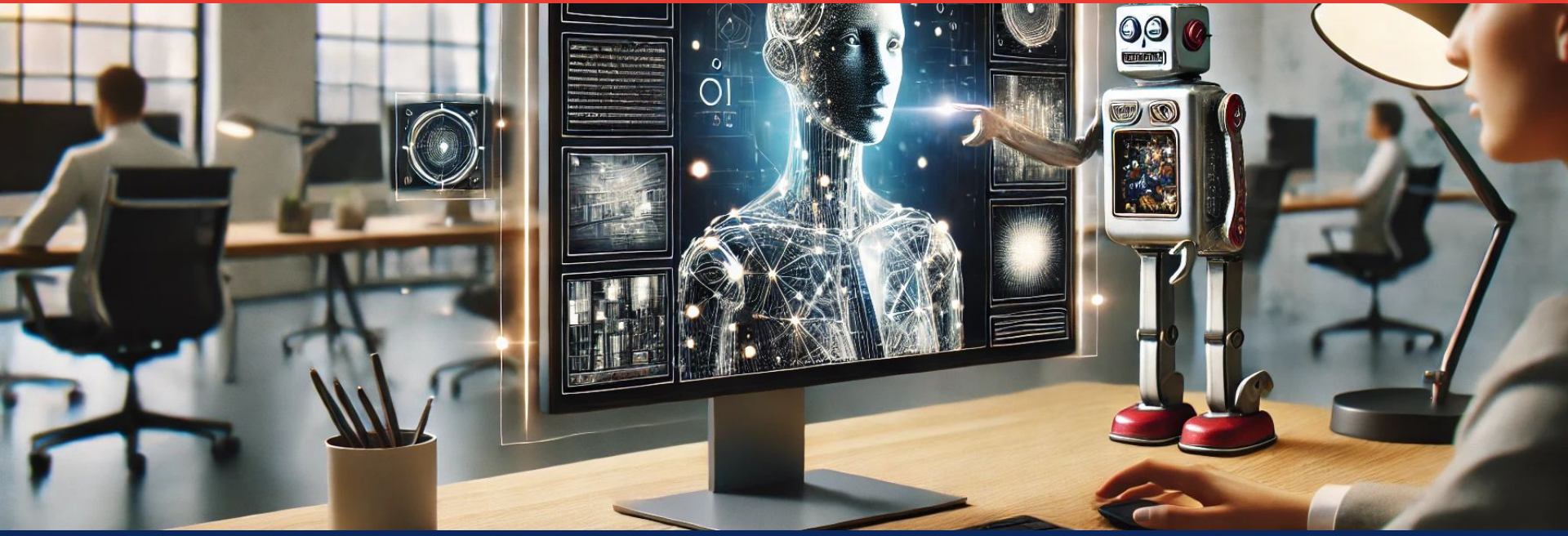


# Curso de IA Generativa Aplicada a Negocio

M3 - Clase 10: Técnicas de entrenamiento - 15/oct/24 - Prof: Fran Bartolomé

Cofinanciado por la Unión Europea

GOBIERNO DE ESPAÑA MINISTERIO DE INDUSTRIA Y TURISMO

EOI Escuela de organización industrial

Fondos Europeos

SISTEMA NACIONAL DE GARANTÍA JUVENIL SERVICIO PÚBLICO DE EMPLEO ESTATAL | SEPE

# Módulo 3

## Generación y edición de imagen

1. Introducción a la generación de imágenes.
2. Modelos Generativos.
3. Técnicas de entrenamiento.
4. Implicaciones en el Arte.
5. Herramientas:
    - Dall·E 3.
    - Stable Diffusion.
    - Midjourney.
    - Adobe Firefly.
    - Ideogram.
    - Leonardo.
    - Flux.
    - Krea.
6. Proyecto práctico.

# Técnicas de entrenamiento

# Objetivos del entrenamiento de modelos generativos

El objetivo principal del entrenamiento de modelos generativos es **capturar la distribución de probabilidad subyacente de los datos**. Esto significa que el modelo debe aprender a generar nuevos datos que sean indistinguibles de los datos reales. Otros objetivos específicos pueden incluir:

- **Generar datos de alta calidad**: Los datos generados deben ser realistas y coherentes.
- **Controlar la generación**: Ser capaz de influir en el proceso de generación para obtener resultados específicos.
- **Escalabilidad**: Entrenar modelos en grandes conjuntos de datos para mejorar la calidad de los resultados.

# Desafíos del entrenamiento de modelos generativos

**Minimización de la función de pérdida:**

- Diseño de la función de pérdida.
- Equilibrio entre el generador y el discriminador.
- Modos de colapso.

**Optimización de parámetros:**

- Gran número de parámetros.
- Múltiples escalas de tiempo.
- Mínimos locales.

La Función de pérdida mide la **diferencia entre los datos generados y los datos reales** *(El objetivo del entrenamiento es encontrar los parámetros del modelo que minimicen esta función)*.

La optimización de parámetros es el proceso de encontrar los **valores óptimos de los parámetros del modelo** *(Algoritmos como el descenso del gradiente se utilizan comúnmente para este propósito)*.

# Desafíos del entrenamiento de modelos generativos

**Uso de grandes conjuntos de datos:**

- Disponibilidad de datos.
- Almacenamiento y procesamiento.

**Evaluación de los resultados:**

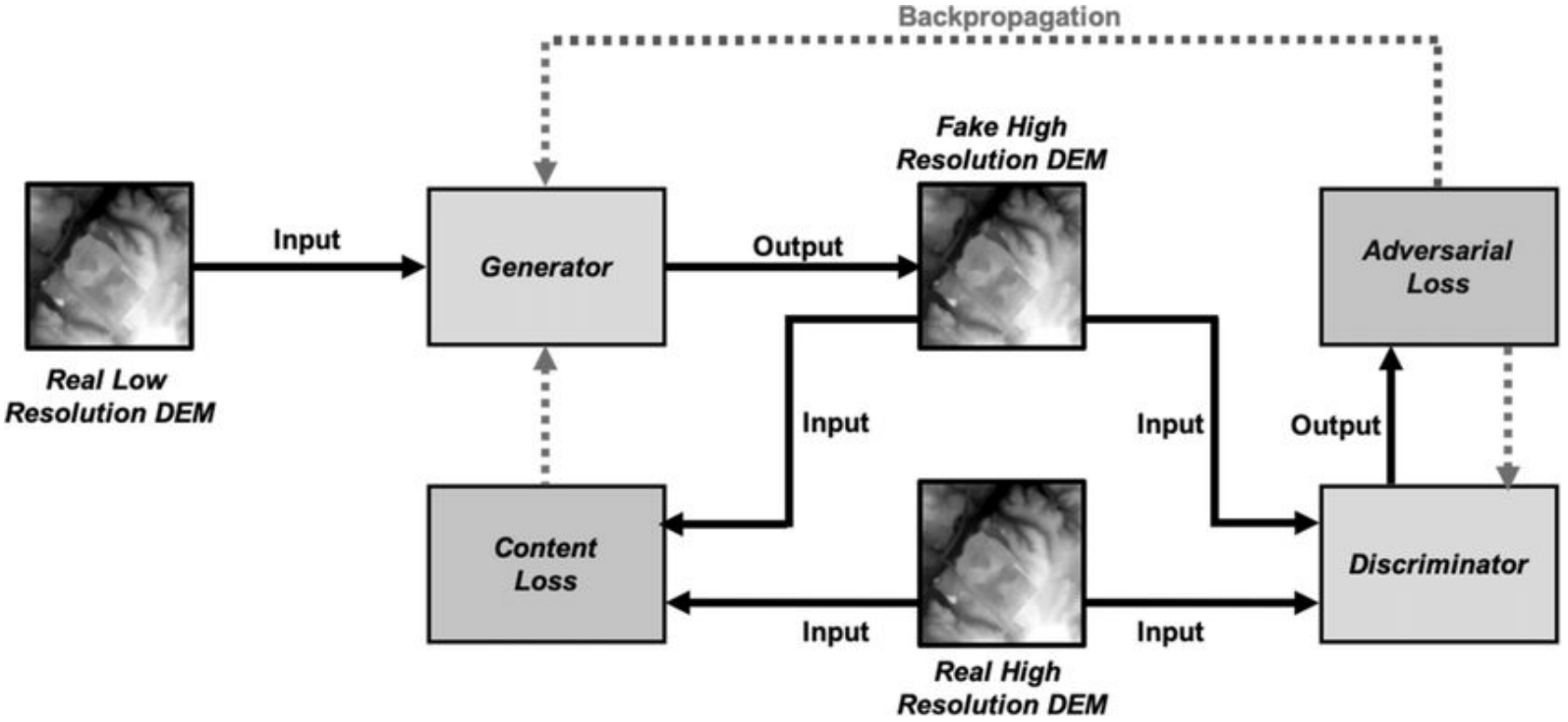
- Métricas adecuadas.
- Subjetividad.

Cuanto más **grande y diverso** sea el conjunto de datos de entrenamiento, más **generalizable** será el modelo. *(Los grandes conjuntos de datos permiten al modelo capturar una mayor variedad de características y patrones).*

# Otros desafíos

- El problema de la **explosión del gradiente**: En redes muy profundas, los gradientes pueden crecer exponencialmente durante la retropropagación, dificultando el entrenamiento.
- El problema del **desvanecimiento del gradiente**: Lo opuesto al problema anterior, los gradientes pueden volverse muy pequeños, lo que hace que el entrenamiento sea lento o incluso se estanque.
- El **modo de colapso en GANs**: Como se mencionó anteriormente, el modo de colapso ocurre cuando el generador colapsa en un pequeño conjunto de muestras, lo que limita la diversidad de los datos generados.

# Proceso de entrenamiento competitivo en las GANs



# Proceso de entrenamiento competitivo en las GANs

1. **Inicialización**: Tanto el generador como el discriminador se inicializan aleatoriamente.
2. **Generación de datos**: El generador crea un conjunto de datos sintéticos a partir de ruido aleatorio.
3. **Clasificación**: El discriminador recibe como entrada tanto los datos reales como los generados. Su objetivo es clasificar correctamente cada muestra como real o falsa.
4. **Cálculo de la pérdida**: Se calcula la pérdida tanto para el generador como para el discriminador. La pérdida del generador se basa en engañar al discriminador, mientras que la pérdida del discriminador se basa en distinguir correctamente los datos reales de los generados.
5. **Actualización de pesos**: Los pesos de ambas redes se actualizan utilizando un algoritmo de optimización, como el descenso del gradiente estocástico, para minimizar sus respectivas pérdidas.

# Función de pérdida en GANs

**Entropía Cruzada en GANs:**

La entropía cruzada es una función de pérdida comúnmente utilizada en problemas de clasificación binaria, y es la elección estándar para entrenar GANs. En este contexto, la entropía cruzada mide la diferencia entre la distribución de probabilidad de las etiquetas verdaderas y la distribución de probabilidad predichas por el modelo.

- **Discriminador**: El objetivo del discriminador es distinguir entre datos reales y generados. Por lo tanto, se busca **maximizar la probabilidad de asignar la etiqueta correcta a cada muestra**. La entropía cruzada se utiliza para penalizar al discriminador cuando clasifica erróneamente una muestra.
- **Generador**: El objetivo del generador es engañar al discriminador, haciendo que este clasifique los datos generados como reales. Por lo tanto, el generador busca **minimizar la probabilidad de que el discriminador clasifique correctamente los datos generados como falsos**. En esencia, el generador trata de maximizar la entropía cruzada del discriminador cuando este evalúa los datos generados.

# Problemas Comunes y la Función de Pérdida

- **Mode Collapse**: Este fenómeno ocurre cuando el generador colapsa en un modo único, generando siempre la misma muestra o un conjunto muy limitado de muestras. Esto puede ocurrir debido a un desequilibrio en la competencia entre el generador y el discriminador, donde **el discriminador se vuelve demasiado bueno y el generador no puede mejorar**.
- **Inestabilidad en el entrenamiento**: Las GANs pueden ser difíciles de entrenar debido a la naturaleza adversaria del proceso. Pequeños cambios en los hiperparámetros o en la arquitectura de la red pueden llevar a un **entrenamiento inestable o divergente**.

# Técnicas de Mejora del Entrenamiento de GANs

- **Wasserstein GAN (WGAN)**: propone una mejora significativa en la función de pérdida utilizada en las GANs tradicionales. En lugar de utilizar la entropía cruzada, la WGAN emplea una distancia de Wasserstein entre las distribuciones de los datos reales y los generados.

**VENTAJAS:**

**Mayor estabilidad en el entrenamiento**: La distancia de Wasserstein proporciona una estimación más suave y continua del error, lo que reduce la oscilación en el entrenamiento.

**Evita el problema del modo de colapso**: Al penalizar las grandes diferencias entre las distribuciones, la WGAN fomenta una exploración más uniforme del espacio de muestras.

**DESVENTAJAS:**

**Requiere un discriminador 1-Lipschitz**: El discriminador debe cumplir con una condición de Lipschitz para garantizar la convergencia. Esto se logra comúnmente mediante la técnica de corte de gradientes.

# Técnicas de Mejora del Entrenamiento de GANs

- **GANs Condicionales (cGANs)**: Las cGANs permiten controlar la salida del generador al proporcionar información adicional como una etiqueta de clase o un atributo específico. Esto es útil para generar imágenes con características particulares, como caras de personas de una determinada edad o estilo.

**FUNCIONAMIENTO**

Se añade un vector condicional tanto al generador como al discriminador. Este vector contiene información sobre la condición que se desea imponer a la salida.

**APLICACIONES**

Generación de imágenes condicionadas a un texto descriptivo.

Super-resolución de imágenes con información de la clase.

Colorización de imágenes en blanco y negro.

# Técnicas de Mejora del Entrenamiento de GANs

- **Gradient Penalty y Regularización**: La penalización de gradiente es una técnica utilizada para regularizar el discriminador y mejorar la estabilidad del entrenamiento. Consiste en agregar una penalización a la función de pérdida del discriminador basada en la norma de los gradientes del discriminador con respecto a sus entradas.

**OBJETIVO**

Promover una mejor estimación de la distancia entre las distribuciones.

Evitar que el discriminador se vuelva demasiado poderoso y colapse el generador.

**OTRAS TÉCNICAS DE REGULARIZACIÓN**

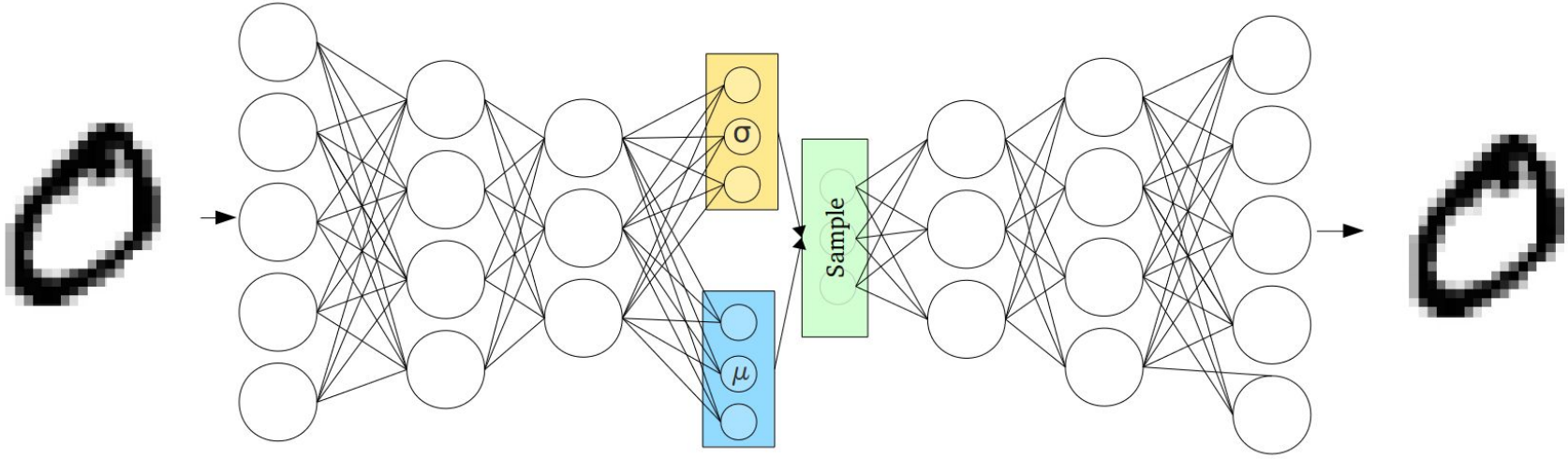
**Dropout**: Ayuda a prevenir el sobreajuste al aleatoriamente desactivar neuronas durante el entrenamiento.

**Normalización por lotes**: Ayuda a estabilizar el entrenamiento al normalizar las entradas de cada capa.

# Técnicas de Mejora del Entrenamiento de GANs

- **Spectral Normalization**: Otra técnica para regularizar el discriminador y garantizar la condición de Lipschitz.
- **Progressive Growing**: Una técnica que permite entrenar GANs de alta resolución de forma más eficiente.
- **Minibatch Discrimination**: Una técnica que permite al discriminador comparar múltiples muestras a la vez, mejorando la capacidad del generador para generar muestras diversas.

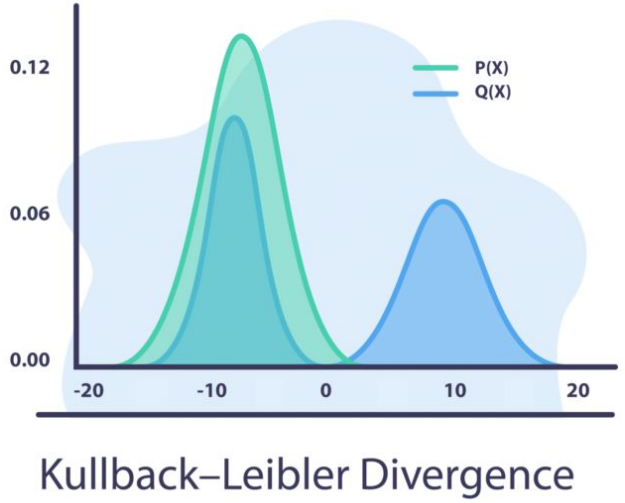
# Técnicas de Entrenamiento para Autoencoders Variacionales



# Divergencia de Kullback-Leibler (KL): Regularizar la Latente

**¿Por qué regularizar la latente?**

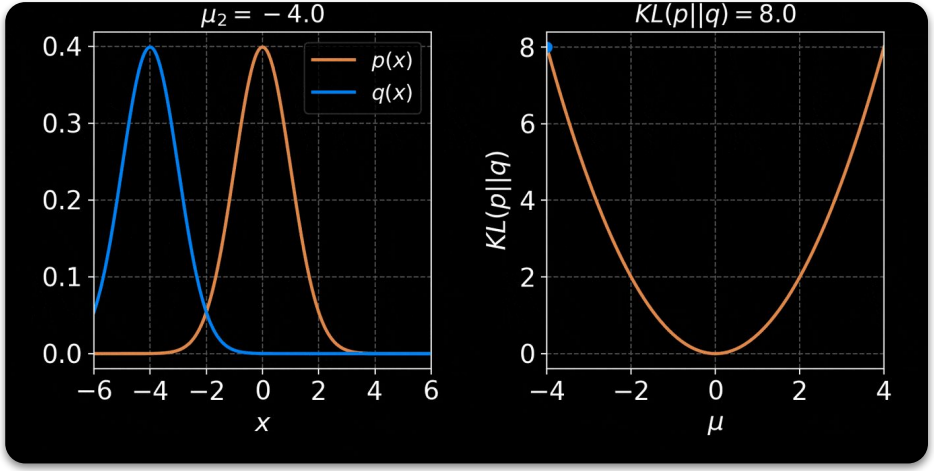
- **Interpretabilidad:** Al forzar a la distribución latente a ser cercana a una distribución conocida (como una gaussiana), se facilita la interpretación de las representaciones latentes.
- **Generalización**: La regularización ayuda a prevenir el sobreajuste y mejora la capacidad del modelo para generalizar a nuevos datos.



# Divergencia de Kullback-Leibler (KL): Regularizar la Latente

**¿Cómo funciona?**

- El VAE consta de un codificador que mapea los datos de entrada a un espacio latente y un decodificador que reconstruye los datos a partir de este espacio.
- La divergencia KL se calcula entre la distribución posterior aproximada (inferida por el codificador) y una distribución previa (generalmente una gaussiana estándar).
- Esta divergencia KL se incluye en la función de pérdida total, penalizando las representaciones latentes que se desvían demasiado de la distribución previa.



# Reconstrucción de los datos

**¿Por qué es importante la reconstrucción?**

- El objetivo principal de un VAE es aprender una representación latente útil que permita reconstruir los datos de entrada.
- Una buena reconstrucción indica que el modelo ha capturado las características más importantes de los datos.

**¿Cómo se mide?**

- Se utiliza una función de pérdida de reconstrucción, como el error cuadrático medio (MSE) para datos continuos o la entropía cruzada para datos categóricos.
- Esta pérdida mide la diferencia entre los datos originales y los datos reconstruidos por el decodificador.

# Función de Pérdida Total

Pérdida Total = Pérdida de Reconstrucción + β * Divergencia KL

**Resumen:**

- **Divergencia KL**: Regulariza la distribución latente, fomentando representaciones más interpretables y generalizables.
- **Pérdida de reconstrucción**: Mide la calidad de la reconstrucción de los datos originales.
- **Pérdida total**: Combina ambas pérdidas para encontrar un equilibrio óptimo entre la regularización y la reconstrucción.

# ¿Por qué la reparametrización para el entrenamiento?

En lugar de muestrear directamente de la distribución posterior aproximada, se introduce una variable aleatoria auxiliar que sigue una distribución simple (por ejemplo, una distribución normal estándar). La muestra de la distribución posterior aproximada se expresa entonces como una función determinista de esta variable aleatoria auxiliar y de los parámetros del modelo.

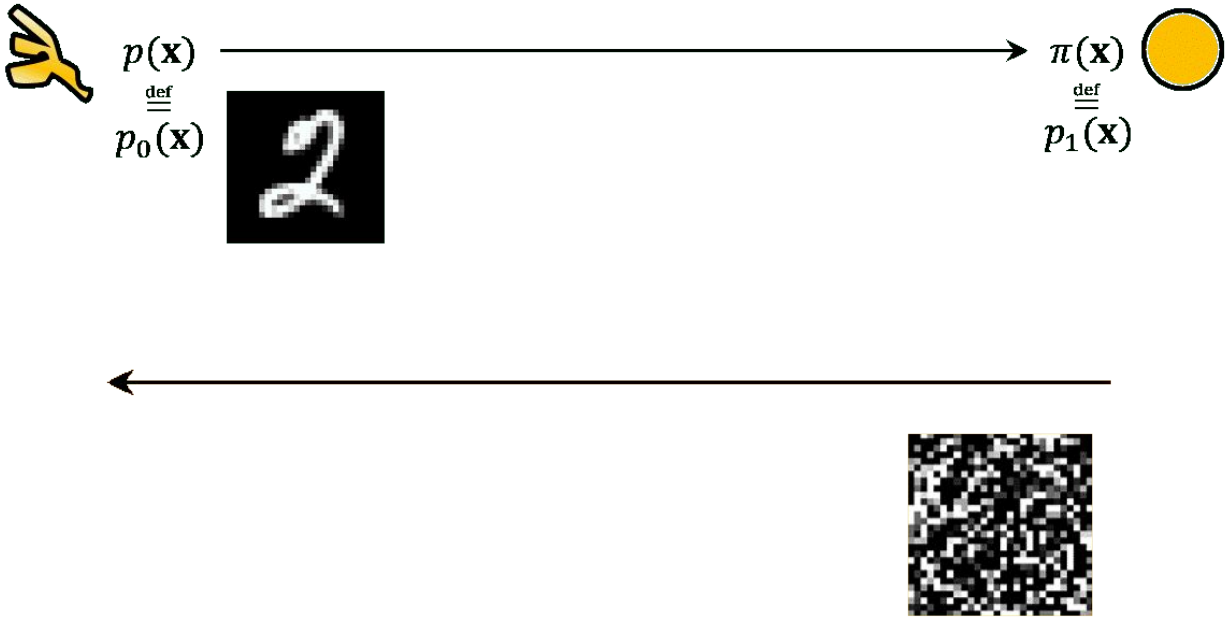
**Ventajas:**

- **Flujo de gradientes**: Permite el cálculo del gradiente a través de la muestra, lo que es esencial para el entrenamiento de modelos variacionales.
- **Flexibilidad**: Se puede aplicar a una amplia variedad de distribuciones posteriores aproximadas.
- **Eficiencia**: Al separar la aleatoriedad del modelo, se pueden utilizar técnicas de optimización más eficientes.

# Extensiones y mejoras de los VAEs

- **Beta-VAEs: Ajuste fino de la regularización**: Los Beta-VAEs son una extensión de los VAEs que introducen un hiperparámetro adicional, β, para controlar el grado de regularización de la distribución latente. Este hiperparámetro modula la importancia relativa de la pérdida de reconstrucción y la divergencia KL.
- **Hierarchical VAEs: Capas latentes múltiples para mayor complejidad**. Los Hierarchical VAEs (HVAEs) introducen múltiples capas latentes en el modelo. Esto permite capturar jerarquías en los datos y modelar distribuciones más complejas.
- **AEs adversariales**: Combinan VAEs con GANs para mejorar la calidad de las muestras generadas.
- **VAEs condicionales**: Permiten controlar la generación de muestras mediante la adición de información condicional.
- **VAEs con regularización de sparsity**: Promueven representaciones latentes dispersas, lo que puede mejorar la interpretabilidad y la eficiencia.

# Entrenamiento de Modelos de Difusión



# Proceso de Difusión Directa

El proceso de difusión directa consiste en añadir ruido a esta imagen de forma gradual hasta que se convierte en ruido puro, es decir, una imagen completamente aleatoria.

**¿Cómo se añade el ruido?**

- **Ruido gaussiano**: Generalmente se utiliza ruido gaussiano, que es un tipo de ruido aleatorio con una distribución en forma de campana.
- **Pasos de tiempo**: El proceso se divide en varios pasos de tiempo. En cada paso, se añade una pequeña cantidad de ruido a la imagen.
- **Ecuación de difusión**: Existe una ecuación matemática que describe cómo evoluciona la imagen a lo largo de los pasos de tiempo, añadiendo cada vez más ruido.

Al añadir ruido de forma gradual, el modelo aprende a predecir la cantidad de ruido que se ha añadido en cada paso. Esta información será crucial para el proceso inverso.

# Proceso de Difusión Inversa

Una vez que el modelo ha aprendido a predecir el ruido añadido, se inicia el proceso de difusión inversa. Este proceso consiste en eliminar el ruido de forma gradual, reconstruyendo la imagen original a partir del ruido puro.

**¿Cómo se elimina el ruido?**

- **Pasos de tiempo inversos**: Se siguen los mismos pasos de tiempo que en el proceso directo, pero en sentido inverso.
- **Predicción del ruido**: En cada paso, el modelo predice la cantidad de ruido que se añadió en ese paso durante el proceso directo.
- **Eliminación del ruido**: A partir de la predicción del ruido, se elimina una parte de este ruido de la imagen, acercándola a la imagen original.
- **Iteración**: Este proceso se repite hasta que se obtiene una imagen clara y nítida.

El modelo ha aprendido durante el proceso directo a predecir el ruido añadido en cada paso. Al utilizar esta información, puede invertir el proceso y reconstruir la imagen original.

# Entrenamiento del Modelo de Difusión

**Función de pérdida:**

Se utiliza una función de pérdida para medir la diferencia entre la predicción del modelo y el ruido real añadido. La función de pérdida guía el proceso de entrenamiento, permitiendo al modelo ajustar sus parámetros para minimizar esta diferencia.

**Optimización:**

Se utilizan técnicas de optimización como el descenso del gradiente estocástico para minimizar la función de pérdida y ajustar los parámetros del modelo.

# Función de Pérdida en modelos de difusión

**¿Qué se predice en un modelo de difusión?**

En cada paso del proceso de difusión, el modelo intenta predecir la cantidad de ruido que se ha añadido a la imagen. Esto es fundamental, ya que esta predicción es la base para el proceso inverso de eliminación de ruido.

**¿Cómo se construye la función de pérdida?**

La función de pérdida más comúnmente utilizada en modelos de difusión es el error cuadrático medio (MSE). Este error calcula la diferencia al cuadrado entre la predicción del modelo y el ruido real añadido en cada paso.

$$MSE = (\text{Predicción del modelo} - \text{Ruido real})^2$$

# Función de Pérdida en modelos de difusión

**¿Por qué es importante el MSE?**

- **Intuitivo**: Es fácil de entender y calcular.
- **Diferenciable**: Permite el uso de técnicas de optimización basadas en gradiente, como el descenso del gradiente estocástico.
- **Efectivo**: Ha demostrado ser muy eficaz en la práctica para entrenar modelos de difusión.

**El papel de la predicción del ruido en la función de pérdida**

- **Capturar las características de la distribución del ruido**: El modelo aprende a entender cómo se distribuye el ruido en cada paso del proceso de difusión.
- **Modelar la relación entre la imagen y el ruido**: El modelo aprende a relacionar la imagen original con el ruido añadido en cada paso.

# Técnicas de Optimización para Modelos de Difusión

**Reducción de Pasos**

- **Discretización no uniforme**: En lugar de utilizar pasos de difusión de igual tamaño, se pueden utilizar pasos más grandes al principio del proceso y pasos más pequeños al final. Esto permite capturar las características generales de la imagen rápidamente y luego refinar los detalles.
- **Saltos de pasos**: En lugar de realizar todos los pasos de difusión, se pueden omitir algunos pasos intermedios. Esto reduce el número de cálculos necesarios sin sacrificar demasiado la calidad de la imagen.
- **Modelos condicionales**: Al condicionar el modelo en una imagen de referencia, se puede reducir el número de pasos necesarios para generar una imagen similar.
- **Aprendizaje de tasas de aprendizaje dinámicas**: Ajustar la tasa de aprendizaje durante el entrenamiento puede ayudar a acelerar la convergencia y reducir el número de pasos necesarios.

# Técnicas de Optimización para Modelos de Difusión

## Condicionamiento en Modelos de Difusión

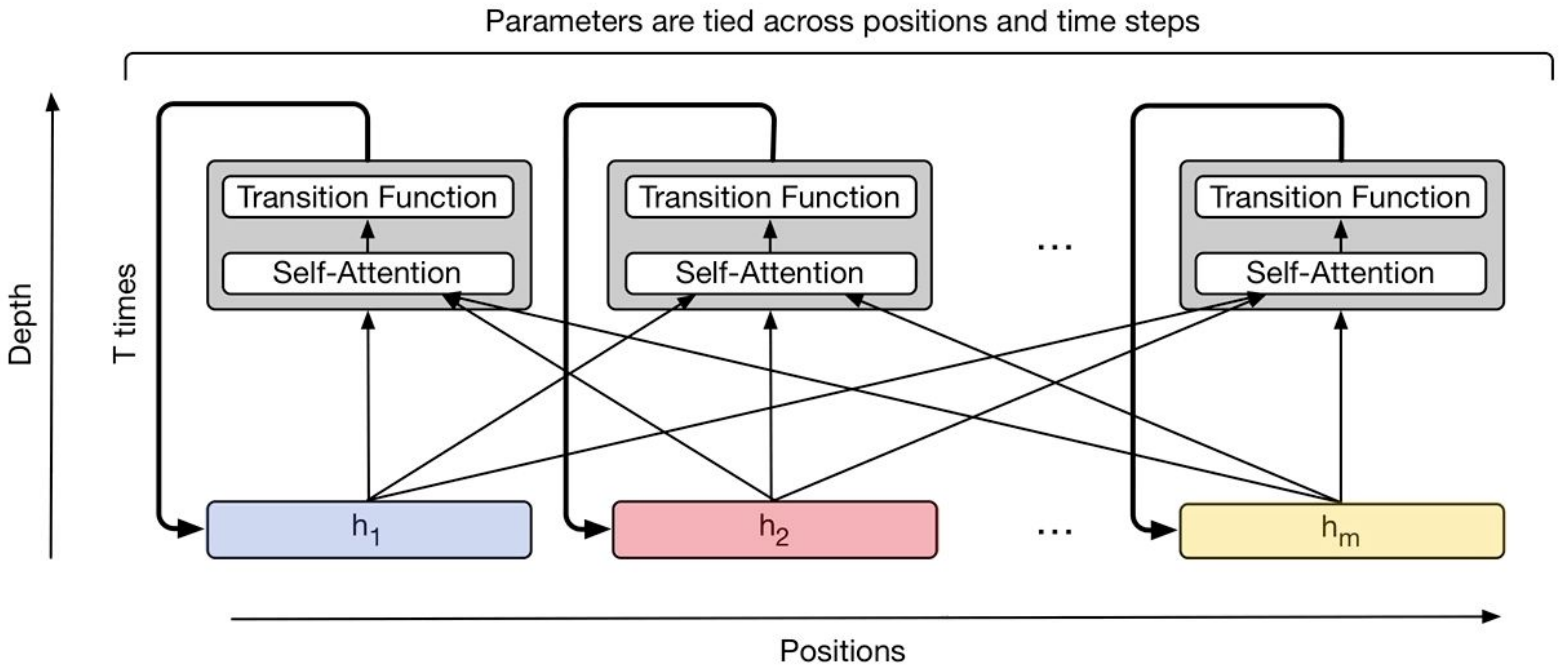
- **Condicionamiento textual:** Se puede utilizar texto para describir el contenido deseado de la imagen. Por ejemplo, se puede pedir al modelo que genere una imagen de un "gato sentado en un sofá".
- **Condicionamiento de imágenes**: Se puede utilizar una imagen de referencia para guiar el proceso de generación. Por ejemplo, se puede utilizar una imagen de un boceto para generar una imagen realista.
- **Condicionamiento de latentes**: Se puede utilizar un vector latente para controlar atributos específicos de la imagen, como el color, la textura o la pose.

# Técnicas de Optimización para Modelos de Difusión

## Condicionamiento en Modelos de Difusión

- **Condicionamiento textual:** Se puede utilizar texto para describir el contenido deseado de la imagen. Por ejemplo, se puede pedir al modelo que genere una imagen de un "gato sentado en un sofá".
- **Condicionamiento de imágenes**: Se puede utilizar una imagen de referencia para guiar el proceso de generación. Por ejemplo, se puede utilizar una imagen de un boceto para generar una imagen realista.
- **Condicionamiento de latentes**: Se puede utilizar un vector latente para controlar atributos específicos de la imagen, como el color, la textura o la pose.

# Entrenamiento para Transformers Generativos



# Entrenamiento Basado en Máscaras (*Masked Training*)

**¿Por qué usar máscaras?**

- **Pre-entrenamiento eficiente:** Al enmascarar partes de la entrada, se fuerza al modelo a aprender representaciones más profundas y contextuales del lenguaje. Esto permite que el modelo capture relaciones semánticas y sintácticas entre las palabras, incluso cuando algunas de ellas están ocultas.
- **Adaptación a diversas tareas**: Los modelos pre-entrenados con esta técnica pueden ser fácilmente adaptados a una amplia variedad de tareas de NLP, como clasificación de texto, traducción automática y generación de texto.
- **Mejora de la robustez**: Al entrenar el modelo a lidiar con datos faltantes o corruptos, se aumenta su robustez y capacidad de generalización.

# Entrenamiento Basado en Máscaras (*Masked Training*)

**¿Cómo funciona?**

1. **Enmascaramiento aleatorio:** Se selecciona aleatoriamente un porcentaje de tokens en la secuencia de entrada y se reemplazan por un token especial (por ejemplo, [MASK]).
2. **Predicción de los tokens enmascarados**: El modelo intenta predecir los tokens originales que fueron enmascarados, basándose en el contexto proporcionado por los tokens visibles.
3. **Cálculo de la pérdida**: Se utiliza una función de pérdida para comparar las predicciones del modelo con los tokens originales. El objetivo es minimizar esta pérdida durante el entrenamiento.

# Ventajas y desafíos

## VENTAJAS:

**Aprendizaje profundo de representaciones**: El modelo aprende a capturar relaciones semánticas y sintácticas a un nivel profundo.

**Flexibilidad**: Puede ser aplicado a una amplia variedad de tareas de NLP.

**Mejora del rendimiento**: Los modelos pre-entrenados con esta técnica suelen obtener mejores resultados en diversas tareas de NLP.

## DESAFÍOS:

**Elección del porcentaje de tokens a enmascarar:** Un porcentaje demasiado alto o demasiado bajo puede afectar negativamente el rendimiento del modelo.

**Diseño de la máscara**: La forma en que se seleccionan los tokens a enmascarar puede influir en el tipo de información que el modelo aprende a capturar.

**Costo computacional**: El entrenamiento de modelos grandes con esta técnica puede ser computacionalmente costoso.

# Pre entrenamiento y Ajuste Fino (*Fine-tuning*)

**Ventajas del pre entrenamiento:**

- **Representaciones sólidas:** El modelo aprende representaciones ricas y significativas del lenguaje, que pueden ser reutilizadas para diversas tareas.
- **Menor cantidad de datos**: Al pre entrenar el modelo en un conjunto de datos masivo, se reduce la necesidad de grandes cantidades de datos etiquetados para tareas específicas.
- **Convergencia más rápida**: El modelo ya ha aprendido una representación inicial del lenguaje, lo que acelera el proceso de entrenamiento en tareas específicas.

# Pre entrenamiento y Ajuste Fino (*Fine-tuning*)

**Pasos del ajuste fino:**

1. **Selección de una arquitectura:** Se elige una arquitectura Transformer preentrenada adecuada para la tarea (e.g., BERT, GPT).
2. **Preparación de los datos**: Se prepara un conjunto de datos etiquetado para la tarea específica.
3. **Entrenamiento**: Se entrena el modelo en el conjunto de datos etiquetado, congelando algunas capas del modelo preentrenado para preservar el conocimiento general y ajustando solo las capas superiores para la tarea específica.

# Ventajas y consideraciones

## VENTAJAS:

**Especialización:** El modelo se adapta a la tarea específica, mejorando su rendimiento.

**Eficiencia**: Se requiere menos datos y tiempo de entrenamiento en comparación con entrenar un modelo desde cero.

**Flexibilidad**: Los modelos preentrenados pueden ser ajustados finamente para una amplia variedad de tareas.

## CONSIDERACIONES:

**Tamaño del conjunto de datos:** La cantidad de datos etiquetados disponibles para el ajuste fino influye en el rendimiento del modelo.

**Tasa de aprendizaje**: La tasa de aprendizaje utilizada durante el ajuste fino debe ser cuidadosamente seleccionada para evitar el sobreajuste.

**Capas a ajustar**: Decidir qué capas del modelo ajustar puede afectar el rendimiento y la eficiencia del ajuste fino.

# Técnicas de Regularización en Transformers

**Dropout:**

Esta técnica consiste en desactivar aleatoriamente un porcentaje de neuronas durante el entrenamiento. Esto obliga al modelo a no depender demasiado de ninguna neurona en particular y a distribuir la información de manera más uniforme.

**Weight decay:**

También conocido como regularización L2, esta técnica añade un término a la función de pérdida que penaliza los grandes valores de los pesos. Esto tiene el efecto de encoger los pesos y hacer que el modelo sea más simple, reduciendo así el riesgo de sobreajuste.

# Técnicas de Regularización en Transformers

**Aprendizaje Adaptativo (Learning Rate Scheduling):**

- **Aprendizaje con tasa de aprendizaje decreciente**: Se inicia con una tasa de aprendizaje relativamente alta para explorar rápidamente el espacio de parámetros y luego se reduce gradualmente para permitir una convergencia más fina.
- **Adaptación basada en el rendimiento**: Se ajusta la tasa de aprendizaje en función del rendimiento del modelo en un conjunto de validación. Por ejemplo, si el rendimiento deja de mejorar, se puede reducir la tasa de aprendizaje.

# Técnicas de Regularización en Transformers

**Otras técnicas de regularización:**

- **Regularización L1:** Similar a la L2, pero penaliza el valor absoluto de los pesos. Esto puede conducir a la sparsidad en el modelo, eliminando características irrelevantes.
- **Early stopping**: Se detiene el entrenamiento cuando el rendimiento en un conjunto de validación comienza a empeorar, evitando que el modelo se sobreajuste.
- **Data augmentation**: Se aumentan los datos de entrenamiento mediante transformaciones aleatorias (por ejemplo, ruido, rotación de imágenes) para hacer que el modelo sea más robusto a variaciones en los datos.

# Optimización y Técnicas avanzadas

# Optimización y Técnicas avanzadas

**Adam (Adaptive Moment Estimation):**

Es uno de los optimizadores más utilizados en la actualidad. Combina lo mejor de otros optimizadores como RMSprop y Momentum. Adam adapta la tasa de aprendizaje para cada parámetro individualmente, lo que acelera significativamente la convergencia en comparación con otros métodos. Además, utiliza estimaciones de primer y segundo momento de los gradientes para controlar la tasa de aprendizaje.

**AdamW (Adam with Weight Decay):**

Es una variante de Adam que incorpora la regularización L2 (weight decay). Esta modificación ayuda a prevenir el sobreajuste y mejora la generalización del modelo.

**VENTAJAS:**

**Convergencia rápida**.

**Adaptación**.

**Robustez**.

**¿Cuándo usarlos?**

Excelentes opciones para una amplia variedad de problemas y suelen ser una buena elección por defecto.

# Optimización y Técnicas avanzadas

**RMSprop (Root Mean Square Propagation):**

Optimizador adaptativo que divide la tasa de aprendizaje para cada parámetro por una estimación de la raíz cuadrada de la media cuadrática de los gradientes anteriores. Esto permite que el optimizador adapte la tasa de aprendizaje a diferentes parámetros, acelerando la convergencia en direcciones con grandes gradientes y ralentizándola en direcciones con pequeños gradientes.

**¿Cuándo usarlo?**

Buena opción cuando se trabaja con datos ruidosos o cuando se espera que los gradientes varíen mucho.

**SGD (Stochastic Gradient Descent) con Momentum:**

Es el optimizador más básico, pero puede ser muy eficaz cuando se combina con *momentum*. Momentum agrega una fracción de la actualización anterior a la actualización actual, lo que ayuda a acelerar la convergencia y a evitar que el optimizador se quede atascado en mínimos locales.

**¿Cuándo usarlo?**

Buena opción para problemas simples o cuando se requiere un control más fino sobre el proceso de optimización.

# Estrategias de Regularización

**Early Stopping (Detención Temprana):**

- **Conjunto de validación**: Durante el entrenamiento, se reserva una parte de los datos (el conjunto de validación) que el modelo no ha visto durante el entrenamiento.
- **Monitoreo del rendimiento**: Se evalúa el rendimiento del modelo en el conjunto de validación después de cada época de entrenamiento.
- **Detención**: Si el rendimiento en el conjunto de validación deja de mejorar, se detiene el entrenamiento. Esto evita que el modelo siga aprendiendo patrones específicos de los datos de entrenamiento y lo ayuda a generalizar mejor.

**Data Augmentation (Aumento de Datos):**

Esta técnica consiste en crear nuevas muestras de entrenamiento a partir de las existentes, aplicando transformaciones aleatorias. Es como tomar una foto y luego modificarla ligeramente (rotándola, cambiando el brillo, etc.) para obtener varias fotos diferentes.

**Ventajas**

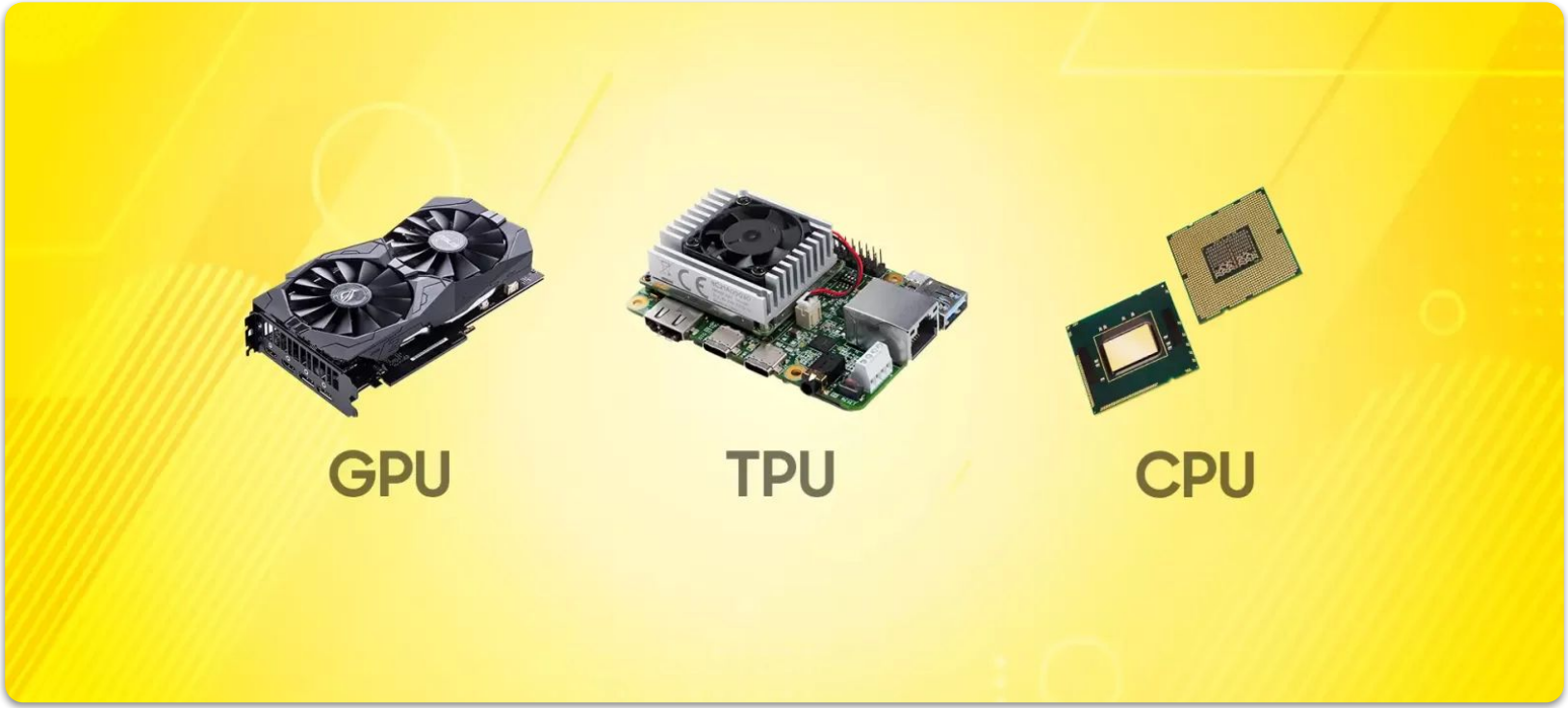
Mayor diversidad.

Prevención del sobreajuste.

# Beneficios de la Regularización

- **Mejor generalización:** Los modelos regularizados son capaces de hacer predicciones más precisas en nuevos datos.
- **Prevención del sobreajuste**: Reducen el riesgo de que el modelo se ajuste demasiado a los datos de entrenamiento.
- **Mayor robustez**: Los modelos regularizados son menos sensibles a pequeñas variaciones en los datos de entrada.

# Entrenamiento Distribuido y Uso de GPU/TPU



# Entrenamiento distribuido con GPU / TPU

**¿Cómo funciona?**

1. Paralelización de datos: Se divide el conjunto de datos en fragmentos más pequeños y se distribuye en múltiples dispositivos. Cada dispositivo procesa su fragmento de datos localmente.
2. Paralelización de modelos: El modelo se divide en partes más pequeñas y se distribuye en diferentes dispositivos. Cada dispositivo calcula el gradiente para su parte del modelo.
3. Comunicación entre dispositivos: Los dispositivos se comunican entre sí para sincronizar los pesos del modelo y actualizarlos de forma colectiva.

**VENTAJAS:**

Aceleración del entrenamiento

Modelos más grandes

Mayor eficiencia

**DESAFÍOS**

Comunicación

Sincronización

Escalabilidad

# Entrenamiento distribuido con GPU / TPU

**Estrategias**

- **Data parallelism**: Cada dispositivo tiene una copia completa del modelo y procesa un subconjunto diferente de los datos.
- **Model parallelism**: El modelo se divide en partes y cada dispositivo procesa una parte diferente del modelo.
- **Pipeline parallelism**: Las diferentes capas de la red se ejecutan en diferentes dispositivos, creando una línea de montaje.

**Frameworks y herramientas:**

**TensorFlow:** Ofrece una API flexible para el entrenamiento distribuido en múltiples GPUs y TPUs.

**PyTorch**: Proporciona herramientas para el entrenamiento distribuido tanto en CPU como en GPU.

**Horovod**: Una biblioteca de optimización de rendimiento para el entrenamiento distribuido en TensorFlow y PyTorch.

# Comparativa de Técnicas de Entrenamiento entre Modelos

| Modelo | F. de Pérdida | Objetivo principal | Características |
|---|---|---|---|
| GANs | Entropía cruzada / Wasserstein | Maximizar la diferencia entre el discriminador y el generador | - Compite entre dos redes<br>- Problemas comunes: mode collapse e inestabilidad en el entrenamiento. |
| | Gradient Penalty | Suavizar la función de pérdida para mejorar la estabilidad | - Utilizado en WGAN para evitar gradientes explosivos o desvanecidos. |
| VAEs | Divergencia de Kullback-Leibler (KL) + MSE | Minimizar la diferencia entre la distribución latente y una distribución normal | - La pérdida combina la reconstrucción de la entrada y la regularización de la latente.<br>- La divergencia KL regula la latente para mantener una distribución normal. |
| Modelos de Difusión | MSE (Error Cuadrático Medio) | Aprender a predecir el ruido añadido en cada paso de la difusión | - Minimiza la diferencia entre el ruido verdadero y el ruido predicho en cada paso. |
| Transformers Generativos | Entropía cruzada con enmascarado (Masked LM) | Predecir la siguiente palabra o token en secuencias de datos | - Utiliza enmascarado para predecir partes de la entrada durante el entrenamiento (masked training). |
| | Pérdida de ajuste fino (Fine-tuning Loss) | Ajustar el modelo a tareas específicas con datos adicionales | - Se adapta a tareas específicas utilizando conjuntos de datos más pequeños y especializados. |

Gracias por tu atención

# ¿Preguntas, dudas, inquietudes, …?

franbvgamazo franbvg@proton.me