# Apuntes del 8/05/2025
Steven Pérez Machado

***Abstract*—El presente documento son los apuntes realizados en la clase de jueves del 8/5/2025, donde se vieron varios temas, se comenzó con una sección de noticias, un repaso sobre los temas vistos en la clase del martes 6/5/2025 y luego como temas nuevos se estudiaron Visualización del Aprendizaje en Redes Convolucionales, Embeddings, Autoencoder.**

## I. Aspectos sobre el proyecto

- Pythorch no se puede utilizar con gráficas de AMD, a menos que se trabaje con Ubuntu
- Hay que subir los archivos de audios al drive porque de ahí se jalan los datos
- SI se tiene todo el dataset de aumentation y filtros es mucha cantidad de información, por lo eso es mejor guardarlo e irlos procesando conforme se van usando, lo que nos ayuda a no utilizar tanta RAM.
- Un aspecto que se observó es que el problema es sencillo de resolver, porque tener un clasificador de los audios son muy separables, analizando el espectrograma se puede saber qué es, porque si se parecen mucho todas las pronunciaciones de las personas con la palabra que están diciendo y el espectrograma que produce, esto es por un fallo en haber escogido ese dataset y no uno más complejo, y no es de extrañarse que desde un inicio con un modelo muy pequeño dé buenos resultados.
- Puede que cambie los puntos extra porque la mayoría va a obtener puntaje alto en esta parte, entonces una de las posibilidades es que estos puntos se trasladen para el siguiente proyecto, o alguna otra posibilidad para este proyecto

## II. Noticias

• Stride

• Artículo de "A Practical guide to building agents" que menciona como crear nuestro propio agente

## III. Repaso clase del martes 6/05/2025(Arquitecturas convulucionales)

### A. *Composición*

La composición de las redes convolucionales se divide en:

- **Convolutional Layer:** La capas de convolución(Convolutional Layer) que son las encargadas de extraer las características de la imagen
- **Pooling Layer:** Procesamiento que se le hace a la imagen para reducir el tamaño
- **Dense Layer:** Toma las características que extrajimos de la imágenes y genera un clasificador basado en esa características

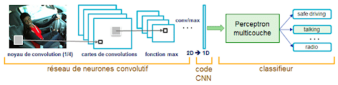



Fig. 1. Composición

Donde se tiene la extracción de las características de la imagen y se va reduciendo hasta llegar al clasificador, donde se tienen las características más valiosas de la imagen y se pasan al clasificador, a partir de aquí lo que se tiene es información del modelo.

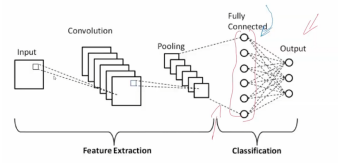



Fig. 2. Composición redes convolucionales

La parte de embeddings se tiene en la etapa de fully connected

### B. *Stacks)*

$$\text{INPUT} \rightarrow [(\text{Conv} \rightarrow \text{ReLU}) * n \rightarrow \text{Pool?}] * m \rightarrow (\text{FC} \rightarrow \text{ReLU}) * k \rightarrow \text{FC}$$

$$\text{donde} \quad 3 \geq n \geq 0, \quad m \geq 0, \quad k \geq 0$$

Decisiones de diseño que al final somos nosotros los que debemos probar dependiendo de las exigencias que tenga el dataset vamos a aumentar o disminuir los stacks, para ver cuál es más eficiente, optimiza los recursos y realiza mejor la tarea.

$$\text{INPUT} \rightarrow [(\text{Conv} \rightarrow \text{ReLU}) * n \rightarrow \text{Pool?}] * m$$

Extraemos información de la imagen y se aplica n veces antes de aplicar un pooling. El pooling puede no ser tan necesario, pero va a depender de las combinaciones de stride, padding, tamaño del kernel, etc, pero la mayoría de veces si va ser requerido.

El concepto de convoluciones y pooling lo vamos a aplicar hasta que tengamos un vector suficientemente pequeño para decir que a partir de ese resultado se va a hacer un clasificador, el cual tiene la siguiente estructura:

$$(\text{FC} \rightarrow \text{ReLU}) * k \rightarrow \text{FC}$$

Aquí se decide cuántas capas ocultas va a tener el clasificador y la capa de salida que nos indica la cantidad de clases que se tienen.

### C. *¿Qué arquitectura preferimos?*

Nota: Algo importante es escoger stacks con convoluciones pequeñas

Si por ejemplo tenemos un stack de 3 convoluciones de 3X3:

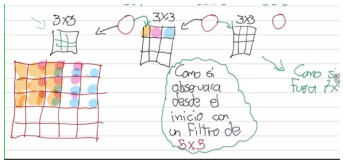



Fig. 3. convolución3x3

- Cada neurona en la primera para la primera convolución va a tener una vista de la imagen de entrada de 3X3
- En la segunda convolución cada neurona tiene un campo receptivo de 3X3, y como esta está conectada con la neurona anterior puede observar un total de 5x5 de la imagen.
- Para la tercera convolución cada neurona con un campo receptivo de 2x3 al estar conectada con la anterior puede observar un total de 7x7 de la imagen

Lo anterior es mucho mejor que tener un campo receptivo inicial de 7x7, esto presenta la desventaja que las neuronas van a ser computadas con una función lineal que está directamente con la entrada, entonces no estoy aprovechando los conceptos de no linealidad que me da ReLU.

### D. *Cantidad de parámetros*

Si continuamos con el ejemplo anterior de 7x7 tenemos que todos los volúmenes tienen C canales, es decir que la convolución de 7x7 tendría C x (7 x 7 x C) = 49C2, mientras que para el stack de 3x3 se tiene que 3 x (C x (3 x 3 x C)) = 27C2 parámetros.

C x (7 x 7 x C) = El 7x7 hace referencia al filtro, así como la C está dentro del paréntesis que hace referencia a los canales de la entrada, mientras que la C que está fuera hace referencia a los canales de salida.

### E. *Algunas reglas*

- El tamaño de la capa de entrada(es la que contiene la imagen) debería ser divisible por 2.
- Las capas convolucionales debería usar campos receptivos peqieños 3x3 o a lo mucho 5x5, utilizando un stride = .
- Padding de ceros de manera que no se altere el espacio, con el padding lo que buscamos es que después de todas las convoluciones todas las imágenes nos queden igual
- La configuración más común para la capa del pooling es max-pooling de Filtro=2 y Stride=2, aunque pueden haber otras configuraciones, pero se requiere más cuidados con las dimensiones, y es raro ver que sean más grandes de 3.

## IV. Arquitecturas

### A. *Arquitectura LeNet*

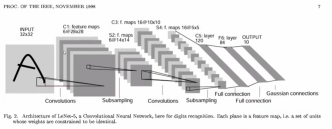



Fig. 4. Arquitectura LeNet

En la década de 1990, Yann LeCun, Leon Bottou, Yosuha Bengio y Patrick Haffner propusieron el diseño de red neuronal LeNet-5 para el reconocimiento de caracteres, tanto en escritura manual como en impresión mecánica. Dado que el diseño es claro y fácil de comprender, se utiliza con frecuencia como primer paso en la enseñanza de redes neuronales convolucionales.

Aunque ya tienen algo de antigüedad aun se siguen utilizando en combinación con conceptos como Transformers para aplicarle mayor computación y tener mejores resultados

### B. *Arquitectura AlexNet*

Arquitectura que resolvía problema de clasificación de mil imágenes

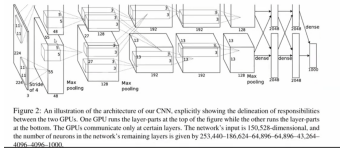


Fig. 5. Arquitectura AlexNet

### C. *Arquitectura ZFNet*

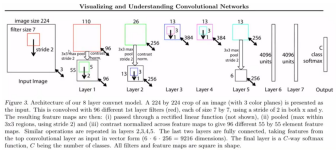



Fig. 6. Enter Caption

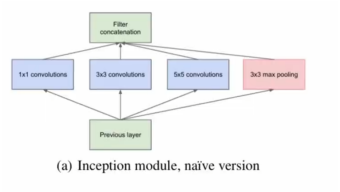
Lo que hace es que toma alexnet, hacerle mejorar publicar el artículo y publicar sus resultados.

**Diferencias entre ZFNet y AlexNet**

La principal diferencia es que AlexNet usaba filtros 11x11, 5x5, 3x3, mientras que ZFNet en la primera capa utilizó 7x7 y el resto de 3x3.

### D. *Arquitectura GoogleNet*

El modelo de AlexNet tenía 60 millones de parámetros, y con esta arquitectura de GoogleNet lo que hace es que se reduce a 4 millones de parámetros, y sus resultados siguen siendo muy buenos.




(a) Inception module, naïve version

Fig. 7. Arquitectura GoogleNet

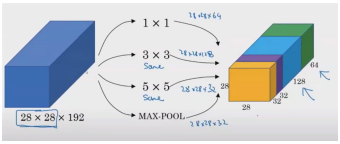



Fig. 8. Arquitectura GoogleNet

Un inception lo que hace es que procesa la capa anterior, pero con diferentes convoluciones, esto para producir la imagen de salida de esta capa

Una convolución de 1x1 lo que hace es una convolución a nivel de volumen por cada pixel, y es bastante útil para reducir las dimensiones.

Convolución 1x1 = convolución profunda

### *E. Arquitectura VGG16*


Fig. 9. Arquitectura VGG16

Su nombre hace referencia a la cantidad de capas que se están procesando, y lo que hace es que aplica Deep Learning y obtener mejores resultados.

### *F. Resnet*

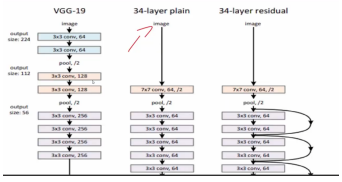



Fig. 10. Resnet

Conexiones residuales: Toma la información de las capas anteriores y la concatena con el resultados de capas fututas Esto lo hace para solucionar el problema de la capa del medio que cuando se aplica el propagation pierde mucho el gradiente.

### *G. DenseNet*

Lo que hace es optimizar la producción de volumen, es decir las cantidad de filtros que se aplican en cada convolución, además el denseNet no se aplica a toda la red si no que se aplica a pequeños bloques.

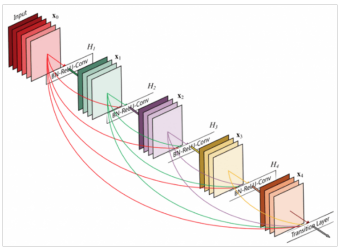


Fig. 11. Resnet

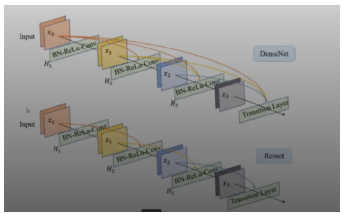



Fig. 12. DenseNet

## V. Visualizar qué aprenden las redes convolucionales

- Uno de los principales problemas es que es bastante complicado explicarla, para esto se usan diferentes técnicas que usan para ver lo que está observado y cuales detalles se enfocan para generar los resultados.
- Features aprendidos no son interpretables
- Se pueden visualizar las activaciones, es ver que después de computar los filtros de una capa de convolución, se toma esa información, se interpreta y la vamos.

### *A. Visualizar los filtros*

Es muy similar, pero en esta se observan los filtros y los pesos de estos para así tratar de ver qué es lo que observa la red.

• Se interprentan con más facilidad en la primera capa convolucional, ya que es está conectada directamente con los datos, además es posible poder visualizar los filtros más inter de la red

• Se muestran los filtro sin patrones de ruido, ya que si la visualización presenta esos patrones quiere decir que no fue entrenada correctamente

### *B. Embeddings*

En los embedding gradualmente transformamos las imágenes a un vector de interpretación y sus clases son separables por un clasificador, a menor dimesiaonalidad tenga menos características tiene el modelo y más probable es equivocarse. Además, los embeddings ayudan a saber qué tan bien o no está creado el modelo y nos permite reducir la dimensionalidad.

• Se puede transformar a una representación de dismesión menos manteniendo distancias iguales que la representación mayor. • Un buen modelo va a generar buenos resultados de embeddings

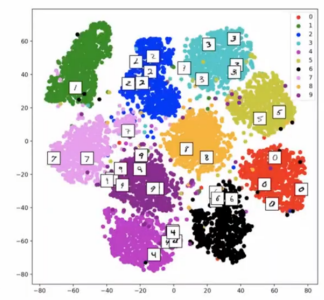


Fig. 13. Embeddings

**T-SNE:** Es una técnica que explica la reducción de dimensionalidad no lineal no supervisada para la exploración y visualización de datos de alta dimensión.

### C. *Ocultar partes de la imagen*

Se puede ocultar la información de la imagen, esto lo que hace es crear un cuadrito dentro de la imagen para ver en qué se está enfocando la red

### D. *Mapas de activación*

Lo que hace es que se enfocaliza en aquellos vectores donde la red está observando, para esto lo representa mediante un mapa de calor aquellas neuronas que excitan más la activación.

## VI. AutoEncoder

A partir del vector de características lo que hace es que reconstruye la imagen, esta nueva imagen es supervisada con la imagen original de entrada, es decir que los parámetros van a ser similares porque lo que se busca es reconstruir toda la data que viene de entrada.

### A. *Unsupervised Learning*

Al final el modelo se clasifica como un aprendizaje no supervisado, a pesar de que si lo hace ya que se le corrige para buscar el mejor resultado de salida tomando como base la entrada inicial.

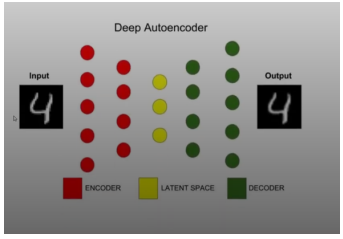


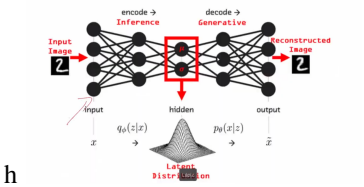

Fig. 14. AutoEncoder

El autoencoder está compuesto por 3 partes:

**1. Encoder:** Extrae las características de la imagen

**2. Latent Space(Espacio latente):** El vector de características es distribuido

**3. Decoder:** A partir de los embeddings se va a reconstruir la imagen, no quedan igual porque estoy recontruyéndala a partir de un vector muy pequeño.




h

Fig. 15. AutoEncoder

## VII. ¿Qué puede hacer un autoencoder?

### A. *Reducción de la dimensionalidad*

- Permite representar la entrada en vector más compacto
- Representación más poderosa en comparación con el PCA
- Permite reconstruir la información con menos perdida

### B. *Detección de anomalías*

Se puede usar el mismo autoencoder para detectar transacciones bancarias fraudulentas, para esto:

- Se entrena para la reconstrucción de datos de una tarea tomando en cuenta si estado postivo(es decir toda aquellas transacciones o tareas correctas)
- Aprende la representación latente de casos positivos
- Si una transferencia es suficimiente distinta a la normal, el autoencoder genera u vector latante deficiente
- Su loss function será muy alto al momento de la reconstrucción

### C. *Procesamiento de imágenes*

- Compresión de imágenes
- Reducción de ruido
- Super resolution