1. TÍTULO

**SISTEMA DE MONITOREO EN CONDUCTORES PARA DETECTAR LA SOMNOLENCIA O CANSANCIO**

1. CARÁTULA
2. INDICE
3. RESUMEN

El tema que se aborda en este proyecto es la detección de la somnolencia durante la conducción, que han sido descritas como causantes de accidentes de tránsito dejando alta siniestralidad en término de pasajeros muertos, heridos y pérdidas materiales.

En este trabajo se ha desarrollado un prototipo de un sistema de control del estado de somnolencia en conductores de vehículos, el sistema utiliza la visión por computador.

El sistema se desarrolló utilizando python para entrenar el modelo para la detección facial, basándose específicamente en el estado de los ojos y java para implementar el modelo en android con el framework tflite, esto mediante técnicas de visión artificial, y las redes neuronales artificiales. El sistema al detectar somnolencia alerta al conductor a través de un sistema de alarmas que le permite estar despierto y así evitar accidentes.

1. ANTECEDENTES

El cansancio y la somnolencia del conductor frente al volante, son factores importantes dentro de las causas de siniestros automovilísticos. Durante los últimos años se han estado planteando distintos tipos de soluciones para su prevención.

1. PROBLEMÁTICA

Un estudio realizado por estudiantes de la cátedra de medicina legal de la Universidad Mayor de San Andrés (UMSA) determinó que una de las causas que ocasiona más accidentes carreteros en Bolivia son los clínicamente denominados “sueños blancos”, que consiste en que debido al agotamiento y falta de descanso, algunos conductores de flotas que prestan el servicio de transporte interprovincial e interdepartamental, llegan a conciliar el sueño en medio del trayecto que realizan, perdiendo la atención en el camino y convirtiéndolos en un riesgo latente de un siniestro.

De 6.800 accidentes reportados en carreteras durante 2020, más de 4.000 casos, casi un 83%, son atribuidos a esta problemática, aunque es difícil determinar certeramente esta realidad, debido a que ninguna norma en el país regula este aspecto.

No existe ninguna normativa o sistema en el país que regule o controle de alguna manera el sueño y la fatiga laboral de los conductores. En el caso de algún control de tránsito solo se observa que el conductor no se encuentre en estado de ebriedad, no exceda los límites de velocidad y que cuente con licencia de conducir, pero no existe ningún mecanismo que verifique la cantidad de horas que hubiese conducido el conductor, así también que haya cumplido las horas de descanso correspondientes para evitar problemas frente al volante.

1. ABORDAJE O PROPUESTA DE SOLUCIÓN

Cuando el conductor entra en estado de fatiga, suele haber una serie de reacciones fisiológicas, como bostezar y cerrar los ojos. Se adquieren múltiples estados de los ojos y la boca, y el estado de fatiga del conductor se evalúa calculando el grado de cierre de ojos PERCLOS EAR y el grado de apertura de boca POM.

La propuesta es un sistema que detecte el rostro y dentro de este los ojos y boca, para determinar si el conductor está somnoliento se realizará un conteo de parpadeos y los estados del ojo que son abierto, medio abierto y cerrado. Si el sistema detecta que los ojos están en los estados medio abierto o cerrado por un determinado tiempo el sistema lanzará una alarma al conductor, otra forma para determinar la somnolencia serán los bostezos.

1. OBJETIVO GENERAL

Detectar la somnolencia, con una precisión de 90% o superior.

1. OBJETIVOS ESPECÍFICOS

**-**Monitorear la somnolencia y fatiga del conductor para poder alertar en caso de quedarse dormido frente al volante, mediante una cámara.

-Desarrollar un programa basado en un lenguaje de programación, para el análisis del comportamiento de los ojos del conductor, que permita alertar en caso de somnolencia.

-Entrenar con el modelo YOLOv4-tiny y el dataset WIDER FACE que contiene (32203 imágenes y 393703 caras marcadas).

1. FUNDAMENTOS TEÓRICOS CONSIDERADOS EN EL TRABAJO
   1. Ámbito de la inteligencia artificial, técnicas, algoritmos, modelos base, entre otros

En este proyecto se implementó inteligencia artificial enfocado a su campo de la visión por computador, se utilizaron métodos y técnicas de aprendizaje profundo (Deep Learning) a través de redes Neuronales Convolucionales (CNN).

Para el entrenamiento de detección de objetos se utilizó el modelo YoloV4-tiny, utilizaremos los pesos entrenados del modelo para una detección en tiempo real con Deep Sort (Seguimiento simple en línea y en tiempo real con una métrica de asociación profunda) para crear un rastreador de objetos en tiempo real de alta precisión.

* 1. Ámbito al que se aplicó la inteligencia artificial

1. DATASET
   1. Descripción y preprocesamiento realizado (Evidencia del antes y del después)

Se utilizó el dataset WIDER FACE es un conjunto de datos de referencia de detección de rostros . Este dataset contiene 32.203 imágenes y 393.703 etiquetas . El conjunto de datos de WIDER FACE está organizado en base a 61 clases de eventos. Para cada clase de evento, seleccionamos aleatoriamente 40% / 10% / 50% de datos como conjuntos de entrenamiento, validación y prueba

En la etapa de preprocesamiento se convirtieron los datos de anotación(.xml) del dataset en etiquetas yolo(.txt).

* 1. Conjunto de entrenamiento, evaluación y validación
  2. Técnicas, criterio y/o métodos aplicados para la conformación de los conjuntos de datos de entrenamiento, prueba y validación.
  3. Información adicional que considere importante incluir.

1. METRICAS DE RENDIMIENTO (sobre los resultados finales)
   1. Matriz de confusión
   2. Exactitud
   3. Precisión
   4. Sensibilidad
   5. Especificidad
   6. True Positive Rate
   7. False Positive Rate
   8. Curvas ROC
   9. F-Score
   10. BLEU, entre otros específicos de cada trabajo realizado.
2. DESCRIPCIÓN DEL TRABAJO REALIZADO EN EL MODELO (Materiales y métodos)
   1. Herramientas utilizadas (Software y Hardware)

Para entrenar y probar el modelo se utilizó Google Colab como plataforma, Python como lenguaje de programación y librerías como opencv, numpy, y otras.

* 1. Modelos utilizados o arquitectura y descripción del modelo generado (esquemas y/o gráficas).

Se utilizó el modelo pre entrenado YoloV4-tiny.

Para la detección de objetos en tiempo real , YOLOv4-tiny es la mejor opción en comparación a otros modelos, ya que un tiempo de inferencia más rápido es más importante que la precisión o exactitud cuando se trabaja con un entorno de detección de objetos en tiempo real.

YOLOv4-tiny es la versión comprimida de YOLOv4. Se propone basado en YOLOv4 para simplificar la estructura de la red y reducir los parámetros para que sea factible para el desarrollo en dispositivos móviles e integrados.

Podemos usar YOLOv4-tiny para un entrenamiento mucho más rápido y una detección mucho más rápida. Ha sido entrenado a partir de 29 capas convolucionales pre-entrenadas.

El FPS (cuadros por segundo) en YOLOv4-tiny alcanza 39 FPS / 25 ms de latencia.

* 1. Valores de hiper parámetros aplicados

batch=64

subdivisions=16

width=416 height=416

max batches = 30000

steps=20000,25000

filters=18 (última capa convolucional)

learning\_rate=0.00261 (velocidad de aprendizaje)

* 1. Técnicas de depuración aplicadas
  2. Otros aspectos importantes
  3. Resultados obtenidos

1. SOFTWARE DE APLICACIÓN DESARROLLADO (Materiales y métodos)
   1. Arquitectura software desarrollado

Para el software de aplicación se utilizó el framework tflite.

Convertimos los pesos de modelo Yolo (.weights) a una representación de TensorFlow (.pb) y finalmente en un archivo Tensor Flow Lite (.flite) para usar en el dispositivo móvil.

Tensor Flow Lite se diseñó para facilitar el aprendizaje automático en dispositivos que están "en el perímetro" de la red, y así, no tener que enviar o recibir datos mediante un servidor. Para los desarrolladores, el uso del aprendizaje automático en dispositivos puede mejorar los siguientes aspectos:

Latencia: no hay ida y vuelta con un servidor.

Privacidad: no es necesario que los datos salgan del dispositivo.

Conectividad: no se requiere una conexión a Internet.

Consumo de energía: las conexiones de red requieren mucha energía.

La aplicación utiliza el modelo entrenado yolov4-tiny con el framework tflite para la detección facial, este modelo retorna un conjunto de rostros ya delimitados, posteriormente se implementó la librería dlib la cual específicamente para este trabajo requiere rostros ya delimitados, para obtener los 68 puntos de referencias faciales de los cuales solo se usaron algunos, ya obtenidos los puntos referencias faciales aplicamos la función EAR(relación respecto al ojo) y Relación respecto a la boca, definimos un umbral para la detección del parpadeo y el bostezo.

Todo el funcionamiento de la aplicación está realizado en tiempo real.

* 1. Esquema y descripción de componentes del software desarrollados
  2. Esquema y descripción de componentes de hardware requeridos
  3. Esquema y descripción de modelos o componentes inteligentes desarrollados
  4. Especificaciones técnicas.
  5. Lenguajes de programación, frameworks, entre otros.

1. RESULTADOS FINALES
2. CONCLUSIONES
3. RECOMENDACIONES
4. REFERENCIAS BIBLIOGRAFICAS (Harvard o APA)
5. ANEXOS