Forma

Descripción generada automáticamente con confianza media

**Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México**

**Escuela de ingeniería y ciencias**

[Inteligencia artificial avanzada para la ciencia de datos II (Gpo 501)](https://experiencia21.tec.mx/courses/315342)

# **Momento de Retroalimentación: Módulo 2 Implementación de un modelo de deep learning.**

Andrea Vianey Díaz Álvarez A01750147

**Profesor:**

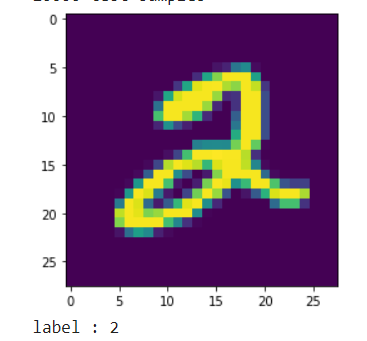
### Julio Arriaga

**Fecha: 02/11/2022**

# **Análisis del modelo CNN**

**Descripción de las imágenes:**

* Cada una de las imágenes contiene un número entre 0 y 9.
* 6000 imágenes de entrenamiento y 1000 imágenes de prueba.
* *Ejemplo:*



**Tamaño del Vecindario:** (28,28)

**Profundidad:** 1

**Arquitectura CNN Típica:**

Diagrama

Descripción generada automáticamente

Basándome en la arquitectura típica, he decidió empezar a cambiar algunos parámetros de cada capa donde:

1. Las capas de **convolución** tienen los parámetros:
   1. Número de kernels tridimensionales (k)
   2. Tamaño de los kernels (F x F)
   3. Tamaño del paso del Kernel (stride)
   4. Tamaño del padding
   5. Función de activación
2. Las capas de **pooling** tienen los parámetros:
   1. Tamaño del Pool (F x F)
   2. Tamaño del paso del Pool (stride)
3. Las capas de **dropout** tienen el parámetro:
   1. Rate entre 0 y 1: % de neuronas que se va a desactivar
4. Las capas de **dense o fully connected** tienen el parámetro:
   1. Función de activación

**Prueba 1:**

* Epochs: 20
* Batch size: 128

Texto

Descripción generada automáticamente con confianza baja

Resultados

Último epoch:

* loss: 0.5334
* sparse\_categorical\_accuracy: 0.8259
* mean\_absolute\_error: 4.3737

Evaluación:

* loss: 0.20601679384708405
* acc: 0.9477999806404114
* mae: 4.3629984855651855

**Prueba 2:**

Para la segunda prueba he decidido cambiar significativamente el número de kernels

* Epochs: 30
* Batch size: 128

**Texto

Descripción generada automáticamente**

Resultados

Último epoch:

* loss: 0.0078
* sparse\_categorical\_accuracy: 0.9971
* mean\_absolute\_error: 4.3737

Evaluación:

* loss: 0.03345528244972229
* acc: 0.991100013256073
* mae: 4.3629984855651855

**Prueba 3**

En la siguiente prueba cambié el tamaño del kernel por el del Pool para ver cómo afectaba esto al modelo.

* Epochs: 30
* Batch size: 128

Texto

Descripción generada automáticamente con confianza media

Resultados

Último epoch:

* loss: 0.4605
* sparse\_categorical\_accuracy: 0.8565
* mean\_absolute\_error: 4.3737

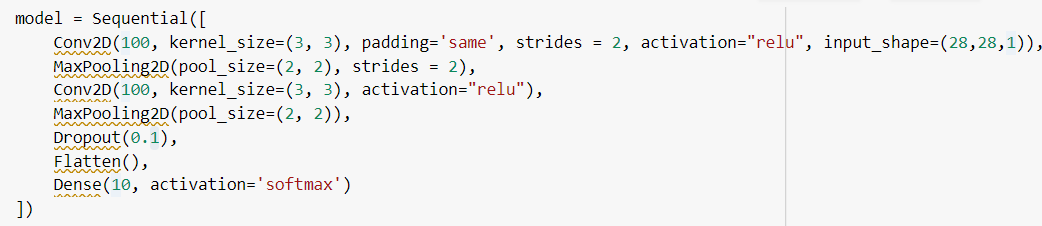
Evaluación:

* loss: 0.30019697546958923
* acc: 0.9103999733924866
* mae: 4.3629984855651855

**Prueba 4**

Para esta prueba cambie el valor del Dropout a uno mucho más pequeño.

* Epochs: 30
* Batch size: 128



Resultados

Último epoch:

* loss: 0.0101
* sparse\_categorical\_accuracy: 0.9970
* mean\_absolute\_error: 4.3737

Evaluación:

* loss 0.03340237960219383
* acc 0.9908000230789185
* mae 4.3629984855651855

**Prueba 5**

Cambiando el valor de strides a 1

Texto

Descripción generada automáticamente

Resultados

Último epoch:

* loss: 0.0033
* sparse\_categorical\_accuracy: 0.9988
* mean\_absolute\_error: 4.3737

Evaluación:

* loss 0.040756337344646454
* acc 0.9908999800682068
* mae 4.3629984855651855

**Análisis de Resultados**

Empecé hacer mis pruebas tomando como base la arquitectura típica además de esto, los parámetros que escogí fueron pensando en lo que probablemente iba conseguir los mejores resultados en este caso, tomando en cuenta el tamaño de mis imágenes, cantidad y los colores de las imágenes.

Para resolver el problema de los bordes que se puede poner el kernel por fuera de las imágenes decidí intentar con 2 soluciones: “valid” que coloca el kernel únicamente en ventanas válidas y “same” que aplica un relleno con ceros. En este caso no hubo diferencia, ya que, en general los números dentro de las imágenes están muy centrados y el padding no afecta notoriamente el resultado.

Para el desplazamiento del kernel, empecé usando un valor de 2, sin embargo, en la última prueba decidí cambiarlo a 1 para ver si esto lograba mejorar mi modelo y hacerlo un poco más exacto. En general el modelo había tenido excelentes resultados con un valor de 2, se logró mejorar el accuracy un poco, pero al hacer la evaluación hasta salió un poco más bajo, por lo tanto, optaría por dejarlo en 2 para tener un modelo más eficiente, ya que no obtuvo ningún cambio significativo.

Una de las pruebas que hice fue hacer una comparación para ver como afecta tener un número bajo de kernels. Primero lo probé con 10 y después con 100. Y este fue el cambio más significativo que pude observar ya que al tener un número muy bajo de kernels tuvo resultados bajos en comparación a un número alto de kernels, esto se debe a que mientras más kernels el modelo tiene más peso que aprender y por lo tanto tendrá mejores resultados.

Por último, también dando resultados significativos fue el cambiar el tamaño tanto del kernel como del pool. En el mejor caso fue tener el kernel con tamaño 3 x 3 y el pool con tamaño 2 x 2 pero para hacer la comparación intercambié los valores. Esto se debe a que el objetivo de las capas de pooling es disminuir las dimensiones y estoy trabajando con volúmenes de dimensión pequeña.

**Conclusión**

Ya que decidí partir de una arquitectura que funciona para muchos casos, tuve resultados positivos desde el principio y las pruebas que hice fueron para ver si podría obtener aun mejores resultados y de que manera podrían afectar los parámetros a mi modelo, en muchos casos es prueba y error hasta obtener los resultados deseados e investigar para poder tener mejor conocimiento de que el lo que mejor funciona para el modelo y los datos que se están utilizando. En el futuro me gustaría hacer más pruebas y cambiar más las variables para aprender a hacer modelos eficientes para cualquier conjunto de imágenes.