digital sin necesidad de capacitación previa.
* Fiabilidad: la aplicación funcionara de manera óptima en dispositivos Android gama media, garantizando un tiempo de procesamiento máximo de 8 segundos.
* Visualización: la interfaz mostrara una previsualización en tiempo real del escaneo, con herramientas para ajustar brillo, contraste. Los resultados del reconocimiento de textos y números se presentarán en un formato tipo tabla, que permita la exportación en formatos contables, manteniendo la estructura original de la nota manuscrita.
* Idioma: la aplicación móvil tendrá por defecto el idioma castellano.

### Definicion Del Alcance.

El alcance de este proyecto de investigación supone tener sus limitaciones y alcances.

#### 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.

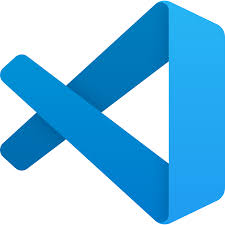
* Diseñar una aplicación móvil basado en Deep learning y HTR.
* Diseñar e implementar una tecnología innovadora para comerciantes minoristas que pertenecen al régimen simplificado de Bolivia.
* Utilizar los recursos en Deep learning, cnn, htr, para la digitalización de recibos manuscritos.
* Generar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

### Establecimiento De Tecnologías.

Primero se tiene algunos conceptos acerca de lo que se utilizara al desarrollar la aplicación móvil:

* 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 15**.

**Figura 15**  
*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 16**. 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 16**  
*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 17**. Es una herramienta que se basa en intelliJ IDEA y ofrece funciones adicionales para mejorar la productividad en el desarrollo de aplicación Android.

**Figura 17**  
*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 18**. Es ampliamente utilizado en diversas áreas como desarrollo web, ciencia de datos, aprendizaje automático y automatización.

**Figura 18**  
*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 19**.

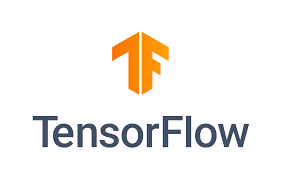
**Figura 19**  
*Logo YoloV5*



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 20**.

**Figura 20**  
*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 21**. 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 21**  
*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 22**. 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 22**  
*Logo GitHub*



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

## FASE DE INICIACION

En esta fase tiene como resultado la realización de actividades de desarrollo y diseño de la aplicación móvil.

### 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.

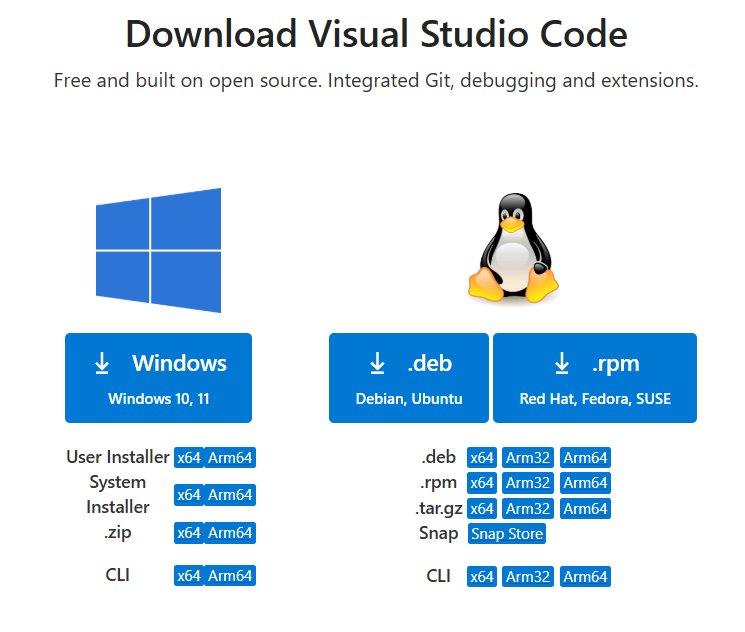
* Tipo de proyecto: aplicación móvil
* Tecnologías: YoloV5, CRNN, TFlite

#### Preparación Del Ambiente.

Instalación de las siguientes herramientas:

* Instalación de Visual Studio Code, descargar de la página oficial podemos ver una captura en **Figura 23**, a continuación el link: <https://code.visualstudio.com/> para Windows 64 bits, instalar extensiones Python.

**Figura 23**  
*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 creo un repositorio.
* Creación de cuenta en Google para el uso de Colab, esto nos servirá para el entrenamiento de las redes neuronales.
* Instalación de Python en el ordenador, link a continuación <https://www.python.org/> versión utilizada v. 3.11.3
* Instalación de Android Studio lo utilizaremos para poder probar la aplicación móvil <https://developer.android.com/>

Tecnologías base de la aplicación:

* Fronted: kotlin
* Backend Local: almacenamiento JSON
* ML: Tensorflow lite para CRNN, YoloV5
* Librerías: OpenCv, TFLite, YoloV5.

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 | * Crear sub áreas de las sub clase detalle para entrenamiento de YoloV5 en palabras. * Detección de palabras de la subsección detalle con YoloV5. |  |
| Iteración 5 | * Integración de CRNN para detección y extracción de texto manuscrito por áreas. |  |
| Iteración 6 | * Generación de JSON estructurado con los datos extraídos. |  |
| Estabilización | Iteración 7 | * Pruebas de simulación en Android Studio, para verificar correcto funcionamiento. * Pruebas físicas en dispositivos móviles. * Optimización de modelos, manejo de errores. |  |
| Iteración 8 | * Comprobación de optimización de modelos. * Medición de precisión y velocidad. * Afinamiento de precisión. * Manejo de errores. * Evaluación general |  |
| Finalización | Iteración 9 | * Versión instalable APK. * Pruebas de usabilidad. |  |

### Diseño De Aplicación.

La aplicación utiliza diferentes capas de detección para extraer los datos y convertirlos a texto digital para utilizarse en JSON.

Podemos aprecias los diferentes procesos simplificados en la **Figura 24**.

**Figura 24**  
*Diseño de aplicación*

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

## FASE DE PRODUCCION

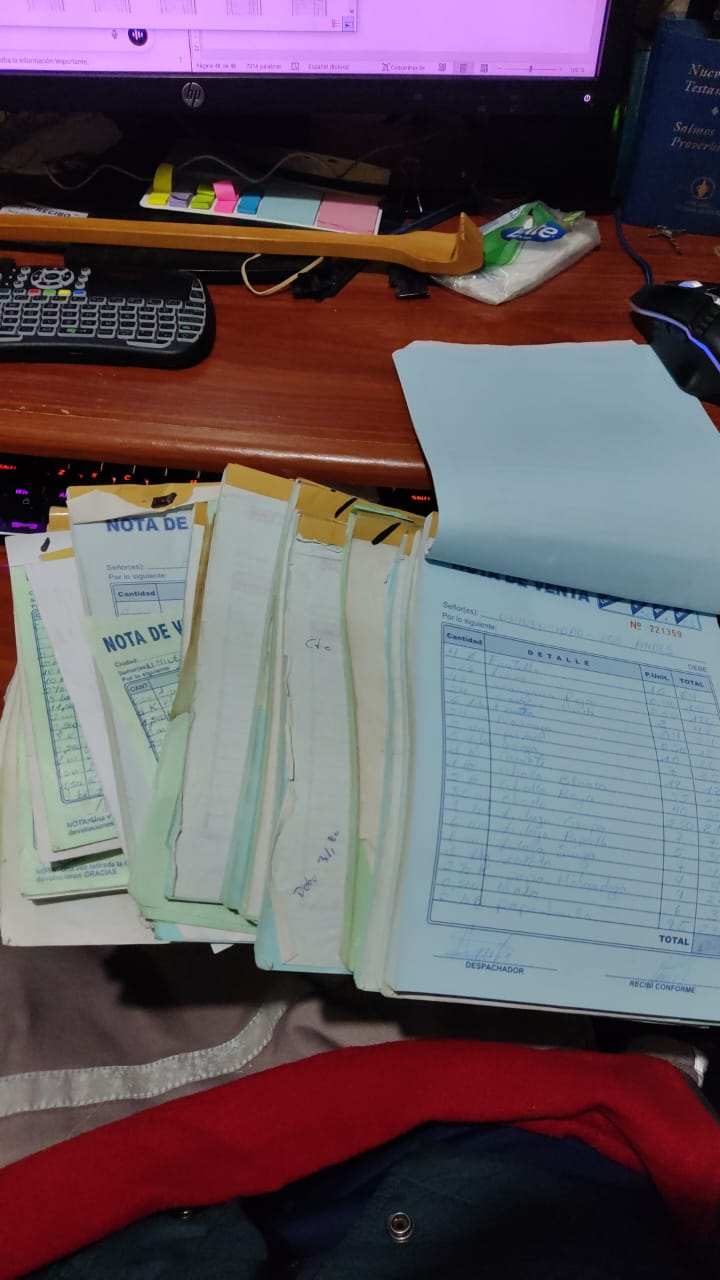
En esta fase se trata de proceder a realizar la funcionalidad de la aplicación móvil junto con todos los requisitos recolectados en las fases anteriores para asegurar que esta cumpla con la calidad deseada de una manera correcta.

### Iteración 1.

#### Captura De Notas De Venta.

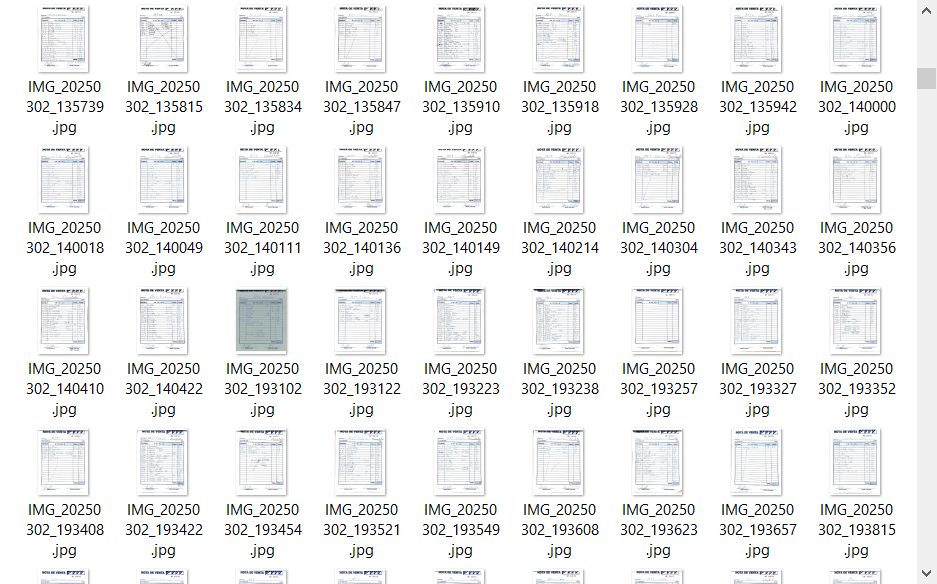
En esta sección nos encargaremos de recolectar la mayor cantidad de notas de venta para su posterior digitalización como parecíamos en la **Figura 25**.

**Figura 25**  
*Notas de venta*



Utiliando una camara de celular se tomo fotografias de diferentes calidades y entornos para tener un gran variedad de muetras, posteriormente se agrupo en una carpeta de imágenes como se ve en la **Figura 26**.

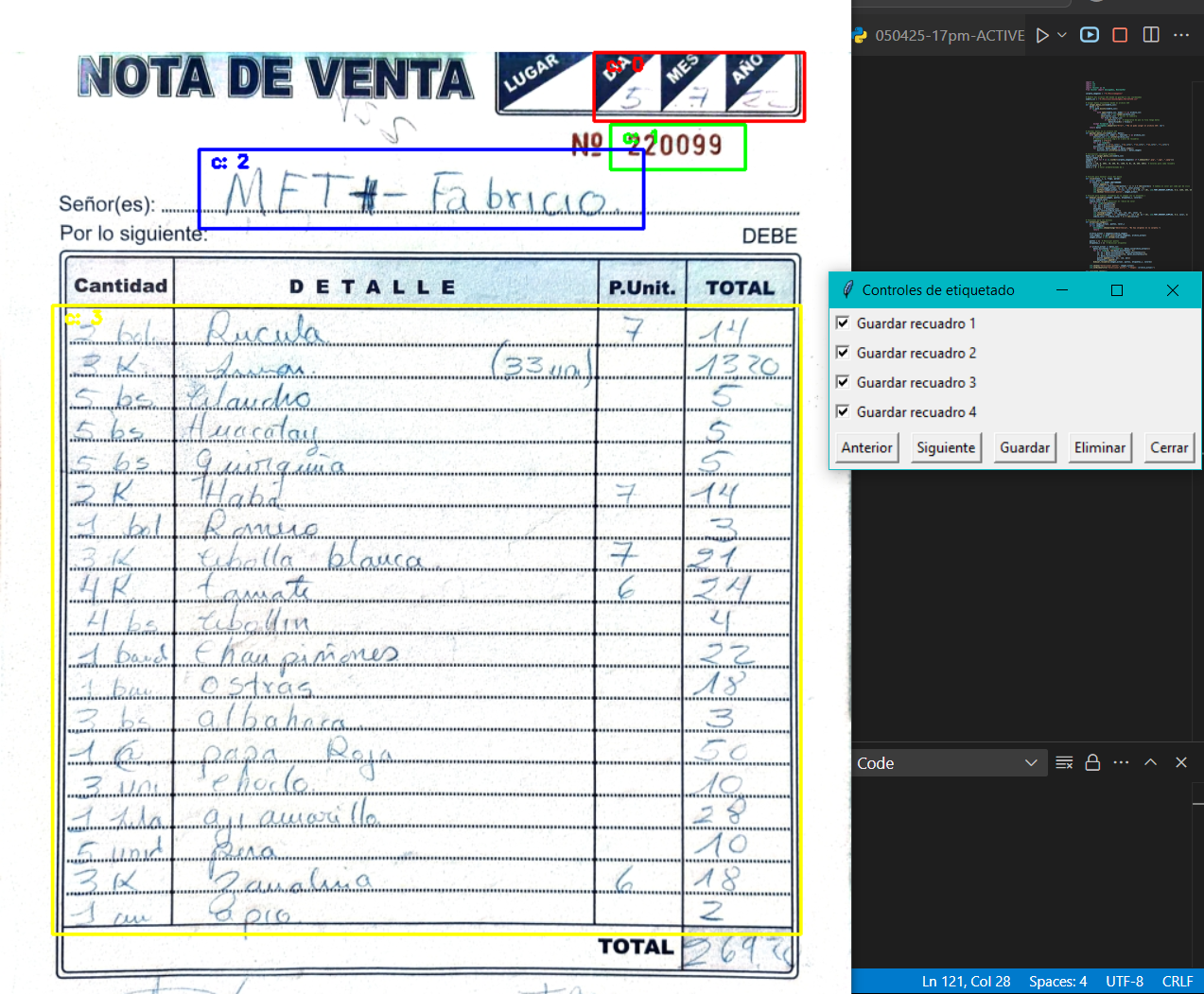
**Figura 26**  
*Notas de venta capturadas*



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

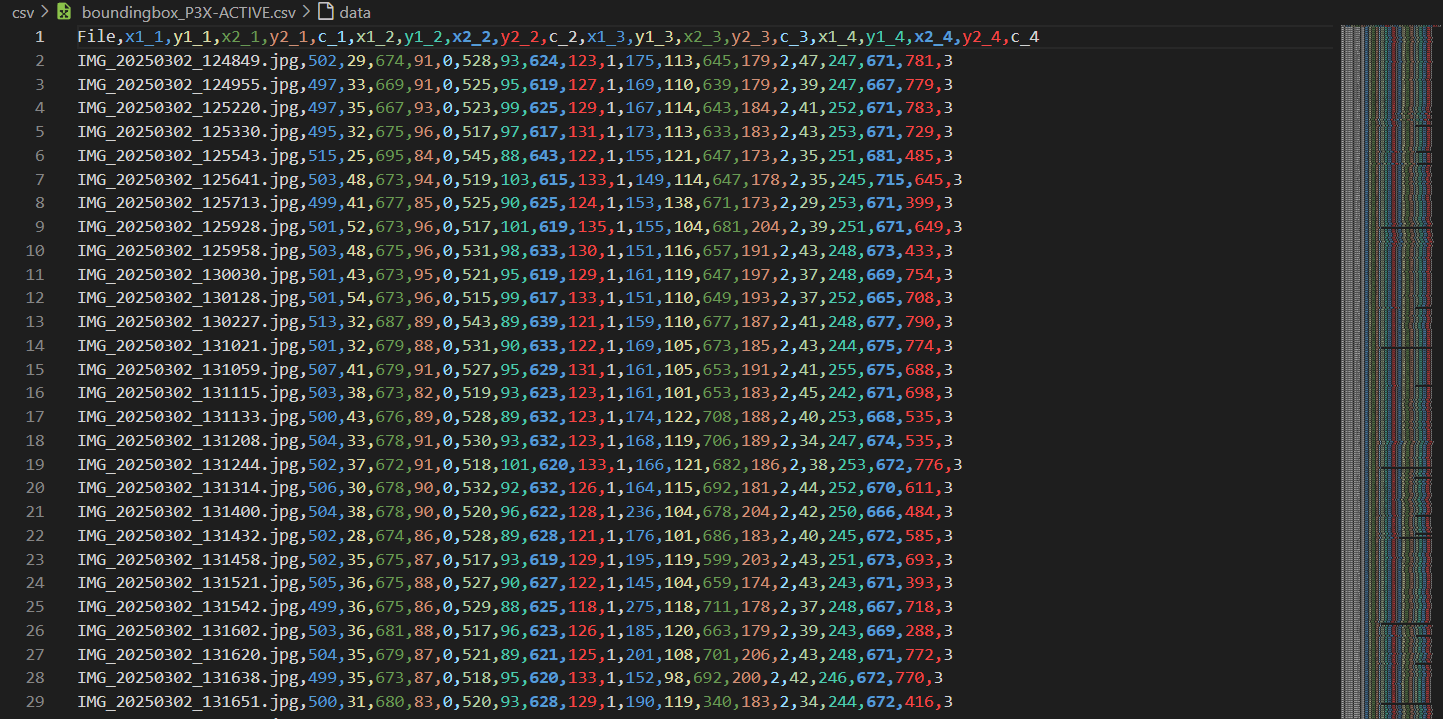
Se diseño un código que permita capturar las áreas de interes para el entrenamiento de YoloV5 para reducir consumo y potencia de computo como se ve en la **Figura 27**.

**Figura 27**  
*Notas de venta generación de bundingbox para el dataset*



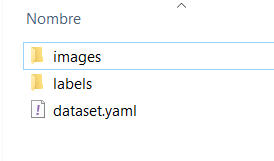
El código consiste en cargar la carpeta de imágenes y crear un archivo csv como se meutra en **Figura 28**, queposteriormente se utilizara para generar áreas de interes, para el entrenamiento, con el respectivo nombre de la imagen.

**Figura 28**  
*boundingbox csv de áreas para el dataset*



Posteriormente a esto se diseño otro código en python para generar el dataset para el entrenamiento de la red neuronal como se aprecia en **Figura 29**.

**Figura 29**  
*formato de dataset de las notas de ventas*

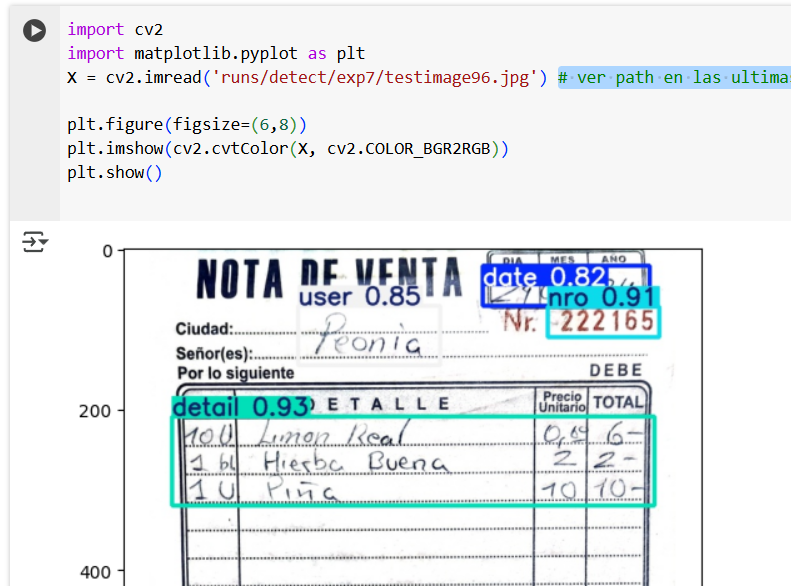


### Iteración 2

#### Entrenamiento De Detección De Áreas 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.

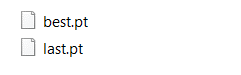
**Figura 30**  
*Detección con YoloV5 de nota de venta*



La detección es posible, al tener una cantidad considerable en el entrenamiento hace que taza de confianza al detectar las áreas sea alta como se aprecia en la **Figura 30**, podemos seguir ampliando el dataset o cambiar la configuración en el aprendizaje, una vez ampliando y probando configuraciones se obtiene una confianza en la detección superior.

Para aprovechar el aprendizaje en colab exportamos los pesos del entrenamiento para probarlo o mejorar su entrenamiento como vemos en la **Figura 31**.

**Figura 31**  
*Pesos de entrenamiento de YoloV5 áreas*

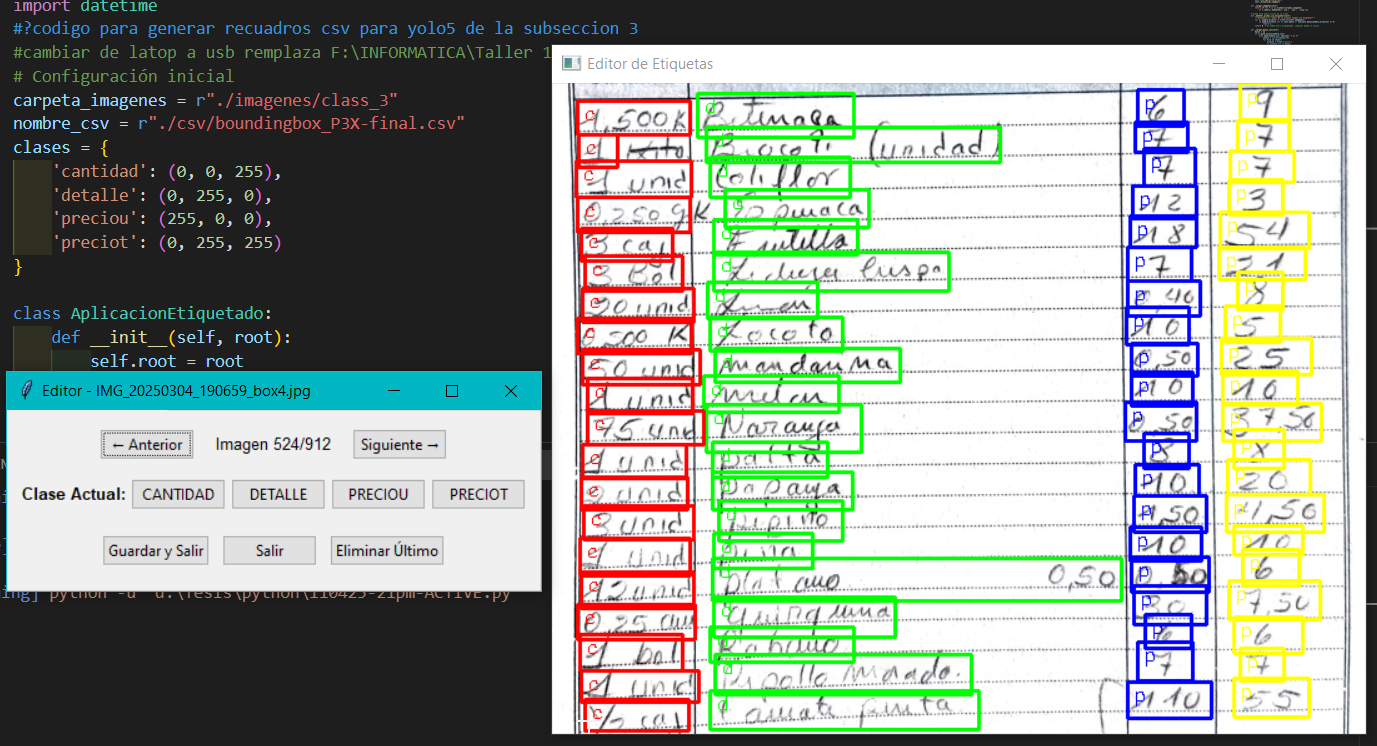


### Iteración 3

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

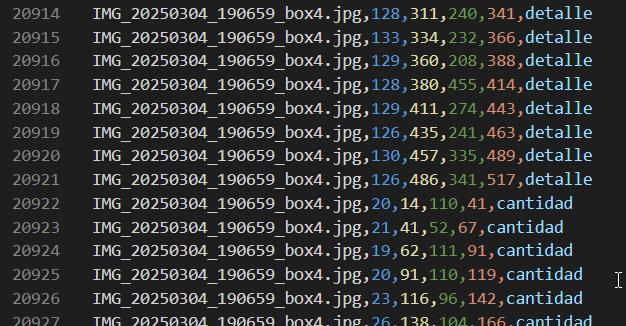
Se diseño un código que permita capturar las subáreas de interes como vemos en la **Figura 32**, que ayudara al entrenamiento de YoloV5 para reducir consumo y potencia de computo.

**Figura 32**  
*Notas de venta detección de celdas del área detalle*



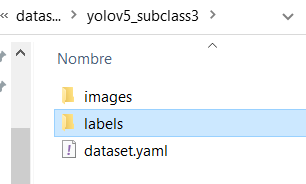
El código consiste en cargar la carpeta de imágenes class3 y crear un archivo csv (**Figura 33**), que se utilizara para generar áreas especificas de interes, para el entrenamiento, con el respectivo nombre de la imagen.

**Figura 33**  
*Áreas de detalle en csv*



Posteriormente a esto se diseño otro código en python para generar el dataset para el entrenamiento de la red neuronal como se aprecia en la **Figura 34**.

**Figura 34**  
*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.

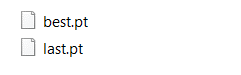
La detección es posible, pero tenemos una taza de confianza promedio aceptable como se ve en la **Figura 35**, debemos ampliar el dataset o cambiar la configuración en el aprendizaje, una vez ampliando y probando configuraciones se obtiene una confianza en la detección aceptable.

**Figura 35**  
*Detección de YoloV5 celdas detalle*



Para aprovechar el aprendizaje en colab exportamos los pesos del entrenamiento para probarlo o mejorar su entrenamiento como se ve en la **Figura 36**.

**Figura 36**  
*Pesos de entrenamiento de área detalle*

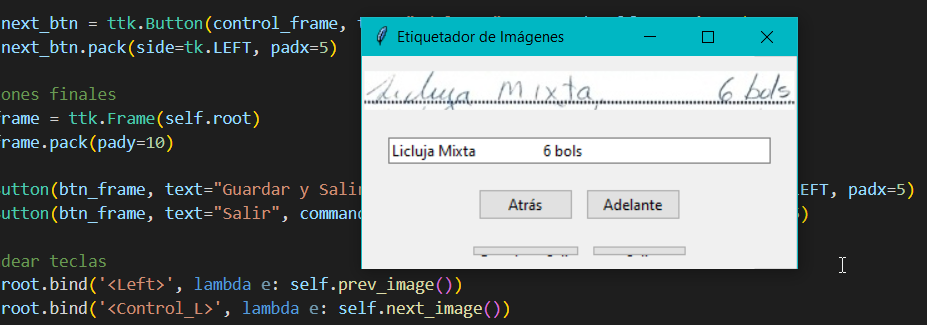


### Iteración 4.

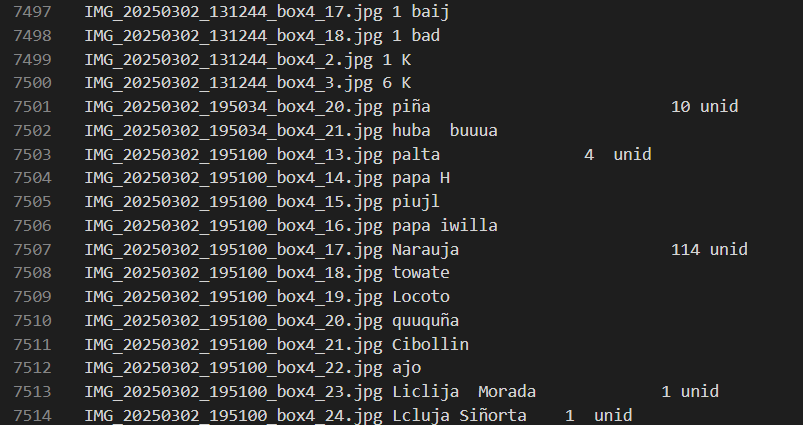
#### Códigos En Python Para Generar El Dataset De Entrenamiento Para La CRNN.

En esta sección, utilizaremos la herramienta Visual Studio Code (Vscode) para desarrollar un script en Python que nos permitirá generar un dataset (**Figura 39**), a partir de las imágenes detectadas en a la iteración anterior (**Figura 37**), donde se identificaron las celdas de detalle. A través de una interfaz gráfica, se cargará la carpeta que contiene estas imágenes, y los datos se almacenaran en un archivo .txt (**Figura 38**). Este archivo luego será procesado para su uso en el entrenamiento del modelo CRNN.

**Figura 37**  
*Captura de datos para el CRNN*

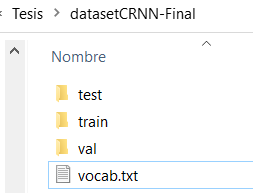


**Figura 38**  
*Archivo Txt con datos capturados*



Para aprovechar el aprendizaje en colab exportamos los pesos del entrenamiento para probarlo o mejorar su entrenamiento como se ve en la **Figura 39**.

**Figura 39**  
*Formato de dataset para entrenamiento*



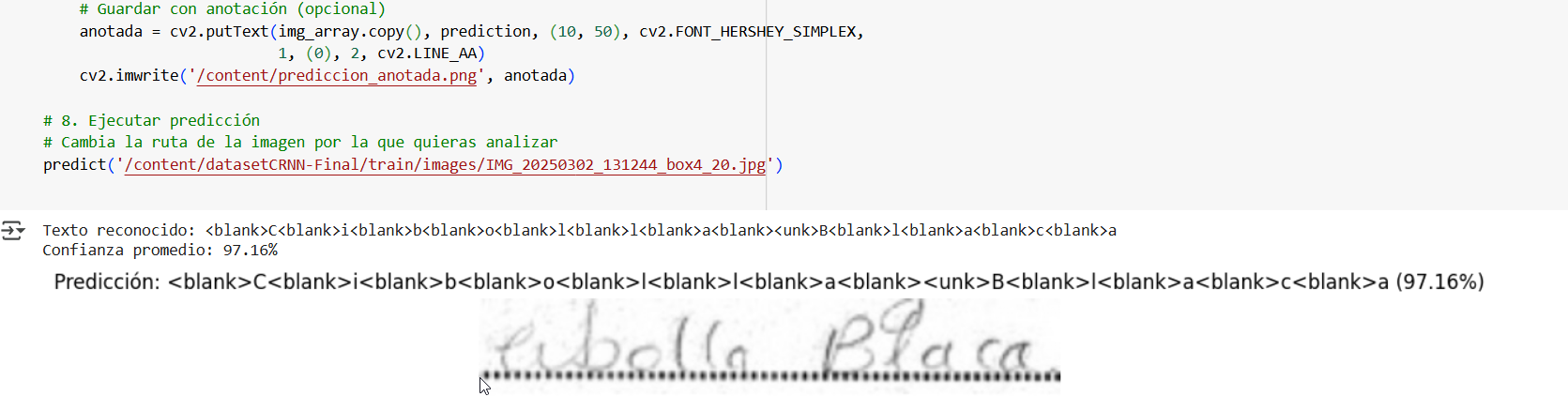
#### Entrenamiento De La CRNN ´Para La Detección De Letras Y Números Manuscritos.

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.

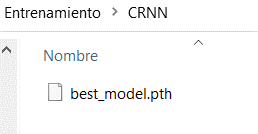
El dataset se estructura a partir de un archivo .txt que contienen las rutas de las imágenes junto a las etiquetas correspondientes (texto real). Las imágenes son procesadas para unificar su tamaño, normalizarse y transformarse en tensores. El texto se codifica utilizando un vocabulario definido previamente en vocab.txt.

El modelo CRNN está compuesto por una red convolucional para extraer características visiales y una red LSTM bidireccional para interpretar la secuencia de caracteres en las imágenes. El entrenamiento utiliza la función de perdida CTC (Connectionist Temporal Classification), adecuada para tareas de renacimiento de secuencias con alineamientos variables **Figura 40**.

**Figura 40**  
*Prueba de detección de entrenamiento de palabras en celda*



**Figura 41**  
*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.

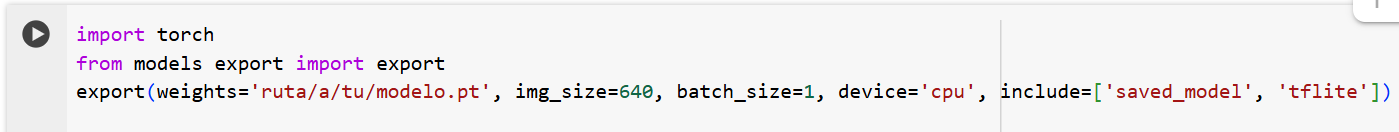
#### Conversión Y Optimización De Modelos Yolov5 Para Implementación En Dispositivos Móviles (Tflite + Kotlin).

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.

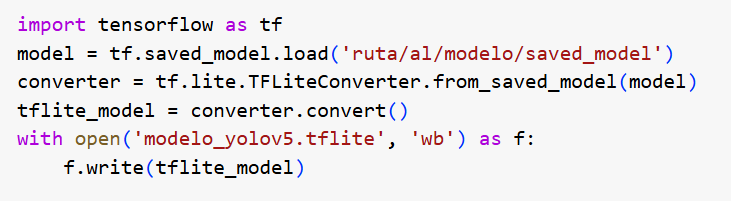
El proceso de conversión se llevó a cabo en varias etapas:

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

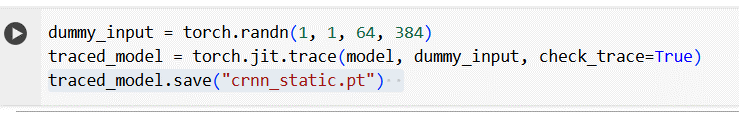
El proceso de adaptación se detectaron problemas en el modelo original que impedían una correcta conversión detallados en la **Tabla 1**.

**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 42**  
*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