



UNIVERSIDAD TÉCNICA DE AMBATO

**FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E
INDUSTRIAL**

CARRERA DE TELECOMUNICACIONES

Tema:

**SISTEMA DE DETECCIÓN TEMPRANA DE EVENTOS DELICTIVOS EN
ENTIDADES COMERCIALES CON VISIÓN ARTIFICIAL Y DEEP
LEARNING**

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la
obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Programación y redes

LÍNEA DE INVESTIGACIÓN: Tecnología de la información y Sistemas de
control

AUTOR: Christian Manuel Oyaque Moncayo

TUTOR: Ing. Edgar Patricio Córdova Córdova, Mg.

Ambato - Ecuador

febrero – 2025

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE DETECCIÓN TEMPRANA DE EVENTOS DELICTIVOS EN ENTIDADES COMERCIALES CON VISIÓN ARTIFICIAL Y DEEP LEARNING, desarrollado bajo la modalidad Proyecto de Investigación por el señor Christian Manuel Oyaque Moncayo, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, febrero 2025.



Firmado electrónicamente por:
EDGAR PATRICIO
CORDOVA CORDOVA

Ing. Edgar Patricio Córdova Córdova, Mg.
TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA DE DETECCIÓN TEMPRANA DE EVENTOS DELICTIVOS EN ENTIDADES COMERCIALES CON VISIÓN ARTIFICIAL Y DEEP LEARNING es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, febrero 2025.



Christian Manuel Oyaque Moncayo

C.C. 185006595-2

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, febrero 2025.



Christian Manuel Oyaque Moncayo

C.C. 185006595-2

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Christian Manuel Oyaque Moncayo, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE DETECCIÓN TEMPRANA DE EVENTOS DELICTIVOS EN ENTIDADES COMERCIALES CON VISIÓN ARTIFICIAL Y DEEP LEARNING, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, febrero 2025.



Firmado electrónicamente por:
FRANKLIN OSWALDO
MAYORGA MAYORGA

Ing. Franklin Oswaldo Mayorga Mayorga, Mg.
PRESIDENTE DEL TRIBUNAL



Firmado electrónicamente por:
GUSTAVO DANILO
MACHADO MERINO

Ing. Gustavo Danilo Machado Merino M.Sc.
PROFESOR CALIFICADOR



Firmado electrónicamente por:
JESUS ISRAEL
GUAMAN MOLINA

Ing. Jesús Israel Guamán Molina Mg.
PROFESOR CALIFICADOR

DEDICATORIA

A todos los estudiantes, docentes e investigadores que, con curiosidad y esfuerzo han desarrollado e impulsado el avance de la ciencia y tecnología, Su imparable búsqueda del conocimiento inspira a soñar en un futuro mejor. Gracias por su invaluable contribución!

AGRADECIMIENTO

A la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, por permitir demostrar el conocimiento adquirido en mis años de estudio, de forma justa e imparcial.

A los buenos docentes, maestros que, con paciencia y dedicación supieron transferir su conocimiento y confiar en la capacidad de sus estudiantes.

A mi familia que apoyo incondicionalmente no solo mi formación académica, sino también mi formación personal.

A mi tutor, que confió en el potencial de mi propuesta de investigación desde el principio.

A todos aquellos que contribuyeron de una u otra manera para que todo esto fuera posible.

Por último, a mí mismo, por no desperdiciar todo el apoyo antes mencionado y utilizarlo en cumplir esta meta.

ÍNDICE GENERAL DE CONTENIDOS

PORTADA.....	i
APROBACIÓN DEL TUTOR.....	ii
AUTORÍA.....	iii
DERECHOS DE AUTOR	iv
APROBACIÓN DEL TRIBUNAL DE GRADO.....	v
DEDICATORIA.....	vi
AGRADECIMIENTO	vii
ÍNDICE GENERAL DE CONTENIDOS	viii
ÍNDICE DE TABLAS.....	xii
ÍNDICE DE FIGURAS.....	xiv
ÍNDICE DE ANEXOS	xix
RESUMEN EJECUTIVO	xx
ABSTRACT	xxi
CAPÍTULO I. MARCO TEÓRICO.....	1
1.1 Tema de investigación.....	1
1.1.1 Planteamiento del problema	1
1.2 Antecedentes investigativos	3
1.3 Fundamentación teórica	5

1.3.1 Sistemas de seguridad privada	5
1.3.2 Sistemas electrónicos de seguridad	5
1.3.3 Sistemas de video vigilancia	7
1.3.4 Machine Learning	17
1.3.5 Deep Learning	34
1.3.6 Visión Artificial	47
1.3.7 Gestión de información	66
1.4 Objetivos	68
1.4.1 Objetivo general	68
1.4.2 Objetivos específicos	68
CAPÍTULO II. METODOLOGÍA	70
2.1 Materiales	70
2.2 Métodos	70
2.2.1 Modalidad de la investigación	70
2.2.2 Población y muestra	71
2.2.3 Recolección de información	71
2.2.4 Procesamiento y análisis de datos	72
CAPÍTULO III. RESULTADOS Y DISCUSIÓN	73
3.1 Situación actual	73

3.1.1 Esquema general	76
3.1.2 Requisitos del proyecto	77
3.1.3 Hardware y software	78
3.2 Datos e información disponible	85
3.2.1 Recolección y descripción.....	85
3.2.2 Exploración y selección	86
3.2.3 Verificación de calidad	88
3.3 Conjunto de datos.....	88
3.3.1 Detección de objetos sospechosos	88
3.3.2 Detección de posturas amenazantes	92
3.4 Modelado.....	94
3.4.1 Entrenamiento y validación.....	94
3.4.2 Sistema gestor de información	102
3.5 Evaluación del sistema	110
3.5.1 Pruebas en entornos reales	110
3.5.2 Métricas de validación	115
3.6 Despliegue.....	117
3.6.1 Arquitectura funcional	118
3.6.2 Presupuesto	119
CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES	121
4.1 Conclusiones	121

4.2 Recomendaciones.....	123
REFERENCIAS BIBLIOGRÁFICAS	125
ANEXOS	133

ÍNDICE DE TABLAS

Tabla 1. Tipos de cámaras en CCTV.	7
Tabla 2. Comparación de cámaras de videovigilancia.....	8
Tabla 3. Comparación de conjuntos de datos para detección de anomalías	16
Tabla 4. Comparativa de entornos Cloud para Machine Learning.	29
Tabla 5. Comparación de metodologías para proyectos de Machine Learning.	34
Tabla 6. Componentes de PyTorch	45
Tabla 7. Librerías de desarrollo en Deep Learning.....	46
Tabla 8. Parámetros de configuración de MediaPipe Pose Landmark.....	49
Tabla 9. Comparación de modelos para estimación de postura.	54
Tabla 10. Descripción de versiones de YOLO más destacadas.	57
Tabla 11. Modelos para detección de objetos.	61
Tabla 12. Algoritmos para rastreo en detección de objetos.	64
Tabla 13. Unidades de procesamiento para visión artificial.	66
Tabla 14. Entornos de desarrollo para el Sistema Gestor de Información.....	68
Tabla 15. Criterios de recolección de información.	72
Tabla 16. Descripción Anomalías UCF-Crime.....	87
Tabla 17. Características de tamaños de modelos.....	95
Tabla 18. Parámetros de entrenamiento para tamaños de modelos.	95
Tabla 19. Resultados de entrenamiento de modelos para detección de objetos.....	96
Tabla 20. Parámetros de entrenamiento SVM para clasificación de postura.....	99

Tabla 21. Costo computacional de entrenamiento.	101
Tabla 22. Detalles de pruebas realizadas en entornos reales.	111
Tabla 23. Resultados de las pruebas realizadas en entornos reales.....	115
Tabla 24. Métricas de validación del sistema.	117
Tabla 25. Costos totales de implementación del sistema.....	120

ÍNDICE DE FIGURAS

Figura 1. Arquitectura sencilla CCTV.	11
Figura 2. Arquitectura Básica de CCTV	12
Figura 3. Arquitectura CCTV con cámara PTZ.	12
Figura 4. Arquitectura CCTV sobre redes IP.	13
Figura 5. Clasificación binaria con $k = 3$	19
Figura 6. Ejemplo de un modelo de regresión lineal.	20
Figura 7. Árbol de decisión con profundidad 1.	21
Figura 8. Árbol de decisión de profundidad 2.	21
Figura 9. Algoritmo Bosque Aleatorio.	22
Figura 10. Vectores de soporte en algoritmo SVM.	23
Figura 11. Clasificación binaria con SVM.	23
Figura 12. Esquema Perceptrón básico.	24
Figura 13. Esquema Perceptrón Multicapa (MLP).	25
Figura 14. Matriz de confusión en clasificación binaria.	25
Figura 15. Matriz de confusión en clasificación multiclase.	26
Figura 16. Flujo de trabajo CRISP-DM	30
Figura 17. Flujo de trabajo ML-Canvas	32
Figura 18. Flujo de trabajo SEMMA.	33
Figura 19. Transformación de información en una red neuronal profunda.	35
Figura 20. Aprendizaje en Deep Learning.	36

Figura 21. Funcionamiento SGD.	37
Figura 22. Proceso extracción de características en CNN.	38
Figura 23. Proceso de convolución en CNN.....	39
Figura 24. Relleno o “Padding” en convolución de CNN.	40
Figura 25. Convolución con “stride=2” y “path=3”.	40
Figura 26. Extracción de características con CNN pre entrenada.....	42
Figura 27. Error de entrenamiento y validación de CNN planas.	43
Figura 28. Diagrama de bloque de aprendizaje residual.	43
Figura 29. Resultados de aplicación de bloques residuales en CNNs.....	44
Figura 30. Arquitecturas de ResNet.	44
Figura 31. Esquema de etapas de Visión Artificial.....	47
Figura 32. Puntos de referencia de MediaPipe Pose Landmark.....	50
Figura 33. Proceso de estimación de postura de OpenPose.	51
Figura 34. Puntos de clave de OpenPose.	51
Figura 35. Pipeline de AlphaPose para estimación de postura.	52
Figura 36. Arquitectura AlphaPose.....	53
Figura 37. Posiciones aproximadas de puntos clave AlphaPose.	53
Figura 38. Funcionamiento de YOLO para detección de objetos.....	55
Figura 39. Arquitectura original YOLO.....	56
Figura 40. Arquitectura Fast R-CNN.....	58
Figura 41. Mecanismos de atención en Faster R-CNN.....	59

Figura 42. Funcionamiento de SSD.	60
Figura 43. Sistemas de seguridad más utilizados entre los encuestados.	73
Figura 44. Frecuencia de monitorización entre los encuestados.	74
Figura 45. Delitos no detectados a tiempo entre los encuestados.	74
Figura 46. Limitaciones de los sistemas de seguridad entre los encuestados.	75
Figura 47. Nivel de aceptación del sistema de seguridad actual entre los encuestados.	75
Figura 48. Esquema general del sistema propuesto.	76
Figura 49. Flujo de procesamiento para etapa de reconocimiento.	80
Figura 50. Proceso de extracción de fotogramas de videos para la creación del conjunto de datos de entrenamiento de la Rama Objeto.	89
Figura 51. Etiquetado de Imágenes en plataforma RoboFlow.	90
Figura 52. Estructura del conjunto de datos en formato Yolo.	91
Figura 53. Estructura del conjunto de datos en formato COCO.	92
Figura 54. Ejemplificación visual del proceso de extracción de posturas de entrenamiento.	93
Figura 55. Resultados de entrenamiento de yolov8m con 40 épocas.	96
Figura 56. Matriz de confusión de validación del modelo yolov8m.	97
Figura 57. Curva de Precisión del modelo entrenado.	98
Figura 58. Curva de Recall de modelo entrenado.	98
Figura 59. Curva de F1-Score de modelo entrenado.	99
Figura 60. Resultados de entrenamiento SVM con diferentes parámetros.	100

Figura 61. Mapa de calor de entrenamientos con diferentes valores de gamma y C.	100
Figura 62. Versión de Python utilizado.	102
Figura 63. Paquetes instalados dentro del entorno virtual.	102
Figura 64. Configuración de firewall.	103
Figura 65. Diagrama de flujo del proceso de análisis de video	105
Figura 66. Diagrama entidad-relación.....	106
Figura 67. Función para registro en la base de datos.	106
Figura 68. Función para consulta en la base de datos.	107
Figura 69. Interfaz principal del sistema.....	107
Figura 70. Autenticación del sistema.	108
Figura 71. Historial de detecciones del sistema.	108
Figura 72. Configuración Bot Telegram.	109
Figura 73. Mensaje de alerta en entorno controlado.	110
Figura 74. Notificación de delito inminente con arma en entorno real.....	111
Figura 75. Notificación de delito inminente con cuchillo en entorno real.....	112
Figura 76. Notificación de delito inminente con arma con diferente persona en entorno real.....	112
Figura 77. Falso positivo en detección entorno real.	113
Figura 78. Detección en entorno real con dibujo de postura activado.	113
Figura 79. Notificación de sospechoso en prueba en entorno real.....	114
Figura 80. Notificación temprana del evento delictivo.....	114

Figura 81. Matriz de confusión DOS.	116
Figura 82. Matriz de confusión DPS.....	116
Figura 83. Matriz de confusión DT.....	117
Figura 84. Arquitectura funcional del sistema	118

ÍNDICE DE ANEXOS

Anexo A. Categorización de variables	133
Anexo B. Cámara IP SN-MFPT20-ICS Sinovision	135
Anexo C. Nvidia Jetson Nano Developer Kit	136
Anexo D. Pruebas del sistemas.	137
Anexo E. Diagramas del sistema	140
Anexo F. Código del sistema	142

RESUMEN EJECUTIVO

En la actualidad, los sistemas de videovigilancia de una entidad comercial dependen en gran medida de la monitorización humana, lo que eleva los gastos asumidos por la entidad para este fin. Por ello, se ha propuesto un sistema de detección temprana de eventos delictivos con visión artificial y Deep learning, que realiza dos formas de procesamiento complementarias. La primera parte utiliza YOLOv8 para detección objetos sospechosos y la segunda un algoritmo SVM que clasifica los puntos clave extraídos con MediaPipe para la detección posturas amenazantes. El sistema se ejecuta en un módulo Nvidia Jetson Nano, que procesa vídeos en tiempo real y los muestra en un sistema gestor de información desarrollado con Flask. También guarda las detecciones en una base de datos SQLite, las cuales alimentan de forma continua a un conjunto de datos nuevo etiquetado automáticamente para futuras actualizaciones del modelo, además es capaz de mostrar registros históricos de las detecciones. El sistema se evaluó en tres aspectos: detección de objetos sospechosos, con un f1-score del 87,5%; detección de posturas amenazantes, con un 74%; y detección temprana, con un 78%. Finalmente, se evidencia la importancia del procesamiento de partes complementarias, permitiendo establecer un mejor contexto de la escena. Por otro lado, la calidad y extensión del conjunto de datos de entrenamiento son cruciales para el éxito o fracaso de los modelos de visión artificial y machine learning en general, especialmente en la tarea de detección de objetos, donde se necesitan la mayor cantidad posible de imágenes de diferentes tamaños, formas y perspectivas del objeto a detectar para que el modelo generalice correctamente. En el caso de las posturas corporales, los modelos SVM funcionan bien, aunque tienen la limitación de establecer contextos espacio-temporales.

Palabras clave: Sistemas de videovigilancia, eventos delictivos, visión artificial, detección de posturas, detección de objetos.

ABSTRACT

Presently, video surveillance systems in commercial entities depend considerably on human monitoring, which results in increased costs for these entities. Consequently, a system for early detection of criminal events using computer vision and deep learning has been proposed. This system employs two complementary processing methods: the first utilizes YOLOv8 for suspicious object detection, and the second employs an SVM algorithm that classifies key points extracted with MediaPipe for threat posture detection. The system operates on an Nvidia Jetson Nano module, which processes videos in real time and displays them in an information management system developed with Flask. It also stores detections in a SQLite database, which continuously feeds into a newly automatically tagged dataset for future model updates, and is capable of displaying historical records of detections. The system's efficacy was evaluated across three key aspects: detection of suspicious objects, with an f1-score of 87.5%; detection of threatening postures, with 74%; and early detection, with 78%. The study's findings underscore the significance of processing complementary parts, facilitating the establishment of a more comprehensive contextual understanding of the scene. The quality and breadth of the training dataset are foundational for the success or failure of computer vision and machine learning models, particularly in the context of object detection, where a diverse array of images of varying sizes, shapes, and perspectives of the object to be detected is essential for the model to generalize correctly. In the case of body postures, support vector machine (SVM) models demonstrate efficacy, though they are constrained in their ability to establish spatio-temporal contexts.

Keywords: Video surveillance systems, criminal events, computer vision, pose detection, object detection.

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA DE DETECCIÓN TEMPRANA DE EVENTOS DELICTIVOS EN ENTIDADES COMERCIALES CON VISIÓN ARTIFICIAL Y DEEP LEARNING

1.1.1 Planteamiento del problema

Ecuador en los últimos años se ha convertido en uno de los países más inseguros de América Latina, se presume que estos datos son resultado de malas administraciones y el aumento del crimen organizado en general. Las inversiones y el desarrollo económico se han visto muy perjudicados a raíz de este problema, lo que empeora aún más la seguridad del país, de la mano con la falta de empleo y condiciones de pobreza extrema.

Según datos publicados por el Banco Interamericano de Desarrollo, en 2023 la tasa de homicidios por 100000 habitantes fue de 18 frente 5,6 que es el promedio mundial, es decir tres veces más que el promedio. Por otro lado, más del 50% de los homicidios están ligados al crimen organizado y afecta de manera desproporcionada a mujeres, niños, niñas y poblaciones diversas. Esto, según el BID, afecta tanto al desarrollo económico de la región, como al bienestar y calidad de vida de las personas, donde el 25% de las empresas afirman que la inseguridad es una restricción muy fuerte para hacer negocios.[1]

El ámbito local, presenta la misma tendencia, situando al Ecuador según el Índice global de crimen organizado 2023 con uno de los mayores incrementos de niveles de criminalidad de la región (+0.82). Además, el país se perfila entre los 10 países con más crimen organizado del mundo, sumado a la gobernanza de grupos mafiosos en ciudades del país como Guayaquil, Quito, Manta, entre otros, contribuyen al trágico panorama del comercio y la economía en sí.[2]

En 2023 Ecuador registro una tasa de victimización por delincuencia de 36%, siendo esta la más alta en los últimos años, además alrededor del 64% de los ecuatorianos

manifiestan sentirse inseguros en su entorno.[3] Este temor generalizado de la población ante el asedio de la delincuencia provoca que el 90% de los empresarios reporte a la extorsión como su principal problema de seguridad [4], perjudicando directamente al desarrollo económico y social del país.

Por último, el Observatorio Económico y Social de Tungurahua pública en el informe de Inseguridad y delincuencia en Ambato octubre de 2020 – marzo de 2022, que el 62.8% de los ambateños temen la inseguridad y la delincuencia, según datos encuestados, alrededor del 10% de los delitos cometidos se producen en entidades comerciales, almacenes o tiendas, y el 92.8% de habitantes no se sabe en la actuación y eficiencia de los servidores policiales. Estos datos evidencian la falta de medidas de seguridad confiables en Ambato y sus alrededores.[5]

Ambato al ser una ciudad céntrica en Ecuador, es atractiva para inversionistas y emprendedores, pero esto se ve afectado por el incremento de los índices de inseguridad en el país y el mundo, mermando el desarrollo social y económico de la región. Una de las partes más importantes para el desarrollo económico de una entidad comercial es su sistema de seguridad confiable, por ende, el departamento que maneje este sistema debe estar capacitado para prevenir, neutralizar y manejar eventos delictivos.

Un sistema de seguridad ineficiente hace que la empresa sea un blanco fácil para cometer delitos, ya que el evento puede pasar desapercibido hasta que sea neutralizado. Esto afecta a la reputación y principalmente a la economía de la entidad, que se ve obligada a correr con los gastos de indemnización y reparación de daños, lo que supone pérdidas económicas imprevistas para la empresa. Estos gastos, en muchas ocasiones, están fuera de las posibilidades económicas de la empresa, lo que puede provocar su fracaso económico.

Por último, es cierto que muchas veces no se pueden prevenir los delitos, pero, en estos casos, la presencia de personal de seguridad cualificado y preparado para evitar y neutralizar estas situaciones hace que los delitos sean más difíciles de cometer o que su finalidad se vea frustrada. Junto con tecnologías de vanguardia, el personal capacitado evita gran parte de los delitos que se cometen dentro y fuera de entidades comerciales.

1.2 Antecedentes investigativos

Se realizó una búsqueda bibliográfica de investigaciones previas en temas como Human Action Recognition (HAR) y Human Anomaly Action Detection (HAAD), los cuales contribuyeron con valiosa información de partida en el desarrollo del sistema propuesto.

En 2020, Markovits et al en la investigación “Graph Embedded Pose Clustering for Anomaly Detection” realizada en la Universidad de Tel-aviv, implementaron un modelo para detección de anomalías en acciones humanas (HAAD) basado en representaciones de poses humanas como grafos espacio temporales. Su arquitectura a breves rasgos se compone de tres etapas; la primera llamada “Pose Estimation”, encargada de obtener la estimación de postura en forma de grafos, la segunda llamada “Encoding”, aquí se procesan los grafos por medio de un ST-GCAE (Spatio-Temporal Graph Convolutional AutoEncoder), codificándolos en un espacio latente, y por último el “Clustering”, encargado de organizar las acciones en clústeres basados en sus similitudes. Además se utilizó un modelo de Mezcla de Procesos de Dirichlet (DPMM) que aprende las asignaciones suaves del clúster para calcular un puntaje de normalidad a cada acción, obteniendo en promedio un 85% de precisión en comparación al estado del arte.[6]

En la investigación “Detección de Objetos y Patrones de Movimiento Usando Redes Neuronales y Algoritmos Genéticos para la Identificación de Robo a Mano Armada”, desarrollada por Walter Iván Leturia Rodríguez y Luis Vladimir Urrelo-Huiman en la Universidad Privada Antenor Orrego Perú, el año 2021, utilizaron algoritmos basados en Redes Neuronales Recurrente con OpenCv/YOLOv3 combinados con técnicas de Algoritmos Genéticos para detectar armas, tales como armas blancas, bates de béisbol armas de fuego entre otros. Entrenaron el modelo con un data set de 100 imágenes en 3 épocas con variaciones, rotaciones y transformaciones, además utilizaron una Red Neuronal LSTM para hacer el tracking del comportamiento sospechoso, evaluando parámetros como permanencia y dirección de individuo dentro de un entorno (pedestrian-walk). Obteniendo un 96.5% de precisión en detecciones de robos a mano armada.[7]

En 2023, Yuxing Yang et al, de Intelligent Sensing and Communications Research Group, Newcastle University, en la investigación “Pose driven human activity anomaly detection in a CCTV like environment”, desarrollaron una arquitectura de tres ramas de paralelas y complementarias de procesamiento; Pose-Based Branch para analizar posturas corporales, RGB-Based Branch para suplementar las características de visuales y de apariencia que la rama de estimación de postura no toma en cuenta y por ultimo Object-Position-Based Branch la cual se encarga de la detección de objetos inusuales y su posición dentro de la escena, esto otorgó al enfoque un 95.8% de precisión, comparado con el modelos únicamente basado en estimación de postura como MLSTM-FCN y otro basado en procesamiento RGB como ResNetCRNN.[8]

En 2024, Maeda Shun et al, en la investigación “Frequency-Guided Multi-Level Human Action Anomaly Detection with Normalizing Flows”, plantearon un modelo no supervisado para detectar comportamientos semánticamente anómalos, entrenando únicamente con ejemplos de acciones predeterminadas como normales. Su enfoque fue multi-nivel mediante redes convolucionales de grafos (GCN), por un lado procesando la representación completa de la estimación de postura y por otro, la parte superior e inferior por separado, luego ambos flujos de analizaron con un perceptrón multicapa (MLP). Además utilizaron la transformada discreta Coseno (DCT) para codificar las secuencias temporales de estimación de postura en función de la frecuencia, reduciendo el ruido antes de procesar cada fotograma Por ultimo mediante Fijos de Normalización (NF) en la etapa de entrenamiento mapearon las entradas en un distribución gaussiana maximizando las probabilidades de las acciones normales y minimizando la función de pérdida, en la etapa de testeo las probabilidades de cada acción se evalúan respecto a la distribución normal, los valores fuera de este rango se etiquetan como Anómalas. Esto les di como resultado un 84.3% de precisión con HumanAct12 Data set y un 95% con UESTC datase.[9]

Erick Sandoval en la investigación “Sistema de detección de situaciones delictivas en establecimientos comerciales usando inteligencia artificial”, implemento un sistema que se compone de tres fases: adquisición de datos, procesamiento y visualización. Para la captura de imágenes en tiempo real se utiliza una cámara Rohxizw 1520p HD Wifi, cuyos datos se envían al microordenador Nvidia Jetson Nano. El procesamiento de las imágenes es realizado por el algoritmo de visión artificial Yolov8, desarrollado

en Python, encargado de identificar indicadores de robos con armas. Los resultados y alertas generadas se envían a través de Telegram, con imágenes del delito, una descripción detallada y la ubicación. El procesamiento ocurre en tiempo real, logrando una efectividad del 84% tras pruebas.[10]

1.3 Fundamentación teórica

Un sistema de seguridad inteligente capaz de detectar eventos delictivos de forma oportuna, precisa conocimientos en sistemas de videovigilancia, así como también base teórica en algoritmos de visión artificial y Deep Learning, por ende, a continuación, se presenta una síntesis de dichos conceptos. Además en el Anexo A se especifican las variables de investigación utilizadas para la fundamentación teórica.

1.3.1 Sistemas de seguridad privada

Según la Ley Orgánica de Vigilancia y Seguridad Privada las actividades que realizan los sistemas de Seguridad Privada son el conjunto de procesos, procedimientos, métodos y técnicas, que ejecutan los prestadores de servicios de seguridad privada, para cumplir su objetivo societario, para proteger personas y activos; y, velar por el normal funcionamiento de las actividades de los usuarios.[11]

1.3.2 Sistemas electrónicos de seguridad

Se define como sistema electrónico de seguridad al conjunto de equipos y elementos necesarios para salvaguardar a personas y bienes materiales de amenazas o agresiones externas. Estos equipos se controlan mediante centrales de alarmas o unidades de control que, según el tipo de situación y riesgo potencial, emiten diferentes tipos de alerta. Se clasifican en; Seguridad de Instalaciones y Seguridad Privada. [12]

a. Seguridad de instalaciones

Comprende el conjunto de equipos necesarios para salvaguardar la integridad de los bienes materiales de las instalaciones, entre los más comunes se encuentran; Sistemas de seguridad contra incendios y Sistemas de detección de gas. En muchos de los casos

su instalación es obligatoria, como en edificios públicos, hospitales, instituciones educativas entre otros.[12]

La detección temprana de un incendio es la finalidad de los sistemas de seguridad contra incendios para evitar afectaciones humanas y daños materiales en la instalación. Su principal respuesta es la alerta sonora y luminosa, pero en algunos casos también activa sistemas de extinción de incendios automatizados, cortando los suministros de electricidad y gas o comunicando la situación de emergencia a la estación de Bomberos más cercana.[12]

Los sistemas de detección de gas tienen como objetivo detectar acumulaciones de gases o vapores inflamables, riesgo de intoxicación por presencia de gases tóxicos, niveles anómalos de oxígeno entre otros. Por ende, los criterios de instalación defenderán del tipo de vapor o gas a detectar, como por ejemplo propano, butano, gas natural, dióxido de carbono, oxígeno entre otros.[12]

b. Seguridad privada

Aquí encontramos el conjunto de equipos encargados de salvaguardar la integridad de personas y bienes materiales de vulneraciones externas, como los más comunes tenemos; Sistemas antirrobo o intrusión y Sistemas de videovigilancia. Para la instalación de este tipo de sistemas existen normativa y regulaciones tanto nacionales como internacionales.[12]

Los sistemas antirrobo o intrusión se definen como el conjunto de equipos encargados de gestionar la intrusión no autorizada o allanamiento a un determinado perímetro, el robo o atraco a personas, bienes o inmuebles, el control de presencia de personas dentro de una edificación y el control de acceso registrando la entrada y salida de personas y vehículos a un terminado establecimiento.[12]

Los sistemas de video vigilancia constan equipos encargados de visualizar y almacenar imágenes captadas por cámaras de seguridad estratégicamente ubicadas para controlar y monitorear la situación en tiempo real dentro y fuera de una instalación. La mayoría se compone de cámaras, monitores, dispositivos de almacenamiento digital y otros dispositivos de tratamiento de señales de video y audio, controlados por internet.[12]

1.3.3 Sistemas de video vigilancia

Los sistemas de video vigilancia son un medio de protección activa formado por equipos de captación, visualización, control y grabación de audio y video para vigilar, gestionar y supervisar instalaciones en tiempo real, tanto interna y externamente. Muchas veces se suele interpretar los términos CCTV y sistemas de video vigilancia como similares, pero es importante definir su diferencia. Los circuitos cerrados de televisión permiten visualizar las imágenes captadas por las cámaras en tiempo real, y se convierten en sistemas de video vigilancia, capaces de almacenar y procesar grabaciones.[12]

a. Equipos y elementos

- **Cámaras**

Este es uno de los más importantes del sistema, ya que es el encargado de captar las imágenes de una zona hacia la que ha sido orientada, consta de varios elementos que definen sus propiedades de uso y funcionamiento, tales como; sensor de captación, el objetivo y la lente, carcasa externa, soporte o posicionador y accesorios complementarios. En la Tabla 1 se muestra las clasificaciones más comunes de cámaras de CCTV. [12]

Tabla 1. Tipos de cámaras en CCTV. [12]

Criterio	Clasificación
Movilidad	fijas o estáticas
	Móviles
	Domo o mini-Domo
Dimensión y apariencia externa	Estándar
	Compactas
	Bullet
Otros	Análogas
	IP
	HD

Algunas características técnicas importantes en la selección de cámaras de sistemas CCTV son las dimensiones y peso, características del sensor y lente, resolución y

sensibilidad, tensión e intensidad nominal de funcionamiento, potencia de consumo, temperatura de funcionamiento entre otras. [12]

Las cámaras son el principal componente de un sistema de videovigilancia por ende los parámetros antes mencionados deben ser comparados teniendo en cuenta la aplicación. La Tabla 2 muestra las características de algunas de las cámaras más utilizadas en circuitos CCTV.

Tabla 2. Comparación de cámaras de videovigilancia [13] [14] [15][16]

Parámetro	DS-2CE10KF0T-LFS	DS-2CE70DF0T-PF	DH-HAC-HFW1209C-LED	SN-MFPT20-ICS
Dimensiones	161.1 mm × 70 mm × 70 mm	Φ 85.02 mm × 81.25 mm	161.3 mm × 69.7 mm × 70.0 mm	105 mm × 170 mm × 55 mm
Peso	Aprox. 400 g (0.89 lb.)	Aprox. 161 g (0.36 lb)	Aprox. 200 g (0.44 lb)	Aprox. 128 g (0.3 lb.)
Sensor Óptico	3K CMOS	2 MP CMOS	1/2.8 MP CMOS	2 MP CMOS
Lente	2.8 mm 3.6 mm	2.8 mm 3.6 mm 6 mm	2.8 mm 3.6 mm	3.6mm
Angulo y alcance de captación	Pan: 0° to 360°, Tilt: 0° to 90°, Rotation: 0° to 360°	Pan: 0° to 360°, Tilt: 0° to 75°, Rotation: 0° to 360°	Pan: 0° to 360°, Tilt: 0° to 90°, Rotation: 0° to 360°	Pan: 0° to 355°, Tilt: 0° to 90°, Rotation: 0° to 360°
Resolución	2960 (H) × 1665 (V)	1920 (H) × 1080 (V)	1920 (H) × 1080 (V)	1920 (H) × 1080 (V)
Salida de Video	TVI AHD CVI CVBS	TVI AHD CVI CVBS	TVI AHD CVI CVBS	MPEG4 RTSP ONVIF
Entrada de Audio	Si	No	No	Si
Alimentación	12 VDC ± 25%	12 VDC ± 25%	12 VDC ± 30%	12 VDC
Potencia de consumo	3.9 W	2.4 W	3.1 W	2.1 W
Condiciones de uso	Temp desde -40 °C a 60 °C Humedad máxima de 90% (Sin condensación)	Temp desde -40 °C a 60 °C Humedad máxima de 90% (Sin condensación)	Temp desde -40 °C a 60 °C Humedad máxima de 95% (Sin condensación)	Temp desde -10 °C a 40 °C Humedad máxima de 90% (Sin condensación)
Fabricante	HIKVISION	HIKVISION	DAHUA	SINOVISION
Costo	\$ 54.99	\$ 30.90	\$ 26.50	\$ 29.36

- ***Medios de transmisión***

Comprende el medio sobre el cual se transmite las señales de audio y video, pueden ser cableados mediante par trenzado, cable coaxial o fibra óptica, o inalámbricos por medio de ondas electromagnéticas o tecnología WI-FI.[12]

- ***Dispositivos de reproducción***

Son dispositivos, también llamados transductores, capaces de recuperar las señales eléctricas de las cámaras y convertirlas en imágenes, para visualizar y monitorizar la escena en tiempo real. Antiguamente se usaban reproductores en blanco y negro, actualmente son equipos sofisticados de gran resolución llamados monitores. Los tipos más comunes de monitores son [12]:

- Monitores Analógicos convencionales (CRT)
- Monitores Digitales tipo Liquid Crystal Display (LCD)
- Monitores Digitales TFT
- Monitores Digitales LED
- Monitores Plasma

Estos son los parámetros más importantes para elegir un monitor de visualización CCTV. [12]

- Tamaño
- Resolución
- Brillo
- Entradas/Salidas
- Frecuencia de refresco
- Puertos
- Capacidad de reproducción de audio

- Tensión e intensidad nominal de funcionamiento
- Potencia de consumo

- ***Dispositivos y sistemas de control***

Es imprescindible tener un sistema de control en sistemas CCTV que permita gestionar las cámaras y las señales correspondientes, los más utilizados en la actualidad son sistemas de telemetría y distribuidores y matrices de video.[12]

Sistemas de telemetría. Método de señalización electrónica para control remoto de determinados parámetros y funciones de las cámaras como control de movimiento horizontal y vertical, zoom o ambos (PTZ) [12].

Distribuidores de video. Son aquellos que permiten conmutar varias señales de video a una o varios monitores, multiplicando y amplificando las salidas de video.[12]

Matrices de video. Dispositivo electrónico que permite la gestión de varias entradas y salidas de video y audio hacia monitores conectados.[12]

- ***Dispositivos de grabación***

Son dispositivos que permiten almacenar la información visual captada por las cámaras, además pueden reproducirlas después para supervisar eventos producidos durante el momento de monitorización o si no han sido revisados por personal de supervisión. La clasificación de tecnologías de grabación más común es; Grabadores sobre cinta de video magnética (VCR) y Grabadores digitales como DVR o NVR.[12]

A continuación, los parámetros más importantes a la hora de elegir un dispositivo de grabación CCTV.[12]

- Dimensiones y peso
- Capacidad del disco de almacenamiento
- Numero de canales
- Formato de codificación de video

- Formato de compresión de audio y video
- Niveles de calidad de grabación de video
- Velocidad de actualización de grabación
- Resolución de grabación
- Posibilidad de conexión TCP/IP
- Tensión e intensidad nominal de funcionamiento
- Potencia de consumo

b. Arquitecturas

La arquitectura de un circuito cerrado de televisión depende de la infraestructura en la que se vaya a implementar, todas se rigen por modelos estandarizados, manteniendo los mismos elementos ya antes detallados. Entre las más comunes encontramos las siguientes.

La arquitectura más básica consiste en conectar directamente la cámara al monitor de visualización, esto solo permitiría monitorizar únicamente la zona en la que enfoca la cámara. Por esta razón un sistema básico de CCTV requiere por lo menos 4 cámaras, un dispositivo controlador, un monitor y un equipo de grabación, estableciendo 4 puntos diferentes puntos de visión además de la posibilidad de grabar las imágenes captadas. La Figura 1 esquematiza una arquitectura sencilla de CCTV. [17]

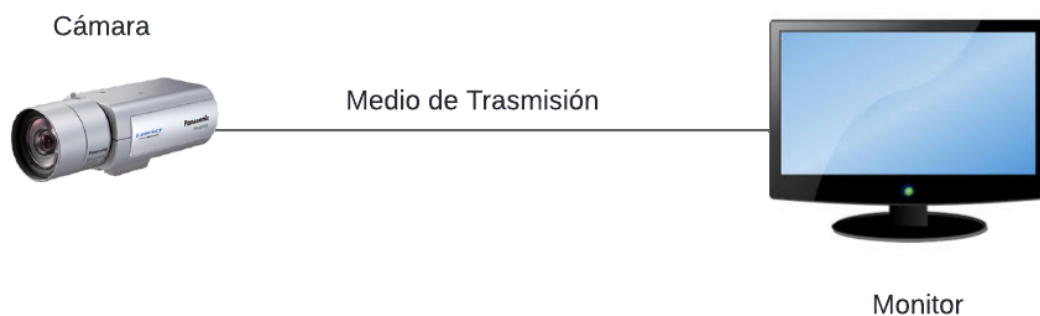


Figura 1. Arquitectura sencilla CCTV. [17]

La Figura 2 esquematiza una arquitectura básica con cámaras fijas.

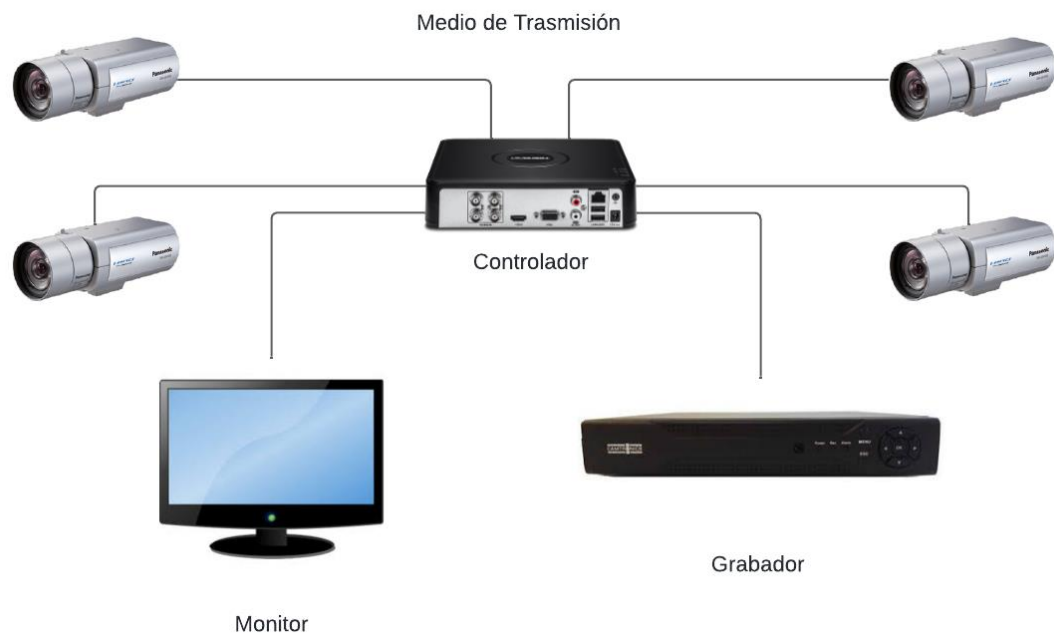


Figura 2. Arquitectura Básica de CCTV [17]

La Figura 3. Arquitectura CCTV con cámara PTZ. [17] esquematiza una arquitectura básica con cámara móvil

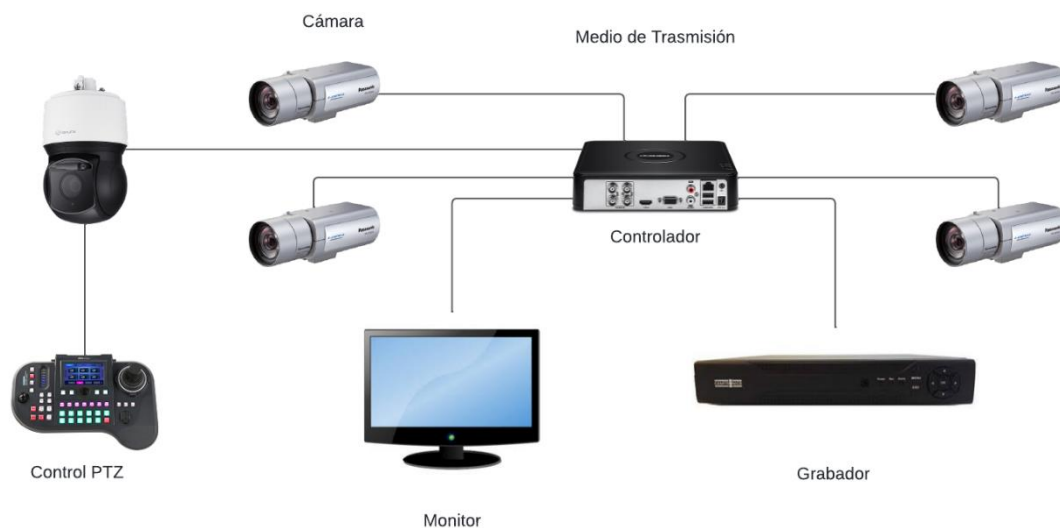


Figura 3. Arquitectura CCTV con cámara PTZ. [17]

Si el sistema se instala sobre redes IP, las cámaras IP se conectan directamente a la red [17], mediante un dispositivo conmutador (switch), además de un dispositivo controlador, que en la actualidad puede ser un ordenador con un software específico. Inclusive se puede conectar cámaras IP inalámbricas, mediante un “access point” conectado a la red. La Figura 4 esquemática una arquitectura básica de CCTV sobre IP.

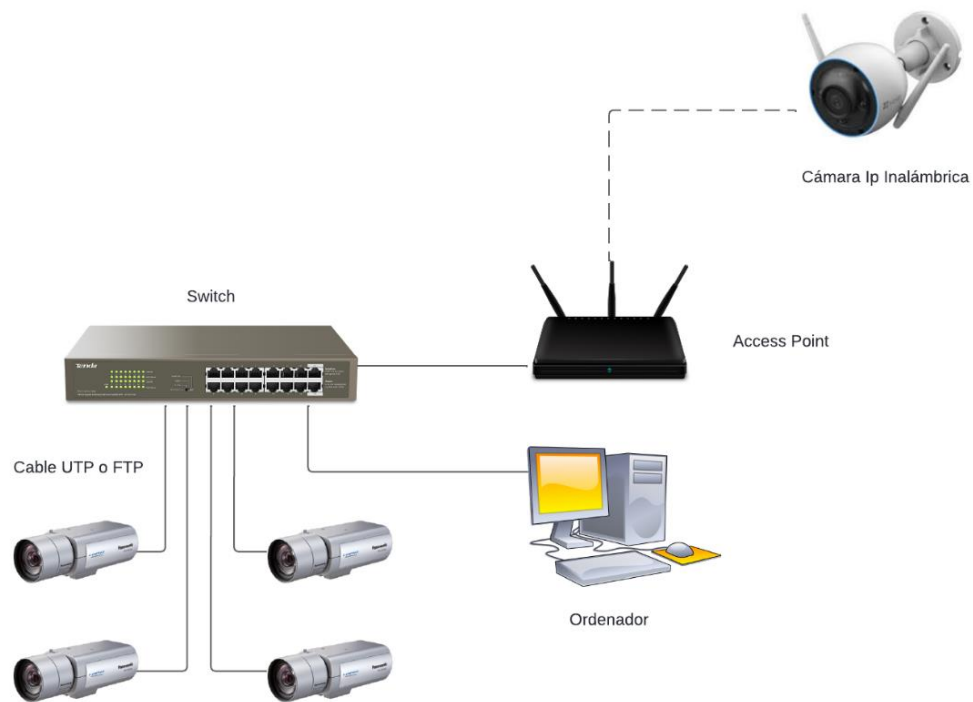


Figura 4. Arquitectura CCTV sobre redes IP. [17]

c. Normativa

Según la Ley Orgánica de Vigilancia y Seguridad Privada, publicada en el Registro Oficial en febrero de 2024 establece un marco regulador estricto para garantizar que las empresas y organizaciones que operan sistemas de vigilancia lo hagan dentro de los parámetros legales, protegiendo tanto la seguridad de las instalaciones como los derechos de las personas, principalmente en lo que respecta a la privacidad y protección de datos. [11]

Uno de los aspectos clave de la normativa es que las empresas que instalen y operen sistemas de videovigilancia deben estar debidamente registradas y certificadas por las

autoridades correspondientes. Este registro asegura que las empresas cumplen con estándares técnicos y de calidad para la instalación y mantenimiento de estos sistemas. Además, la ley exige que las cámaras instaladas cubran únicamente áreas necesarias para la seguridad y no invadan espacios privados sin autorización, lo cual es fundamental para respetar el derecho a la intimidad de las personas. [11]

La ley establece que las grabaciones de CCTV deben almacenarse seguramente y por un tiempo determinado, para estar disponibles solo para propósitos de seguridad y bajo las condiciones legales establecidas. Las organizaciones tienen la obligación de informar a las personas cuando están siendo grabadas mediante la señalización adecuada en los espacios donde se encuentran las cámaras. [11]

Finalmente, la normativa dicta que cualquier uso indebido de imágenes o grabaciones, como la divulgación no autorizada o el uso con fines distintos a la seguridad, puede conllevar sanciones severas para las empresas responsables y los involucrados. Este marco normativo refuerza la importancia del manejo responsable de los sistemas de vigilancia en Ecuador, asegurando un equilibrio entre la seguridad y los derechos fundamentales de las personas. [11]

d. Conjuntos de datos públicos

- ***UCF-Crime***

Este conjunto de datos fue creado por “UCF Center for Research in Computer Vision” para evaluar el método presentado en la investigación “*Real-world Anomaly Detection in Surveillance Videos*”, consta de un conjunto amplio de videos de vigilancia sin recortar que cubren 13 anomalías del mundo real, incluyendo abuso, arresto, incendio provocado, asalto, accidente de tránsito, explosión, pelea, robo, tiroteo, hurto, hurto en tiendas y vandalismo. Estas anomalías fueron seleccionadas ya que tienen un impacto significativo en la seguridad pública. [18]

- ***UBI-Fights***

El conjunto de datos UBI-Fights fue introducido en la investigación “*Human Activity Analysis: Iterative Weak/Self-Supervised Learning Frameworks for Detecting Abnormal Events*” y contiene de 80 horas de vídeo con alrededor de 1000 videos,

donde 216 videos contienen un evento de peleas, y 784 son situaciones normales de la vida diaria. Todos los segmentos de vídeo innecesarios (por ejemplo, introducciones de vídeo, noticias, etc.) que pudieran perturbar el proceso de aprendizaje fueron eliminados. Todos los videos están redimensionados a 640x360 píxeles, y la velocidad de fotogramas se configuraba a 30 fps.[19]

- ***ShanghaiTech Campus***

ShanghaiTech es un conjunto de datos de 437 videos de vigilancia del campus. Tiene 130 eventos anormales repartidos en 13 escenas, por otro lado, todos los videos anormales están en el conjunto de pruebas, ya que el conjunto de datos se propone para aplicaciones de aprendizaje sin supervisión. [20] Fue creado para la investigación “*A Revisit of Sparse Coding Based Anomaly Detection in Stacked RNN Framework*”, además contiene más de 270.000 fotogramas de entrenamiento.[21]

- ***UBnormal***

UBnormal es un conjunto de datos compuesto por múltiples escenas virtuales para la detección de anomalías de vídeo. A diferencia de los conjuntos de datos existentes, este conjunto de datos contiene eventos anormales anotados a nivel de píxeles durante el momento de la creación. Por otro lado, para preservar la formulación típica de conjunto abierto, incluyen conjuntos separados de tipos de anomalías en colecciones de videos de capacitación y prueba. Además permite una comparación directa y justa entre modelos de conjunto abierto de una clase y modelos de conjunto cerrado supervisados.[22]

- ***X-MAN dataset***

X-MAN (eXplanations of Multiple Sources de ANomalas) es conjunto de datos complementario a los ya existentes para detección de anomalías públicas y proporciona etiquetas que describen la razón detrás de un evento anómalo. Contiene tres clases posibles de etiquetas gruesas: "objeto", "acción" y "ubicación". Captan la razón de alto nivel detrás de la clasificación de un evento como anómalo. Esto ya que cubren todas las anomalías en los conjuntos de datos de anomalías públicas y cualquier ejemplo obvio de la vida real.[23]

- **CUHK Avenue**

El conjunto de datos Avenue Dataset contiene 16 videos de entrenamiento y 21 video de prueba. Los videos se capturan en la avenida del campus CUHK con un total de 30652 fotogramas de los cuales 15328 son de entrenamiento y 15324 para pruebas. Este conjunto de datos fue desarrollado para la investigación "*Abnormal Event Detection at 150 FPS in Matlab*".[24] Hay 14 eventos inusuales que incluyen correr, arrojar objetos y holgazanear.[25] La Tabla 3 compara los conjuntos de datos explicados anteriormente mediante parámetros como enfoque o tipo de anomalía.

Tabla 3. Comparación de conjuntos de datos para detección de anomalías

Parámetro	UCF-Crime	UBI-Fights	Shanghai Tech	UBnormal	X-MAN Dataset	CUHK Avenue
Enfoque	Crímenes y anomalías en videos de vigilancia	Peleas en videos	Anomalías en videos de vigilancia peatonal	Anomalías en entornos variados	Anomalías en personas y grupos	Anomalías peatonales
Número de videos	1900	1000	437	543	1000	37
Duración promedio	~1-15 minutos	1-2 minutos	1-3 minutos	1-5 minutos	1-3 minutos	1-2 minutos
Etiquetas	Nivel de video y fotograma en los videos de prueba	Nivel de video	Anotaciones a nivel de fotograma	Nivel de fotograma y acción	Nivel de fotograma y eventos	Nivel de fotograma (comportamientos anómalos)
Entorno	Entornos comerciales, calles, estacionamientos	Calles, plazas, espacios públicos	Campus universitario y calles	Calles, áreas peatonales, estaciones	Entornos urbanos y públicos	Campus universitario (peatonal)
Tipos de anomalías	Robos, vandalismo, asaltos, accidentes, peleas	Peleas, altercados físicos	Gente corriendo, arrojar objetos, comportamiento inusual	Robos, agresiones, situaciones sospechosas	Peleas, robos, vandalismo	Personas corriendo, arrojar objetos, caminar erráticamente
Etiquetado	Etiquetado global y temporal	Binario	Rango temporal anomalías a nivel de fotograma	Marcado por tipo de anomalía	Binario	Rango temporal a nivel de fotograma
Multitudes	Ocasionalmente	No frecuente	No relevante	Moderado	Presente	Escenario peatonal
Tamaño	~128 GB	~13 GB	~7 GB	~15 GB	-	~1 GB

1.3.4 Machine Learning

El aprendizaje automático surge de la pregunta: ¿podría una computadora ir más allá de lo programado y aprender por sí sola cómo realizar una tarea específica? En lugar de que los programadores diseñen sistemas de procesamiento de datos con reglas a mano, ¿podría una computadora aprenderlas automáticamente al observar los datos? Esta pregunta abre la puerta a un nuevo paradigma de programación. [26]

En la programación clásica, el paradigma programación simbólica, los humanos ingresan las reglas en forma de programa y los datos son procesados de acuerdo con estas reglas para obtener la salida esperada. Con Machine Learning, los humanos ingresan datos, así como las salidas esperadas para estos datos, y el algoritmo mediante entrenamiento aprende las reglas. Estas reglas se pueden aplicar luego a nuevos datos para producir salidas a datos nunca antes vistos en entrenamiento. [26]

a. Métodos de aprendizaje

Las formas de aprendizaje que siguen los algoritmos de aprendizaje automático generalmente se dividen en cuatro categorías; Aprendizaje supervisado cuando los datos de entrenamiento son etiquetados en su totalidad por humanos, Aprendizaje no supervisado en la que el algoritmo aprende de los datos sin etiquetar, Semi supervisado cuando los datos están parcialmente etiquetados y Aprendizaje por refuerzo cuando el algoritmo aprende mediante incentivos.[26]

- ***Aprendizaje supervisado***

Consiste en proporcionar al algoritmo pares de entradas y salidas deseadas los datos etiquetados generalmente por humanos, y el algoritmo encuentra cómo producir la salida deseada dada una entrada. Después, el algoritmo es capaz de proporcionar una salida para una entrada nunca vista sin ayuda de un humano, las principales aplicaciones de este tipo de aprendizaje son tareas de clasificación y regresión. [27]

- ***Aprendizaje no supervisado***

En el aprendizaje no supervisado, sólo se conocen los datos de entrada y no se proporcionan datos de salida conocidos al algoritmo.[27] El modelo aprende a

categorizar o encontrar relaciones entre los datos de entrada para producir una salida. Son muy útiles en compresión o eliminación de ruido, agrupamiento y categorización de datos o simplemente para comprender mejor las correlaciones presentes en los datos entrada.[26]

- ***Aprendizaje semi supervisado***

El aprendizaje semi supervisado es un aprendizaje supervisado con datos parcialmente etiquetados o sin etiquetas anotadas por humanos, puede considerarse como un aprendizaje supervisado sin ningún ser humano involucrado. Todavía hay etiquetas involucradas, pero se generan a partir de los datos de entrada, generalmente usando un algoritmo heurístico.[26]

- ***Aprendizaje por refuerzo***

En este tipo de aprendizaje un agente recibe información sobre su entorno y aprende a elegir acciones que maximizará alguna recompensa.[26] Este tipo de aprendizaje es utilizado principalmente en problemas o tareas de optimización, así como videojuegos o simulaciones de comportamiento.

b. Algoritmos

Existen muchos algoritmos y modelos que son desarrollados en base a los diferentes tipos de aprendizaje. El aprendizaje supervisado se utiliza siempre que queremos predecir un determinado resultado a partir de una entrada determinada, entrenando mediante ejemplos de pares de entrada/salida. El objetivo es hacer predicciones precisas para datos nuevos, nunca antes vistos. A menudo requiere un gran esfuerzo humano para construir el conjunto de datos de entrenamiento.[27]

Hay dos tipos principales de problemas de aprendizaje automático supervisados, llamados clasificación y regresión. En la clasificación, el objetivo es predecir una etiqueta de clase. La clasificación a veces se divide en clasificación binaria, cuál es el caso especial de distinguir entre exactamente dos clases y clasificación multiclase, que es la clasificación entre más de dos clases. Para las tareas de regresión, el objetivo es predecir un número continuo o un punto flotante. [27]

- ***k-Nearest Neighbors (KNN)***

El algoritmo k-NN consiste únicamente en almacenar el conjunto de datos de entrenamiento. Para hacer una predicción para un nuevo punto, el algoritmo encuentra los puntos más cercanos en el conjunto de datos de entrenamiento. En lugar de considerar sólo el vecino, también se puede considerar un número arbitrario, k , de vecinos. Cuando se considera más de un vecino, contamos cuántos vecinos pertenecen a cada clase, luego se asigna la clase que es más frecuente es decir, la clase mayoritaria entre los k vecinos más cercanos. La Figura 5 muestra un ejemplo de clasificación binaria con $k = 3$. [27]

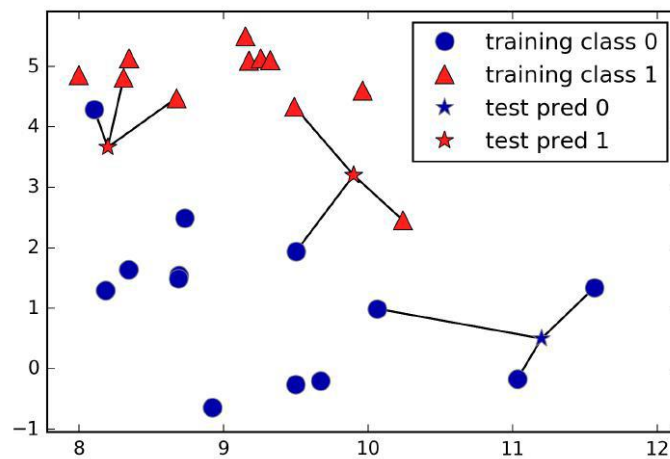


Figura 5. Clasificación binaria con $k = 3$. [27]

- ***Modelos lineales***

Los modelos lineales hacen una predicción utilizando una función lineal de las características de entrada. Para regresión, la fórmula de predicción general para un modelo lineal es la siguiente:

$$y = w[0] * x[0] + w[1] * x[1] + w[0] * x[0] + \dots + w[p] * x[p] + b \quad (1)$$

Donde $x[0]$ hasta $x[p]$ son las características $w[0]$ es la pendiente y b es el desplazamiento del eje y. Para más funciones, w contiene el pendientes a lo largo de cada eje de característica. Alternativamente, se puede pensar en la respuesta prevista. como una suma ponderada de las características de entrada, con ponderaciones que pueden ser negativas dadas por las entradas de w . La Figura 6 muestra un ejemplo de regresión lineal [27]

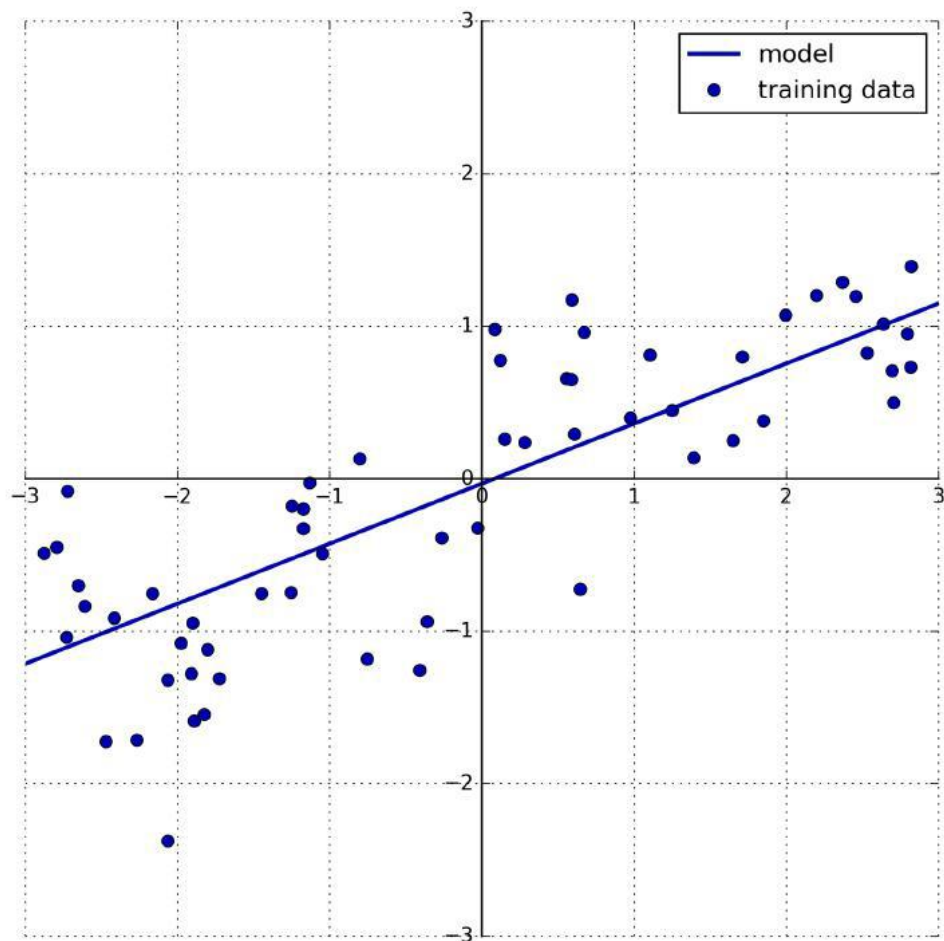


Figura 6. Ejemplo de un modelo de regresión lineal. [27]

- ***Clasificadores de Bayes***

Los clasificadores de Bayes son una familia de clasificadores que son bastante similares a los modelos lineales. Sin embargo, tienden a ser incluso más rápidos en entrenamiento. Esto debido a que aprenden parámetros mediante extracción cada característica individualmente y recopilación estadísticas simple por clase de cada característica. Estos modelos a menudo proporcionan un rendimiento de generalización ligeramente peor que el de clasificadores lineales como Regresión Logística o SVC lineal. [27]

- ***Árboles de decisión***

Los árboles de decisión son modelos ampliamente utilizados para tareas de clasificación y regresión. Aprenden una jerarquía de preguntas condicionales, lo que los lleva a una decisión. Para construir un árbol, el algoritmo busca entre todas las

pruebas posibles y encuentra la que es más informativa sobre la variable objetivo. La Figura 7 muestra un ejemplo de clasificación binaria con un árbol de decisión de profundidad 1. [27]

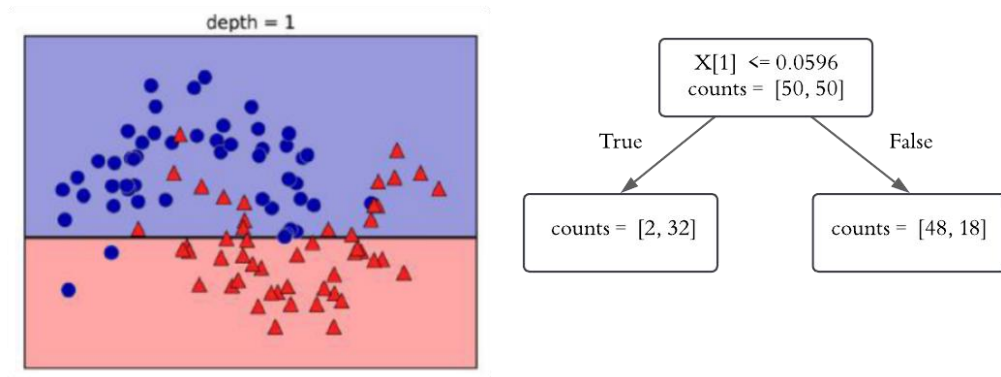


Figura 7. Árbol de decisión con profundidad 1. [27]

En el ejemplo anterior la clasificación se realiza mediante un condicional que evalúa si el dato de entrada es inferior o igual a 0.0596, en entrenamiento, si es verdadero se contabiliza 2 puntos de clase azul y 32 de clase roja, si es falso se obtiene 48 de clase azul y 18 de clase roja. La Figura 8 de muestra el mismo ejemplo pero con profundidad 2, haciendo divisiones verticales después del primer condicional haciendo posible una mejor clasificación. [27]

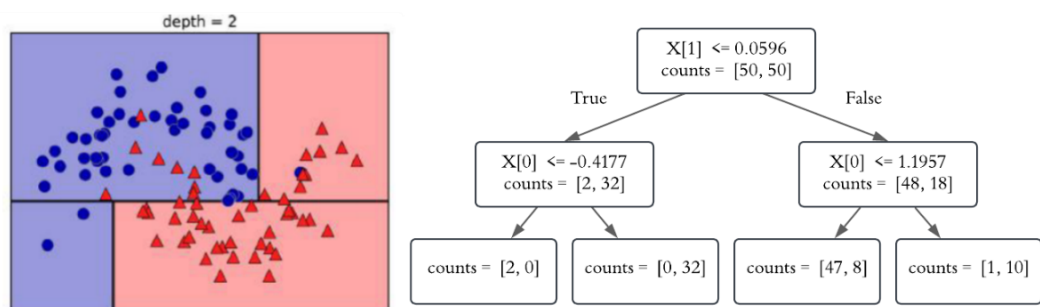


Figura 8. Árbol de decisión de profundidad 2. [27]

Un árbol de decisión tiende a caer el sobreajuste, para abordar este problema se tiene los algoritmos de bosque aleatorio que son una colección de árboles de decisión, donde cada árbol es ligeramente diferente de los demás. La idea detrás de los bosques

aleatorios es que cada árbol podría hacer relativamente un buen trabajo de predicción, pero probablemente se sobre ajustará en parte de los datos. Si construimos muchos árboles, todos los cuales funcionan bien y se adaptan de diferentes maneras, podemos reducir la cantidad de sobreajuste promediando sus resultados. Como se ejemplifica en la Figura 9. [27]

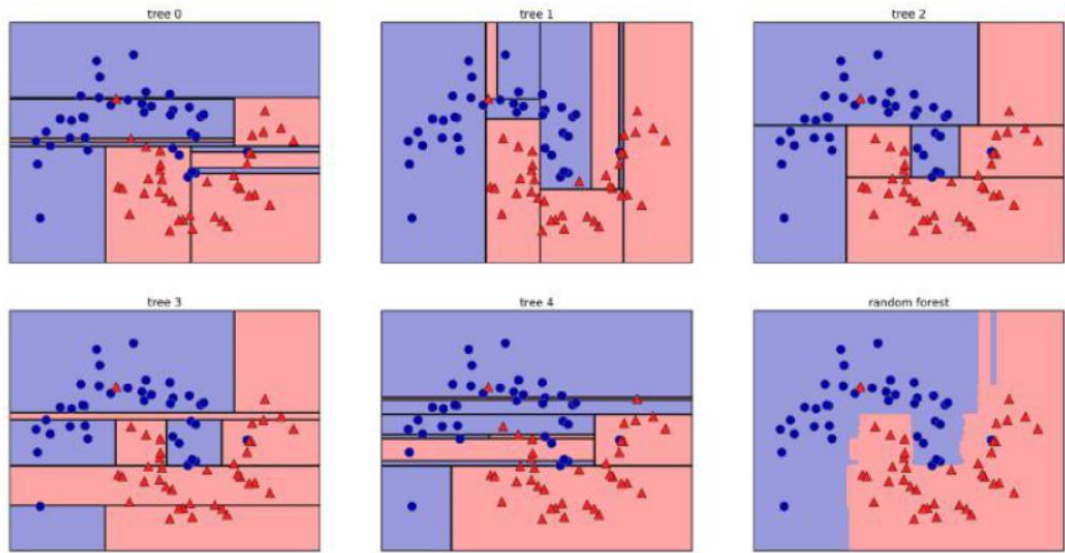


Figura 9. Algoritmo Bosque Aleatorio. [27]

- **Support Vector Machine (SVM)**

Estos algoritmos son una extensión de los modelos lineales para clasificación que permite modelos más complejos que no están definidos simplemente por hiperplanos en el espacio de entrada. Durante el entrenamiento, SVM aprende cuán importante es cada uno de los puntos de datos de entrenamiento para representar el límite de decisión entre dos o más clases. Normalmente sólo un subconjunto de los puntos de entrenamiento importa para definir el límite de decisión llamados vectores de soporte.

Para hacer una predicción para un nuevo punto se toma una decisión de clasificación en función de las distancias al vector de soporte, y la importancia de los vectores de soporte que se aprendieron durante el entrenamiento. La distancia entre puntos de datos se mide mediante el núcleo gaussiano (rbf).[27]

$$K_{rbf}(x_1, x_2) = e^{\gamma \|x_1 - x_2\|^2} \quad (2)$$

Donde x_1 y x_2 son puntos de datos, $\|x_1 - x_2\|$ es la distancia euclidiana y γ (gamma) es un parámetro que controla el ancho del núcleo gaussiano. La Figura 10 muestra gráficamente el funcionamiento de SVM. [27]

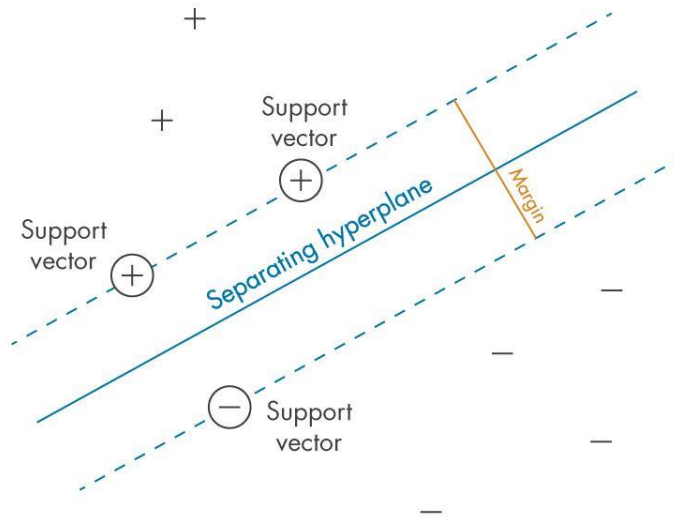


Figura 10. Vectores de soporte en algoritmo SVM. [28]

Para entrenar un modelo SVM también se puede definir un parámetro C de regularización, similar al utilizado en los modelos lineales. Limita la importancia de cada punto. La Figura 11 ejemplifica la clasificación de dos clases mediante SVM con diferentes parámetros. [27]

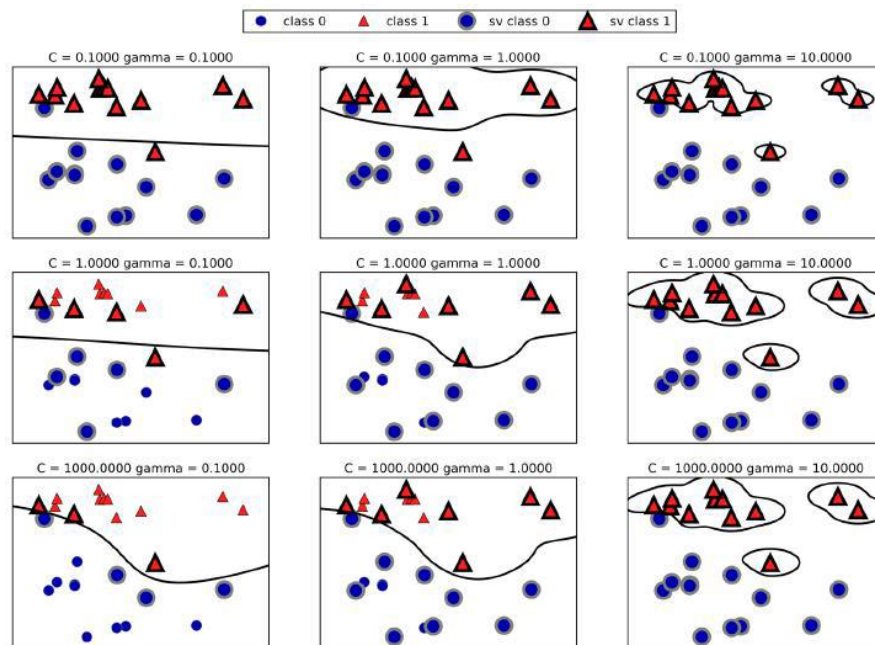


Figura 11. Clasificación binaria con SVM. [27]

- **Redes neuronales**

Los algoritmos conocidos como redes neuronales han adquirido una notable popularidad en el campo del Deep Learning. En secciones posteriores, se abordará con mayor profundidad este tema. Los perceptrones multicapa, utilizados en la clasificación y regresión, constituyen una base para métodos más complejos. Estos perceptrones multicapa, también denominados redes neuronales de retroalimentación, han desempeñado un papel significativo en el desarrollo de la IA. [27]

El perceptrón puede verse como generalizaciones de modelos lineales que realizan múltiples etapas de procesamiento para llegar a una decisión. Recordando la ecuación (1), la salida de la función lineal es una suma ponderada de las características de entrada $x[0]$ a $x[p]$, ponderada por los coeficientes aprendidos $w[0]$ a $w[p]$. La Figura 12 muestra esta idea visualmente. [27]

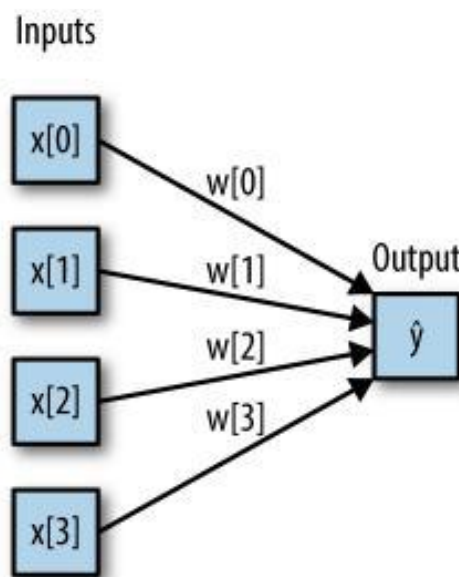


Figura 12. Esquema Perceptrón básico. [27]

Cada nodo de la capa de “Input” representa una característica de entrada, las líneas de conexión representan los coeficientes aprendidos, y el nodo “Output” representa la salida, que es una suma ponderada de las entradas. En un Perceptrón multicapa, este proceso de calcular sumas ponderadas se repite varias veces, primero calcular unidades ocultas que representan un paso de procesamiento intermedio, que son nuevamente combinados usando sumas ponderadas para producir el resultado final. La Figura 13 muestra este proceso.[27]

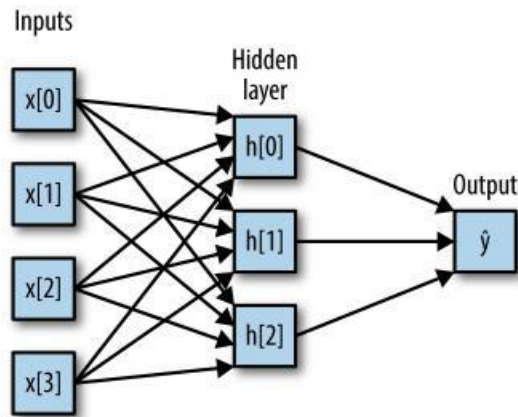


Figura 13. Esquema Perceptrón Multicapa (MLP). [27]

c. Métricas de evaluación de modelos

Al seleccionar una métrica de evaluación, siempre debe tener en mente el objetivo final de la aplicación, no solo pensando en hacer predicciones de forma precisas, sino también utilizar dichas predicciones como parte de un proceso de toma de decisiones más amplio. A continuación se detallan las métricas de evaluación más utilizadas en modelos de Machine Learning en tareas como clasificación y regresión. [27]

- **Matriz de confusión**

Una de las formas más completas de representar el resultado de la evaluación de la clasificación es mediante matrices de confusión, donde las filas corresponden a las clases verdaderas y las columnas corresponden a las clases predichas. Cada entrada cuenta la frecuencia con la que una muestra que pertenece a la clase correspondiente a la fila se clasificó como la clase correspondiente a la columna. La Figura 14 muestra la matriz de confusión de un modelo clasificación binaria. [27]

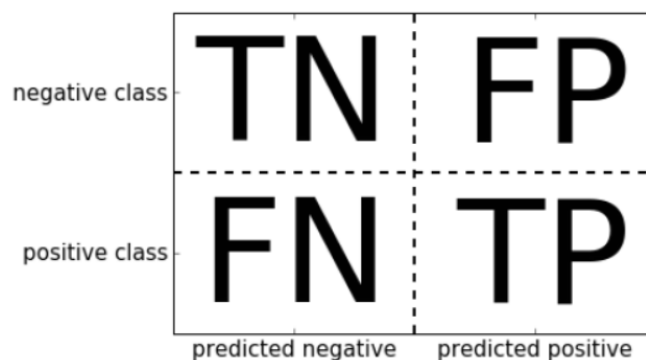


Figura 14. Matriz de confusión en clasificación binaria. [27]

Para el caso de clasificación multiclase, es exactamente la misma con diferencia en la dimensión de la matriz, siendo el número de clase para las que el modelo es capaz de clasificar, se desea que la mayor concentración de predicciones se concentre en la diagonal principal de la matriz. En la Figura 15 se muestra una matriz de confusión para un modelo clasificador de dígitos 0-9. [27]

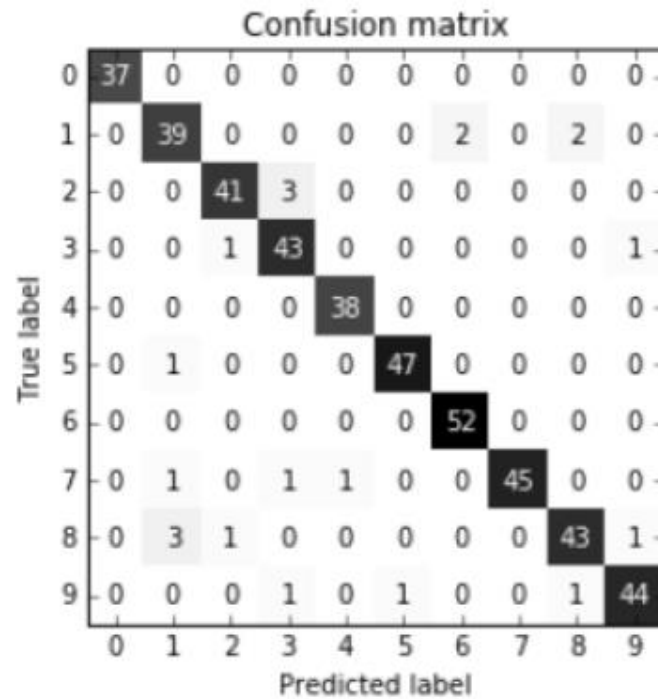


Figura 15. Matriz de confusión en clasificación multiclase. [27]

- **Accuracy**

La exactitud es la cantidad de predicciones correctas dividida por la cantidad de todas las muestras es decir todas las entradas de la matriz de confusión sumadas. [27] Puede ser expresada de la siguiente manera:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

Donde:

TP: Verdaderos positivos

TN: Verdaderos Negativos

FP: Falsos Positivos

FN: Falsos Negativos

- **Precision**

La precisión mide cuántas de las muestras predichas como positivas son realmente positivas, se utiliza como una métrica de rendimiento cuando el objetivo es limitar la cantidad de falsos positivos. La precisión también se conoce como valor predictivo positivo (VPP). [27] Puede calcularse de la siguiente manera:

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

- **Recall**

La recuperación o “recall” mide cuántas de las muestras positivas son capturadas por las predicciones positivas, se utiliza como métrica de rendimiento cuando necesitamos identificar todas las muestras positivas; es decir, cuando es importante evitar los falsos negativos. También se conoce como sensibilidad, tasa de aciertos o tasa de verdaderos positivos (TPR). [27] Se puede calcular de la siguiente manera:

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

- **F-Score**

La precisión y la recuperación son medidas muy importantes, pero miden puntos de vistas específicos del modelo, una forma de unificar estas métricas es la medida *f* también conocida como “F-score, que es la media armónica de la precisión y la recuperación. “F1 score” es una de las variantes más utilizadas. [27] Se puede calcular de la siguiente manera:

$$f1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (6)$$

d. Entornos Cloud

El entrenamiento de un modelo de machine learning requiere de grandes capacidades computacionales, las cuales nos siempre están disponibles de forma física, es por esto que existen alternativas de Cloud Computing para el desarrollo, entrenamiento y

despliegue de modelos. A continuación, se detallan las principales plataformas disponibles.

- ***Amazon Web Service***

Es una de las principales plataformas de Cloud Computing en el mundo, ofrece más de 200 servicios integrales a nivel mundial, ofreciendo desde tecnologías de infraestructura como cómputo, almacenamiento y bases de datos hasta tecnologías emergentes como Machine Learning e inteligencia artificial. Amazon SageMaker es uno de los productos ofertados por AWS para aplicaciones de Machine Learning. [29]

Este es un servicio totalmente administrado que reúne un amplio conjunto de herramientas de alto rendimiento y bajo costo para el desarrollo de soluciones de Machine Learning. Suede crear, entrenar e implementar modelos a escala con herramientas como Notebooks, depuradores, generadores de perfiles, canalizaciones, MLOP y más, todo en un entorno de desarrollo integrado (IDE). [30]

SageMaker ofrece acceso a cientos de modelos pre entrenados, incluidas máquinas virtuales disponibles públicamente, que puede implementar fácilmente. El nivel gratuito de SageMaker ofrece una prueba gratis de 2 meses que proporciona 250 horas al mes de uso de cuadernos t2.medium o t3.medium, 50 horas al mes de m4.xlarge o m5.xlarge para entrenamiento y 125 horas al mes de m4.xlarge o m5.xlarge para hosting. [30]

- ***Google Colab***

Colab es un servicio alojado de Jupyter Notebook que no requiere configuración y que ofrece acceso sin coste económico a recursos de computación, como GPUs y TPUs. Es una solución especialmente adecuada para el aprendizaje automático, la ciencia de datos y la educación. Esta plataforma da prioridad a los usuarios que estén programando activamente en un cuaderno. [31]

También restringe las acciones que afecten negativamente a otras personas o que estén orientadas a eludir nuestras políticas contra el abuso. Los recursos de Colab no están garantizados ni son ilimitados, y los límites de uso a veces varían. Estas restricciones son necesarias para que Colab pueda ofrecer recursos sin coste económico. [31]

- **Kaggle**

Es una plataforma de comunidad en línea para científicos de datos y aficionados al Machine Learning que permite a los usuarios colaborar con otros usuarios, encontrar y publicar conjuntos de datos, utilizar notebooks integrados GPU, y competir con otros científicos de datos para resolver los desafíos de la ciencia de los datos. El objetivo es ayudar a los profesionales y estudiantes a alcanzar sus objetivos en su viaje de la ciencia de los datos con las poderosas herramientas y recursos que ofrece.[32]

Además proporciona acceso a Notebooks que permite editar y ejecutar código para las tareas de ciencia de datos en el navegador, mediante fuertes recursos en la nube con un uso máximo de 30 horas de GPU y 20 horas de TPU por semana. Pudiendo cargar conjuntos de datos propios a Kaggle o también descargar los conjuntos de datos de otros usuarios también.[32] La Tabla 4 compara los tres entornos de desarrollo explicados.

Tabla 4. Comparativa de entornos Cloud para Machine Learning.

Parámetro	Google Colab	Kaggle	Amazon SageMaker
Tipo	Notebooks en la nube	Notebooks, Competencias, Datasets, modelos y foros	Plataforma completa de ML
Costo	Gratuito con límites	Gratuito	Pago según uso
Recursos de cómputo	GPU/TPU gratuitos (con limitaciones)	GPU gratuitos	GPU/CPU escalable según necesidad
Soporte para bibliotecas	Amplio (TensorFlow, PyTorch, etc.)	Amplio (compatible con la mayoría)	Amplio (incluye servicios de ML)
Colaboración	Muy buena (compartir fácilmente)	Muy buena (discusiones y compartir kernels)	Moderada (uso centrado en proyectos)
Almacenamiento	Integración con Google Drive	Almacenamiento limitado en kernels	Integración con S3 y almacenamiento escalable
Facilidad de uso	Intuitivo y accesible para principiantes	Intuitivo, orientado a competencias	Requiere conocimientos más avanzados
Despliegue de modelos	Limitado (requiere más trabajo)	Limitado (no está diseñado para despliegue)	Fácil (integrado con servicios de AWS)
Funciones adicionales	Acceso a herramientas de Google (Drive, Sheets, etc.)	Competencias y datasets accesibles directamente	Herramientas de monitoreo y ajuste de modelos
Usuarios objetivo	Estudiantes, investigadores, desarrolladores	Científicos de datos, competidores	Empresas, desarrolladores y científicos de datos

e. Metodologías para proyectos Machine Learning

Para desarrollar soluciones de Machine Learning existen varias metodologías que permiten llevar a cabo estas implementaciones desde la comprensión del entorno hasta el despliegue y evaluación del modelo. Entre las más populares tenemos CRISP-DM, ML-Canvas y KDD, a continuación se describe cada una de ellas.

- **Cross-Industry Standard Process for Data Mining (CRISP-DM)**

Bussiness Understanding. Esta etapa busca entender los objetivos del proyecto así como los requisitos desde un punto de vista de negocio y convertir esas necesidades en un plan de trabajo para Machine Learning.[34]

Data Understanding. En esta etapa se realiza una recolección inicial de datos con la intención de familiarizarse con ellos, detectar problemas de calidad del dato, obtener ya alguna conclusión preliminar y, eventualmente, descubrir subconjuntos de datos interesantes para el entrenamiento del modelo.[34]

Data Preparation. En esta etapa se construye el conjunto de datos entrenar y validar el modelo, a partir de los datos en bruto de la etapa anterior, incluye la selección de datos, su limpieza, transformación, aumento entre otros.[34]

Modeling. Esta es la etapa central de la metodología aunque no la más extensa, consiste en la selección y aplicación de diferentes modelos de machine learning / data mining. Con frecuencia, y durante esta fase, se plantea la necesidad de retomar la fase anterior para adaptar o refinar datos.[34]

Evaluation. Aquí se trata de valorar si los modelos construidos satisfacen las necesidades de negocio. Al final de esta fase se decide si los resultados obtenidos consiguen los objetivos perseguidos y si, por tanto, se puede pasar a desplegarlos.[34]

Deployment. Esta es la fase más técnica y de software que consiste en desplegar los modelos definidos para que puedan ser usados ya en los procesos de análisis y decisión de la compañía.[34]

- ***Machine Learning Canvas (ML-Canvas)***

Este marco de trabajo fue creado por Louis Dorard en el año 2015 y sirve para pensar y estructurar proyectos de machine learning. Este canvas tiene 10 recuadros para tener en cuenta. En líneas generales se puede ver como 4 grandes secciones, primero definir el objetivo del sistema, definir cómo va a aprender el modelo, cómo van a ser las predicciones y finalmente cómo se evaluará el modelo. [35] La Figura 17 esquematiza el flujo de trabajo de esta metodología.



Figura 17. Flujo de trabajo ML-Canvas [36]

Goal. Es la propuesta de valor, es decir responder a las preguntas. ¿Qué es lo que se busca hacer? ¿Cuál es su importancia? ¿Quién se beneficiará? [36]

Learn. Aquí se define la fuente y la recolección de datos, además se define la entrada del modelo en función de los datos en bruto, así también, en estas sección se define los tiempos de creación de modelos y actualización con nuevos datos de entrenamiento. [36]

Predict. En esta sección se defina la tarea de Machine Learning a utilizarse, también se establece la forma en que se procesa las predicciones y se evalúa el modelo con métricas de evaluación antes del despliegue. [36]

Evaluate. Esta etapa monitorea y evalúa el modelo en tiempo real.[36]

- **Sampling, Exploring, Modifying, Modeling, and Assessing (SEMMA)**

SEMMA fue creada por el SAS Institute en 2012, y es una metodología enfocada al Data Mining que establece es un proceso que incluye esas cinco fases. La Figura 18 muestra flujo de trabajo de esta metodología. [37]

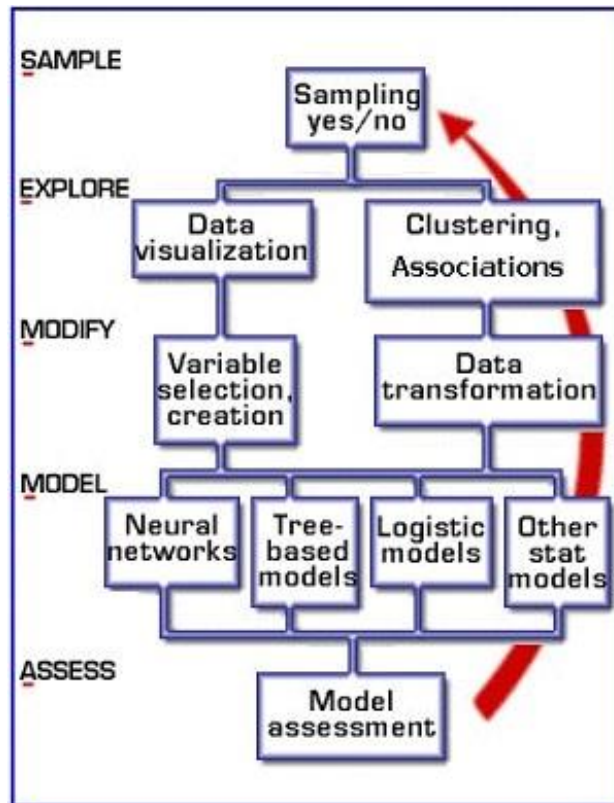


Figura 18. Flujo de trabajo SEMMA.[37]

Sample. Esta fase comprende una recolección de datos prestando atención al tamaño de las muestras de manera que el conjunto de datos sea suficientemente extenso para que sea representativo pero a la vez suficientemente pequeño como para permitir el procesamiento.[37]

Explore. En esta fase se pretende entender la naturaleza de los datos mediante visualización, agrupamiento, entre otros, de forma que se puedan observar relaciones, tendencias y cualquier otra información que brinde una idea general sobre los datos y la relación entre ellos.[37]

Modify. Esta fase se encarga de preprocesar los datos de forma que queden listos para el posterior modelado mediante transformaciones, seleccionando variables e incluso creándolas a partir de datos en bruto.[37]

Model. Esta etapa determina el modelo más adecuado para cumplir con los requerimientos del proyecto, seleccionando entre las familias disponibles y afinando el modelo en función a los datos y las salidas requeridas.[37]

Asses. Evaluar el funcionamiento del modelo implementado respecto a fiabilidad, utilidad, entre otros.[37]

La Tabla 5 compara estas metodologías con parámetros como fases, flexibilidad y aplicabilidad entre otros.

Tabla 5. Comparación de metodologías para proyectos de Machine Learning.

Parámetro	CRISP-DM	ML-Canvas	SEMMA
Tipo	Metodología estándar	Plantilla visual y iterativa	Metodología de análisis
Fases o etapas	Business understanding Data understanding Data preparation Modeling Evaluation Deployment	Goal Learn Predict Evaluate	Sample Explore Modify Model Assess
Enfoque	Enfoque secuencial y cíclico	Enfoque visual y colaborativo	Enfoque en la preparación de datos y la modelación
Flexibilidad	Moderada	Alta	Moderada
Uso de documentación	Sí, detallada y estructurada	Sí, pero más visual	Sí, más técnica
Adaptable a diferentes dominios	Sí	Sí	Limitado
Público objetivo	Data scientists, analistas	Equipos multidisciplinarios	Científicos de datos
Facilidad de uso	Buena, pero puede ser técnica	Muy alta	Moderada
Énfasis en la evaluación	Alto	Moderado	Alto
Iteración	Posible, pero menos explícita	Energiza la iteración	Focalizada en el ajuste

1.3.5 Deep Learning

El Deep Learning es un subcampo específico del Machine Learning, plantea una nueva visión del aprendizaje de representaciones a partir de datos que pone énfasis en el aprendizaje de capas sucesivas de representaciones cada vez más significativas. La cantidad de capas que contribuyen a un modelo de datos se denomina profundidad del modelo. A menudo implica decenas o incluso cientos de capas sucesivas de representación y todas se aprenden automáticamente a partir de la exposición a los datos de entrenamiento. [26]

Los modelos de Deep Learning casi siempre son redes neuronales profundas, estas transforman la información de entrada en representaciones que son cada vez más diferentes de la original y cada vez más informativas sobre el resultado final. Se puede pensar en una red profunda como una operación de destilación de información de varias etapas, donde la información pasa por filtros sucesivos y sale cada vez más purificada. En la Figura 19 se muestra este proceso para un sistema clasificador de Dígitos. [26]

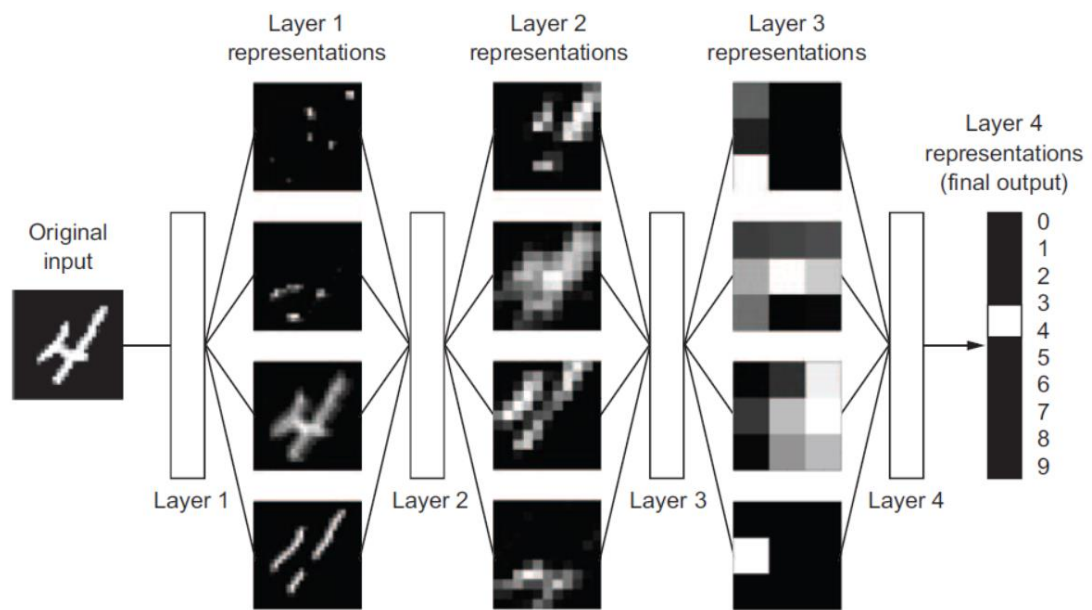


Figura 19. Transformación de información en una red neuronal profunda. [26]

El aprendizaje puede verse como encontrar un conjunto de valores para los pesos también llamados parámetros de todas las capas en una red, de modo que la red asigne correctamente las entradas de ejemplo a sus objetivos asociados. Para controlar esto se debe poder medir qué tan lejos está esta salida de lo esperado. Este es el trabajo de la función de pérdida “loss function”. [26]

La función de pérdida toma las predicciones de la red y el objetivo real y utilizar esta puntuación como señal de retroalimentación para ajustar un poco el valor de los pesos, en una dirección que reducirá la puntuación de pérdida para el ejemplo actual. Este ajuste es trabajo del optimizador, que implementa lo que se llama el algoritmo de retro propagación o “back propagation”. La Figura 20 muestra gráficamente este proceso. [26]

Los datos pueden ser representados por contenedores de datos, casi siempre datos numéricos llamados tensores, estos pueden ser arreglos multidimensionales como escalares, vectores y matrices. Esta última sirve mucho en arreglos o secuencias de datos en los que el tiempo u orden de ocurrencia importa, estos se los representa en tensores 3D en los que uno de sus ejes está reservado para el tiempo. [26]

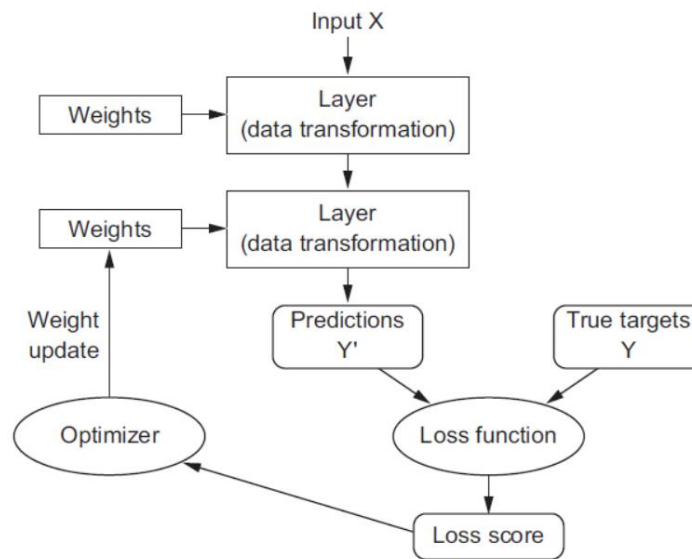


Figura 20. Aprendizaje en Deep Learning. [26]

a. Optimización basada en gradientes

El ajuste gradual, sucede dentro de lo que se llama un ciclo de entrenamiento, que funciona de la siguiente manera; Primero se dibuja un lote de muestras de entrenamiento x y sus etiquetas y . Después se procesa los datos con la red en x para obtener predicciones y' , con estos valores se calcula la pérdida de la red en el lote, una medida del desajuste entre y' predicha y la salida real. Por último se actualiza todos los pesos de la red de manera que reduzca ligeramente la pérdida en esta lote. [26]

Un gradiente es la derivada de una operación tensorial. El objetivo es encontrar su mínimo es decir un punto donde la derivada es 0, el optimizador encuentra todos los puntos donde la derivada llega a 0 y verifica cuáles de estos puntos la función tiene el valor más bajo. Uno de los más utilizados en el gradiente estocástico descendente

SGD. En la Figura 21 se muestra el proceso de búsqueda de un mínimo con SGD. El parámetro “learning rate”, puede ser definido en el ciclo de entrenamiento. [26]

En la práctica, la función de la red neuronal consta de muchas operaciones tensoriales encadenadas, cada una de las cuales tiene una derivada simple y conocida. Cada operación es modificada mediante un algoritmo llamado retropropagación. La cual comienza con el valor de pérdida final y avanza hacia atrás desde las capas superiores a las inferiores, aplicando la regla de la cadena para calcular la contribución que cada parámetro tuvo en el valor de pérdida. [26]

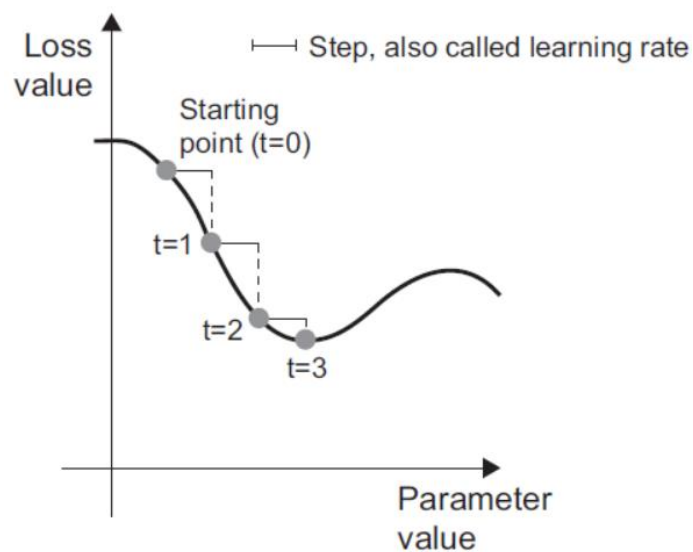


Figura 21. Funcionamiento SGD. [26]

b. Red neuronal convolucional

Las CNN por su siglas en inglés *Convolutional Neuronal Network*, son un tipo de modelo de Deep Learning que se utiliza casi universalmente en aplicaciones de visión artificial. Su principal aplicación tiene que ver con problemas de clasificación de imágenes y detección de objetos. La diferencia con las redes neuronales densamente conectadas es que utilizan capas convolucionales para la extracción de características. [26]

- ***Capas de convolución***

La diferencia fundamental entre una capa densamente conectada y una capa de convolución es principalmente la forma de aprender los patrones. Mientras que las

capas densas aprenden patrones globales en su espacio de características de entrada, las capas de convolución aprenden patrones locales, en el caso de las imágenes, patrones que se encuentran en pequeñas ventanas 2D de las entradas. [26]

Después de aprender un patrón una CNN puede reconocerlo en cualquier lugar de la imagen. Una red densamente conectada tendría que aprender el patrón de nuevo si apareciera en una nueva ubicación. Esto hace que los datos sean eficientes al procesar imágenes ya que necesitan menos muestras de entrenamiento para aprender representaciones que tengan poder de generalización. [26]

Otra ventaja es que una CNN puede aprender jerarquías espaciales de patrones. Mientras que la primera capa de convolución aprenderá pequeños patrones locales como bordes, una segunda capa de convolución aprenderá patrones más grandes hechos de las características de las primeras capas, y así sucesivamente. Esto permite que aprendan de manera eficiente conceptos visuales cada vez más complejos y abstractos. La Figura 22 esquematiza este proceso. [26]

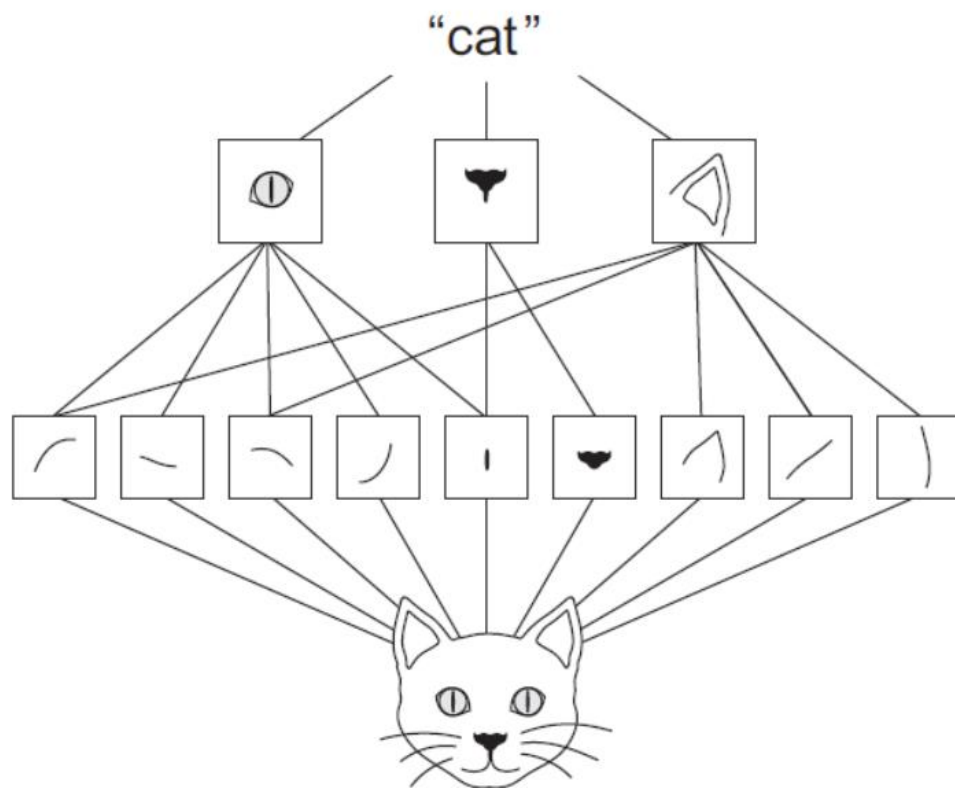


Figura 22. Proceso extracción de características en CNN. [26]

Las convoluciones operan sobre tensores 3D, llamados mapas de características, con dos ejes espaciales y un eje de profundidad. La operación de convolución extrae parches de su mapa de características de entrada y aplica la misma transformación a todos estos parches, produciendo un mapa de características de salida. Este mapa de características de salida sigue siendo un tensor 3D: tiene un ancho, una altura y profundidad arbitraria. [26]

La profundidad de salida es un parámetro de la capa y los diferentes canales en ese eje representan filtros. Las convoluciones están definidas por dos parámetros muy importantes; El tamaño de los parches generalmente con valores de 3×3 o 5×5 , y la profundidad del mapa de características de salida que representa la cantidad de filtros calculados por la convolución. La Figura 23 muestra el proceso de convolución. [26]

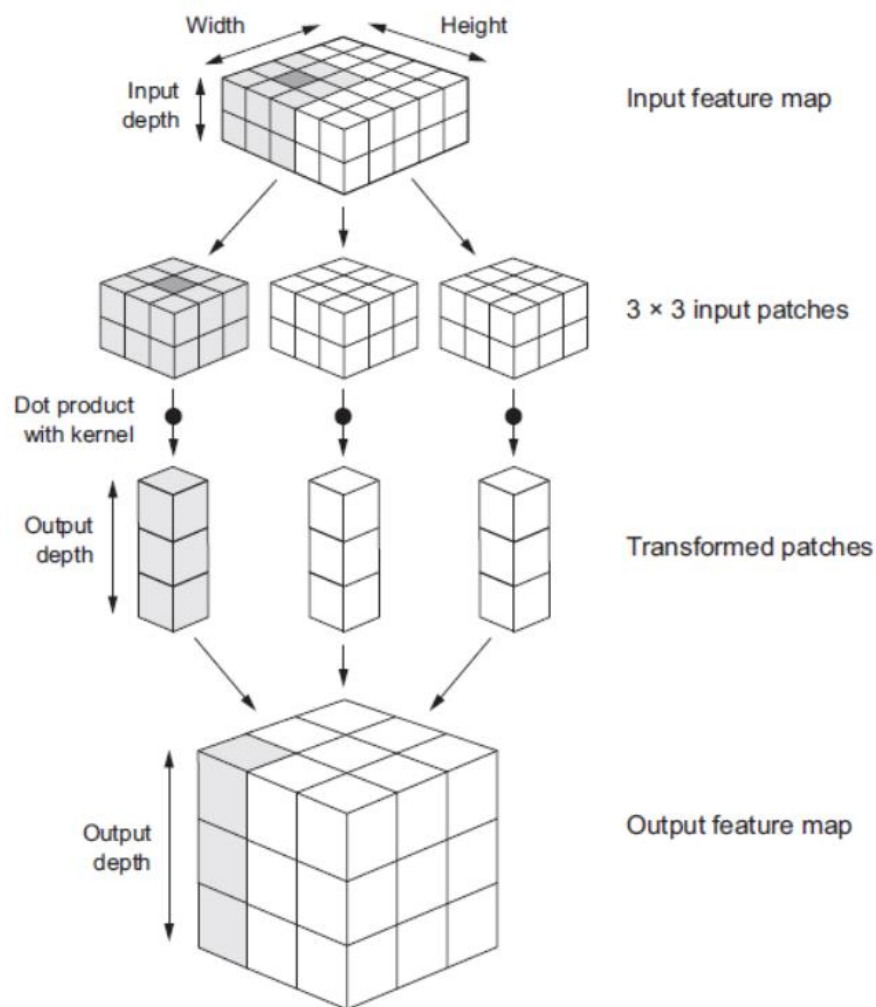


Figura 23. Proceso de convolución en CNN. [26]

- **Padding**

El padding consiste en agregar una cantidad adecuada de filas y columnas a cada lado del mapa de características de entrada para que sea posible colocar ventanas de convolución centrales alrededor de cada mosaico de entrada. Puede tomar dos valores: "valid", que significa que no hay relleno y "same", que rellena de tal manera que tenga una salida con el mismo ancho y alto que la entrada". La Figura 24 muestra el efecto del relleno en capas de convolución. [26]

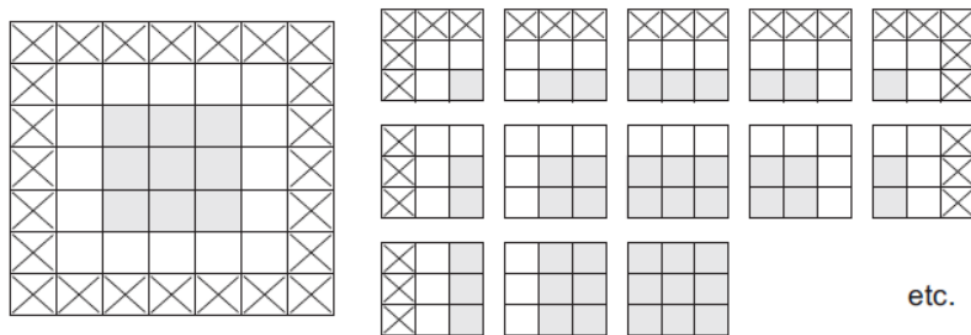


Figura 24. Relleno o “Padding” en convolución de CNN. [26]

- **Strides**

El otro factor que puede influir en el tamaño de salida es la noción de pasos, que comprende la distancia entre dos ventanas sucesivas, llamado “stride”, que por defecto es 1. Para convoluciones con un paso mayor que 1 el ancho y la altura del mapa de características se reducen en dicho valor de paso. Este tipo de convoluciones rara vez se utilizan. La Figura 25 muestra el efecto para un valor de paso igual a 2. [26]

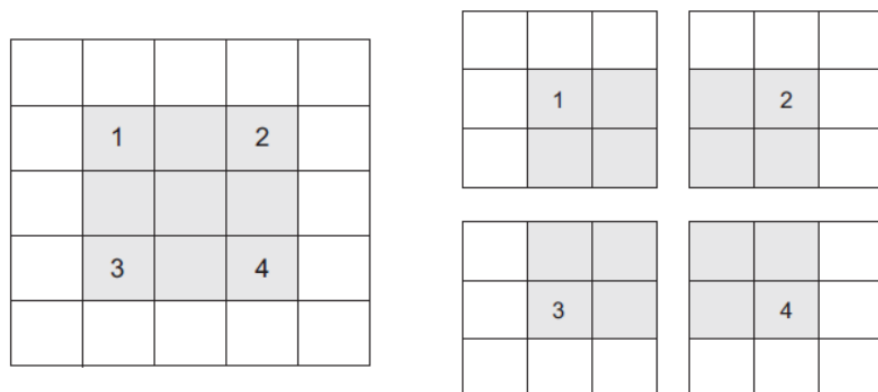


Figura 25. Convolución con “stride=2” y “path=3”. [26]

- ***Max Pooling***

La agrupación máxima también llamada “Max pooling”, consiste en extraer ventanas de los mapas de características de entrada y generar el valor máximo de cada canal. Es conceptualmente similar a la convolución, excepto que en lugar de transformar parches locales a través de una transformación lineal aprendida, se transforman a través de una operación de tensor máximo codificada. [26]

Generalmente se realiza con ventanas de 2×2 y paso 2, para reducir la resolución de los mapas de características en un factor de 2. Esto con el fin de reducir la cantidad de coeficientes del mapa de características que se deben procesar, así como inducir jerarquías de filtros espaciales al hacer que las capas de convolución sucesivas observen ventanas cada vez más grandes. Esta capa al igual que la de convolución también se puede utilizar pasos. [26]

También se puede utilizar el agrupamiento promedio, donde cada parche de entrada local se transforma tomando el valor promedio de cada canal sobre el parche, en lugar del máximo. Pero el agrupamiento máximo tiende a funcionar mejor que estas soluciones alternativas, ya que es más informativo observar la presencia máxima de diferentes características que su presencia promedio. [26]

c. Transfer Learning

Consiste en usar una red pre entrenada en un conjunto de datos grande. La jerarquía espacial de características aprendidas por esta red puede actuar efectivamente como un modelo genérico del mundo visual resultando útiles sus características para nuevos problemas que involucren clases completamente diferentes a las de la tarea original. Hay dos formas: extracción de características y ajuste fino. [26]

- ***Extracción de características***

Las CNN generalmente tienen dos partes; Capas de agrupación y convolución llamado también base convolucional para la extracción de características y un clasificador densamente conectado. La extracción de características consiste en tomar la base convolucional de una red previamente entrenada, pasar los nuevos datos a través de

ella y entrenar un nuevo clasificador sobre la base de los resultados. La Figura 26 muestra este proceso. [26]

- ***Fine tuning***

El ajuste fino consiste en descongelar algunas de las capas superiores de una base de modelo congelada utilizada para la extracción de características y entrenar conjuntamente tanto la parte recién agregada del modelo como estas capas superiores. Esto ajusta ligeramente las representaciones más abstractas del modelo que se está reutilizando, con el fin de hacerlas más relevantes para el problema en cuestión. [26]

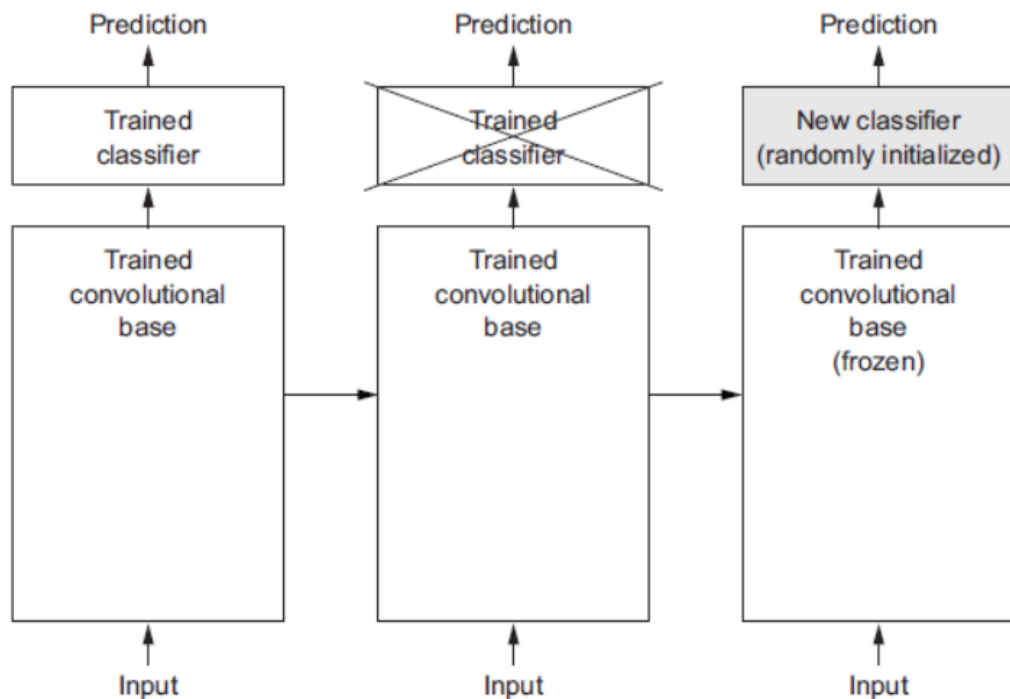


Figura 26. Extracción de características con CNN pre entrenada. [26]

d. Redes neuronales residuales

Las redes neuronales más profundas son más difíciles de entrenar. ResNet es un marco de aprendizaje residual para facilitar el entrenamiento de redes que son sustancialmente más profundas. La profundidad de la red convolucional es de importancia crucial, pero para que la red aprenda mejor no solo debe apilar más capas, al contrario esto trae consigo problemas como el sobre ajuste o desvanecimiento del gradiente. [38]

Para abordar este problema, se realiza un cambio en la construcción de modelo más profundo, donde las capas agregadas son mapeo de identidad y las otras capas se copian del modelo más superficial aprendido. La existencia de esta solución construida indica que un modelo más profundo no debería producir un error de entrenamiento mayor que su contraparte más superficial. La Figura 27 muestra como un modelo más profundo en CNN planas trae consigo un error más alto. [38]

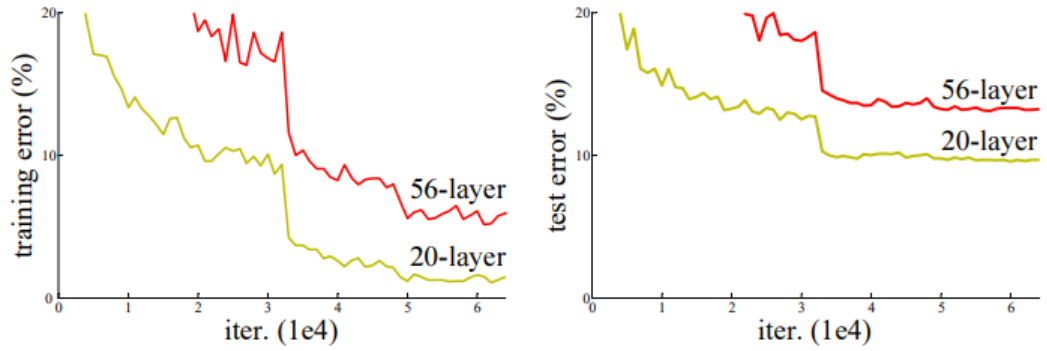


Figura 27. Error de entrenamiento y validación de CNN planas. [38]

La solución propuesta de aprendizaje residual profundo en lugar de esperar que cada una de las pocas capas apiladas se ajuste directamente a una asignación subyacente deseada, deja explícitamente que estas capas se ajusten a una asignación residual. La Figura 28 muestra la arquitectura de un bloque de aprendizaje residual, donde x es la entrada, $F(x)$ como la función de activación y un flujo residual *identity*. [38]

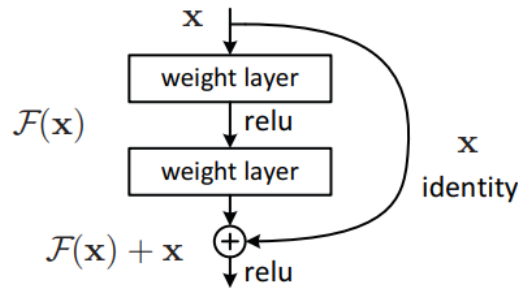


Figura 28. Diagrama de bloque de aprendizaje residual. [38]

El bloque de aprendizaje residual se define así según [38].

$$y = F(x, [W_i]) + x \quad (4)$$

Donde x representa la entrada e y la salida del bloque de aprendizaje residual, la función $F(x, [W_i])$ representa el mapeo residual que para el caso del diagrama de la Figura 28 es el resultado de las operaciones lineales de los pesos y los sesgos de los dos capas del bloque. Esta función es aplicable a múltiples capas de convolucionales.[38]

Los resultados experimentales mostrados en [38], reflejan que el uso de bloques de aprendizaje residual profundo mejorar el comportamiento de la red cuando se aumenta su profundidad. La Figura 29 muestra los resultados experimentales en arquitecturas de 18 y 34 capas, a la izquierda CNN planas, a la derecha utilizando aprendizaje residual evidenciado el mejor desempeño de las CNN con 38 capas de profundidad.

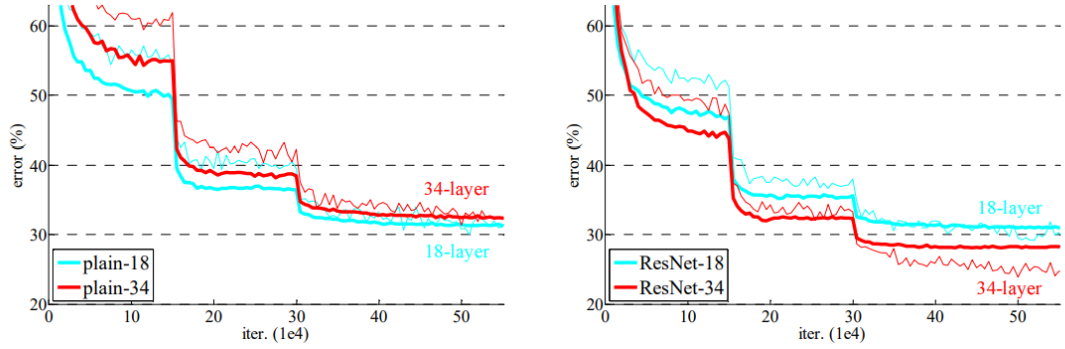


Figura 29. Resultados de aplicación de bloques residuales en CNNs. [38]

La Figura 30 muestra las diferentes arquitecturas para el conjunto de datos ImageNet, los bloques de construcción se muestran entre paréntesis, con la cantidad de bloques apilados. El submuestreo se realiza mediante conv3_1, conv4_1 y conv5_1 con un paso de 2. [38]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 30. Arquitecturas de ResNet. [38]

e. Librerías

- ***TensorFlow***

TensorFlow es una plataforma de código abierto para el desarrollo de proyectos de Machine Learning, cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos comunitarios que permite a los investigadores impulsar el aprendizaje automático de última generación y a los desarrolladores crear e implementar fácilmente aplicaciones basadas en aprendizaje automático. [39]

Fue desarrollado originalmente por investigadores e ingenieros que trabajaban en el equipo de Inteligencia Artificial de Google Brain para realizar investigaciones en Machine Learning y redes neuronales. Sin embargo, el marco es lo suficientemente versátil como para usarse también en otras áreas. Además proporciona API estables para Python y C++ , así como una API compatible con versiones anteriores no garantizada para otros lenguajes. [39]

- ***PyTorch***

PyTorch es un paquete de Python que proporciona dos características de alto nivel para cálculos de tensores con fuerte aceleración de GPU. Además de Redes neuronales profundas construidas sobre un sistema de auto graduación basado en cinta. Puede reutilizarse junto a los paquetes de Python como NumPy, SciPy y Cython, entre otros. La siguiente Tabla muestra los componente de PyTorch. [40]

Tabla 6. Componentes de PyTorch

Componente	Descripción
torch	Una biblioteca Tensor como NumPy, con fuerte soporte para GPU
torch.autograd	Una biblioteca de diferenciación automática basada en cinta que admite todas las operaciones de tensor diferenciables en la antorcha
torch.jit	Una pila de compilación (TorchScript) para crear modelos serializables y optimizables a partir del código de PyTorch
torch.nn	Una biblioteca de redes neuronales profundamente integrada con Autograd diseñada para una máxima flexibilidad
torch multiprocessing	Multiprocesamiento de Python, pero con intercambio mágico de memoria de tensores de antorcha entre procesos. Útil para la carga de datos y el entrenamiento de Hogwild
torch.utils	DataLoader y otras funciones de utilidad para mayor comodidad

- **Keras**

Keras 3 es un marco de aprendizaje profundo multi-backend, compatible con JAX, TensorFlow, PyTorch y OpenVINO. Permite crear y entrenar modelos sin esfuerzo para visión artificial, procesamiento de lenguaje natural, procesamiento de audio, pronóstico de series temporales, sistemas de recomendación, etc. Desarrollo acelerado de modelos mediante soluciones de aprendizaje profundo más rápido gracias a la experiencia de usuario (UX) de alto nivel de Keras y la disponibilidad de tiempos de ejecución fáciles de depurar como PyTorch o la ejecución ansiosa de JAX. [41]

Presenta un rendimiento de vanguardia ya que permite elegir el backend más rápido para la arquitectura de su modelo. Permite la capacitación a escala de centro de datos escalando con confianza desde su computadora portátil a grandes clústeres de GPU o TPU. Cuenta con casi tres millones de desarrolladores, desde nuevas empresas emergentes hasta empresas globales. [41] La Tabla 7 compara las librerías de desarrollo explicadas.

Tabla 7. Librerías de desarrollo en Deep Learning.

Parámetro	TensorFlow	PyTorch	Keras
Flexibilidad	Alta, permite programación imperativa y funcional	Muy alta, permite definición dinámica de gráficos	Moderada, más estructurada pero flexible en capas
Facilidad de uso	Complejo para principiantes	Intuitivo y amigable	Muy fácil, diseño simple y directo
Ecosistema	Amplio, incluye TensorFlow Extended (TFX), TensorBoard	Menor, pero en crecimiento, con buena integración a NumPy	Depende de TensorFlow o Theano como backend
Soporte para GPU	Excelente, completamente optimizado	Excelente, soporte nativo con CUDA	Depende del backend, generalmente bueno
Modelos Preentrenados	TensorFlow Hub	Torchvision y otros	Keras Applications
API de bajo nivel	Sí, con TensorFlow Core	Sí, altamente accesible	Limitado, más abstracto en su enfoque
Documentación	Completa pero a veces confusa	Clara y en crecimiento	Muy clara, especialmente para principiantes
Comunidad y soporte	Gran comunidad y soporte robusto	Comunidad creciente y activa	Comunidad activa, más centrada en principiantes
Exportación a producción	Alta, con TensorFlow Serving y TFX	Puede requerir más trabajo	Buen soporte mediante TensorFlow Serving

1.3.6 Visión Artificial

La visión artificial es una disciplina que engloba todos los procesos y elementos que proporcionan ojos a una máquina, también se conoce como visión computacional y describe la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales de ese mundo. Para cumplir con este objetivo se tienen varias etapas fundamentales relacionadas con hardware y software, además de conceptos teóricos de Procesamiento digital de Imágenes. La Figura 31 muestra estructura básica de un sistema de visión artificial. [42]

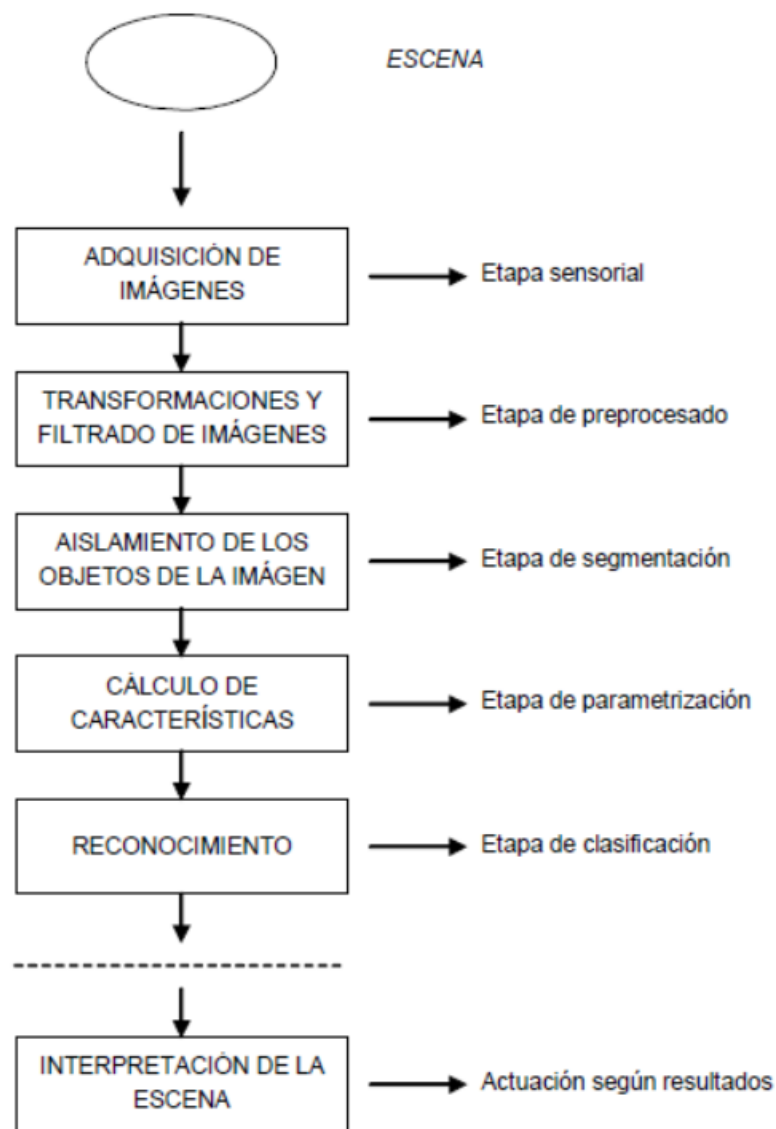


Figura 31. Esquema de etapas de Visión Artificial [42]

La primera etapa es la adquisición de la imagen digitalizada, para esto se precisa de un sensor o cámara, después es necesario el pre procesamiento de esta imagen para mejorar la calidad y eliminar el ruido, esto se logra mediante transformación y filtrado de imágenes, después de este proceso está la etapa de segmentación, una de las más importantes dentro del proceso de reconocimiento de una escena con visión computacional. [42]

La segmentación tiene por objetivo dividir la imagen en las partes u objetos que la forman, siendo la segmentación autónoma uno de los más difíciles, ya que es determinante el éxito o fracaso del reconocimiento de la imagen. Existen dos tipos de segmentación; Representación por frontera apropiada cuando el objetivo es la centra en el reconocimiento de formas, y Representación por región eficiente en representaciones internas de textura o representaciones esqueléticas. [42]

La etapa de parametrización toma los datos segmentados y extrae rasgos que producen información cuantitativa o rasgos diferenciadores de objetos. Por último, la etapa de clasificación, la cual es la encargada de interpretar los datos parametrizados, clasificando el objeto o escena, en esta etapa puede estar presente un modelo de Machine Learning entrenado en el reconocimiento de imágenes o secuencias de ellas, ayudando a la clasificación. [42]

a. Estimación de postura

La estimación de postura humana es un enfoque del muy relevante en el campo de la visión artificial, cuyo objetivo es identificar la postura de una figura humana a partir de una imagen. Esta pose se define a partir de puntos clave, generalmente articulaciones, mediante la posición (x, y) de cada punto en detección 2D, o (x, y, z) si se trata de detección en 3D. [43]

Se clasifica en dos grandes categorías; El enfoque “bottom-up” primero localiza todos los puntos clave de la imagen y luego forma la representación gráfica de la persona, por otro lado, el enfoque “top-down” detecta primero a las personas que aparecen en la imagen, obteniendo la caja delimitadora de cada una, y después aplicamos alguna técnica para obtener los puntos clave para cada persona identificada. [43]

En la obtención de puntos clave, existen dos enfoques principales: los mapas de calor y la regresión directa. Los mapas de calor se generan con una distribución gaussiana bidimensional centrada en el punto clave, mientras que en la regresión directa se busca obtener directamente la posición del punto en cada dimensión. [43]

- **MediaPipe Pose Landmarker**

Esta solución permite detectar puntos de referencia de cuerpos humanos en una imagen o video, la cual se puede utilizar para identificar las ubicaciones clave del cuerpo, analizar la postura, y categorizar los movimientos. Usa modelos de aprendizaje automático que trabajar con imágenes o videos individuales. La tarea muestra los puntos de referencia de la postura corporal en una imagen y en coordenadas mundiales tridimensionales. La Tabla 8 muestra las opciones de configuración de la Tarea [44]

Tabla 8. Parámetros de configuración de MediaPipe Pose Landmark. [44]

Parámetro	Descripción	Rango de Valores	Valor predeterminado
Running_mode	Establece el modo de ejecución de la tarea.	{IMAGE, VIDEO, LIVE_STREAM}	IMAGE
num_poses	La cantidad máxima de poses que puede detectar el Pose aterrizaje.	Integer > 0	1
min_pose_detection_confidence	La puntuación de confianza mínima para la detección de poses correctamente.	Float [0.0,1.0]	0.5
min_pose_presence_confidence	La puntuación de confianza mínima de la presencia en poses en la detección de puntos de referencia de poses.	Float [0.0,1.0]	0.5
min_tracking_confidence	La puntuación de confianza mínima para el seguimiento de poses para ser considerada exitosa.	Float [0.0,1.0]	0.5
output_segmentation_masks	Establece si el marcador de posición de la postura genera una máscara de segmentación para el elemento detectado la pose.	Boolean	False

El marcador de posición de poses usa una serie de modelos para predecir puntos de referencia de poses. La primera detecta la presencia de cuerpos humanos en el marco de una imagen, y la segunda localiza puntos de referencia en los cuerpos, esto mediante una red neuronal convolucional similar a MobileNetV2 que está optimizada para aplicaciones de entrenamiento en tiempo real y en el dispositivo. El modelo registra 33 ubicaciones de puntos de referencia corporales, la Figura 32 representa la ubicación aproximada de estos puntos. [44]

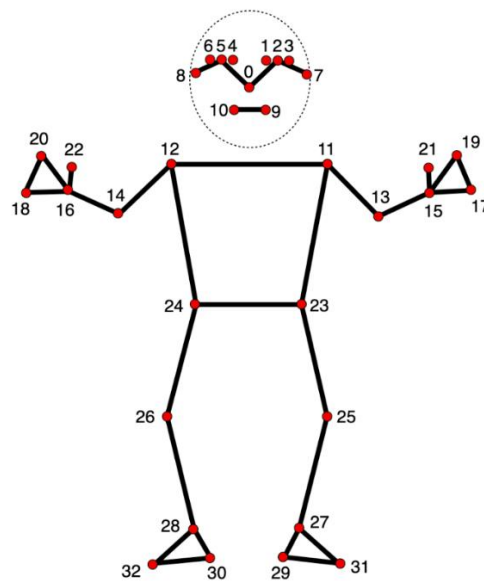


Figura 32. Puntos de referencia de MediaPipe Pose Landmark. [44]

- ***OpenPose***

OpenPose es un enfoque en tiempo real para detectar la pose 2D de varias personas en una imagen. Utiliza una representación no paramétrica, conocida como Part Affinity Fields (PAF), para aprender a asociar partes del cuerpo con individuos en la imagen. Este sistema tiene un enfoque bottom-up y logra una alta precisión y rendimiento en tiempo real, independientemente del número de personas en la imagen. [45]

El proceso general de OpenPose toma como entrada, una imagen en color de tamaño $w \times h$ (a) y produce las ubicaciones 2D de los puntos clave anatómicos para cada persona en la imagen, (e) . Primero, una red de avance predice un conjunto de mapas de confianza 2D S de ubicaciones de partes del cuerpo (b) y un conjunto de campos vectoriales 2D L de campos de afinidad de partes (PAF), que codifican el grado de asociación entre partes (c). En la Figura 33 se muestra este proceso. [45]

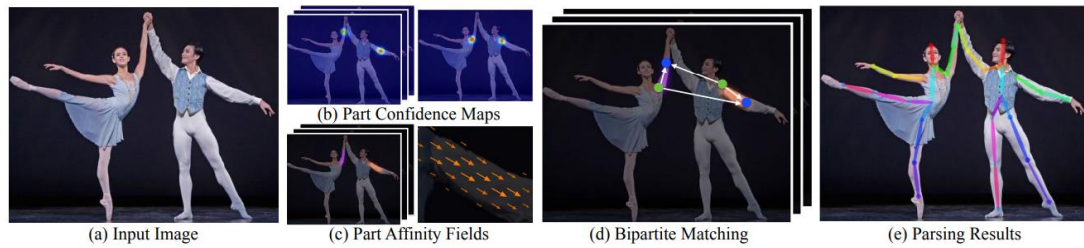


Figura 33. Proceso de estimación de postura de OpenPose. [45]

OpenPose consta de tres bloques diferentes: detección de cuerpo + pie, detección de manos y detección de rostro. Alternativamente puede utilizar el modelos originales solo de cuerpo entrenados con los conjuntos de datos COCO y MPII. Según la salida del detector corporal o facial, las propuestas del cuadro delimitador se pueden estimar aproximadamente a partir de algunas partes del cuerpo, en particular orejas, ojos, nariz, y cuello.

De manera análoga, las propuestas del cuadro delimitador manual se generan con los puntos clave del brazo. Esta metodología hereda los problemas de los enfoques top-down. La biblioteca también incluye detección de postura en 3D, mediante la realización de triangulación 3D con refinamiento no lineal, el tiempo de inferencia supera a todos los métodos de última generación, al tiempo que conserva resultados de alta calidad. La Figura 34 muestra la posiciones aproximadas de los puntos clave que utiliza OpenPose. [45]

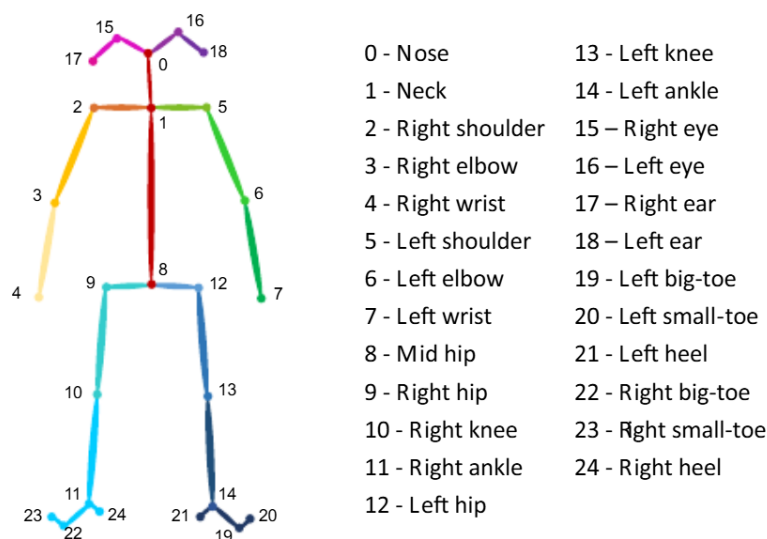


Figura 34. Puntos de clave de OpenPose. [46]

- **AlphaPose**

AlphaPose es un sistema que puede realizar mediciones precisas de estimación y seguimiento de posturas de todo el cuerpo de forma conjunta mientras se ejecuta en tiempo real. Utiliza varias técnicas como Symmetric Integral Keypoint Regression (SIKR) para una localización rápida y precisa, Parametric Pose Non-Maximum-Suppression (P-NMS) para eliminar detecciones humanas redundantes y Pose Aware Identity Embedding para estimación y seguimiento de poses conjuntas. [47]

Su enfoque es top-down, primero (i) dada una imagen de entrada se realiza detecciones humanas utilizando detectores de objetos. Cada humano detectado es recortado, redimensionado y reenviado a través de redes de seguimiento y estimación de pose para obtener la postura humana de cuerpo completo y las características de Re-ID. [47]

La regresión integral simétrica (a) se adopta para la localización de puntos clave de nivel fino. Utiliza Pose NMS (b) para eliminar poses redundantes. El módulo (c) de alineación guiada por postura (PGA) se aplica a la característica de Re-ID predicha para obtener características de Re-ID humana alineadas con la postura. La multi-stage identity matching (MSIM) (d) utiliza las posturas humanas, características de Re-ID y cuadros detectados para producir la identidad de seguimiento final. [47]

Durante el entrenamiento, se adoptaron la generación de propuestas y la destilación de conocimientos (e) para mejorar la capacidad de generalización de las redes. En la Figura 35 se muestra este funcionamiento para la estimación de postura de una o varias personas a la vez. [47]

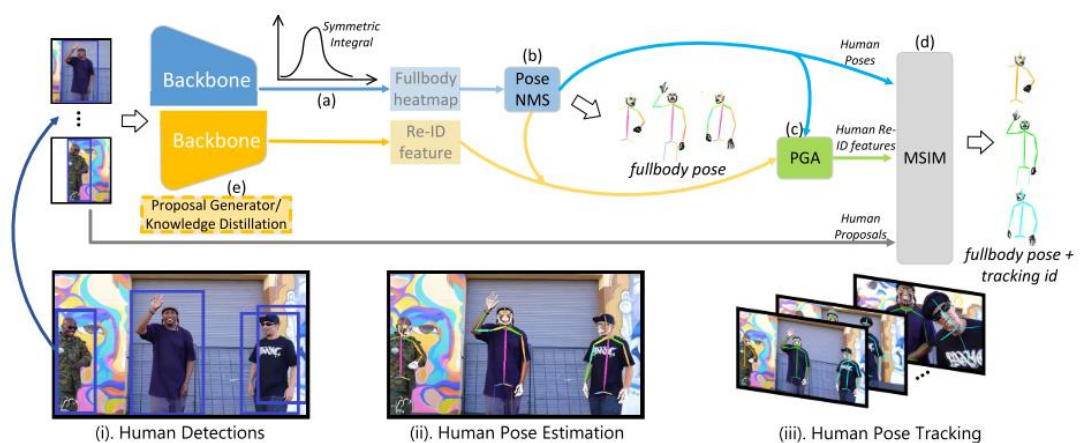


Figura 35. Pipeline de AlphaPose para estimación de postura. [47]

AlphaPose está dividido en cinco módulos, primero (a) módulo de carga de datos, (b) módulo de detección que proporciona detecciones humanas, (c) módulo de transformación de datos para procesar los resultados de la detección y recortar cada persona, (d) módulo de estimación de pose que genera puntos clave y/o identidad humana para cada persona, (e) módulo de post-procesamiento que procesa y guarda los resultados de la pose. El cuadro discontinuo indica los componentes opcionales de cada módulo, la Figura 36 esquematiza esta arquitectura. [47]

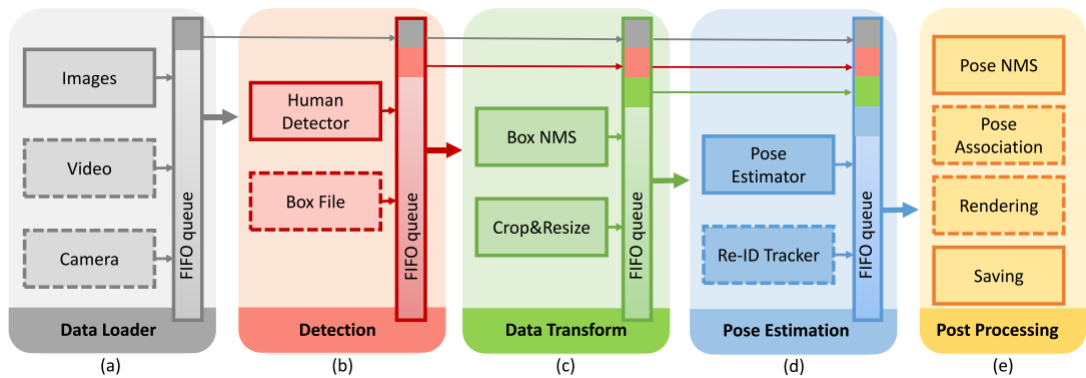


Figura 36. Arquitectura AlphaPose. [47]

Formato de punto clave anotado en el conjunto de datos Halpe-Full Body para (a) cuerpo y pie, (b) cara, (c) mano respectivamente. La Figura 37 muestra las posiciones aproximadas de los puntos clave descritos anteriormente. [47]

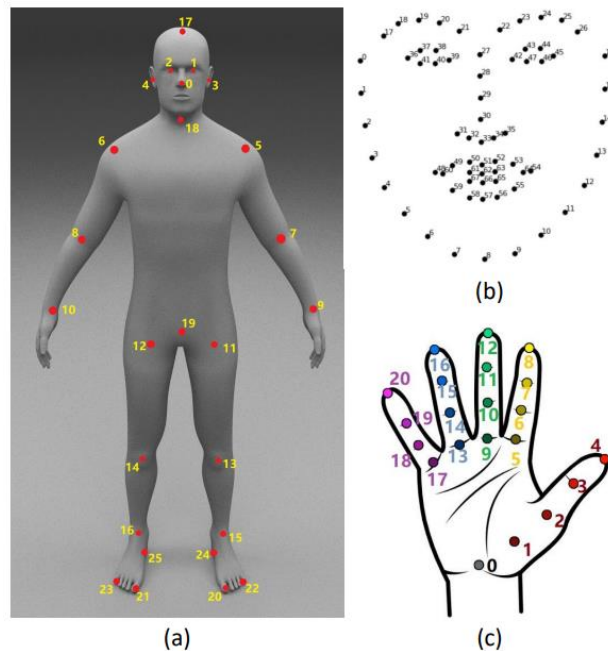


Figura 37. Posiciones aproximadas de puntos clave AlphaPose. [47]

La Tabla 9 compara los modelos de estimación de postura explicados anteriormente, mediante parámetros como puntos clave, requisitos de hardware y software entre otros.

Tabla 9. Comparación de modelos para estimación de postura.

Parámetro	MediaPipe Pose	OpenPose	AlphaPose
Desarrollador	Google	Carnegie Mellon University	AlphaPose Team
Año de lanzamiento	2019	2016	2019
Enfoque	Top-down	Top-down	Bottom-up
Números de puntos clave	33	25	18
Velocidad	Muy rápida y en tiempo real	Rápida y con suficiente optimización	Rápida y puede manejar múltiples personas
Precisión	Alta precisión en entornos controlados	Alta precisión, especialmente en configuraciones complejas	Alta precisión, optimizado para múltiples personas
Requerimientos de hardware	Bajo a moderado además que funciona en dispositivos móviles	Moderado a alto, ya requiere más recursos	Moderado, pero optimizado para GPU
Plataformas soportadas	Web, móvil, escritorio para Windows, Linux, macOS	Escritorio para Windows, Linux, macOS	Escritorio para Windows, Linux, macOS
Uso de GPU	Opcional	Indispensable	Opcional
Documentación	Muy completa y accesible	Completa, pero puede ser técnica	Buena, con ejemplos y tutoriales
Facilidad de uso	Facil y sencillo	Requiere configuración y comprensión	Facilidad de uso con su API
Desventajas	Limitaciones en condiciones de iluminación	Puede ser lenta sin optimización	Más difícil de configurar para principiantes

b. Detección de objetos

• YOLO

YOLO plantea la detección de objetos como un problema de regresión a cuadros delimitadores separados espacialmente y probabilidades de clase asociadas. Una única red neuronal predice cuadros delimitadores y probabilidades de clase directamente desde imágenes completas en una sola evaluación. El modelo YOLO procesa imágenes en tiempo real a 45 fotogramas por segundo. Por otro lado es capaz de generalizar desde imágenes naturales a otros dominios como el arte. [48]

El modelo procesa los componentes separados de la detección de objetos en una única red neuronal la cual utiliza características de toda la imagen para predecir cada cuadro delimitador, además predice todos los cuadros delimitadores de todas las clases de una imagen simultáneamente. Esto significa que la red razona globalmente sobre la imagen completa y todos los objetos de la imagen. El diseño de YOLO permite un entrenamiento de extremo a extremo y velocidades en tiempo real manteniendo al mismo tiempo una alta precisión promedio. La Figura 38 esquematiza el funcionamiento del modelo YOLO. [48]

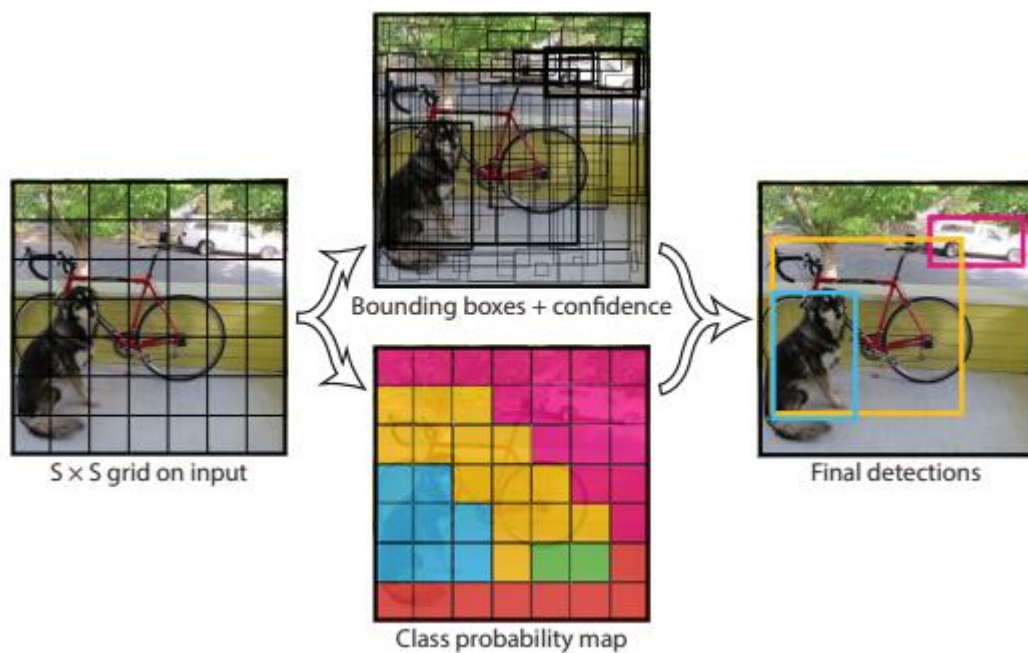


Figura 38. Funcionamiento de YOLO para detección de objetos. [48]

El sistema divide la imagen de entrada en una cuadrícula $S \times S$. Si el centro de un objeto cae dentro de una celda de la cuadrícula, esa celda de la cuadrícula es responsable de detectar ese objeto. Cada celda predice cuadros delimitadores y puntuaciones de confianza para esos cuadros. Si no existe ningún objeto en esa celda, las puntuaciones de confianza deben ser cero. De lo contrario, se desea que la puntuación de confianza sea igual a la intersección sobre la unión (IOU) entre el cuadro predicho y la verdad fundamental. [48]

Cada cuadro delimitador consta de 5 predicciones: x , y , w , h y confianza. Las coordenadas (x, y) representan el centro del cuadro en relación con los límites de la celda de la cuadrícula. El ancho y el alto se predicen en relación con toda la imagen.

Finalmente, la predicción de confianza representa la relación entre el cuadro predicho y cualquier cuadro de verdad fundamental. Cada celda de la cuadrícula también predice probabilidades de clase condicional, las cuales están condicionadas a la celda de la cuadrícula que contiene un objeto. [48]

Solo se predice un conjunto de probabilidades de clase por celda de la cuadrícula, independientemente del número de casillas. En el momento de la prueba se multiplica las probabilidades de clase condicional y las predicciones de confianza de las cajas individuales, lo que da puntuaciones de confianza específicas de clase para cada casilla. [48]

La arquitectura original de YOLO consta de 24 capas convolucionales seguidas de 2 capas completamente conectadas. La alternancia de capas convolucionales 1×1 reduce el espacio de características de las capas anteriores. Las capas convolucionales están pre entrenadas en la tarea de clasificación de ImageNet a la mitad de la resolución es decir la imagen de entrada es de 224×224 , y luego se duplica la resolución para la detección. La Figura 39 muestra esta arquitectura. [48]

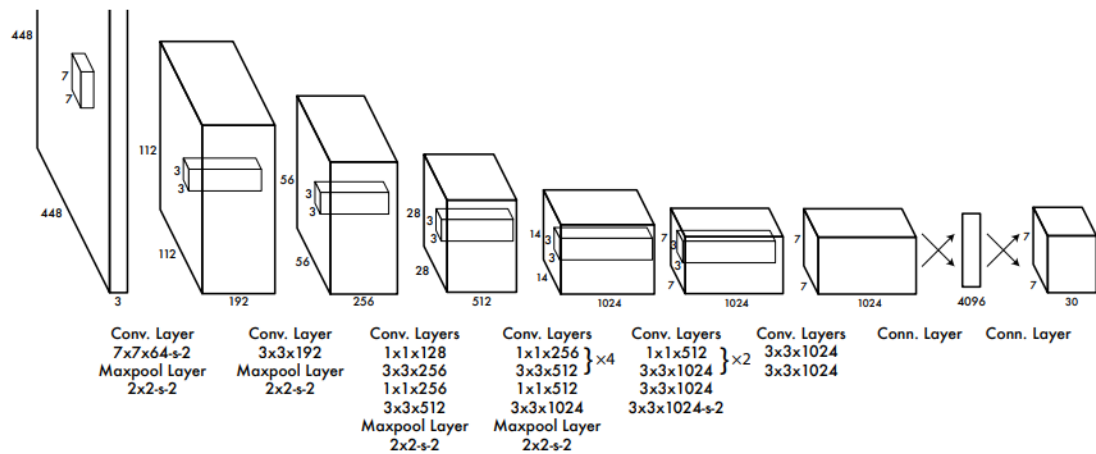


Figura 39. Arquitectura original YOLO. [48]

La documentación de Ultralytics ofrece soporte para una amplia gama de modelos YOLO, cada uno adaptado a tareas específicas como la detección de objetos, la segmentación de instancias, la clasificación de imágenes, la estimación de poses y el seguimiento multi objeto. La Tabla 10 muestra un resumen de las versiones más destacadas de YOLO y sus principales características. [49]

Tabla 10. Descripción de versiones de YOLO más destacadas. [49]

Modelo	Descripción
YOLOv3	La tercera iteración de la familia de modelos YOLO publicado por Joseph Redmon, conocida por su eficaz capacidad de detección de objetos en tiempo real.
YOLOv4	Una actualización nativa de la red oscura de YOLOv3, publicada por Alexey Bochkovskiy en 2020.
YOLOv5	Una versión mejorada que ofrece mejores prestaciones y compensaciones de velocidad en comparación con las versiones anteriores.
YOLOv6	Lanzado por Meituan en 2022 y utilizado en muchos de los robots de reparto autónomos de la empresa.
YOLOv7	Modelos actualizados de YOLO publicados en 2022 por los autores de YOLOv4.
YOLOv8	La última versión de la familia YOLO, con funciones mejoradas como la segmentación de instancias, la estimación de poses/puntos clave y la clasificación.
YOLOv9	Un modelo experimental entrenado en el Ultralytics YOLOv5 código base que implementa la Información de Gradiente Programable (PGI).
YOLOv10	Por la Universidad de Tsinghua, con formación sin NMS y una arquitectura orientada a la eficiencia y la precisión, que ofrece un rendimiento y una latencia de vanguardia.
YOLO11	Ultimos modelos que ofrecen un rendimiento de vanguardia (SOTA) a través de múltiples tareas.
SAM	Modelo Segment Anything original de Meta (SAM).
SAM2	La nueva generación del Segment Anything Model de Meta (SAM) para vídeos e imágenes.
MobileSAM	MobileSAM para aplicaciones móviles, por la Universidad Kyung Hee.
FastSAM	FastSAM por el Grupo de Análisis de Imagen y Vídeo del Instituto de Automatización de la Academia China de Ciencias.
YOLO-NAS	YOLO Modelos de búsqueda de arquitectura neuronal (NAS).
RT-DETR	Modelos de transformadores de detección en tiempo real de Baidu PaddlePaddle (RT-DETR).
YOLO-Mundo	Modelos de detección de objetos de vocabulario abierto en tiempo real de Tencent AI Lab.

YOLOv8 es la iteración no experimental más reciente de la serie YOLO de detectores de objetos en tiempo real, que ofrece un rendimiento puntero en términos de precisión y velocidad. Basándose en los avances de las versiones anteriores de YOLO , YOLOv8 introduce nuevas funciones y optimizaciones que lo convierten en la opción ideal para diversas tareas de detección de objetos en una amplia gama de aplicaciones. [50]

Emplea arquitecturas backbone y de cuello de última generación, lo que se traduce en un rendimiento mejorado de la extracción de características y la detección de objetos, además adopta un cabezal dividido sin anclajes Ultralytics, que contribuye a una mayor precisión y a un proceso de detección más eficaz en comparación con los enfoques basados en anclajes.[50]

Es adecuado para tareas de detección de objetos en tiempo real en diversas áreas de aplicación. También ofrece una gama de modelos pre entrenados para atender a diversas tareas y requisitos de rendimiento, lo que facilita encontrar el modelo adecuado para su caso de uso específico. Estos modelos están diseñados para satisfacer distintos requisitos, desde la detección de objetos hasta tareas más complejas como la segmentación de instancias, la detección de poses/puntos clave, la detección de objetos orientados y la clasificación. [50]

- ***Faster R-CNN***

Faster R-CNN se basa en trabajos anteriores para clasificar de manera eficiente propuestas de objetos utilizando redes convolucionales profundas. Emplea varias innovaciones para mejorar la velocidad de entrenamiento y test además que al mismo tiempo aumentar la precisión de la detección. Este modelo entrena la red VGG16 muy profunda 9 veces más rápido que R-CNN, es 213 veces más rápido en el momento de la prueba y logra un mAP más alto. [51]

Una imagen de entrada y múltiples regiones de interés también llamadas RoI se ingresan en una red totalmente convolucional. Cada RoI se agrupa en un mapa de características de tamaño fijo y luego se asigna a un vector de características mediante capas completamente conectadas o FC. La red tiene dos vectores de salida por RoI; probabilidades softmax y compensaciones de regresión del cuadro delimitador por clase. En la Figura 40 se esquematiza la arquitectura descrita. [51]

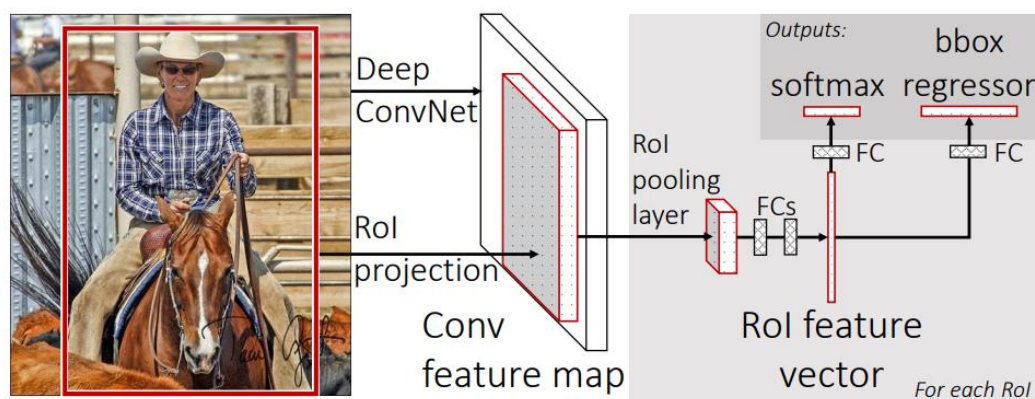


Figura 40. Arquitectura Fast R-CNN. [51]

Ahora bien, Faster R-CNN presenta una red de propuestas regionales (RPN) que comparte características convolucionales de imagen completa con la red de detección, lo que permite propuestas regionales de bajo costo computacional. Una RPN es una red totalmente convolucional que predice simultáneamente los límites de los objetos y las puntuaciones de objetividad en cada posición, está entrenado para generar propuestas de región de alta calidad, que Fast R-CNN utiliza para la detección.[52]

Además, fusiona RPN y Fast R-CNN en una sola red al compartir sus características convolucionales; utilizando mecanismos de "atención", el componente RPN le dice a la red unificada dónde buscar. Faster R-CNN y RPN son la base de las entradas ganadoras del primer lugar en varias pistas y el código está a disposición del público. La Figura 41 muestra el proceso de RPN también llamados mecanismos de atención. [52]

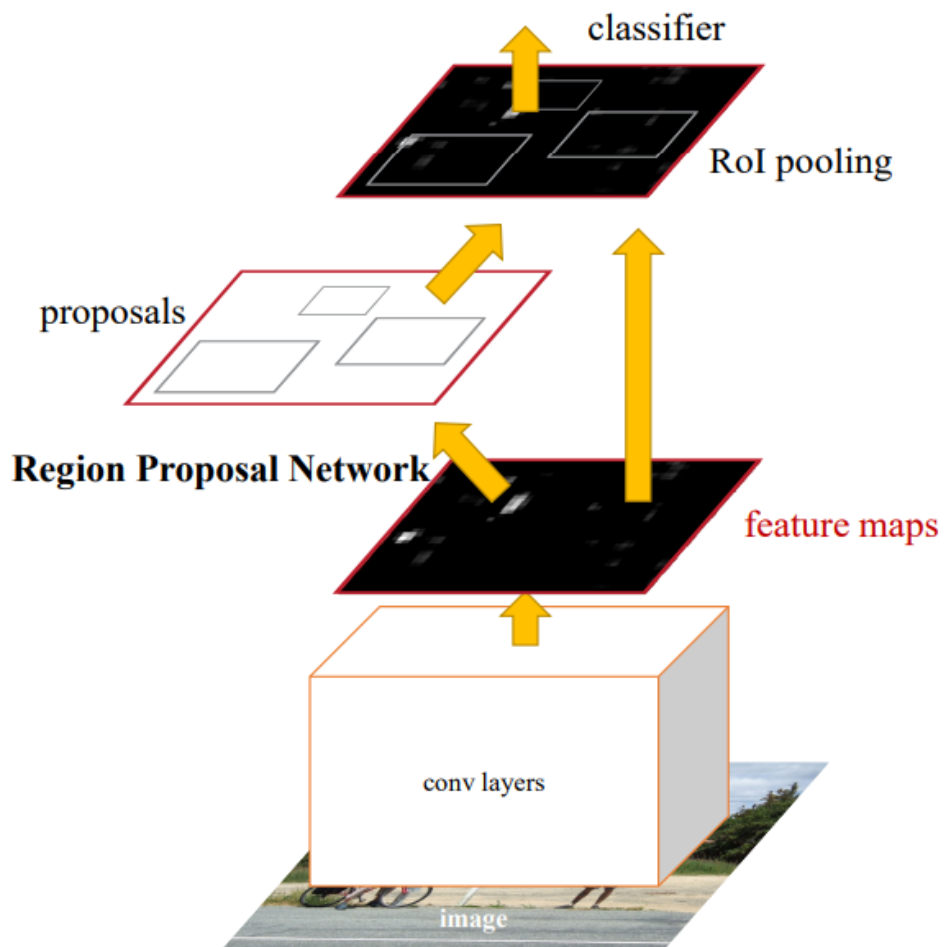


Figura 41. Mecanismos de atención en Faster R-CNN. [52]

- **Single Shot MultiBox Detector (SSD)**

Este modelo discretiza el espacio de salida de los cuadros delimitadores en un conjunto de cuadros predeterminados en diferentes relaciones de aspecto y escalas por ubicación del mapa de características. En el momento de la predicción, la red genera puntuaciones para la presencia de cada categoría de objeto en cada cuadro predeterminado y produce ajustes en el cuadro para que coincida mejor con la forma del objeto. [53]

Además, la red combina predicciones de múltiples mapas de características con diferentes resoluciones para manejar naturalmente objetos de varios tamaños. SSD es simple en relación con los métodos que requieren propuestas de objetos porque elimina por completo la generación de propuestas y las etapas posteriores de re muestreo de características o píxeles y encapsula todos los cálculos en una sola red. [53]

El procesamiento de SSD es el siguiente; Solo necesita una imagen de entrada y cuadros de verdad del terreno para cada objeto durante el entrenamiento, (a). De forma convolucional, evaluamos un pequeño conjunto de cuadros predeterminados de diferentes relaciones de aspecto en cada ubicación en varios mapas de características con diferentes escalas por ejemplo, 8×8 y 4×4 en (b) y (c). Para cada cuadro predeterminado, predecimos tanto los desplazamientos de forma como las confianzas para todas las categorías de objetos. En la Figura 42 esquematiza este procesamiento. [53]

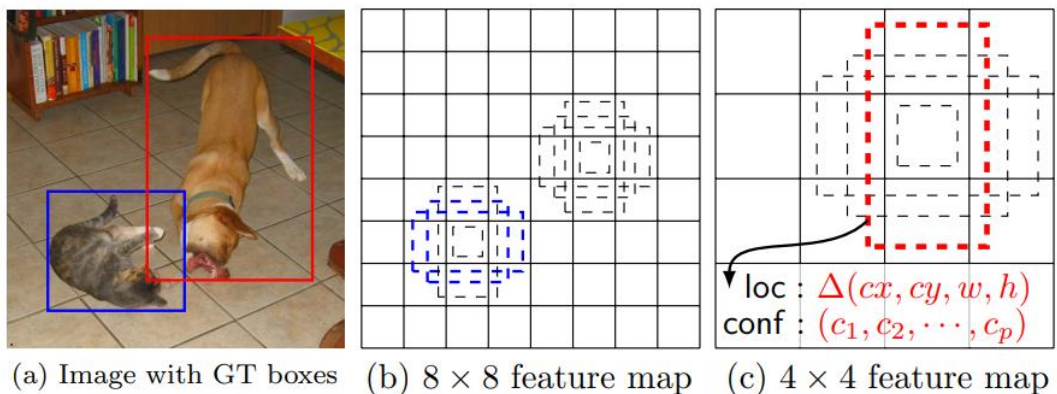


Figura 42. Funcionamiento de SSD. [53]

La Tabla 11 muestra la comparativa de las arquitecturas explicadas para la detección de objetos, evaluando parámetros como velocidad, precisión, requisitos de hardware entre otros parámetros.

Tabla 11. Modelos para detección de objetos.

Parámetro	YOLO	Faster R-CNN	SSD
Tipo de Arquitectura	Detección en tiempo real	Detección de precisión	Detección en tiempo real
Velocidad	Entre 45 y 155 FPS	Entre 5 y 20 FPS	Entre 30 y 60 FPS
Precisión	Buena, con menor desempeño en objetos pequeños	Alta, muy precisa	De buena a moderada
Estructura	Un solo modelo	Dos etapas regiones propuestas y clasificación	Un solo modelo
Uso de características	Captura de contexto global	Regiones de interés	Combina características de distintas capas
Tamaño del modelo	Pequeño	Grande	Moderado
Facilidad de entrenamiento	Fácil de entrenar	Requiere más recursos	Relativamente fácil
Requisitos de hardware	Bajo	Alto	Moderado
Mejor rendimiento	Videos e imágenes	Videos e imágenes	Videos e imágenes
Flexibilidad	Limitada con cajas de tamaños fijos	Muy flexible	Moderadamente flexible

c. Rastreo de objetos

El rastreo de objetos o también llamado “Tracking” es una funcionalidad muy importante especialmente en de detección de objetos en videos o secuencias de fotogramas, no solo limitándose a la detección de objeto objetivo sino también rastreando o siguiendo su trayectoria. A continuación se explican algunos algoritmos de rastreo de objetos más comunes.

- **Kernelized Correlation Filters**

El algoritmo de rastreo Kernelized Correlation Filter (KCF) está basado en características de Histograma de Gradientes Orientados en lugar de píxeles sin procesar. Este enfoque optimiza el rendimiento al aprovechar propiedades de matrices circulantes y la Transformada Discreta de Fourier (DFT), permitiendo realizar cálculos complejos en el dominio de Fourier de forma eficiente. [54]

El objeto que se va a rastrear se selecciona generalmente mediante un cuadro delimitador rectangular. El rastreador sigue el objeto en el video actualizando los parámetros del cuadro delimitador. La idea básica del seguimiento con filtro de correlación es estimar un filtro de imagen óptimo de modo que el filtrado con la imagen de entrada produzca una respuesta deseada, esta es típicamente de forma gaussiana centrada en la ubicación del objetivo, por lo que la puntuación disminuye con la distancia. [55]

El filtro se entrena a partir de instancias trasladadas del parche objetivo. Al realizar pruebas, se evalúa la respuesta del filtro y el máximo proporciona la nueva posición del objetivo. El filtro se entrena y actualiza sucesivamente con cada fotograma para que el rastreador se adapte a cambios moderados del objetivo. La ventaja principal del rastreador con filtro de correlación es la eficiencia de cálculo. La razón es que el cálculo se puede realizar de manera eficiente en el dominio de Fourier. Por lo tanto, el rastreador se ejecuta en tiempo real. [55]

- ***Channel and Spatial Reliability Tracker***

Es un algoritmo de aprendizaje con integración eficiente y sin problemas en la actualización del filtro y el proceso de seguimiento. Utiliza un mapa de confiabilidad espacial ajustando el soporte del filtro a la parte del objeto adecuada para el seguimiento. Esto permite ampliar la región de búsqueda y mejora el seguimiento de objetos no rectangulares. [56]

Los puntajes de confiabilidad reflejan la calidad de los filtros aprendidos por canal y se utilizan como coeficientes de ponderación de características en la localización. Experimentalmente, con solo dos conjuntos de características estándar simples, HoGs y nombres de colores, el novedoso método CSR-DCF logra resultados de vanguardia en VOT 2016, VOT 2015 y OTB100. Además se ejecuta casi en tiempo real en una CPU. [56]

- ***SORT***

El método es un algoritmo de seguimiento de objetos que se describe mediante los componentes clave de detección, propagación de estados de objetos en fotogramas

futuros, asociación de detecciones actuales con objetos existentes y gestión de la vida útil de los objetos rastreados. Para la etapa de detección se puede utilizar cualquier modelo basado en CNN.[57]

La representación y el modelo de movimiento utilizados para propagar la identidad de un objetivo al siguiente fotograma, se hace mediante aproximaciones de desplazamientos entre fotogramas de cada objeto con un modelo de velocidad constante lineal que es independiente de otros objetos y del movimiento de la cámara. El estado de cada objetivo se modela de la siguiente manera: [57]

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T \quad (3)$$

Donde u y v son la ubicación de píxeles horizontal y vertical del centro del objetivo , mientras que la escala s y r representan el área y la relación de aspecto del cuadro delimitador del objetivo respectivamente, la relación de aspecto se considera constante. Cuando una detección se asocia a un objetivo, el cuadro delimitador detectado se utiliza para actualizar el estado del objetivo donde los componentes de velocidad se resuelven de manera óptima a través de un filtro de Kalman. [57]

Si no hay ninguna detección asociada al objetivo, su estado simplemente se predice sin corrección utilizando el modelo de velocidad lineal. Al asignar detecciones a objetivos existentes, la geometría del cuadro delimitador de cada objetivo se estima al predecir su nueva ubicación en el fotograma actual. Luego, la matriz de costo de asignación se calcula como la distancia de intersección sobre unión (IOU) entre cada detección y todos los cuadros delimitadores predichos de los objetivos existentes. [57]

Cuando los objetos entran y salen de la imagen, se deben crear o destruir rastreadores únicos, considerando que cualquier detección con una superposición menor que IOU mínimo significa la existencia de un objeto sin seguimiento. El rastreador se inicializa utilizando la geometría del cuadro delimitador con la velocidad establecida en cero, además pasa por un período de prueba en el que el objetivo debe asociarse con detecciones para acumular evidencia suficiente evitando el seguimiento de falsos positivos. [57]

La Tabla 12 compara los algoritmos de rastreo explicados.

Tabla 12. Algoritmos para rastreo en detección de objetos.

Parámetro	KCF	CSRT	SORT
Tipo	Seguimiento monocular	Seguimiento monocular	Seguimiento múltiple
Velocidad	Muy rápido	Rápido	Muy rápido
Precisión	Buena	Alta	Buena
Robustez ante oclusiones	Moderada	Alta	Moderada
Capacidad para variantes	Limitada	Alta	Buena
Modelo de objeto	Adaptativo	Adaptativo	No adaptativo
Costo computacional	Bajo	Moderado	Bajo
Uso de características	Características de imagen	Características de imagen	Predicción básica
Análisis temporal	Limitado, sin seguimiento múltiple	Limitado, sin seguimiento múltiple	Predicción de movimiento
Aplicaciones típicas	Seguimiento de objetos en video	Detección y seguimiento en tiempo real	Detección y seguimiento en tiempo real

d. Unidades de procesamiento

Existen muchas unidades o módulos de procesamiento capaces de correr modelos de Machine Learning para aplicaciones de visión artificial, su parámetro más importante sin duda es la unidad de procesamiento de gráficos o GPU la cual permite correr modelos más complejos de forma fluida. A continuación se detallan las unidades de procesamiento más comunes en aplicaciones de visión artificial.

- **Nvidia Jetson Nano**

Es un Sistema en Módulo (SOM) con una dimensión de 70×45 mm, ofrece grandes resultados cuando se trata de implementar la IA en dispositivos de borde de múltiples industrias, desde ciudades y fábricas inteligentes hasta la agricultura y la robótica. El Kit para Desarrolladores Jetson Nano es una computadora pequeña y poderosa que te permite ejecutar múltiples redes neuronales en paralelo para aplicaciones como clasificación de imágenes, detección de objetos, segmentación y procesamiento de voz. [58]

Jetson Nano es compatible con NVIDIA JetPack™ con el mismo conjunto de software CUDA-X™ que se utiliza para productos innovadores basados en IA en todas las industrias. JetPack incluye las últimas herramientas de NVIDIA para el desarrollo y la

optimización de aplicaciones y es compatible con tecnologías nativas de entornos Cloud, como la organización en contenedores y la orquestación para simplificar el desarrollo y las actualizaciones.[58]

Los entornos de desarrollo creados con JetPack brindan a los desarrolladores de Jetson un mayor rendimiento y escalabilidad a medida que crean productos finales específicos de las industrias. Estos incluyen NVIDIA DeepStream y Transfer Learning Toolkit para el análisis de video inteligente, NVIDIA Clara™ para la generación de imágenes de atención médica, la genómica y el monitoreo de pacientes, y NVIDIA Isaac™ para la robótica. [58]

- ***Coral Dev Board***

Esta es una placa de desarrollo de placa única que es ideal para realizar inferencias rápidas de Machine Learning en un formato pequeño. Se puede usar la placa de desarrollo para crear prototipos de su sistema integrado y luego escalarlo a producción utilizando el módulo de sistema Coral integrado combinado con su hardware de PCB personalizado. [59]

El coprocesador Edge TPU integrado es capaz de realizar 4 billones de operaciones por segundo (TOPS), utilizando 0.5 vatios por cada TOPS. Además cuenta con SoC + ML + conectividad inalámbrica, todo en la placa que ejecuta un derivado de Debian Linux llamado Mendel, para que pueda ejecutar cualquier herramienta Linux con esta placa. Los modelos de TensorFlow Lite se pueden compilar para ejecutarse en Edge TPU. [59]

- ***Rock PI 4***

ROCK Pi 4 de Radxa es un mini computador de una pieza que se basa en el procesador Rockchip RK3399. Es compatible con Android y algunas Distribuciones de Linux. Las características del ROCK Pi 4 son: Un procesador de seis núcleos ARM, una memoria de 64bit de doble canal 3200Mb/s LPDDR4, salida HDMI hasta 4K@60hz, MIPI DSI, MIPI CSI, salida de auriculares 3.5mm jack con micrófono, WIFI 802.11 ac, BT 5.0, Puertos USB, Red GbE, GPIO a color de 40 pines compatible con PI

estándar, RTC. También soporta USB PD y energía mediante QC. [60] La Tabla 13 muestra la comparativa de las unidades de procesamiento explicadas.

Tabla 13. Unidades de procesamiento para visión artificial.

Parámetro	NVIDIA Jetson Nano	Coral Dev Board	ROCK Pi 4
Procesador	Quad-core ARM Cortex-A57	Quad-core Cortex-A53	Hexa-core ARM Cortex-A72 (2) + Cortex-A53 (4)
GPU	128 CUDA cores (NVIDIA Maxwell)	GPU integrado (Mali-T860)	Mali-T860
Memoria	4 GB LPDDR4	1 GB DDR3L (versión) / 2 GB (versión superior)	1 GB / 2 GB / 4 GB LPDDR4
Almacenamiento	16 GB eMMC + ranura microSD	8 GB eMMC + ranura microSD	MicroSD + USB 3.0 + SATA (con adaptador)
Conectividad	Ethernet 1 Gbps	Wi-Fi 802.11ac, Bluetooth 4.1	Wi-Fi 802.11ac, Bluetooth 5.0
Puertos USB	4 x USB 3.0	1 x USB 3.0, 1 x USB 2.0	2 x USB 3.0, 2 x USB 2.0
Salida de video	HDMI 2.0, DisplayPort 1.2	HDMI 2.0, MIPI DSI	HDMI 2.0, MIPI DSI
GPIO	40 pines (compatible con Raspberry Pi)	40 pines (compatible con Raspberry Pi)	40 pines (compatible con Raspberry Pi)
Sistema operativo	Ubuntu basada en L4T (Linux for Tegra)	Mendel Linux (basado en Debian)	Android y varias distribuciones de Linux
Consumo de energía	5-10 W	3-7 W	5-12 W

1.3.7 Gestión de información

El sistemas gestor de información es el software capaz de procesar la información tanto de bakend con el fronted de un proyecto. Ademas almacena esta información con la finalidad de mostrarla o analizarla en cualquier momento. A continuación se explican varios entornos de desarrollo para este tipo de sistemas.

a. Entornos de desarrollo

- **Django**

Django es un entorno de desarrollo web de alto nivel basado en Python que fomenta el desarrollo rápido y un diseño limpio y pragmático. Creado por desarrolladores experimentados, se encarga de gran parte de las complicaciones del desarrollo web,

para que puedas concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto. [61]

Las principales herramientas que el mismo Django proporciona son un conjunto de scripts de Python para crear y trabajar con proyectos Django, junto con un simple servidor web de desarrollo que puedes usar para probar de forma local aplicaciones web Django con el explorador web del computadora. Además es compatible con Windows, Linux (Ubuntu), y Mac OS X. [62]

- ***Flask***

Flask es un marco de desarrollo web ligero. Está diseñado para que comenzar sea rápido y fácil, con la capacidad de escalar a aplicaciones complejas. Depende del conjunto de herramientas WSGI Werkzeug, el motor de plantillas Jinja y el conjunto de herramientas CLI Click. Flask proporciona configuración y convenciones, con valores por defecto sensatos, para empezar. Se recomendada de Python es 3.9 o superior. [63]

Werkzeug implementa WSGI, la interfaz estándar de Python entre aplicaciones y servidores, Jinja y MarkupSafe para renderizar las páginas de la aplicación y evitar ataques de inyección, ItsDangerous firma de forma segura los datos para asegurar su integridad y proteger la cookie de sesión de Flask. Click como marco para escribir aplicaciones de línea de comandos permitiendo añadir comandos de gestión personalizados y Blinker que proporciona soporte para Señales. [63]

Se recomienda un entorno virtual para gestionar las dependencias del proyecto, tanto en desarrollo como en producción evitando que las nuevas versiones de las librerías para un proyecto pueden romper la compatibilidad en otro proyecto. Los entornos virtuales son grupos independientes de bibliotecas Python, uno para cada proyecto. Los paquetes instalados para un proyecto no afectarán a otros proyectos o a los paquetes del sistema operativo. [63]

- ***Pyramid***

Pyramid es un entorno que facilita la creación de aplicaciones web. Se puede empezar con esta aplicación web de solicitud/respuesta ofrece muchas funciones que hacen que

escribir software complejo requiera menos esfuerzo, funciona en todas las versiones compatibles de Python. [64]

Los desarrolladores pueden sumergirse en la documentación narrativa de Pyramid o explorar la extensa referencia de API . Pyramid tiene una gran cantidad de recursos útiles de los cuales sacar provecho. Extending Pyramid es una lista seleccionada y filtrable de complementos, paquetes y aplicaciones diseñados para funcionar con Pyramid. [64] La Tabla 14 muestra la comparativa de los entornos de desarrollo explicados.

Tabla 14. Entornos de desarrollo para el Sistema Gestor de Información.

Parámetro	Django	Flask	Pyramid
Tipo	Full-Stack	Micro-Framework	Full-Stack
Curva de aprendizaje	Moderada	Baja	Moderada
Rendimiento	Bueno	Muy bueno	Bueno
Soporte de ORM	Integrado (Django ORM)	No integrado, opcional	Opcional (SQLAlchemy)
Autenticación	Integrada	No integrada, requiere extensiones	Opcional (extensiones)
Seguridad	Alta (protección integrada)	Media (depende de extensiones)	Media
Comunidad	Muy activa	Muy activa	Activa
Documentación	Extensa y detallada	Buena	Buena
Componentes Integrados	Múltiples (administración, autenticación, etc.)	Muy pocos (control total)	Múltiples (extensibles)
Asincronía	Limitada	No soporta	Limitada

1.4 Objetivos

1.4.1 Objetivo general

Implementar un sistema de alerta temprana de eventos delictivos en entidades económicas mediante el uso de Visión Artificial y algoritmos de Deep Learning.

1.4.2 Objetivos específicos

- Identificar la estructura general de los sistemas de seguridad existentes, estableciendo falencias y vulnerabilidades.

- Determinar la arquitectura, hardware y software del sistema de detección temprana de eventos delictivos.
- Desarrollar el sistema de Visión Artificial entrenando el modelo Deep Learning para detección de temprana de eventos delictivos.
- Elaborar un Sistema de Gestión de Información para visualización de métricas de vigilancia y alertas de eventos sospechosos.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

En el desarrollo de la investigación se utilizó hardware y software específico para implementaciones de visión artificial. Para la obtención de las señales de video se utilizó una cámara de seguridad IP Modelo SN-MFPT20-ICS de la marca Sinovision con resolución de 1080p, estas señales fueron analizadas por el sistema de detección temprana de eventos delictivos alojado en el dispositivo Nvidia Jetson Nano, el sistema utilizó librerías como OpenCV para el pre procesamiento del video, MediaPipe Pose Landmark para la estimación de postura, un modelo YOLO entrenado con el conjunto de datos de objetos sospechosos mediante técnicas de Transfer Learning y un modelo SVM entrenado para detectar posturas amenazantes. Para la visualización y gestión de la información se utilizó el entorno de desarrollo FLASK para desarrollar una interfaz de usuario amigable e intuitiva, además se programó un Bot de Telegram que envía notificaciones de personas sospechosas y delito inminente, por último se guardó las detecciones en una base de datos con SQLite.

2.2 Métodos

2.2.1 Modalidad de la investigación

En esta sección se detalla qué tipos de investigación se utilizó en el desarrollo de la investigación, así como el proceso que implicó este fin. Se realizó una investigación Aplicada, debido a que se utilizaron herramientas, tanto de hardware como de software ya existentes, así mismo el conocimiento que hasta el momento se tiene respecto a Visión Artificial y Deep Learning. Esto hace que el proyecto de investigación, no genere nuevo conocimiento, sino por el contrario, aplique el ya existente a la resolución de un problema.

Por otro lado, también se realizó una investigación bibliográfica, ya que se utilizaron algoritmos, técnicas y modelos publicados en trabajos investigativos anteriores, como artículos científicos, trabajos de investigación, repositorios GitHub, entre otros. Así

mismo, se fundamentó teóricamente la investigación con conocimiento publicado en libros y revistas científicas.

Por último, el proyecto fue a su vez una investigación de campo, debido a que se recolectaron datos acerca de los sistemas de seguridad existentes, con el fin de establecer puntos de partida y parámetros evaluables en la implementación, así como pruebas de funcionamiento en entornos reales, facilitando así la validación del sistema de detección temprana de eventos delictivos en entidades comerciales.

2.2.2 Población y muestra

La investigación incluye la implementación de un prototipo libre, por lo tanto, se puede llevar a cabo en cualquier institución comercial, y no requirió de población y muestra.

2.2.3 Recolección de información

El desarrollo de la investigación requirió gran cantidad de datos así como de información teórica, los conjuntos de datos utilizados en el entrenamiento de modelos de Machine Learning fueron recolectados de varias fuentes, entre las cuales están; Recolección en buscadores especializados en conjuntos de datos para Data Science, Plataformas de desarrollo de Machine Learning, Google Imágenes, y redes sociales como Telegram, Facebook, X y YouTube.

Por otro lado también se crearon varias secuencias de video en las cuales se dramatizaron situaciones delictivas, estas secuencias realizadas por el investigador. Además la información teórica necesaria para fundamentar esta investigación fue recolectada de repositorios, bibliotecas y foros especializados en implementaciones de Inteligencia Artificial.

Por último también se realizó una recolección de información mediante entrevistas al personal de seguridad, lo que nos permitió establecer puntos de partida, tales como arquitecturas, hardware y software necesarios, entre otros. La Tabla 15 muestra los criterios de recolección de información.

Tabla 15. Criterios de recolección de información.

Criterio	Descripción
¿Para qué?	Detectar tempranamente actitudes sospechosas o eventos delictivos potenciales
¿De qué personas u objetos?	Para entidades comerciales de la ciudad de Ambato
¿Como?	A través del uso de visión artificial y algoritmos de deep Learning
¿Con que?	Con un sistema de detección temprana de eventos delictivos
¿Cuándo?	En el periodo de octubre 2024 a enero 2025
¿Dónde?	En entidades comerciales de la ciudad de Ambato

2.2.4 Procesamiento y análisis de datos

La información obtenida en la investigación fue procesada de la siguiente manera:

- Se realizó una búsqueda bibliográfica de implementaciones anteriores de sistemas de seguridad similares al propuesto.
- Se contrastó las metodologías y tecnologías utilizadas en dichas implementaciones, definiendo las más óptimas y eficientes.
- Se estableció un conjunto de datos limpio y representativo, es decir preprocesándolo antes de utilizarlo en el entrenamiento del modelo.
- Se evaluó el funcionamiento del modelo mediante métricas de validación, así como también en entornos comerciales.
- Se documentó el desarrollo de esta investigación mediante un informe de la metodología utilizada y resultados obtenidos.

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

La investigación se basó en la metodología CRISP-DM , la cual fue elegida en base a la Tabla 5 mostrada en el Capítulo I. La primera etapa es Situación actual, en la cual detalla el esquema general, hardware y software necesario, después se tiene la etapa llamada Datos e información disponible en la cual se detalla el proceso de recolección de datos necesarios, seguido la etapa de Conjunto de los datos en donde se detalla la limpieza y etiquetado de los mismo.

En la etapa de Modelado se muestran los resultados del entrenamiento así como el sistema gestor de información, después tenemos la etapa de Evaluación del sistema en la que se detallan las pruebas y métricas de validación realizadas al sistema y por último la etapa de Despliegue en la que se muestra el diagrama funcional así como los costos de implementación del sistema.

3.1 Situación actual

Entre los sistemas de seguridad más utilizados según la encuesta se encuentran; Sistemas CCTV con un 100% de encuestados, sensores de movimiento con un 88.9% y Alarmas de intrusos con un 77.8%, por otro lado, un 11.1% utilizan sistemas de reconocimiento facial y el 66,7% cuenta con controles de acceso. Además el 100% de los encuestados cuenta con al menos un tipo de sistema electrónico de seguridad. La Figura 43 muestra esta distribución.

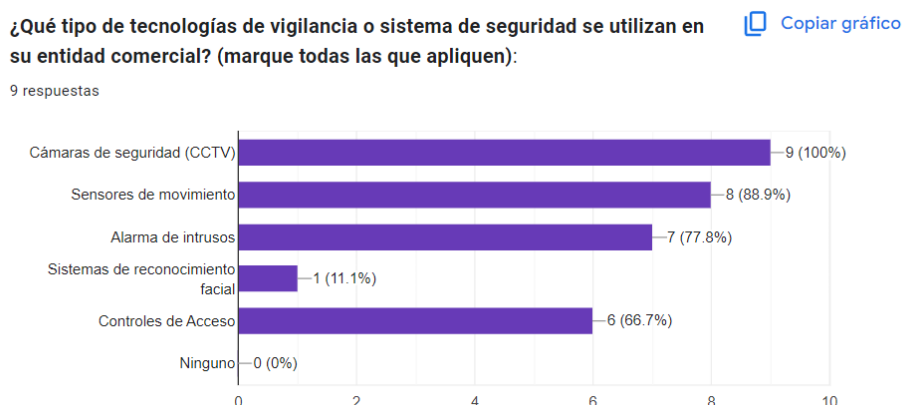


Figura 43. Sistemas de seguridad más utilizados entre los encuestados.

Los sistemas de seguridad precisan de monitorización, de los encuestados el 66.7% monitorea las cámaras de seguridad todo el tiempo, el 11.1% hace revisiones periódicas y el 22.2% solo lo hace cuando ocurre un incidente. Lo que muestra claramente que los sistemas de seguridad actuales dependen de la revisión humana para tener niveles de efectividad aceptables. La Figura 44 muestra los resultados obtenidos.

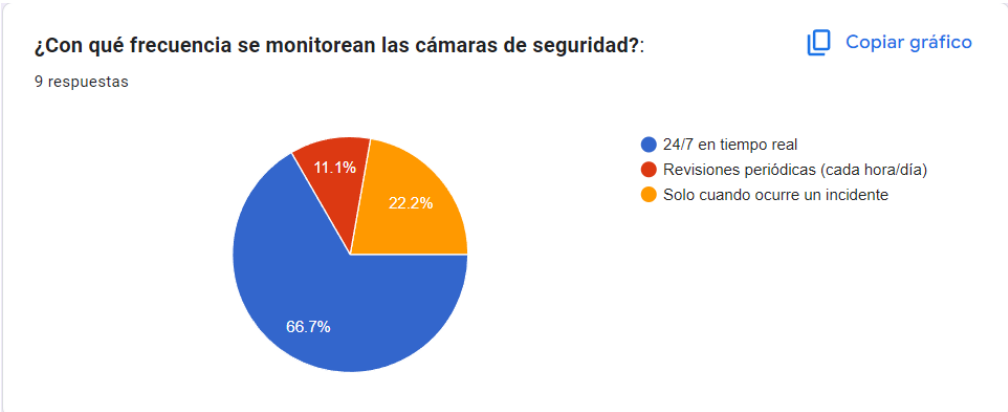


Figura 44. Frecuencia de monitorización entre los encuestados.

De los encuestados el 55.6% reporta ocasionalmente ha existido delitos que no fueron detectados por el sistema de seguridad, el 11.1% reporta que los incidentes no detectados fueron frecuentes y un 33.3% no reporta incidentes no detectados por el sistema de seguridad actual, evidenciando que la efectividad del sistema en ciertos casos no llega a ser convincente. La Figura 45 muestra la distribución obtenida.

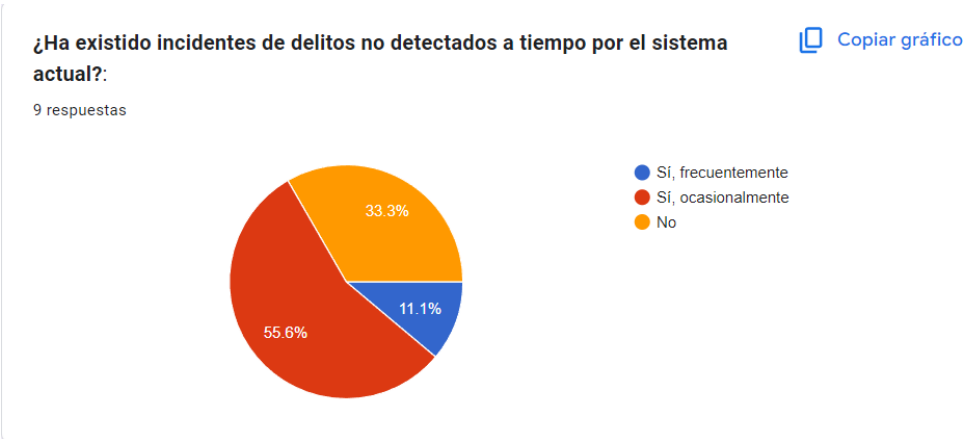


Figura 45. Delitos no detectados a tiempo entre los encuestados.

Respecto a los factores limitantes del sistema de seguridad, de los encuestados el 100% considera que la cobertura insuficiente es un factor limitante, el 77.8% cree que es la

dependencia humana, el 33.3 % considera el retardo en la respuesta ante incidentes y la baja calidad de imagen/video y el 11.1% elige la falta de energía eléctrica, falta de análisis automatizado, y otros factores. Lo que muestra que la cobertura ineficiente es el factor más determinante en la efectividad del sistema, según los encuestados. La Figura 46 muestra los resultados obtenidos.

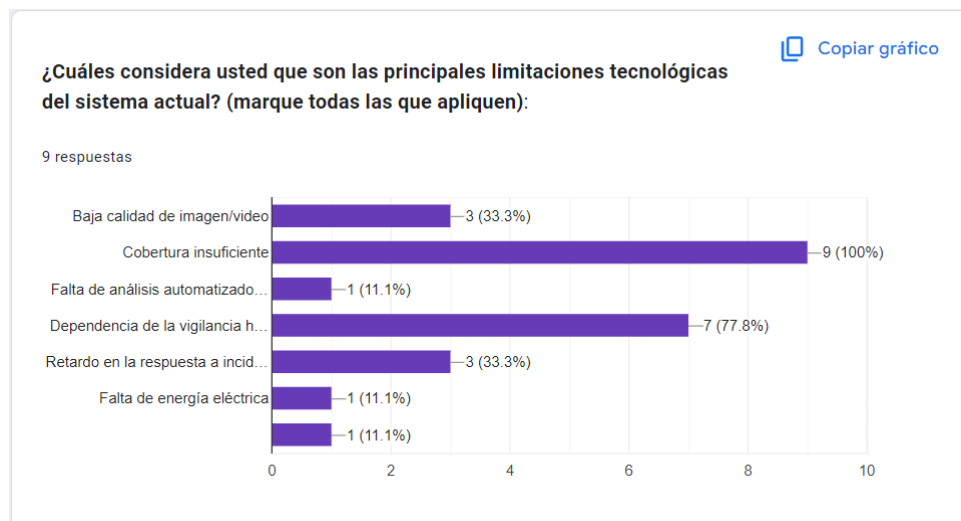


Figura 46. Limitaciones de los sistemas de seguridad entre los encuestados.

Los encuestados califican su sistema de seguridad en una escala de 1 a 10 de la siguiente manera; el 44.4% lo califica con un 8 y el 22.2% con un 9 y 7 respectivamente y el 11.1% con 6. Lo que evidencia claramente que la percepción de los encuestados respecto a su sistema de seguridad oscila entre 6 y 9 con un pico en 8 tomando en cuenta una escala de 1 a 10 siendo 1 muy inefectivo y 10 muy efectivo. La Figura 47 muestra los resultados obtenidos.

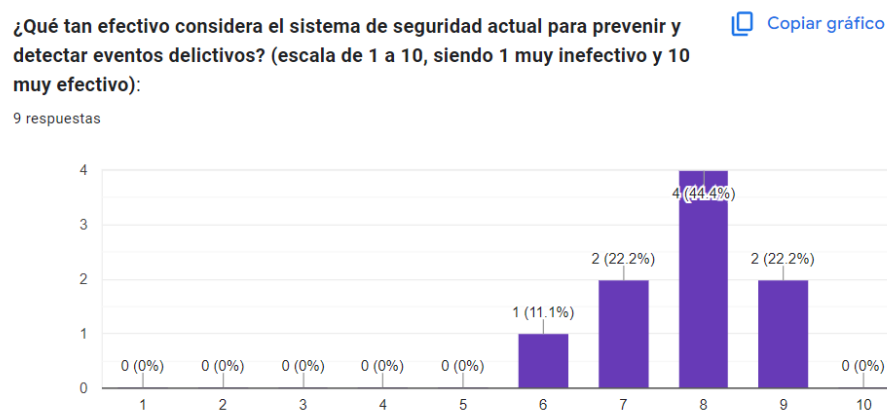


Figura 47. Nivel de aceptación del sistema de seguridad actual entre los encuestados.

Los resultados obtenidos ayudan a entender la situación actual de los sistemas de seguridad en entidades comerciales, se puede concluir que el sistema más común son los circuitos cerrados de televisión, pero gran parte de los encuestados tienen además otros sistemas complementarios como sensores de movimientos y alarmas, por otro lado, se observa que un gran porcentaje monitorea los sistemas de seguridad 24 horas los 7 días a la semana, aun así más de la mitad reporta que ha existido delitos no detectados a tiempo. Lo que puede representar viabilidad para la implementación del proyecto implementado en esta investigación.

3.1.1 Esquema general

El esquema general brinda una visión general del flujo de información que presenta el sistema de detección temprana de eventos delictivos, consta de varias etapas, las cuales son básicas en un sistema de visión artificial y fueron detalladas en el Capítulo I. La Figura 48 muestra la estructura general y las etapas clave que se usó en el sistema propuesto en esta investigación.

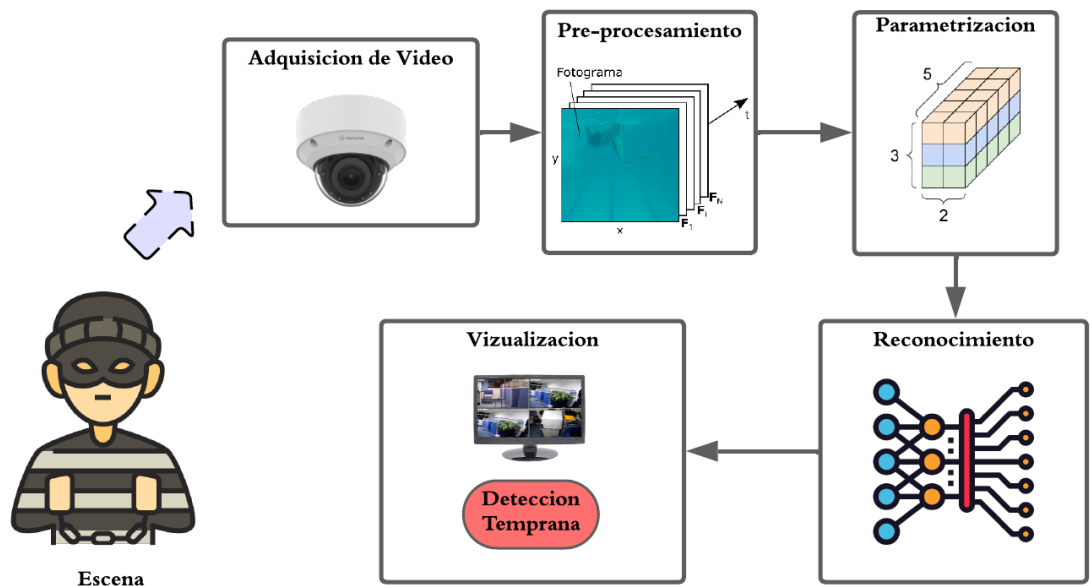


Figura 48. Esquema general del sistema propuesto.

Adquisición de video. Es la encargada de la captación de video en tiempo real por medio de cámaras de video vigilancia, esta información es convertida a señales eléctricas que son enviadas al dispositivo de procesamiento de la siguiente etapa.

Preprocesamiento. Toma las señales eléctricas provenientes de la etapa de captación y las convierte a fotogramas digitales, realiza un filtrado para reducir ruido a las imágenes y crea secuencias fijas de fotogramas y las envía a la siguiente etapa.

Parametrización. Los fotogramas preprocesados en la etapa anterior son parametrizados con el fin de adaptarlos a las dimensiones que acepta la capa de entrada del modelo de la etapa de reconocimiento.

Reconocimiento. Los eventos delictivos son detectados en función a los datos parametrizados de la etapa anterior, se maneja un enfoque multimodal; “Procesamiento de postura” que da una predicción de postura amenazante mediante modelo de detección de postura corporal, y “Procesamiento de objeto” que detecta objetos relacionados con eventos delictivos.

Visualización. Es la etapa encargada de mostrar el video en tiempo real captado por la etapa de Adquisición juntamente con las predicciones o alertas tempranas provenientes de la etapa de Reconocimiento.

3.1.2 Requisitos del proyecto

Aquí se detallan los requisitos del sistema de detección temprana de eventos delictivos, empezando por los recursos y datos necesarios, describiendo a breves rasgos su proceso recolección y creación, también se habla de la capacidad del sistema, y la tecnología que hace posible que se consiga esto.

a. Recursos y datos

Se preciso un conjunto de datos representativo para el entrenamiento de modelos detectores de objetos así como clasificadores de posturas, estos datos fueron recolectados de conjunto de datos públicos, redes sociales informativas, foros de noticias, entre otros. También se crearon secuencias propias simulando situaciones

delictivas con el fin de robustecer el conjunto de datos y mejorar la generalización de modelo.

b. Capacidad

El sistema debe ser capaz de detectar individuos sospechosos en función de los objetos que lleve consigo, así como de la postura corporal que presente. Por otro lado también debe alertar oportunamente de la presencia de objetos sospechosos, armas, y posturas amenazantes, por último estas detecciones deben guardarse en una base de datos para su posterior análisis, además debe contar con un sistema gestor de información que lleve a cabo estos requerimientos.

c. Tecnología

El sistema de detección temprana de eventos delictivos propuesto en esta investigación, requiere de hardware y software especializados como cámaras de seguridad con buena resolución, unidades de procesamiento con gran capacidad de cómputo, librerías especializadas en aplicaciones de Visión Artificial y Deep Learning, así como entornos de desarrollo de sistemas de gestión de información.

3.1.3 Hardware y software

Para cumplir con el objetivo de cada etapa descrita anteriormente se realizó una comparación de tecnologías de Hardware y Software disponibles en el mercado teniendo en cuenta parámetros técnicos, costo, documentación entre otros aspectos relevantes para la elección de una de estas tecnologías. A continuación, se detalla con mayor profundidad cada etapa, así como sus requerimientos.

a. Adquisición de video

En la fundamentación teórica se detalló la estructura de los sistemas CCTV, una de las partes importantes de estos sistemas son las cámaras o dispositivos de captación de video. El sistema propuesto en esta investigación requiere de una calidad de video aceptable similar al utilizado en los sistemas de videovigilancia actuales, la Tabla 2 detallada en el capítulo I, muestra la comparativa entre varias cámaras utilizadas en sistemas de videovigilancia.

En base a esta comparativa se seleccionó la cámara IP SN-MFPT20-ICS de la marca SINOVISION ya que presenta una resolución apropiada para aplicaciones de visión artificial, además cuenta con protocolo RTSP, el cual es necesario para procesar la señal de video mediante Python u otros lenguajes capaces de procesar video. Por último, al ser una cámara IP permite una conexión con el servidor de procesamiento más sencilla. La Tabla 16 resumen las características técnicas de la cámara y el Anexo B muestra los detalles físicos de la misma.

Tabla 16. Características técnicas cámara IP SN-MFPT20-ICS.

Parametro	Descripcion
Marca	SINOVISION
Modelo	MFPT20
Clasificacion de Resistencia al agua	IP66
Movimineto	PTZ
Control	App, Pc, Remoto
Conectividad	IP, LAN, Wi-Fi
Resolucion	2MP
Distancia focal de lente	3.6 mm
Sensor de imagen	1/2.7" BSI Sensor
Resolucion de video	Full HD
Compresion de video	H265
Tipo de almacenamiento	Micro-SD, Nube, NVR
Capacidad de almacenamieto	64GB
Alimentacion	12 V
Consumo	2.1W
Asistentes virtuales compatibles	Alexa, Google Assistant, Siri

b. Unidad de procesamiento

Se utilizó una unidad de procesamiento para desarrollar las etapas de Preprocesamiento de video, segmentación y parametrización de los datos, así mismo para desplegar el modelo de Deep Learning correspondiente a la etapa de Reconocimiento, obtener una predicción y mostrarla en el Sistema Gestor de Información implementado en la etapa de Visualización.

En base a la Tabla 13 mostrada en el capítulo I, donde se comparan algunas de la unidades de procesamiento más utilizadas en aplicaciones de Machine Learning se seleccionó al Modulo Nvidia Jetson Nano de 4 Gb de memoria RAM 64-Bit LPDDR4 con 128-Core NVIDIA Maxwell GPU, la cual es capaz de correr inferencias de modelos de Machine Learning, además es compatible con Frameworks como Torch,

Tensorflow y Python lo que la hace adecuada para la implementación de esta investigación. En el Anexo C se muestran las especificaciones de este módulo.

c. *Reconocimiento*

El reconocimiento se realiza mediante un enfoque multimodal de dos ramas de procesamiento; Rama Postura que detecta posturas amenazantes como “Apuntar con un arma” o “Amenazar con un cuchillo”, y Rama Objetos la cual se encarga de detectar 5 objetos sospechosos como “Arma”, “Cuchillo”, “Gorra”, “Mascarilla” y “Pasamontaña”. La Figura 49 muestra el esquema de procesamiento de este enfoque.

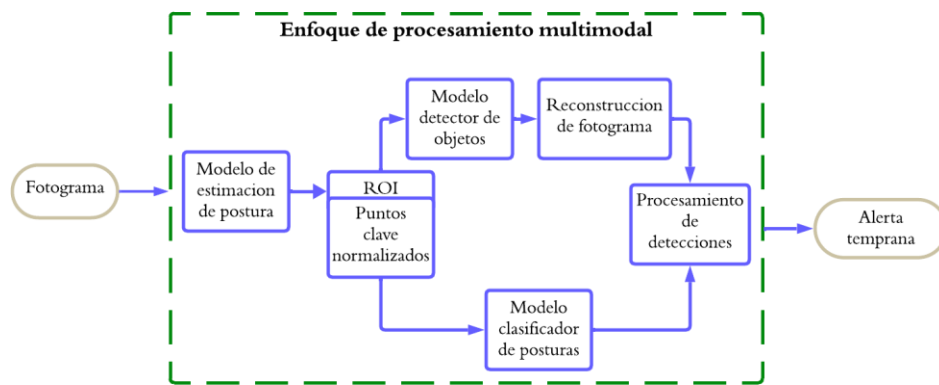


Figura 49. Flujo de procesamiento para etapa de reconocimiento.

Las predicciones de cada etapa son evaluadas mediante una lógica de procesamiento, respecto al procesamiento de objetos se asignó ponderaciones específicas a cada objeto detectado para detectar sospechosos, por otro lado, las predicciones del procesamiento de postura son evaluadas siempre y cuando se detecte un arma o un cuchillo dentro de la región de interés, esto ya que las posturas dependen del contexto de la escena y podrían ser interpretadas de forma incorrecta.

- **Análisis preliminar**

Previo al modelado del sistema para detección temprana de eventos delictivos, se realizaron exploraciones preliminares de enfoques que podrían cumplir con los objetivos del sistema. Se entrenó un modelo CNN + LSTM para detectar secuencias de video anómalas. Se planteó este enfoque para encontrar patrones espacio-temporales en secuencias de video de 50 fotogramas. Se aplicó una capa convolucional “TimeDistributed” a cada fotograma de la secuencia, las cuales arrojan características

extraídas, las cuales son analizadas por una capa LSTM. La Figura 50 muestra este modelo para detección multiclase.

Layer (type)	Output Shape	Param #
time_distributed (TimeDistributed)	(None, 50, 62, 62, 16)	448
time_distributed_1 (TimeDistributed)	(None, 50, 31, 31, 16)	0
time_distributed_2 (TimeDistributed)	(None, 50, 29, 29, 32)	4,640
time_distributed_3 (TimeDistributed)	(None, 50, 14, 14, 32)	0
time_distributed_4 (TimeDistributed)	(None, 50, 6272)	0
lstm (LSTM)	(None, 64)	1,622,272
dropout (Dropout)	(None, 64)	0
dense (Dense)	(None, 128)	8,320
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 11)	1,419

Figura 50. Modelo preliminar CNN+LSTM.

Este modelo se entrenó con un conjunto de datos de 1350 secuencias de 50 fotogramas cada una, extraídas del conjunto de datos UCF Crime. Se realizó pruebas para detección binaria, es decir “evento delictivo” y “evento normal”, y también clasificación multiclase, es decir con 10 tipos de anomalías y eventos normales. Los resultados obtenidos después del entrenamiento, mostraban un claro signo de sobre ajuste después de alcanzar el 70% de precisión, la Figura 51 muestra las curvas de aprendizaje. Además este enfoque solo identifica la posible ocurrencia de un evento delictivo, lo que es muy pobre para un sistema gestor de información.

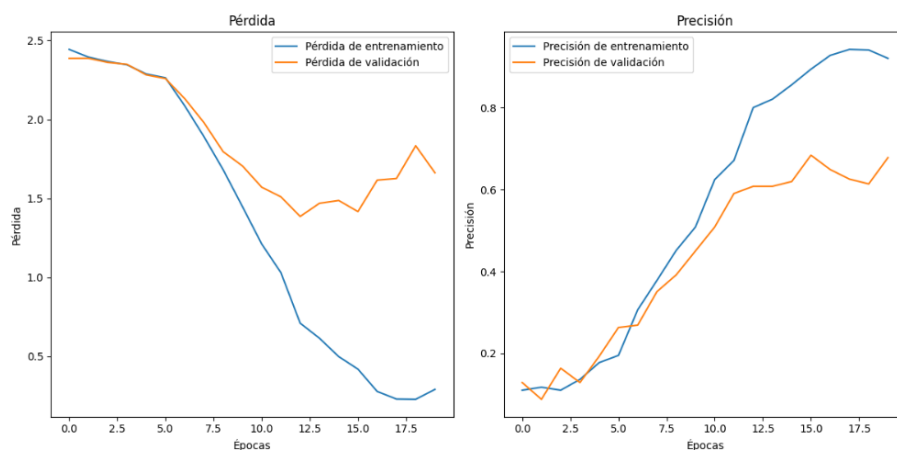


Figura 51. Curvas de aprendizaje enfoque CNN+LSTM.

Ademas, se hicieron pruebas utilizando Resnet50 como capa convolucional, aprovechando sus pesos pre entrenados, los resultados del entrenamiento mejoraron en cierto punto pero no lo suficiente. La Figura 52 las capas y parámetro de este enfoque y la Figura 53 muestras las curvas de aprendizaje del mismo.

Layer (type)	Output Shape	Param #
time_distributed (TimeDistributed)	(None, 50, 2, 2, 2048)	23,587,712
time_distributed_1 (TimeDistributed)	(None, 50, 8192)	0
lstm (LSTM)	(None, 32)	1,052,800
dropout (Dropout)	(None, 32)	0
dense (Dense)	(None, 11)	363

Total params: 24,640,875 (94.00 MB)
Trainable params: 1,053,163 (4.02 MB)
Non-trainable params: 23,587,712 (89.98 MB)

Figura 52. Modelo Resnet50 + LSTM.

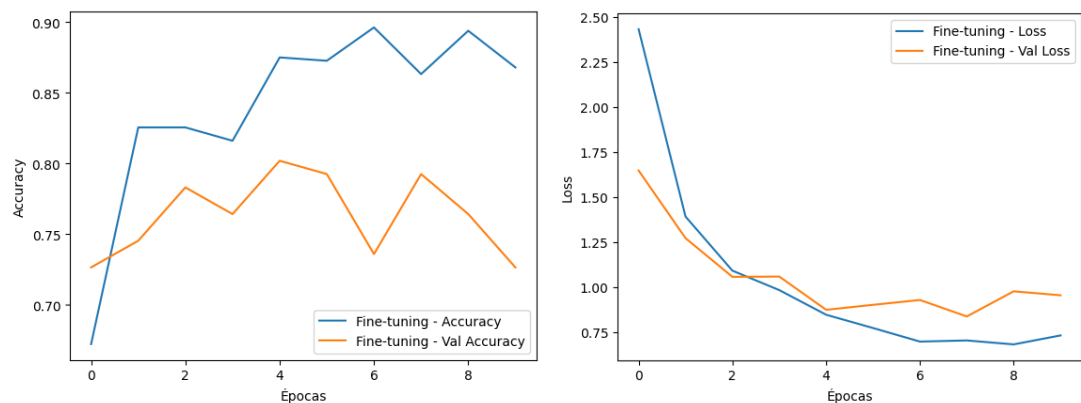


Figura 53. Curvas de aprendizaje Resnet50+LSTM

Por otro lado también se hicieron pruebas exploratorias en la detección de posturas con otros modelos como GCN por sus siglas en ingles de Graph Convolutional Network, que clasifica las posturas corporales en función de la características extraídas mediante las convolución aplicada a grafos. El modelo se entrenó con el mismo conjunto de datos de posturas detallado más adelante. La Figura 54 muestras las curvas de aprendizaje con este modelo.

Esto resultados, demuestran que el enfoque CNN+LSTM no fue el adecuado para cumplir con el fin del sistema de detección temprana de eventos delictivos, además el modelo GCN presenta una precisión de validación inferior al 60%. Por ende se optó

por otros enfoques, como la detección de objetos sospechosos y las clasificación de posturas con otro modelo más eficiente.

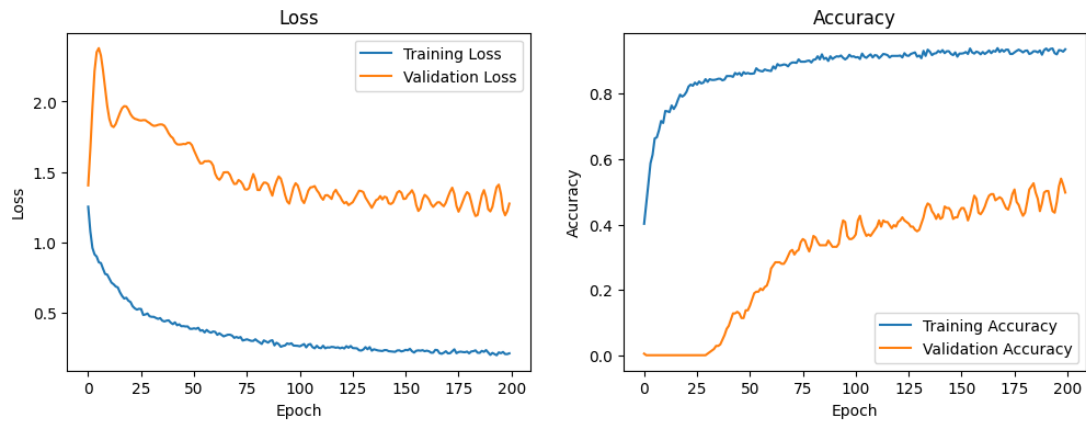


Figura 54. Curva de aprendizaje de modelo GCN para clasificación de posturas.

- ***Procesamiento de postura***

En el Capítulo I se expusieron varios modelos especializados en la estimación de postura corporal, y en base a la Tabla 9 se seleccionó a MediaPipe por su sencillez de uso, amplia documentación y por ser liviano en comparación a otros modelos. Por otro lado la detección de postura amenazante requirió de un modelo que sea capaz de clasificar los puntos clave normalizados que fueron extraídos con MediaPipe.

El Algoritmo SVM o “Suport Vector Machine” que fue explicado en el capítulo I fue seleccionado para la clasificación de posturas mediante las coordenadas normalizadas. Este algoritmo tiene un gran desempeño en el agrupamiento de distribuciones de puntos en el espacio, y las posturas corporales pueden ser representadas como distribuciones de puntos clave en un fotograma.

- ***Procesamiento de objeto***

Se precisó un modelo capaz de detectar múltiples objetos de diferentes clases en tiempo real, en este caso se comparó modelos pre-entrenados para aplicar Transfer Learning. En base a la Tabla 10 y Tabla 11 se seleccionó Yolov8 y una arquitecturas Faster RCNN con ResNet50 como backbone para el entrenamiento de la Rama Objeto debido las ventajas que presenta su arquitectura, entrenamiento y uso optimizado para procesamiento de video.

Para desarrollar la arquitectura Faster RCNN, se seleccionó PyTorch en base a la Tabla 7 debido a su flexibilidad, soporte para GPU y principalmente documentación y ejemplos disponibles, utilizados como punto de partida para la creación, entrenamiento y validación del modelo Faster RCNN. Además PyTorch cuenta con varios modelos pre entrenados, entre ellos ResNet 50 que pueden ser importados con el módulo torchvision.

- ***Rastreo de objetos***

Como algoritmo rastreo se seleccionó SORT en base a la Tabla 12, por su capacidad para realizar un seguimiento múltiple de objetos de manera rápida y eficiente. A diferencia de otros algoritmos como KCF y CSRT, que están diseñados para el seguimiento monocular y tienen ciertas limitaciones en la robustez ante oclusiones y la capacidad de adaptarse a variantes. Además, su bajo costo computacional permite su implementación en dispositivos con recursos limitados, facilitando su adopción en una variedad de aplicaciones prácticas.

d. Entorno y framework para etapa de Entrenamiento

En la etapa de entrenamiento se utilizó un entorno Cloud de desarrollo para entrenar los enfoques del modelo debido a la capacidad de procesamiento que ofrecen estas plataformas, además se utilizó un entorno de desarrollo de alto nivel especializado en Machine Learning para facilitar la programación y entrenamiento, en el Capítulo I se detalló los principales entornos Cloud.

En base a la Tabla 4 donde se comparan varias opciones de entornos Cloud disponibles se seleccionó a Kaggle como plataforma de entrenamiento, debido principalmente a la disponibilidad de recurso como GPU. Esta plataforma cuenta con una cuota semanal de uso y no diaria. Además, la facilidad de uso de los conjuntos de datos disponibles fue otro factor decisivo en la elección de este entorno.

e. Visualización

Las predicciones de cada enfoque fueron procesadas por un sistema gestor de información, desarrollado con FLASK, el cual fue seleccionado en base a la Tabla 14. El Sistema gestor de información procesa la predicciones en tiempo real, asignando

ponderaciones a las posturas y objetos detectados, mostrando visualmente las detecciones en una GUI, guardando la ocurrencia de delito en una base de Datos generando mensajes de alerta temprana por medio de Telegram.

3.2 Datos e información disponible

Para el entrenamiento de los modelos de Deep Learning se precisó de un amplio conjunto de datos que contengan eventos delictivos, teniendo en cuenta los lineamientos de privacidad establecidos en la Ley Orgánica de Vigilancia y Seguridad Privada, publicada en el Registro Oficial en febrero de 2024 se recopiló, comparó y eligió un conjunto de datos que cumpla con los requerimientos para el entrenamiento del sistema.

Esto se realizó mediante tres etapas, empezando con la recolección y descripción de conjuntos de datos referentes a eventos delictivos en sistemas de video vigilancias, buscadores de imágenes y grabaciones propias de situación dramatizadas, después se realizó una exploración de los datos recopilados y por último se generó un conjunto de datos representativo en una fase llamada verificación de calidad.

3.2.1 Recolección y descripción

Se buscó conjuntos de datos públicos en buscadores como *Google Dataset Search*, *Kaggle* y *Papers with code* filtrando los resultados por acrónimos como “HAAD” (Human Action Anomaly Detection) O “HAR” (Human Action Recognition), también por frases como “Surveillance Systems” o “Video Anomaly Detection”, obteniendo como resultado varios conjuntos de datos, los cuales fueron descritos en el Capítulo I.

Al ser la implementación en Ecuador, fue necesario enriquecer el conjunto de datos con videos de eventos delictivos ocurridos en el país, para conseguirlo se realizó una búsqueda en redes sociales como X, Facebook y Youtube, filtrando por frases como; “Robo Ecuador”, “Asalto Ecuador”, “Delincuencia Ecuador”, “Inseguridad Ecuador”, además se recopilaron varios videos referentes a robos y asaltos en los grupos de Telegram ; “Rutas del Conflicto” y “Radares Ambato”.

También se crearon situaciones dramatizadas de acciones como apuntar con un arma, amenazar con un cuchillo y comportamiento normal, las cuales fueron capturadas en video por la cámara IP utilizada en la implementación, esto para abordar la detección de posturas corporales amenazantes, dando como resultados tres videos de 15 minutos de duración correspondientes a “Arma”, “Cuchillo” y “Normal” respectivamente. La cuales también se utilizaron para el entrenamiento de detección de objetos.

Por último, también se recolecto conjuntos de imágenes relacionados a los objetos a detectar, mediante una búsqueda en Google imágenes, filtrando por frases como; “delincuentes Ecuador”, “Asaltantes Ecuador videovigilancia”, “Personas con gorras en cctv”, “Personas con mascarillas en cctv”, “Delincuentes Armados” y “Delincuentes con cuchillos”, además se filtró la calidad de las imágenes a “más de 1024x720” en las herramientas de búsqueda avanzada.

3.2.2 Exploración y selección

Después de analizar y comparar los conjuntos de datos públicos descritos en el Capítulo I, y en función a la información descrita en la Tabla. Se selecciono el conjunto de datos UCF-Crime como punto de partida, ya que recoge gran cantidad de videos de video vigilancia que contienen eventos delictivos, la Tabla 17 muestra más a detalle el tipo de anomalía y cantidad de videos respectivamente de conjunto de datos UCF-Crime.

La investigación se centra en eventos delictivos en entidades comerciales, por ende se seleccionaron únicamente los videos correspondientes a las categorías “Robbery”, “Shooting” y “Assault”, de los cuales se seleccionaron únicamente videos en los que el delito se comenta dentro de la entidad comercial, que tenga una resolución de mínimo 360p y que el evento delictivo sea captado de forma clara. Dando como resultados 15 videos de la categoría “Robbery”, 10 videos de la categoría “Shooting” y 6 videos de la categoría “Assault”.

Por otro lado, la búsqueda de videos de eventos delictivos en Ecuador, arrojo como resultado 90 videos de los cuales fueron seleccionados 75 descartando videos de baja calidad, donde no se logra distinguir el evento delictivo y videos repetidos. Además se

encontraron imágenes de sospechosos de robo, las cuales también fueron utilizadas en la detección de objetos.

Tabla 17. Descripción Anomalías UCF-Crime [18]

Etiqueta	Descripción	Numero de videos
Abuse	Comportamientos malos, crueles o violentos contra niños, ancianos, animales y mujeres.	50
Burglary	Personas (ladrones) entrando en un edificio o casa con la intención de cometer un robo. No incluye el uso de la fuerza contra las personas.	100
Robbery	Ladrones sustrayendo dinero ilícitamente por la fuerza o mediante amenaza de fuerza. Estos vídeos no incluyen tiroteos.	150
Stealing	Personas tomando bienes o dinero sin permiso. No se incluyen los hurtos en tiendas.	100
Shooting	Acto de disparar a alguien con un arma.	50
Shoplifting	Personas robando artículos de una tienda mientras se hacen pasar por compradores.	50
Assault	Ataque físico repentino o violento a alguien. Ten en cuenta que en estos vídeos la persona agredida no se defiende.	50
Fighting	Vídeos en los que dos o más personas se agreden mutuamente.	50
Arson	Personas prendiendo fuego deliberadamente a una propiedad	50
Explosion	Explosión destructiva de algo. Este evento no incluye vídeos en los que una persona provoca intencionadamente un incendio o una explosión.	50
Arrest	Vídeos que muestran a la policía arrestando a personas.	50
Road Accident	Accidentes de tráfico en los que se ven implicados vehículos, peatones o ciclistas.	150
Vandalism	Acciones que implican la destrucción deliberada o daños a la propiedad pública o privada. El término incluye daños a la propiedad, como pintadas y desfiguraciones dirigidas a cualquier propiedad sin permiso del propietario.	50
Normal Event	Vídeos en los que no se ha producido ningún delito. Estos vídeos incluyen escenas en interiores (como un centro comercial) y exteriores, así como escenas diurnas y nocturnas.	950

Por último, la búsqueda realizada en Google imágenes dio como resultado alrededor de 800 imágenes de las cuales fueron seleccionadas 450, mediante criterios como; calidad de imagen, visibilidad del objeto, y balance de categoría. Respecto a esto último se procuró balancear las categorías de objetos para generalizar la detección de objetos de mejor manera.

3.2.3 Verificación de calidad

Una vez recopilado los datos, se realizó un último filtrado visual a cada uno de los videos e imágenes, revisando si los objetos son visibles claramente, si los puntos de vista son los adecuados. Para el caso de los videos utilizados en la obtención de posturas, se recortaron los primeros y los últimos segundos de cada video para garantizar que la estimación de postura sea la correcta y no capture instantes irrelevantes dentro de la escena.

3.3 Conjunto de datos

Se crearon dos conjuntos de datos, el primero para entrenar el modelo detector de objetos sospechosos, mediante imágenes etiquetadas de objetos sospechosos comunes en eventos delictivos, y el segundo con las coordenadas normalizadas de posturas corporales amenazantes, a continuación se detalla este proceso

3.3.1 Detección de objetos sospechosos

a. Limpieza de datos

Antes de crear el conjunto de datos para el entrenamiento y validación de los modelos, fue necesario hacer una limpieza de los datos, tanto para el entrenamiento del modelo detector de objetos sospechosos, como en detección de posturas amenazantes. Esto realizo, al cargar los fotogramas e imágenes en el software utilizado para etiquetado, eliminando los fotogramas e imágenes donde se el objeto se encuentre difuso.

b. Etiquetado

Se utilizó la plataforma Roboflow para el etiquetado de imágenes, primero se subieron las imágenes resultantes de la etapa de recolección, las cuales, después del filtrado y verificación de calidad, quedaron en un total de 400 imágenes. Para el caso de los videos, Roboflow permite subir videos y extraer fotogramas de estos, estableciendo cuantos fotogramas se extraen por cada segundo de video, se estableció que este parámetro sea un fotograma por segundo, para evitar que los fotogramas sea muy parecidos entre sí. La

Figura 55 ejemplifica este proceso.

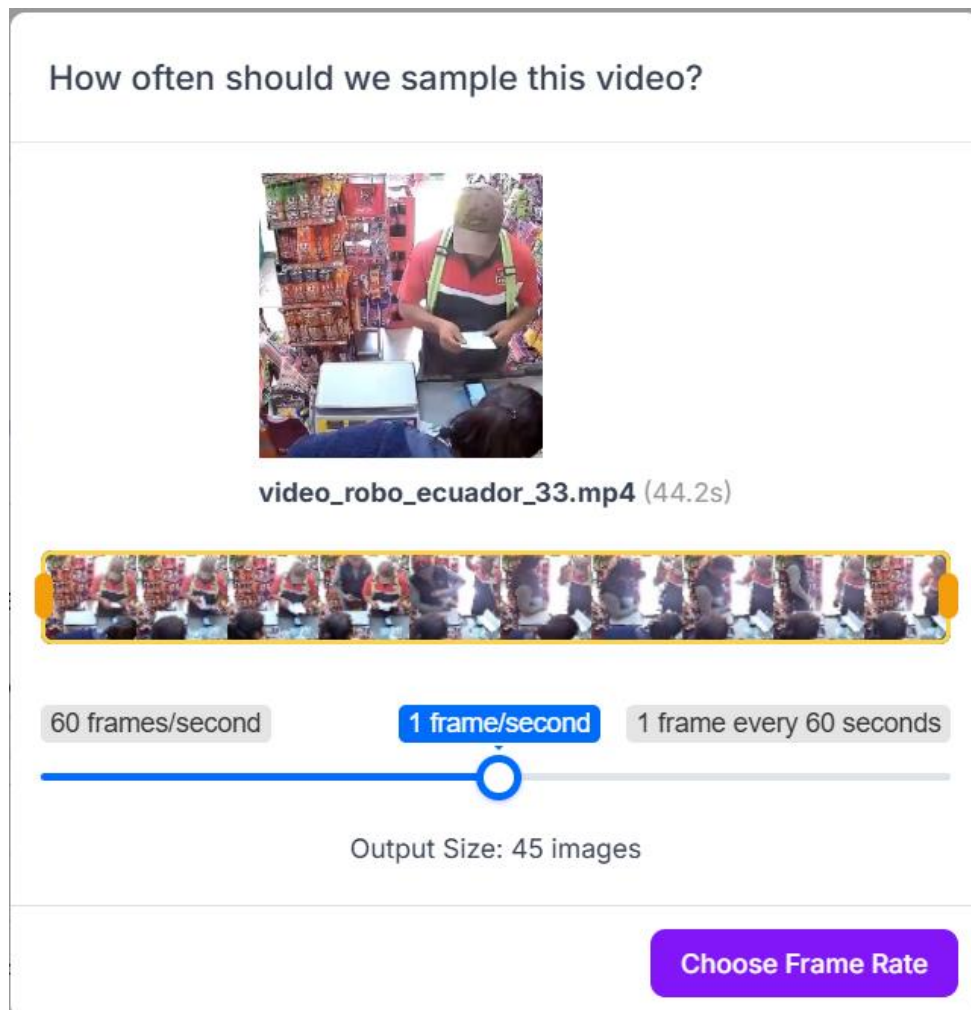


Figura 55. Proceso de extracción de fotogramas de videos para la creación del conjunto de datos de entrenamiento de la Rama Objeto.

Después de haber cargado las imágenes y fotogramas, se procedió a etiquetar los objetos en 5 categorías; “arma”, “cuchillo”, “gorra”, “mascarilla” y “pasamontaña”, para el caso de las armas y cuchillos se etiqueto cada objeto procurando incluir el contexto, es decir la mano que sostiene el objeto, esto para mejorar la generalización del modelo detector. Para el caso de mascarillas, gorras y pasamontañas, los objetos se etiqueto tradicionalmente, es decir conteniendo solo al objeto. La Figura 56 ejemplifica este proceso.

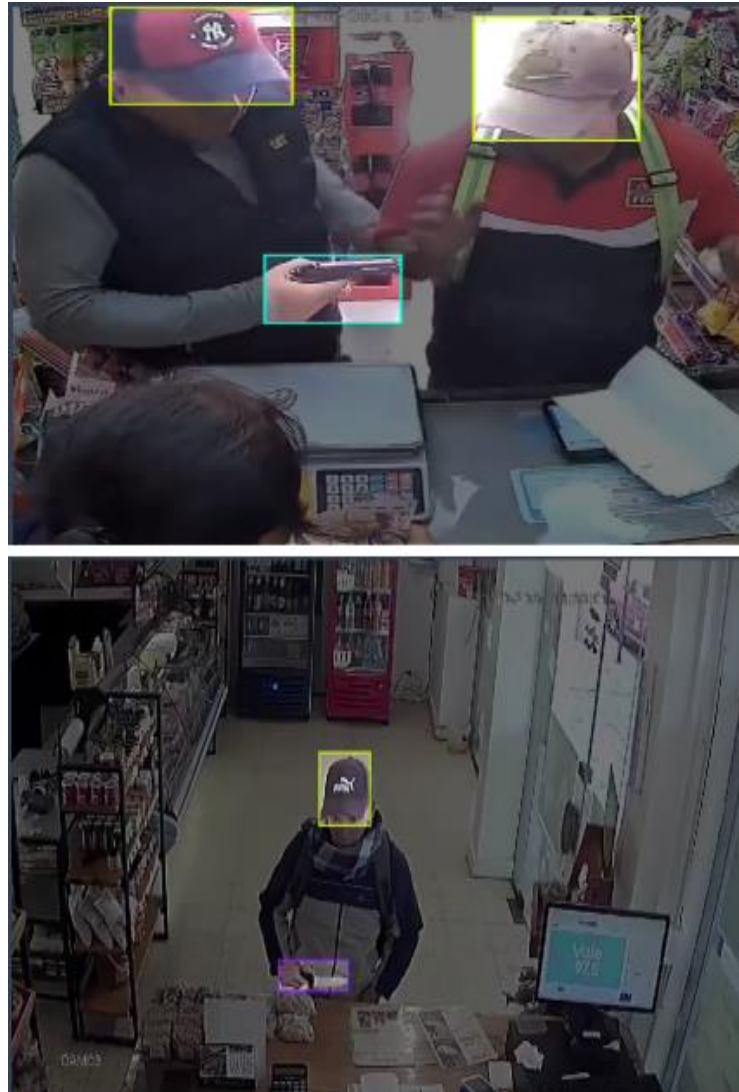


Figura 56. Etiquetado de Imágenes en plataforma RoboFlow.

Después de etiquetar se obtuvo como resultado un conjunto de datos con 1587 imágenes para entrenar el modelo de detección de objetos sospechosos. Roboflow permitió generar varias versiones del conjunto de datos, en diferentes formatos tales como; COCO, Tensor Flow TFRecords o en este caso YOLO.

c. Estructura

Después de limpiar y etiquetar las imágenes, Roboflow permite generar el conjunto de datos aplicando varios parámetros como pre procesamiento, redimensionamiento, filtros entre otros. Por ende se aplicó Auto Orientación, redimensionamiento de 640 x 640, para el Aumento de datos se utilizó transformaciones horizontales y verticales rotaciones entre de 15 grados, ruido y filtros blanco y negro.

La división fue 80% para entrenamiento, 10% validación y 10% para testeo, se exportaron dos versiones del conjunto de datos. La primera en el formato apropiado para entrenar modelos Yolo, la Figura 57 muestra es estructura de esta versión, en la que se tienen los directorios “train”, “valid” y “test”, dentro de estas están directorios con las imágenes en formato jpg y sus anotaciones en formato txt, y por último el archivo “data.yaml” dentro de este se encuentran las rutas de las imágenes y anotaciones, así como las especificaciones de clases de objetos.

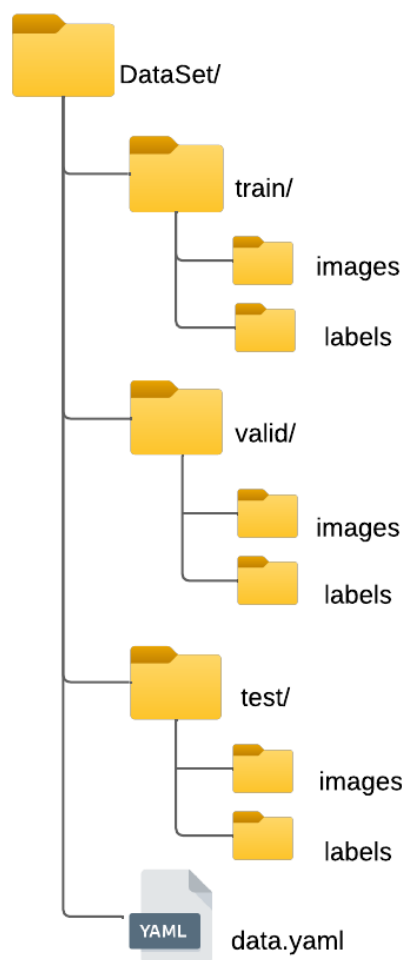


Figura 57. Estructura del conjunto de datos en formato Yolo.

La segunda versión fue exportada en versión COCO, formato apropiado para entrenar modelos Faster RCNN, la Figura 58 muestra la estructura de este formato del conjunto de datos, consta de tres carpetas, “train”, “valid” y “test”, cada una de estas con las imágenes y las etiquetas de las mismas un archivo en formato json.

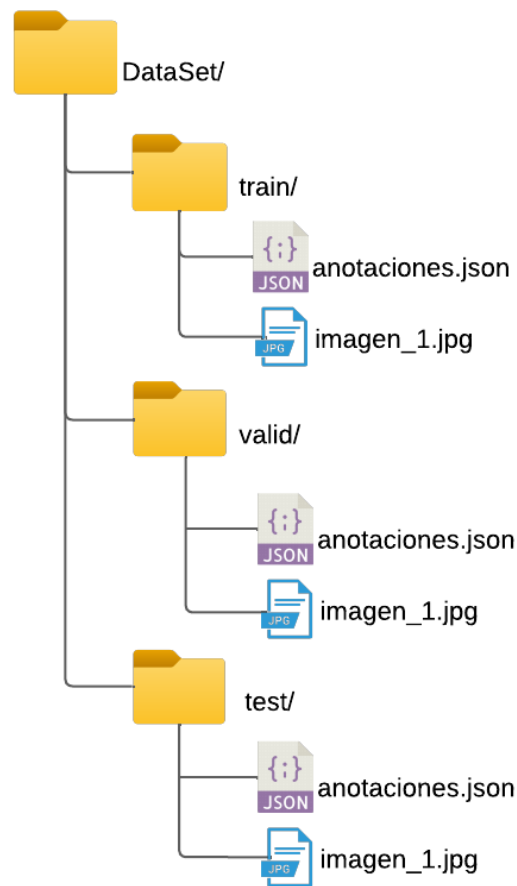


Figura 58. Estructura del conjunto de datos en formato COCO.

3.3.2 Detección de posturas amenazantes

Se utilizaron tres videos de 15 minutos de duración en los cuales se dramatizo comportamiento normal, amenaza con arma y amenaza con cuchillo respectivamente, se eliminó fragmentos irrelevantes de cada video. Después del preprocesamiento, se extrajeron posturas corporales de cada video, dos detecciones por cada segundo.

Se utilizo Mediapipe como extractor de posturas corporales, debido a la sencillas de uso, la postura objetivo fue el siguiente conjunto de puntos; 0 “nose”, 11, “left_shoulder”, 12 “right_shoulder”, 13 “left_elbow”, 14 “right_elbow”, 15 “left_wrist”, 16 “right_wrist”, 23 “left_hip”, 24 “right_hip”. La Figura 59 ejemplifica visualmente el proceso de extracción de postura



Figura 59. Ejemplificación visual del proceso de extracción de posturas de entrenamiento

El dataframe se genera mediante las coordenadas normalizadas respecto a un ROI o región de interés definida por un recuadro que contenga a todos los puntos clave más un rango de expansión de 50% en alto y ancho, esto para definir a su vez el área para detección de objetos. Las coordenadas se guardan en un archivo .csv el cual se usó en el entrenamiento del modelo detector de posturas. En el Anexo E se muestra el diagrama de flujo del proceso de extracción de posturas.

Después de la extracción de las posturas corporales y creación del dataframe con las coordenadas normalizadas respecto al área de interés, se obtuvo como resultado un conjunto de datos con 1055 registros. El dataframe contiene el siguiente encabezado, la primera columna corresponde a la etiqueta; 0 para Normal, 1 amenaza con cuchillo y 2 para amenaza con pistola, las siguientes columnas con las coordenadas tanto en x como y para cada punto clave.

3.4 Modelado

3.4.1 Entrenamiento y validación

Se realizó el entrenamiento y validación de los modelos de detección de objetos sospechosos y posturas amenazantes, se realizaron varios entrenamientos con diferentes configuraciones de parámetros de entrenamiento como “batch size”, “epochs” y “learning rate”, para el caso de los modelos de detección de objetos y “kernel” para el caso de SVM. Además se evaluaron los modelos con métricas de validación explicadas en el Capítulo I. La Figura 60 muestra el esquema general de entrenamiento del sistema.

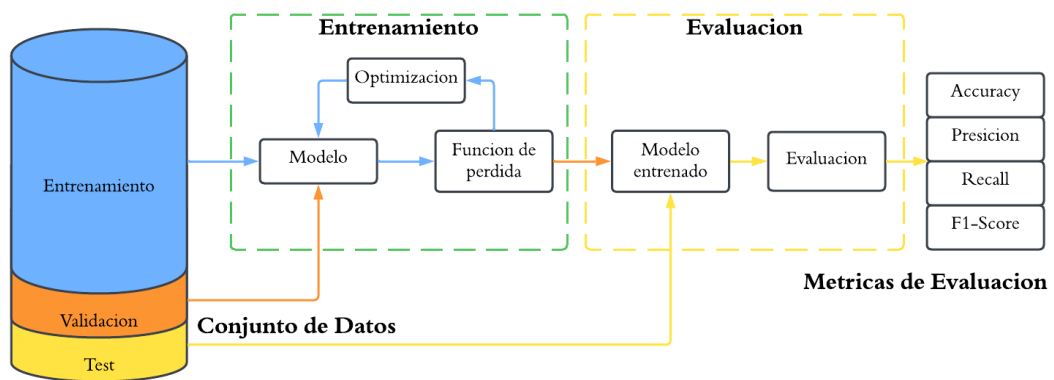


Figura 60. Esquema de entrenamiento del sistema.

El esquema de entrenamiento muestra el proceso que realiza el modelo para ajustar sus pesos mediante la retroalimentación de las salidas que produce en cada iteración de entrenamiento, por otro lado una vez que el modelo se ha entrenado, este es evaluado con una parte de conjunto de datos no vistos en el entrenamiento. Las métricas calculadas sirvieron como pauta para la selección del modelo más óptimo.

a. Modelo detección de objetos sospechosos

Para esta etapa se realizaron varios entrenamientos con diferentes tamaños de yolov8 y una arquitectura Faster RCNN ResNet 50 como backbone, con el mismo conjunto de datos, esto con el fin de seleccionar el modelo más óptimo para el sistema, tomando en cuenta métricas de validación y complejidad del modelo, buscado el modelo más

preciso en la detección que a su vez sea el ligero y versátil para las capacidades de la unidad de procesamiento.

Para el caso de Yolo se utilizaron los tamaños “nano”, “small” y “medium” para el entrenamiento, esto con el fin de tener un modelo ligero para no sobrecargar la capacidad del módulo Nvidia Jetson nano. Por otro lado para Faster RCNN se utilizó ResNet50 como backbone del enfoque, al igual que Yolo se establecieron parámetros de entrenamiento para optimizar los recursos de entrenamiento.

La Tabla 18 resume los características de los tamaños utilizados en el entrenamiento.

Tabla 18. Características de tamaños de modelos. [65]

Tamaño	Parámetros	Gradientes	GFLOPs
Yolov8n	3 011 823	3 011 807	8.2
Yolov8s	11 137 535	11 137 519	28.7
Yolov8m	25 859 215	25 859 199	79.1
ResNet50	25 636 712	25 583 592	4.09

Se entrenaron los tamaños descritos anteriormente con ligeras diferencias en sus parámetros de entrenamiento, debido a las capacidades del entorno de entrenamiento. La Tabla 19 se resumen los parámetros utilizados con cada tamaño de modelo utilizado, cabe destacar que si no se define parámetros entrenamiento específicos se utilizan los valores por defecto.

Tabla 19. Parámetros de entrenamiento para tamaños de modelos.

Tamaño	epochs	batch	imgsz	Plots
Yolov8n	80/60/40	64	640	True
Yolov8s	80/60/40	64	640	True
Yolov8m	80/60/40	32	640	True
ResNet50	20/10/5	16	-	-

Después de cada entrenamiento se revisaron tanto las gráficas de resultados de entrenamiento en las que se visualiza las curvas de aprendizaje, matrices de confusión y métricas de validación para seleccionar la mejor versión del entrenamiento. La Tabla 20 resume los resultados de cada entrenamiento como métricas de validación, numero de épocas y tiempo de duración.

Tabla 20. Resultados de entrenamiento de modelos para detección de objetos.

Modelo	Épocas	Métricas de Validación				Duración (Horas)
		Presicion	Recall	mAP50	mAP50-95	
Yolov8n	80	0.876	0.806	0.879	0.660	0.399
	60	0.865	0.758	0.872	0.663	0.305
	40	0.859	0.738	0.858	0.652	0.212
Yolov8s	80	0.894	0.836	0.891	0.685	0.931
	60	0.888	0.806	0.894	0.683	0.649
	40	0.901	0.813	0.889	0.678	0.349
Yolov8m	80	0.848	0.832	0.888	0.690	1.429
	60	0.887	0.814	0.886	0.678	1.079
	40	0.893	0.806	0.895	0.690	0.727
ResNet50	20	0.861	0.636	0.860	0.563	1.35
	10	0.826	0.623	0.813	0.567	0.97
	5	0.789	0.639	0.771	0.557	0.48

Después de revisar los resultados del entrenamiento se eligió la versión yolov8m entrenada con 40 épocas y mAP50 de 0.895, esto significa que el modelo es eficaz en la detección de objetos de entrenamiento. La Figura 61 muestra los resultados del entrenamiento del modelo seleccionado, aquí se muestra el rendimiento delo modelo a lo largo del proceso de entrenamiento.

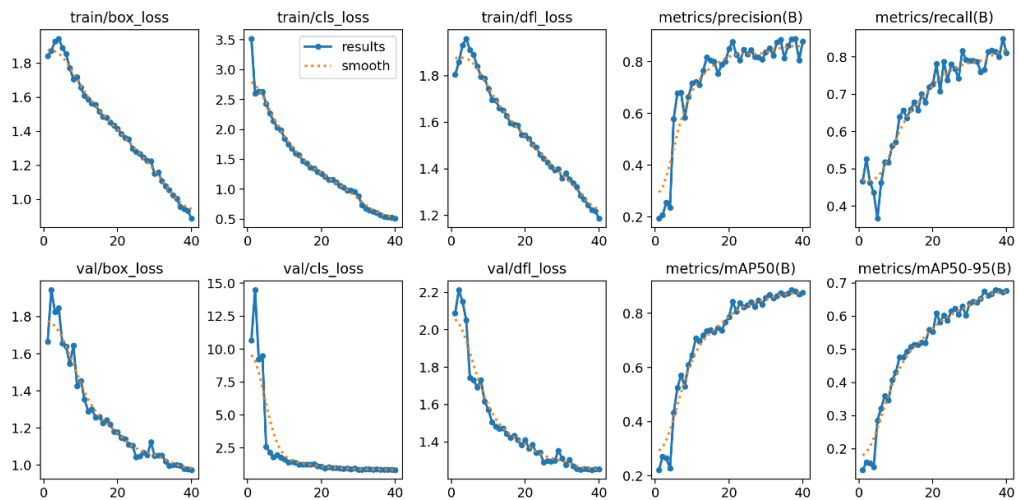


Figura 61. Resultados de entrenamiento de yolov8m con 40 épocas.

La perdida la similitud entre las tendencias de las pérdidas de caja y de clases muestran que el modelo tiene buena capacidad de generalización, por otro lado las métricas de validación tienen una tendencia ascendente. Al final del entrenamiento se guardan el mejor modelo y el modelo final, la métricas de validación se realizaron evaluando el mejor modelo “best.pt”. La Figura 62 muestra las matriz de confusión del modelo.

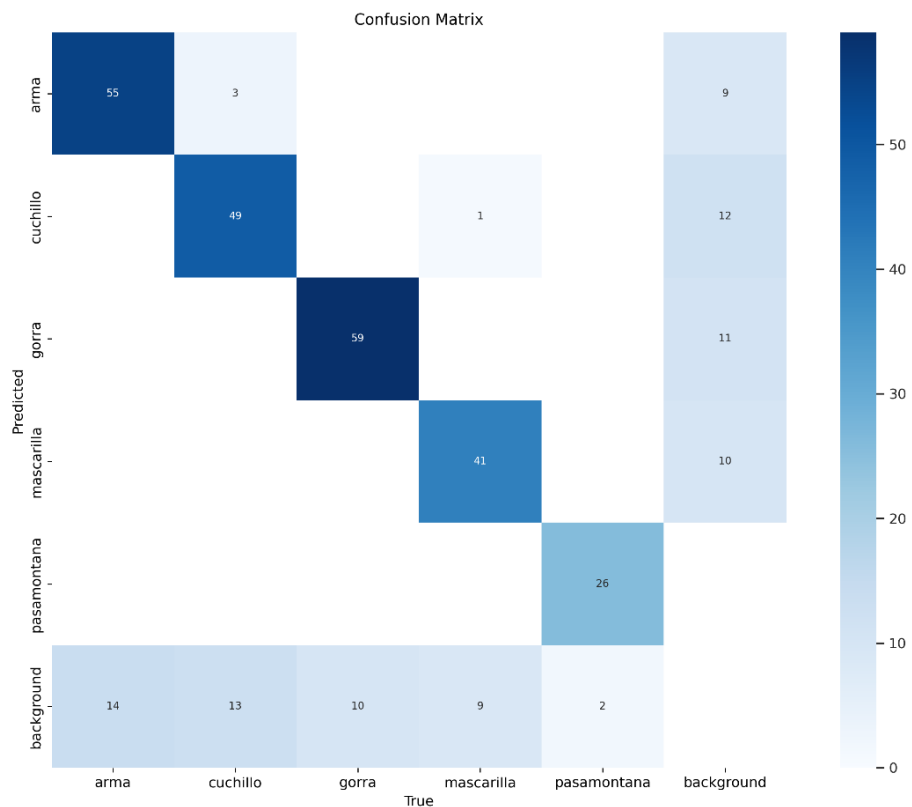


Figura 62. Matriz de confusión de validación del modelo yolov8m

La matriz de confusión da mucha información sobre el rendimiento del modelo entrenado, a priori muestra un gran desempeño en detección de armas y cuchillos, por otro lado el desempeño disminuye en la detección de mascarillas y pasamontañas, en conclusión el modelo presenta poca confusión entre clases, ya que la mayor concentración de detecciones se encuentran en la diagonal principal de la matriz, debido a la complejidad de forma, tamaño y color de los objetos, existe confusión con la clase background.

La Figura 63 muestra la precisión del modelo frente a diferentes valores de confianza, todas las clases tienen una curva de precisión y tienden a estabilizarse cuando la confianza aumenta, en general para todas las clases la precisión llega a 1 cuando la confianza es 0.945. Esto muestra que cuando la confianza de detección es alta el rendimiento del modelo es excelente.

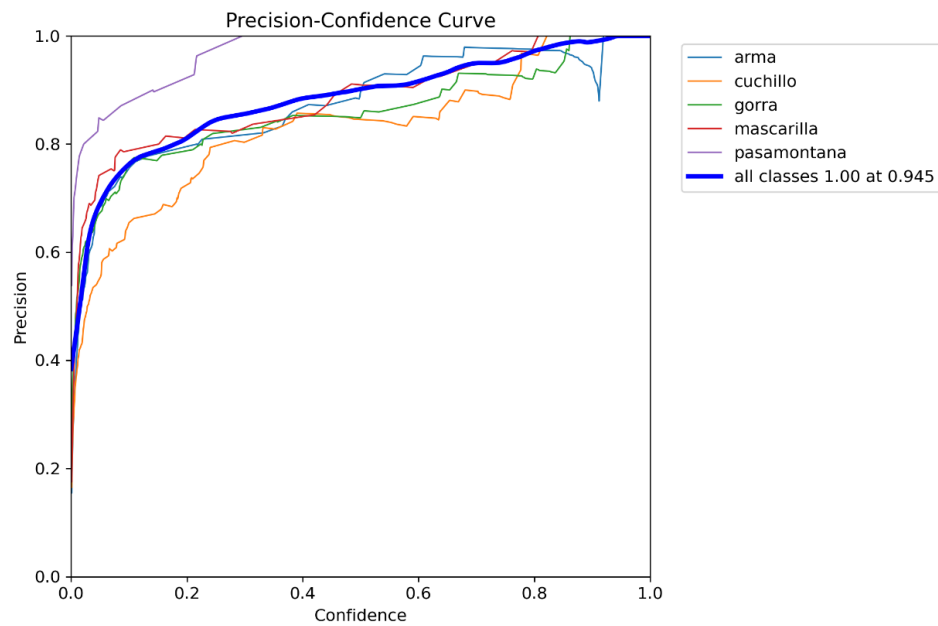


Figura 63. Curva de Precisión del modelo entrenado

La Figura 64 muestra la relación entre el Recall o la tasa de verdaderos positivos y distintos valores de confianza, la gráfica muestra que el modelo logra un recall de 0.93 con 0.0 de valor de confianza, evidenciando que el 93% de las instancias clasificadas como positivas son verdaderas. El desempeño es mejor con las clases arma y cuchillo que el resto de clases.

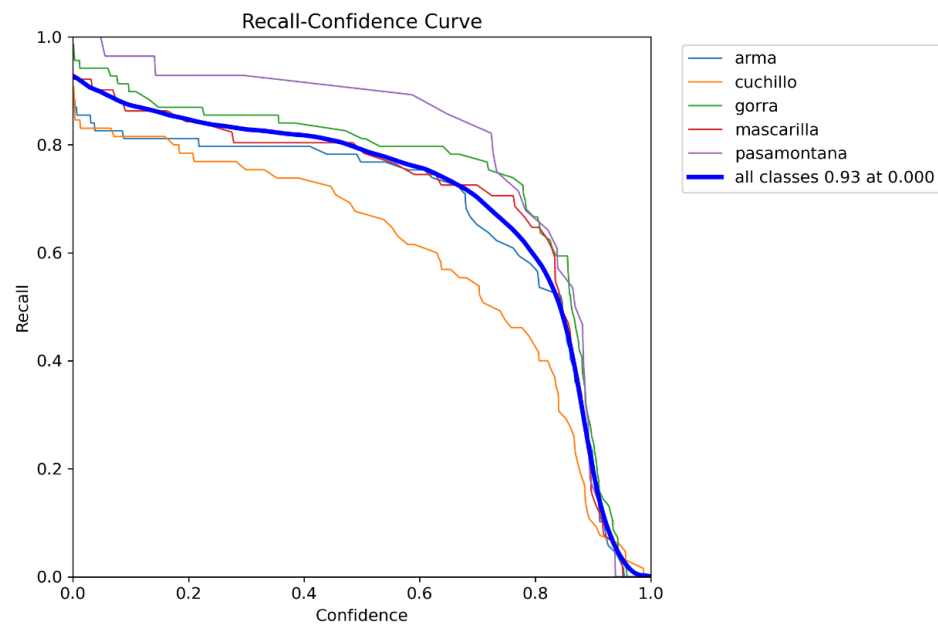


Figura 64. Curva de Recall de modelo entrenado.

La Figura 65 muestra la relación entre f1-score frente a diferentes valores de confianza, esta métrica toma en cuenta la precisión y recall. Las curva que toma en cuenta todas las clases de objetos se estabiliza en un valor de 0.85 con un valor de confianza de 0.406. Este valor es importante ya que sirvió para establecer el valor de confianza en el código de procesamiento del backend del proyecto, descartando las detecciones con valores de confianza inferiores.

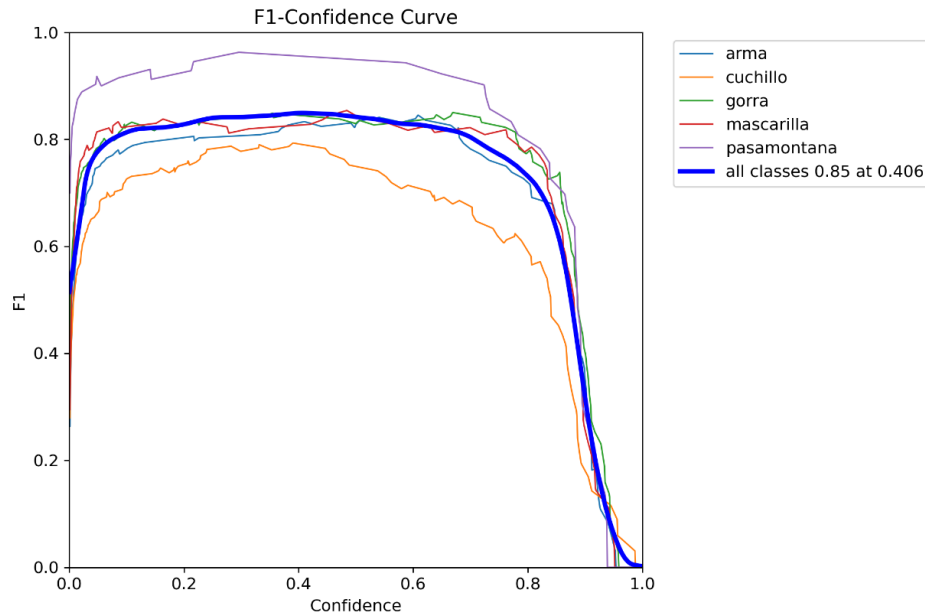


Figura 65. Curva de F1-Score de modelo entrenado.

b. Modelo detección de posturas amenazantes

Se entrenó un modelo SVM para clasificar posturas corporales en tres categorías; “Normal”, “Cuchillo”, “Pistola”. El modelo recibe las coordenadas de cada punto clave extraído por medio de MediaPipe. El entrenamiento se realizó estableciendo diferentes valores de “gamma” y “C” para establecer el mejor modelo. La Tabla 21 muestra un resumen de los valores de los parámetros de entrenamiento que se evaluaron.

Tabla 21. Parámetros de entrenamiento SVM para clasificación de postura

Parámetro	Valor
Kernel	Linear
	Poly
	Rbf
C	0.001, 0.01, 0.1, 1, 10, 100
Gamma	0.001, 0.01, 0.1, 1, 10, 100

Al final fueron 108 iteraciones con todas las posibles combinaciones de los parámetros de entrenamiento dentro de los rangos establecidos. La Figura 66 muestra los resultados obtenidos, los mejores parámetros y las puntuaciones del mejor modelo con los datos de validación y Test.

```
Resultados de la Validación:
  kernel      C      gamma      score
0  linear  0.001  0.001  0.383721
1  linear  0.010  0.001  0.703488
2  linear  0.100  0.001  0.796512
3  linear  1.000  0.001  0.854651
4  linear 10.000  0.001  0.848837
..      ...      ...      ...
103  rbf    0.010 100.000  0.383721
104  rbf    0.100 100.000  0.383721
105  rbf    1.000 100.000  0.395349
106  rbf   10.000 100.000  0.412791
107  rbf  100.000 100.000  0.412791

[108 rows x 4 columns]

Mejor score en set de Validación: 0.92
Mejores Parámetros: {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
Test set score con mejores parámetros: 0.88
```

Figura 66. Resultados de entrenamiento SVM con diferentes parámetros.

La Figura 67 muestra el mapa de calor de los resultados del entrenamiento con los diferentes valores de gamma y C. Se puede observar que los mejores resultados se encuentra con los valores 0.1 y 1.0 del parámetro gamma. Por otro lado un valor de 0.001 del parámetro C tiene un bajo rendimiento con valores de gamma entre 0.001 y 0.1, en resumen los valores de $C = 100$, $\text{gamma} = 0.1$ y el kernel = 'rbf' logran establecer el mejor hiperplano para la clasificación de posturas.

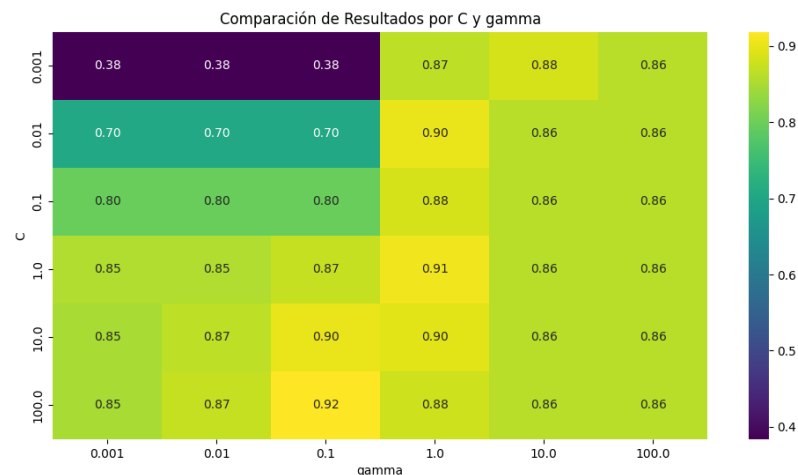


Figura 67. Mapa de calor de entrenamientos con diferentes valores de gamma y C.

La Figura 68 muestra la matriz de confusión y matriz de confusión normalizada del del mejor modelo evaluado con un conjunto de datos para Test. Se puede observar claramente un gran desempeño en la clasificación de postura “Normal” con la etiqueta 0 y “Cuchillo” con la etiqueta 1, pero para posturas de “Pistola” con la etiqueta 2 existe una ligera confusión con posturas normales, Por ende la importancia de asocial la postura corporal con la detección de objetos sospechosos.

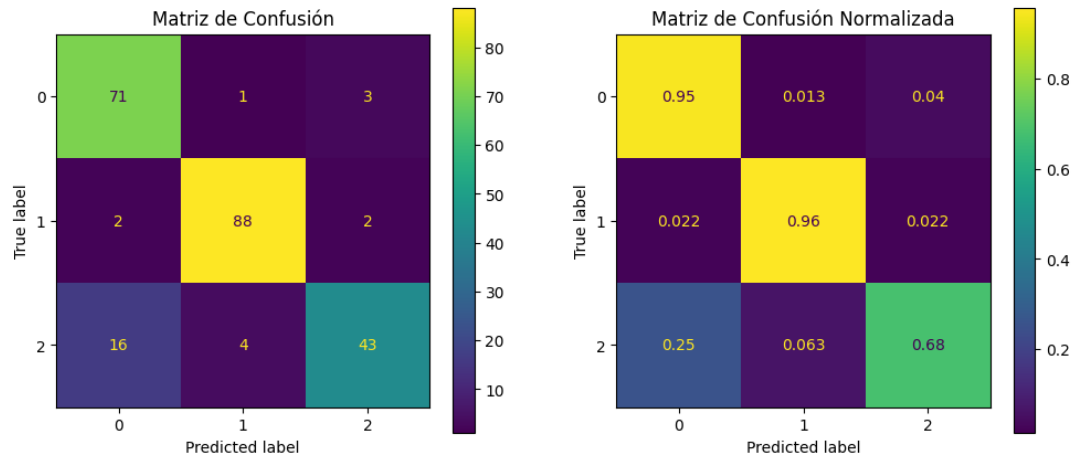


Figura 68. Matrices de confusión de modelo SVM para clasificación de posturas

Los recursos que se utilizaron en el entrenamiento corresponde al costo computacional, se utilizó una tarjeta gráfica NVIDIA Tesla P100 PCIE, la cual está basada en la arquitectura Pascal, que ofrece un aumento significativo en el rendimiento y la eficiencia energética en comparación con generaciones anteriores de GPU. Además es compatible con CUDA 12.6. La Tabla 22 resume el hardware utilizado y el tiempo de entrenamiento.

Tabla 22. Costo computacional de entrenamiento.

Característica	Detalles
GPU	Tesla P100-PCIE-16 GB
Memoria Ram	29 GB
Versión del controlador	560.35.03
Disco	57.6 GB
Versión de CUDA	12.6
Bus ID	00000000:00:04.0
ECC Volátil No Corregido	No
Temperatura	34°C
Modo de rendimiento	P0
Uso de energía	26W / 250W
Tiempo de Entrenamiento	8.80 Horas

3.4.2 Sistema gestor de información

a. Configuración inicial

La tarjeta Nvidia Jetson Nano cuenta con Ubuntu 20.04 como sistema operativo, se realizó la configuración inicial de FLASK siguiendo este proceso. Primero se actualizó la lista de paquetes instalados y después se procedió a instalar Python mediante el gestor de paquetes pip. La Figura 69 muestra las versiones instaladas para este proyecto.

```
manuman1505@Pruebas:~$ python3 --version
Python 3.12.3
manuman1505@Pruebas:~$ pip3 --version
pip 24.0 from /usr/lib/python3/dist-packages/pip (python 3.12)
manuman1505@Pruebas:~$
```

Figura 69. Versión de Python utilizado.

Después de esto se creó un entorno virtual para evitar conflictos de dependencias, además es una buena práctica de desarrollo. El entorno virtual se llamó “flask_env”. Dentro de este entorno se instalaron todos los paquetes y dependencias necesarias para correr este proyecto como FLASK, OpenCV, MediaPipe, numpy, joblib, ultralytics, sqlite3 y request. También fue necesario instalar dependencias como torch, torchvision como dependencias secundarias. La Figura 70 muestra los paquetes instalados en el entorno virtual.

```
(flask_env) manuman1505@Pruebas:~$ pip list
Package Version
-----
absl-py 2.1.0
attrs 24.3.0
blinker 1.9.0
certifi 2024.12.14
cffi 1.17.1
charset-normalizer 3.4.1
click 8.1.8
click 1.3.1
contourpy 0.12.1
cycler 3.16.1
filelock 1.4.5
filterpy 3.1.0
Flask 24.12.23
fonttools 4.55.3
fsspec 2024.12.0
idna 3.10
itsdangerous 2.2.0
jax 0.4.38
jaxlib 0.4.38
Jinja2 3.1.5
joblib 1.4.2
kiwisolver 1.4.8
MarkupSafe 3.0.2
matplotlib 3.10.0
mediapipe 0.10.20
ml_dtypes 0.5.0
mpmath 1.3.0
networkx 3.4.2
numpy 1.26.4
nvidia-cublas-cu12 12.4.5.8
nvidia-cuda-cupti-cu12 12.4.127
nvidia-cuda-nvrtc-cu12 12.4.127
nvidia-cuda-runtime-cu12 12.4.127
nvidia-nccl-cu12 2.21.5
nvidia-nvjitlink-cu12 12.4.127
nvidia-nvtx-cu12 12.4.127
opencv-contrib-python 4.10.0.84
opencv-python 4.10.0.84
opt_einsum 3.4.0
packaging 24.2
pandas 2.2.3
pillow 11.0.0
pip 24.0
protobuf 4.25.5
psutil 6.1.1
py-cpuinfo 9.0.0
pyparsing 3.2.2
python-dateutil 2.9.0.post0
pytz 2024.2
PyYAML 6.0.2
requests 2.32.3
scikit-learn 1.6.0
scipy 1.14.1
seaborn 0.13.2
sentencepiece 0.2.0
setuptools 75.6.0
six 1.17.0
sounddevice 0.5.1
sympy 1.13.1
threadpoolctl 3.5.0
torch 2.5.1
torchaudio 2.5.1
torchvision 0.20.1
tqdm 4.67.1
triton 3.1.0
typing_extensions 4.12.2
tzdata 2024.2
ultralytics 8.3.55
ultralytics-thop 2.0.13
urllib3 2.3.0
Werkzeug 3.1.3
(flask_env) manuman1505@Pruebas:~$
```

Figura 70. Paquetes instalados dentro del entorno virtual.

Además se configuró el puerto 5000 ya que es por el cual se expone frontend del proyecto, asegurándonos que este permitido en el firewall. La Figura 71 muestra que el tráfico está permitido por el puerto 5000. Por último se debe exportar el proyecto en este caso se hizo mediante SCP.

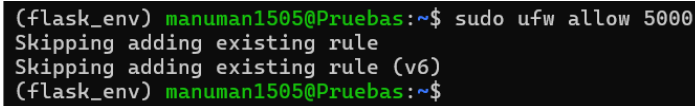
A terminal window with a black background and green text. The prompt is '(flask_env) manuman1505@Pruebas:~\$'. The command 'sudo ufw allow 5000' has been entered. The output shows 'Skipping adding existing rule' twice, once for the default rule and once for rule (v6). The prompt returns to '(flask_env) manuman1505@Pruebas:~\$'.

Figura 71. Configuración de firewall.

b. Arquitectura del servidor

Se desarrollo una arquitectura simple en la que el backend es gestionado por el archivo “app.py”, también se tiene cuatro directorios; “ModelosDeteccion” con los modelos entrenados para la detección de objetos sospechosos, “ModelosPostura” con el modelo para detección de posturas amenazantes, “templates” con los archivos frontend y “Static” que contiene los archivos CSS, JavaScript para la funcionalidad de la interfaz de usuario, además de un subdirectorio “Capturas” con fotogramas en los que sucede un evento sospechoso o delictivo.

La base de datos se encuentra en el archivo “detecciones.db”, también se tiene el archivo “sort.py”, el cual sirve para el tracking de las detecciones. Para la gestión de frontend se tiene tres archivos; “index.html” con la página principal de la interfaz gráfica, “historial.html” para gestionar la visualización histórica de las detecciones y “login.html” para gestionar la autenticación del administrador del sistema. En el Anexo E se esquematiza esta arquitectura.

c. Backend

Como se explicó en la arquitectura, el backend es gestionado por el archivo “app.py”, este archivo gestiona el funcionamiento general del sistema, primero valida la autenticación del usuario, si es correcta redirige al “index.html”, dentro del cual se muestran tanto el video analizado en tiempo real como los indicadores de sospecha, la barra de navegación permite visualizar también las detección históricas mediante una consulta a la base de datos. En el Anexo E se muestran los códigos más importantes.

El proceso empieza inicializando los modelos de detección de posturas y objetos, variables de estado y etiquetas, además del objeto “tracker” que realiza el seguimiento a la persona detectada. Después se verifica una variable de proceso la cual inicializa la captura de video, esta variable es modificada desde el frontend, una vez verificada la lectura de fotograma se extrae la postura corporal clasificando la postura y estableciendo un valor en la variable pose_score de 0 si es Normal, 70 si es amenaza con cuchillo y 100 si es amenaza con arma.

La región de interés se calcula en función de la posición de los puntos clave especificados, añadiendo un margen de 50% en el ancho y un 75% en el alto, esto para abarcar los posibles objetos detectados. Esta región es rastreada por medio del algoritmo SORT, manteniendo el mismo identificador de detección en su aparición en cada fotograma.

Únicamente esta región es procesada por el modelo detector de objetos, estableciendo un valor en la variable object_score con la suma de los valores establecidos para cada objeto, 25 para gorras y mascarillas respectivamente, 80 para cuchillos y pasamontañas respectivamente y 100 para armas de fuego. El valor de la variable se mapea en un rango de 0 a 100 para evitar valores superiores. La región de interés es seguida por medio del algoritmo de Sort y es dibujada con una etiqueta.

La etiqueta es una variable de estado que puede ser “Normal” si la variable object_score es menor a 25, “Sospechoso” si object_score está entre 25 y 80 y la variable pose_score es menor a 70 y por último “Delito inminente” cuando object_score es mayor que 80 y pose_score es mayor que 70, ya que esta variable toma esos valores cuando es una postura amenazante. La etiqueta, los objetos detectados y una captura del fotograma son enviados por el Bot de Telegram únicamente cuando object_score es mayor a 50, esto cada 30 fotogramas, por último se guarda la información enviada como alerta en la base de datos. En la Figura 72 resume el flujo de procesamiento del sistema.

- **Base de Datos**

La base de datos fue gestionada con SQLite. Se creó una tabla llamada “detenciones” con los campos; “ID” como clave primaria, “Fecha”, con la hora y fecha de la

detección, “Objetos_Detectados” para la lista de objetos detectados, “Ruta_Captura” con la dirección de la captura de fotograma asociada a la detección y “Ruta_label” para guardar las cajas delimitadoras y clase asociadas a cada detección en formato txt, esto con el fin de crear alimentar el conjunto de datos para una futura actualización del modelo. La Figura 73 se esquematiza el diagrama entidad-relación, donde se muestra la relación muchos a mucho entre los usuarios finales y la tabla “detecciones”.

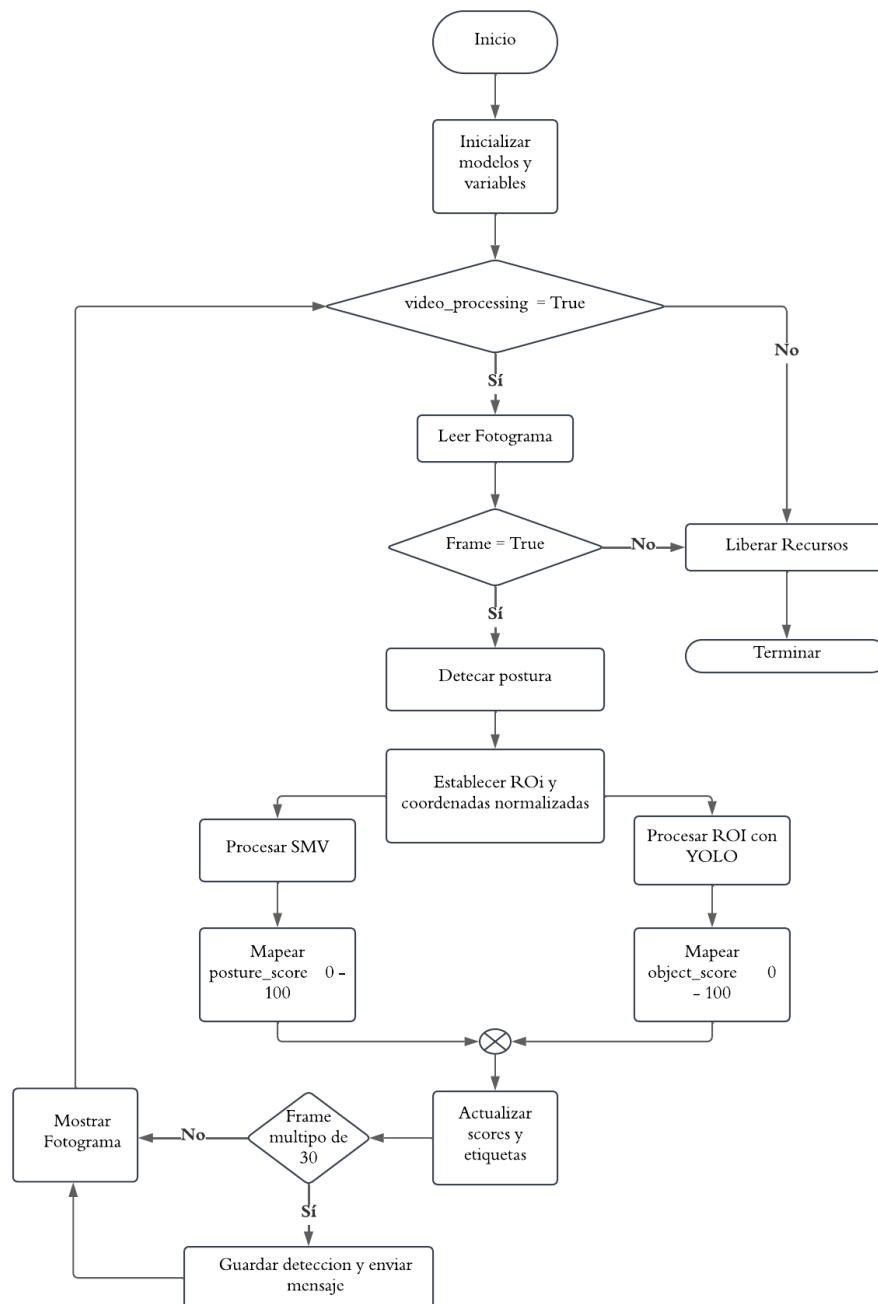


Figura 72. Diagrama de flujo del proceso de análisis de video

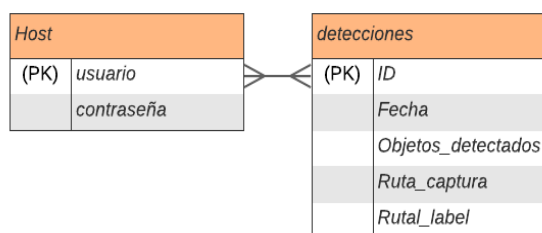


Figura 73. Diagrama entidad-relación.

Las detecciones se guardan en un hilo de procesamiento paralelo, gracias a la librería “Thread” que permite correr varias ramas de procesamiento al mismo tiempo, esto para optimizar el proceso. Para esto, primero se importó la librería, después se inicializo el hilo de procesamiento.

Durante el procesamiento de la escena, se realiza un registro automático en la base de datos cada 30 fotogramas detectados, es decir una notificación por segundo, evitando guardar muchas detecciones innecesarias. La Figura 74 muestra la función con la cual se realiza los registros, por otro lado, también se hace consultas a la base de datos cuando el frontend lo pide por medio del archivo historial.html, la Figura 75 muestra la función para esto.

```
def save_detection(objects_detected, frame, yolo_results):
    if not os.path.exists(labels_dir):
        os.makedirs(labels_dir)
    timestamp = datetime.now().strftime("%Y-%m-%d_%H%M%S")
    captura_path = f"{captura_dir}/captura_{timestamp}.jpg"
    label_path = f"{labels_dir}/captura_{timestamp}.txt"
    # Guardar imagen de la detección
    cv2.imwrite(captura_path, frame)
    # Guardar bounding boxes en .txt
    h, w, _ = frame.shape
    with open(label_path, "w") as f:
        for result in yolo_results[0].boxes.data:
            x1, y1, x2, y2, conf, cls = map(float, result)
            cls_id = int(cls)
            x_center = ((x1 + x2) / 2) / w
            y_center = ((y1 + y2) / 2) / h
            width = (x2 - x1) / w
            height = (y2 - y1) / h
            f.write(f"{cls_id} {x_center} {y_center} {width} {height}\n")
    # Guardar en la base de datos
    conn = sqlite3.connect("detecciones.db")
    cursor = conn.cursor()
    cursor.execute('INSERT INTO detecciones (Fecha, Objetos_Detectados, Ruta_Captura, Ruta_Label)
    VALUES (?, ?, ?, ?)',
    (datetime.now().strftime("%Y-%m-%d %H:%M:%S"), str(objects_detected), captura_path, label_path))
    conn.commit()
    conn.close()
    send_telegram_alert(objects_detected, captura_path)
```

Figura 74. Función para registro en la base de datos.

```
def historial():
    fecha = request.args.get('fecha')
    objeto = request.args.get('objeto')
    query = "SELECT * FROM detecciones WHERE 1=1"
    params = []
    if fecha:
        query += " AND Fecha LIKE ?"
        params.append(f"{fecha}%")
    if objeto:
        query += " AND Objetos_Detectados LIKE ?"
        params.append(f"%{objeto}%")
    conn = sqlite3.connect("detecciones.db")
    conn.row_factory = sqlite3.Row
    cursor = conn.cursor()
    cursor.execute(query, params)
    resultados = cursor.fetchall()
    conn.close()
    return render_template('historial.html', resultados=resultados)
```

Figura 75. Función para consulta en la base de datos.

d. Frontend

Se crearon varios archivos HTML para gestionar la GUI del sistema, el index.html como página principal en donde se muestran tanto el video como las métricas de detección. La Figura 76 muestra la interfaz principal, la cual consta de una sección donde se muestra el video analizado, una sección con indicadores tanto para objetos sospechosos como posturas amenazantes, botones para iniciar y detener el procesamiento y una barra de navegación que redirecciona al historial de detección, cerrar sesión y varias opciones.

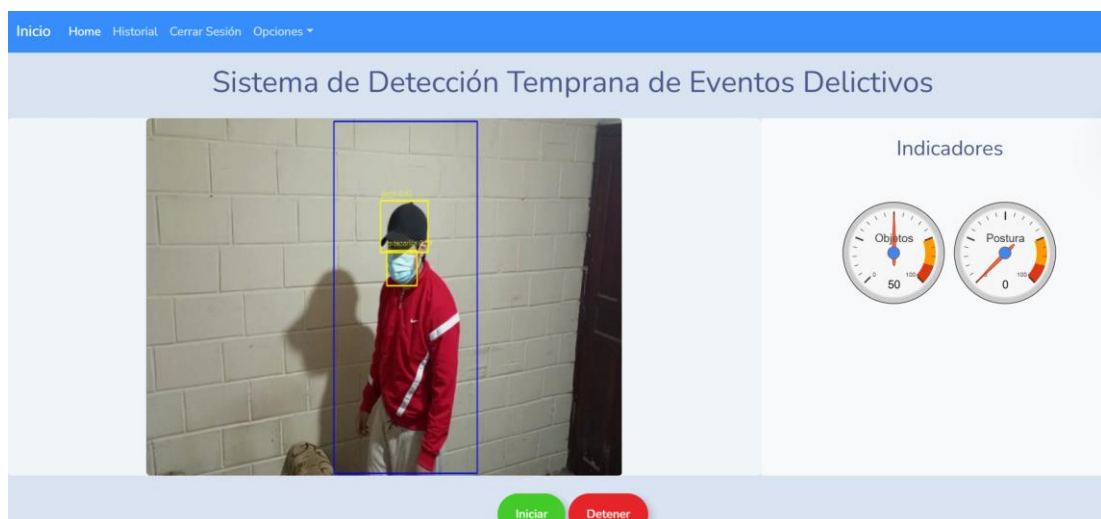
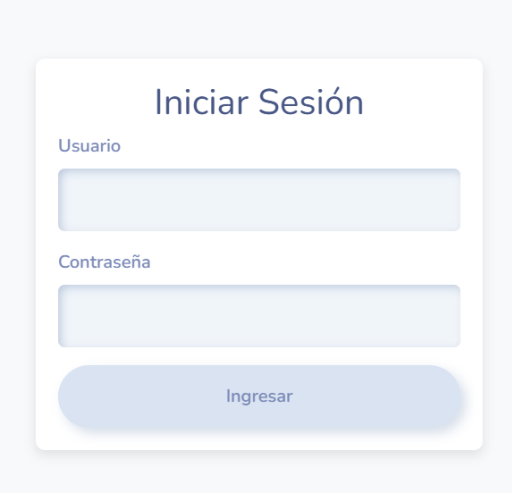


Figura 76. Interfaz principal del sistema.

Para gestionar la autenticación del usuario se realizó mediante un formulario codificado en el archivo login.html, el cual recibe los datos de usuario y contraseña

establecido para poder ingresar a la página principal, la Figura 77 muestra el diseño del formulario de autenticación, esta es la primera página que se muestra al iniciar el servidor del sistema, al presionar cerrar sesión de la barra de navegación del índice, se redirige a este formulario nuevamente.

El formulario de inicio de sesión tiene un título "Iniciar Sesión" en azul. Debajo hay dos campos de entrada: "Usuario" y "Contraseña", ambos con bordes azules. Al final del formulario hay un botón redondeado azul con el texto "Ingresar" en blanco.

Iniciar Sesión

Usuario

Contraseña

Ingresar

Figura 77. Autenticación del sistema.

El índice principal también ofrece la opción de verificar el historial de detecciones, la Figura 78 muestra el diseño de esta pestaña, la cual fue codificada en el archivo `historial.html`, en la parte superior existe elemento de ayuda para búsqueda de un registro en específico, mediante filtrado por fecha u objeto, al presionar el botón regresar no redirige al índice principal.

La interfaz "Historial de Detecciones" incluye una barra de búsqueda superior con campos para "Fecha" (formato dd/mm/aaaa) y "Tipo de Objeto" (ejemplo: arma), y botones "Buscar" y "Regresar". Debajo hay una tabla con cuatro columnas: ID, Fecha, Objetos Detectados y Captura. La tabla muestra seis registros de detección.

ID	Fecha	Objetos Detectados	Captura
1	2024-12-10 12:43:55	{'cuchillo', 'gorra', 'mascarilla'}	
2	2024-12-10 12:43:55	{'cuchillo', 'gorra', 'mascarilla'}	
3	2024-12-10 12:43:56	{'cuchillo', 'gorra', 'mascarilla'}	
4	2024-12-10 12:44:12	{'arma', 'gorra'}	
622	2024-12-18 20:55:04	{'arma'}	
623	2024-12-18 20:55:05	{'arma'}	

Figura 78. Historial de detecciones del sistema.

e. Alerta temprana

Para gestionar las alertas temprana de sospechosos y delitos inminentes, se utilizó un Bot de Telegram. Después de descargar e instalar la aplicación de mensajería, se utilizó BotFather para la configuración del Bot, al ingresar al chat se envía un mensaje de “/start” y seguimos las instrucciones que el asistente nos da para la configuración del Bot, como nombre el Bot, nombre de usuario, y el token para el acceso a la API HTTP. La Figura 79 evidencia este proceso.

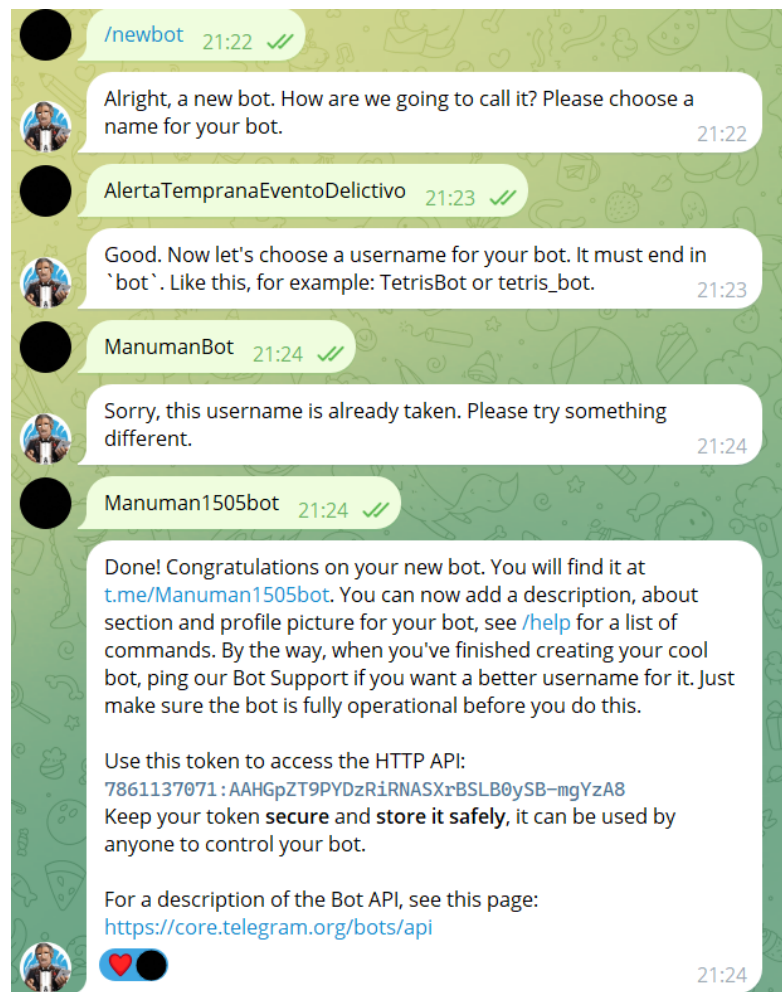


Figura 79. Configuración Bot Telegram.

El siguiente paso fue buscar el al Bot creado y mandarle un mensaje cualquiera e ingresando a la API HTTP mediante el token generado, esto para obtener el “chat_i”, estos datos son necesarios para gestionar el envío automático de mensajes. Por último nos queda codificar la estructura del mensaje. La Figura 80 muestra un mensaje de

alerta temprana especificando la etiqueta, la hora, fecha y objetos de la detección, así como una captura del fotograma en el que se detecta.



Figura 80. Mensaje de alerta en entorno controlado.

3.5 Evaluación del sistema

Se realizaron una serie de dramatizaciones de eventos delictivos principalmente asaltos con arma de fuego y cuchillo, además de estancias normales dentro de entornos comerciales, se utilizaron diferentes tipos de objetos para evaluar la generalización, así como diferentes personas en la escena. A continuación se describen con mayor detalle las pruebas realizadas.

3.5.1 Pruebas en entornos reales

Se realizaron 20 pruebas, de las cuales las primeras 10 correspondieron a situaciones de eventos delictivos entre asalto con cuchillo y arma de fuego y las otras 10 fueron situaciones normales. Se evaluó la capacidad del sistema para detectar un posible sospechoso, mediante los siguientes criterios; Detección objetos sospechosos, Detección de posturas sospechosas, detección temprana de evento delictivo. La Tabla 23 especifica los detalles de las pruebas realizadas y en el Anexo D se muestran algunas imágenes de estas pruebas.

Tabla 23. Detalles de pruebas realizadas en entornos reales.

Número	Sujeto de Prueba		Objetos	Etiqueta
	Género	Edad		
1	Hombre	25	Gorra, arma	Asalto con arma
2	Hombre	25	Pasamontana, cuchillo	Asalto con cuchillo
3	Hombre	25	Gorra, mascarilla, arma	Asalto con arma
4	Mujer	18	Pasamontana, arma	Asalto con arma
5	Hombre	27	Gorra, Mascarilla, arma	Asalto con arma
6	Hombre	27	Gorra, arma	Asalto con arma
7	Mujer	40	Arma	Asalto con arma
8	Hombre	25	Gorra, Cuchillo	Asalto con cuchillo
9	Mujer	15	Gorra, mascarilla, arma	Asalto con arma
10	Mujer	15	Cuchillo	Asalto con cuchillo
11	Mujer	15	No relevantes	Normal
12	Mujer	15	Mascarilla	Normal
13	Mujer	40	No relevantes	Normal
14	Mujer	40	Gorra, mascarilla	Normal
15	Hombre	45	Gorra, mascarilla	Normal
16	Hombre	45	No relevantes	Normal
17	Mujer	18	Gorra	Normal
18	Mujer	18	Mascarilla	Normal
19	Hombre	25	Gorra	Normal
20	Hombre	25	Mascarilla	Normal

La Figura 81 muestra la notificación temprana de una de las pruebas realizadas se desactivo el dibujo de la postura corporal para optimizar recursos. Se evidencia que se detectaron correctamente los objetos arma, y gorra, ademar las etiqueta “Delito inminente” refleja que la postura corporal fue detectada como amenazante. También se muestra la fecha y hora de la detención así como una lista de los objetos detectados.



Figura 81. Notificación de delito inminente con arma en entrono real.

La Figura 82 muestra otro de los escenarios dramatizados, en la que se evidencia la notificación de la detección de cuchillo y gorra, de la misma forma se muestra la etiqueta en este caso “Delito Inminente”, la fecha y hora de la detección y una lista con los objetos detectados.



Figura 82. Notificación de delito inminente con cuchillo en entrono real.

La Figura 83 muestra la notificación temprana de detección de un “Delito Inminente”, ya que se detectó un arma y gorra, el mensaje detalla la etiqueta, la fecha y hora de detección e igualmente una lista de los objetos detectados.



Figura 83. Notificación de delito inminente con arma con diferente persona en entrono real.

La Figura 84 muestra el desempeño del sistema en otro entorno de prueba, con diferente persona y activada el dibujo de la postura corporal, se evidencia que la detección de correcta del arma pero existe un falso positivo, etiquetando como mascarilla la barba de la persona en la escena. Ahora bien, el sistema deja de confundirse en este objeto cuando el sujeto está más cerca de la cámara, definiendo mejor las características del rostro, véase en la Figura 85.



Figura 84. Falso positivo en detección entorno real.



Figura 85. Detección en entorno real con dibujo de postura activado.

La Figura 86 muestra el desempeño del sistema otro entorno de prueba, en este caso el interior de un supermercado, se puede observar que el sistema alerta de un posible sospechoso, ya que cuenta con dos objetos detectados, mascarilla y gorra, el mensaje de alerta detalla la etiqueta “Sospechoso con objeto”, la fecha y hora de la detección y una lista de los objetos detectados.



Figura 86. Notificación de sospechoso en prueba en entorno real.

La Figura 87 muestra el funcionamiento del sistema de detección en tiempo real, y las notificaciones mediante el Bot de Telegram, me puede evidenciar la alerta temprana del delito debido a los objetos sospechosos que se detectan así como la postura corporal.



Figura 87. Notificación temprana del evento delictivo.

Como se explicó anteriormente se evaluaron tres tipos de detecciones; Detección de objetos sospechosos (DOS) estableciendo como “verdadero” si se detectaron todos los objetos aproximadamente más del 50% de la duración de la escena dramatizada, Detección de posturas sospechosas (DPS) marcando como “verdadero” si detecto correctamente la postura, y Detección temprana (DT) si la notificación de sospecha llega segundos antes del delito flagrante. La Tabla 24 resume las detecciones realizadas.

Tabla 24. Resultados de las pruebas realizadas en entornos reales.

Número	Tiempo (s)	D. Camara (m)	Etiqueta real			Detection por Sistema		
			DOS	DPS	DT	DOS	DPS	DT
1	45	1.5	V	V	V	V	V	V
2	39	1.5	V	V	V	V	V	V
3	54	1.5	V	V	V	F	V	F
4	38	1.5	V	V	V	V	F	V
5	47	1.5	V	V	V	V	V	V
6	51	1.5	V	V	V	V	F	V
7	45	1.5	V	V	V	V	V	V
8	35	1.5	V	V	V	V	V	V
9	56	1.5	V	V	V	V	F	V
10	39	1.5	V	V	V	V	V	V
11	30	1.5	F	F	F	V	F	F
12	35	1.5	V	F	F	F	F	V
13	36	1.5	F	F	F	V	V	F
14	34	1.5	V	F	F	V	F	F
15	39	1.5	V	F	F	V	F	V
16	25	1.5	F	F	F	F	F	F
17	28	1.5	V	F	F	V	F	F
18	35	1.5	V	F	F	V	V	V
19	29	1.5	V	F	F	F	F	F
20	35	1.5	V	F	F	V	F	V

3.5.2 Métricas de validación

Se procesaron los resultados obtenidos en la fase de pruebas para extraer métricas de validación descritas en el Capítulo I; Matriz de confusión, Accuracy, Precision, Recall y F1-Score para cada criterio de detección; Detección de objetos sospechosos (DOS), Detección de posturas sospechosas (DPS) y Detección temprana (DT). La Figura 88, Figura 89 y Figura 90, muestran las matrices de confusión para estos tres tipos de detecciones y la Tabla 25 resume las métricas calculadas.

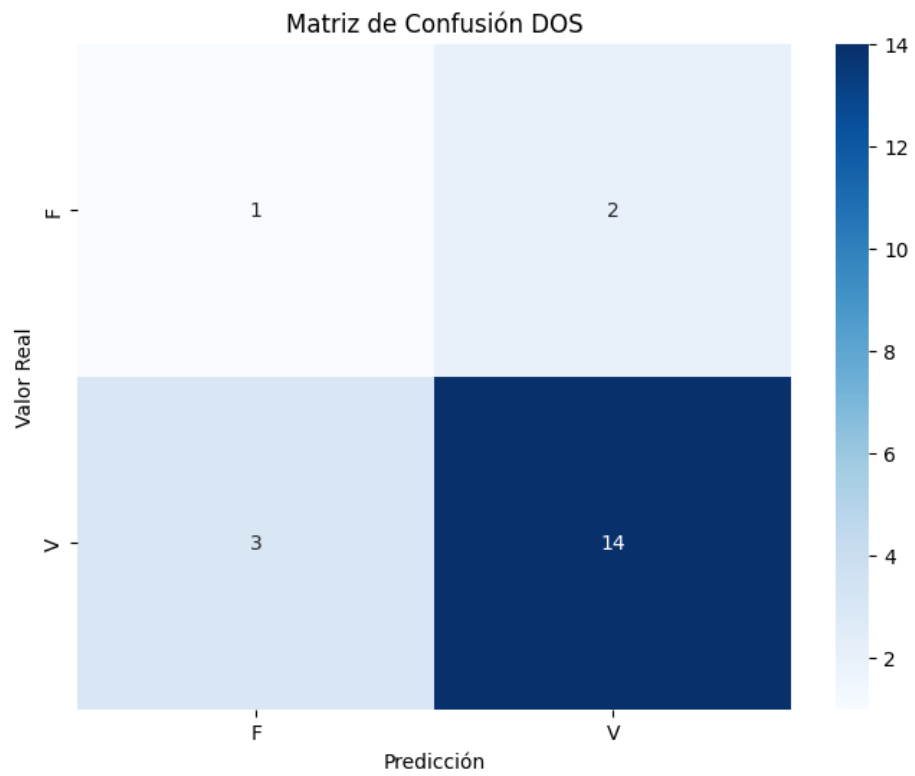


Figura 88. Matriz de confusión DOS.

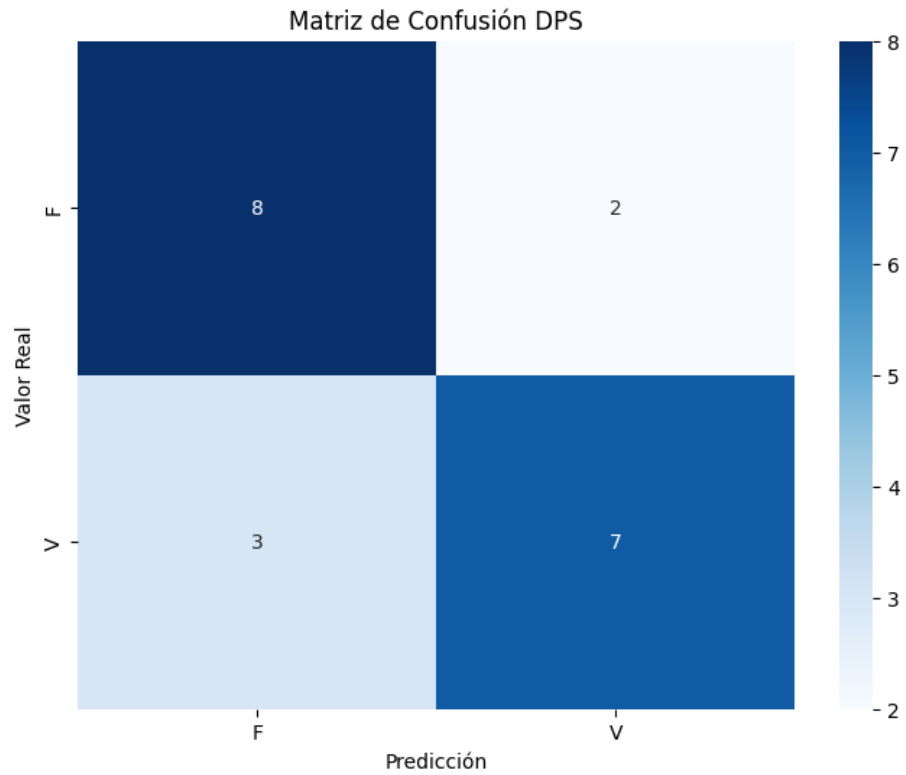


Figura 89. Matriz de confusión DPS.

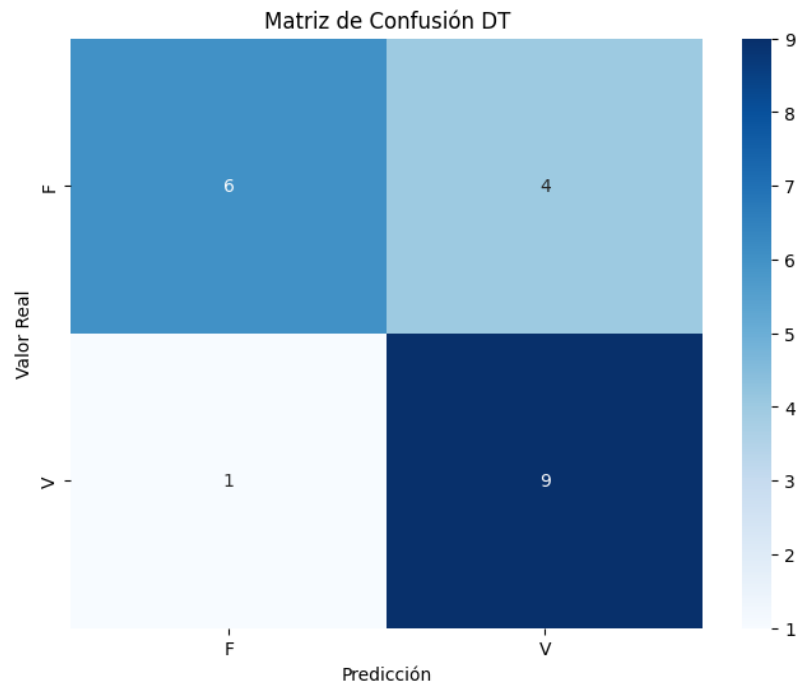


Figura 90. Matriz de confusión DT.

Tabla 25. Métricas de validación del sistema.

Detección	Accuracy	Precision	Recall	F1-Score
DOS	0.75	0.875	0.82	0.85
DPS	0.75	0.78	0.7	0.74
DT	0.75	0.69	0.9	0.78
Total	0.75	0.78	0.81	0.79

En base a las métricas obtenidas para cada criterio de detección evaluado, se estableció el rendimiento del sistema estableciendo un porcentaje al valor del F1-score total del sistema, de la siguiente manera:

$$\begin{aligned}
 \text{Rendimiento} &= f1_{score} * 100\% \\
 &= 0.79 * 100\% \\
 &= 79\%
 \end{aligned}
 \tag{7}$$

3.6 Despliegue

Después de la evaluación del sistema de detección temprana de eventos delictivo en entidades comerciales, se desarrolló un sistema gestor de información para procesar

las predicciones del enfoque multimodal, esto se desarrolló mediante FLASK. Este entorno de desarrollo permite gestionar el backend encargado del procesamiento del video y el frontend, el cual corresponde a la Interfaz gráfica de usuario.

3.6.1 Arquitectura funcional

El prototipo consta de la tarjeta Nvidia Jetson Nano como dispositivo de procesamiento, la cual se conecta a la red LAN por medio de cable de red UTP categoría 5, por otro lado la cámara IP se conecta a la WLAN por medio de conexión inalámbrica, al igual que el host para visualización en tiempo real. Por otro lado se requiere conexión a internet para poder enviar la notificación de alerta por medio un Bot de Telegram. La Figura 91 muestra la arquitectura funcional del sistema.

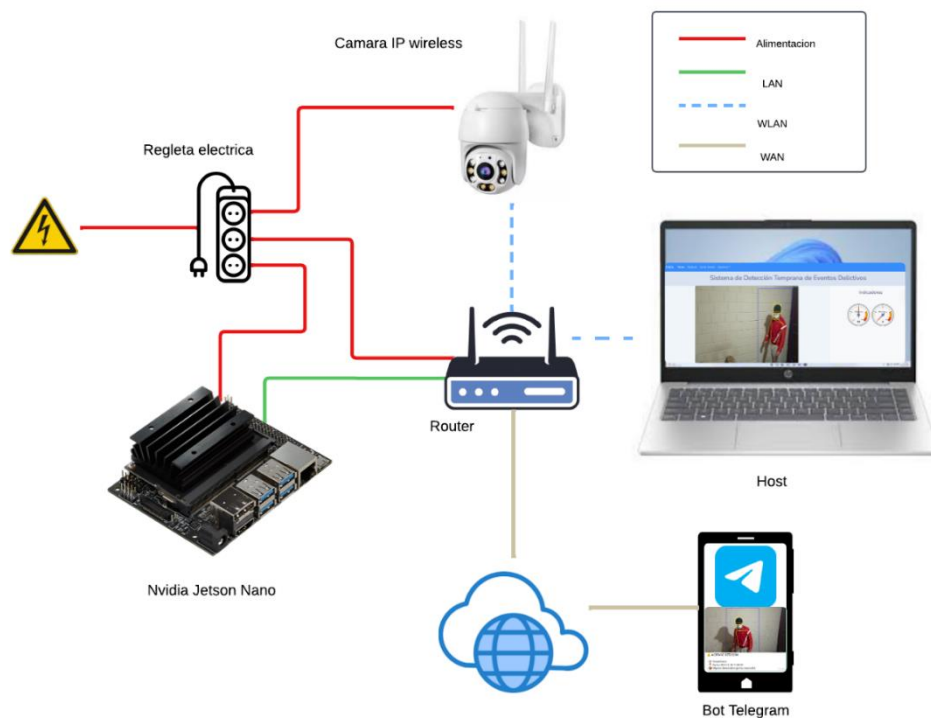


Figura 91. Arquitectura funcional del sistema.

3.6.2 Requisitos mínimos de hardware y software para despliegue

Tras las pruebas realizadas en la sección de Evaluación del Sistema, se han establecido ciertos requisitos mínimos de hardware y software para el despliegue del sistema. La Tabla 26 resume estos requisitos.

Tabla 26. Requisitos mínimos para despliegue de sistema.

	Etapa	Parametro	Descripcion
Hardware	Adquisicion de video	Conectividad	IP-WiFi
		Resolucion	2MP
		Resolucion de video	Full HD/ 1080p
		Transmision de video	RTSP - MPEG4
	Unidad de procesamiento	GPU	128-Core Maxwell
		CPU	Quad-Core ARM A57 @ 1.43GHz
		Memoria	4 GB 64-bit LPDDR4 25.6 GB/s
		Conectividad	Gigabit Ethernet
Software	Etapa	Libreria	Version
	Pre-procesamiento	Python	3.10.6
		Numpy	1.25.2
		Pandas	2.2.0
		OpenCV	4.9.0.80
	Reconocimiento	Scikit-Learn	1.4.0
		Ultralytics	8.3.36
		TensorFlow	2.15.0
		Torch	2.1.2
		Joblib	1.3.2
	Visualizacion	Flask	3.0.0
		Werkzeug	3.0.1
		Jinja2	3.1.2
		SQLite	3.46.1

3.6.3 Presupuesto

El desarrollo del sistema de alerta temprana de eventos delictivos para entidades comerciales con visión artificial y Deep learning, llevo consigo costos de diseño e implementación. A continuación se detalla estos costos

a. Costos de Diseño

El Ministerio del Trabajo del Ecuador establece el salario mensual de un ingeniero en Telecomunicaciones de 858 dólares. Tomando en cuenta 21 días laborables al mes con jornadas de 8 horas diaria, se estableció el costo de la hora de trabajo.

$$\begin{aligned}
 \text{Salario}_{\text{dia}} &= \frac{\text{Salario}_{\text{mes}}}{\text{dias de trabajo al mes}} \\
 &= \frac{\$ 858}{21} = \$ 40.86
 \end{aligned}
 \tag{8}$$

$$\begin{aligned}
 \text{Salario}_{\text{hora}} &= \frac{\text{Salario}_{\text{dia}}}{\text{horas de trabajo al dia}} \\
 &= \frac{\$ 40.86}{8} = \$ 5.11
 \end{aligned}
 \tag{9}$$

El costo de diseño se estableció en base a las horas de trabajo invertidas en el diseño, desarrollo e implementación de este sistema de detección temprana, las cuales fueron alrededor de 120 horas. multiplicándolas por el valor de la hora de trabajo de un ingeniero en Telecomunicaciones establecido en 5.11 dólares.

$$\begin{aligned}
 \text{Costo}_{\text{diseño}} &= \text{Horas invertidas} * \text{Salario}_{\text{hora}} \\
 &= 120 * \$5.11 \\
 &= \$ 613.2
 \end{aligned}
 \tag{10}$$

b. Costos de implementación

Los costos de implementación conllevan todos los materiales utilizados en la implantación del sistema, incluyendo impuestos e imprevistos. La Tabla 27 resume los costos de implementación del proyecto.

Tabla 27. Costos totales de implementación del sistema

Item	Recurso	Cantidad	Costo unitario	Costo Total
1	Nvidia Jetson Nano Kit Developer Kit	1	\$ 219.99	\$ 219.99
2	Camara IP DS-2CD2423G0-I(W)	1	\$ 29.36	\$ 29.36
3	Cable de Red	2	\$ 1.99	\$ 3.98
4	Regleta eléctrica	1	\$ 2.75	\$ 2.75
Total				\$ 256.08

c. Costo total del proyecto

El costo total del proyecto cubre tanto el diseño como la implementación por ende se sumaron dichos valores antes calculados.

$$\begin{aligned}
 \text{Costo}_{\text{total}} &= \text{Costo}_{\text{diseño}} + \text{Costo}_{\text{implemetacion}} \\
 &= \$ 613.2 + \$ 256.08 = \$ 869.28
 \end{aligned}
 \tag{11}$$

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Las entidades comerciales encuestadas cuentan con al menos un sistema electrónico de seguridad, destacándose el uso universal del CCTV (100%), generalmente con tecnología IP, mientras que otros sistemas complementarios como sensores de movimiento (88.9%) y alarmas de intrusos (77.8%) refuerzan esta estructura básica. Sin embargo, se evidencia una dependencia marcada en tecnologías tradicionales, ya que sólo un 11.1% utiliza sistemas de reconocimiento facial y el 66.7% dispone de controles de acceso. Esto revela una vulnerabilidad en la actualización y diversificación de las herramientas de seguridad, lo que limita la capacidad de detección temprana ante eventos delictivos complejos. En definitiva, aunque la estructura actual es robusta en ciertos aspectos, es necesario integrar tecnologías más avanzadas para cubrir las falencias existentes y mejorar la eficacia global del sistema.
- La dependencia humana es uno de los problemas más comunes en los sistemas de seguridad de entidades comerciales, especialmente en los sistemas de videovigilancia, y requiere una monitorización de 24 horas los 7 días de la semana para detectar o prevenir oportunamente eventos delictivos. Por lo tanto, la necesidad de contar con sistemas autónomos de vigilancia supone un gran avance en el rendimiento y la fiabilidad del sistema, ya que evita la monitorización durante periodos de tiempo en los que se producen situaciones normales.
- Para implementaciones de visión artificial, el hardware más importante son las cámaras y sus características, en particular, un sensor óptico de 2 MP CMOS o superior, una lente de 3,6 mm o superior y una resolución de 1080p como mínimo, ya que son parámetros a tener en cuenta a la hora de elegir estos dispositivos. En cuanto al procesamiento, se requieren equipos con GPU para acelerar el rendimiento de los modelos de machine learning más habituales en

visión artificial y una memoria RAM superior a 4 GB si se van a ejecutar varios modelos a la vez.

- En el ámbito del desarrollo de modelos de Machine Learning, la integración de Python en su versión 3.10.6, un lenguaje de alto nivel, con sus librerías de análisis matemático, como Numpy en su versión 1.25.2, y de procesamiento de datos, como la versión 2.2.0 de Pandas, conjuntamente con librerías complementarias, tales como OpenCV versión 4.9.0.80, Scikit Learn versión 1.4.0, Ultralytics versión 8.3.36, TensorFlow versión 2.15.0 y Torch versión 2.1.2 contribuyen significativamente a la reducción de la complejidad de la programación, centrando el esfuerzo en la definición, entrenamiento y la validación de dichos modelos de manera sencilla e intuitiva, en lugar de tecnicismos de programación de bajo nivel.
- El éxito o fracaso de un modelo de Machine Learning depende en gran medida de la calidad de los datos con los que se entrena el modelo. Por lo tanto, la creación de un conjunto de datos es la parte más importante de un proyecto de esta índole. En el caso de la detección de objetos, es estrictamente necesario que las características de la cámara con la que se probará el modelo se correspondan con las características de las imágenes de entrenamiento, especialmente en lo que respecta a la resolución y la distancia focal. Por otro lado, para la detección de posturas amenazantes con modelos SVM, se requiere un conjunto de datos extenso y representativo para lograr una generalización adecuada.
- Los modelos YOLOv8 presentan un excelente rendimiento en la tarea de detección de objetos en vídeo, ya que su arquitectura está optimizada para este tipo de aplicaciones, tanto en precisión alcanzando un 89.4% superando el 75% de modelos CNN+LSTM. Además, YOLOv8 es más ligero y rápido que arquitecturas Faster RCNN, y permite usar otros modelos de procesamiento en paralelo. Su eficiencia para configurar el entrenamiento lo hace muy atractivo para implementaciones en las que se requiera una buena relación entre precisión y optimización de recursos.

- Para la detección de posturas amenazantes, SVM como modelo clasificador de puntos clave extraídos por MediaPipe, demostró una precisión de 92% superando el 60 % obtenido con el GCN. Además el reconocimiento de posturas amenazantes como procesamiento complementario al reconocimiento de objetos sospechosos resultó eficiente a la hora de detectar oportunamente un evento delictivo, ya que centra el procesamiento en una región de interés específica, extraída en base a la postura corporal.
- En el desarrollo de sistemas informáticos para la gestión de información, Flask demostró ser un entorno de desarrollo ligero y modular, con una capacidad de procesamiento de más de 2000 solicitudes por segundo. Esta característica lo hace idóneo para la creación de sistemas gestores de información en tiempo real, evitando la sobrecarga que pueden presentar entornos más pesados, como Django. Basado en WSGI y Werkzeug, optimiza el manejo de solicitudes HTTP y permite la integración fluida de modelos de aprendizaje automático, como YOLOv8 para la detección de objetos y SVM para la clasificación de posturas, ejecutándolos en el backend sin depender de servicios externos. Su escalabilidad se potencia con herramientas como Gunicorn y Nginx, permitiendo manejar múltiples solicitudes concurrentes en producción. Además, su compatibilidad con SQLite ofrece una solución ligera para almacenar registros de detecciones sin necesidad de un servidor de base de datos. La utilización de Jinja2 para la renderización de plantillas HTML y un sistema de rutas RESTful permite la interacción entre frontend y backend, simplificando el desarrollo y la administración del sistema. En consecuencia, Flask se presenta como una opción eficiente y escalable para la detección de eventos delictivos.

4.2 Recomendaciones

- El conjunto de datos de entrenamiento debe ser extenso, variado y representativo, especialmente en tareas para detección de objetos, debe contener una gran variedad de perspectivas, tamaños, colores, apariencias, oclusiones, distancias y ángulos de enfoque, pero principalmente deben tener

características lo más cercanas a las que proporciona la cámara con la cual se implementara el sistema.

- Se recomienda probar modelos de estimación de postura para múltiples personas, ya que el sistema actual evalúa una persona a la vez. Lo cual podría resultar ineficiente en situaciones no controladas o con mucha aglomeración de personas.
- Los objetos seleccionados como sospechosos no son los únicos en eventos delictivos, establecer una lista de objetos sospechosos universal puede ser imposible, debido a los nuevos e ingeniosos métodos delincuenciales, por ende se recomienda añadir más objetos al conjunto de datos y entrenar el modelo para mejorar la eficacia y generalización del sistema.
- Se sugiere realizar la detección de posturas amenazantes con otros modelos, como Redes Convolucionales de Grafos, o incluso elevar la dimensión de análisis al evaluar temporalmente secuencias de posturas para dar una mejor predicción estableciendo el contexto temporal al análisis espacial.
- Se recomienda mejorar el Sistema Gestor de Información tanto en apariencia como en funcionalidad, añadiendo más perspectivas de la escena, así como más opciones de control y gestión. Además se puede mejorar la alerta temprana con la funcionalidades que ofrecen los Bots de Telegram.

REFERENCIAS BIBLIOGRÁFICAS

- [1] Banco Interamericano de Desarrollo, “Seguridad Ciudadana en América Latina y el Caribe.” Accessed: Sep. 18, 2024. [Online]. Available: <https://www.iadb.org/es/noticias/seguridad-ciudadana-en-america-latina-y-el-caribe>
- [2] Global Initiative Against Transnational Organized Crime, “Índice global de crimen organizado 2023,” 2023. [Online]. Available: www.globalinitiative.net
- [3] D. Montalvo, “Ecuador registra los niveles más altos de crimen, inseguridad y delincuencia del continente.” [Online]. Available: <https://www.ohchr.org/es/press-briefing-notes/2023/03/el-salvador-state-emergency>
- [4] J. Mantilla, “Tipologías y resultados de la Encuesta Nacional de Victimización de Casos de Extorsión Empresarial,” 2023. [Online]. Available: <https://thepalladiumgroup.com/>
- [5] T. Valle, A. Sánchez, T. Vayas, F. Mayorga, and C. Freire, “Inseguridad y delincuencia en Ambato octubre 2020 - marzo 2022,” 2022. Accessed: Sep. 18, 2024. [Online]. Available: https://obest.uta.edu.ec/wp-content/uploads/2022/07/inseguridad_y_delincuencia_en_ambato_2022.pdf
- [6] A. Markovitz, G. Sharir, I. Friedman, L. Zelnik-Manor, and S. Avidan, “Graph Embedded Pose Clustering for Anomaly Detection,” *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, doi: 10.1109/CVPR42600.2020.01055.
- [7] W. Iván Leturia-Rodriguez, L. Vladimir Urrelo-Huiman, and W. Leturia-Rodriguez, “Detección de Objetos y Patrones de Movimiento Usando Redes Neuronales y Algoritmos Genéticos para la Identificación de Robo a Mano Armada,” *LATIN-AMERICAN JOURNAL OF COMPUTING (LAJC)*, vol. VIII, no. 1, 2021.

- [8] Y. Yang, F. Angelini, and S. M. Naqvi, "Pose-driven human activity anomaly detection in a CCTV-like environment," *IET Image Process*, vol. 17, no. 3, pp. 674–686, Feb. 2023, doi: 10.1049/ipr2.12664.
- [9] S. Maeda, C. Gu, J. Yu, S. Tokai, S. Gao, and C. Zhang, "Frequency-Guided Multi-Level Human Action Anomaly Detection with Normalizing Flows," Apr. 2024, [Online]. Available: <http://arxiv.org/abs/2404.17381>
- [10] E. Sandoval, "Sistema de detección de situaciones delictivas en establecimientos comerciales usando inteligencia artificial," UNIVERSIDAD TÉCNICA DE AMBATO, Ambato, 2024. Accessed: Sep. 19, 2024. [Online]. Available: <https://repositorio.uta.edu.ec/bitstream/123456789/41254/1/t2563te.pdf>
- [11] *Ley Orgánica de Vigilancia y Seguridad Privada*. Asamblea Nacional del Ecuador, 2024. [Online]. Available: www.asambleanacional.gob.ec
- [12] J. Rodríguez, *Circuito cerrado de televisión y seguridad electrónica*, Segunda. España: Ediciones Parainfo, S.A, 2018.
- [13] "DS-2CE10KF0T-LFS - Cámaras Turbo HD - Hikvision." Accessed: Oct. 19, 2024. [Online]. Available: <https://www.hikvision.com/es-la/products/Turbo-HD-Products/Turbo-HD-Cameras/ColorVu-Series/ds-2ce10kf0t-lfs/>
- [14] "DS-2CE70DF0T-PF - Cámaras Turbo HD - Hikvision." Accessed: Oct. 19, 2024. [Online]. Available: <https://www.hikvision.com/es-la/products/Turbo-HD-Products/Turbo-HD-Cameras/ColorVu-Series/ds-2ce70df0t-pf/?q=ds-2ce70df0t-pfs&pageNum=1&position=1&hiksearch=true>
- [15] "System Overview DH-HAC-HFW1209C-LED 2MP Full-color HDCVI Bullet Camera 1080p IP67 Series Lite Audio DC 12V," 2020, Accessed: Oct. 19, 2024. [Online]. Available: www.dahuasecurity.com
- [16] "Camara IP Wifi Sinovision Domo PT 1080P Smart Seguimiento," Distribuidor Hikvision. Venta de camaras alarmas en Catamarca. Accessed:

- Jan. 13, 2025. [Online]. Available: <https://www.persaltumseguridad.com.ar/camara-ip-wifi-sinovision-domo-pt-1080p-smart-seguimiento--342>
- [17] A. Pérez, *Instalaciones de Telecomunicaciones*. España: EdicionesParaninfo, 2017.
- [18] CRCV, “CRCV | Center for Research in Computer Vision at the University of Central Florida.” Accessed: Oct. 16, 2024. [Online]. Available: <https://www.crcv.ucf.edu/projects/real-world/>
- [19] B. Degardin and H. Proença, “UBI Event Detection Abnormal Event Detection Dataset .” Accessed: Oct. 16, 2024. [Online]. Available: http://socialab.di.ubi.pt/EventDetection/#authors_section
- [20] “COVE - Computer Vision Exchange.” Accessed: Oct. 16, 2024. [Online]. Available: <https://cove.thecvf.com/datasets/562>
- [21] W. Luo, W. Liu, and S. Gao, “A Revisit of Sparse Coding Based Anomaly Detection in Stacked RNN Framework”.
- [22] A. Acsintoae *et al.*, “UBnormal: New Benchmark for Supervised Open-Set Video Anomaly Detection”, Accessed: Oct. 16, 2024. [Online]. Available: <https://github.com/lilygeorgescu/>
- [23] S. Szymanowicz, J. Charles, and R. Cipolla, “X-MAN: Explaining multiple sources of anomalies in video”.
- [24] “Avenue Dataset.” Accessed: Oct. 16, 2024. [Online]. Available: <https://www.cse.cuhk.edu.hk/leojia/projects/detectabnormal/dataset.html>
- [25] C. Lu, J. Shi, and J. Jia, “Abnormal Event Detection at 150 FPS in MATLAB”.
- [26] F. Chollet, *Deep Learning with Python*. Manning Publications Co., 2018.

- [27] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python A GUIDE FOR DATA SCIENTISTS*. O'Reilly Media, Inc., 2017.
- [28] “Support Vector Machine (SVM) - MATLAB & Simulink.” Accessed: Dec. 21, 2024. [Online]. Available: <https://www.mathworks.com/discovery/support-vector-machine.html>
- [29] “¿Qué es AWS? - Computación en la nube con Amazon Web Services,” Amazon Web Services, Inc. Accessed: Nov. 29, 2024. [Online]. Available: https://aws.amazon.com/es/what-is-aws/?nc1=f_cc
- [30] “Datos, análisis e IA unificados,” Amazon Web Services, Inc. Accessed: Nov. 29, 2024. [Online]. Available: https://aws.amazon.com/es/sagemaker/?nc2=h_ql_prod_ml_sm
- [31] “Google Colab,” Google Research - Explore Our Latest Research in Science and AI. Accessed: Nov. 29, 2024. [Online]. Available: <https://research.google.com/colaboratory/intl/es/faq.html>
- [32] Pona Maximiliano, “¿Qué es Kaggle? | MaselData.” Accessed: Nov. 29, 2024. [Online]. Available: <https://www.maseldata.com/post/que-es-kaggle>
- [33] “Conceptos básicos de ayuda de CRISP-DM - Documentación de IBM.” Accessed: Nov. 29, 2024. [Online]. Available: <https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview>
- [34] Gavilán Ignacio G. R., “Metodología para Machine Learning (I): CRISP-DM.” Accessed: Nov. 29, 2024. [Online]. Available: <https://ignaciogavilan.com/metodologia-para-machine-learning-i-crisp-dm/>
- [35] Costa Pocho, “Machine Learning Canvas.” Accessed: Nov. 29, 2024. [Online]. Available: <https://pochocosta.com/podcast/machine-learning-canvas/>
- [36] Gavilán Ignacio G. R., “Metodología para Machine Learning (II): Machine Learning Canvas.” Accessed: Nov. 29, 2024. [Online]. Available:

<https://ignaciogavilan.com/metodologia-para-machine-learning-ii-machine-learning-canvas/>

[37] Gavilán Ignacio G. R., “Metodología para Machine Learning (III): SEMMA.” Accessed: Dec. 01, 2024. [Online]. Available: <https://ignaciogavilan.com/metodologia-para-machine-learning-iii-semma/>

[38] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” 2015, Accessed: Jan. 10, 2025. [Online]. Available: <http://image-net.org/challenges/LSVRC/2015/>

[39] M. Abadi *et al.*, “TensorFlow, Large-scale machine learning on heterogeneous systems,” Nov. 2015. doi: 10.5281/zenodo.4724125.

[40] J. Ansel *et al.*, “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation,” in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, ACM, Apr. 2024. doi: 10.1145/3620665.3640366.

[41] “GitHub - keras-team/keras: Deep Learning for humans.” Accessed: Jan. 12, 2025. [Online]. Available: <https://github.com/keras-team/keras?tab=readme-ov-file>

[42] González Marcos Ana, Martínez de Pisón Ascacíbar Francisco Javier, and Pernía Espinoza Alpha Verónica, *Técnicas y algoritmos básicos de visión artificial*. España: Universidad de La Rioja, 2006.

[43] S. Pérez, “Detección de poses humanas mediante Deep Learning.” Accessed: Sep. 24, 2024. [Online]. Available: <https://blog.damavis.com/deteccion-de-poses-humanas-mediante-deep-learning/>

[44] “Guía de detección de puntos de referencia de posiciones | Google AI Edge | Google AI for Developers.” Accessed: Dec. 21, 2024. [Online]. Available:

https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker?hl=es-419

[45] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,” Dec. 2018, [Online]. Available: <http://arxiv.org/abs/1812.08008>

[46] M. Zhang *et al.*, “Multi-view emotional expressions dataset using 2D pose estimation,” *Sci Data*, vol. 10, no. 1, Dec. 2023, doi: 10.1038/s41597-023-02551-y.

[47] H.-S. Fang *et al.*, “AlphaPose: Whole-Body Regional Multi-Person Pose Estimation and Tracking in Real-Time,” *IEEE Trans Pattern Anal Mach Intell*, 2022, Accessed: Dec. 21, 2024. [Online]. Available: <https://github.com/MVIG-SJTU/AlphaPose>.

[48] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” Jun. 2015, [Online]. Available: <http://arxiv.org/abs/1506.02640>

[49] “Modelos compatibles con Ultralytics - Ultralytics YOLO Docs.” Accessed: Dec. 26, 2024. [Online]. Available: <https://docs.ultralytics.com/es/models/>

[50] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics YOLOv8,” 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>

[51] R. Girshick, “Fast R-CNN,” Apr. 2015, [Online]. Available: <http://arxiv.org/abs/1504.08083>

[52] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” Jun. 2015, [Online]. Available: <http://arxiv.org/abs/1506.01497>

[53] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” Dec. 2015, doi: 10.1007/978-3-319-46448-0_2.

- [54] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-Speed Tracking with Kernelized Correlation Filters,” *IEEE Trans Pattern Anal Mach Intell*, 2014.
- [55] CourseWare Wiki, “Tracking with Correlation Filters (KCF),” CourseWare Wiki. Accessed: Jan. 13, 2025. [Online]. Available: https://cw.fel.cvut.cz/b182/courses/mpv/labs/4_tracking/4b_tracking_kcf
- [56] A. Lukežič, T. Vojíř, L. Lukač, L. Zajc, J. Matas, and M. Kristan, “Discriminative Correlation Filter Tracker with Channel and Spatial Reliability”.
- [57] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “SIMPLE ONLINE AND REALTIME TRACKING,” 2017, Accessed: Jan. 09, 2025. [Online]. Available: <https://github.com/abewley/sort>
- [58] NVIDIA, “Jetson Nano Lleva el Poder de la IA Moderna a los Dispositivos en el Edge,” NVIDIA. Accessed: Jan. 11, 2025. [Online]. Available: <https://www.nvidia.com/es-la/autonomous-machines/embedded-systems/jetson-nano/product-development/>
- [59] Coral, “Dev Board,” Coral. Accessed: Jan. 11, 2025. [Online]. Available: <https://coral.ai/products/dev-board/>
- [60] Radxa, “Radxa Wiki.” Accessed: Jan. 11, 2025. [Online]. Available: https://wiki.radxa.com/Rockpi4/es_es
- [61] Django, “The web framework for perfectionists with deadlines.,” Django. Accessed: Jan. 11, 2025. [Online]. Available: <https://www.djangoproject.com/>
- [62] “Puesta en marcha de un entorno de desarrollo Django,” MDN Web Docs. Accessed: Jan. 11, 2025. [Online]. Available: https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Server-side/Django/development_environment

- [63] Flask, “Flask Documentation (3.1.x).” Accessed: Jan. 11, 2025. [Online]. Available: <https://flask.palletsprojects.com/es/stable/>
- [64] Pyramid, “Welcome to Pyramid, a Python Web Framework.” Accessed: Jan. 11, 2025. [Online]. Available: <https://trypyramid.com/>
- [65] Ultralytics, “YOLOv8,” Home - Ultralytics YOLO Docs. Accessed: Dec. 19, 2024. [Online]. Available: <https://docs.ultralytics.com/es/models/yolov8/#usage-examples>

ANEXOS

Anexo A. Categorización de variables

En la Figura A1 se muestran la categorización de variables para relacionadas al desarrollo del trabajo de titulación.

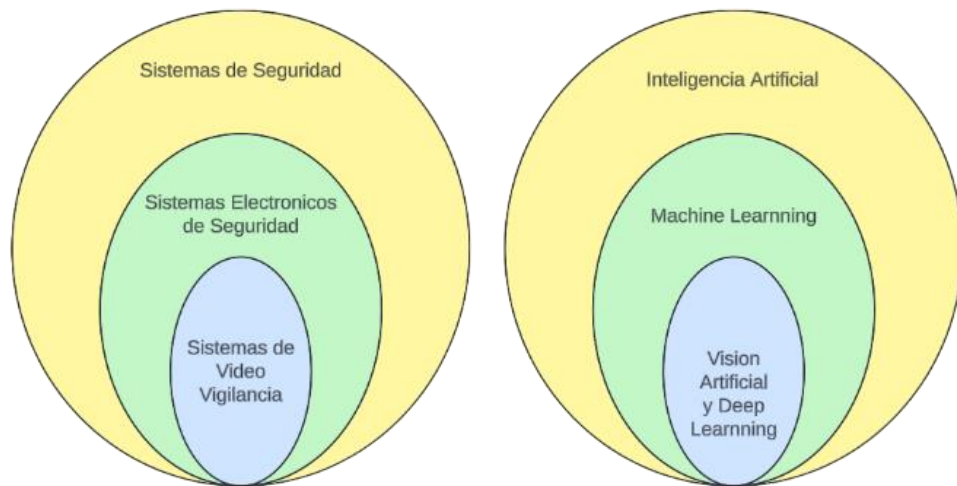


Figura A1. Categorización de variables.

En la Figura A2 y Figura A3 se muestra la constelación de ideas para el desarrollo del trabajo de titulación.

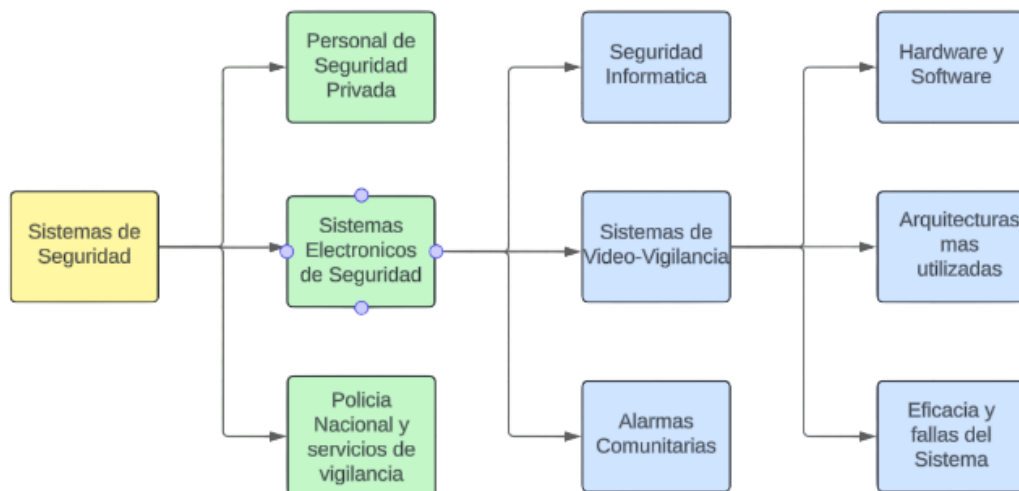


Figura A2. Constelación de ideas, variable 1

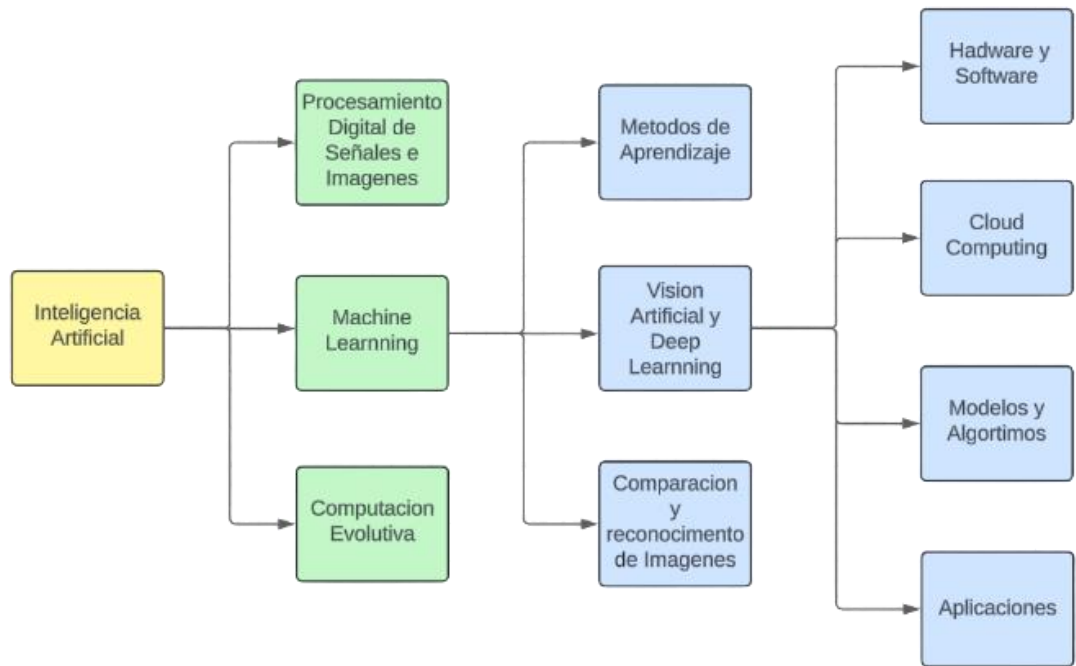


Figura A3. Constelación de ideas, variable 2

Anexo B. Cámara IP SN-MFPT20-ICS Sinovision

En la Figura B1 se muestra las características de la cámara utilizada en la implantación del trabajo de titulación.



Figura B1. Características cámara IP SN-MFPT20-ICS Sinovision.

Anexo C. Nvidia Jetson Nano Developer Kit

En la Figura C1 se muestra las especificaciones técnicas del módulo procesador utilizado en la implantación del trabajo de titulación.

NVIDIA JETSON NANO DEVELOPER KIT
TECHNICAL SPECIFICATIONS

DEVELOPER KIT	
GPU	128-Core Maxwell
CPU	Quad-Core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (Not Included)
Video Encoder	4K @ 30 4x 1080p @ 30 9x 720p @ 30 (H.264/H.265)
Video Decoder	4K @ 60 2x 4K @ 30 8x 1080p @ 30 18x 720p @ 30 (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 and eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I ² C, I ² S, SPI, UART
Mechanical	100 mm x 80 mm x 29 mm

*Please refer to NVIDIA documentation for what is currently supported.

Figura C1. Especificaciones técnicas de Nvidia Jetson Nano Developer Kit

Anexo D. Pruebas del sistemas.

En la Figura D1 y Figura D2 se muestra una detecciones en un entornos comerciales.

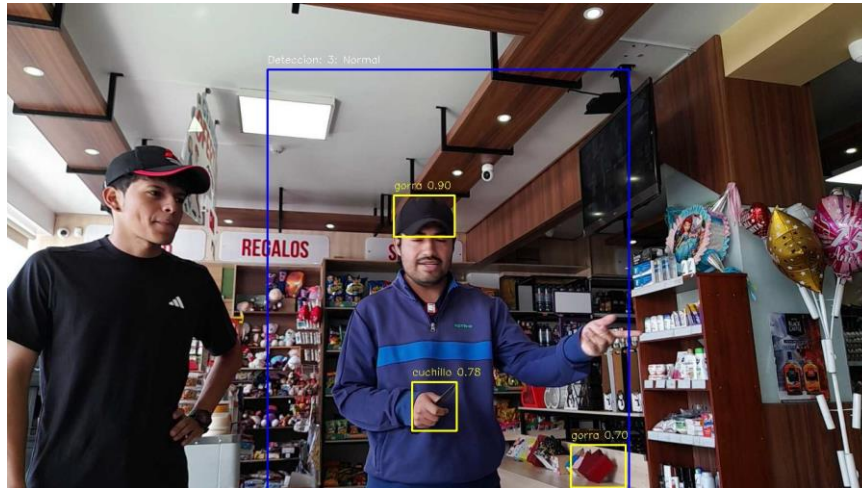


Figura D1. Detección en entorno real con dibujo de postura desactivada con dos personas en escena.

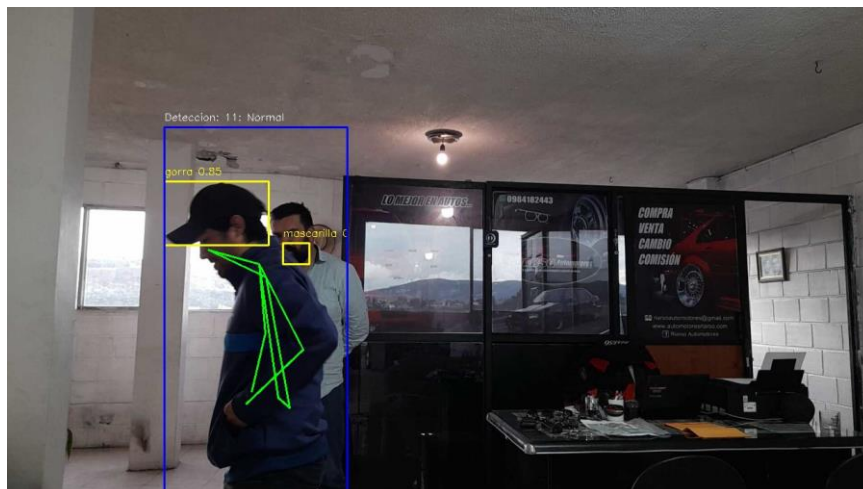


Figura D2. Detección en entorno real con dibujo de postura activada con dos personas en escena.

En la Figura D3 y Figura D4 se muestra una detección en un entorno controlado, se desactivo el dibujo de la postura corporal para optimizar recursos.

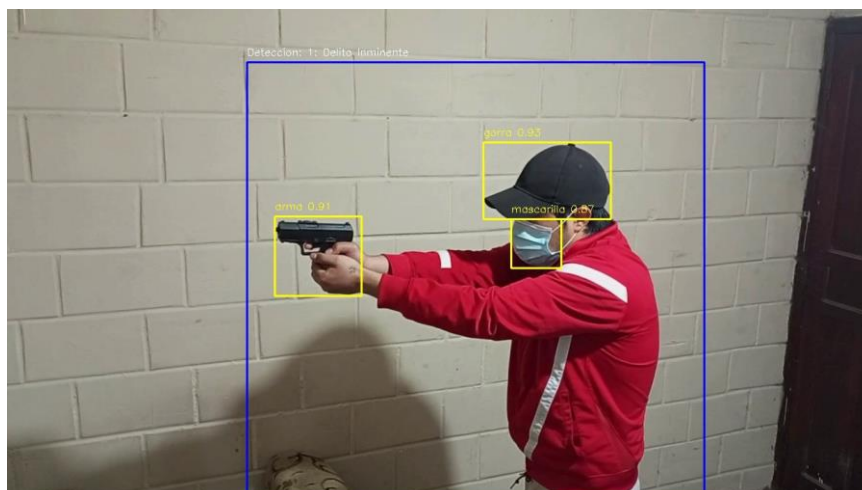


Figura D3. Detección en entorno controlado con dibujo de postura desactivada.

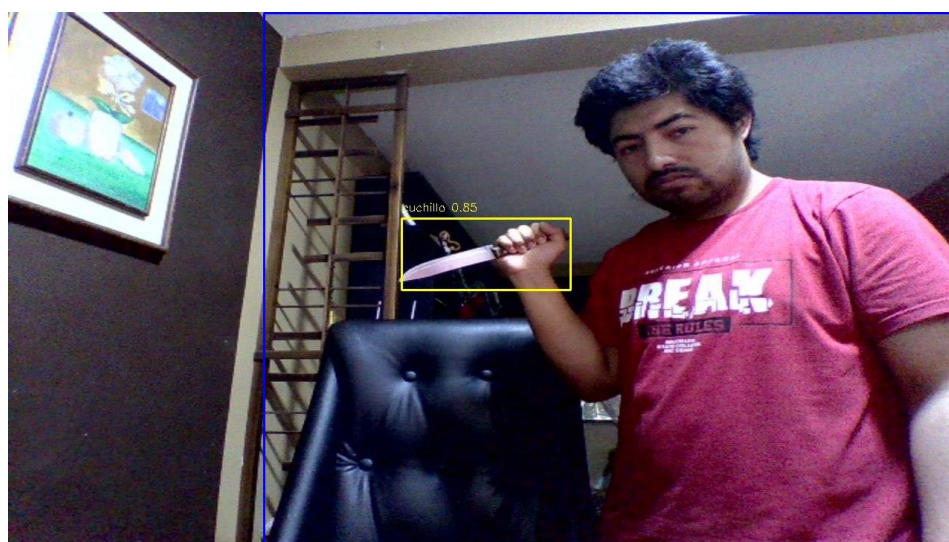


Figura D4. Detección en entorno controlado con dibujo de postura desactivada.

En la Figura D5 y Figura D6 se muestra una detección en un entorno controlado, se activó el dibujo de la postura corporal.

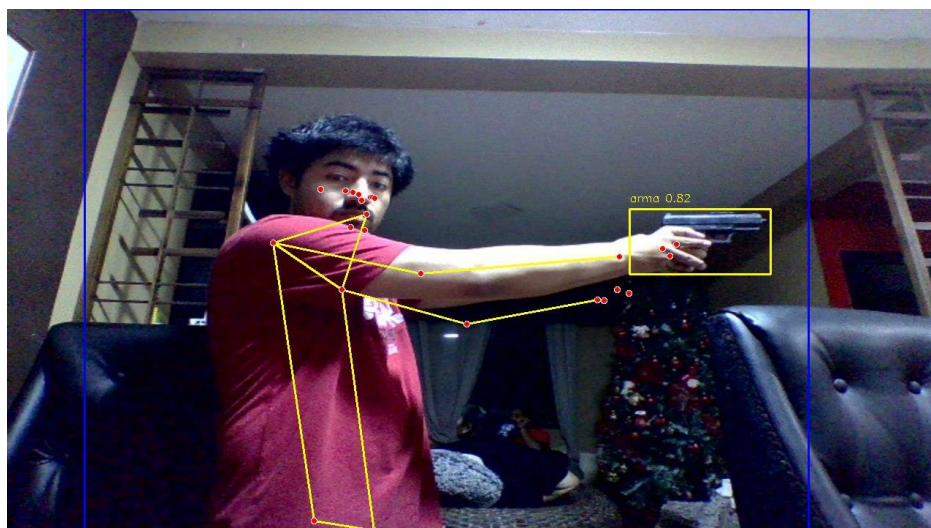


Figura D5. Detección en entorno controlado con dibujo de postura activada.



Figura D6. Detección en entorno controlado con dibujo de postura activada y etiquetas impresas en fotograma.

Anexo E. Diagramas del sistema

En la Figura E1 se muestra el diagrama de flujo del proceso de creación de conjunto de datos para la detección de posturas amenazantes.

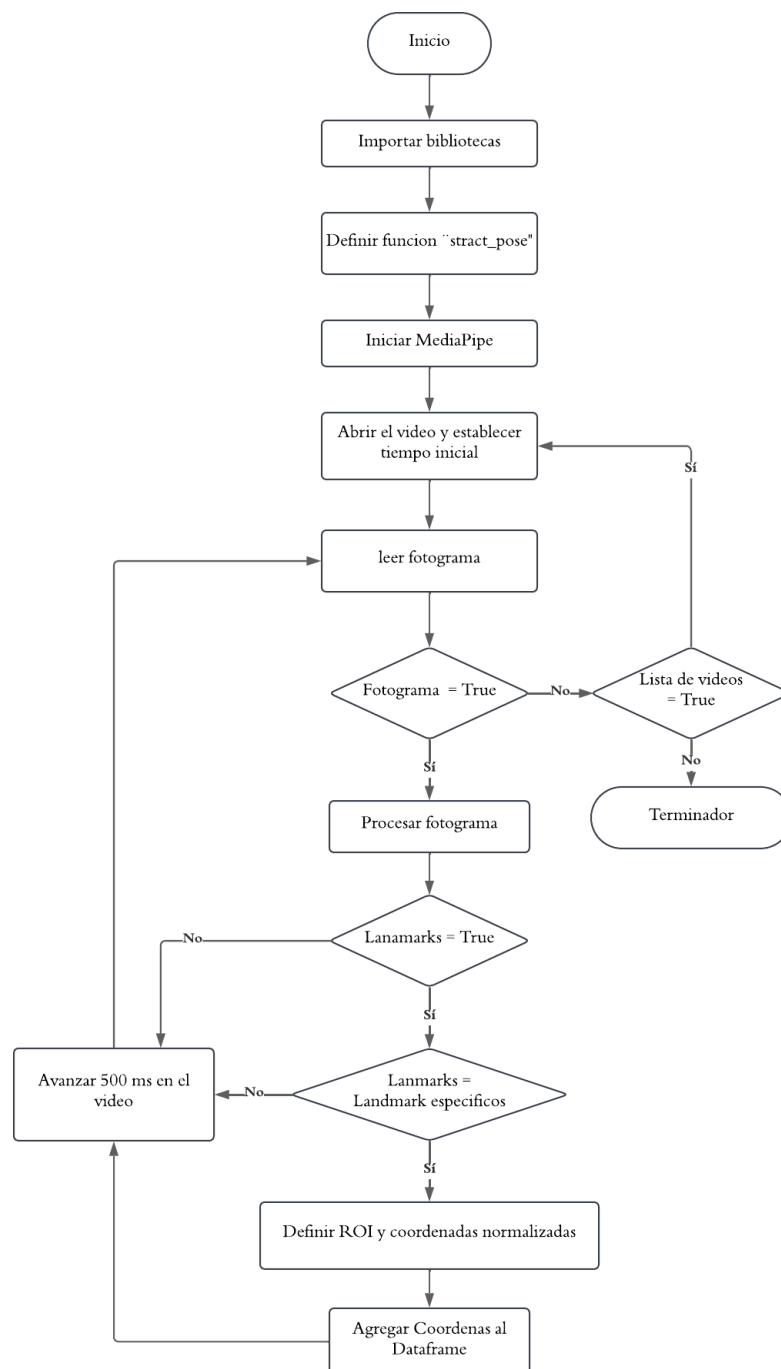


Figura E1. Diagrama de fujo para creación de conjunto de datos de entrenamiento de Rama Postura.

En la Figura E2 se muestra la arquitectura del sistemas gestor de información, mostrando la ubicación de archivos y carpetas.

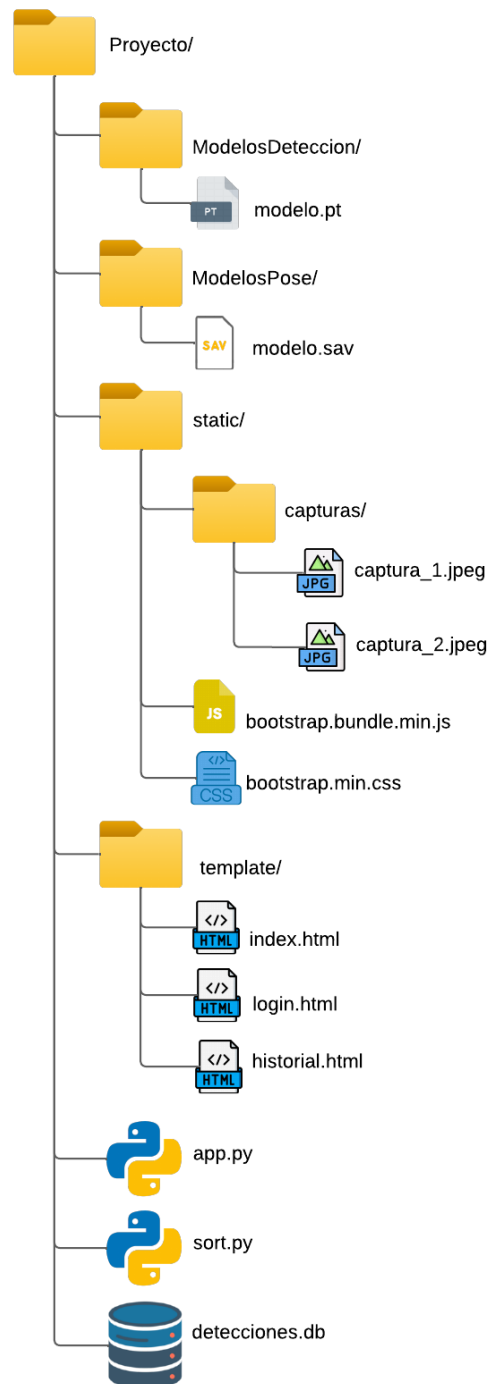


Figura E2. Arquitectura del sistema gestor de información.

Anexo F. Código del sistema

```
from flask import Flask, jsonify, Response, render_template,
request, redirect, url_for, session
import cv2
import mediapipe as mp
import numpy as np
import joblib
from ultralytics import YOLO
import sqlite3
from datetime import datetime
import os
import requests
import threading
from sort import Sort

app = Flask(__name__)

app.secret_key = "clave_secreta_para_sesiones"

#path_prueba =
r"C:\Users\User\Downloads\datasetEcuador\videospropios\Video_prueba_
2.mp4"
path_prueba = "http://192.168.85.140:8080/video"
#path_prueba = "http://192.168.3.4:8080/video"
video_processing = False
cap = cv2.VideoCapture(path_prueba)

# variables Telegram
TELEGRAM_TOKEN = "7861137071:AAHGpZT9PYDzRiRNASXrBSLB0ySB-mgYzA8" #
Reemplaza con tu token
CHAT_ID = "994549390" # Reemplaza con tu chat ID

# Credenciales
USER = "Manuman1505"
PASSWORD = "Rockandroll11850065952"

# Cargar modelos
svm_model = joblib.load('ModelosPose/modelo_entrenado_Pose_2.sav')
yolo_model = YOLO('ModelosDeteccion/best_detect_ob_15.pt')

pose = mp.solutions.pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5)
mp_drawing = mp.solutions.drawing_utils

labels = {0: "Normal", 1: "Cuchillo", 2: "Pistola"}
window_width, window_height = 640, 480
object_weights = {"gorra": 25, "mascarilla": 25, "arma": 90,
"cuchillo": 80, "pasamontana": 70}

tracker = Sort()

def calculate_roi_landmarks(landmarks, w, h):
    x_coords = [lm.x for lm in landmarks]
```

```

y_coords = [lm.y for lm in landmarks]
xmin, xmax = min(x_coords), max(x_coords)
ymin, ymax = min(y_coords), max(y_coords) # Normalizar
normalized_landmarks = [(lm.x - xmin) / (xmax - xmin), (lm.y -
ymin) / (ymax - ymin)) for lm in landmarks] # Calcular ROI con
márgenes
    x_min = max(0, int(min(x_coords) * w - 0.5 * (xmax - xmin) * w))
    y_min = max(0, int(min(y_coords) * h - 0.75 * (ymax - ymin) *
h))
    x_max = min(w, int(max(x_coords) * w + 0.5 * (xmax - xmin) * w))
    y_max = min(h, int(max(y_coords) * h + 0.75 * (ymax - ymin) *
h))

    return normalized_landmarks, (x_min, y_min, x_max, y_max)

def process_yolo_detections(roi, yolo_results, detected_objects):
    for result in yolo_results[0].boxes.data:
        x1, y1, x2, y2, conf, cls = map(float, result)
        cls_name = yolo_model.names[int(cls)]
        if cls_name in object_weights:
            detected_objects.add(cls_name)
            cv2.rectangle(roi, (int(x1), int(y1)), (int(x2), int(y2)),
(0, 255, 255), 2)
            cv2.putText(roi, f"{cls_name} {conf:.2f}", (int(x1), int(y1
- 10))), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1)

#path_prueba =
r"C:\Users\User\Downloads\datasetEcuador\videosproprios\Video_prueba_
2.mp4"
def generate_frames():
    global object_score, posture_score, etiqueta, video_processing
    etiqueta = "Normal"
    contador_frame = 0
    #cap = cv2.VideoCapture(0)
    while video_processing:
        ret, frame = cap.read()
        if not ret:
            break

        else:
            frame = cv2.resize(frame, (1280, 720))
            h, w, _ = frame.shape
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            results = pose.process(frame_rgb)
            detected_objects = set() # Evitar duplicados
            contador_frame += 1

            if results.pose_landmarks:
                colors = [(0, 255, 0)]
                landmarks = results.pose_landmarks.landmark
                upper_body_landmarks = [landmarks[i] for i in [0,
11, 12, 13, 14, 15, 16, 23, 24]]
                normalized_landmarks, (x_min, y_min, x_max, y_max) =
calculate_roi_landmarks(upper_body_landmarks, w, h)
                detections = np.array([[x_min, y_min, x_max, y_max,
1.0]]) # Agregar confianza fija

```

```

        tracked_objects = tracker.update(detections)

        for track in tracked_objects:
            x_min, y_min, x_max, y_max, track_id = map(int,
track)

            features = np.array([coord for coords in
normalized_landmarks for coord in coords])
            prediction = svm_model.predict([features])[0]
            predicted_label = labels[prediction]
            posture_score = {"Normal": 0, "Cuchillo": 70,
"Pistola": 100}.get(predicted_label, 0)

            roi = frame[y_min:y_max, x_min:x_max]

            if roi is None or roi.size == 0:
                print("ROI no válido, saltando frame")
                continue

            try:
                yolo_results = yolo_model(roi, conf=0.5)
                process_yolo_detections(roi, yolo_results,
detected_objects)

                frame[y_min:y_max, x_min:x_max] = roi

                cv2.rectangle(frame, (x_min, y_min), (x_max,
y_max), (255, 0, 0), 2)
                cv2.putText(frame, f"Deteccion: {int(track_id)}:
{etiqueta}", (x_min, y_min - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(255, 255, 255), 1)

                # Dibujar conexiones personalizadas
                connections = [
                    (16, 14), (14, 12), (12, 11), (11, 13), (0,
12),
                    (13, 15), (12, 24), (24, 23), (23, 11), (0,
11)
                ]
                for connection in connections:
                    start_idx, end_idx = connection
                    start_point = (
int(results.pose_landmarks.landmark[start_idx].x * w),
int(results.pose_landmarks.landmark[start_idx].y * h)
                    )
                    end_point = (
int(results.pose_landmarks.landmark[end_idx].x * w),
int(results.pose_landmarks.landmark[end_idx].y * h)
                    )

                    cv2.line(frame, start_point, end_point, (0,
255, 0), 2)

                # Dibujar landmarks y conexiones personalizadas

```

```

        #mp_drawing.draw_landmarks(frame,
results.pose_landmarks)

        object_score = min(100, sum(object_weights[obj]
for obj in detected_objects))
        if object_score >=0 and object_score <=25:
            etiqueta = "Normal"

        elif object_score > 25 and object_score <= 80:
            if posture_score >= 70:
                etiqueta = "Delito Inminente"
            else:
                etiqueta = "Sospechoso con objeto"
        else:
            etiqueta = "Delito Inminente"

        if object_score >= 30 and posture_score >= 0:
            if contador_frame % 30 == 0:
                save_detection_async(detected_objects,
frame)

        except Exception as e:
            print(f"Error en detección de objetos: {e}")

        frame_resized = cv2.resize(frame, (window_width,
window_height))
        _, buffer = cv2.imencode('.jpg', frame_resized)
        frame_bytes = buffer.tobytes()
        yield (b'--frame\r\n'
            b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes +
b'\r\n')

    cap.release()

def get_detections():
    conn = sqlite3.connect("detecciones.db")
    cursor = conn.cursor()
    cursor.execute("SELECT ID, Fecha, Objetos_Detectados FROM
detecciones")
    detections = cursor.fetchall()
    conn.close()
    return detections

def send_telegram_alert(objects_detected, capture_path):
    url = f"https://api.telegram.org/bot{TELEGRAM_TOKEN}/sendPhoto"
    message = (
        f"⚠️ *ALERTA DE DETECCIÓN*\n\n"
        f"- 💀 {etiqueta}\n"
        f"- 🕒 Fecha: {datetime.now().strftime('%Y-%m-%d
%H:%M:%S')}\n"
        f"- 📦 Objetos detectados: {' '.join(objects_detected)}\n"
    )
    with open(capture_path, "rb") as photo:
        payload = {"chat_id": CHAT_ID, "caption": message,
"parse_mode": "Markdown" }

```

```

        response = requests.post(url, data=payload, files={"photo":
photo})
        return response.status_code == 200

def save_detection_async(objects_detected, frame):
    """Ejecuta la función de guardado y alerta en un hilo
separado."""
    thread = threading.Thread(target=save_detection,
args=(objects_detected, frame))
    thread.start()

def save_detection(objects_detected, frame):
    captura_dir = "static/capturas"
    if not os.path.exists(captura_dir):
        os.makedirs(captura_dir)
    timestamp = datetime.now().strftime("%Y-%m-%d_%H%M%S")
    captura_path = f"{captura_dir}/captura_{timestamp}.jpg"
    cv2.imwrite(captura_path, frame)
    conn = sqlite3.connect("detecciones.db")
    cursor = conn.cursor()
    cursor.execute('INSERT INTO detecciones (Fecha,
Objetos_Detectados, Ruta_Captura)VALUES (?, ?, ?)',
(datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
str(objects_detected), captura_path))
    conn.commit()
    conn.close()
    send_telegram_alert(objects_detected, captura_path)

@app.route('/')
def login():
    if 'logged_in' in session and session['logged_in']:
        return redirect(url_for('index'))
    return render_template('login.html')

@app.route('/login', methods=['POST'])
def authenticate():
    username = request.form['username']
    password = request.form['password']
    if username == USER and password == PASSWORD:
        session['logged_in'] = True
        return redirect(url_for('index'))
    else:
        return render_template('login.html', error="Credenciales
inválidas")

@app.route('/logout')
def logout():
    session.pop('logged_in', None)
    return redirect(url_for('login'))

@app.route('/historial', methods=['GET'])
def historial():
    fecha = request.args.get('fecha')
    objeto = request.args.get('objeto')
    query = "SELECT * FROM detecciones WHERE 1=1"
    params = []

```

```

    if fecha:
        query += " AND Fecha LIKE ?"
        params.append(f"{fecha}%")
    if objeto:
        query += " AND Objetos_Detectados LIKE ?"
        params.append(f"%{objeto}%")
    conn = sqlite3.connect("detecciones.db")
    conn.row_factory = sqlite3.Row
    cursor = conn.cursor()
    cursor.execute(query, params)
    resultados = cursor.fetchall()
    conn.close()
    return render_template('historial.html', resultados=resultados)

@app.route('/index')
def index():
    # Verifica si está autenticado
    if 'logged_in' in session and session['logged_in']:
        return render_template('index.html')
    return redirect(url_for('login'))

@app.route('/video_feed')
def video_feed():
    global video_processing
    if video_processing:
        return Response(generate_frames(), mimetype='multipart/x-
mixed-replace; boundary=frame')
    else:
        return Response('', mimetype='multipart/x-mixed-replace;
boundary=frame')

@app.route('/indicator_data')
def indicator_data():
    return {"object_score": object_score, "posture_score":
posture_score}

@app.route('/start_video', methods=['POST'])
def start_video():
    global video_processing
    video_processing = True
    return jsonify({'status': 'Video started'})

@app.route('/stop_video', methods=['POST'])
def stop_video():
    global video_processing
    video_processing = False
    return jsonify({'status': 'Video stopped'})

if __name__ == "__main__":
    app.run(debug=True)

```