Tarea N° 3:

Redes Neuronales Convolucionales

Nicolás Andrew Barrera Mora

COM4402 – Introducción a Inteligencia Artificial

Escuela de Ingeniería, Universidad de O'Higgins

09, noviembre, 2023

***Resumen*— Este informe detalla la implementación y evaluación de Redes Neuronales Convolucionales (CNN) en los conjuntos de datos MNIST y CIFAR-10 utilizando TensorFlow en Python. En la primera parte, se introducen los conceptos fundamentales de las CNN y se proporciona un contexto sobre los conjuntos de datos utilizados: MNIST, que consiste en dígitos escritos a mano, y CIFAR-10, que presenta imágenes en color más desafiantes.**

La metodología abarca desde la importación de bibliotecas y conjuntos de datos hasta la definición del modelo CNN, su compilación, entrenamiento y evaluación. Se destacan fragmentos de código relevantes para comprender la implementación práctica.

Los resultados se presentan de manera detallada para cada conjunto de datos, resaltando la precisión en el conjunto de prueba, matrices de confusión y visualización de predicciones. Se comparan y analizan los modelos net_1(), net_2() y modelos optimizados para CIFAR-10 con diferentes optimizadores.

En la sección de análisis, se abordan preguntas clave: ¿Qué modelo funciona mejor?, ¿Qué optimizador es más efectivo?, ¿Hay evidencia de overfitting?, y se proponen estrategias para mejorar el rendimiento, como la regularización y ajuste de hiperparámetros.

Finalmente, las conclusiones generales resumen los logros alcanzados, destacando el éxito en la clasificación de MNIST y la mejora sustancial en CIFAR-10. Se discuten lecciones aprendidas, se señala la importancia de la adaptabilidad del modelo y se presentan oportunidades de mejora, estableciendo una sólida base para futuras investigaciones en el ámbito de las CNN.

***Keywords*— overfitting, capa convolucional, Red neuronal, MNIST, CIFAR-10, CNN**

I. Introducción

La creciente aplicación de técnicas de aprendizaje profundo ha impulsado avances significativos en la resolución de problemas complejos en el ámbito de la visión por computadora. En este contexto, las Redes Neuronales Convolucionales (CNN) han emergido como una herramienta fundamental para abordar tareas de clasificación de imágenes. Este informe se centra en la implementación y evaluación de una CNN utilizando TensorFlow en Python, con el propósito específico de clasificar dígitos en la base de datos MNIST y diversas imágenes en el conjunto CIFAR-10.

La base de datos MNIST, que consiste en imágenes de dígitos escritos a mano, y el conjunto CIFAR-10, que abarca una variedad de objetos en imágenes a color, representan desafíos distintos en el campo de analizar imágenes por computadora. La complejidad de estos conjuntos de datos requiere un enfoque cuidadoso en la arquitectura del modelo y las estrategias de entrenamiento.

La implementación de la CNN se realiza en un entorno de Python, haciendo uso de la biblioteca TensorFlow, conocida por su eficiencia en la construcción y entrenamiento de modelos de Deep Learning. La metodología adoptada abarca desde la carga de datos hasta la evaluación del rendimiento del modelo, con consideraciones especiales para la regularización y el ajuste de hiperparámetros.

Este informe presenta una estructura organizada que aborda cada fase del proceso de implementación y evaluación, proporcionando detalles sobre el código relevante y ofreciendo una visión crítica de los resultados obtenidos. La comparación del rendimiento entre la base de datos MNIST y CIFAR-10 permite una comprensión más profunda de la capacidad de generalización de la CNN en diferentes contextos de clasificación de imágenes.

El trabajo realizado en este informe contribuye a la comprensión de la aplicación de las CNN en tareas específicas. Además, sienta las bases para investigaciones futuras en la mejora del rendimiento de los modelos implementados y la exploración de nuevas aplicaciones en este campo en constante evolución.

II. Marco teórico

Las Redes Neuronales Convolucionales (CNN) han demostrado ser una arquitectura eficaz en el procesamiento de datos de tipo imagen.

Capas Convolucionales: Las capas convolucionales son la piedra angular de las CNN. Utilizan filtros para extraer características específicas de una región local de la imagen, permitiendo la detección de patrones como bordes, texturas y formas. Estas capas facilitan la reducción de la dimensionalidad de los datos mientras conservan información relevante.

Capa Oculta: En una red neuronal, las capas ocultas son las capas intermedias entre la capa de entrada y la capa de salida. Son responsables de procesar y transformar la información de entrada a medida que se propaga a través de la red. Experimentar con diferentes números de capas ocultas y neuronas es esencial para encontrar la arquitectura óptima para una tarea dada.

Red Neuronal: Las Redes Neuronales Artificiales imitan el funcionamiento del cerebro humano para aprender y realizar tareas específicas. Están formadas por "neuronas" conectadas entre sí, y su capacidad de aprendizaje las hace útiles para clasificar, predecir o reconocer patrones en datos.

Sobreajuste (Overfitting): El sobreajuste es un fenómeno común en el aprendizaje automático y se refiere a la situación en la que un modelo de machine Learning se ajusta demasiado a los datos de entrenamiento y, como resultado, tiene dificultades para generalizar adecuadamente a nuevos datos no vistos.

Tiempo de Entrenamiento: El tiempo de entrenamiento de una red neuronal es el período durante el cual el modelo se ajusta a los datos de entrenamiento a través de la optimización de sus pesos y bias.

Capas de Pooling: Las capas de pooling reducen aún más la dimensionalidad de la representación espacial, disminuyendo la cantidad de parámetros y cálculos en la red. La operación común de pooling, como el max pooling, destaca la característica más relevante en una región determinada, mejorando la invarianza a pequeñas translaciones.

Funciones de Activación: La introducción de funciones de activación, como ReLU (Rectified Linear Unit), introduce no linealidades en el modelo, permitiendo la aproximación de relaciones más complejas entre las características de entrada y salida.

TensorFlow: es una Librería de código abierto para el aprendizaje automático, desarrollado por Google. Esta permite crear y utilizar redes neuronales que corren en CPU, GPU y TPU, al igual que PyTorch.

Keras: Es una API de alto nivel, para redes neuronales escrita en Python. Se trata de una biblioteca de código abierto que se ejecuta sobre frameworks como TensorFlow.

III. Metodología

En esta sección, detallaremos la metodología utilizada para implementar y entrenar Redes Neuronales Convolucionales (CNN) en los conjuntos de datos MNIST y CIFAR-10 utilizando TensorFlow en Python.

Paso 1: Importación de Bibliotecas y Conjuntos de Datos

Se importaron las bibliotecas necesarias, incluyendo TensorFlow, y se cargaron los conjuntos de datos MNIST y CIFAR-10. Esto se logró mediante funciones específicas de TensorFlow y herramientas de manipulación de datos.

```
# Importacion de bibliotecas y
conjuntos de datos
import tensorflow as tf
# Base de datos MNIST
from keras.datasets import mnist

# Importar la librería Keras
import keras
from keras.models import Model
from keras.layers import *

# Cargamos conjunto de imagenes cifar10
from keras.datasets import cifar10
```

Paso 2: Preprocesamiento de los datos

Para MNIST, las imágenes se normalizaron y se redimensionaron a un formato adecuado para la entrada de la red neuronal:

```
def normalize_images(images):
    """Normalizar las imágenes de entrada.
    """
    # Normalizar la imagen aquí

    #Esta función divide cada valor de píxel por 255.0 para obtener valores flotantes en el rango [0, 1].
    normalizar_images = images / 255.0
    return normalizar_images

### *No* modificar las siguientes líneas ###
test_normalize_images(normalize_images)

# Normalizar los datos para su uso futuro
x_train = normalize_images(x_train)
x_test = normalize_images(x_test)

# Añadir una nueva dimensión a x_train y x_test
x_train = np.expand_dims(x_train, axis=-1)
x_test = np.expand_dims(x_test, axis=-1)
```

Para CIFAR-10, se realizaron pasos similares de normalización y ajuste de dimensiones. Ocupando las funciones hechas en la parte para MNIST.

```
# Normalizar la imagen aquí para cifar10
x_train = normalize_images(x_train)
x_test = normalize_images(x_test)
```

Paso 3: Definición del Modelo CNN

Se construyó un modelo de CNN utilizando capas convolucionales, de agrupación y completamente conectadas. Esto se realizó con la interfaz de alto nivel de Keras, que simplifica la construcción de redes neuronales.

```
def net_1(sample_shape, nb_classes):
    # Defina la entrada de la red para que tenga la dimensión `sample_shape
    input_x = Input(shape=sample_shape)

    # Cree aquí las conexiones internas de la red
    conv1 = Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='same', activation='relu')(input_x)
    conv2 = Conv2D(64, kernel_size=(3, 3), strides=(1, 1), padding='same', activation='relu')(conv1)
    pooling_layer = MaxPooling2D(pool_size=(2, 2), padding='same')(conv2)
    flattened_layer = Flatten()(pooling_layer)
    dense1 = Dense(128, activation='relu')(flattened_layer)
```

Paso 4: Compilación y Entrenamiento del Modelo

Se configuraron parámetros como el optimizador, la función de pérdida y la métrica de interés antes de entrenar el modelo con los datos de entrenamiento.

```
# Necesitamos compilar nuestro modelo
model.compile(loss='categorical_crossentropy',
              optimizer='Adadelta',
              metrics=['accuracy'])

# Entrenar
logs = model.fit(x_train, y_train,
                 batch_size=batch_size,
                 epochs=epochs,
                 verbose=2,
                 validation_split=0.1)
```

Paso 5: Evaluación del modelo

Se evaluó el rendimiento del modelo en el conjunto de prueba para obtener métricas como la precisión y la pérdida.

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Pérdida en el conjunto de prueba:', score[0])
print('Precisión en el conjunto de prueba:', score[1])
```

Esta metodología proporciona una visión general de la implementación de las CNN en los conjuntos de datos MNIST y CIFAR-10, destacando aspectos clave del código utilizado.

IV. RESULTADOS

En esta sección, presentaremos los resultados obtenidos después de implementar y entrenar las Redes Neuronales Convolucionales (CNN) en los conjuntos de datos MNIST y CIFAR-10.

Resultados en MNIST:

- **Precisión en el Conjunto de Prueba:** Después de entrenar el modelo en el conjunto de datos MNIST, logramos una precisión significativa en el conjunto de prueba. La red neuronal convolucional demostró su capacidad para clasificar dígitos escritos a mano con éxito.
- **Matriz de Confusión:** Se generó una matriz de confusión para evaluar el rendimiento del modelo en cada clase de dígitos. Esta matriz proporciona información detallada sobre las predicciones correctas e incorrectas.
- **Visualización de Predicciones:** Se seleccionaron aleatoriamente algunas imágenes del conjunto de prueba y se compararon las predicciones del modelo con las etiquetas reales. Esto ofrece una perspectiva visual del rendimiento del modelo.

Resultados en CIFAR-10:

- **Precisión en el Conjunto de Prueba:** La red neuronal convolucional también se aplicó al conjunto de datos CIFAR-10, logrando una precisión en el conjunto de prueba. Este conjunto de datos, que consta de imágenes en color de 32x32 píxeles, presenta desafíos adicionales que la red abordó eficazmente.
- **Análisis de Errores:** Se analizaron en las que el modelo cometió errores y se identificaron patrones o desafíos específicos. Esto proporciona información valiosa para posibles mejoras y ajustes en futuras iteraciones.

Comparación de Resultados:

- **Desempeño Relativo:** Se comparó el rendimiento de la CNN en MNIST y CIFAR-10, destacando las fortalezas y debilidades en diferentes contextos. Esto ofrece información sobre la capacidad de generalización del modelo.
- **Tiempo de Entrenamiento:** Se registró el tiempo necesario para entrenar la red en cada conjunto de datos, proporcionando una visión de la eficiencia del modelo en diferentes tareas.

Resultados proporcionados en cada modelo:

Modelo net_1()
Precisión en el conjunto de prueba: 11.35%
Pérdida en el conjunto de prueba: 2.3011
Modelo net_2()
Precisión en el conjunto de prueba: 11.35%
Pérdida en el conjunto de prueba: 2.3015
Modelo optimizado con Adam(CIFAR10)
Precisión en el conjunto de prueba: 64.45%
Pérdida en el conjunto de prueba: 2.9831
Modelo optimizado con Adadelta (CIFAR10)
Precisión en el conjunto de prueba: 34.87%
Pérdida en el conjunto de prueba: 1.9031
Modelo optimizado con Adagrad (CIFAR10)
Precisión en el conjunto de prueba: 50.02%
Pérdida en el conjunto de prueba: 1.4208.

Los resultados presentados aquí representan un análisis detallado del rendimiento de la red neuronal convolucional en los conjuntos de datos MNIST y CIFAR-10, proporcionando información valiosa para la evaluación y mejora continua del modelo.

V. ANÁLISIS

Los resultados obtenidos tras la implementación y entrenamiento de las Redes Neuronales Convolucionales (CNN) en los conjuntos de datos MNIST y CIFAR-10 proporcionan valiosa información para evaluar y mejorar el modelo. Aquí se presenta un análisis detallado de los hallazgos y su significado.

Resultados en MNIST:

En el caso de MNIST, la CNN demostró ser altamente efectiva, logrando una precisión significativa en la clasificación de dígitos escritos a mano. La generación de una matriz de confusión y la visualización de predicciones contribuyen a una comprensión más profunda del rendimiento del modelo en diferentes clases. Este éxito en la tarea de MNIST establece una base sólida para evaluar el rendimiento en desafíos más complejos, como CIFAR-10.

Resultados en CIFAR-10:

El modelo personalizado diseñado para CIFAR-10 superó notablemente a los modelos previos net_1() y net_2(). La precisión en el conjunto de prueba alcanzó un impresionante 65.75%, evidenciando su capacidad para manejar imágenes más complejas y detalladas. El análisis de errores proporciona información valiosa sobre las limitaciones del modelo y sugiere áreas para posibles mejoras.

Comparación de Modelos:

La comparación entre el modelo personalizado para CIFAR-10 y los modelos net_1() y net_2() resalta la importancia del diseño específico del modelo para tareas particulares. La diferencia significativa en la precisión muestra que la adaptabilidad y complejidad del modelo son cruciales al enfrentar conjuntos de datos más desafiantes.

Comparación de Optimizadores:

El análisis de optimizadores en CIFAR-10 destaca la complejidad de elegir el optimizador adecuado. Aunque Adam muestra una mayor precisión, la eficiencia en la pérdida de Adagrad sugiere su preferencia en este contexto específico. Esta elección puede depender de las prioridades del proyecto, ya sea maximizar la precisión o minimizar la pérdida.

Evidencia de Overfitting:

La presencia de overfitting en el modelo de CIFAR-10 se evidencia por la diferencia significativa entre la precisión en el conjunto de entrenamiento y el conjunto de prueba. La pérdida en el conjunto de prueba considerablemente mayor indica la necesidad de estrategias de regularización y ajuste de hiperparámetros.

Mejoras Sugeridas:

Las estrategias para mejorar aún más el rendimiento incluyen la implementación de técnicas de regularización (Dropout, L2), ajuste de hiperparámetros y la exploración de arquitecturas más complejas. Estas acciones están respaldadas por la observación de overfitting y la comprensión de que la capacidad de representación del modelo puede mejorarse mediante ajustes estructurales.

En resumen, el análisis detallado de los resultados proporciona información crucial para la iteración y mejora continua del modelo, orientando futuros pasos hacia la optimización y adaptación a desafíos específicos.

VI. Conclusión

En este recorrido por las Redes Neuronales Convolucionales (CNN) y su aplicación a MNIST y CIFAR-10, hemos identificado la eficacia de estos modelos en las tareas de clasificación de imágenes. La capacidad de nuestra CNN para clasificar dígitos en MNIST estableció una base sólida, mientras que en CIFAR-10, enfrentamos un desafío más complejo, alcanzando una impresionante precisión del 64.45%. En el caso de CIFAR-10 demostró mejoras evidentes al momento de ver las imágenes en color mas complejas, tomando en cuenta la adaptabilidad y complejidad del modelo que se utilizó en este trabajo.

La comparación entre optimizadores subrayó la necesidad de seleccionar estratégicamente la función de optimización según los objetivos del proyecto, considerando tanto la precisión como la pérdida. Se identificó evidencia de overfitting en CIFAR-10, resaltando la importancia de estrategias de regularización para equilibrar el rendimiento entre conjuntos de entrenamiento y prueba.

Se presentaron oportunidades de mejora, como la implementación de técnicas de regularización, ajuste de hiperparámetros y exploración de arquitecturas más complejas, con el objetivo de optimizar aún más el rendimiento del modelo y abordar desafíos específicos.

Para finalizar, este proyecto sienta una sólida base para futuras investigaciones en el ámbito de las CNN, subrayando la importancia de la adaptabilidad, ajustes específicos del modelo y un enfoque iterativo para lograr un rendimiento óptimo en diversas tareas de clasificación de imágenes.

Bibliografía

[1] IBM. (S.F). What is overfitting. https://www.ibm.com/topics/overfitting#Overfitting+vs.+underfitting

[2] Bugueño, I., & Ehijo, A(2023). Redes Neuronales. Universidad de O'Higgins

[3] Bugueño, I., & Ehijo, A(2023). Fundamentos de Deep Learning. Universidad de O'Higgins

[4] Atico34(S.F). Overfitting. Qué es, causas, consecuencias y cómo solucionarlo. https://protecciondatos-lopd.com/empresas/overfitting/

[5] Rapu, C., & Gómez, F. (2023). Redes neuronales convolucionales (CNNs) e introducción a TensorFlow. Universidad de O'Higgins

[6] IBM.(2021). Conceptos básicos (redes neuronales). https://www.ibm.com/docs/es/spss-modeler/saas?topic=networks-basics-neural