Inteligencia Artificial II

Juan Pablo Restrepo Uribe

Ing. Biomedico - MSc. Automatización y Control Industrial

jprestrepo@correo.iue.edu.co

2023

Institución Universitaria de Envigado

Técnicas de mejoramiento de CNN

Existen variadas formas para mejorar la performance de las CNNS. Entre las que destacan

Normalización de datos

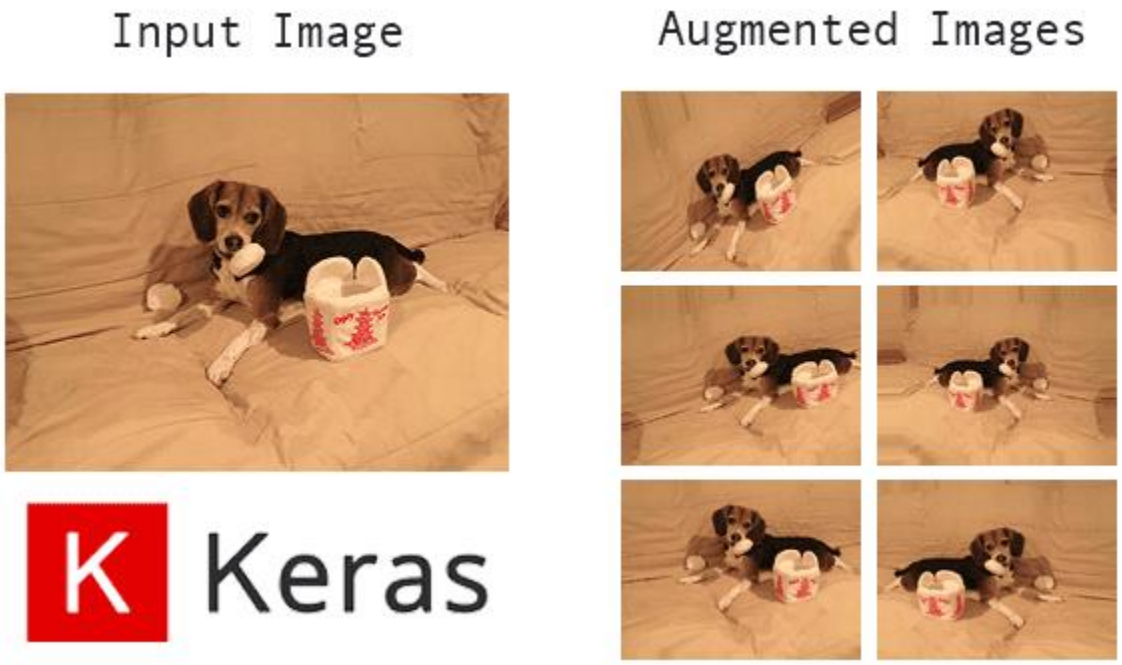
Asegúrate de normalizar tus datos de entrada para que tengan media cero y desviación estándar uno. Esto acelera la convergencia y puede mejorar el rendimiento.

$$x = \frac{x - x_{min}}{x_{max} - x_{min}}$$

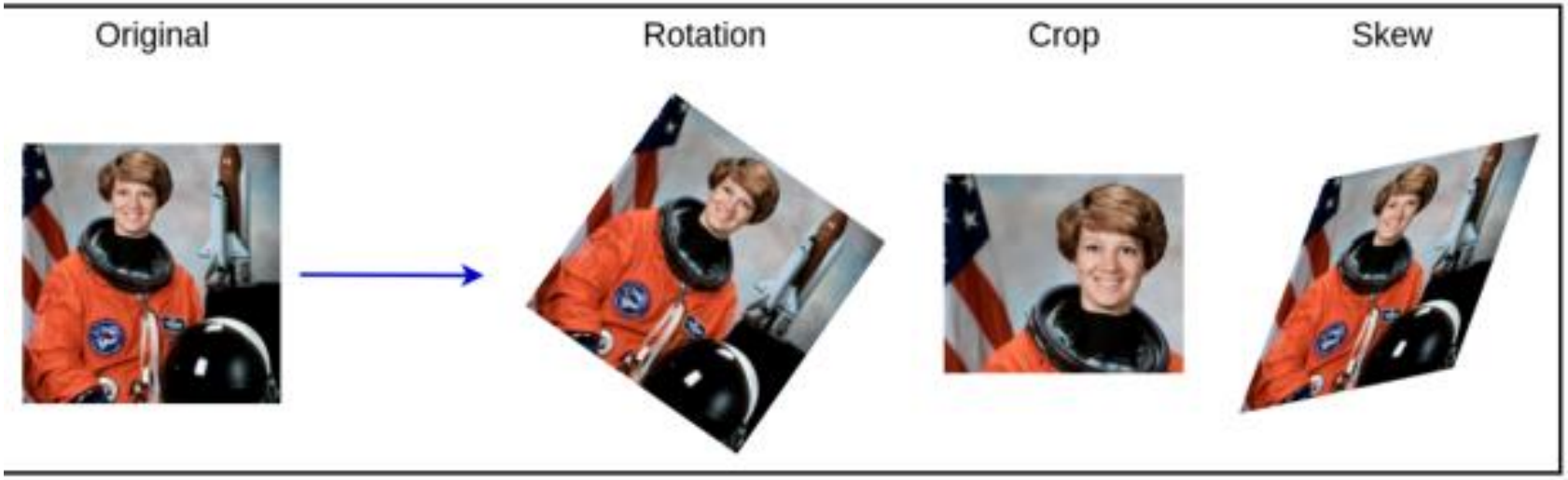
$$x = \frac{x - \mu}{\sigma}$$

Aumento de datos (Data Augmentation)

Genera nuevas muestras de entrenamiento al aplicar transformaciones como rotaciones, recortes, y cambios de brillo y contraste a las imágenes originales. Esto ayuda a la red a generalizar mejor.

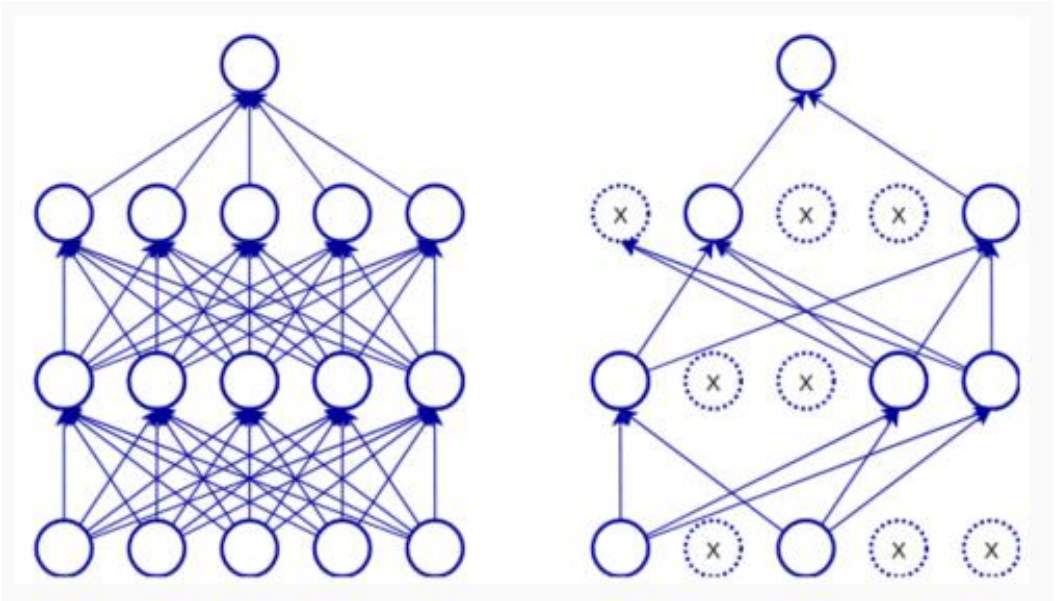


Aumento de datos (Data Augmentation)



Regularización

Regularización: Utiliza técnicas de regularización como dropout para reducir el sobreajuste (overfitting).



Otros métodos

- Arquitectura de red: diferentes arquitecturas como VGG, ResNet, Inception, etc.
- Funciones de activación: funciones de activación como ReLU, Leaky ReLU o unidades sigmoides
- Optimización: algoritmos de optimización como Adam, SGD, RMSprop, etc., y ajusta la tasa de aprendizaje.
- Tamaño de lote (Batch Size): tamaño de lote para optimizar la velocidad de entrenamiento y la convergencia.
- **Aprendizaje por transferencia (Transfer Learning):** modelos pre-entrenados en conjuntos de datos grandes, puedes utilizarlos como punto de partida y afinarlos para tu tarea específica.
- Capas de convolución y pooling: Experimenta con el número de capas y la configuración de pooling en tu red para encontrar la mejor combinación.

Generación de imágenes

Hasta ahora hemos utilizado RRNN como modelos discriminatorios. Esto significa que dado un set de datos de entrada, un modelo lo etiquetará con cierta etiqueta. Por ejemplo, la clasificación de imágenes MNIST entre 0 y 9.

Uno podría pensar esto de forma opuesta, en lugar de predecir y, dado ciertas características, podríamos intentar predecir las características de entrada dato una clase, $P(X|Y = y)$.

Por ejemplo, un modelo generativo permitirá crear una imagen de un dígito escrito a mano. Los procesos clásicos generativos son Variational Autoencoders (VAE) y Generative Adversarial Networks (GAN).

Generative Adversarial Networks

Quizás es uno de los modelos mas populares de generación hoy en día. Se introdujo el año 2014 por Ian Goodfellow et al. Este marco de trabajo puede funcionar con cualquier tipo de datos, pero es muy popular por su capacidad en generación de imágenes.

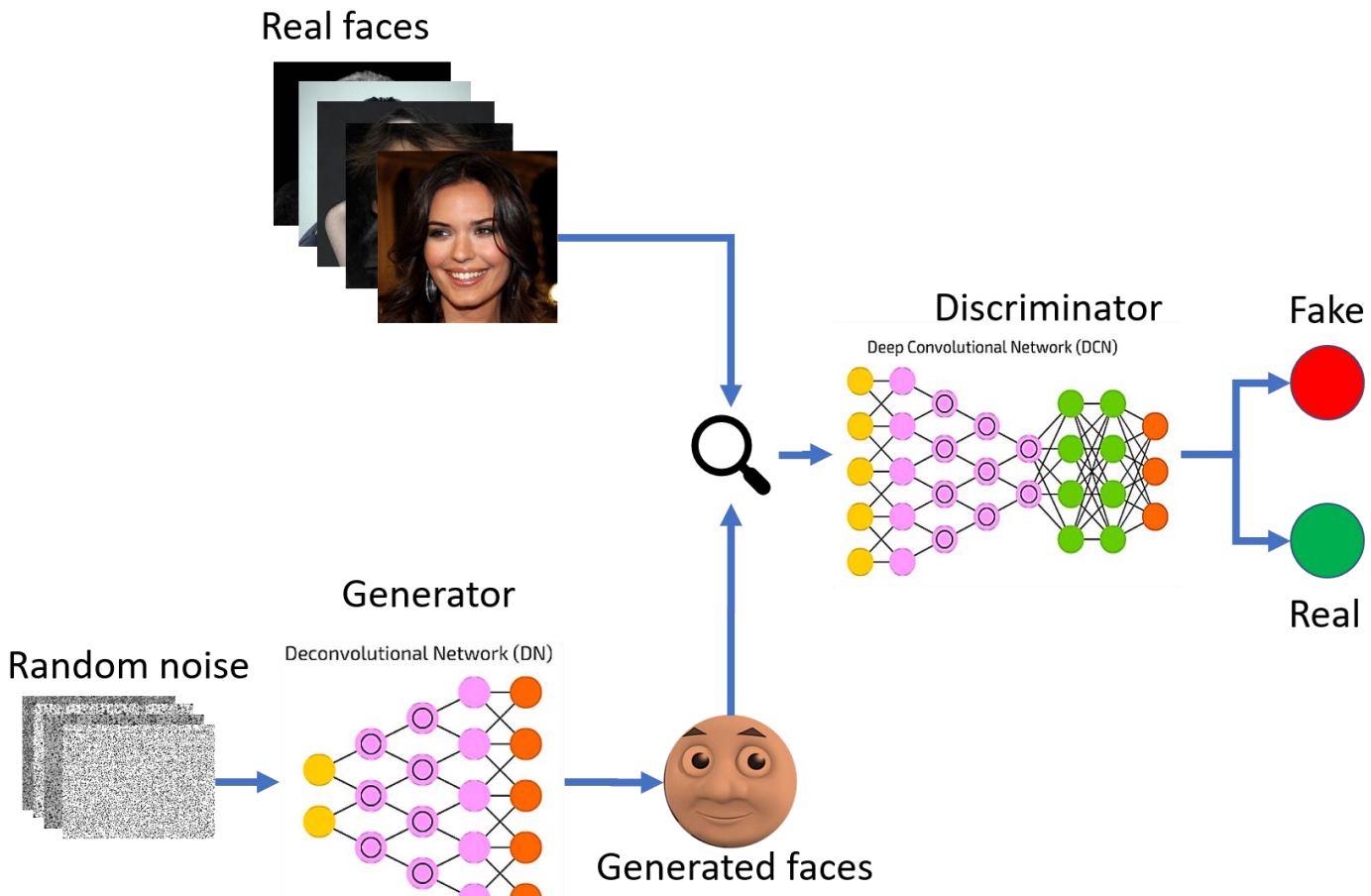
Generative Adversarial Networks

Un GAN es un sistema de dos componentes (rrnn):

- Generador: Corresponde al modelo generativo en si mismo. Toma una probabilidad de distribución como input y genera una imagen de salida realista.
- Discriminador: Toma dos entradas alternadas, la real de entrenamiento y la falsa generada. Luego intenta determinar si la imagen de entrada viene de la real o de la generada.

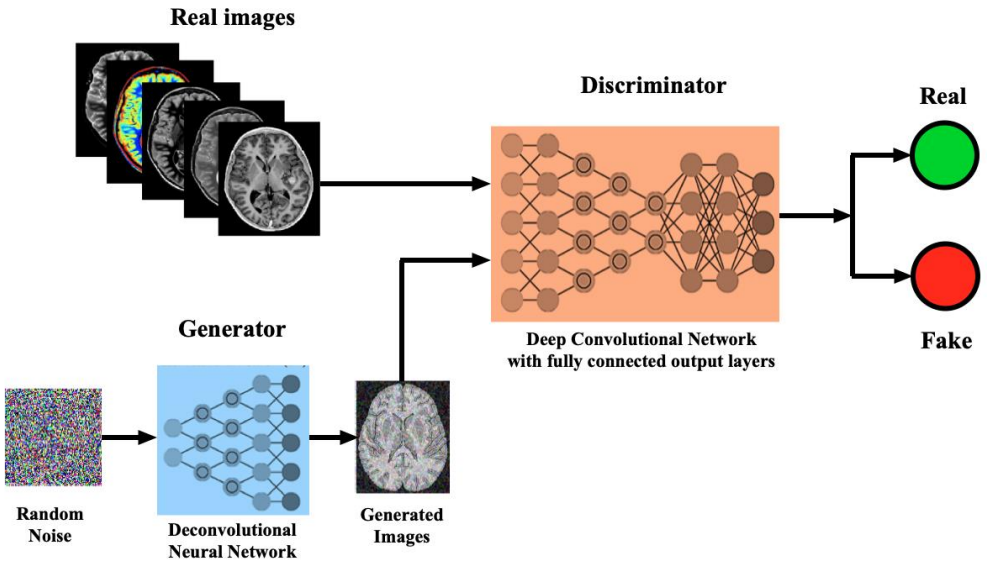
Pese a que el discriminador hace clasificación, una GAN no es supervisada, dado que no necesitamos etiquetar las imágenes.

Generative Adversarial Networks



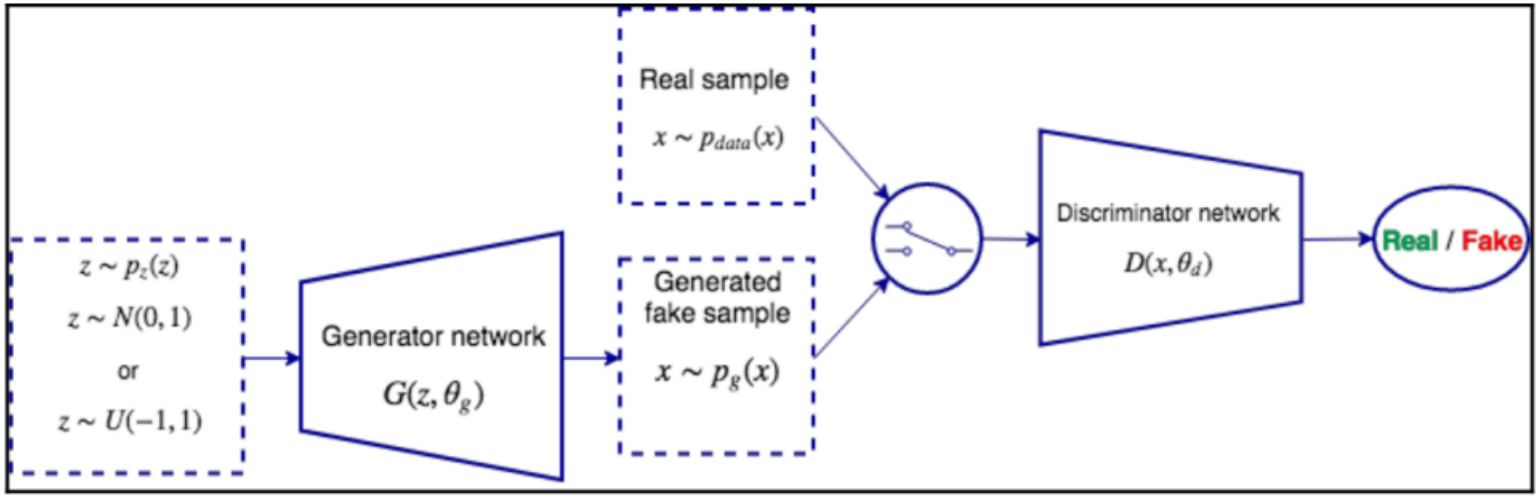
Entrenando una GAN

- El objetivo general de un generador es producir imágenes realistas, por ejemplo vehículos. Entrenaremos para ello el generador y el discriminador de forma secuencial separada (uno después del otro), alternando múltiples veces.



Notación

- Generador $G(z, \theta_g)$ donde θ_g son los pesos del generador
- Discriminador $D(x, \theta_d)$, donde θ_d son los pesos del discriminador
- Durante el entrenamiento denotamos el discriminador y la función de perdida del generador con J^D y J^G respectivamente.



Entrenamiento

GAN es distinto a DNN standard, ya que hay dos redes. Que puede ser pensada cono un juego secuencial minimax Zero-sum

- Secuencial: Los jugadores toman turnos, uno después del otro. El discriminador intenta minimizar J^D , ajustando los pesos. Luego el generador intenta minimizar J^D ajustando los pesos. Este proceso se repite múltiples veces.
- Zero-sum: La ganancia-perdida de un jugador, es balanceada por la ganancia-perdida del otro.

$$J^G = -J^D$$

- Minimax: La estrategia de un el primer jugador (generador), es minimizar el puntaje máximo del oponente (discriminador). Cuando se entrena el discriminador, se hace mejor en distinguir entre muestras reales y falsas (minimizando J^D). Luego cuando se entrena el generador, este ajusto el nivel de el discriminador-mejorado (minimizar J^G es equivalente a maximizar J^D).

Entrenamiento

Asumamos que luego de un gran número de entrenamientos, ambas J^G y J^D están en un mínimo local. Entonces la solución al juego minmax es llamada equilibrio de Nash.

El equilibrio de Nash ocurre cuando uno de los actores no cambia su acción, a pesar de que el otro actor podría hacerlo. Equilibrio de Nash en GAN ocurre cuando el generador es tan bueno que el discriminador no es capaz de distinguir entre la imagen generada y la real.

Entrenando el Discriminador

El discriminador es una red neuronal de clasificación que puede ser entrenada de forma standard. Sin embargo el set de entrenamiento esta compuesto de partes iguales (real y generadas).

- Dependiendo del input (real o generado), tenemos dos caminos:
  - Seleccionar la muestra de los datos reales y usarla para producir.
  - Generar una muestra falsa, acá generador y discriminador trabajan como una red simple.
- Calculamos la función de perdida, que refleja la dualidad de los datos de entrenamiento.
- Propagamos el error –backpropagate– y actualizamos los pesos. Acá sólo actualizaremos los pesos del discriminador,

Entrenando el Generador

Entrenaremos el generador haciendo lo mejor y engañando al discriminador. Para hacer esto necesitamos ambas redes, similar a la forma que entrenamos antes:

- Comenzamos con un vector al azar z, y lo alimentamos a través del generador y discriminador, para producir un output D(G(z)).
- La función de pérdida es la misma que la perdida del discriminador. Sin embargo, nuestro objetivo es maximiarla, en lugar de minizarla, dado que queremos engañar al discriminador.
- En el paso backward, los pesos del discriminador son bloqueados, y sólo podemos ajustar θg. Esto fuerza a maximizar la pérdida del discriminador, mediante la mejora del generador, en lugar de empeorar el discriminador.

En esta parte solo usamos datos generados.

https://www.technologyreview.com/2017/01/04/154761/5-big-predictions-for-artificial-intelligence-in-2017/