

State of the Art: peligros y pizarra del aula inteligente

Ingeniería de Mantenimiento de Computadores y Redes

Propuesta por Ivan

18 de febrero de 2026

Objetivo

Complementar sistemas reactivos basados en *hardware* dedicado con sistemas proactivos basados en **visión artificial** (en adelante, CV por *Computer Vision*).

- **Hardware tradicional:** Sensores unifuncionales (humos, humedad, PIR). Entradas activas: botón, palanca, lógica con enfoque más «síncrono».
- **Enfoque software/IA:** Uso de cámaras CCTV (cámaras de circuito cerrado, usadas para monitoreo de vídeo de seguridad) existentes + procesamiento en el borde (*Edge Computing*) o «nube».

¿Qué riesgos hay en un aula?

- **Seguridad:**

- Violencia física (peleas, acoso).
- Presencia de armas o bultos de objetos contundentes.
- Intrusos en horarios restringidos.

- **Integridad del entorno:**

- Incendios (análisis espectral y de movimiento).
- Inundaciones (análisis de textura en suelos).
- Bloqueo de salidas de emergencia (análisis espacial).
- Terremotos y actividad sísmica (temblores).
- Desprendimientos de objetos así como lanzamiento.
- Vandalismo (pintadas, comportamiento canalla...).

- **Salud y bienestar:**

- Caídas y desmayos.
- Detección de pánico o aglomeraciones.
- Contaminación acústica (ruido estridente, chillidos...).

- **Rápido y preciso.**
- Sensibilidad a oclusiones.
- **Navaja suiza:** Detección, segmentación, seguimiento, poses...
- Código abierto (OSS, *Open Source Software*).
<https://github.com/ultralytics/ultralytics>.
- Usado también para **atención de alumnos:**
<https://www.mdpi.com/1424-8220/25/22/6972>
- En **tiempo real** a través de cámaras IP a servidor local.
- Usado para detectar robo, violencia, vandalismo, fuego...
- Plausible en *hardware* limitado si se relajan restricciones.

- Rápida sturación con multitud.
- Eficiente al poder ejecutarse en **móviles** (pensado para sin GPU dedicada).
- Mejor captura de la profundidad.
- OSS.
- **Navaja suiza.**
- ¿Optimizado para Raspberry Pi? («*¿Gemini hablando bien de Google?*»).
- <https://github.com/google-ai-edge/mediapipe>.

CNN (*Convolutional Neural Networks*)

- **Extracción de características:** Aprende automáticamente a detectar bordes, texturas y formas sin programación manual.
- **Jerarquía Visual:**

Bordes \rightarrow Formas simples \rightarrow Objetos complejos

- **Invarianza:** Reconoce el objeto independientemente de su posición (x, y) en la imagen.
- El **kernel** (filtro) recorre la matriz de píxeles calculando el producto escalar para detectar patrones.
- Sirve para audio también.

- La violencia es un **acto temporal** → secuencia memoria.
- **YOLO** por si solo le cuesta → YOLO-Pose. Precisamos de una TPU si hacer en el *edge*.
- **YOLO-Pose/Google MediaPipe+LSTM** (*Long Short Term Memory*).
 - Clasificador temporal.
 - *Hardware* modesto → **privacidad**.
 - **Oclusiones y OSS**.
- **SlowFast**
 - Preciso. De Facebook. OSS.
 - Computación pesada.
 - Dos modos de funcionamiento: *slow* (¿quién es?), *fast* (movimiento)
- Ejemplo detección de violencia y vandalismo
<https://github.com/Ab-code00/SurakshaAI---Real-Time-Suspicious-Activity-Detection-System>

Detección de intrusos

- **Objetivo:** saltar alarma si viene alguien no deseado en determinado horario.
- Cuádruple reto: **reconocimiento facial** + **horarios** + **lista blanca** + **privacidad**.
- Variantes:
 - ① Naive: hay personas \wedge fuera de horario \rightarrow alarma.
 - ② Reconocer caras \rightarrow ¿Privacidad? ¿Niños? ¿Ejercer derecho?. Necesidad de una DB actualizada \rightarrow lío legal.
 - ③ ¿Quién decide entrar y hasta dónde puede? (Absurdo no dejar entrar visita, familiares, servicios públicos, nuevo personal \notin DB) \rightarrow ¿1984?
- Merodeos: https://github.com/nwojke/deep_sort.
- <https://github.com/hectorpadin1/Network-Intrusion-Detection-System>.
- Otro enfoque, analizar la red:
<https://yardenfalik.github.io/IDS-Project/>. (Esto sale en la teoría de Sistemas Distribuidos).

Fuego

- **no tiene forma fija, «parpadea», brilla, reflja.**
- Puede producirse en **sitios ocultos** donde la cámara no vea.
- Versiones *custom* de YOLO.

⇒ Detección por IA como complementario para anticipar.

Agua

- **No se busca encontrar líquido** sino otras características.
- **Reflejos** (da igual si es sucia o transparente).
- Detección del suelo mojado píxel a píxel (U-Net, DeepLabV3+).
- Causas naturales → otros asuntos, meteo...

⇒ Interés en prevenir (tuberías rotas, fraturas...). Conocer aula concreta.

Caídas y desmayos

- Heurísticas sobre la pose del esqueleto.
 - ① Cambio en la **relación de aspecto**.
 - ② **Velocidad** de descenso (¿se sienta o se cae?).
 - ③ **Inactividad** (¿tropiezo o desmayo?).
- Otras situaciones.
 - Deporte → *fine tuning*, más movimiento.
 - *Trolling* → al menos, 18 añitos tiene la criatura.
- ¿Oclusión? Inferencia en base a lo visible.
- Tecnologías a gastar: YOLO, YOLO-pose, BoT-SORT y ByteTrack.
- **OpenPifPaf**:
https://github.com/cwlroda/falldetection_openpifpaf.
- **AlphaPose**:
<https://github.com/GajuuzZ/Human-Falling-Detect-Tracks>.
- **Multicam**:
<https://github.com/taufeeque9/HumanFallDetection>.

Detectar objetos (mochilas, mesas) que permanecen en vías de evacuación o que impidan una salida normal. **Diferenciar un bloqueo real** de un tránsito momentáneo.

Algoritmo: IoU (*Intersection over Union*) Temporal

Sea P_{salida} el polígono de la salida, B_{bulto} la caja del objeto y T el tiempo transcurrido desde que se detecta la obstrucción:

$$\text{Solape} = \frac{\text{Área}(P_{salida} \cap B_{bulto})}{\text{Área}(P_{salida})}$$

$$(\text{Solape} > 0,3) \wedge (T > 60s) \rightarrow \text{Alerta}$$

Audio: gritos, auxilio, ruido estridente...

- **Clasificador de sonido de Tensorflow** YAMNet <https://www.tensorflow.org/hub/tutorials/yamnet?hl=es-419>.
- Micrófono I2S: <https://opencircuit.es/producto/fermion-i2s-mems-microphone-breakout>.

Procesamiento (YAMNet):

- 1 **Entrada:** Micrófono I2S (MEMS) en la Raspberry Pi.
- 2 **Preprocesar:** Transformada de Fourier → Mel-Spectrogram (Redes de los Computadores).
- 3 **Clasificación:** CNN ligera (MobileNet) entrenada en *AudioSet*.

Eventos Críticos

- **Gritos:** Agresión o Pánico.
- **Destrucción:** Vandalismo.
- **Palabras clave:** «¡Socorro!», «¡Ayuda!» (TinyML).

Ejemplo: <https://github.com/Varun-310/SCREAM>

Privacidad garantizada. Almacenar ¿qué eventos peligrosos (*Edge*).

No todo se detecta mejor con IA

- **Fuegos, inundaciones.**
- **Sismos** (ESP32 + MPU-6050): <https://github.com/serdaraltin/earthquake-warning-system>.
- **SmartUniversity** IoT de la UA que captura diferentes parámetros de afluencia y temporal.

- **Fuego:** <https://github.com/MuhammadMoinFaisal/FireDetectionYOLov8>.
- **Armas:**
<https://www.nature.com/articles/s41598-025-07782-0>,
<https://github.com/swatified/Weapon-Detection-System>.
- **Violencia:** <https://www.youtube.com/watch?v=a2xWqkFDYuU>,
<https://www.youtube.com/watch?v=D4mjEBgAXPU>,
<https://www.youtube.com/watch?v=qeFrjFa5Rxc>
- **Caídas:** <https://www.youtube.com/watch?v=vEtsmg7-fWs>
- **Ataque de pánico, aglomeraciones:**
<https://github.com/moego0/panic-attack-detector>,
https://github.com/jinay-k-jain/Panic_detector_CCTV.
- **Vandalismo:**
<https://ieeexplore.ieee.org/document/10895139>.

¿Qué tenemos a disposición?

- 600€ de presupuesto en nuevo *hardware*.
- Electrónica de otros años.
 - Placas de Arduino.
 - Raspberry Pi.
 - ESP32.
- Ordenadores del aula.
 - GPU: Nvidia GTX 1660.

¡Relajar restricciones!

- 1 Una cámara → un propósito.
- 2 Reducir FPS → no tiempo real.
- 3 Procesamiento centralizado (SPOF). ¿vigilancia urbanización?
- 4 Combiando estrategias (o aplicando divide y vencerás para el reparto de responsabilidad): *edge* detecta que hay personas, *server* averigua quienes son.

Ejecución en *hardware* limitado

No todo el *hardware* puede procesar las millones de operaciones matriciales de una CNN. Clasificación por niveles:

Dispositivo	Capacidad CNN	Tecnología	Uso
Raspberry Pi 5	Alta	Python, TensorFlow, PyTorch	Corre YOLO/MediaPipe para análisis complejo.
ESP32-S3	Limitada (TinyML)	TFLite Micro, C++	Sensor auxiliar. Solo clasificación simple (Person/No-Person).
Arduino Uno	Nula	-	Solo control de relés/luces. No procesa imagen.
Arduino Portenta	Media	OpenMV, Edge Impulse	Similar al ESP32 pero grado industrial.

Es importante la **cuantización**. Para correr en ESP32, los modelos se comprimen de float32 a int8, reduciendo el tamaño del modelo de 50MB a < 500KB.

¡Optimizar!

La pizarra es un recurso muy utilizado por el profesorado para ilustrar ideas, resolución de ejercicios y conceptos. La información de la pizarra al cabo de una sesión no está disponible. Lo cual dificulta el seguimiento de la clase para aquellos más despistados, sentados más lejos, lentitud, mala visibilidad o cualquier otra dificultad en tanto que interfiera...

- Facilitar la **grabación y streaming** de clases.
- **Seguimiento de movimiento** del profesor para el punto anterior.
- Guardar un **historial** del trazado de la pizarra.
- **Transcripciones de voz** de la lección.
- **Accesibilidad** (intérpretes, traductores...).

- ❶ Durante la pandemia del **covid**:
 - Las reuniones a través de videollamadas.
 - Poca preparación, mal audio, aplicaciones propietarias (no FOSS).
 - Compatir pantalla, pizarra con mala visibilidad.
 - Profesores pueden emplear otras para difundir conocimiento:
https://youtu.be/_KNxz2YyhgA.
- ❷ El seguimiento del movimiento del profesor puede hacerse de **forma automática** desde la configuración de algunas cámaras.
 - https://youtu.be/ZA-tUyM_y7s&t=120.
 - Aunque siempre se deba contemplar una opción manual.
- ❸ **Guardar algo escrito de la pizarra.**
 - Fotos a la pizarra → mala calidad al detalle \wedge rostros no deseados.
 - Materiales docentes, almacenamiento en la «nube»...
- ❹ A partir de los vídeos **es posible extraer una transcripción**.
- ❺ Garantizar **accesibilidad** para personas ciegas (braille), sordas de lo acontecido en la sección (generar intérpretes).

El sistema debe:

- 1 Estar **a merced del profesor** como una herramienta. Y es este quién decida cuando hacerla servir.
- 2 Garantizar la **privacidad** (GDPR).

El sistema utiliza una cámara cenital o frontal para digitalizar la pizarra analógica automáticamente.

Disparadores (Triggers)

La captura se activa mediante lógica de software:

- **Botón Físico:** Ad-hoc por el profesor al terminar una explicación.
- **App Estudiante:** Petición individual vía WebSocket.
- **Modo "Ad Quorum":** Se dispara solo si el $X\%$ de los alumnos lo solicita en 30 segundos.

Detección Automática

Mediante **Background Subtraction**, el software detecta cuando la pizarra está "llenaz el profesor se aparta, disparando la foto automáticamente.

Uno de los retos del software en aulas es la privacidad de los alumnos y el profesor.

Pipeline de Anonimización:

- ➊ **Detección de Rostros:** Uso de modelos ligeros (ej. MTCNN o Haar Cascades) para localizar caras en la imagen de la pizarra.
- ➋ **Aplicación de Blur (Difuminado):** Aplicación de filtro Gaussiano sobre las regiones de interés (ROIs) detectadas.
- ➌ **Rectificación de Perspectiva:** Algoritmo de homografía para transformar la foto lateral en una imagen plana ".escaneada".
- ➍ **Mejora de Contraste:** Binarización adaptativa para convertir trazos de tiza/rotulador en digital nítido.

Resumen de la Propuesta

- La infraestructura se basa en **software** sobre hardware genérico (cámaras), reduciendo costes de instalación.
- La IA permite una detección de peligros más rápida y contextual (ve el fuego, no solo huele el humo).
- La digitalización de la pizarra respeta la privacidad (blurring) y democratiza el apunte (quorum).

¿Preguntas?