INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

TC2035. Diseño de redes neuronales y aprendizaje profundo (Gpo 101)

Proyecto de Aprendizaje Supervisado

Inteligencia artificial y métodos cognitivos

PROFESOR:

Dr. Santiago Enrique Conant Pablos

ESTUDIANTES:

Christian Jaffé Alarcón Acosta	A00832881
David Vázquez Moreno	A01735864
Grace Aviance Silva Aróstegui	A01285158

09 de septiembre del 2023

Introducción

Para este proyecto se llevarán a cabo dos modelos utilizando métodos de aprendizaje supervisado, específicamente redes neuronales para clasificar la base de datos Fashion MNIST.

Esta base de datos es parte de la librería de python “keras” y está compuesta por imágenes del catálogo de Zalando, que es una tienda de ropa y zapatos alemana, en estas imágenes hay diferentes productos como botines, pantalones, playeras, entre otros. Estas imágenes tienen un tamaño de 28x28, dando un total de 784 píxeles y están en escala de grises, asociadas con un etiqueta de clase, son 10 clases en total.

La base de datos igual contiene un set de entrenamiento con 60,000 muestras y un set de prueba con 10,000 muestras,que serán los utilizados para entrenar y probar el modelo.
El objetivo de este proyecto es que el modelo sea capaz de clasificar correctamente cada imágen a su clase correspondiente aplicando dos tipos diferentes de modelos basados en Redes Neuronales, uno visto en clase, que será el modelo por Red Neuronal Convolucional y uno investigado por nosotros, que será Capsule Neural Network.

Modelo 1: Red neuronal convolucional (visto en clase)

Descripción

El modelo por Redes Neuronales Convolucionales o Convolutional Neural Networks en inglés, es uno de los más utilizados para tareas de procesamiento de imágenes y reconocimiento visual.

Este modelo es caracterizado por utilizar capas de convolución que se utilizan para extraer características clave de las imágenes aplicando filtros mediante kernels que van explorando la imagen, después están las capas de pooling, que se utilizan para reducir la dimensionalidad de las características y así reducir el costo computacional, finalmente se utiliza una capa completamente conectada que de cierta forma, conecta las capas de convolución y pooling para tomar las decisiones con base en las características extraídas.

Preparación de datos

Una vez que ya se han cargado los datos, dividiéndolos en nuestro train y test set, debemos hacer una preparación adecuada para los datos, ya que, al tener imágenes, estamos trabajando con un tamaño de entrada de 28x28, que debe ser transformado para poder aplicarse al modelo.

Para ello, normalizamos los datos escalando las imágenes a un rango [0,1], posteriormente nos aseguramos que las imágenes tengan el tamaño correcto, que es (28,28,1) y finalmente, convertimos los vectores de clase a matrices de tipo binario, es decir, las volvemos variables categóricas.

Implementación y entrenamiento.

Notebook: **FashionMNIST-CNN.ipynb**
Secuencia de pasos realizados:

1. Importar las librerías necesarias.
2. Cargar el dataset Fashion MNIST desde Tf Keras dividiendo en Train y Test set.
3. Inicializar una variable con el tamaño de las imágenes (28,28,1)
4. Visualizar algunas de las imágenes a clasificar.
5. Observar los tamaños del train y test set para verificar que sean correctos.
6. Iniciar preparación de datos normalizando x_train y x_test.
7. Expandir el tamaño de las imágenes para asegurar que tengan el tamaño correcto(28x28x1)
8. Convertir los vectores de clase a matrices binarias(Volver las variables categóricas).
9. Iniciar construcción del modelo usando la función Sequential()
10. Agregar primera capa Convolucional 2D con un kernel de tamaño 3x3 y operación Max pooling.
11. Agregar segunda capa Convolucional 2D con un kernel de tamaño 3x3, función de activación ReLu y operación Max pooling.
12. Agregar capa completamente conectada con función de activación ReLu.
13. Agregar capa dropout 0.5 para evitar overfitting.
14. Agregar capa Output con función de activación Softmax.
15. Visualizar la arquitectura de modelo con visualkeras.
16. Compilar el modelo con un optimizador adam, función de pérdida categorical cross entropy y métricas de tipo accuracy.
17. Entrenar el modelo con 100 épocas, un batch de 128 y verbose = 1.
18. Evaluar el modelo con el conjunto de datos de prueba.
19. Obtener pérdida y precisión del modelo.

Evaluación del modelo entrenado

Parámetro	Valor
Tamaño del kernel	9
Tamaño de la capa Max Pooling	4
Cantidad de nodos en la capa 1	32

Cantidad de nodos en la capa 2	64
Épocas	50
Tamaño de batch	128
Precisión promedio	92.25%
Pérdida promedio	45.48%

Conclusiones del método con la base de patrones utilizada.
Al usar este modelo, obtuvimos un buen resultado, en algún momento se llegó a pensar que había overfitting, por lo que se añadió una capa de dropout para reducir este overfitting.

Podemos concluir que el modelo tiene un buen porcentaje de precisión, aunque la pérdida promedio no es de lo mejor, ya que es de casi un 50%, pero consideramos el modelo aceptable-bueno para la clasificación de las imágenes.

Particularidades del método con el dataset utilizado
Es un dataset de clasificación de moda, por lo que tiene clases de ropa, no se presentó alguna particularidad en sí, es una base de datos muy usada para entrenar modelos de aprendizaje supervisado, como es este caso. Es un dataset muy práctico, ya que, al estar implementado en la librería keras, no se necesita descargar directamente en el equipo para poder cargarlo.
Finalmente, se encuentra ya dividido con un train y test set y 10 clases o etiquetas, por lo que facilita su uso.

Comportamiento para predecir la categoría
Gracias a que el modelo tenía una buena precisión, no hubo demasiados errores ni complicaciones al realizar la predicción de las categorías y presentaba una cantidad mínima de ruido.

Modelo 2: Capsule Neural Network (nuevo)

Descripción
La arquitectura de redes de tipo Capsule Neural Networks, es bastante reciente e interesante. Fue propuesta y desarrollada por Geoffrey Hinton, uno de los padrinos del aprendizaje profundo.

En 2011, Hilton publicó en el artículo “Transforming Autoencoders” las principales características del modelo y los beneficios que este tendría por encima de los demás modelos de redes existentes, esto son:

- Unidades más robustas que las neuronas para manejar la perturbación de datos de mejor manera (cápsulas).
- Procesamiento de imágenes por características súper particulares. Es decir, detectar el borde de un ojo o una punta de una nariz, y no toda la cara a la vez.
- Entendimiento de la posición y orientación de las características que detecta. Esto para resolver el “Problema de Picasso”.
- Enrutamiento Dinámico: En lugar de enviar la activación de una capa a la siguiente (como se hace tradicionalmente), se utiliza un proceso de enrutamiento dinámico para determinar cómo se conectan las cápsulas entre sí.

A pesar de que era un modelo bastante interesante e innovador, no se hicieron muchos avances durante esos años, hasta que en 2017, Hilton y su equipo de trabajo pudieron establecer un mecanismo confiable de ruteo por acuerdo iterativo, que permitía a una cápsula activa enviar su output a cápsulas superiores que su vector tuviera un producto escalar mayor a la predicción de la cápsula inferior.
Así, se lograba estimar el parámetro de instancia más probable para cada cápsula superior, y permitir una mayor armonía entre unidades de procesamiento.

En el artículo de 2017 "Dynamic Routing Between Capsules" de Sara Sabour, Nicholas Frosst y Geoffrey E. Hinton, se demostró que las Capsule Networks (CapsNets) superan a las redes neuronales tradicionales en la clasificación de dígitos, utilizando el conjunto de datos MNIST. La principal mejora se centró en identificar dígitos superpuestos, lo que resalta su utilidad en la detección de objetos en imágenes, incluso cuando están rotados, en diferentes fondos o colores. Las CapsNets ofrecen un enfoque prometedor para la visión por computadora en entornos complejos y variados.

Preparación de datos

En primer lugar, al igual que en el modelo anterior, se importa el conjunto de datos "fashion mnist" desde la biblioteca de datos de Keras Tensor Flow. Esta es una ventaja ya que elimina la necesidad de descargar todo el conjunto de datos directamente a nuestro equipo y nos permite dividir los datos en conjuntos de entrenamiento y prueba de manera conveniente. Una vez que los datos están cargados, llevamos a cabo la transformación mencionada anteriormente. Esto implica la creación de una función que generará un mapa de características utilizando un modelo generativo basado en redes neuronales convolucionales..

En un primer paso, definimos un tamaño de kernel, en este caso, establecimos un valor de 9. Este tamaño de kernel se utiliza para crear un mapa de características reducido. Dado que el kernel es de 9, cada imagen experimenta una reducción de tamaño de 9-1=8 píxeles después de cada capa convolucional. Por ejemplo, una imagen original de 28x28 píxeles se reduce primero a 20x20 píxeles y luego a 12x12 píxeles. Además, en la segunda capa convolucional, utilizamos un paso (stride) de 2, lo que significa que el tamaño de la imagen se divide por 2 en cada dirección. Esto nos lleva a obtener mapas de características finales de tamaño 6x6.

Luego, realizamos una modificación en la salida para obtener un conjunto de vectores de 8 dimensiones que representan las salidas de las cápsulas primarias. La salida de la capa convolucional "conv2" es una matriz que contiene 256 mapas de características para cada instancia de datos, donde cada mapa de características tiene un tamaño de 6x6 píxeles. Por lo tanto, la forma de esta salida es (tamaño del lote, 6, 6, 256).

Nuestro objetivo es dividir esos 256 valores en 32 vectores, cada uno con 8 dimensiones. Para lograr esto, podríamos reorganizar la salida a una forma como (tamaño del lote, 6, 6, 32, 8), pero dado que esta primera capa de cápsulas se conectará completamente a la siguiente capa de cápsulas, optamos por aplanar las cuadrículas de 6x6. Esto significa que al final obtendremos una forma de (tamaño del lote, 6x6x32, 8)

Finalmente, en la fase de transformación de datos, se emplea la función de "squash" para procesar la salida de las cápsulas con el objetivo de reducir el ruido y mejorar la calidad de la representación. La función "squash" tiene una forma que se asemeja a una curva y va desde el valor máximo (1) hasta el valor mínimo (0), atravesando valores intermedios. Esta función se aplica a la salida de cada cápsula después de que esta haya activado su respuesta y antes de que esa salida se transmita a las cápsulas superiores en la red.

Implementación y entrenamiento

Notebook: **CapsNetsProyecto.ipynb**
La secuencia de pasos realizados se enumera a continuación:

1. Importar todas las librerías necesarias.
2. Cargar los datos de “*fashion mnis*t” desde TF Keras.
3. Verificar que se han cargado los datos correctamente.
4. Hacer la transformación de datos de entrada en mapas de características (primer capa / cápsulas primarias).
5. Aplicar la función *squash*.
6. Construir las demás capas deseadas mediante la predicción de los vectores de salida anteriores (uno para cada pareja de cápsulas primarias/ de prenda de ropa).

7. Aplicar el ruteo por acuerdo, aplicar la función softmax para calcular los pesos de enrutamiento.
8. Generar las probabilidades estimadas de cada clase. Esto para predecir la clase de cada instancia, ya que ahora podemos seleccionar el label que tenga la probabilidad estimada más alta, y esa será nuestra predicción.
9. Ya casi estaríamos terminando, porque ya obtuvimos lo que queríamos: para cada instancia, ahora tenemos el índice del vector de salida más grande. Ahora, eliminamos las dos últimas dimensiones usando tf.squeeze(), que elimina las dimensiones de tamaño 1. Esto nos da la clase predicha para cada instancia.
10. Agregar ahora un decoder de 3 capas densas totalmente conectadas (ReLU, ReLU, sigmoide) que aprenderá a reconstruir las imágenes de entrada en función de la salida de la red de cápsulas.
    a. Esto obligará a la red de cápsulas a preservar toda la información necesaria para reconstruir los dígitos en toda la red. Esta restricción regulariza el modelo: reduce el riesgo de overfitting y ayuda a generalizar a nuevas prendas de ropa que puedan incluirse.
    b. El paper menciona que durante el entrenamiento, en lugar de enviar todas las salidas de la red de la cápsula a la red del decoder, debemos enviar solo el vector de salida de la cápsula que corresponde al objetivo.
11. Una vez ejecutada toda esta transformación, se dividen los datos de validación.
12. Finalmente se agregan los últimos retoques y elementos necesarios: medidas de precisión, pérdida, tamaño de batches, épocas, etc.

Entrenar nuestra red de cápsulas es bastante estándar. Para simplificar, no haremos ningún ajuste complicado de hiper parámetros, *dropout* ni nada parecido, simplemente ejecutaremos la operación de entrenamiento una y otra vez, mostrando la pérdida, y al final de cada época, mediremos la precisión en el conjunto de validación.

Evaluación del modelo entrenado

Nuestros resultados se resumen en la siguiente tabla:

Parámetro	Valor
Tamaño de kernel	9
Cantidad de mapas en la capa 1	32
Cantidad de cápsulas en la capa 1	6
Épocas	12
Tamaño de batch	18
Precisión promedio	88.25%

Pérdida promedio	0.096%

Se puede apreciar que los resultados alcanzados fueron bastante positivos, y que definitivamente este tipo de red tiene mucho potencial.

Conclusiones del método con la base de patrones utilizada.

Al ser un método que investigamos por nuestra cuenta, concluimos que se adaptó correctamente, se obtuvieron muy buenos resultados, el modelo predice con 88.25% de precisión y una pérdida promedio de 0.096%, por lo que a pesar de no tener una precisión tan alta, no hay una cantidad grande de pérdida, por lo que podemos concluir que es una buena adaptación del método al dataset Fashion MNIST.

Particularidades del método con el dataset utilizado

Una particularidad del dataset utilizado en este método fue la transformación de los datos, ya que, para poder emplear el método correctamente por cápsulas, debimos decidir bien el cómo aplicar la transformación de los pixeles de las imágenes para que funcionara correctamente y de igual manera con el tamaño del batch.

Comportamiento para predecir la categoría

El método se comporta correctamente al predecir la categoría, al hacer una visualización de las imágenes correctas para la clase y las predichas, se observa que hay un poco de ruido en las predicciones, sin embargo, el modelo predice de buena manera y clasificando correctamente.

Comparativa de modelos

¿Cuál fue el más latoso? ¿por qué?

Considero que el modelo que de cierta forma ocasionó más complicaciones fue el de CapsNet, esto debido a que era un modelo investigado por nuestra cuenta y presentaba partes en la arquitectura y preparación de los datos que tuvimos que comprender más allá de los modelos previos que teníamos, aspectos como las cápsulas digitales, los vectores de salida y el ruteo por aceptación. A diferencia de un método por Redes Neuronales Convolucionales que ya teníamos una idea de cómo trabajaba y presenta una arquitectura más sencilla.

¿Cuál se entrenó mejor?

Tomando en cuenta la cantidad de tiempo que tomó entrenar el modelo, podemos decir que CapsNet tuvo un mejor entrenamiento, ya que, tardó menos tiempo en entrenarse y obtuvo mejores resultados.

¿Cuál tuvo mejor desempeño de acuerdo con las métricas seleccionadas?
El método por Redes Neuronales Convolucionales tuvo un mejor desempeño en el aspecto de la precisión, sin embargo, el CapsNet tuvo una pérdida promedio mucho menor al otro, por lo que concluimos que el modelo CapsNet tuvo un mejor desempeño al no tener una diferencia tan grande en el porcentaje de precisión de las predicciones.

Observaciones sobre tiempos y recursos requeridos.
Pudimos observar que, al menos en el caso de mi computadora, el tiempo por época para el CNN fue en promedio de 65 segundos, a 50 épocas, tardó aproximadamente una hora en entrenarse, mientras que el CapsNet utilizando el GPU tardó únicamente 10 minutos.

Conclusiones

Posibles mejoras de cada método y en general
Empezando con el método CNN, considero que su principal punto de oportunidad es en el aspecto de la pérdida promedio, ya que, un 45%, que fue el resultado obtenido, no es de lo mejor, además del tiempo de entrenamiento, tal vez se pueden hacer mejores modificaciones en la arquitectura para obtener un mejor resultado.

Con respecto al CapsNet, su área de oportunidad se encuentra en el porcentaje de precisión, ya que, si bien un 88% no es malo, tampoco es lo mejor, por lo que se podrían hacer modificaciones para aumentar este porcentaje sin aumentar demasiado la pérdida promedio.

En general se obtuvieron buenos resultados con ambos modelos, cada uno tuvo un área de oportunidad diferente, lo que es bueno, porque nos ayuda a ver cómo se comportan los diferentes métodos con un mismo objetivo y una misma base de datos.

Conclusiones generales (¿cuál método fue el mejor?)
Concluimos que el mejor método para realizar la clasificación fue el CapsNet, ya que, a pesar de tener un menor porcentaje de precisión, tiene un mejor balance con la pérdida promedio, no es alta, ni siquiera de 1%, además de que el tiempo de entrenamiento fue menor.
Por eso, recomendamos el modelo por Capsule Networks antes que el CNN para la clasificación del dataset Fashion MNIST.

¿Qué aprendiste del proyecto?
Del proyecto aprendí a poder identificar qué tipos de métodos con Redes Neuronales nos son más útiles para los diversos casos que se nos van a presentar, como fue este con respecto a la clasificación de imágenes.

De igual manera el entender más a fondo la arquitectura de los modelos y el cómo los cambios afectan en los resultados finales.

¿Qué fue lo que más te gustó?

En general, es una materia muy interesante y de gran ayuda para nuestra carrera, por lo que el ponerlo en práctica para el proyecto y el tener la oportunidad de buscar métodos más allá de los vistos en clase fue algo de lo que más me gustó, el poder explorar las diversas opciones que hay sin estar atados a una, pero de igual manera poder comparar con las que vimos.

¿Qué fue lo que menos te gustó?

Lo que menos me gustó y supongo que es algo muy común es el tiempo que tardan en entrenarse los modelos, ya que, eso nos limita un poco al querer explorar una diversa cantidad de opciones y cambios, porque cuando un modelo ya es lo bastante complejo, ciertos cambios que hacer para buscar mejorar los resultados podrían tardar horas o hasta días.

Referencias

1. Hinton, G. E., Krizhevsky, A., & Wang, S. D. (2011). Transforming auto-encoders. In Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21 (pp. 44-51). Springer Berlin Heidelberg.
2. Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules (Versión 2). arXiv. https://doi.org/10.48550/ARXIV.1710.09829
3. Sabour, S., Tagliasacchi, A., Yazdani, S., Hinton, G. E., & Fleet, D. J. (2020). Unsupervised part representation by Flow Capsules (Versión 2). arXiv. https://doi.org/10.48550/ARXIV.2011.13920
4. Fashion mnist. (s. f.). Recuperado 9 de septiembre de 2023, de https://www.kaggle.com/datasets/zalando-research/fashionmnist