

# Apuntes De Clase del 22/05/2025 Inteligencia Artificial

Manuel Alejandro Rodríguez Murillo

## I. INTRODUCCIÓN

EN la clase del 22 de mayo de 2025 se abordaron múltiples conceptos y herramientas. Se explican distintas arquitecturas de autoencoders, incluyendo variantes como el undercomplete, sparse, denoising y variational autoencoder, junto con sus aplicaciones. Además, se profundiza en la quantization como método para optimizar modelos, detallando sus tipos, ventajas, procesos matemáticos y estrategias para minimizar errores. Además, se discutieron avances recientes presentados en eventos tecnológicos como Google I/O, así como herramientas emergentes como Veo 3 y LangChain.

## II. NOTICIAS

### A. Google I/O

Se habló sobre diferentes tecnologías presentadas en la conferencia, entre esto se habla sobre el avance de la realidad aumentada y Veo 3 que es una herramienta de generación de video basada en inteligencia artificial. También se habló un poco sobre la conferencia de Microsoft y LangChain el cual sirve para conectar un LLM.

## III. REPASO

### A. Encoder

- Conjunto de bloques convolucionales seguidos de módulos de pooling.
- Encargada de extraer las características principales de la imagen de entrada y comprimir la información.
- Expectativas: Aprender información importante de la entrada.
- Downsampling.

Al final lo que buscamos es realizar una reducción de nuestro dato en un vector. Este vector se vuelve una representación mucho más pequeña de nuestra información inicial.

### B. Cuello botella

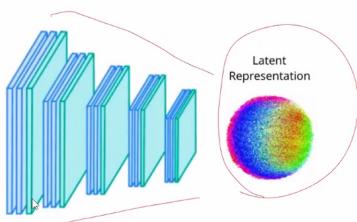


Figura 1: Cuello de botella

- Parte más importante y pequeña del modelo
- Representación en espacio latente
- Restringe el flujo de información proveniente del encoder al decoder.

Un ejemplo de como se ve la representación de los datos conforme se van haciendo las iteraciones:

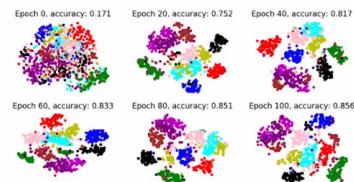


Figura 2: Representación

### C. Decoder

La idea después de lo anterior es que con otras operaciones a partir de un vector se reconstruya la imagen y esta debe ser lo más parecida posible a lo que se tenía en el Encoder.

- Conjunto de convoluciones que hacen upsampling.
- Reconstruyen la imagen basada en el vector latente.

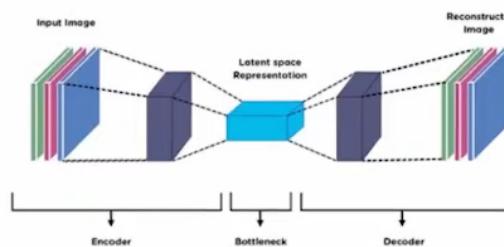


Figura 3: Encoder and Decoder

### D. Hiper parámetros a considerar

#### Tamaño de la codificación.

- Decide cuanto se comprime los datos

#### Número de capas

- Profundidad del encoder y del decoder
- Mayor=Modelo complejo
- Menor=Modelo rápido

La profundidad es decisión del programador solo considerar que, entre mayor modelo más complejo y más difícil de entrenar, y menor modelo más rápido.

#### Función de reconstrucción

- Altamente dependiente de la entrada y salida esperada
- MSE, reconstrucción de imágenes
- BinaryCrossEntropy, valores en rangos [0,1]

Para tratar de reconstruir nuestra imagen se utiliza MSE para determinar que también construida esta nuestra aplicación.

### E. Undercomplete Autoencoder

Disminuir la dimensionalidad para aumentarla después para reconstruir la imagen.

- Toma una imagen de entrada e intenta predecir la misma imagen de salida.
- La dimensionalidad del espacio latente es menor que el espacio de entrada.

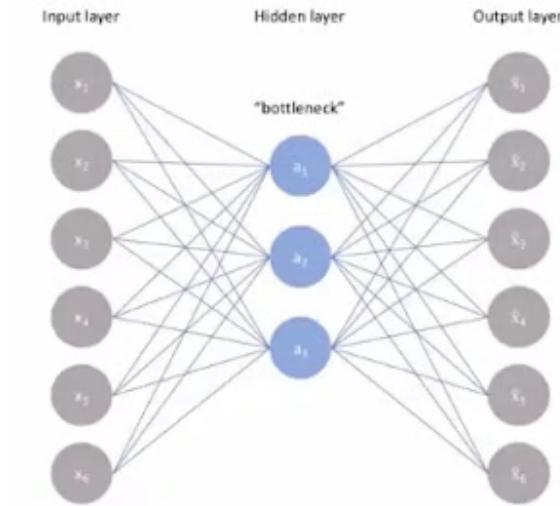


Figura 4: Undercomplete

### F. Sparse Autoencoder

Mantiene las mismas dimensiones, pero se van apagando neuronas para que vaya aprendiendo a reconstruir la información de la imagen durante toda la arquitectura. **NEVITA la reducción de dimensionalidad en las capas ocultas**

**Se penalizan activaciones para simular dinámicamente esta reducción**

- Aplican factores de regulación a la función de pérdida.
- Activan y apagan neuronas de la red.

**Selectivamente activa regiones de la red dependiendo de la entrada.**

- Toma una imagen de entrada e intenta predecir la misma imagen de salida.
- La dimensionalidad del espacio latente es menor que el espacio de entrada.

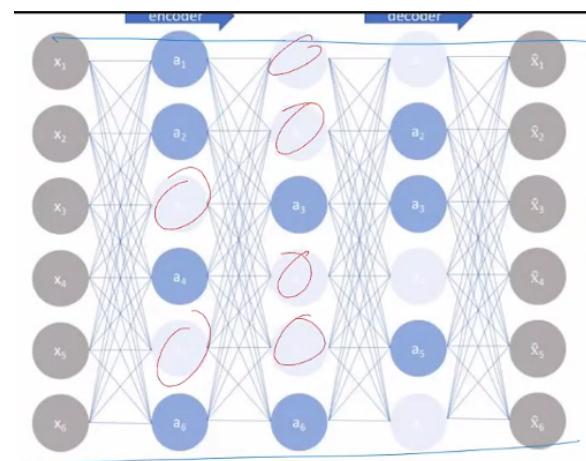


Figura 5: Sparse

### G. Denoising Autoencoder

Hace que mi autoencoder elimine el ruido y en general hace una tarea específica que me sea útil, como por ejemplo Super-resolution.

### H. Variational Autoencoder

Ese vector latente que vamos a construir siguiera una distribución normal o siguiera una distribución del todo para poder nosotros interpolar ciertos valores que hay en esa distribución y obtener nuevos datos.

Por ejemplo, cuando tenemos un vector latente como tal lo que tenemos es cada una de las representaciones o features semánticos.

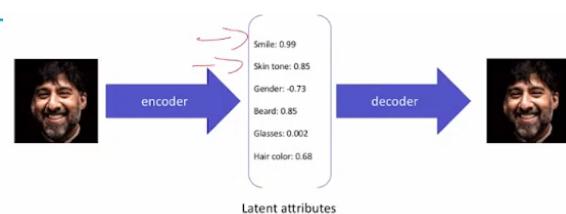


Figura 6: Atributos

Sí esos features siguieran una distribución de densidad, van a seguir una probabilidad entonces de esta forma vamos a tratar es de extraer valos que modifiquen la imagen. Las cuales el autoencoder nunca ha visto en su entrenamiento, pero aprendió a interpolar.

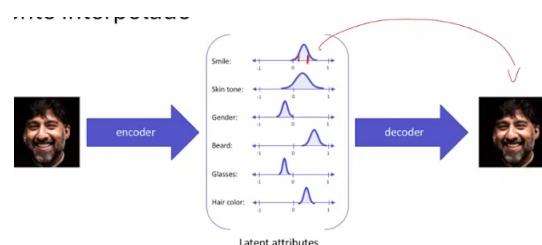


Figura 7: Distribución de densidad

Eso hace que podamos tener estos resultados:

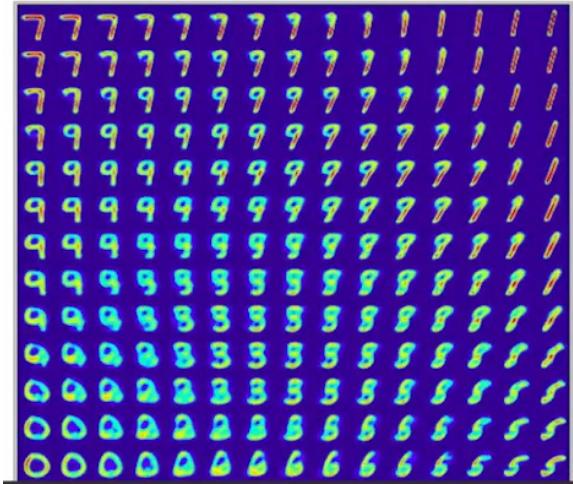


Figura 8: Ejemplo resultado

Nuestro Encoder debe producir un  $\mu$  y  $\sigma$  a partir de la imagen. Cuando el empieza a extraer características aplicando convoluciones al final debe llegar a extraer un  $\mu$  y  $\sigma$ . Con esto se hace la distribución normal  $\mathcal{N}(\mu, \sigma) + \varepsilon$

Y de ahí se hacen dos cosas, primero evaluar la reconstrucción de la imagen que básicamente es el MSE comparando los pixeles de entrada con los de salida y también nos dice que también se reconstruyó la imagen. Lo otro es que debo calcular la KL Divergence que es la divergencia que existe entre una distribución que sigue  $\mathcal{N}(0,1)$  (a esta me intento ajustar) y otra distribución que sigue  $\mathcal{N}(\mu, \sigma)$  (de esta está intentando aprender) lo que permite ver es como a que distancia esta la distribución de seguir a la normal estándar.

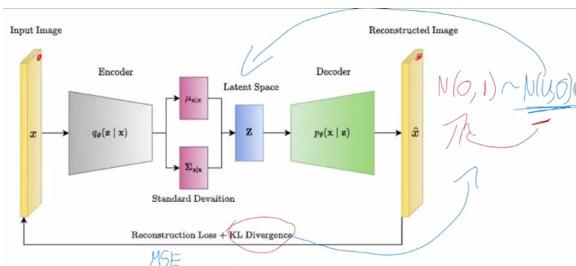


Figura 9: Distribuciones

Ejemplo, se tienen dos poblaciones y al utilizar el KL podemos ver divergencia entre ellas y la distribución normal estándar, entre mejor se ajuste los datos a la distribución menor será la diferencia entre ellas. Se busca ajustar la  $\mu, \sigma$  para que se parezca lo más posible a la distribución y así se resuelve este tipo de problemas.



Figura 10: Ejemplo distribuciones

## I. U-Net: Convolutional Networks for Biomedical Image Segmentation

En este caso la tarea es un poco diferente, se debe tratar de hacer segmentación de imágenes. Por ejemplo, si se tuviera una foto de un perro y un gato que se resalte donde esta el gato y donde esta el perro en la misma imagen.

La idea es producir esa segmentación de mapas y con U-Net se hace muy similar siguiendo un autoencoder normal a excepción de que tiene skip connections.

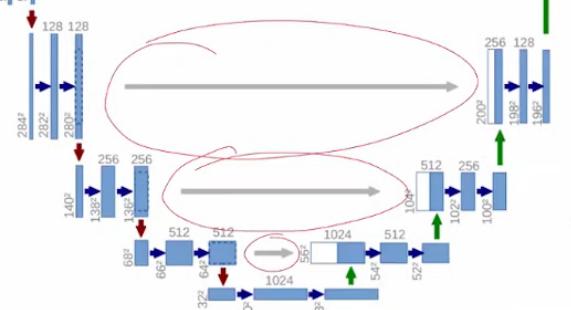


Figura 11: Skip connection

## Ejemplos segmentación de imágenes:



Figura 12: Ejemplo segmentación de imágenes 1



Figura 13: Ejemplo segmentación de imágenes 2

## IV. QUANTIZATION

### A. ¿Qué es quantization?

Los modelos cuentan con muchísimos parámetros, pero esos modelos se entran y se especifican para una sola tarea. Modelos más grandes como los de lenguaje son gigantes y tienen una cantidad de parámetros muy grande, por ejemplo, LLaMA 2 tienen 70 mil millones de parámetros y se cada parámetro fuera de 32 bits, estaríamos hablando de 28GB que tenemos que ver como lo cargamos, lo cual es bastante caro y eso solo es almacenamiento de disco. Cargarlo en memoria no podría ser realizado por una computadora convencional. Y también las computadoras son lentas para las operaciones de punto flotante.

El quantization propone es disminuir el uso de bits que nosotros tenemos para representar los parámetros y eso lo vamos a lograr cambiando la representación de punto flotante que tenemos ahora a una representación de enteros con esto vamos a tratar de hacer el modelo mucho más pequeño y si normalmente estamos a costumbres a que un punto flotante sea de 32 bits vamos a tratar de reducirlo a 8 bits. Al hacer esto se va a perder precisión en la pasada hacia delante de la red, pero esto es un precio a pagar para tener un modelo mucho más veloz y que pueda responder de forma más optima.

Estas compresiones se pueden realizar en diferentes tamaños de bits:

- 8 bit
  - 5 bit
  - 2 bit
  - 1 bit

## *B. Aclaraciones*

- Quantization no significa aplicar técnicas de redondeo a nuestros pesos y convertirlos a enteros.
  - La técnica tiene otros cálculos.

### **C. Ventajas**

### **Menor consumo de memoria**

- Al momento de cargar los modelos
  - Se puede usar devices

### **Menor tiempo de inferencia**

- Datos simples

### **Menor consumo de energía**

- Menos computación más barato

Generalmente se representan los números en bloques de 8 bits

- 16 bits / 2 bytes (short)
  - 32 bits / 4 bytes (integer)
  - 64 bits / 8 bytes (long)

#### D. Complemento a 2

La mayoría de los enteros están representados utilizando complemento a 2. Este es aquel donde el primer bit me decía cuál era el signo 0+, 1-. La ventaja de esto es que puedo tener una única representación del 0. Para representar punto flotante básicamente incluye números elevados al negativo de la base.

$$85,612 = 8 \times 10^1 + 5 \times 10^0 + 6 \times 10^{-1} + 1 \times 10^{-2} + 2 \times 10^{-3}$$

Figura 14: Ejemplo representación punto flotante

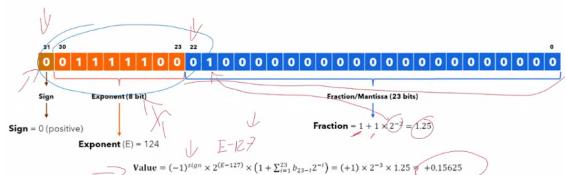


Figura 15: Calculo flotante

## *E. Redes neuronales – Pesos*

Matrices de pesos son representadas con punto flotante y la idea nuestra es tratar de no perder calidad en los resultados cuando hagamos quantization.

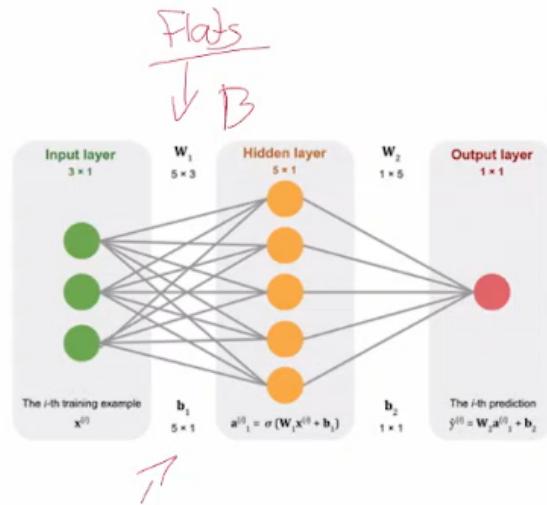


Figura 16: Matriz

Lo queremos hacer es obtener operaciones enteras. Entonces para sacar el máximo provecho el cpu para hacer operaciones en enteros lo que debemos transformar a enteros es el w (los pesos de la red) por consecuente el x (que es mi entrada) y también el b.

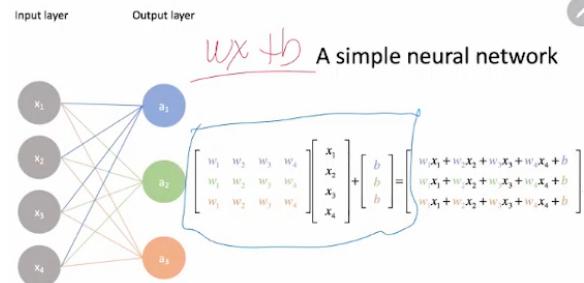


Figura 17:  $wx+b$

Entonces transformamos el x,w y b las salidas serian integers, si estos datos se deben pasar a otra capa se deben volver a transformar a flotante. El principio es que la capa no debe darse cuenta de que los cálculos hechos en las capas anteriores fueron hechos con enteros y así sucesivamente con todas las capas, a este proceso se le llama dequantization.

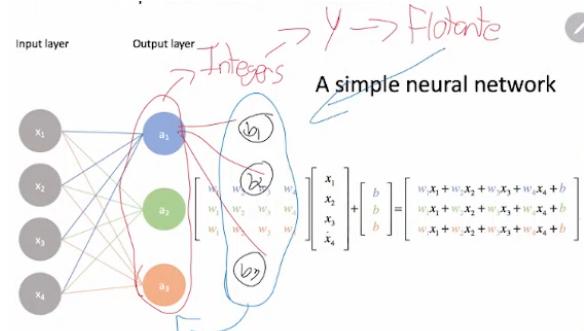


Figura 18: Dequantization

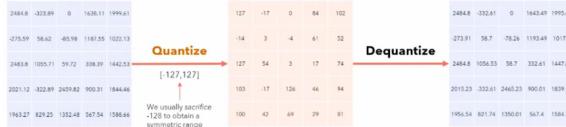


Figura 19: Quantization process

## V. QUANTIZATION SIMÉTRICA Y ASIMÉTRICA

En el caso de Asimétrico podemos ver que no hay números negativos, solo hay valores entre 0 y el mayor número posible con respecto a los bits que este aplicando, entonces mis valores cuantizados van a estar en ese margen. Tengo el valor 0 sin cuantizar, en ese caso se va a parametrizar a un valor específico Z, ese Z va a ser mi nuevo 0 o la forma que tengo para representar el 0.

El simétrico si tiene valores positivos y negativos. El 0 se mantiene en la posición del 0 y todos los demás valores del tensor que estemos transformando se van a transformar en positivos o en negativos.

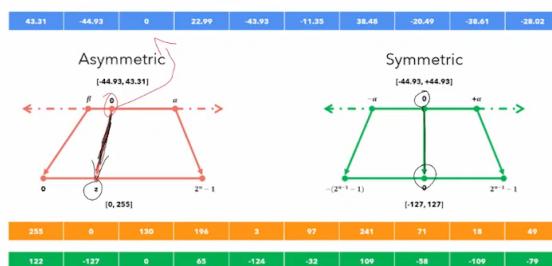


Figura 20: Asimétrico y Simétrico

### A. Quantization Asimétrico

Nosotros vamos a tratar de mapear los valores en un rango específico, ese rango va a tener el 0 como mínimo hasta el  $2^n - 1$  (que es el total de valores que puedo representar usando los n bits). Para esto debo sacar un  $\alpha$  (Alpha) y un  $\beta$  (Beta).

Mapear valores entre un rango  $[\beta, \alpha]$  a un rango  $[0, 2^n - 1]$

- $\beta$  = el número menor de los valores a convertir.
- $\alpha$  = el número mayor de los valores a convertir.

Para convertir mis valores de mi tensor a un vector cuantizado. Lo primero es conseguir S y Z.

S = Factor de parametrización (en cuanto voy a escalar los datos).

- S = Factor de parametrización (en cuanto voy a escalar los datos).

$$S = \frac{\alpha - \beta}{2^n - 1}$$

$2^n - 1$  = El rango de salida

Figura 21: Formula S

- Z = Es el desplazamiento con respecto al 0.

$$z = \left\lfloor -1 \cdot \frac{\beta}{S} \right\rfloor$$

Figura 22: Formula Z

- n = El número de bits

$$\bullet x_q = clamp \left( \left\lfloor \frac{x_f}{S} \right\rfloor + z; 0; 2^n - 1 \right)$$

$x_f = Valor\ fijoante$

Figura 23: Formula general asimétrico

Con respecto a la formula voy a cotar mis valores en 0 y  $2^n - 1$ . Eso quiere decir que si al momento de ejecutar esa operación me da un valor negativo como mínimo voy a colocar 0 y si me da un valor mayor al  $2^n - 1$  voy a colocar  $2^n - 1$  que es mi mayor valor.

### B. Dequantization Asimétrico

Para hacer el dequantization se hace el proceso inverso. Si en el otro sumábamos el neutro aquí se lo restamos al valor ya cuantizado y esto nos va a introducir un grado de error.

$$x_f = s(x_q - z)$$

Figura 24: Formula dequantization asimétrico

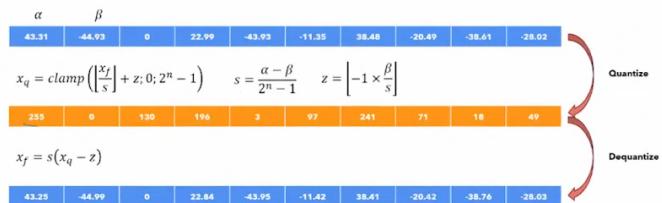


Figura 25: Quantization y Dequantization asimétrico

### C. Quantization Simétrico

En este caso voy a tomar un valor de  $\alpha$  y voy a tratar de dar la salida entre  $[-\alpha, \alpha]$ .  $\alpha$  en este caso es el mayor valor de manera absoluta que exista en el tensor y el rango que voy a dar es de  $[-(2^n - 1), (2^n - 1)]$

$$x_q = clamp \left( \left\lfloor \frac{x_f}{S} \right\rfloor; -(2^{n-1} - 1); 2^{n-1} - 1 \right)$$

Figura 26: Formula general simétrico

- S = Parámetro de escalado

$$S = \frac{abs(\alpha)}{2^{n-1} - 1}$$

Figura 27: Formula S simétrico

- n = Número de bits

### D. Dequantization Simétrico

El dequantization es aún más sencillo, solo debo multiplicar por el escalador, ya no tengo un parámetro neutro

$$x_f = s x_q$$

Figura 28: Formula dequantization simétrico

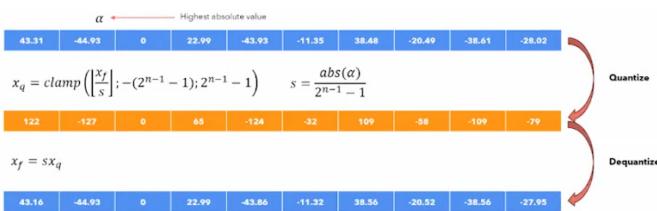
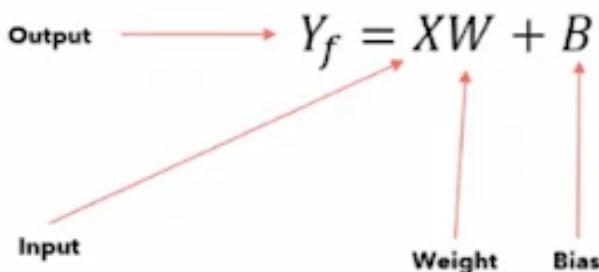


Figura 29: Quantization y Dequantization Simétrico

## VI. DYNAMIC QUANTIZATION

Ya tenemos el valor del W y del B pero necesitamos X pero esos valores empiezan a variar ese quantization lo puedo calcular de forma dinámica cada que el valor pasa por  $Y = XW + B$

Figura 30:  $y=xw+b$ 

Transforma a enteros la entrada de la capa. El otro reto es que yo debo hacer dequantization una vez que sale de cada capa.

### A. Calibration

**¿Como lo hacemos si no sabemos cuál es su parámetro escalador ni su 0?**

Nosotros no podemos saber eso porque varia dependiendo de la entrada o la clase que este de entrada.

Lo que hace es tratar de captura estadísticas típicas en cada una de las capas para poder a partir de esos datos hacer los cálculos de alfas y betas que sean razonables para obtener un escalador y un valor Z.

Transforma a flotantes nuevamente la salida de la capa que está en enteros.

### B. Estrategia de selección del rango

Hay formas de obtener los valores de  $\beta$  y  $\alpha$  para hacer el cálculo de quantization. En el caso asimétrico seleccionamos los valores de los extremos del tensor  $[\beta, \alpha]$

- Es muy sensible a los outliers. Al presentarlos el quantization va estar bien pero el dequantization va presentar diferencia numéricas grandes.



Figura 31: Outlier asimétrico

En el caso simétrico seleccionamos el mayor valor en términos absolutos.  $[-\alpha, \alpha]$  Pero no es la única estrategia de selección por que introducen más grado de error.

### C. Solución Asimétrico

Para intentar solucionarlo se puede usar la distribución del percentil. En lugar de tomar los valores alfa y beta como los mayores y menores, tomamos del todo el vector aquél que me representa el percentil 99% y el percentil 1%. De esa forma yo voy a tratar de acotar ese outlier, reduciendo mucho el error. Esa reducción de error tiene que verse excluyendo el outlier.



Figura 32: Outlier asimétrico percentil

Otra estrategia:

- Tratar de hacer una búsqueda GridSearch entre parámetros de  $\beta, \alpha$  que lo que haga es minimizar el MSE.
- Otras que se aplican para casos de LLM pueden ser Greedy, Top-P y Softmax Layer LLM

### D. Quantization Granularity (Convolución)

Las convoluciones están hechas por muchos de filtros

- Aprendidos con valores y distribuciones distintas para cada uno. Observan distintos features de la imagen. No podemos aplicar el mismo valor de  $\beta, \alpha$  para todos los filtros.
- Mejor buscar valores de  $\beta, \alpha$  para cada filtro.

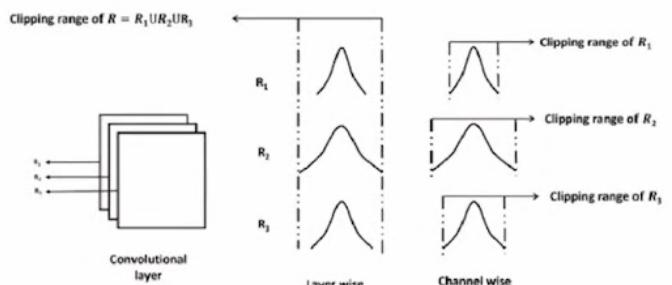


Figura 33: Quantization Granularity

## VII. CONCLUSIONES

Se exploraron conceptos esenciales para el diseño y optimización de modelos de redes neuronales. Se profundizó en el funcionamiento de los autoencoders y sus variantes, incluyendo el sparse autoencoder, denoising autoencoder y variacional autoencoder. Asimismo, se revisó el proceso de cuantización como una técnica crítica para reducir el tamaño de los modelos y mejorar su eficiencia computacional, analizando tanto la cuantización simétrica como asimétrica.