



1. Datos Generales de la asignatura

Nombre de la asignatura:	Visión por Computadora y Machine Learning
Clave de la asignatura:	INM-2508
SATCA¹:	2-4-6
Carreras:	Ingeniería en Sistemas Computacionales, Ingeniería Industrial, Ingeniería en Gestión Empresarial, Licenciatura en Administración, Ingeniería en Administración.

2. Presentación

Caracterización de la asignatura
<p>Esta asignatura aporta al perfil del Ingeniero en Sistemas Computacionales las competencias profesionales:</p> <ul style="list-style-type: none">• Aplicar conocimientos matemáticos y estadísticos en la solución de problemas de visión por computadora y machine learning con un enfoque interdisciplinario.• Implementar el procesamiento de imágenes a través de manipulación, segmentación, filtrado, transformaciones y extracción de características.• Analizar, modelar, desarrollar, implementar sistemas de visión por computadora por lo que debe tener una comprensión de los conceptos y técnicas de visión por computadora, como detección de objetos, reconocimiento facial, seguimiento de objetos y segmentación semántica.• Abordar problemas complejos y tener habilidades analíticas para analizar, diseñar e implementar soluciones efectivas en el campo de visión por computadora y machine learning.• La importancia de esta asignatura, es que permite al estudiante crear e implementar modelos de inteligencia artificial en la nube capaces de resolver problemas complejos en diversos sectores, con capacidad para analizar datos visuales, potencial para la automatización y eficiencia, para después consumir estos modelos a través de servicios en aplicaciones con tecnologías en las que hoy nos conectamos desde teléfonos móviles, tabletas, eBooks, netbooks, computadoras y otra gama de dispositivos.

¹ Sistema de Asignación y Transferencia de Créditos Académicos



Intención didáctica

La asignatura debe ser práctica y capaz de desarrollar en el estudiante la habilidad para desarrollar habilidades en el análisis y procesamiento de imágenes, aplicar técnicas de reconocimiento de patrones y aprendizaje automático, diseño y entrenamiento de redes neuronales convolucionales. Así también fomentar la creación de modelos y su despliegue en la nube con el lenguaje Python y frameworks y librerías de Inteligencia Artificial.

- En la unidad I. El estudiante será capaz de adquirir conocimiento y técnicas para cargar, limpiar, transformar y procesar datos; así como explorar, identificar patrones, tendencias y relaciones entre variables y obtener información y conocimientos a partir de éstos. utilizando herramientas disponibles en Python, implica el uso de librerías como Pandas, NumPy y Matplotlib para realizar visualizaciones, cálculos estadísticos y generar gráficos descriptivos.
- En la unidad II. El estudiante será capaz de adquirir conocimientos sólidos sobre los principios y conceptos clave del Deep Learning incluyendo las redes neuronales artificiales, las capas, las funciones de activación y el proceso de aprendizaje. Aplicar el Deep Learning en problemas de clasificación y reconocimiento, como la clasificación de imágenes, el reconocimiento de voz, el procesamiento de texto o la detección de objetos. Desarrollar una perspectiva crítica sobre el Deep Learning, comprendiendo sus limitaciones, desafíos y posibles sesgos. Esto implica aprender a evaluar la calidad y robustez de los modelos, a interpretar sus resultados y a tomar decisiones informadas sobre su uso en diferentes contextos y aplicaciones.
- En la unidad III. El estudiante adquirirá conocimientos sólidos sobre la arquitectura y el funcionamiento de las redes neuronales convolucionales, incluyendo conceptos como las capas convolucionales, las capas de agrupación (pooling), las funciones de activación y las capas completamente conectadas. También, adquirirá experiencia práctica en la implementación de redes neuronales convolucionales utilizando frameworks y bibliotecas populares como TensorFlow y Keras. Esto implica aprender a diseñar, entrenar y evaluar modelos de redes neuronales convolucionales, así como a ajustar sus hiperparámetros y utilizar técnicas de regularización y optimización. Desarrollar una perspectiva crítica sobre las redes neuronales convolucionales, comprendiendo sus limitaciones, desafíos y posibles sesgos.
- En la unidad IV. Adquirirá conocimientos sobre la arquitectura y el funcionamiento de las redes neuronales recurrentes, incluyendo conceptos como las células de memoria recurrente (por ejemplo, las LSTM o GRU), el mecanismo de retroalimentación temporal y la capacidad de procesar secuencias de datos. Utilizar los conocimientos adquiridos para aplicar redes neuronales recurrentes en problemas de procesamiento del lenguaje natural, como la clasificación de texto, la generación de texto, la traducción automática o el modelado de lenguaje. Esto implica aprender a procesar y representar secuencias de texto, seleccionar arquitecturas de redes neuronales recurrentes adecuadas y utilizar técnicas como la atención y la transferencia de aprendizaje.



3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	Participantes	Observaciones
Agosto-octubre 2024	Instituto Tecnológico de: Zitácuaro Huetamo Morelia Jiquilpan Lázaro Cárdenas La Piedad Uruapan	

4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
Aplicar eficazmente técnicas y algoritmos de visión por computadora y machine learning para analizar, interpretar y extraer información significativa de imágenes y datos visuales, con el fin de resolver problemas complejos en diversos dominios, como reconocimiento de objetos, detección de patrones, segmentación, clasificación, seguimiento y generación automática de contenido visual.
Además, implica la capacidad de abordar problemas complejos, adaptar y combinar diferentes técnicas y algoritmos, seleccionar y preprocesar adecuadamente los datos, evaluar y optimizar los modelos desarrollados, y comunicar los resultados obtenidos de manera clara y efectiva.

5. Competencias previas

Esta asignatura, es la aplicación práctica del conocimiento matemático y estadístico, de programación orientada a objetos, lógica y funcional, de fundamentos del machine learning y procesamiento de imágenes a través de los métodos y técnicas adecuados para la implementación de modelos de inteligencia artificial y su despliegue en la nube para después consumirlo como un servicio.
Esta asignatura tiene relación previa con las signaturas:
<ul style="list-style-type: none">• Programación orientada a objetos.• Inteligencia artificial.• Programación web.• Arquitectura orientada a servicios• Análisis y Exploración de Datos• Ciencia de Datos



Considerando las asignaturas anteriormente mencionadas se consideran las siguientes competencias específicas:

- Aplicar un lenguaje orientado a objetos para la solución de problemas.
- Conocer los principios y el desarrollo de la Inteligencia Artificial, identificando sus aplicaciones (robótica, visión computacional, lógica difusa, redes neuronales y procesamiento de lenguaje natural) para emplearlas en el diseño e implementación de sistemas inteligentes que faciliten las tareas del ser humano. Modelar casos de uso acorde a los requerimientos del proyecto.
- Desarrollar aplicaciones web que involucre lenguajes de marcas, de presentación, del lado del cliente, del lado del servidor, con la integración de servicios web.
- Documentar el proyecto.
- Tomar decisiones con base en los elementos teórico-práctico adquiridos que permitan optimizar costos en soluciones informáticas bajo ambiente Web.



6. Temario

No.	Temas	Subtemas
1	Procesamiento y análisis de Datos.	<ul style="list-style-type: none">1.1. Matrices.<ul style="list-style-type: none">1.1.1. Arreglos unidimensionales y multidimensionales1.1.2. Creación de arreglos1.1.3. Operaciones aritméticas con arreglos1.1.4. Uso de índices en arreglos1.1.5. Programación orientada a los arreglos1.1.6. Funciones universales1.2. Graficación de información.<ul style="list-style-type: none">1.2.1. Figuras y subplots1.2.2. Gestión de gráficas1.2.3. Guardado de gráficas en archivos1.3. Manipulación y tratamiento de datos.<ul style="list-style-type: none">1.3.1. Series y matrices en dos dimensiones1.3.2. Indizado, selección y filtrado1.3.3. Aritmética y alineamiento de datos1.3.4. Manipulación algebraica de dato
2	Fundamentos de Deep Learning	<ul style="list-style-type: none">2.1. ¿Qué es Deep Learning?<ul style="list-style-type: none">2.1.1. ¿Porqué Deep Learning?2.1.2. ¿Porque ahora?2.2. Redes neuronales.<ul style="list-style-type: none">2.2.1. Anatomía de la red neuronal.2.2.2. Componentes matemáticos de redes neuronales.2.2.3. Librerías para gestión de redes neuronales.2.3. Perceptrones multicapa – MLP.2.4. Aprendizaje supervisado.2.5. Aprendizaje no supervisado2.6. Algoritmos de aprendizaje<ul style="list-style-type: none">2.6.1. Descenso por gradiente.2.6.2. Backpropagation.2.6.3. Hiperparámetros.2.6.4. Métricas de desempeño.2.7. Creación de modelos de red neuronal.2.8. Funciones de activación para redes neuronales multicapa2.9. Despliegue de modelos



3	Redes neuronales convolucionales	<ul style="list-style-type: none">3.1. ¿Qué es la convolución?3.2. Convolución 2D.3.3. Convolución 3D.3.4. Filtros de procesamiento de imágenes.3.5. Componentes de una red neuronal convolucional3.6. Retropropagación a través de la capa convolucional3.7. Retropropagación a través de las capas de agrupación3.8. Técnicas de regularización<ul style="list-style-type: none">3.8.1. Dropout3.8.2. Data augmentation3.9. Arquitecturas de redes neuronales convolucionales3.10. Transferencia de conocimiento3.11. Creación de modelos con redes neuronales convolucionales
4	Redes neuronales recurrentes	<ul style="list-style-type: none">4.1. Redes generativas antagónicas GAN.<ul style="list-style-type: none">4.1.1. Autoencoders como predecesores4.1.2. Mecanismo de las redes generativas y discriminativas4.2. Redes neuronales recurrentes RNN.<ul style="list-style-type: none">4.2.1. El mecanismo de las redes recurrentes4.2.2. Tipos de RNN: LSTM y GRU4.2.3. Procesamiento del lenguaje natural – NLP4.2.4. Bolsas de palabras y word embedding4.3. Creación de modelos para aplicaciones de lenguaje natural.



7. Actividades de aprendizaje de los temas

1. Procesamiento y análisis de Datos	
Competencias	Actividades de aprendizaje
<p><i>Específica(s):</i></p> <ul style="list-style-type: none">• Proporcionar habilidades prácticas en programación, resolución de problemas, manipulación de datos numéricos, análisis de datos y optimización del rendimiento.• Adquirir habilidades esenciales en visualización de datos, análisis exploratorio, comunicación efectiva y presentación de resultados.• Tener habilidades en manipulación de datos, preparación de datos y trabajo con datos estructurados. <p><i>Genéricas:</i></p> <ul style="list-style-type: none">• Capacidad de análisis y síntesis.• Capacidad de organizar y planificar.• Habilidad para buscar y analizar información proveniente de fuentes diversas.• Comunicación oral y escrita.	<ul style="list-style-type: none">• Realizar prácticas de carga de imágenes, manipulaciones y transformaciones de datos utilizando NumPy y Pandas, y visualizar resultados utilizando Matplotlib.• Realizar prácticas de limpieza de datos, de normalización, extracción de características y manejo de datos faltantes utilizando Pandas y NumPy.• Implementar algoritmos de clasificación, regresión u otros en Python utilizando estas bibliotecas
2. Fundamentos de Deep Learning	
Competencias	Actividades de aprendizaje
<p><i>Específica(s):</i></p> <ul style="list-style-type: none">• Proporcionar una base sólida para trabajar en proyectos de aprendizaje profundo y aprovechar las oportunidades en campos como la inteligencia artificial, la visión por computadora y el procesamiento de lenguaje natural. <p><i>Genérica(s):</i></p> <ul style="list-style-type: none">• Capacidad de análisis y síntesis.• Capacidad de organizar y planificar.• Comunicación oral y escrita.• Capacidad de aplicar los conocimientos en la práctica.• Toma de decisiones.	<ul style="list-style-type: none">• Desplegar el modelo en la nube y consumirlo desde una aplicación web y/o, móvil.



3.Redes neuronales convolucionales	
Competencias	Actividades de aprendizaje
<p><i>Específica(s):</i></p> <ul style="list-style-type: none">• Desarrollar habilidades y conocimientos específicos en el diseño, implementación y aplicación de redes neuronales convolucionales en el campo del aprendizaje automático y la visión por computadora. <p><i>Genéricas:</i></p> <ul style="list-style-type: none">• Capacidad de análisis y síntesis.• Capacidad de organizar y planificar.• Comunicación oral y escrita.• Habilidad para buscar y analizar información proveniente de fuentes diversas.• Solución de problemas.• Toma de decisiones.• Capacidad crítica y autocrítica.• Trabajo en equipo.• Habilidades interpersonales.• Capacidad de aplicar los conocimientos en la práctica.	<ul style="list-style-type: none">• Explorar arquitecturas CNN más avanzadas, como ResNet, Inception, VGG o DenseNet. Estudiar su estructura y los principios en los que se basan.• Aplicar transferencia de conocimiento. Utilizar redes preentrenadas en conjuntos de datos específicos. Cargar modelos CNN preentrenados, como los entrenados en ImageNet, y ajústalos a problemas de clasificación o detección de objetos en conjuntos propios de datos.
4. Redes neuronales recurrentes	
Competencias	Actividades de aprendizaje
<p><i>Específica(s):</i></p> <ul style="list-style-type: none">• Desarrollar habilidades sólidas en el uso de redes neuronales recurrentes para el procesamiento del lenguaje natural, la traducción automática, la generación de texto.	<ul style="list-style-type: none">• Experimentar con diferentes arquitecturas. Explorar diferentes arquitecturas de redes neuronales recurrentes más avanzadas, como las RNN bidireccionales o las RNN apiladas• Aplicar RNN en tareas de procesamiento de lenguaje natural. Utiliza RNN para resolver problemas de procesamiento del lenguaje natural, como la generación de texto o la traducción automática.



Genéricas:

- Capacidad de análisis y síntesis.
- Capacidad de organizar y planificar.
- Comunicación oral y escrita.
- Habilidad para buscar y analizar información proveniente de fuentes diversas.
- Solución de problemas.
- Toma de decisiones.
- Capacidad crítica y autocrítica.
- Trabajo en equipo.
- Habilidades interpersonales.
- Capacidad de aplicar los conocimientos en la práctica.

8. Práctica(s)

Unidad I

Práctica 1. Análisis exploratorio de datos de ventas mensuales.

1. Utiliza Pandas para cargar un conjunto de datos que contenga información mensual de ventas.
2. Utiliza Pandas y NumPy para limpiar y preparar los datos, eliminando filas o columnas innecesarias y tratando los valores faltantes si los hay.
3. Utiliza NumPy para calcular estadísticas descriptivas de las ventas, como el total de ventas, promedio mensual, máximo y mínimo.
4. Utiliza Matplotlib para crear gráficos como un gráfico de líneas que muestre la tendencia de las ventas a lo largo del tiempo y un gráfico de barras que muestre las ventas mensuales.
5. Agrega etiquetas, títulos y leyendas a los gráficos para mejorar la presentación y la comprensión de los datos.

Práctica 2. Preprocesamiento de datos de imágenes para clasificación.

1. Utiliza Pandas para cargar un conjunto de datos que contenga imágenes y sus respectivas etiquetas de clasificación.
2. Utiliza NumPy para cargar y procesar las imágenes, convirtiéndolas en matrices numéricas y realizando operaciones de normalización si es necesario.
3. Utiliza Pandas para realizar transformaciones en las etiquetas, como codificarlas en valores numéricos si son categóricas.
4. Utiliza Matplotlib para visualizar algunas de las imágenes antes y después del preprocesamiento para asegurarte de que los datos estén correctamente cargados y listos para su uso.
5. Utiliza las funciones de división de datos de Pandas y NumPy para dividir los datos en conjuntos de entrenamiento y prueba antes de aplicar modelos de clasificación.



Práctica 3. Análisis de series de tiempo de datos climáticos.

1. Utiliza Pandas para cargar un conjunto de datos que contenga registros climáticos a lo largo del tiempo.
2. Utiliza Pandas para realizar cualquier manipulación necesaria en los datos, como cambiar el formato de fechas o filtrar datos irrelevantes.
3. Utiliza NumPy para realizar cálculos en los datos, como el cálculo de promedios móviles o diferenciaciones.
4. Utiliza Matplotlib para crear gráficos que muestren patrones en los datos climáticos, como gráficos de líneas para la evolución de la temperatura a lo largo del tiempo.
5. Agrega elementos visuales a los gráficos, como sombreado para resaltar intervalos de tiempo o líneas de referencia para mostrar promedios históricos.

Práctica 4. Detección y reconocimiento de placas en tiempo real.

1. Instalación de bibliotecas

Paso 1: Asegúrate de tener Python instalado en tu sistema.

Paso 2: Instala las bibliotecas necesarias, como OpenCV, NumPy y Tesseract OCR. Puedes usar pip para instalarlos.

2 Preparación del modelo y los archivos de configuración

Paso 1: Descarga el modelo pre-entrenado para la detección de placas de vehículos, como el modelo YOLOv3, y los archivos de configuración correspondientes.

Paso 2: Configura los archivos de configuración para establecer los parámetros adecuados, como el umbral de confianza y los nombres de las clases.

3 Configuración de la cámara

Paso 1: Conecta la cámara al sistema o utiliza una fuente de video en tiempo real.

Paso 2: Configura los parámetros de la cámara, como la resolución y la velocidad de cuadros, si es necesario.

4 Detección de placas en tiempo real

Paso 1: Carga el modelo pre-entrenado y los archivos de configuración en tu aplicación de Python utilizando OpenCV.

Paso 2: Inicializa el objeto de detección de objetos y configura los parámetros necesarios.

Paso 3: Captura el video en tiempo real y, en cada cuadro, realiza la detección de objetos utilizando el modelo cargado.

Paso 4: Filtra las detecciones para obtener solo las detecciones de placas de vehículos.

Paso 5: Dibuja un cuadro delimitador alrededor de cada placa detectada en el cuadro de video.

5 Reconocimiento de texto en las placas detectadas

Paso 1: Recorta y preprocesa cada región de la placa detectada para obtener una imagen de la placa individual.

Paso 2: Aplica OCR (reconocimiento óptico de caracteres) a la imagen de la placa utilizando Tesseract OCR u otra biblioteca de tu elección.

Paso 3: Obtén el texto reconocido y realiza cualquier postprocesamiento necesario, como limpieza de ruido y formato de salida.



6 Mostrar resultados en tiempo real

Paso 1: Muestra el video en tiempo real con los cuadros delimitadores y los textos reconocidos superpuestos.

Paso 2: Actualiza continuamente la visualización en tiempo real para mostrar los resultados de la detección y el reconocimiento.

Unidad II

Practica 1 Reconocimiento facial, utilizando redes neuronales

1. Preparación de datos

Paso 1: Importa las bibliotecas necesarias, como TensorFlow o PyTorch, y carga el conjunto de datos de imágenes faciales etiquetadas.

Paso 2: Preprocesa los datos de imágenes, como el cambio de tamaño, la normalización de los valores de píxeles y la codificación de etiquetas.

Paso 3: Divide el conjunto de datos en conjuntos de entrenamiento y prueba.

2 Construcción de la red neuronal

Paso 1: Diseña y construye una red neuronal para el reconocimiento facial, definiendo la arquitectura de la red neuronal con capas densas y funciones de activación.

Paso 2: Asegúrate de incluir una capa de salida con un número de neuronas igual al número de clases de reconocimiento facial.

3 Entrenamiento del modelo

Paso 1: Compila el modelo con la función de pérdida adecuada y el optimizador, y define las métricas de evaluación.

Paso 2: Entrena el modelo utilizando el conjunto de datos de entrenamiento y ajusta los pesos de la red neuronal mediante el proceso de retropropagación.

4 Evaluación del modelo

Paso 1: Evalúa el modelo utilizando el conjunto de datos de prueba y calcula métricas de rendimiento, como la precisión y la matriz de confusión.

Paso 2: Realiza predicciones con el modelo entrenado en nuevas imágenes faciales y verifica su precisión.

5 Detección y reconocimiento facial en tiempo real

Paso 1: Utiliza bibliotecas y técnicas de visión por computadora, como OpenCV, para capturar video en tiempo real desde una cámara.

Paso 2: Procesa los cuadros de video para detectar rostros utilizando técnicas de detección facial, como el detector de Haar o el detector de puntos de referencia faciales.

Paso 3: Extrae las regiones faciales detectadas y aplica preprocesamiento, como el cambio de tamaño y la normalización de los valores de píxeles.

Paso 4: Utiliza el modelo entrenado para realizar el reconocimiento facial en las regiones faciales y muestra el resultado, como el nombre o la etiqueta asociada a cada rostro reconocido.



Practica 2 Reconocer el sexo de una persona.

1. Preparación de datos

Paso 1: Importa las bibliotecas necesarias, como TensorFlow o PyTorch, y carga el conjunto de datos de imágenes etiquetadas con información de sexo.

Paso 2: Preprocesa los datos de imágenes, como el cambio de tamaño y la normalización de los valores de píxeles.

Paso 3: Codifica las etiquetas de sexo, asignando un valor numérico (por ejemplo, 0 para masculino y 1 para femenino).

Paso 4: Divide el conjunto de datos en conjuntos de entrenamiento y prueba.

2 Construcción de la red neuronal

Paso 1: Diseña y construye una red neuronal para el reconocimiento del sexo, definiendo la arquitectura de la red neuronal con capas densas y funciones de activación.

Paso 2: Asegúrate de incluir una capa de salida con una neurona para clasificar entre los sexos.

3 Entrenamiento del modelo

Paso 1: Compila el modelo con la función de pérdida adecuada y el optimizador, y define las métricas de evaluación.

Paso 2: Entrena el modelo utilizando el conjunto de datos de entrenamiento y ajusta los pesos de la red neuronal mediante el proceso de retropropagación.

4 Evaluación del modelo

Paso 1: Evalúa el modelo utilizando el conjunto de datos de prueba y calcula métricas de rendimiento, como la precisión y la matriz de confusión.

Paso 2: Analiza los resultados para determinar la efectividad del modelo en el reconocimiento del sexo de una persona.

5 Reconocimiento del sexo en imágenes

Paso 1: Utiliza bibliotecas y técnicas de procesamiento de imágenes, como OpenCV, para cargar y preprocesar imágenes de personas.

Paso 2: Utiliza el modelo entrenado para predecir el sexo de una persona en una imagen.

Paso 3: Muestra el resultado del reconocimiento, indicando el sexo predicho para la persona en la imagen.

Practica 3 Análisis de sentimientos en comentarios y clasificarlos como positivos o negativos

1. Preparación de datos

Paso 1: Importa las bibliotecas necesarias, como TensorFlow o PyTorch, y carga el conjunto de datos etiquetado con comentarios y sus respectivas etiquetas de sentimiento (positivo o negativo).

Paso 2: Realiza una limpieza de los datos, eliminando caracteres especiales, números o cualquier información no relevante para el análisis de sentimientos.

Paso 3: Dividir el conjunto de datos en conjuntos de entrenamiento y prueba.

2 Procesamiento de texto

Paso 1: Realiza la tokenización de los comentarios, dividiendo el texto en palabras individuales o unidades de texto más pequeñas.



Paso 2: Crea un vocabulario, asignando un índice numérico a cada palabra única presente en los comentarios.

Paso 3: Convierte los comentarios en secuencias de números utilizando el vocabulario creado.

3 Construcción de la red neuronal

Paso 1: Diseña y construye una red neuronal para el análisis de sentimientos, utilizando capas de embeddings para representar las palabras y capas densas para la clasificación.

Paso 2: Define la arquitectura de la red neuronal, incluyendo la cantidad de capas ocultas, las funciones de activación y la capa de salida.

4 Entrenamiento del modelo

Paso 1: Compila el modelo con la función de pérdida adecuada, como la entropía cruzada, y el optimizador, como Adam o SGD.

Paso 2: Entrena el modelo utilizando el conjunto de datos de entrenamiento, ajustando los pesos de la red neuronal mediante el proceso de retropropagación.

5 Evaluación del modelo

Paso 1: Evalúa el modelo utilizando el conjunto de datos de prueba y calcula métricas de rendimiento, como la precisión y el puntaje F1.

Paso 2: Analiza los resultados para determinar la efectividad del modelo en el análisis de sentimientos.

6 Clasificación de nuevos comentarios

Paso 1: Preprocesa los nuevos comentarios siguiendo los pasos de tokenización, creación de secuencias y conversión a números utilizando el vocabulario creado.

Paso 2: Utiliza el modelo entrenado para realizar predicciones en los nuevos comentarios y clasificarlos como positivos o negativos.

Paso 3: Muestra los resultados del análisis de sentimientos para cada comentario.

Unidad III

Práctica 1. Clasificación de imágenes con redes neuronales convolucionales.

1. Preparación del entorno de desarrollo

Paso 1: Puedes utilizar entornos de desarrollo como Jupyter Notebook o Google Colab para escribir y ejecutar tu código.

Paso 2: Instala las bibliotecas necesarias, como TensorFlow, Keras y NumPy, utilizando el gestor de paquetes de Python, pip.

2 Preparación de los datos

Paso 1: Descarga el conjunto de datos MNIST, que contiene imágenes de dígitos escritos a mano, desde el sitio web oficial de MNIST o utilizando funciones de carga de datos disponibles en bibliotecas como Keras.

Paso 2: Carga los datos en tu entorno de desarrollo y realiza una exploración inicial para familiarizarte con ellos.

Paso 3: Preprocesa los datos, dividiéndolos en conjuntos de entrenamiento y prueba. Puedes usar la función `train_test_split` de la biblioteca scikit-learn para esto.



Paso 4: Realiza el escalado de las imágenes para normalizar los valores de píxeles. Puedes dividir los valores de píxeles entre 255 para obtener valores en el rango de 0 a 1.

3 Diseño y entrenamiento del modelo

Paso 1: Define la arquitectura de la red neuronal convolucional. Puedes utilizar capas convolucionales, capas de agrupamiento (pooling) y capas completamente conectadas.

Paso 2: Compila el modelo especificando la función de pérdida y el optimizador. Para la clasificación de múltiples clases, puedes usar la función de pérdida `categorical_crossentropy` y el optimizador `adam`.

Paso 3: Entrena el modelo utilizando el conjunto de datos de entrenamiento. Ajusta el número de épocas de entrenamiento y el tamaño del lote según sea necesario.

Paso 4: Mientras entrenas el modelo, puedes monitorear la precisión y la pérdida en cada época para evaluar el rendimiento del modelo.

4 Evaluación y mejora del rendimiento

Paso 1: Evalúa el modelo utilizando el conjunto de datos de prueba. Utiliza la función `evaluate` para obtener la precisión y la pérdida del modelo en el conjunto de prueba.

Paso 2: Analiza los resultados obtenidos y, si es necesario, realiza ajustes en la arquitectura de la red neuronal convolucional. Puedes agregar más capas, ajustar los hiperparámetros o probar diferentes optimizadores para mejorar el rendimiento.

Paso 3: Realiza pruebas adicionales utilizando conjuntos de datos diferentes o imágenes reales para evaluar el desempeño del modelo en situaciones del mundo real.

Practica 2. Detección de enfermedades en aguacates utilizando redes neuronales convolucionales.

1. Preparación del entorno de desarrollo

Paso 1: Configura tu entorno de desarrollo, como Jupyter Notebook o Google Colab, e instala las bibliotecas necesarias, como TensorFlow, Keras, OpenCV y NumPy.

2 Preparación de los datos

Paso 1: Recopila un conjunto de datos que contenga imágenes de aguacates afectados por diferentes enfermedades comunes en el estado de Michoacán, como la mancha de sol, el hongo Anthracnose o la pudrición de la raíz.

Paso 2: Preprocesa los datos, asegurándote de que las imágenes estén correctamente etiquetadas con la información de las enfermedades correspondientes.

Paso 3: Divide los datos en conjuntos de entrenamiento y prueba.

3 Diseño y entrenamiento del modelo

Paso 1: Define la arquitectura de la red neuronal convolucional para la detección de enfermedades en aguacates. Puedes utilizar una arquitectura pre-entrenada como VGG16 o ResNet como punto de partida.

Paso 2: Ajusta los hiperparámetros del modelo, como el tamaño del lote, el número de épocas y la tasa de aprendizaje, según sea necesario.

Paso 3: Compila el modelo especificando la función de pérdida y el optimizador adecuados para la detección de enfermedades en aguacates.

Paso 4: Entrena el modelo utilizando el conjunto de datos de entrenamiento.



4 Evaluación y mejora del rendimiento

Paso 1: Evalúa el modelo utilizando el conjunto de datos de prueba. Calcula métricas como la precisión, el recall y el F1-score para medir la efectividad del modelo en la detección de enfermedades en aguacates.

Paso 2: Analiza los resultados obtenidos y, si es necesario, realiza ajustes en la arquitectura de la red neuronal convolucional. Puedes agregar capas adicionales, ajustar los hiperparámetros o utilizar técnicas de regularización para mejorar el rendimiento del modelo.

Paso 3: Realiza pruebas adicionales utilizando imágenes reales de aguacates afectados por enfermedades y evalúa el desempeño del modelo en situaciones del mundo real.

5 Aplicación en tiempo real

Paso 1: Utiliza el modelo entrenado para realizar la detección de enfermedades en aguacates en tiempo real. Esto implica capturar imágenes de aguacates en los huertos o en los puntos de inspección.

Paso 2: Procesa las imágenes capturadas utilizando OpenCV para detectar las regiones de interés (ROI) que podrían contener enfermedades.

Paso 3: Utiliza el modelo entrenado para clasificar las ROI y determinar la presencia y el tipo de enfermedad en cada una.

Paso 4: Visualiza los resultados obtenidos, ya sea mostrando las imágenes con las áreas detectadas o generando informes con la información de las enfermedades encontradas en los aguacates.

Practica 3. Aplicando los mismos pasos de la práctica anterior, desarrollar un sistema de reconocimiento de emociones en tiempo real utilizando redes neuronales convolucionales. El objetivo es detectar y clasificar las emociones expresadas por las personas en imágenes o secuencias de video en tiempo real.

Unidad IV

Practica 1. Análisis de sentimientos en redes sociales utilizando el conjunto de datos Sentiment140. Repositorio del conjunto de datos: Stanford (<https://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip>)

1. Descarga del conjunto de datos:

Paso 1: Accede al repositorio de conjuntos de datos de Stanford.

Paso 2: Descarga el archivo zip que contiene el conjunto de datos "Sentiment140".

2 Preprocesamiento de datos:

Paso 1: Extrae el archivo zip descargado y obtén los archivos de datos necesarios.

Paso 2: Lee los archivos y realiza el preprocesamiento necesario, como la limpieza de texto y la codificación de etiquetas de sentimiento.

3 Construcción del modelo:

Paso 1: Crea un modelo de red neuronal recurrente utilizando una capa LSTM o GRU.

Paso 2: Define la arquitectura del modelo, incluyendo el número de capas, la función de activación y otros hiperparámetros.



4 Entrenamiento del modelo:

Paso 1: Divide el conjunto de datos en conjuntos de entrenamiento y prueba.

Paso 2: Entrena el modelo utilizando los datos de entrenamiento y ajusta los pesos de las conexiones de la red neuronal recurrente.

Paso 3: Realiza ajustes en los hiperparámetros para mejorar el rendimiento del modelo.

5 Evaluación del modelo:

Paso 1: Evalúa el rendimiento del modelo utilizando los datos de prueba.

Paso 2: Calcula métricas como la precisión, la recuperación y la puntuación F1 para medir la calidad de las clasificaciones de sentimientos.

Practica 2. Predicción de cosechas utilizando redes neuronales recurrentes. Conjunto de datos: Datos históricos de producción agrícola, clima y factores ambientales.

1. Recopilación de datos:

Paso 1: Obtén datos históricos de producción agrícola de diferentes cultivos, incluyendo información sobre la cantidad de cosechas realizadas en años anteriores.

Paso 2: Obtén datos climáticos como la temperatura, la humedad, la precipitación, etc., para la misma región y período de tiempo.

Paso 3: Recopila información adicional relevante para la predicción de cosechas, como datos de fertilizantes, prácticas agrícolas, enfermedades de los cultivos, etc.

2 Preprocesamiento de datos:

Paso 1: Limpia y normaliza los datos para eliminar ruido y asegurar la consistencia.

Paso 2: Asegúrate de que los datos de producción agrícola y los datos climáticos estén alineados en términos de tiempo y ubicación geográfica.

Paso 3: Divide los datos en conjuntos de entrenamiento y prueba.

3 Construcción del modelo de red neuronal recurrente:

Paso 1: Define la arquitectura de la red neuronal recurrente, que puede incluir capas LSTM o GRU.

Paso 2: Considera la inclusión de capas adicionales, como capas densas, para mejorar la capacidad de predicción.

Paso 3: Define la función de pérdida y el algoritmo de optimización adecuados para el problema de predicción de cosechas.

1. Entrenamiento del modelo:

Paso 1: Alimenta los datos de entrenamiento a la red neuronal recurrente.

Paso 2: Ajusta los pesos de la red neuronal utilizando el algoritmo de optimización seleccionado.

Paso 3: Realiza múltiples iteraciones de entrenamiento hasta que el modelo converja o alcance un nivel de rendimiento satisfactorio.

4 Evaluación del modelo y predicción de cosechas:

Paso 1: Utiliza los datos de prueba para evaluar el rendimiento del modelo.

Paso 2: Calcula métricas de evaluación, como el error medio absoluto (MAE) o el error cuadrático medio (MSE), para medir la precisión de las predicciones.



Paso 3: Utiliza el modelo entrenado para predecir cosechas futuras en base a datos climáticos y otros factores relevantes.

Recuerda ajustar los hiperparámetros del modelo y realizar experimentos adicionales para mejorar su rendimiento. Además, considera la posibilidad de agregar técnicas de regularización y validación cruzada para evitar el sobreajuste y garantizar la generalización adecuada del modelo.

Ejemplos de conjuntos de datos etiquetados relacionados con el sector agrícola y sus respectivos repositorios:

1. Conjunto de datos: Global Maize Production
Repositorio: <https://www.kaggle.com/unitednations/global-maize-production>
2. Conjunto de datos: Global Wheat Production
Repositorio: <https://www.kaggle.com/unitednations/global-wheat-production>
3. Conjunto de datos: Crop Yield Prediction
Repositorio: <https://www.kaggle.com/vishalmane109/crop-yield-prediction>
4. Conjunto de datos: Climate Change and Agriculture
Repositorio: <https://www.kaggle.com/venky97/climate-change-and-its-effect-on-agriculture>
5. Conjunto de datos: Plant Pathology
Repositorio: <https://www.kaggle.com/c/plant-pathology-2020-fgvc7>

Estos conjuntos de datos contienen información etiquetada sobre diversos aspectos agrícolas, como la producción de maíz, trigo, predicciones de rendimiento de cultivos, cambio climático y efectos en la agricultura, y patología de plantas. Puedes acceder a ellos a través de los enlaces proporcionados y explorar los datos etiquetados disponibles para entrenar y evaluar tus modelos de redes neuronales recurrentes en el contexto agrícola.

9. Proyecto de asignatura

Fundamento:

El objetivo de este proyecto es crear una solución escalable y accesible basada en redes neuronales convolucionales para el análisis de imágenes en tiempo real. La solución permitirá a los usuarios cargar imágenes a través de una aplicación web y/o móvil, llamar al modelo en la nube a través de una API y recibir resultados de detección y clasificación de objetos.

Planeación:

Definir requisitos y alcance:

Identificar los requisitos del proyecto en términos de funcionalidad, rendimiento, seguridad y usabilidad.

Definir el alcance del proyecto, incluyendo las características clave, los casos de uso y los usuarios objetivo.

Recopilar y etiquetar conjuntos de datos:

Buscar conjuntos de datos relevantes en línea que contengan imágenes etiquetadas para entrenar y evaluar el modelo.



Etiquetar las imágenes para las clases de objetos o características de interés específicas para el proyecto.

Preprocesamiento y entrenamiento del modelo:

Realizar el preprocesamiento de los datos, como redimensionar, normalizar y dividir en conjuntos de entrenamiento y prueba.

Seleccionar y configurar el modelo de red neuronal convolucional adecuado para el problema en cuestión.

Entrenar el modelo utilizando el conjunto de datos etiquetado y optimizar los hiperparámetros para lograr un rendimiento óptimo

Implementación del modelo en la nube:

Configurar el entorno en la nube utilizando servicios como Google Cloud, AWS o Microsoft Azure para alojar y ejecutar el modelo.

Desplegar el modelo entrenado en la infraestructura en la nube y asegurarse de que esté disponible a través de una API.

Desarrollo de la aplicación web y/o móvil:

Seleccionar un marco de desarrollo web y/o móvil adecuado, como Django, Flask, React Native o Flutter.

Diseñar e implementar la interfaz de usuario de la aplicación, incluyendo la capacidad de cargar imágenes y enviar solicitudes a la API del modelo.

Implementar la lógica de comunicación con la API, incluyendo la autenticación y el manejo de respuestas del modelo.

Pruebas y refinamiento:

Realizar pruebas exhaustivas de la aplicación para asegurarse de que funcione correctamente y cumpla con los requisitos establecidos.

Recopilar y analizar comentarios de los usuarios y realizar ajustes y mejoras según sea necesario.

Implementación y despliegue:

Desplegar la aplicación web en un servidor web o en un servicio de alojamiento en la nube.

Para la aplicación móvil, compilar el código en un archivo ejecutable y publicarlo en las tiendas de aplicaciones correspondientes.

Desarrollo

1. Definir requisitos y alcance:

- Identificar los requisitos funcionales y no funcionales del proyecto en términos de funcionalidad, rendimiento, seguridad y usabilidad.
- Determinar el alcance del proyecto, incluyendo las características clave, los casos de uso y los usuarios objetivo.

2. Recopilar y etiquetar conjuntos de datos:

- Buscar conjuntos de datos relevantes en línea que contengan imágenes etiquetadas para entrenar y evaluar el modelo.



- Etiquetar las imágenes para las clases de objetos o características de interés específicas para el proyecto. Puedes utilizar herramientas de etiquetado como LabelImg o RectLabel.
3. Preprocesamiento y entrenamiento del modelo:
- Realizar el preprocesamiento de los datos, como redimensionar, normalizar y dividir en conjuntos de entrenamiento y prueba.
 - Seleccionar y configurar el modelo de red neuronal convolucional adecuado para el problema en cuestión. Puedes utilizar bibliotecas como TensorFlow o PyTorch.
 - Entrenar el modelo utilizando el conjunto de datos etiquetado y optimizar los hiperparámetros para lograr un rendimiento óptimo. Puedes utilizar técnicas de transfer learning si es necesario.
4. Implementación del modelo en la nube:
- Configurar el entorno en la nube utilizando servicios como Google Cloud, AWS o Microsoft Azure para alojar y ejecutar el modelo. Puedes utilizar servicios como Google Cloud AI Platform, AWS SageMaker o Azure Machine Learning.
 - Desplegar el modelo entrenado en la infraestructura en la nube y asegurarse de que esté disponible a través de una API. Puedes utilizar frameworks como Flask o Django para crear la API.
5. Desarrollo de la aplicación web y/o móvil:
- Seleccionar un marco de desarrollo web y/o móvil adecuado, como Django, Flask, React Native o Flutter.
 - Diseñar e implementar la interfaz de usuario de la aplicación, incluyendo la capacidad de cargar imágenes y enviar solicitudes a la API del modelo. Puedes utilizar bibliotecas como React o Vue.js para el desarrollo web, y Flutter o React Native para el desarrollo móvil.
 - Implementar la lógica de comunicación con la API, incluyendo la autenticación y el manejo de respuestas del modelo. Puedes utilizar bibliotecas como Axios para realizar solicitudes HTTP.
6. Pruebas y refinamiento:
- Realizar pruebas exhaustivas de la aplicación para asegurarse de que funcione correctamente y cumpla con los requisitos establecidos. Puedes utilizar herramientas de pruebas como Jest, Pytest o Selenium.
 - Recopilar y analizar comentarios de los usuarios y realizar ajustes y mejoras según sea necesario. Puedes utilizar herramientas de seguimiento de problemas como Jira o Trello para gestionar los problemas y solicitudes de los usuarios.
7. Implementación y despliegue:
- Desplegar la aplicación web en un servidor web o en un servicio de alojamiento en la nube. Puedes utilizar servicios como Heroku, Netlify o AWS S3.
 - Para la aplicación móvil, compilar el código en un archivo ejecutable y public

Evaluación

La rúbrica contemplará los siguientes criterios:

Planeación 20%

Desarrollo 40%

Publicación 10%

Exposición del proyecto 20%

Trabajo en equipo 10%



10. Evaluación por competencias

La evaluación debe ser continua y cotidiana por lo que se debe considerar el desempeño en cada una de las actividades de aprendizaje, haciendo especial énfasis en:

- Rúbricas o productos, señalados en cada unidad académica dentro de las actividades de aprendizaje.
- Prácticas propuestas y su presentación y exposición en plenaria. Algunas se evaluarán por equipo.
- Portafolio de evidencias de Información recabada durante las consultas e investigaciones solicitadas, plasmadas en documentos escritos.
- Descripción de otras experiencias concretas que se obtendrán al participar en discusiones, exposiciones o cualquier otro medio didáctico-profesional que trate sobre la materia y que deberán realizarse durante el curso académico.
- Exámenes teórico-prácticos para comprobar la efectividad del estudiante en la resolución de casos prácticos.

11. Fuentes de información

1. Santanu Pattanayak. Pro Deep Learning with Tensorflow. Apress. 2017
2. Tom M. Mitchell. Machine Learning. McGraw-Hill. 1997. 3. Joseph Howse. OpenCV Computer Vision with Python. BIRMINGHAM – MUMBAI. 2013.
3. Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.
4. Adrian Rosebrock. Deep Learning for computer vision with python. PyImageSearch. 2017
5. Richard Szeliski. Computer Vision: Algorithms and Applications. Springer. 2010.
6. Adrian Rosebrock. Python for computer vision with OpenCV and Deep Learning. PyImageSearch. 2021
7. Ian Goodfellow, Yoshua Bengio, and Aaron Corville. Deep Learning. MIT Press. 2016.
8. Mohamed Elgendy. Deep Learning for vision Systems. Manning Publications. 2020.
9. OBosch Rué, A. Casas Roma, J. & Lozano Bagén, T. (2019). Deep learning: principios y fundamentos: (ed.). Editorial UOC, Disponible en ELibro
10. Duque Domingo, J. Gómez García-Bermejo, J. & Zalama Casanova, E. (2024). Visión artificial: componentes de los sistemas de visión y nuevas tendencias en Deep Learning: (1 ed.). RA-MA Editorial, Disponible en ELibro