Generación de Imágenes Sintéticas de Palas de Aerogeneradores utilizando DCGAN y LSGAN

Ana Cortazzo
Maestría en Ciencia de Datos y Aprendizaje Automático
Facultad de Ingeniería, UdelaR
Montevideo, Uruguay
Email: ana.cortazzo@fing.edu.uy

***Resumen*—El uso de Redes Generativas Adversarias (GAN) para la generación de imágenes ha ganado relevancia en el último tiempo, en particular para aquellas aplicaciones de visión por computadora que requieren un gran numero de imágenes para su entrenamiento que no siempre está disponible. En este trabajo se presentan dos arquitecturas de GANs, DCGAN y LSGAN, para la generación de imágenes realistas de palas de aerogeneradores, con el objetivo de ser utilizadas para aumentar la cantidad de datos disponibles para el entrenamiento de modelos de visión por computadora para la detección de fallas. Los datos utilizados para el entrenamiento fueron imágenes de palas de aerogeneradores tomadas con drones, adquiridas como parte del proyecto "Sistema multi-UAV para la InSpección de Palas de AerogeneradoreS - SUperAVISPAS". El entrenamiento de los modelos se realizó durante números de épocas variables y se analizó la evolución en la calidad de imágenes a medida que aumentan las iteraciones. Finalmente se analizan las imágenes obtenidas luego de 100 épocas y se presentan métricas para evaluar y comparar las redes. Los resultados muestran que ambas redes tienen características similares, presentando resultados aceptables para la generación de imágenes.**

I. Introducción

En el marco de la transformación energética que ha llevado a cabo Uruguay en los últimos años, la energía eólica tiene un rol fundamental, representando el 31 % de la potencia instalada para la generación de energía eléctrica. Esto trae consigo la necesidad de profundizar en el desarrollo de conocimientos técnicos y tecnológicos en operación y mantenimiento de parques eólicos, que busquen reducir los los costes de mantenimiento y los tiempos de inactividad de los aerogeneradores.

Las palas de los aerogeneradores son un elemento clave en el funcionamiento de los mismos; la presencia de fallas o daños en las palas reduce la eficiencia en la generación de energía, causando pérdidas económicas y aumentando el riesgo de accidentes [1]. La inspección periódica del estado de las palas y las tareas de mantenimiento preventivo y correctivo son fundamentales para el funcionamiento seguro de los parques eólicos. En este contexto, se vuelve relevante el desarrollo de un sistema de inspección de daños en las palas y detección automática de las mismas.

Uno de los desafíos asociados a estos sistemas de visión por computadora es la necesidad de contar con un gran número de imágenes para entrenar los modelos, que no siempre están disponibles con la calidad y características requeridas. Para abordar este problema, el uso de técnicas de generación de datos, como las *Generative Adversarial Networks* (GAN)[1] ha ganado popularidad para la generación de imágenes realistas [2, 3, 4] o en la mejora de los datos [5].

En este contexto, el presente trabajo se centra en la aplicación de dos arquitecturas de GANs, DC-

[1]En español: Redes Generativas Adversarias o Antagónicas

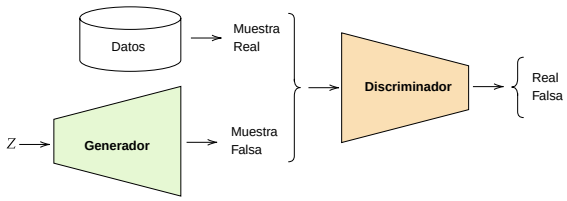


Figura 1: Diagrama general de una GAN

GAN (*Deep Convolutional GAN*) y LSGAN (*Least Squares GAN*), para la generación de imágenes realistas de palas de aerogeneradores, con el objetivo de ser utilizadas para aumentar la cantidad de datos disponibles para el entrenamiento de modelos de visión por computadora para la detección de fallas. Se presentan en detalle ambas arquitecturas, los parámetros utilizados en el diseño y la preparación de los datos para el entrenamiento. La evaluación y comparación de los modelos se realizará en términos de calidad visual de las imágenes generadas, análisis de la evolución de la pérdida y *accuracy* del discriminador. Todos los códigos desarrollados se encuentran en el Repositorio GitHub.

II. Trabajos relacionados

II-A. GAN

Las GAN son un tipo de modelo generativo que pueden interpretarse como una batalla entre dos adversarios: el generador (G) y el discriminador (D) (ver Fig.1). El generador intenta convertir ruido aleatorio (vectores del espacio latente z) en observaciones que parezcan haber sido muestreadas del conjunto de datos original, mientras que el discriminador intenta predecir si una observación proviene del conjunto de datos original o es una falsificación del generador [6]. Durante el entrenamiento de las GAN, el generador y el discriminador se actualizan de forma simultanea.

Sea x cualquier muestra real de datos, z las variables de entrada del espacio latente y $G(z) = x'$ cualquier muestra falsa. El objetivo de G es estimar la distribución p_{data} de los datos reales de tal forma que se maximice la probabilidad de que cualquier muestra falsa sera clasificada como real por D, es decir, que se maximice $D(G(z))$; el objetivo del discriminador es maximizar la probabilidad de que cualquier muestra real sea clasificada como real, i.e. maximizar $D(x)$ al mismo tiempo que se minimiza la probabilidad que que cualquier muestra falsa sea clasificada como real, es decir, minimizar $D(G(z))$. El entrenamiento de las GANs consiste en este problema de optimización **minmax**, que puede ser formulado como:

$$\begin{aligned}\text{mín}_G\, \text{máx}_D\, V(D,G) = \mathbb{E}_{x\sim p_{data}}\left[\log D(x)\right] \\ +\mathbb{E}_{z\sim p_z}\left[\log(1-D(G(z)))\right]\end{aligned}$$

Si bien las GANs han sido ampliamente utilizadas en tareas de visión por computadora y generación de imágenes, siguen existiendo tres retos principales: la calidad de la imagen, la estabilidad del entrenamiento y la evaluación de las GANs [7]. En este sentido han surgido numerosas investigaciones que buscan dar respuesta a estos retos.

II-B. DCGAN y LSGAN para la generación de imágenes

Con el objetivo de mejorar la calidad de las imágenes, Radford et al. (2015)[8] introducen las DCGAN, que son una mejora significativa de las GANs originales al utilizar redes convolucionales profundas para generar imágenes realistas. Los autores también establecen algunas directrices básicas a seguir en la arquitectura de las DCGAN para garantizar la estabilidad:

- Reemplazar capas de *pooling* con convoluciones estratificadas (paso mayor que uno) en el discriminador y convoluciones transpuestas en el generador.
- Utilizar *batch normalization* tanto en el generador como en el discriminador.
- Eliminar capas ocultas completamente conectadas para arquitecturas más profundas.
- Utilizar activación `ReLU` en el generador para todas las capas, excepto para la salida, que utiliza Tangente hiperbólica (`Tanh`). Utilizar activación `LeakyReLU` en el discriminador para todas las capas.

Generalmente la función de pérdida utilizada en en entrenamiento de las GANs es la de entropía cruzada binaria[2] (BCELoss); Mao et al. (2017)

[2]En inglés *Binary Cross Entropy*

[9] reportan que esta función de pérdida puede provocar el conocido problema de desvanecimiento del gradiente durante el proceso de aprendizaje, y proponen las LSGAN para superar este problema, las cuáles adoptan la función de pérdida de mínimos cuadrados[3] para el discriminador, manteniendo una arquitectura similar a las DCGAN.

Las LSGAN utilizan un esquema de codificación $a - b$ para el discriminador, donde a y b son las etiquetas de los datos falsos y los datos reales, respectivamente. Las funciones objetivo de las LSGAN pueden definirse como:

$$\begin{aligned} \text{mín}_D\, V(D) = \tfrac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[(D(x)-b)^2] \\ +\tfrac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z))-a)^2] \end{aligned}$$

$$\text{mín}_G\, V(G) = \tfrac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z))-c)^2]$$

dónde c representa el valor que G quiere que D crea para los datos falsos [9].

III. Metodología

III-A. Procesamiento de datos

Los datos utilizados para este trabajo consisten en un conjunto de 9156 imágenes de palas de aerogeneradores, tomadas con drones, adquiridas como parte del proyecto "Sistema multi-UAV para la InSpección de Palas de AerogeneradoreS - SUperAVISPAS"[4]. Los datos se encuentran organizados en 36 carpetas que corresponden a los aerogeneradores, dentro de cada una de ellas, se encuentran tres subcarpetas correspondiente a cada una de las palas, y dentro se organizan en 4 categorías: Ataque, Fuga, Depresión y Presión.

El tamaño de cada imagen es de $4608 \times 3456 \times 3$ el cuál debe ser adaptado a las dimensiones de los modelos: $64 \times 64 \times 3$. Se procesaron los datos para obtener un dataset con las dimensiones adecuadas al modelo y se guardaron en formato de tensor de `PyTorch`. En la Figura 2 se presenta una muestra aleatoria de imágenes reales por cada una de las categorías, luego del procesamiento de las mismas.

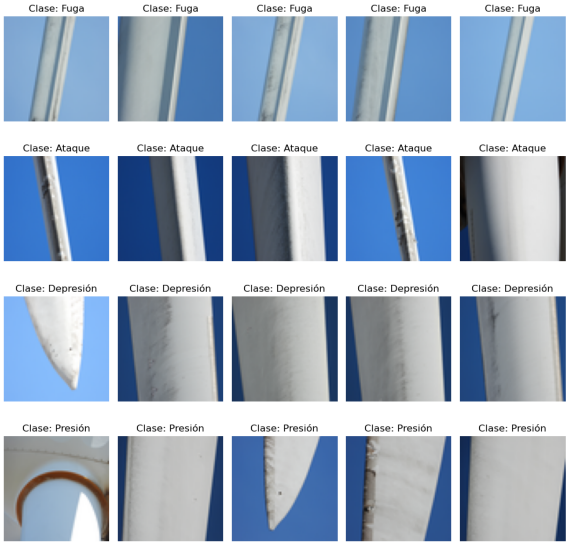


Figura 2: Muestra de imágenes reales

Finalmente se definió el `dataloader` con un tamaño de lote (`batch size`) de 128 y se normalizaron las imágenes para que tengan una media de 0,5 y una desviación estándar de 0,5.

III-B. Modelos propuestos

Se presentan dos modelos para la generación de imágenes sintéticas de palas de aerogeneradores: DCGAN y LSGAN. El desarrollo de ambos modelos se realizó utilizando la biblioteca de código abierto `PyTorch`. A continuación se detallan las arquitecturas propuestas para cada modelo.

DCGAN

El modelo DCGAN presentado en este trabajo consiste en dos redes neuronales convolucionales: el generador (G) y el discriminador (D). La arquitectura[5] para cada una de las redes se muestra en la Tabla I.

El generador recibe un vector del espacio latente z que pasa a través de 5 capas convolucionales transpuestas que mapean z al espacio de datos con el tamaño $64 \times 64 \times 3$. Luego de cada capa

[3] En inglés *Least Squares*

[4] https://superavispas.pages.fing.edu.uy/homepage/

[5] Adaptada de: PyTorch: DCGAN Tutorial

Tabla I: Arquitectura de la DCGAN.
Conv refiere a la capa convolucional `Conv2d`, ConvT indica la capa convolucional transpuesta (`ConvTranspose2d`), BNorm refiere a `BatchNorm2d` y LReLU refiere a la función `LeakyReLU`

Generador	**Discriminador**
Entrada (z=100)	Entrada (64x64x3)
ConvT (100, 512)	Conv (3, 64)
BNorm, ReLU	LReLU
ConvT (512, 256)	Conv (64, 128)
BNorm, ReLU	BNorm, LReLU
ConvT (256, 128)	Conv (128, 256)
BNorm, ReLU	BNorm, LReLU
ConvT (128, 64)	Conv (256, 512)
BNorm, ReLU	BNorm, LReLU
ConvT (64, 3)	Conv (512, 1)
Tanh	Sigmoid

Tabla II: Arquitectura de la LSGAN.
Conv refiere a la capa convolucional `Conv2d`, ConvT indica la capa convolucional transpuesta (`ConvTranspose2d`), BNorm refiere a `BatchNorm2d` y LReLU refiere a la función `LeakyReLU`

Generador	**Discriminador**
Entrada (z=100)	Entrada (64x64x3)
ConvT (100, 512)	Conv (3, 64)
BNorm, ReLU	LReLU
ConvT (512, 256)	Conv (64, 128)
BNorm, ReLU	BNorm, LReLU
ConvT (256, 128)	Conv (128, 256)
BNorm, ReLU	BNorm, LReLU
ConvT (128, 64)	Conv (256, 512)
BNorm, ReLU	BNorm, LReLU
ConvT (64, 3), Tanh	Conv (512, 1)

convolucional transpuesta se aplica la normalización por lotes (*batch normalization*) que ajusta y normaliza los valores de activación para ayudar en el entrenamiento y estabilidad de la red, y se utiliza la función de activación ReLU (*Rectified Linear Unit*). En la última capa se utiliza la función de activación `Tanh` para asegurar que los valores de salida estén en el rango $[-1, 1]$.

El discriminador recibe imágenes reales e imágenes generadas por G y tiene como objetivo clasificarlas correctamente como reales o falsas. La arquitectura propuesta para la red D consiste en 5 capas convolucionales, que reciben imágenes de dimensión $64 \times 64 \times 3$ y las mapean a un valor en el rango $[0, 1]$ que puede interpretarse como la probabilidad de que la imagen sea real o no; para esto se utiliza en la última capa la función de activación sigmoide. Al igual que en la arquitectura de G, luego de cada convolución se aplica *batch normalization* y se utiliza como función de activación LeakyReLu (*Leaky Rectified Linear Unit*).

Cómo función de pérdida se utiliza la entropía cruzada binaria (*Binary Cross Entropy*) y el método de Adam como optimizador. El entrenamiento de la DCGAN se realiza durante un número determinado de épocas, se calcula accuracy del discriminador y la evolución de la pérdida para ambas redes.

LSGAN

El modelo LSGAN presentado en este trabajo consiste en dos redes neuronales convolucionales: el generador (G) y el discriminador (D). La arquitectura para cada una de las redes se muestra en la Tabla II.

La arquitectura de G es idéntica a la presentada para la DCGAN; para D la única diferencia es que se elimina la activación sigmoide en la última capa, que ya no es necesaria en la LSGAN. Cómo función de pérdida se utiliza mínimos cuadrados (*Least Squares*) y se mantiene el método de Adam como optimizador. El entrenamiento de la LSGAN se realiza durante un número determinado de épocas, se calcula accuracy del discriminador y la evolución de la pérdida para ambas redes.

III-C. Evaluación de los modelos

Para evaluar y comparar los modelos se consideran estrategias cualitativas, analizando la calidad de las imágenes obtenidas mediante inspección visual, y los siguientes parámetros cuantitativos:

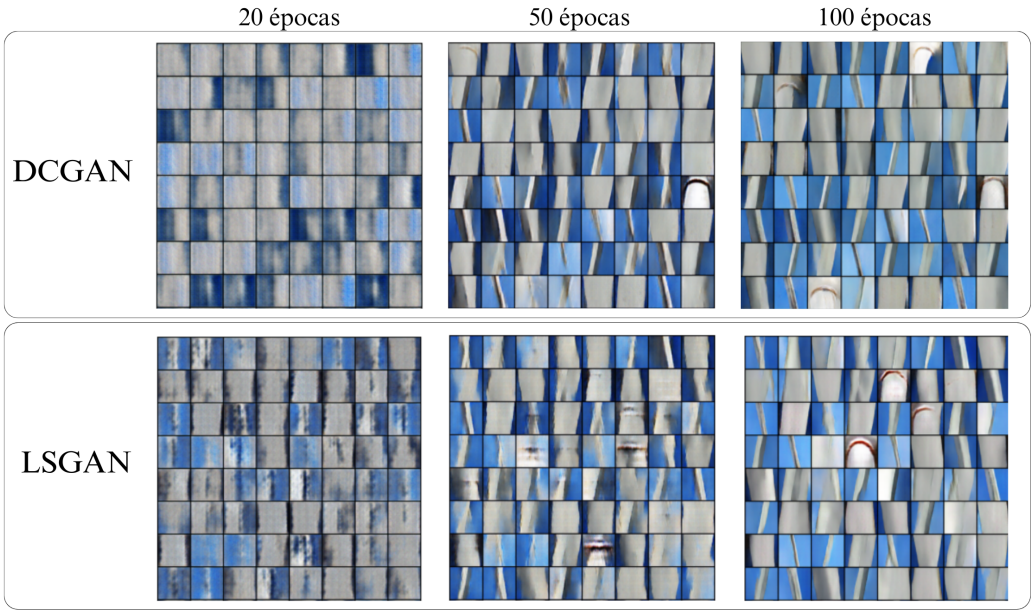


Figura 3: Evolución de las imágenes generadas para diferentes épocas de entrenamiento

- Tiempo de entrenamiento
- Tamaño del modelo (parámetros)
- Memoria RAM requerida durante el entrenamiento
- Evolución de la pérdida en el discriminador y en el generador
- Accuracy del discriminador
- Índice de Similitud Estructural (SSIM)

Se define el accuracy (o acierto) como la proporción de instancias clasificadas correctamente sobre el total de instancias. Sea V_p y V_n los verdaderos positivos y negativos y F_p y F_n los falsos positivos y negativos respectivamente, se puede calcular como:

$$acc = \frac{V_p + V_n}{V_p + V_n + F_p + F_n}$$

Calcular el accuracy del discriminador es una estrategia para evaluar el comportamiento del mismo para diferenciar las muestras falsas de las reales y puede ser un parámetro más a ser considerado para comparar las diferentes arquitecturas de GAN propuestas. Este valor no debe ser considerado como una medida del rendimiento o desempeño de la GAN, ya que refiere solo al discriminador.

Una forma para determinar cuál de las imágenes generadas tiene mayor similitud con las reales, es determinar el Índice de Similitud Estructural (SSIM)[6], una métrica utilizada para medir la calidad de una imagen que se basa en tres características fundamentales: luminancia, contraste y estructura; el valor de SSIM va desde 0 (ninguna similitud) a 1 (imágenes idénticas). En este trabajo se determinó el SSIM utilizando la biblioteca `OpenCV` de `Python`.

IV. Experimentación

Todos los experimentos fueron realizados en una computadora con las siguientes especificaciones: 12GB de memoria RAM, procesador Intel Core i7-1065G7 con 8 núcleos de 1.30GHz. El sistema operativo utilizado fue Ubuntu 22.04, con Python 3.9.16 como entorno de desarrollo.

[6]En inglés: *Structural Similarity Index*

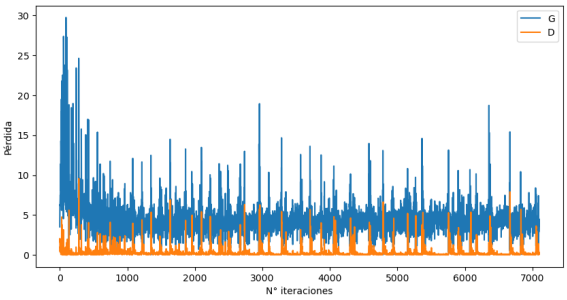


Figura 4: Evolución de la pérdida para G y D de la DCGAN durante el entrenamiento en 100 épocas

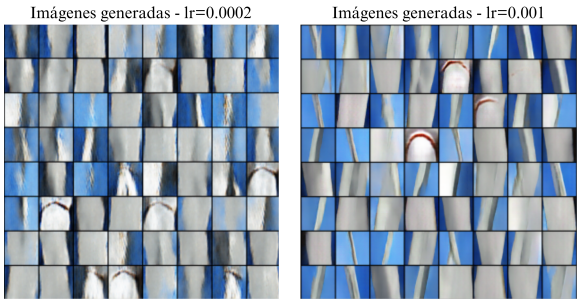


Figura 5: Imágenes generadas pos la LSGAN para diferentes valores de tasa de aprendizaje

IV-A. Modelo: DCGAN

Con los datos preparados y las arquitecturas definidas se entrenó el modelo durante 20, 50 y 100 épocas; esto nos permite ver con mayor claridad la evolución en la calidad de las imágenes a medida que aumenta el número de iteraciones (ver Figura 3).

Todos los pesos de la red neuronal se inicializaron utilizando una distribución normal con una media de 0,0 para las capas convolucionales, de 1 para las capas de normalización y una desviación estándar de 0,02. Los parámetros utilizados fueron los siguientes:

- Tamaño del vector del espacio latente `nz = 100`
- Número de filtros de las capas convolucionales de G `ngf = 64`
- Número de filtros de las capas convolucionales de D `ndf = 64`
- Optimizador:
  - Tasa de aprendizaje `lr = 0.0002`
  - `beta1 = 0.5`, `beta2 = 0.999`

La evolución de la pérdida para G y D durante 100 épocas se presenta en la Figura 4. Se observa que la pérdida de ambas redes disminuye a medida que aumentan las iteraciones, esto es esperable para G, ya que debe mejorar constantemente en las imágenes que genera, y se espera que la pérdida de D aumente a medida que G se perfecciona. El ideal es encontrar un equilibrio entre ambas redes.

IV-B. Modelo: LSGAN

Con los datos preparados y las arquitecturas definidas se entrenó el modelo con los mismos parámetros de la DCGAN durante 100 épocas. Mediante una inspección visual de la calidad de las imágenes, se detectó que el resultado no era el esperado por lo que se decidió modificar la tasa de aprendizaje del optimizador a un valor de `lr=0.001`, manteniendo los demás parámetros en con los mismos valores. En la Figura 5 se puede observar la diferencia de las imágenes generadas luego de 100 épocas para los dos valores de tasa de aprendizaje, obteniendo mejores resultados para el valor de `lr=0.001`.

Fijado el nuevo valor de tasa de aprendizaje y manteniendo los demás parámetros idénticos a los utilizados en el entrenamiento de la DCGAN, se entrenó el modelo durante 20, 50 y 100 épocas; la Figura 3 presenta las imágenes obtenidas en cada entrenamiento. Finalmente, la evolución de la pérdida para G y D durante 100 épocas se presenta en la Figura 6.

IV-C. Análisis de los resultados

Analizando la Figura 3 es posible observar que la calidad de las imágenes generadas experimentó una mejora a medida que aumenta el numero de épocas de entrenamiento, las imágenes luego de 100 épocas son las que tienen mejores características, siendo más realistas y de mejor calidad. Se puede observar también que las imágenes generadas por la DCGAN luego de 50 épocas presentan mejores características que las generadas por la LSGAN,

Tabla III: Resumen de las métricas obtenidas durante el entrenamiento de los modelos luego de 100 épocas

	DCGAN	LSGAN
N° de épocas	100	100
Parámetros de red	3576704	3576704
Memoria utilizada (MB)	3844,191406	4211,363281
Tiempo total transcurrido (minutos)	249,6895	239,7085
Accuracy D	0,5391	0,5000
SSIM	0.27246	0.27239

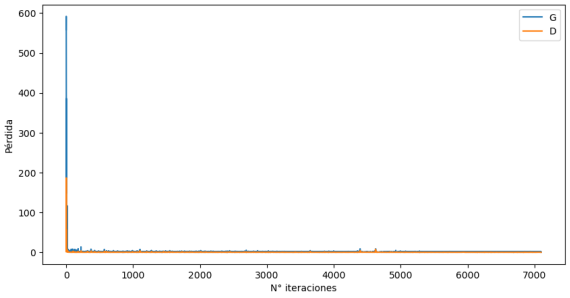


Figura 6: Evolución de la pérdida para G y D de la LSGAN durante el entrenamiento en 100 épocas

esto puede indicar que la DCGAN reuiere menor cantidad de iteraciones para alcanzar los resultados óptimos.

Con el objetivo de comparar las imágenes generadas con las reales, en la Figura 7 se presentan las imágenes obtenidas por cada uno de los modelos luego de entrenarlos durante 100 épocas. Esto sugiere que tanto la DCGAN como la LSGAN son capaces de aprender patrones significativos en los datos y generar imágenes más realistas con un entrenamiento más prolongado.

El la Tabla III se presenta el valor de SSIM obtenido para cada modelo al comparar las imágenes generadas por G con las imágenes reales. Se observa que el valor es similar para ambos modelos y está alrededor de 0.27, lo que sugiere dos cosas: que las imágenes generadas por ambas redes son similares entre sí pero aún hay diferencias significativas con las imágenes reales.

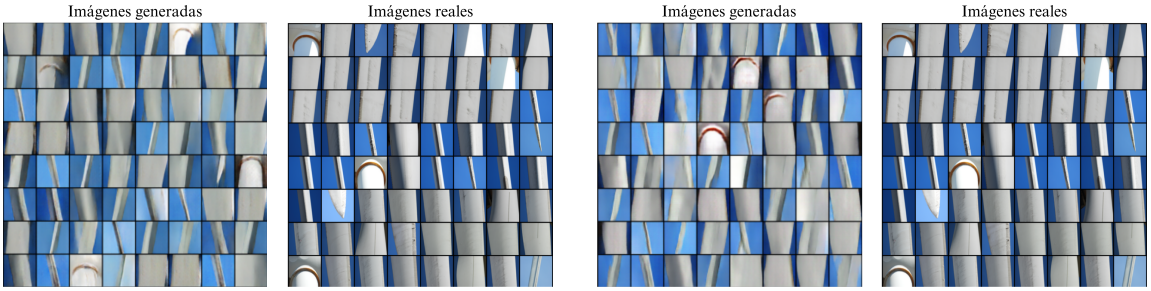
La tabla III resume las métricas utilizadas para evaluar los modelos. Se puede observar que tanto la DCGAN como la LSGAN tienen el mismo número de parámetros de red, un resultado esperado ya que las arquitecturas son idénticas en el número de capas y dimensiones involucradas en cada una de ellas. Este valor nos indica que los modelos con comparables en términos de complejidad. La LSGAN requirió una cantidad de memoria ligeramente superior a la DCGAN, mientras que su tiempo de entrenamiento fue ligeramente menor. El valor de accuracy de D fue similar también para ambos modelos; este valor sugiere que G fue capaz que generar imágenes falsas que discriminador clasificó como reales casi la mitad de las veces.

V. Conclusión

A partir de los resultados obtenidos se puede observar que tanto la DCGAN como la LSGAN son efectivas para la generación de imágenes de palas de aerogeneradores. Comparando ambas arquitecturas se puede concluir que el rendimiento en términos de calidad de imágenes, tiempo de entrenamiento y memoria utilizada es similar en ambas, lo que sugiere que las arquitecturas propuestas son comparables en términos de complejidad y resultados.

Si bien las imágenes obtenidas son satisfactorias, aún no cumplen con los estándares de calidad requeridos para ser utilizadas en un sistema automático de detección de fallas. Para obtener mejores resultados se necesitan arquitecturas más potentes que sean capaces de trabajar con imágenes en alta resolución, y para esto se requiere contar con hardware adecuado que sea capaz de manejar grandes volúmenes de datos de alta calidad, y redes neuronales con un número de parámetros aún mayor.

Los resultados obtenidos señalan la necesidad de continuar investigando y desarrollando arquitecturas



(a) DCGAN (b) LSGAN

Figura 7: Imágenes generadas por los modelos luego del entrenamiento de 100 épocas

más avanzadas para satisfacer las demandas de calidad y resolución en la generación de imágenes sintéticas para aplicaciones específicas como la inspección de palas de aerogeneradores.

REFERENCIAS

[1] Liang Lv, Zhongyuan Yao, Enming Wang, Xin Ren, Ran Pang, Hua Wang, Yu Zhang, and Hao Wu. Efficient and accurate damage detector for wind turbine blade images. *IEEE Access*, 10:123378–123386, 2022.

[2] Lakkshmi Yogesh NA, G Bharathraj, Sanjay Prasanth AS, S Shreyas, and CB Rajesh. Generating synthetic dataset for cotton leaf using dcgan. In *2023 4th International Conference on Signal Processing and Communication (ICSPC)*, pages 249–252. IEEE, 2023.

[3] Nugraha Priya Utama and Muhammad Faris Muzakki. Utilizing generative adversarial network for synthetic image generation to address imbalance challenges in chest x-ray image classification. 17:373–384, Dec. 2023.

[4] Zecheng Li and Qianduoer Wan. Generating anime characters and experimental analysis based on dcgan model. In *2021 2nd International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*, pages 27–31. IEEE, 2021.

[5] Shiyu Zhou and Hongwei Ma. Surface defect data set enhancement method for wind turbine based on res-dcgan. In *2022 2nd International Symposium on Artificial Intelligence and its Application on Media (ISAIAM)*, pages 83–88. IEEE, 2022.

[6] David Foster. *Generative deep learning*. O'Reilly Media, Inc., 2023.

[7] Xudong Mao and Qing Li. *Generative adversarial networks for image generation*. Springer, 2021.

[8] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[9] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2017.