Sistema de Seguridad Inteligente: Detección de Situaciones Delictivas y Violentas Utilizando IA

Nain Rossetti

***Resumen*—Este estudio presenta el desarrollo de un modelo de inteligencia artificial para la detección en tiempo real de situaciones delictivas en videos de vigilancia. Utilizando una arquitectura de redes neuronales recurrentes con convoluciones (LRCN), el modelo fue entrenado con un conjunto de datos compuesto por videos de situaciones normales y violentas. La metodología incluyó el procesamiento y normalización de frames de video, y el entrenamiento del modelo en una MacBook Pro con chip M1. Los resultados muestran que el modelo puede identificar actividades delictivas con alta precisión, aunque enfrenta desafíos en la generalización a nuevos datos. Se observó que las gráficas de pérdida y exactitud reflejan un buen aprendizaje pero también indican posibles sobreajustes. Este sistema tiene el potencial de mejorar significativamente la seguridad pública al proporcionar alertas automáticas y permitir una respuesta más rápida a situaciones delictivas.**

***Palabras Claves*—** IA, Redes Neuronales, CCN, LSTM, LRCN

I. Nomenclaturas

[IA]: Inteligencia Artificial
[LRCN]: Redes Convolucionales Recurrentes a Largo Plazo
[LSTM]: Redes con Memoria a Largo y Corto Plazo

II. Introduccion

A pesar de la existente distribucion de cámaras de seguridad en diversas ubicaciones de la ciudad de Buenos Aires, persiste un problema significativo en cuanto a la prevención y respuesta efectiva ante situaciones delictivas y de violencia. La supervisión de estas cámaras se lleva a cabo manualmente por parte de operadores humanos, lo que resulta, en una limitada cobertura y capacidad de respuesta en tiempo real. Esta falta de cobertura dificulta la detección oportuna de actividades delictivas en todos los rincones de la ciudad.

III. Caso de estudio

Mediante el uso de tecnología de inteligencia artificial y la instalación estratégica de cámaras en la comuna 14, este proyecto tiene como objetivo la detección en tiempo real de actividades delictivas, particularmente robos agravados y tentativas de robo. La implementación permitirá la activación automática de alertas ante la ocurrencia de tales eventos, posibilitando una respuesta inmediata y así poder reducir los daños y pérdidas.

IV. Marco teorico

IV-A. Inteligencia Artificial

La inteligencia artificial (IA) es un campo interdisciplinario de la ciencia y la ingeniería que se enfoca en crear máquinas y sistemas capaces de realizar tareas que típicamente requieren inteligencia humana. Estas tareas incluyen el reconocimiento de patrones, el procesamiento de lenguaje natural, la toma de decisiones, la percepción visual y más. La IA se sustenta en diversas disciplinas como la informática, el análisis de datos, las matemáticas, la neurociencia y la psicología.

La inteligencia artificial (IA) puede clasificarse de varias maneras, principalmente según las etapas de desarrollo o las capacidades que poseen las máquinas. Se reconocen generalmente cuatro categorias principales de IA:

- Máquinas reactivas: Este tipo de IA es limitada y solo responde a estímulos específicos basados en reglas preprogramadas. No utiliza memoria y, por lo tanto, no puede aprender de nuevos datos.
- Memoria limitada: La mayoría de la IA moderna cae en esta categoría. Este tipo de IA puede usar memoria para mejorar con el tiempo mediante el entrenamiento con datos nuevos, típicamente a través de redes neuronales artificiales o modelos de aprendizaje profundo.
- Teoría de la mente: Actualmente, no existe IA con teoría de la mente, pero se están investigando distintas posibilidades. Este término se refiere a una IA que podría emular la mente humana, teniendo capacidades de toma de decisiones similares a las de un ser humano, incluyendo el reconocimiento y recuerdo de emociones, y la reacción en situaciones sociales como lo haría una persona.
- Autoconocimiento: Este concepto describe una IA hipotética que tendría conocimiento de su propia existencia y capacidades intelectuales y emocionales comparables a las de un ser humano. Al igual que la IA con teoría de la mente, la IA con autoconciencia no existe en la actualidad.

IV-B. Redes Neuoronales

Las redes neuronales son modelos computacionales inspirados en el funcionamiento del cerebro humano, diseñados para realizar tareas de procesamiento y análisis de datos. Están compuestas por un conjunto de nodos de procesamiento, llamados neuronas artificiales o perceptrones, organizados en capas interconectadas. Estas redes reciben entradas, las procesan a través de múltiples capas y generan salidas en función de los patrones identificados en los datos de entrada.
Cada tipo de red neuronal tiene sus propias características y aplicaciones específicas, lo que las hace adecuadas para diferentes tipos de problemas y tareas de inteligencia artificial. Para la detección de video, se pueden combinar dos tipos de arquitecturas, las CNN y las LSTM, que en conjuto forman las LRCN (Redes Convolucionales Recurrentes a Largo Plazo). Este tipo de arquitectura es especialmente útil en aplicaciones que requieren comprensión de videos, análisis de secuencias de imágenes o procesamiento de datos temporales, como la clasificación de acciones en videos, el reconocimiento de actividades humanas o la generación de descripciones de video automática. En esta arquitectura, las CNN se encargan de extraer características espaciales de cada cuadro individual de un video, capturando detalles como bordes, texturas y patrones. Posteriormente, estas características se alimentan secuencialmente a través de una LSTM, que modela las dependencias temporales y aprende cómo evolucionan estas características a lo largo del tiempo. Esta combinación permite que las LRCN analicen no solo el contenido visual de cada cuadro, sino también la secuencia temporal en la que ocurren, lo que las hace especialmente efectivas para tareas como el reconocimiento de acciones en videos, la generación automática de descripciones de videos y la clasificación de secuencias de imágenes.

IV-C. Situación delictiva actual en la comuna 14 de la Ciudad Autonoma De Buenos Aires

Según los últimos informes proporcionados por el gobierno de la Ciudad de Buenos Aires como se observa en el grafico 1, la tasa de robos agravados por el resultado de lesiones y/o muertes es de 46.9 por cada 100,000 habitantes, mientras que la tasa de tentativas de robo agravado por el resultado de lesiones y/o muertes es de 10.9 por cada 100,000 habitantes. Estos valores muestran un incremento con respecto a los datos publicados en años anteriores. En el caso de análisis específico de la Comuna 14, se observa que la tasa de robos agravados por el resultado de lesiones y/o muertes corresponde al 35.2 % , mientras que la tasa de tentativas de robo agravado por el resultado de lesiones y/o muertes es del 7.5 %. Con el propósito de reducir estas cifras y promover acciones más efectivas y seguras, es que nace el proposito de este proyecto.

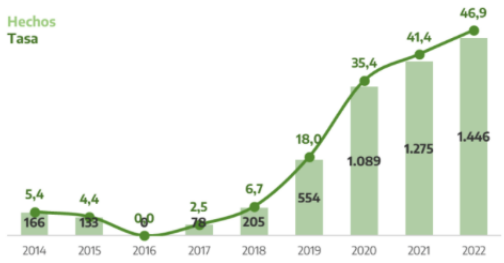


Figura 1. Hechos de robos agravados por el resultado de lesión y/o muertes por año.

IV-D. Interfaz de Programación de Aplicaciones

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de definiciones y protocolos que se utiliza para diseñar e integrar el software de las aplicaciones. Permite la comunicación entre productos y servicios, sin necesidad de conocer su implementación interna. Las API simplifican el desarrollo de aplicaciones, ofreciendo flexibilidad, facilidad en el diseño y administración, y oportunidades de innovación. También funcionan como contratos con documentación que representa un acuerdo sobre cómo deben estructurarse las solicitudes y respuestas.
Las API facilitan la colaboración entre equipos comerciales y de TI, permitiendo la integración rápida de nuevas aplicaciones en arquitecturas existentes.

V. Desarrollo

V-A. Funcionalidad de la API

El objetivo de la API es la detección temprana de situaciones delictivas en entornos urbanos utilizando tecnologías de inteligencia artificial, mediante la intervención de un modelo desarrollado y entrenado por redes neuronales LRCN. Este modelo, capacitado con un amplio conjunto de datos que incluyen tanto situaciones delictivas como no delictivas, es capaz de capturar de manera efectiva las características espaciales y temporales de un conjunto de frames de video, permitiendo discernir entre actividades criminales y actividades cotidianas. A su vez, el sistema fragmenta las filmaciones en frames y analiza cada uno de ellos para determinar la presencia de comportamientos o patrones asociados a actividades delictivas. Una vez completado el análisis, el sistema emite una alerta en caso de detectar una situación sospechosa, lo que permite una respuesta rápida y eficaz por parte de las autoridades correspondientes. Este enfoque presenta un potencial significativo para mejorar la seguridad urbana y prevenir la incidencia delictiva en áreas urbanas densamente pobladas.

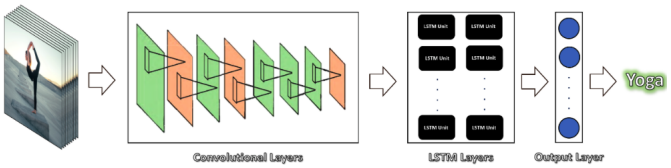


Figura 2. Funcionamiento del modelo LRCN

V-B. Requerimientos de la API

Requerimientos Funcionales:

- La API debe permitir la carga automatizada de archivos de video en formato MP4 a través de solicitudes HTTP.
- Para el procesamiento de video se debe extraer frames de los videos cargados a intervalos regulares. Estos frames deben poder ser redimensionados a un tamaño predeterminado y convertidos a escala de grises.
- La predicción se realizara utilizando el modelo entrenado con la arquitectura LRCN para analizar los frames extraídos de los videos previamente cargados y se debe retornar un resultado indicando si se detectó una situación delictiva o no, junto con la probabilidad de dicha detección.
- En caso de detectar una situación delictiva, se debe generar una alerta que se pueda enviar a los sistemas de monitoreo correspondientes, además de registrar dichas alertas en un log para auditoría y seguimiento.

Requerimientos no funcionales:

- La API debe ser capaz de manejar múltiples solicitudes de carga y procesamiento de videos de manera concurrente. Por otro lado, debe optimizar el uso de recursos computacionales para permitir el análisis en tiempo real.
- Debe contar con una autenticación y autorización para el acceso a la API y garantizar la protección de los datos cargados y generados.
- Minimizar el tiempo de procesamiento para cada video cargado. Asegurar una latencia mínima para las predicciones y generación de alertas.

V-C. Tecnologías y Herramientas

Para la implementación de esta API se utilizará el lenguaje de programación Python, además de frameworks y librerías como Flask o Django para la creación de la misma. Para la gestion del modelo de aprendizaje profundo se usaran las libreias TensorFlow y Keras, mientras que la librearia OpenCV se encargar de la manipulación y procesamiento de imágenes. Por otro lado, el almacenamiento de logs y registros de alertas se gestionará mediante bases de datos como PostgreSQL o MongoDB. Además, el despliegue de la API se realizará en un entorno de nube, utilizando servicios como AWS, Azure o Google Cloud para garantizar la escalabilidad y disponibilidad.

V-D. Arquitectura

La arquitectura de la API se estructurará en diferentes capas para garantizar un funcionamiento eficiente y escalable. En primer lugar, el enrutador se encargará de dirigir las solicitudes entrantes a las rutas adecuadas de la API, según el verbo HTTP utilizado y la URL especificada. Luego, el controlador procesará estas solicitudes, interactuando con la capa de procesamiento para llevar a cabo las operaciones necesarias. El modelo de detección de situaciones delictivas, basado en redes neuronales, será el núcleo de la API. Este modelo recibirá los datos de entrada, que consistirán en los frames de video procesados, y realizará las predicciones correspondientes sobre la presencia de actividades delictivas en el video. Una vez que se obtengan las predicciones, el controlador gestionará la respuesta,

devolviendo los resultados al cliente. Además, se implementará una capa de acceso a datos para interactuar con la base de datos, donde se almacenarán los registros y logs de la API. Esto permitirá realizar un seguimiento de las solicitudes entrantes, los resultados de las predicciones y cualquier otro dato relevante para el análisis y la mejora continua del sistema.

V-E. Diseño y configuración

Para el procesamiento de los videos, se debe implementar un mecanismo para extraer una cantidad específica de frames de cada video. Este debe comenzar leyendo el video y calculando el intervalo necesario para seleccionar los frames, asegurando que la secuencia de estos esté equitativamente espaciada. Cada frame seleccionado se ajustará a un tamaño predefinido, para luego convertirse a blanco y negro y por último normalizar para que los valores de los píxeles estén entre 0 y 1(véase Figura 2). Esto tendrá como beneficio una reducción significativa de los tiempos de entrenamiento del modelo y los costos computacionales.

```python
def frames_extraction(video_path):

    frames_list = []

    video_reader = cv2.VideoCapture(video_path)

    # Se obtiene la cantidad total de frames del video
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    # Calculo de los saltos de frames en base a una constante definida previamente
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)

    for frame_counter in range(SEQUENCE_LENGTH):

        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)
        success, frame = video_reader.read()
        if not success:
            break

        # Redimension del frame
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

        # Conversion a blanco y
        resized_frame = cv2.cvtColor(resized_frame,cv2.COLOR_BGR2GRAY)

        # Normalizacion del Frame
        normalized_frame = resized_frame / 255

        frames_list.append(normalized_frame)

    video_reader.release()
    return frames_list
```

Figura 3. Función de extracción de frames del video.

El diseño de la red neuronal tendrá varias capas Conv2D distribuidas en el tiempo, seguidas de capas MaxPooling2D y Dropout. Las características extraídas de las capas Conv2D se aplanarán utilizando la capa Flatten y se alimentarán a una capa LSTM. Luego, la capa Dense con activación softmax utilizará la salida de la capa LSTM para predecir la acción que se está realizando. (véase Figura 3)

La configuración de las capas sera:

- Capas Conv2D y MaxPooling2D Distribuidas en el Tiempo:
  - La primera capa Conv2D tiene 16 filtros con un tamaño de kernel de (3, 3) y una activación ReLU, seguida de una capa MaxPooling2D con un tamaño de pool de (4, 4) y una capa Dropout del 25 %.
  - La segunda capa Conv2D tiene 32 filtros con un tamaño de kernel de (3, 3) y una activación ReLU, seguida de una capa MaxPooling2D con un tamaño de pool de (4, 4) y una capa Dropout del 25 %.
  - La tercera capa Conv2D tiene 64 filtros con un tamaño de kernel de (3, 3) y una activación ReLU, seguida de una capa MaxPooling2D con un tamaño de pool de (2, 2) y una capa Dropout del 25 %.
  - La cuarta capa Conv2D tiene 64 filtros con un tamaño de kernel de (3, 3) y una activación ReLU, seguida de una capa MaxPooling2D con un tamaño de pool de (2, 2).
- Capas de Aplanamiento y LSTM:
  - Una capa TimeDistributed(Flatten) para procesar la salida de las capas Conv2D.
  - Una capa LSTM con 32 unidades para procesar esta ultima salida.
- Capa de Predicción:
  - Por ultimo una capa Dense con activación softmax que se utilizara para predecir la acción que se está realizando, basada en la salida de la capa LSTM.

```python
def create_model():
    model = Sequential()

    ##Arquitectura
    model.add(layers.TimeDistributed(layers.Conv2D(16, (3, 3), padding='same',activation = 'relu'),
                                     input_shape = (SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH, 1)))

    model.add(layers.TimeDistributed(layers.MaxPooling2D((4, 4))))
    model.add(layers.TimeDistributed(layers.Dropout(0.25)))

    model.add(layers.TimeDistributed(layers.Conv2D(32, (3, 3), padding='same',activation = 'relu')))
    model.add(layers.TimeDistributed(layers.MaxPooling2D((4, 4))))
    model.add(layers.TimeDistributed(layers.Dropout(0.25)))

    model.add(layers.TimeDistributed(layers.Conv2D(64, (3, 3), padding='same',activation = 'relu')))
    model.add(layers.TimeDistributed(layers.MaxPooling2D((2, 2))))
    model.add(layers.TimeDistributed(layers.Dropout(0.25)))

    model.add(layers.TimeDistributed(layers.Conv2D(64, (3, 3), padding='same',activation = 'relu')))
    model.add(layers.TimeDistributed(layers.Flatten()))

    model.add(layers.LSTM(32))

    model.add(layers.Dense(len(CLASSES_LIST), activation = 'softmax'))
    model.summary()

    return model
```

Figura 4. Configuración de la red neuronal LRCN

Para el entrenamiento del modelo, el conjunto de datos se debe dividir en dos partes: el 75 % será para el entrenamiento y el 25 % para la validación. Esta división permite evaluar el rendimiento del modelo en datos no vistos durante el entrenamiento, garantizando una mejor generalización y precisión en la detección de situaciones delictivas. En cuanto al Hardware a utilizar para el entrenamiento y la implementación del modelo se realizarán en una MacBook Pro con chip M1, 16 GB de RAM y macOS Ventura (13.5.2). El potente rendimiento del chip M1 y la capacidad de memoria permitirán manejar el procesamiento intensivo de video y el entrenamiento del modelo de manera eficiente.

V-F. Métricas del entrenamiento

Durante el entrenamiento del modelo, se utilizó un conjunto de datos compuesto por 950 videos de situaciones normales (no violentas) y 1150 videos de situaciones violentas. El modelo se entrenó durante 25 épocas, con una división de los datos del 75 % para el entrenamiento y el 25 % para la validación.

Los resultados del entrenamiento del modelo son los siguientes:
Exactitud en el conjunto de entrenamiento: 85.14 %
Pérdida en el conjunto de entrenamiento: 0.3879
Exactitud en el conjunto de validación: 89.08 %
Pérdida en el conjunto de validación: 0.4825

El gráfico 5, que detalla la pérdida, muestra una disminución constante del conjunto de entrenamiento, mientras que la pérdida del conjunto de validación presenta algunas oscilaciones, sugiriendo que el modelo podría estar sobreajustándose a los datos de entrenamiento. El gráfico 6, que detalla la exactitud indica una mejora progresiva del conjunto de entrenamiento, con la exactitud de validación alcanzando un valor

más alto, aunque también presenta fluctuaciones. Estos resultados reflejan que el modelo tiene una alta capacidad para aprender de los datos de entrenamiento, pero enfrenta desafíos en la generalización a nuevos datos, lo que se evidencia por las variaciones en la pérdida y exactitud de validación.

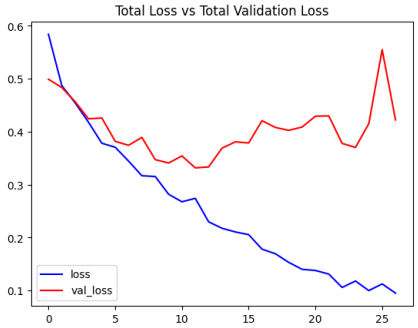


Figura 5. Total Loss vs Total Validation Loss

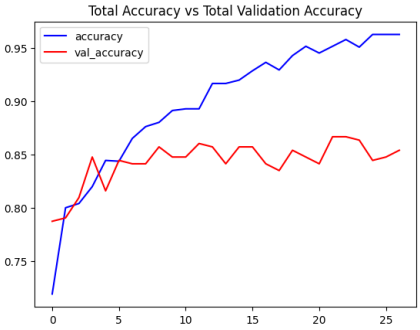


Figura 6. Total Accuracy vs Total Validation

VI. Conclusión

Me propuse desarrollar un modelo de detección de situaciones delictivas en tiempo real utilizando un enfoque basado en redes neuronales recurrentes con convoluciones (LRCN) para analizar grabaciones de video. Para el entrenamiento del modelo, se utilizó un conjunto de datos compuesto por 950 videos de situaciones normales (no violentas) y 1150 videos de situaciones violentas en la calle.
El modelo fue entrenado y validado con una división de los datos del 75 % para entrenamiento y el 25 % para validación, y se implementó utilizando frameworks y librerías como TensorFlow, Keras y OpenCV en Python. Durante el proceso de entrenamiento, se monitorizó la pérdida y la exactitud tanto en los conjuntos de entrenamiento como de validación.
Logré obtener un modelo con una exactitud de entrenamiento de aproximadamente 85.14 % y una pérdida de entrenamiento de 0.3879. La exactitud en el conjunto de validación alcanzó el 89.08 %, mientras que la pérdida de validación fue de 0.4825. Estos resultados indican que el modelo tiene una alta capacidad para aprender de los datos de entrenamiento. Sin embargo, la inestabilidad observada en las curvas de validación sugiere que el modelo podría estar sobreajustándose a los datos de entrenamiento y no generalizando adecuadamente a nuevos datos.
Las gráficas de pérdida y exactitud a lo largo de las épocas mostraron una clara mejora en el rendimiento del modelo, pero también revelaron problemas de sobreajuste, evidenciados por las oscilaciones en la pérdida y exactitud de validación. A pesar de estos desafíos, el modelo muestra un desempeño prometedor para la detección de situaciones delictivas en tiempo real.

VII. Lineas futuras de investigación

En futuras investigaciones, se podrían explorar diversas técnicas para mejorar la generalización y robustez del modelo. Una de las áreas clave es la implementación de técnicas de regularización, como el dropout adicional y la regularización, para reducir el riesgo de sobreajuste y mejorar el rendimiento en datos no vistos. Además, se podrían utilizar métodos de aumento de datos para mejorar el conjunto de entrenamiento con variaciones, mejorando la capacidad del modelo para manejar diferentes condiciones y escenarios.
Otro aspecto importante es la optimización de los hiperparámetros del modelo utilizando técnicas como la búsqueda en malla (grid search) o la búsqueda aleatoria (random search) para encontrar la configuración óptima de la red neuronal.
Es fundamental también considerar la implementación de sistemas de evaluación continua y retroalimentación en tiempo real para ajustar y mejorar el modelo continuamente basándose en su rendimiento en el mundo real.Por otro lado, la colaboración con expertos en seguridad y análisis de video podría proporcionar conocimientos valiosos para ajustar el modelo a las necesidades específicas de las aplicaciones de vigilancia y seguridad.

VIII. Bibliografia

Estadísticas criminales de la República Argentina. (2019, noviembre 14). Argentina.gob.ar. https://www.argentina.gob.ar/seguridad/estadisticascriminales

Informes sobre estadísticas criminales. (2019, noviembre 14). Argentina.gob.ar. https://www.argentina.gob.ar/seguridad/estadisticascriminales/informes

Nacion, L. A. (2021, marzo 8). La División 911, donde están los oídos y los ojos que guían la intervención de la policía en la Ciudad. LA NACION.

https://www.lanacion.com.ar/seguridad/la-division-911-donde-estan-los-oidos-y-los-ojos-que-guian-la-intervencion-de-la-policia-en-la-nid08032021/

¿Qué es la inteligencia artificial o IA? (s/f). Google Cloud. Recuperado el 26 de mayo de 2024, de https://cloud.google.com/learn/what-is-artificial-intelligence?hl=es-419

¿Qué es una API y cómo funciona? (s/f). Redhat.com. Recuperado el 3 de junio de 2024, de https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., & Darrell, T. (2014). Long-term recurrent convolutional networks for visual recognition and description. En arXiv [cs.CV]. http://arxiv.org/abs/1411.4389

IX. CV

Nain David Rossetti.
Estudiante de Ingenieria en Informática.
Desarrollador de sotfware.