Instituto Tecnológico y de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias

Inteligencia Artificial Avanzada para la Ciencia de Datos I

Ingeniería en Ciencias de Datos y Matemáticas

Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución Portafolio Análisis

Profesor
Jesús Adrián Rodríguez Rocha

Andrea Galicia Jimenez A01177643

Monterrey, Nuevo León. 10 de septiembre de 2023

1. Introducción

Los modelos de Machine Learning son una parte fundamental de la inteligencia artificial que se centra en la creación y entrenamiento de algoritmos y sistemas que pueden aprender de datos y tomar decisiones basadas en patrones y experiencias previas, en lugar de depender exclusivamente de programación explícita. Estos modelos son la columna vertebral de muchas aplicaciones y tecnologías modernas, y su utilidad es diversa y amplia.

En esencia, los modelos de ML están diseñados para automatizar tareas o tomar decisiones complejas mediante el análisis y la interpretación de datos. Pueden clasificar imágenes, predecir el futuro valor de una acción, detectar fraudes en transacciones financieras, recomendar productos a usuarios en línea, traducir idiomas y mucho más. La flexibilidad y el potencial de los modelos de ML se derivan de su capacidad para adaptarse y mejorar con el tiempo a medida que se les proporciona más información y datos.

A lo largo de estas 5 semanas, hemos aprendido distintos modelos de aprendizajes automáticos, su funcionamiento, su implementación y en general todo lo que rodea a este mismo. Uno de los modelos aprendidos recientemente fueron las redes neuronales y gracias a su resolución de problemas, específicamente para categorizar, se me hizo de gran interés su funcionamiento por lo que se decidió utilizar este modelo para esta asignatura.

Antes de empezar su implementación se tiene que entender bien que es una red neuronal. Esta esta descrita como un modelo computacional inspirado en el cerebro humano que se utiliza para aprender y tomar decisiones basadas en datos y patrones complejos. Está compuesta por neuronas artificiales organizadas en capas interconectadas y se utiliza en una amplia gama de aplicaciones de aprendizaje automático, como visión por computadora, procesamiento de lenguaje natural y más. Su flexibilidad y capacidad para manejar datos no estructurados la hacen fundamental en la inteligencia artificial y nos ayudaran en el proceso de todo el entrenamiento y categorización que buscaremos.

2. Base de Datos

Como se menciono anteriormente se decidió trabajar con una red neuronal, específicamente usando la arquitectura de LeNet-5 la cual sirve principalmente para el reconocimiento de caracteres como dígitos en cheques bancarios, tipos de comida, modelos de automóviles y en este caso tipo de prendas de ropa.

Ahora para la implementación de esta red neuronal se decidió no implementar un csv tal cual sino que se utilizo una base de datos llamada **Fashion MNIST** la cual contiene imagenes de prendas de vestir y accesorios. Una gran ventaja de esta base de datos es que, como mencionado anteriormente, al no ser tal cual un csv, se obtiene mediante la biblioteca de **Keras** y **TensorFlow** para así no tener problemas con datos faltantes u información no coherente.

Hablando un poco mas sobre la estructura de esta base de datos, esta contiene 70,0000 imágenes de ropa y accesorios en blanco y negro, donde 60,000 imágenes son utilizadas para entrenamiento (train) y las otras 10,000 para el conjunto de prueba (test), divididas en 10 categorías. Cada imagen tiene una resolución de 28x28 pixeles aunque en nuestro código intentamos modificar esto para obtener una imagen mas HD.

3. Modelo Implementado

A continuación se explicara a detalle cada parte y proceso que se realizo dentro del código, desde la carga de librerías hasta los resultados finales con las diferentes pruebas:

3.1. Librerias

Se utilizaron diferentes librerías para diversas tareas en el proyecto:

- **TensorFlow y Keras:** Estas librerías se emplearon para construir y entrenar modelos de redes neuronales, lo que permitió la implementación del aprendizaje automático. Específicamente con estas dos librerías se utilizaron muchas extensiones de estas mismas para tareas mas especificas.
- **Matplotlib:** Fue utilizada para crear visualizaciones y gráficos que facilitaron la comprensión y presentación de datos.
- **NumPy:** Facilitó la manipulación y operaciones numéricas con datos, especialmente para trabajar con arreglos multidimensionales.
- **Scikit-Learn:** Se utilizó para métricas de evaluación y técnicas de validación cruzada, así como para generar informes de clasificación y matrices de confusión.

En conjunto, estas librerías ofrecieron las herramientas necesarias para desarrollar, evaluar y visualizar modelos de aprendizaje automático en el proyecto.

3.2. Carga y Distribución de Datos

Una vez teniendo todas las librerias se inicia cargando el conjunto de datos de Fashion MNIST utilizando Keras y lo divide en conjuntos de entrenamiento y prueba. Las imágenes de entrenamiento y sus etiquetas se almacenan en x_train y y_train, mientras que las imágenes de prueba y sus etiquetas se almacenan en x_test y y_test. Esto proporciona los datos necesarios para entrenar y evaluar modelos de aprendizaje automático en el conjunto de datos de Fashion MNIST.

Una vez que se dividen los datos para entrenar y evaluar, simplemente imprimimos las dimensiones de estos mismos para comprender la estructura de los datos.

3.3. Preprocesamiento de Datos

Gracias a que en si no se esta cargando la base de datos de un csv sino directamente de la biblioteca de **Keras y TensorFlow**, el preprocesamiento de datos no es tan extenso y nos ayuda a no tener futuros problemas con datos faltantes o incongruencia con las dimensiones.

Para esta parte se crea una lista llamada $class_names$ que asocia nombres descriptivos con las etiquetas numéricas en el conjunto de datos. Después se realiza la normalización de las intensidades de los píxeles de las imágenes, dividiendo todos los valores de los píxeles por 255.0 para escalarlos al rango de 0 a 1. Finalmente, se muestra visualmente 25 imágenes de prendas de vestir de Fashion MNIST junto con sus nombres de clase asociados. Esto ayuda a comprender las categorías de prendas representadas en el conjunto de datos.

Figura 1: Primeras 25 imagenes del dataset

3.4. Construir el Modelo

Para empezar a construir el modelo se realizó un proceso de reconfiguración de las imágenes de entrenamiento y prueba. Las imágenes originales se reformatearon para que se ajustaran a la estructura requerida por una red neuronal convolucional (CNN). Esto implicó cambiar la forma de las imágenes a una representación tridimensional de 28x28 píxeles con un solo canal, que es comúnmente utilizado en imágenes en escala de grises. Esta adaptación es esencial para garantizar que las imágenes sean compatibles con la entrada de la CNN, que se utilizará posteriormente en el proyecto. Además, se imprimieron las formas

resultantes de los arreglos para verificar que las imágenes se hubieran reformateado de manera adecuada.

Despues, se aplica una técnica llamada çodificación one-hot.ª las etiquetas de clase. Esto significa que las etiquetas numéricas se transforman en vectores binarios, donde un valor específico es igual a 1 y los demás son 0. Esto se hace tanto para el conjunto de entrenamiento como para el de prueba, con la finalidad de preparar las etiquetas para su uso en modelos de clasificación de múltiples clases, como es el caso en el conjunto de datos de Fashion MNIST, donde hay 10 clases diferentes. La codificación **one-hot** facilita el proceso de entrenamiento y evaluación de modelos de aprendizaje automático en problemas de clasificación.

Finalmente, se crea nuestro modelo de la red neuronal con distintas capas que crean el funcionamiento total y especifico de este modelo, las capas que se implementaron son las siguientes:

- Capa de convolución (C1): Esta capa aplica convoluciones en las imágenes de entrada para extraer características. Tiene 6 filtros de convolución con un tamaño de kernel de 5x5, activación ”tanhz se asegura de mantener el mismo tamaño de salida que la entrada mediante el parámetro ’same’.
- Capa de submuestreo (S2): Esta capa realiza un submuestreo o ”pooling”para reducir la dimensionalidad y preservar las características más importantes. Utiliza una operación de ”max poolingçon un tamaño de ventana de 2x2.
- Capa de convolución (C3): Similar a la capa C1, aplica convoluciones adicionales con 16 filtros de 5x5 y activación ”tanh”.
- Capa de submuestreo (S4): Al igual que S2, realiza un submuestreo con ”max pooling”para reducir aún más la dimensionalidad.
- Capa de aplanamiento (C5): Esta capa aplanadora transforma la salida de las capas anteriores en un vector unidimensional, preparándolo para ser alimentado a capas completamente conectadas.
- Capa completamente conectada (F6): Una capa densa con 120 neuronas y activación ”tanh”.
- Capa completamente conectada (F7): Otra capa densa con 84 neuronas y activación ”tanh”.
- Capa de salida (Output): La capa final tiene 10 neuronas con activación softmax, que se utiliza para la clasificación multiclase. Cada neurona representa una clase diferente en el conjunto de datos.

3.5. Entrenamiento

Una vez teniendo nuestra red neuronal estructurada es momento de entrenar al modelo. Para esto se decidio tener 3 pruebas diferentes donde se manipularon el numero de epochs y el numero de batches (lotes) para analizar un poco mas el comportamiento del modelo y poder ver con cuales valores se ajusta mejor la red neuronal.

Los pasos para este entrenamiento y diagnostico fueron los siguientes:

1. Se establecen los hiperparametros de el numero de epochs, el tamaño del lote (batch), la tasa de aprendizaje que controla la magnitud de los ajustes de peso durante el entrenamiento y un optimizador SGD que se utiliza para ajustar los pesos del modelo durante su entrenamiento. **Es importante mencionar que estos hiperparametros seran los que se modificaran en las diferentes pruebas para su futuro analisis**

2. Se configura el modelo de red neuronal para el entrenamiento. Se especifica el optimizador (SGD con una tasa de aprendizaje), la función de pérdida (categorical crossentropy) y la métrica de evaluación (precisión) que se utilizarán durante el proceso de entrenamiento.

3. Se entrena el modelo de red neuronal con un conjunto de datos de entrenamiento y se valida con un conjunto de datos de prueba. El proceso se repite durante un número específico de epochs y utilizando la cantidad de muestras para los lotes que se hayan especificado. El objetivo es que el modelo aprenda a relacionar las imágenes con las etiquetas de clase.

4. Evaluamos nuestro modelo ajustando dos gráficas diferentes, una que nos muestra el entrenamiento y validación de la precisión y otra que nos muestra el entrenamiento y validación de la pérdida, esto con el fin de poder ver visualmente nuestro resultado del modelo.

5. Se calcula y muestra la precisión final del modelo en el conjunto de entrenamiento y en el conjunto de validación después de completar el proceso de entrenamiento. La precisión se obtiene del historial de entrenamiento y se imprime en la consola.

6. Finalmente se hace un diagnostico sobre el grado de bias (sesgo) en el modelo como bajo, medio o alto, el grado de varianza tambien como bajo, medio o alto, y el nivel de ajuste del modelo como underfitt, fitt o overfitt

4. Resultados

El procedimiento explicado anteriormente desde el preprocesamiento de los datos hasta el diagnostico de ciertas características se realizo con tres pruebas diferentes donde se obtuvieron los siguientes resultados:

4.1. Prueba 1:

Para esta primera prueba se utilizaron los siguientes hiperparametros:

- numero de epochs = 10
- tamaño de lotes (batch) =64
- taza de aprendizaje: 0.01

Implementando estos hiperparametros tuvimos las siguientes curvas de aprendizaje y precisión final de entrenamiento y validación:

Figura 2: Gráfico Prueba 1 “Training and Validation Accuracy”

Figura 3: Gráfico Prueba 1 “Training and Validation Loss”

```
Precisión final de entrenamiento: 0.9526500105857849
Precisión final de validación: 0.9065999984741211
```

Figura 4: Resultados precisión de entrenamiento y validación - Prueba 1

Finalmente para esta primera prueba se obtuvieron los siguientes diagnósticos:

```
Diagnóstico de Sesgo (Bias): Sesgo alto (overfitting hacia datos de entrenamiento)
Diagnóstico de Varianza: Varianza baja (bien generalizado)
Diagnóstico de Ajuste del Modelo: Ajuste adecuado (modelo bien equilibrado)
```

Figura 5: Muestra de diagnósticos - Prueba 1

Para esta primera prueba podemos notar que los resultados de la precision del ultimo epoch de entrenamiento y validacion son muy altos, asumiendo que se tiene un muy buen resultado que tambien se puede apreciar en las graficas. De igual manera los resultados del diagnostico indican que el modelo presenta un alto sesgo (overfitting hacia los datos de entrenamiento), lo que significa que se ajusta demasiado a los datos de entrenamiento, capturando incluso el ruido en lugar de los patrones reales. Sin embargo, la varianza es baja (bien generalizado), lo que implica que el modelo generaliza bien a nuevos datos y no es muy sensible a pequeñas fluctuaciones en los datos de entrenamiento. En resumen, **El modelo parece estar bien equilibrado y es capaz de aprender patrones importantes en los datos sin ajustarse en exceso.**

4.2. Prueba 2:

Para esta segunda prueba se utilizaron los siguientes hiperparametros:

- numero de epochs = 50
- tamaño de lotes (batch) = 32
- taza de aprendizaje: 0.01

Implementando estos hiperparametros tuvimos las siguientes curvas de aprendizaje y precisión final de entrenamiento y validación:

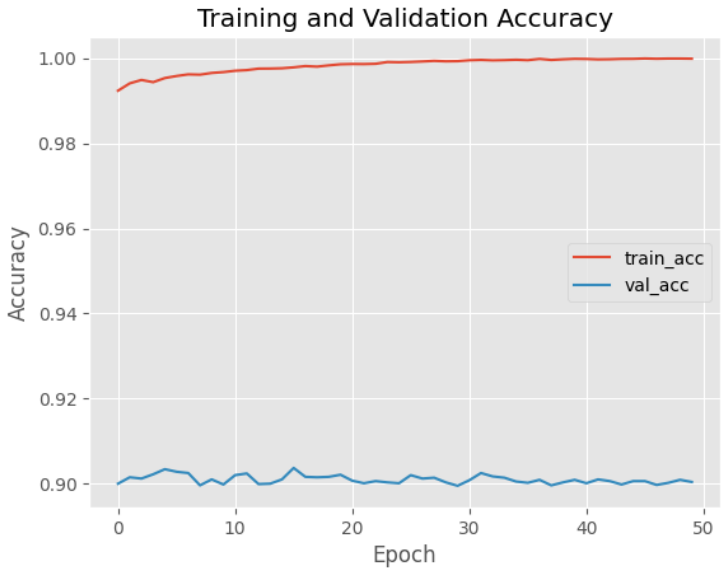

Figura 6: Gráfico Prueba 2 “Training and Validation Accuracy”

Figura 7: Gráfico Prueba 2“Training and Validation Loss”

```
Precisión final de entrenamiento: 0.9998999834060669
Precisión final de validación: 0.9003999829292297
```

Figura 8: Resultados precisión de entrenamiento y validación - Prueba 2

Finalmente se obtuvieron los siguientes diagnósticos:

```
Diagnóstico de Sesgo (Bias): Sesgo alto (overfitting hacia datos de entrenamiento)
Diagnóstico de Varianza: Varianza baja (bien generalizado)
Diagnóstico de Ajuste del Modelo: Overfitting (modelo complejo, posible sobreajuste)
```

Figura 9: Muestra de diagnósticos - Prueba 2

Para esta segunda prueba ambas graficas de precision y perdida muestran comportamientos muy extraños ya que se muestra casi una linea recta horizontal para el entrenamiento y la validacion enseñando que mientras los epochs van aumentando este mismo se mantiene casi inactivo lo cual no muestra mucha coherencia con los resultados numericos de la precision. En cuanto al diagnostico los resultados señalan que el modelo está experimentando un alto sesgo (overfitting hacia los datos de entrenamiento), lo que indica que se ajusta excesivamente a esos datos específicos, incluso al ruido presente. Aunque la varianza es baja (indicando una buena capacidad de generalización), el diagnóstico de ajuste del modelo sugiere que podría estar sobreajustando debido a su complejidad. En resumen, **el modelo parece estar sesgado hacia los datos de entrenamiento y puede estar sobreajustando, lo que podría requerir ajustes para mejorar su capacidad de generalización.**

4.3. Prueba 3:

Para esta tercera prueba se utilizaron los siguientes hiperparametros:

- numero de epochs = 5
- tamaño de lotes (batch) = 16
- taza de aprendizaje: 0.01

Implementando estos hiperparametros tuvimos las siguientes curvas de aprendizaje y precisión final de entrenamiento y validación:

Figura 10: Gráfico Prueba 3 “Training and Validation Accuracy”

Figura 11: Gráfico Prueba 3 “Training and Validation Loss”

```
Precisión final de entrenamiento: 0.9917500019073486
Precisión final de validación: 0.8963000178337097
```

Figura 12: Resultados precisión de entrenamiento y validación - Prueba 3

Finalmente se obtuvieron los siguientes diagnósticos:

```
Diagnóstico de Sesgo (Bias): Sesgo alto (overfitting hacia datos de entrenamiento)
Diagnóstico de Varianza: Varianza baja (bien generalizado)
Diagnóstico de Ajuste del Modelo: Overfitting (modelo complejo, posible sobreajuste)
```

Figura 13: Muestra de diagnósticos - Prueba 3

Estos diagnósticos indican que el modelo tiene un alto sesgo (overfitting hacia los datos de entrenamiento) debido a su complejidad excesiva, lo que impide una buena generalización a nuevos datos. La baja varianza sugiere que el modelo es consistente en su rendimiento. Para abordar este problema de overfitting, se necesita simplificar el modelo o aplicar técnicas de regularización.

Teniendo un resultado final de las tres diferentes pruebas podemos decir que para esta ocasión los hiperparametros de la **Prueba 1** fueron los mas exitosos por su porcentaje de precisión, ajuste en ambas gráficas y diagnostico especifico sobre las características del sesgo, varianza y ajuste del modelo. A comparación de la prueba 2 y 3, se puede notar como con los parámetros de la prueba 1 el modelo si esta aprendiendo a entrenar de forma correcta por el incremento en la precisión y la disminución en la perdida.

4.4. Análisis Extra

Finalizando las tres diferentes pruebas se decidió analizar un poco mas sobre el desempeño del modelo utilizando predicciones y una matriz de confusión.

El primer análisis realiza predicciones con el modelo entrenado en el conjunto de prueba y muestra las clases predichas (números) y sus etiquetas de clase correspondientes (nombres de prendas de vestir). Y después se calcula y muestra la matriz de confusión, que evalúa cómo el modelo de clasificación se desempeñó en la predicción de las clases en el conjunto de prueba. La matriz de confusión refleja las clasificaciones correctas e incorrectas para cada clase y proporciona información sobre el rendimiento del modelo en la clasificación. Finalmente, se desgloso un resumen general de todo el modelo.

Con nuestra matriz de confusión y estadísticas de desempeño podemos ver que se tuvieron números altos al momento de buscar el promedio de los pesos junto con su precisión y soporte de la red. Finalmente, lo último que nos arroja el código es un resumen de todo lo obtenido anteriormente, explicando capa por

capa sus parámetros y figura así como el resultado final de parámetros simples y entrenados, el cual nos podemos dar cuenta que todos los parámetros obtenidos fueron entrenados.

5. Conclusión

En conclusión, el proceso de entrenamiento y evaluación de modelos de redes neuronales es un componente fundamental del aprendizaje automático. A través de la configuración de modelos, el ajuste de hiperparámetros y la interpretación de resultados, podemos abordar problemas complejos y obtener información valiosa de los datos. Estas técnicas y conceptos tienen aplicaciones en una amplia gama de campos y desempeñan un papel crucial en la construcción de sistemas inteligentes que pueden tomar decisiones basadas en datos. El aprendizaje automático y las redes neuronales continúan siendo áreas de investigación y desarrollo prometedoras con un impacto significativo en la sociedad y la tecnología.