Escola Tècnica Superior d'Enginyeria ETSE-UV

MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS

TRABAJO DE FIN DE MÁSTER

REDES NEURONALES ADVERSARIAS (GANS) PARA DETECCIÓN DE NUBES Y GENERACIÓN DE ESCENAS A PARTIR DE IMÁGENES DEL SATÉLITE DE LA NASA LANDSAT 8

AUTOR:
SERGIO DAVID PÉREZ NAVARRO

TUTORES:
JORDI MUÑOZ MARI
VALERO LAPARRA PÉREZ-MUELAS
SEPTIEMBRE, 2018

VNIVERSITAT DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria ETSE-UV

MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS

TRABAJO DE FIN DE MÁSTER

REDES NEURONALES ADVERSARIAS (GANS) PARA DETECCIÓN DE NUBES Y GENERACIÓN DE ESCENAS A PARTIR DE IMÁGENES DEL SATÉLITE DE LA NASA LANDSAT 8

AUTOR:
SERGIO DAVID PÉREZ NAVARRO

TUTORES:
JORDI MUÑOZ MARI
VALERO LAPARRA PÉREZ-MUELAS

SEPTIEMBRE, 2018

TRIBUNAL:

PRESIDENTE/A:

VOCAL 1:

VOCAL 2:

FECHA DE DEFENSA:

CALIFICACIÓN:

Índice general

Abstract

Estamos en plena revolución tecnológica en cuanto aprendizaje máquina se refiere, y uno de los grandes retos es la identificación de elementos y la creación de datos sintéticos. En este trabajo se ha entrenado una red neuronal adversaria construida por dos redes convolucionales (CNN) (*Convolutional Neural Network*) en el lenguaje de programación Python 3.5 con la librería PyTorch versiones 3.0 y 4.0. Se ha realizado el entrenamiento empleando centenares de imágenes provenientes del satélite Landsat 8 de la NASA. Los resultados muestran que la identificación de nubes es del 0.8 en la métrica Sørensen–Dice. Además ha sido posible una satisfactoria generación de imágenes sintéticas partiendo de una máscara inicial.

Introducción

> Artificial intelligence is the new electricity
>
> Andrew Ng

El aprendizaje máquina y más en concreto, el aprendizaje profundo están cambiando el mundo. La idea de que esto va a ser así está cada vez más presente en la sociedad. En el futuro, la gente trabajará de forma diferente, se divertirá de forma diferente, se transportará de forma diferente y vivirá en general, de forma diferente gracias a los cambios que están ocurriendo estos años.

Este trabajo no es más que uno de los miles de casos donde se puede aplicar con un cierto nivel de éxito. La evolución de estos algoritmos no tiene intención de dejar de mejorar. El avance del conocimiento sobre el aprendizaje máquina es inabordable para cualquier persona y la gente ya no puede estar al día de todo.

Detrás de las librerías más famosas se encuentran nombres como Google, Facebook, Microsoft, Nvidia, Uber y un largo etcétera de titanes de la tecnología. Cada una tiene un ejército de personas trabajando para mejorar las herramientas. Las publicaciones que consisten en las cientos de combinaciones de algoritmos y parámetros, para encontrar esa puntuación en determinada métrica, que es considerada en ese momento como el estándar, son cada vez más abundantes. El cómo hemos llegado a esta situación es debido a una combinación de grandes avances realizados por unos pocos, y la enorme capacidad de cálculo de la que disponemos ahora. Esto permite que miles y miles de personas tengan y produzcan nuevas ideas.

Por desgracia, como todo lo que está de moda, se crean rápidamente muchos problemas. Uno de los principales es el concepto de la caja negra. Cada vez abunda más gente que mantiene la opinión de que si funciona, aunque no lo entiendas, úsalo. Confiamos en la caja negra que nos dice qué camino hay que tomar. Este es un concepto práctico pero peligroso, pues no hay problema cuando estamos tratando de distinguir entre números escritos a mano. Si el cinco resulta ser un siete, no pasa nada. Pero cuando se trata de si la figura que ha captado la cámara del coche es un ser humano o una sombra, el sistema ha de tomar la decisión de si frenar poniendo en riesgo a

los pasajeros o continuar con normalidad, la situación, toma un cariz diferente. Casos como este nos hace replantearnos cómo de permisivos somos ante los fallos de la caja, y la aceptación de esta filosofía negra disminuye de forma drástica.

La maldición de la caja negra parece difícil de superar si tenemos en cuenta que actualmente las redes neuronales empleadas se componen de millones y millones de neuronas interaccionando entre sí. Es complicado tratar de entender qué ocurre, y aun suponiendo que se dedique un tiempo absurdamente grande para tratar de entenderlas (si es que se puede), ¿qué pasará cuando tengamos una nueva arquitectura de red, aun más complicada que la anterior? Una persona puede pensar que después de miles y miles de casos que la red ha *estudiado* dando una gran probabilidad de éxito, siempre va a ser mejor que un humano (que presenta una tasa de error superior).

Pero es bueno recordar, como ya se ha demostrado en diferentes ocasiones, que las redes son susceptibles de ataques para que den resultados radicalmente diferentes. Estos ataques son tan efectivos que basta con modificar un solo píxel, como podemos ver en [13]. Estos ataques no solo engañan para dar un resultado similar pero diferente, sino que dan un resultado completamente diferente. Este es un claro ejemplo de que algo todavía falla, y lo lleva haciendo ya unos años [9].

En la opinión del autor, esto es solo el comienzo de algo grande, pero hay un momento de ceguera colectiva que poco a poco será superado conforme modelos más robustos y mejor definidos tomen paso. Este es posiblemente el momento más excitante para la inteligencia artificial que ha tenido la humanidad. Es difícil imaginar qué se logrará en los próximos 5 o 10 años.

Es posible que los coches autónomos sean comunes, que los radiólogos existan para confirmar que la máquina no se equivoca [7] y [1], que las fábricas del mundo apenas tengan operarios humanos (esto empieza a ser una realidad hoy en día) o que cuando un usuario doméstico tenga problemas con su conexión a Internet, mantenga una conversación con un *chatbot* que sea capaz de comprender la explicación que le cuenta el usuario, y capaz de arreglar el problema.

El problema que queremos tratar de resolver no es simplemente segmentar nubes en fotografías por satélite. Es la capacidad de aprender a generar nuevos datos sintéticos que permitan ser empleados en otras aplicaciones. Como es el caso de tratar de entrenar una red con datos sintéticos. Esto puede ser de gran utilidad en campos como la medicina. Quizás en un futuro es posible el entrenamiento de redes aun más complejas empleando datos sintéticos que han sido producidos mediante esta técnica.

CAPÍTULO 1

Redes Neuronales Artificiales

> Any sufficiently advanced technology
> is indistinguishable from magic.
>
> Arthur C. Clarke

En este capítulo vamos a hablar de las redes neuronales artificiales o ANN (*Artificial Neural Network* por sus siglas en inglés) que son la base fundamental de este trabajo. Comenzaremos describiendo una pieza clave y que es anterior a la red neuronal: el perceptrón.

El perceptrón es la idea fundamental en la que se basan las neuronas, que son la unidad más básica de las ANN. Fue inventado en el año 1957 en el Cornell Aeronautical Laboratory por Frank Rosenblatt. Se trata de una técnica de aprendizaje supervisado para clasificación binaria que en esencia consiste en una función que relaciona unos datos de entrada con una salida. Esto se realiza mediante la siguiente operación. Cada uno de los datos de entrada es multiplicado por un número denominado peso, y el resultado de estas operaciones es sumado. A continuación, una función denominada función de activación evalúa si la salida ha de ser un uno o un cero, pues como ya se ha mencionado, los perceptrones dan resultados binarios. Matemáticamente la salida la operación es expresada de la siguiente forma:

$$y = f(x) = \begin{cases} 1 & \text{if } \theta \cdot \mathbf{x} + b > 0 \\ 0 & \text{if } \theta \cdot \mathbf{x} + b \leqslant 0 \end{cases} \tag{1.1}$$

donde θ es el vector que representa los pesos y $\mathbf{x}$ es el vector que representa los datos de entrada. Ambos vectores contienen números reales y la operación $\theta \cdot \mathbf{x}$ es el producto escalar $\sum_i^n \theta_i x_i$ donde n es el número de entradas que se introducen. b no es más que un parámetro de sesgo o *bias*. Los valores iniciales de los pesos son comúnmente inicializados con el valor cero o de forma aleatoria a números próximos a cero. A continuación, el perceptrón tiene que tratar de adaptar estos pesos con el fin de ajustar en mejor medida los resultados. Esto se realiza mediante la expresión:

$$\theta_i = \theta_i + (d - y)x_i \tag{1.2}$$

es decir, que para cada dato $\mathbf{x}$, se actualiza el vector de pesos θ. En ocasiones esta forma de actualizar los pesos es excesivamente rápida. Es por eso que la expresión (1.2) es a menudo expresada con un factor de aprendizaje α:

$$\theta_i = \theta_i + \alpha(d - y)x_i \tag{1.3}$$

Por lo tanto, los perceptrones esquemáticamente solo se componen de dos fases. La primera fase consiste en la entrada de datos, y en la multiplicación de cada uno de ellos por el peso correspondiente. Esto da paso a la siguiente fase: la predicción. En la terminología técnica se habla de capas. Por lo tanto tenemos una capa de entrada y una capa de salida.

Las redes neuronales tienen diferencias importantes. Tal como se ha comentado, la unidad básica de las redes neuronales son las neuronas o nodos, cuyo funcionamiento es muy similar al perceptrón. Estas neuronas se interconectan entre sí dando forma a la red. Concretamente, una red neuronal se compone de tres tipos de fases o capas: capa de entrada (*input*), capa oculta (*hidden*) y capa de salida (*output*). Cada una de las neuronas de una capa está conectada con todas las neuronas de las capas adyacentes. Cuando se dice que están conectadas, se expresa en el sentido de que la salida de la neurona es introducida en una neurona de la siguiente capa multiplicada por el peso correspondiente entre esas neuronas. Esto puede verse en la figura 1 expresado mediante flechas. Mientras que solo puede haber una capa de entrada y una de salida, las capas ocultas pueden ser múltiples y variar en cantidad de neuronas.

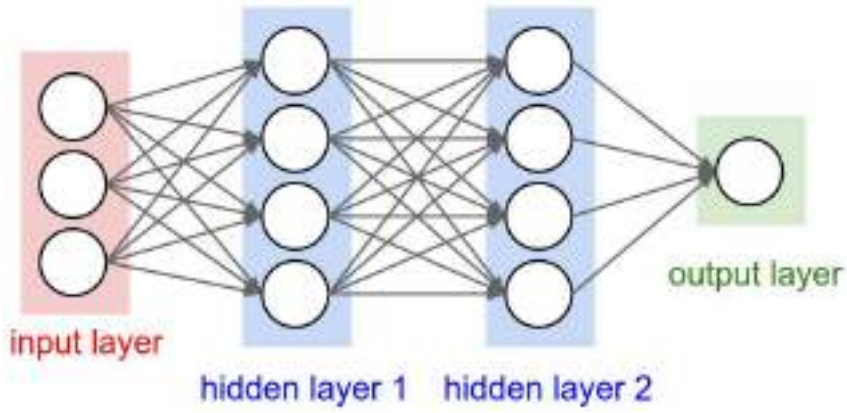


Figura 1.1: Esquema de una red neuronal con dos capas en la capa oculta.

El funcionamiento de la red neuronal tradicional tiene cuatro componentes. Una activación $a_j(t)$ con un umbral determinado por parámetro γ. Una función de activación para obtener la nueva activación en el siguiente paso $a(t+1) = f(a(t), X(t), \gamma)$ y una función de salida que da los resultados $o(t) = f_{out}(a(t))$.

1.1. Métrica

El coeficiente Sørensen–Dice es empleado para medir la similitud de dos muestras. Este coeficiente es comúnmente utilizado en medicina para la segmentación de órganos. En este tipo de tareas se recurre a redes convolucionales, como es el caso de la red U-

Net que aquí se emplea para crear una máscara en la localización del órgano que se pretende estudiar [11]. El coeficiente Sørensen–Dice esta definido de la forma:

$$SDC = 2\frac{|X \cup Y|}{|X| + |Y|} \tag{1.4}$$

donde $|X|$ e $|Y|$ representan el tamaño del vector. Cuando se aplica a resultados booleanos, la definición es la misma que la métrica F1:

$$DSF = \frac{2TP}{2TP + FP + FN} \tag{1.5}$$

CAPÍTULO 2

Red generativa antagónica

> The development of full artificial intelligence could spell the end of the human race. It would take off on its own, and re-design itself at an ever increasing rate. Humans, who are limited by slow biological evolution, couldn't compete, and would be superseded.
>
> Stephen Hawking

Las redes GAN (*Generative Adversarial Networks* por sus siglas en inglés) fueron definidas por Ian Goodfellow et al. en el año 2014 [3]. En este capítulo vamos a dar una explicación del funcionamiento de este tipo de redes.

Las redes generativas antagónicas consisten en una arquitectura donde dos redes neuronales tratan de cumplir objetivos diferentes. Una de estas redes es la denominada red generadora (G), mientras que la otra red se denomina discriminadora (D). El funcionamiento, que puede observarse en la figura 2.1, consiste en que la red G crea datos sintéticos mientras que la red D trata de discernir si los datos son auténticos o generados por la red G.

La función de pérdida de la red G es el acierto de la red D. Si la red D lo hace muy bien, quiere decir que la red G no está haciendo un buen trabajo engañándola. Lo mismo se puede expresar invirtiendo los papeles. Si la red G siempre logra engañar a la red D, significa que la red D no está haciendo un buen trabajo identificando casos sintéticos. Si una de las redes siempre acierta no quiere decir que esté bien entrenada. Puede darse el caso de que sea la otra la que esté haciendo un pésimo trabajo. Es por esto por lo que ambas redes han de evolucionar simultáneamente.

Para nuestro caso de imágenes por satélite el proceso completo se puede resumir de esta forma.

1. Se decide qué tipo de imagen se va a introducir en D.
2. Si es sintética, se genera un ruido aleatorio que se introduce en la red G para que esta genere una imagen que posteriormente es introducida en D. Si es real, se toma una de las imágenes de entrenamiento y se introduce en D.
3. La red D concluye si es real o sintética.
4. La red D comprueba si ha acertado y se entrena como una red neuronal normal.
5. Si la red D evaluó una imagen sintética, la red G se entrenará buscando la falsa identificación de la imagen por parte de D.

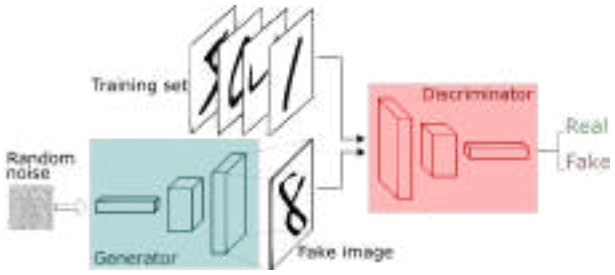


Figura 2.1: Esquema de funcionamiento de una GAN.

Si expresamos matemáticamente el proceso descrito tenemos que el generador G tiene una distribución p_g para los datos $\mathbf{x}$. A su vez, se tiene definida una distribución de probabilidad inicial (lo que en probabilidad Bayesiana se conoce como un *prior*) para las variables ruido (aleatorias) $p_z(z)$ que están mapeadas por $G(z;\theta_g)$, donde θ_g no es más que los pesos de la red G. Por otra parte, se define la red discriminadora $D(\mathbf{x};\theta_d)$ con un escalar como salida. $D(\mathbf{x})$ indica que si introducimos los datos nos devuelve la probabilidad de que esos datos sean reales o de que por el contrario, hayan sido producidos por p_g.

El entrenamiento consiste en mejorar D para maximizar la probabilidad de asignar correctamente las etiquetas de ambos casos de entrenamiento (los datos $\mathbf{x}$ y los generados por G). Simultáneamente, entrenamos G para minimizar $\log(1 - D(G(\mathbf{z})))$. Esto matemáticamente se puede resumir en la siguiente expresión [8] [14].

$$\min_G \max_D \mathcal{L}(D,G) = \mathbb{E}_{x,p_{\text{data}(x)}}[\log D(x)] + \mathbb{E}_{x,p_z(x)}[1 - D(G(z)))] \tag{2.1}$$

Las redes GAN son probablemente uno de los últimos grandes avances de las redes neuronales. Aún existen muchos problemas para entender cómo funcionan, y se han creado centenares de variantes basadas en la publicación original. La variante que aquí usamos es la denominada Transferencia entre dominios cruzados o *Cross-domain transfer* GAN.

Estas redes, cuyo funcionamiento explicaremos en el siguiente capítulo, han sido empleadas para resolver diferentes problemas, como colorear imágenes, transformar

animales, generar fachadas de edificios, dibujar bolsos o zapatos basándose simplemente en unos pocos trazos, o convertir imágenes a un cierto estilo de pintura, como es el caso particular de Van Gogh y Monet [5] [14].

Otras aplicaciones son la súper resolución, con la que se obtiene una imagen con mayor resolución, [4] y la súper cámara lenta, que permite generar imágenes sintéticas entre los fotogramas de un vídeo con el fin de poder reproducirlo a menor ritmo dando la sensación de tener cámara lenta [6].

Obtener la ropa que lleva puesta un modelo, envejecer la persona que hay en una fotografía, convertir texto en imágenes, ver la cara de una persona desde un ángulo diferente a la foto original, completar partes borradas de una imagen... En definitiva, existen decenas si no centenas de aplicaciones.

CAPÍTULO 3

Imagen a imagen (Pix2Pix)

> By far, the greatest danger of Artificial Intelligence is that people conclude too early that they understand it.
>
> —Eliezer Yudkowsky

Traslación imagen a imagen con redes condicionales antagónicas (*Image-to-Image Translation with Conditional Adversarial Networks*) es un artículo publicado el 21 de Noviembre de 2016 por Phillip Isola, Jun-Yan Zhu, Tinghui Zhou Alexei, A. Efros de la Universidad de Berkeley. En este se demostró que se podía crear redes que aprendan a mapear un conjunto de imágenes con otro conjunto y a su vez a aprender una función de pérdida.

En concreto, según el artículo, se pueden sintetizar fotos partiendo de mapas de etiquetas o máscaras, reconstruir objetos de mapas de vértices o colorear imágenes, entre otras tareas que ya se comentaron en el capítulo anterior.

La estructura de la red consiste en una red generadora basada en al arquitectura de la famosa red convolucional U-Net, ampliamente utilizada en reconocimiento de imagen médica, mientras que como red discriminadora se emplea un clasificador convolucional PatchGAN, el cual tiene la particularidad de penalizar solo trozos de la imagen y no toda la imagen.

Además, se han de tener una serie de aspectos en el diseño. Esto significa que mientras que las imágenes de entrada y de salida son diferentes, la estructura o esqueleto de la imagen ha de ser la misma. En este trabajo se diseñó un generador para poder cumplir con este requisito.

La red U-Net tiene un elemento codificador-decodificador que consiste en introducir los datos por una serie de capas que se hacen sucesivamente más pequeñas en número de neuronas, para luego invertir el proceso y pasar por sucesivas capas en las que se

aumenta el número de neuronas.

La idea es que la estructura o esqueleto de la imagen pase por el cuello de botella, pero no el resto de información. Es posible que de esta forma no pase toda la información necesaria para realizar la tarea que tenemos propuesta. Es por eso que se añaden saltos entre conexiones. En concreto se añaden entre cada capa i y la capa $n-i$ donde n es el número total de capas. Cada uno de estos saltos une cada canal RGB de las imágenes en la capa i con los de la respectiva capa $n-i$.

En cuanto al discriminador, previamente en otros estudios se determinó que las famosas funciones de pérdida. L2 y L1 producen resultados borrosos. En concreto, fallan con las altas frecuencias pero producen buenos resultados a bajas frecuencias. Es por eso que se trata de hacer que el discriminador modele sobre todo las altas frecuencias. Para lograr esto, se emplea la técnica de analizar la imagen a partes (*patches*). Este discriminador al que han denominado PatchGAN, trata de clasificar si cada N $\times$ N imagen es real o falsa.

Es importante además saber que para N muy pequeñas el discriminador muestra muy buenos resultados. Esto es de gran importancia, pues cuanto más pequeñas son las imágenes a comparar, menos parámetros, y por lo tanto menos memoria, ha de usarse. De esta manera el tiempo de entrenamiento se reduce considerablemente.

3.1. Funciones objetivo

Como ya se describió con anterioridad, en las GAN se busca que partiendo de un vector de ruido aleatorio z se obtenga una imagen de salida y, $G : z \rightarrow y$ y. Sin embargo con las cGAN se busca obtener un resultado con una imagen x y el vector de ruido aleatorio z, esto es $y, G : x, z \rightarrow y$ [2]. Lógicamente esto da una expresión ligeramente diferente para la función objetivo, que se expresa de la siguiente forma:

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{x,y}[logD(x, y)] + \mathbb{E}_{x,z}[log(1 - D(x, G(x, z)))] \tag{3.1}$$

Esta función objetivo es la que la red generadora trata de minimizar. Por el contrario, la red discriminadora trata de maximizarla. Por otra parte, como ya se mencionó, la distancias L1 y L2 dan buenos resultados para bajas frecuencias. Es por eso que la función objetivo del generador, pero no del discriminador, tratará de minimizar la distancia L1 definida como

$$\mathcal{L}_{\text{L1}}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1] \tag{3.2}$$

dando una función objetivo final

$$G^{\star} = \arg\min_G \max_D \mathcal{L}_{\text{cGAN}}(G, D) + \lambda \mathcal{L}_{\text{L1}}(G) \tag{3.3}$$

3.2. Generando el ruido inicial empleando *dropout*

El ruido inicial que se emplea en las redes GAN es sujeto de disputa en la actualidad. Normalmente se genera un ruido Gaussiano como semilla inicial que emplea la

red para obtener una imagen de salida. El problema es que en el entrenamiento el generador aprende a ignorar este ruido y a basarse simplemente en la imagen de entrada. Es por eso que el ruido o aleatoriedad se introduce empleando otro método. En cada época del entrenamiento, de forma aleatoria, ciertas neuronas de la red son desactivadas temporalmente. Cuantas más neuronas se desactiven por época, mas aleatoriedad introducimos en el sistema. Esta técnica se denomina en inglés *dropout*. El *dropout* tiene además otros aspectos positivos, pues evita el sobreajuste de los pesos de la red, ya que en todo momento han de saber adaptarse a cambios aleatorios. Otro aspecto a tener en cuenta es el hecho de que al tener menos pesos que ajustar, el entrenamiento se realiza más rápido.

Uno puede pensar que esto implica que cada vez que se ejecuta la red el resultado puede variar sustancialmente, pero este no es el caso. Generalmente las diferencias entre distintos entrenamientos es mínima y solo apreciable en detalles pequeños. Los aspectos principales de la red, como son estructura e información, permanecen, fuertemente invariables salvo pequeñas excepciones.

3.3. Optimización

El entrenamiento funciona de la siguiente forma: comenzando con la red generadora se realiza la actualización por gradiente y a continuación se repite el mismo proceso con la red discriminadora. Una de las principales diferencias entre las dos es que el objetivo de la red discriminadora es dividido por dos para reducir el ritmo al que aprende frente a la generadora.

3.3.1. Descenso estocástico por gradiente en mini lotes *Mini-batch SGD* y Adam

El método del descenso estocástico por gradiente en mini lotes es el empleado para encontrar la dirección por la que se quiere descender en la función de coste. Cuando se usan mini lotes (*minibatches*), implica que el conjunto de entrenamiento es troceado en lotes pequeños. Cada uno de esos lotes son empleados para calcular la función de coste. Una vez tenemos los resultados de todas las funciones de coste y las actualizaciones de peso, la media (o la suma) de cada uno de los resultados es lo que se usará para actualizar los pesos del modelo.

Adam, que es un acrónimo del inglés (*Adaptive Moment Estimation*), es una variante de *SGD* en el que se emplean los resultados de actualizaciones de pesos anteriores creando una media móvil, dando así más importancia a los cambios más recientes. Es considerado hoy en día de los mejores si no el mejor optimizador.

3.4. Evaluación del conjunto de test

Cuando se tiene que evaluar el conjunto de test, la red generadora se emplea de la misma forma que se empleó en el momento de entrenamiento. Esto implica que durante el test se aplica *dropout*. El conjunto de test se normaliza en lotes *batch normalization* empleando la estadística de los datos de test. Cuando el tamaño del lote se define como

uno se denomina normalización de la instancia o *instance normalization*, y es conocida por dar buenos resultados en tareas de generación de datos.

CAPÍTULO 4

Implementación

> On two occasions, I have been asked [by members of Parliament], 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able to rightly apprehend the kind of confusion of ideas that could provoke such a question.
>
> — Charles Babbage

La colección de imágenes empleada y el software empleado son los aspectos que vamos a discutir en este capítulo. La imágenes empleadas pertenecen al satélite Landsat 8. Este tiene dos instrumentos diferentes para capturar la radiación electromagnética: el *Operational Land Imager* OLI y *Thermal Infrared Sensor* (TIRS). Este satélite es el fruto de una colaboración entre la NASA y la USGS (United States Geological Survey) para la observación de la Tierra. Las imágenes que captura son imágenes RGB, donde R corresponde a la banda roja (0.64 - 0.67 μm), G – banda verde (0.53 - 0.59 μm) y B – banda azul (0.45 - 0.51 μm). Esta combinación de bandas es óptima para la visualización de cultivos o cosechas. Es por eso que todas las imágenes empleadas para este proyecto son de contenido rural. Por otra parte, las imágenes capturadas van acompañadas de información de nueve bandas del espectro (Tabla 4.1). La colección de imágenes puede obtenerse en https://landsat.usgs.gov/sparcs.

Tabla 4.1: Longitud de onda, resolución e irradiación solar de las distintas bandas de una imagen.

Banda	Longitud de onda (μm)	Resolución (m)	Irradiación solar ($W/m^2\mu m$)
Band 1 - Coastal / Aerosol	0.433 – 0.453	30	2031
Band 2 - Blue	0.450 – 0.515	30	1925
Band 3 - Green	0.525 – 0.600	30	1826
Band 4 - Red	0.630 – 0.680	30	1574
Band 5 - Near Infrared	0.845 – 0.885	30	955
Band 6 - Short Infrared	1.560 – 1.660	30	242
Band 7 - Short Infrared	2.100 – 2.300	30	82.5
Band 8 - Panchromatic	0.500 – 0.680	15	1739
Band 9 - Cirrus	1.360 – 1.390	30	361

Gracias a estas bandas extra podemos apreciar en las máscaras información que no podemos ver en el espectro visual. Como es el caso de ciertas nubes. A lo largo de los resultados podremos apreciar casos en los que las máscaras dan la información de una nube que a simple vista no es posible detectar.

Los sensores de ambas cámaras pueden captar hasta 4086 niveles de grises. Las imágenes empleadas son de 16-bit escaladas a una escala de grises de 55000 niveles. La interpretación de los valores en cada máscara se muestra en el tabla 4.2.

Tabla 4.2: Significado de los colores para las máscaras

Valor	Interpretación	Color
0	Sombra	Negro
1	Sombra sobre agua	Azul oscuro
2	Agua	Azul
3	Nieve	Cian
4	Tierra	Gris
5	Nube	Blanco
6	Inundado	Marrón claro

Tenemos siete clases. Evidentemente, cuantas más clases los problemas de redes se vuelven complicados de entrenar, pues generalmente no se tienen suficientes casos de cada clase, por lo que el caso que tenemos puede resultar complicado tener buenos resultados en algunas de las clases.

Toda la implementación está realizada con Python 3.5. La principal librería empleada para la construcción de las redes neuronales es PyTorch 3.0 y 4.0. Dicho código se puede obtener en en las referencias [5] y [14]. Los entrenamientos han sido realizados empleando la versión basada en PyTorch 3.0. La evaluación de las imágenes de test con la versión basada en PyTorch 4.0. Se han utilizado estas dos versiones debido a que, en el momento de evaluar las imágenes de test, un error en el código que se encargaba de las transformaciones internas que realiza la red manipulaba de forma sutil pero inconveniente las imágenes de salida. Una nueva versión de la implementación basada en

PyTorch 4.0 salió arreglando este y otros problemas. El entrenamiento de la red se ha realizado empleando la tarjeta gráfica proporcionada por el *Intelligent Data Analysis Laboratory*, al cual quiero agradecer haberme permitido su uso. La tarjeta gráfica es una Tesla M10 compuesta de cuatro gráficas individuales de 8 GB de memoria cada una.
El entrenamiento se tiene que realizar para cada caso, esto es uno para el sentido fotografía a máscara y otro para máscara a fotografía.

CAPÍTULO 5

Entrenamiento y testeo

"A little Learning is a dangerous Thing."

- Alexander Pope

Las imágenes de las que se dispone para entrenar, validar y testear tienen originalmente unas dimensiones de 1000x1000 píxeles. Como primer paso, estas imágenes son divididas en 16 trozos de 250x250 píxeles, tal como se muestra en la figura 5.1.

Como podemos ver, el conjunto de entrenamiento está formado por ocho imágenes contiguas, mientras que los de validación y test se componen de cuatro. No obstante, no nos limitaremos a testear solo con cuatro imágenes, sino que generaremos imágenes adicionales realizando sobre ellas ciertas transformaciones que explicaremos a continuación.

Para el trabajo tenemos que entrenar dos redes. La primera de las redes la denominaremos AtoB. Esto indica que vamos de imágenes de tipo A (fotografías) a tipo B (máscaras).

La segunda red es la denominada BtoA y es la inversa de AtoB, es decir, de imágenes de tipo B (máscaras) a imágenes de tipo A (fotografías). Ambas redes tardan aproximadamente lo mismo en ser entrenadas. La tasa de aprendizaje se mantiene constante las primeras 200 épocas, para luego disminuir linealmente hasta cero en las 400 épocas restantes.

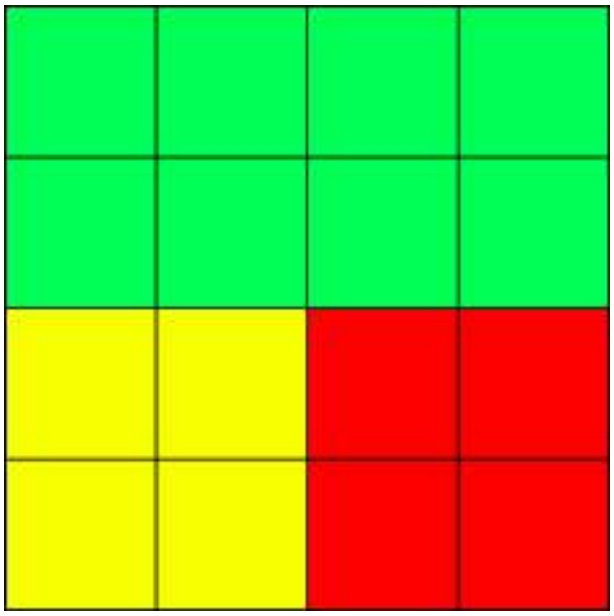


Figura 5.1: Reparto de una imagen 1000x1000 en 16 trozos de 250x250. El verde corresponde a datos para el entrenamiento, el amarillo como validación y el rojo como test.

5.1. Rotación de imágenes de test

La primera de las transformaciones consiste en rotar cada imagen con tres ángulos diferentes (90, 180 y 270) generándose las tres imágenes que se muestran en la figura 5.2.

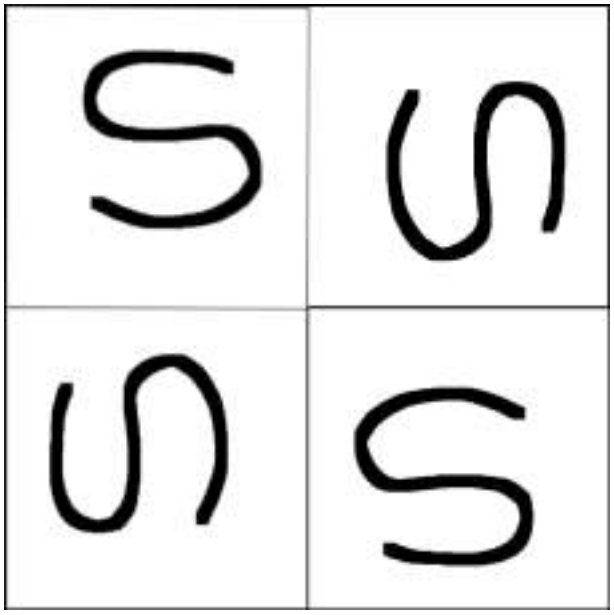


Figura 5.2: Ejemplo de rotación para cada una de las imágenes de test.

5.2. Mezcla de las imágenes de test

Esta segunda transformación es más complicada de explicar, pero es probablemente la más interesante. Partiendo de las imágenes de la región de test, representada en color rojo en la figura 5.1, crearemos imágenes nuevas. Para ello, en primer lugar dividiremos cada una en cuatro secciones. De esta forma pasaremos de una cuadrícula de 2x2 (figura 5.3) a una de 4x4 (figura 5.4).

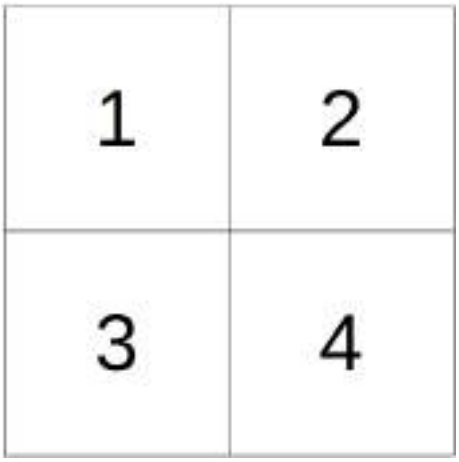


Figura 5.3: Representación de las cuatro imágenes de test divididas a su vez en cuatro secciones.

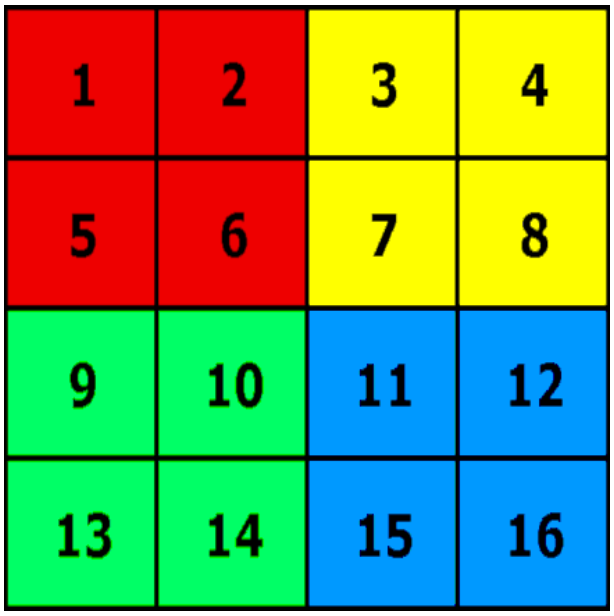


Figura 5.4: Representación de las cuatro imágenes de test divididas a su vez en cuatro secciones.

A continuación, obtendremos cuatro nuevas imágenes como la combinación de dos mitades contiguas. Si observamos la figura 5.4, las nuevas imágenes estarían compuestas por los siguientes recuadros:

- En la parte superior, los recuadros 2,6,3,7 (combinación de la imagen roja y la amarilla).
- A la izquierda, los recuadros 5,6,9,10 (combinación de la imagen roja y la verde).
- A la derecha, los recuadros 7,8,11,12 (combinación de la imagen amarilla y la azul).
- En la parte inferior, los recuadros 10,14,11,15 (combinación de la imagen verde y la azul).

En la figura 5.5 se han dispuesto las cuatro imágenes de test iniciales junto con las imágenes resultado de su combinación.

Figura 5.5: Representación de las cuatro imágenes de test junto con las nuevas imágenes obtenidas al combinar sus distintas mitades.

Para un conjunto de cuatro imágenes de test reales, el resultado final siguiendo este procedimiento puede observarse en la figura 5.6.

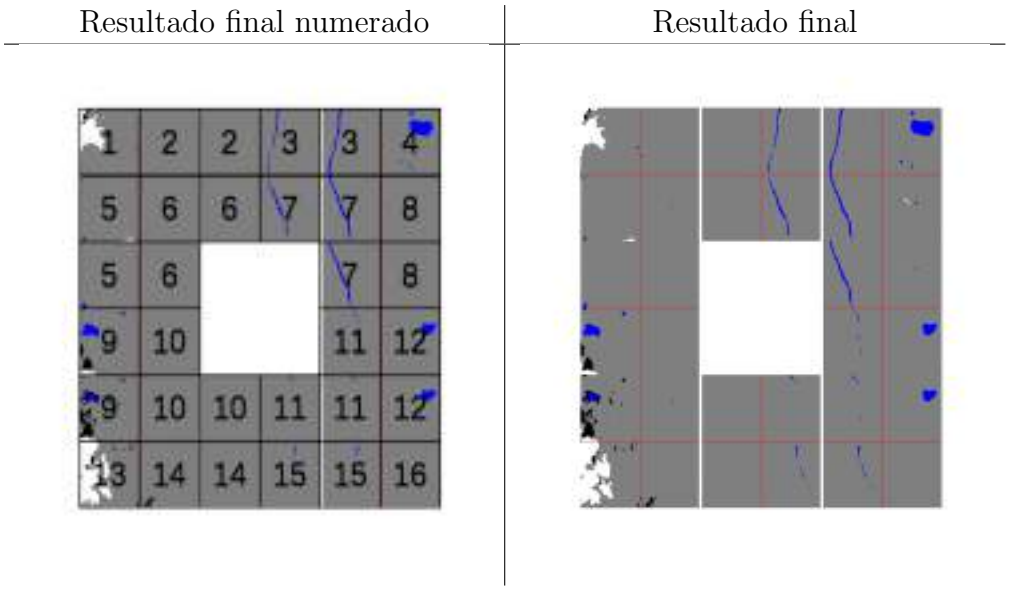


Figura 5.6: Resultado final para una imagen real, numerado (izquierda) y sin numerar (derecha).

CAPÍTULO 6

Resultados

> "Before we work on artificial intelligence why don't we do something about natural stupidity?"
>
> —Steve Polyak

A continuación vamos a hablar de los resultados obtenidos por la red. Han sido en general satisfactorios. No sin errores, las imágenes producidas dan una buena sensación. Evaluar la calidad de las fotos queda como una tarea subjetiva del lector pues no existe una forma cuantitativa clara de evaluar el resultado. Por otra parte las máscaras sí pueden ser evaluadas de una forma cuantitativa con los métodos que se introdujeron en la teoría.

Los resultados se presentarán en figuras que pueden contener algunas de las siguientes columnas.

- Fotografía real: se trata de la fotografía original, que forma parte del conjunto de test.
- Máscara real: se trata de la máscara original, que forma parte del conjunto de test.
- Fotografía generada: se trata de la fotografía generada al introducir una máscara real en la red.
- Máscara generada: se trata de la máscara generada al introducir una fotografía real en la red.

En muchas ocasiones, y gracias a que la red ha realizado un buen trabajo, tanto las versiones reales como generadas serán muy similares. Nótese que en las zonas donde existen diferencias entre la máscara real y la generada, la clasificación a simple vista a partir de la foto real es confusa.

6.1. Gráficas de entrenamiento de las dos redes

Las métricas de entrenamiento de AtoB y BtoA se representan en las figuras 6.1 y figura 6.2 respectivamente.

Con la etiqueta *G GAN* tenemos la función objetivo correspondiente a la red generadora. Como se ve en la expresión 3.3, en la función objetivo se busca maximizar G. A su vez, G trata de minimizar la función de pérdida L1, que podemos ver en las figuras con la etiqueta *G L1*. Por el contrario, la red discriminadora siempre se busca minimizar las funciones objetivo. Se muestra la función objetivo para las imágenes reales que no han pasado por el generador (*D real*) y para las imágenes generadas por el discriminador (*D fake*).

El comportamiento típico de un correcto entrenamiento corresponde a un crecimiento de *G GAN* y una disminución de *G L1*, *D real* y *D fake*.

Este comportamiento ha sido observado en el entrenamiento de ambas redes. Cabe señalar que para el caso de la red BtoA parece alcanzar un estancamiento en la fase final para el caso de la red BtoA, pues ninguna de las gráficas parece mejorar en esta última etapa.

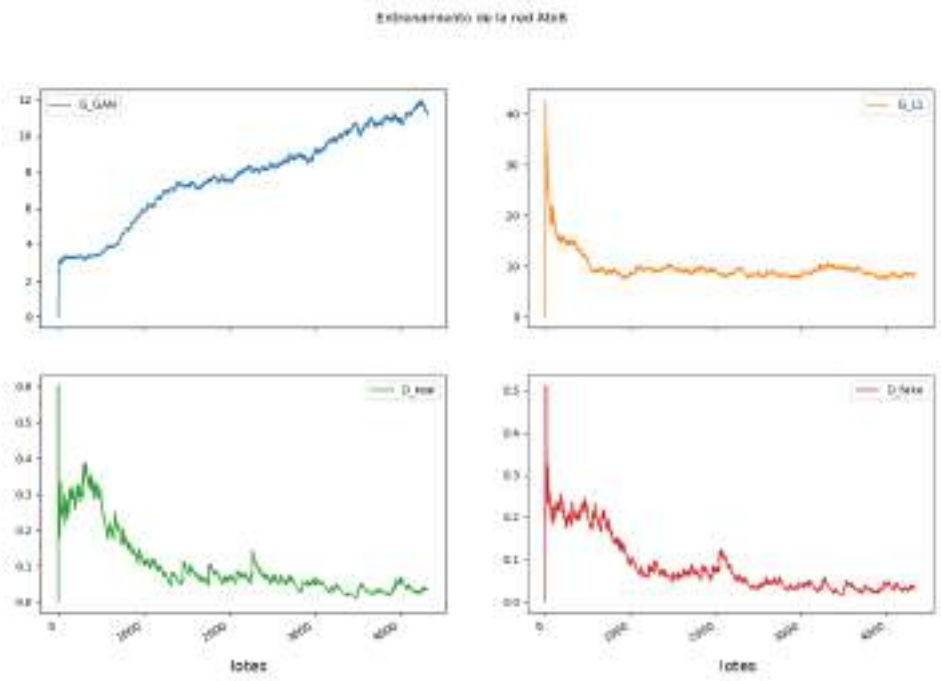


Figura 6.1: Entrenamiento AtoB, fotografía a máscara.

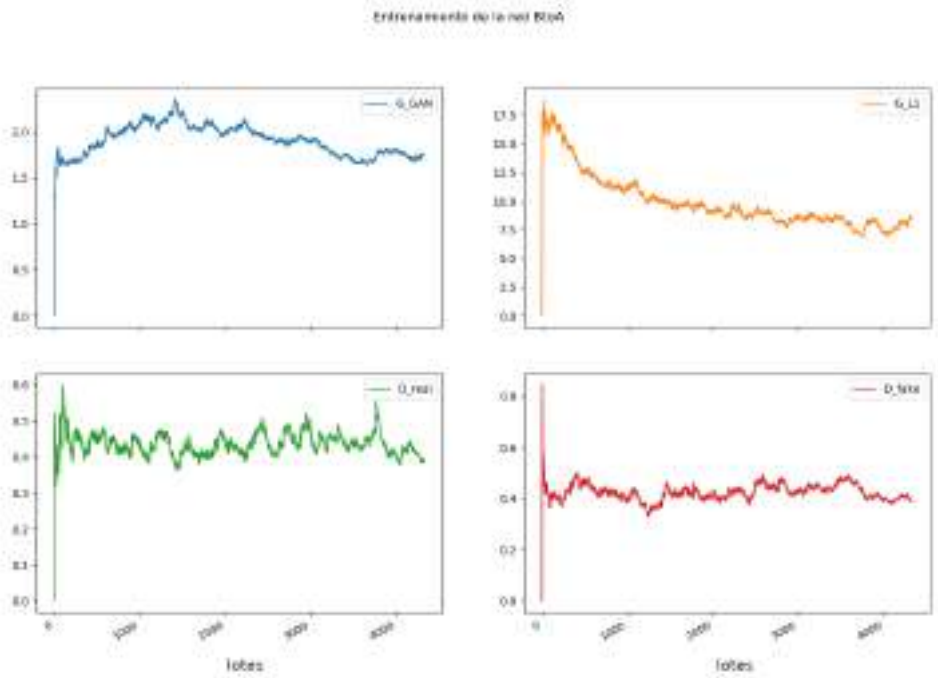


Figura 6.2: Entrenamiento BtoA, máscara a fotografía.

6.2. Generación de máscaras

En esta sección vamos a mostrar máscaras de clasificación generadas por la red. Veremos ejemplos en los que la red realiza un buen trabajo y otros en los que este no es tan satisfactorio.

Comenzamos con un ejemplo de nubes con sombra sobre un prado verde (figura 6.3) y sobre prado marrón (figura 6.4).

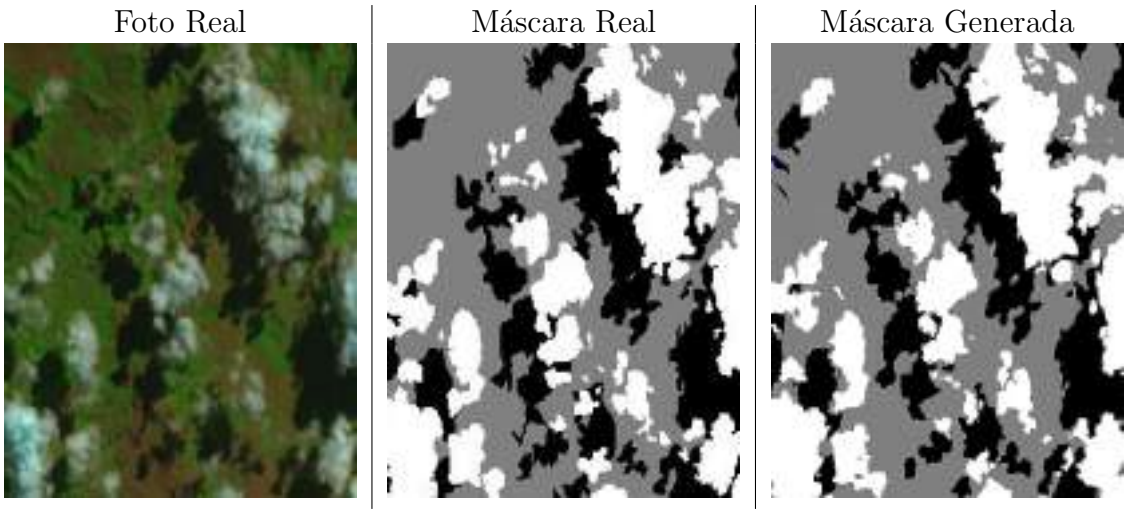


Figura 6.3: Generación de nubes con sombra sobre pradera verde.

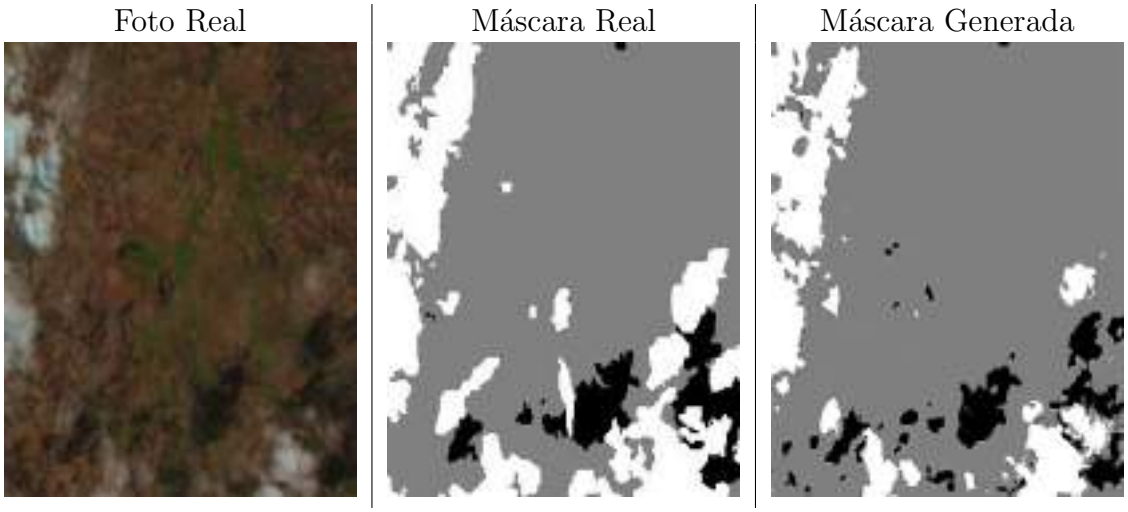


Figura 6.4: Generación de nubes con sombra sobre pradera de tono marrón.

Como podemos observar, el resultado es francamente bueno. La red es capaz de segmentar correctamente las tres clases (pradera, sombras y nubes) y de generar la máscara.

A continuación añadimos una nueva clase a segmentar: el agua. La figura 6.5 muestra la generación de nubes con sombra sobre pradera y agua.

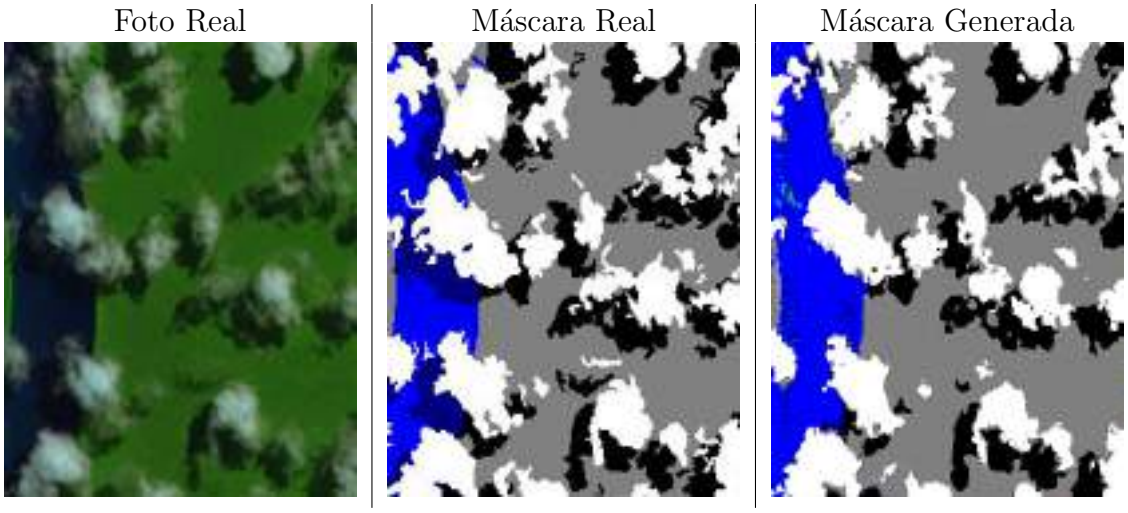


Figura 6.5: Generación de nubes sobre pradera y agua.

Podemos observar que en general, tanto las nubes como las sombras, no son tan precisas en detalle pero sí en estructura. El agua también es correctamente generada, pero se aprecian fallos en la sombra sobre esta.

Cuando tenemos ríos de menor anchura o aislados, el resultado proporcionado por la red no es tan preciso. Por ejemplo podemos ver que en la figura 6.6 que el río que se presenta en la zona inferior izquierda es completamente ignorado.

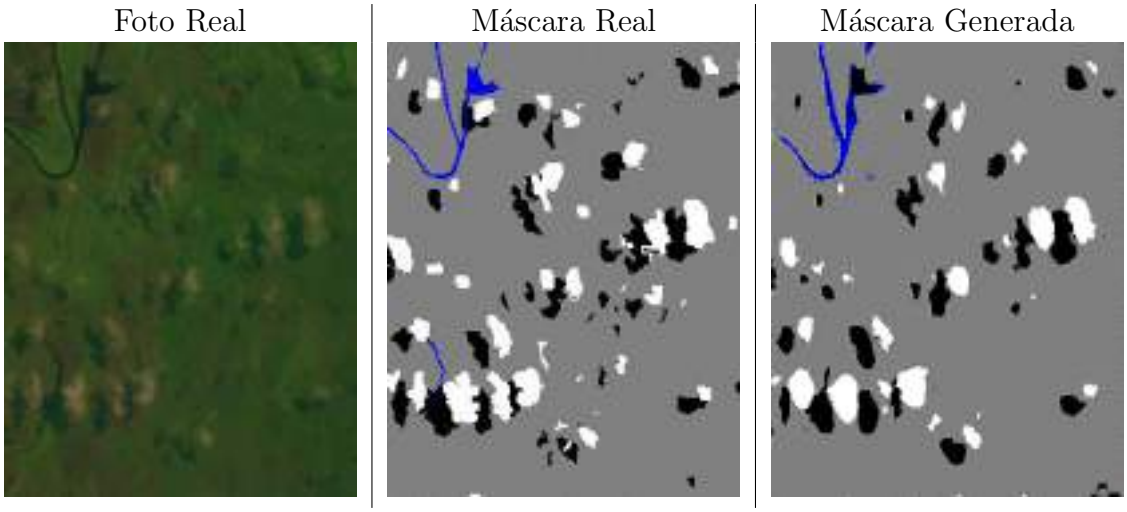


Figura 6.6: Generación de nubes y agua sobre pradera verde.

En la figura 6.7 podemos ver la generación de pradera verde cuando discurren ríos formando un entramado complejo. Los ríos principales son correctamente identificados, pero podemos observar que cuando el río es más estrecho, la red los comienza a ignorar.

Además, la red tiene cierta tendencia a confundir acumulaciones de agua con sombras, pues estas al ser más profundas obtienen una tonalidad más oscura. Por otra parte, podemos fijarnos que cuando hay un detalle *blanquecino* en la imagen, esta es incorrectamente identificada como nube.

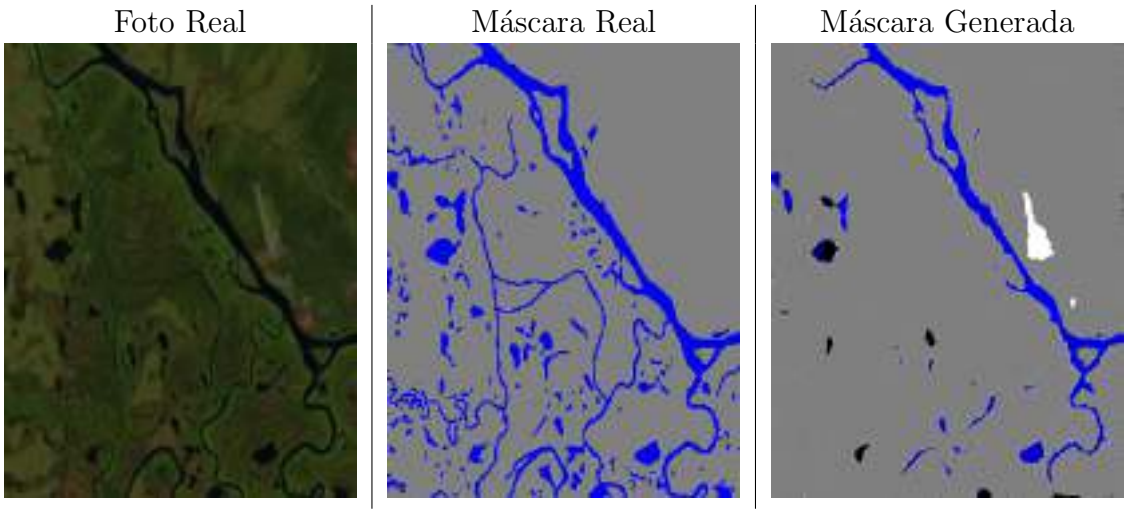


Figura 6.7: Generación de pradera verde con un entramado de ríos complejo.

En el ejemplo mostrado en la figura 6.8 podemos ver que, a pesar de que la fotografía original contiene diferentes cultivos con claras líneas divisorias, lo identifica correctamente como la misma clase.

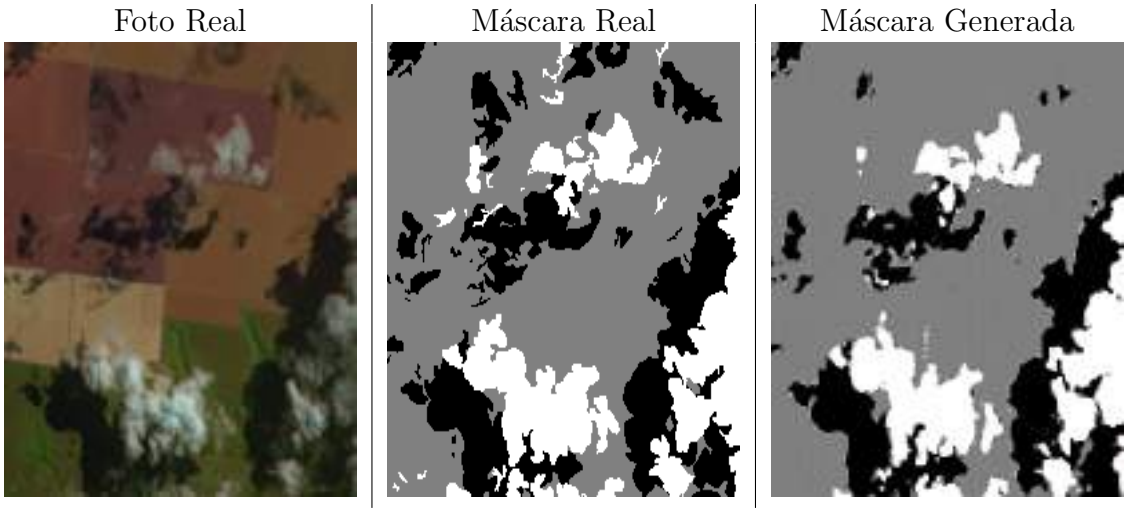


Figura 6.8: Generación de pradera con diferentes cultivos y nubes.

Como es de esperar, los tonos oscuros debido a campos inundados son confundidos con sombras (figura 6.9). Es este un motivo por el cual la métrica Sørensen–Dice es más baja como veremos en la siguiente sección de lo que inicialmente se puede pensar.

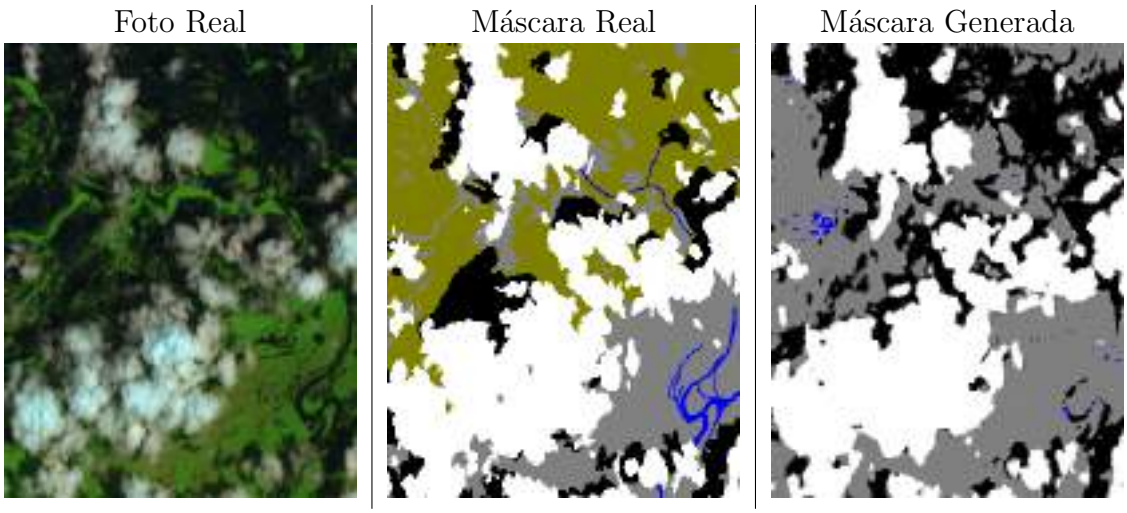


Figura 6.9: Generación de pradera con río y una inundación.

Por último, en la figura 6.10 se muestra uno de los pocos casos en los que la nieve es razonablemente bien reconocida. No obstante fracasa completamente en identificar la gran sombra que hay en la esquina superior derecha. Es importante comentar que incluso un humano tendría dudas de si se trata de sombra o de una acumulación de agua, o incluso de sombra sobre agua. Por tanto este es un fallo perfectamente razonable.

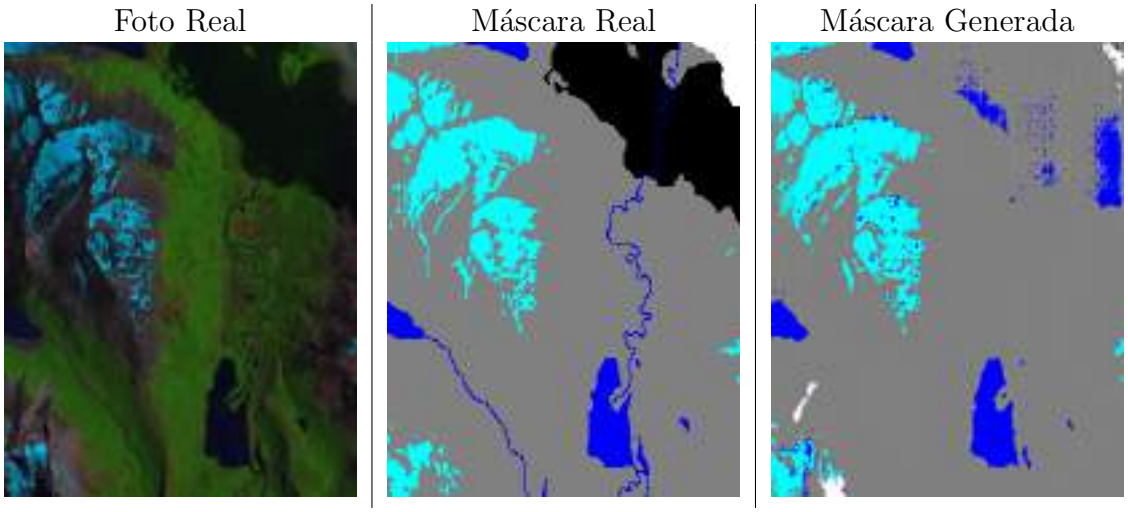


Figura 6.10: Generación de un monte nevado con un lago.

6.2.1. Puntuación de las máscaras

La figura 6.11 muestra la puntuación por la métrica Sørensen–Dice. Esta puntuación se ha calculado para todas las imágenes de test y ha sido separada por clases.

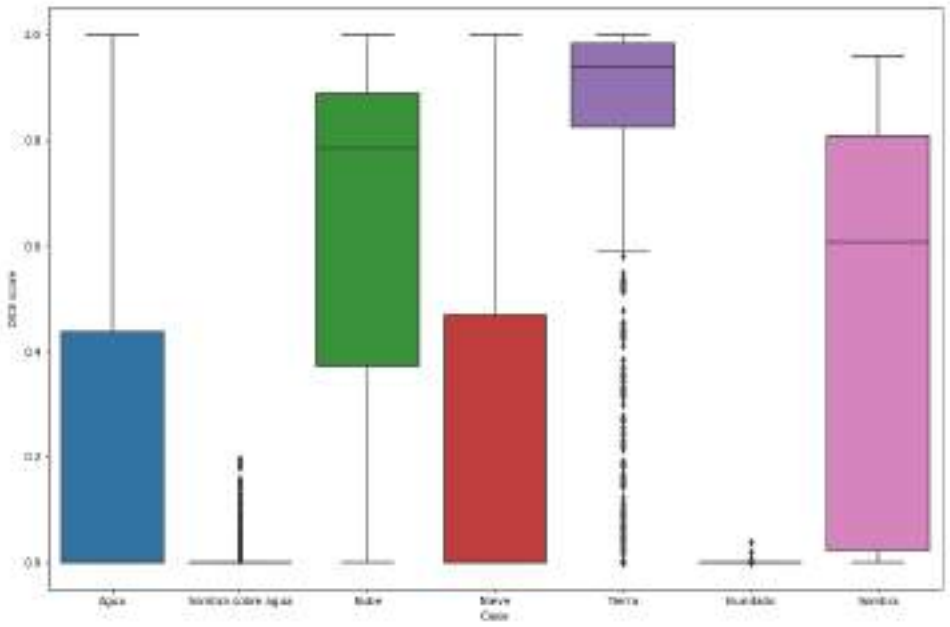


Figura 6.11: Diagrama de caja del coeficiente Sørensen–Dice obtenido en las imágenes de test para cada clase.

No hay que olvidar que esta métrica es una comparación píxel a píxel, y por lo tanto, muy estricta. En general las puntuaciones son altas, pero debido a que la red

tiene problemas con escenarios nevados, en muchas ocasiones la puntuación de las nubes se ve afectada, pues crea falsos positivos.

6.3. Generación de fotos

En esta sección vamos a examinar visualmente la generación de imágenes reales a partir de máscaras de clasificación. Para empezar, comenzaremos con un caso sencillo, una pradera con unos pocos puntos de agua (figura 6.12).

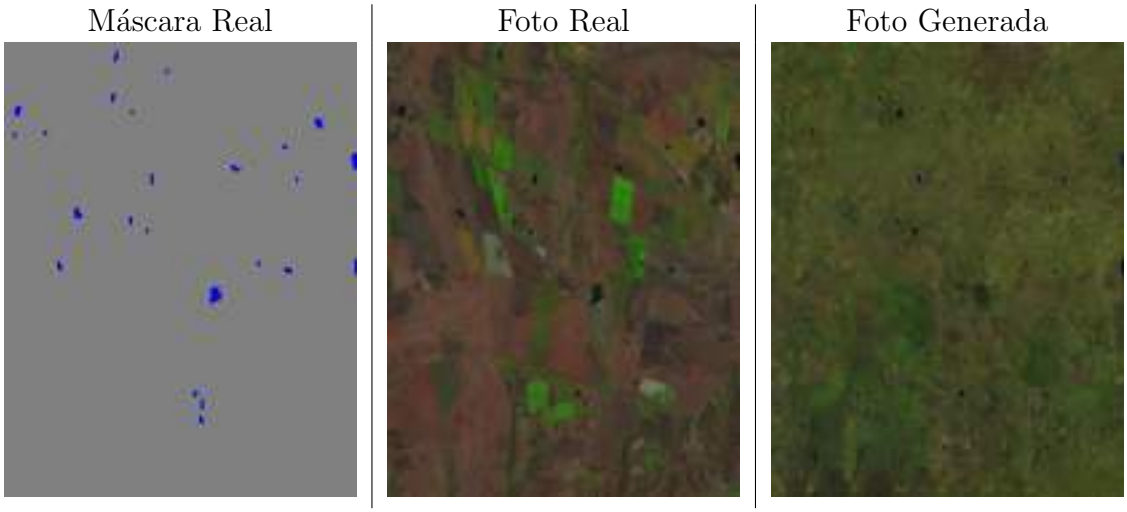


Figura 6.12: Generación de pradera verde con pequeños lagos.

Es evidente que la red no genera de forma arbitraria los campos de cultivos. A continuación (figura 6.13), podemos ver uno de los mejores resultados que ha proporcionado la red. Contiene prácticamente toda la información de la máscara creando una imagen convincente.

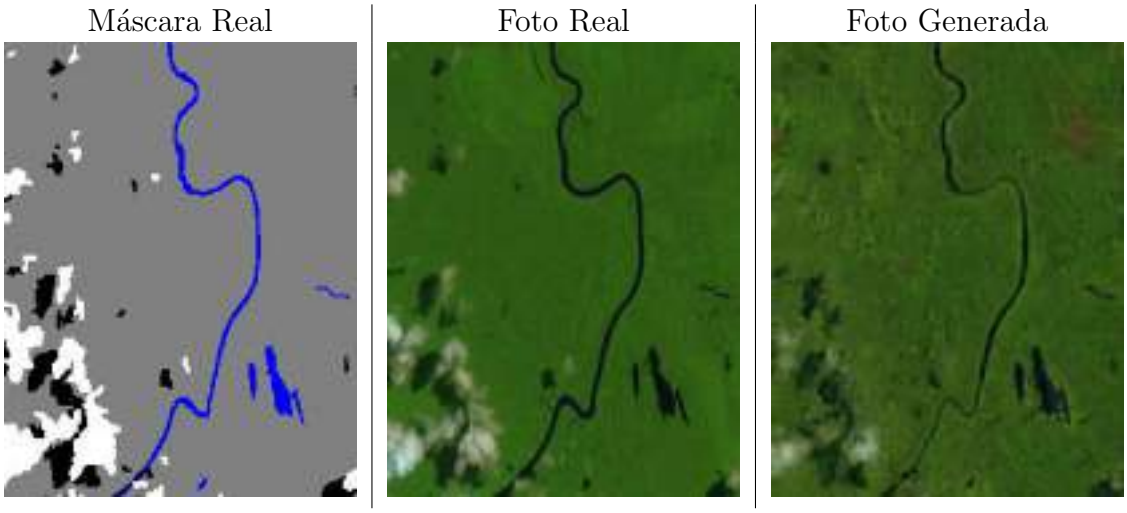


Figura 6.13: Generación de río por una pradera.

Si tratamos de generar una imagen partiendo de una máscara con información de

región inundada obtenemos un resultado razonable (figura 6.14). Los ríos son distinguibles cuanto a penas pero se pueden apreciar. En general ha realizado un mejor trabajo partiendo de una máscara de inundación que en el sentido inverso, como se ha podido apreciar en las muestras de resultados de los casos AtoB.

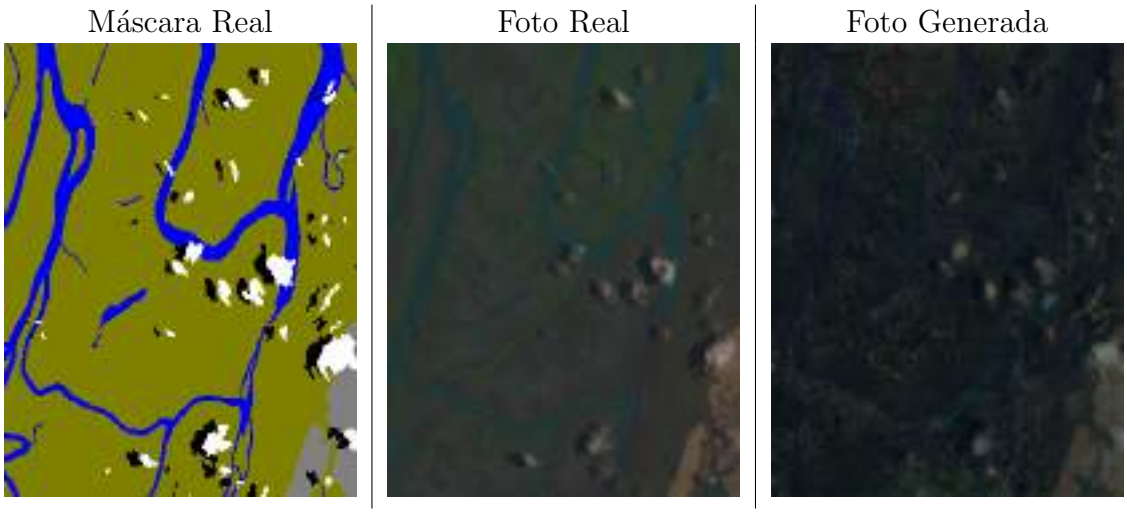


Figura 6.14: Generación de río por región inundada.

Como podemos observar en la figura 6.15. La red no tiene muchos problemas con estructuras de nubes complejas. Mantiene la coherencia geométrica de las nubes. La principal diferencia es que la pradera ha sido generada con tonos verdes, lo cual es correcto.

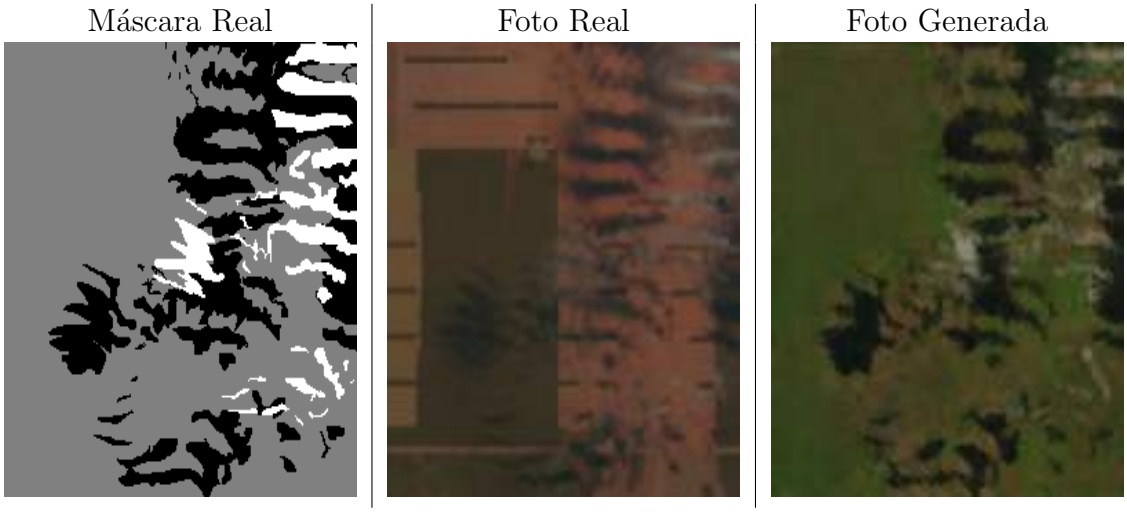


Figura 6.15: Generación de nubes con detalle.

Respecto a los casos con nieve (figura 6.16) si comparamos la fotografía real con la generada, podemos decir que no realiza un mal trabajo. Si nos fijamos, la red ha generado una zona montañosa. Esto es probablemente debido a que generalmente cuando tenemos una zona nevada esta va acompañada de montañas, y la red ha aprendido esta conexión entre ambas.

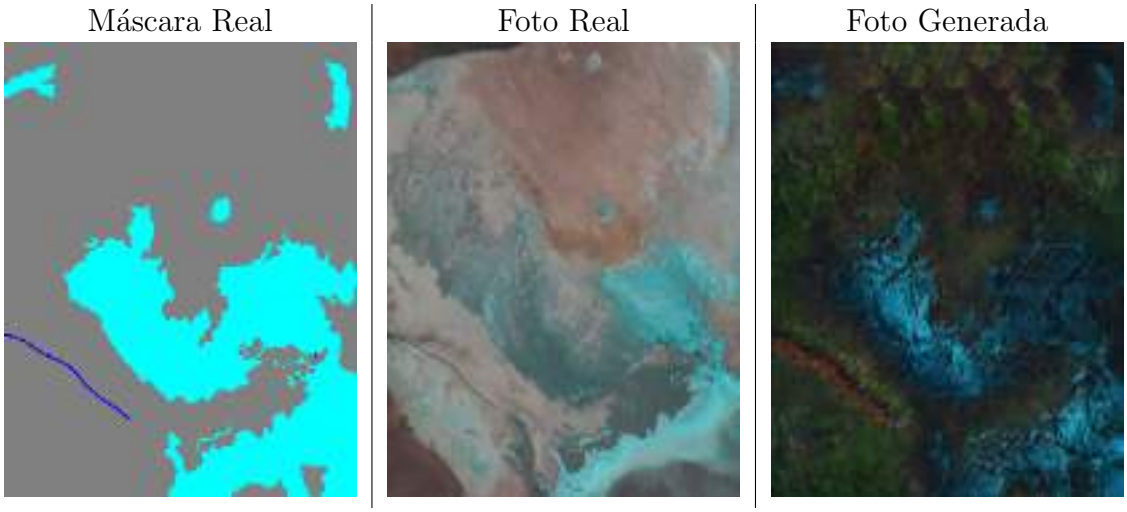


Figura 6.16: Generación de pradera con nieve.

En la figura 6.17 tenemos una composición compleja de clases. La red en este caso genera un resultado muy satisfactorio, siendo posiblemente uno de los resultados más impresionantes.

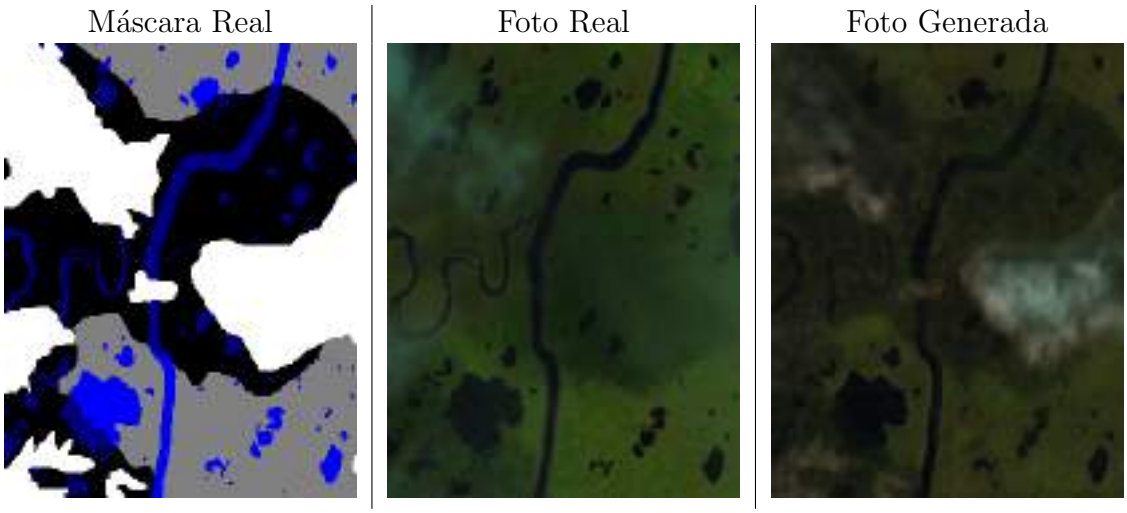


Figura 6.17: Generación de pradera con río y sombras.

6.4. Rotación de imágenes

A continuación vamos a mostrar cómo se comporta la red cuando rotamos las imágenes. Es decir, vamos a tomar una imagen y mediante rotaciones de 90º se generarán tres imágenes más. De esta forma tendremos una imagen sin rotar, una imagen rotada 90º, otra imagen 180º y por último una imagen rotada 270º.

El objetivo es comprobar si distintas orientaciones de los mismas imágenes nos dan resultados diferentes. Por último hay que comentar que todas las figuras que se muestran a continuación han sido rotadas a la orientación original con el de facilitar la comparación.

En general podemos apreciar diminutas diferencias en los colores y en la geometría, pero imagen general mantiene su similitud (figura 6.18. En ciertas regiones, la red decide para el caso BtoA, generar zonas de tierra más secas que en otras rotaciones.

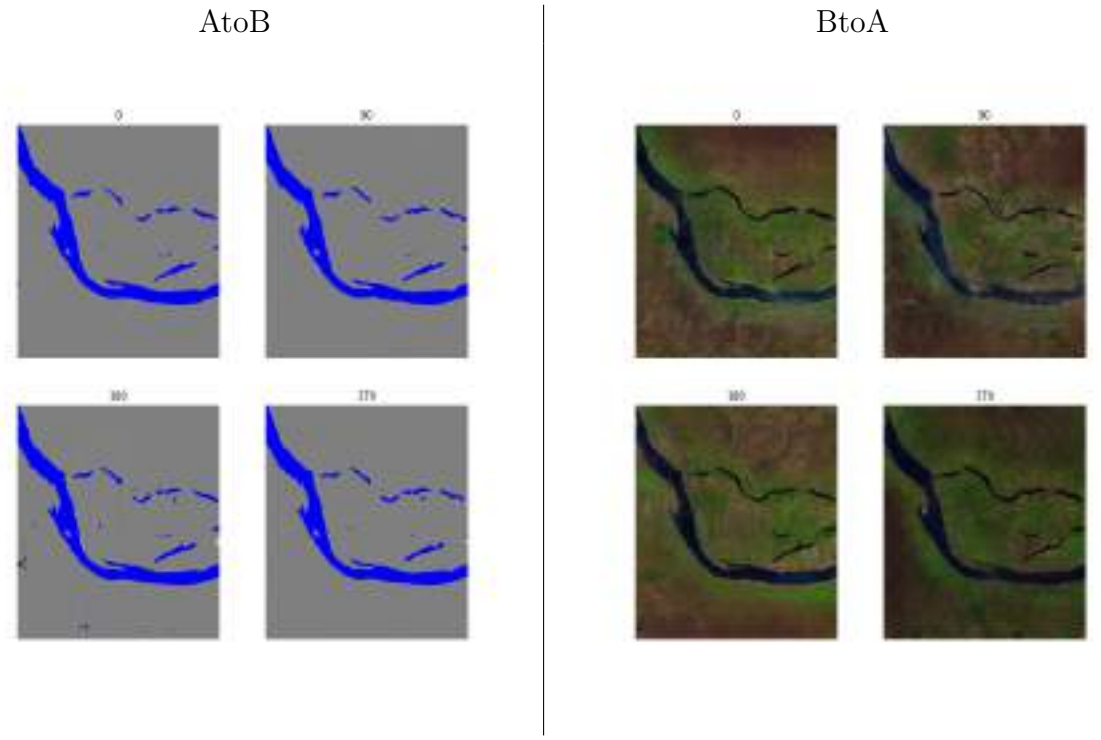


Figura 6.18: Generación de un río por una pradera.

En la figura 6.19, es interesante observar la gran diferencia que existe entre las imágenes desde 0º y 180º. Invertir la imagen parece tener un efecto notable en la calidad de la generación de la máscara.

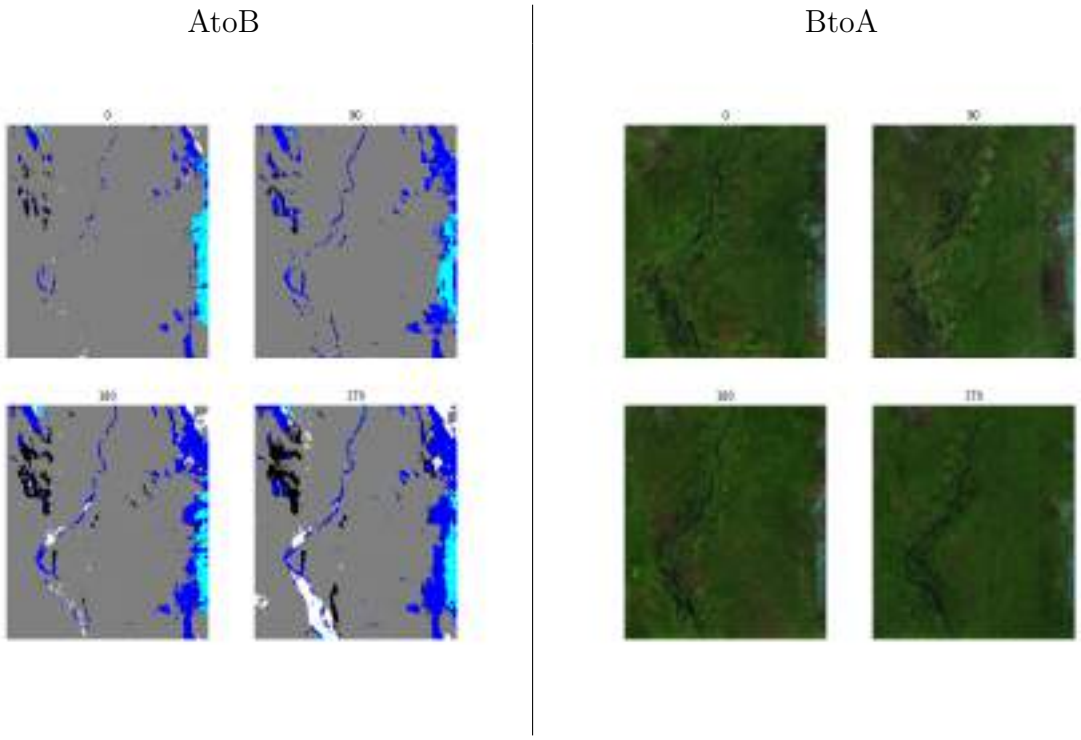


Figura 6.19: Generación de un río por una pradera.

En la figura 6.20 se puede apreciar el mismo fenómeno que se ha comentado anteriormente con la figura 6.19. En ambos casos la rotación afecta a la calidad de la máscara generada.

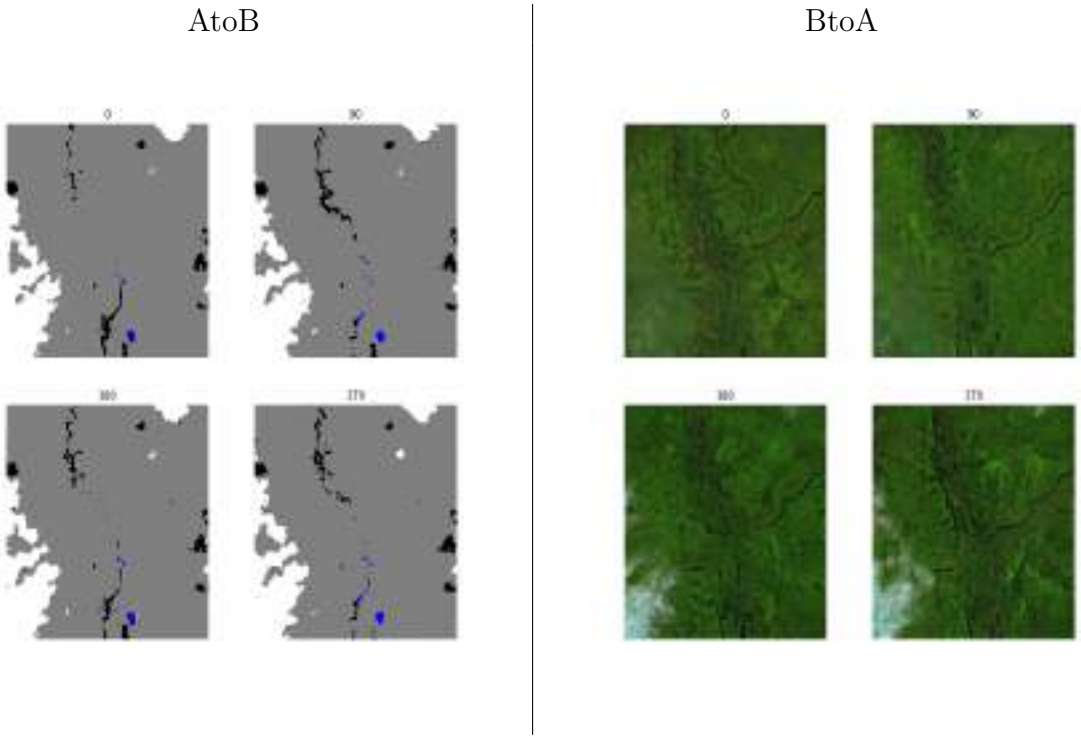


Figura 6.20: Generación de un río pequeño por una pradera y nubes.

En ocasiones la red puede generar patrones o artefactos en las imágenes. Esto se puede apreciar en la figura 6.21. En esta, se ve claramente como la creación de las nubes mantiene una estructura que está alineada con la rotación de las imágenes. Incluso en ocasiones (figura 6.22), la orientación correspondiente a 90º carece de los artefactos visuales que el resto si padece.

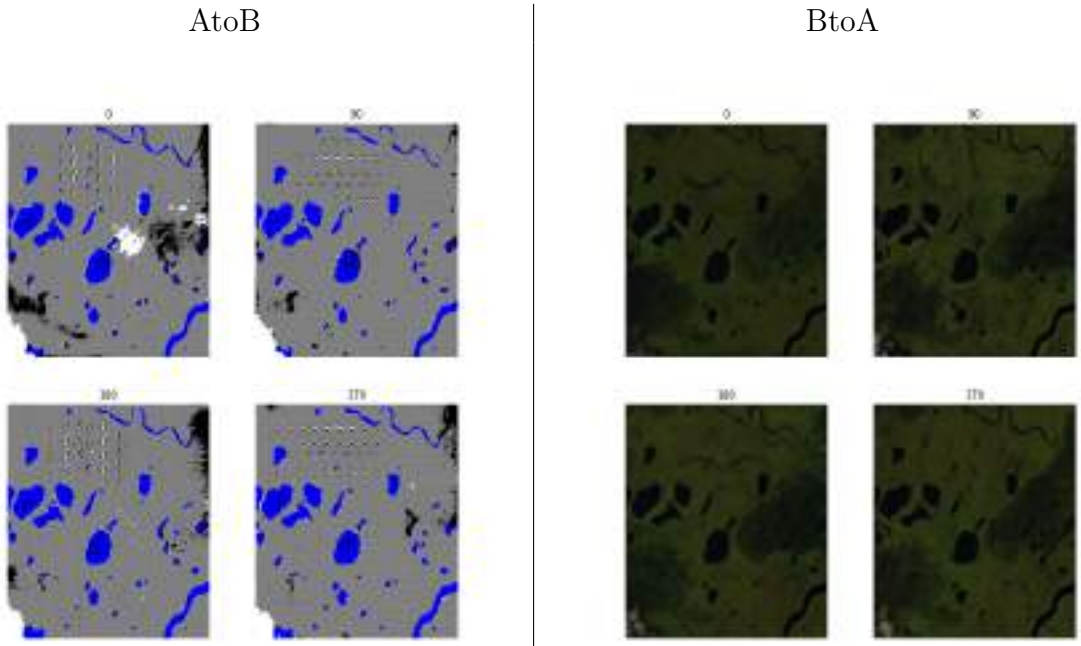


Figura 6.21: Generación pradera con agua con síntomas de patrón.

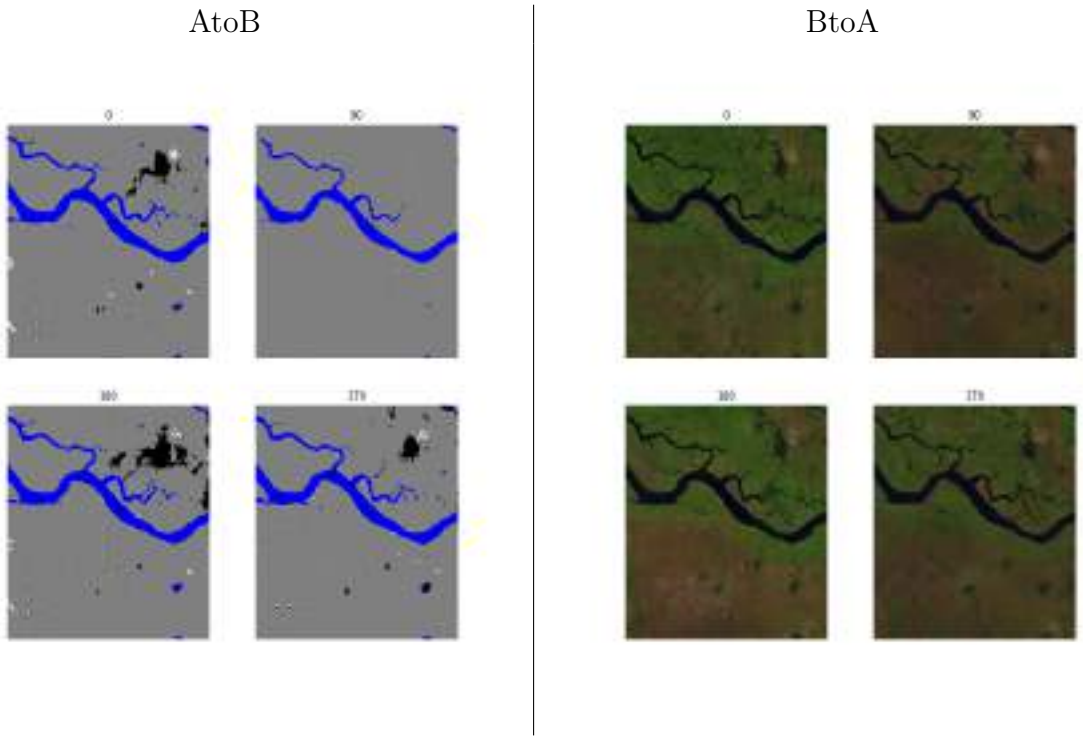


Figura 6.22: Generación pradera con un río, donde una imagen es claramente superior.

Existen más diferencias en la generación de resultados mediante este método. En la figura 6.23, una de las fotografías ha creado un resultado completamente diferente al resto de las imágenes rotadas. Este comportamiento no es erróneo, pero sí evidencia la importancia de la orientación, incluso para los casos más simples. Además, en la máscara cuya rotación corresponde a 270º, sí se aprecia un cuerpo de agua que en las otras máscaras no logra ser detectado.

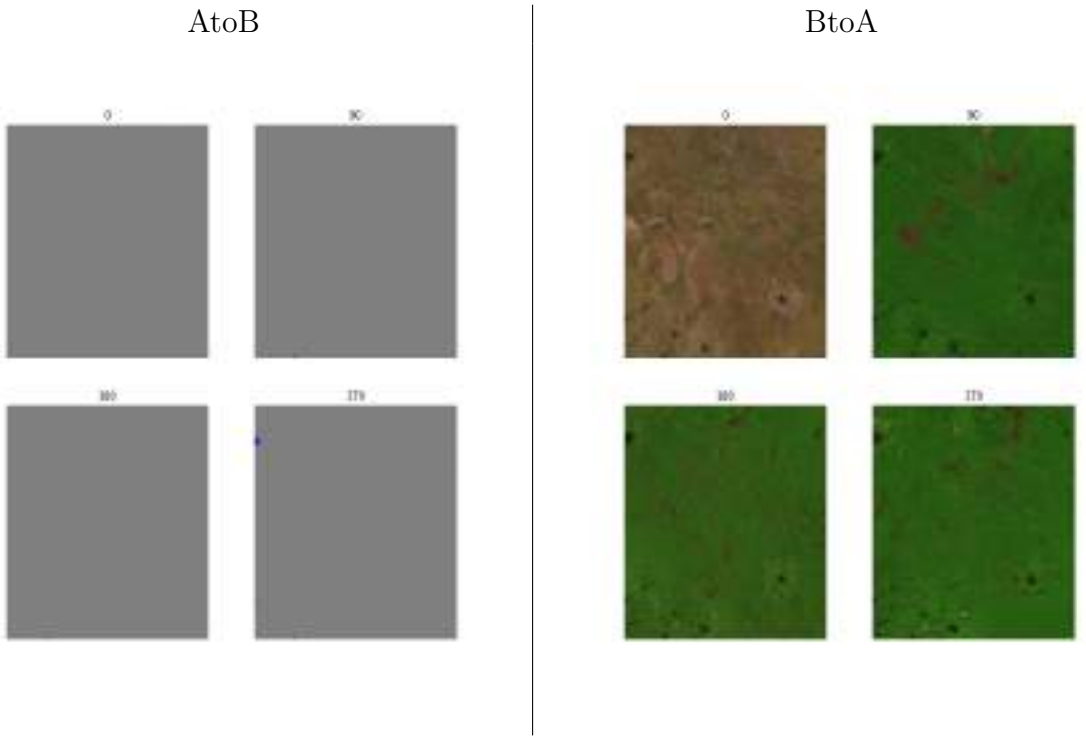


Figura 6.23: Generación pradera con un río, donde una imagen es claramente superior.

6.5. Mezcla de imágenes

Este apartado es probablemente la comprobación más metódica para exponer las diferencias que existen en la creación de imágenes. A continuación se van a mostrar imágenes que han sido generadas por la red, por lo que se podrán apreciar diferencias estructurales. Cada bloque contendrá ocho imágenes situadas en un cuadrado de 3x3, dejando el cuadrado central vacío. En los extremos del cuadrado tendremos cuatro imágenes que vienen del set original de test. Pero además de estas imágenes de test se han generado otras cuatro imágenes adicionales tal y como se explicó en capítulos anteriores.

En la figura 6.24 podemos apreciar que las imágenes mezcladas, solo presentan pequeñas diferencias frente a las originales.

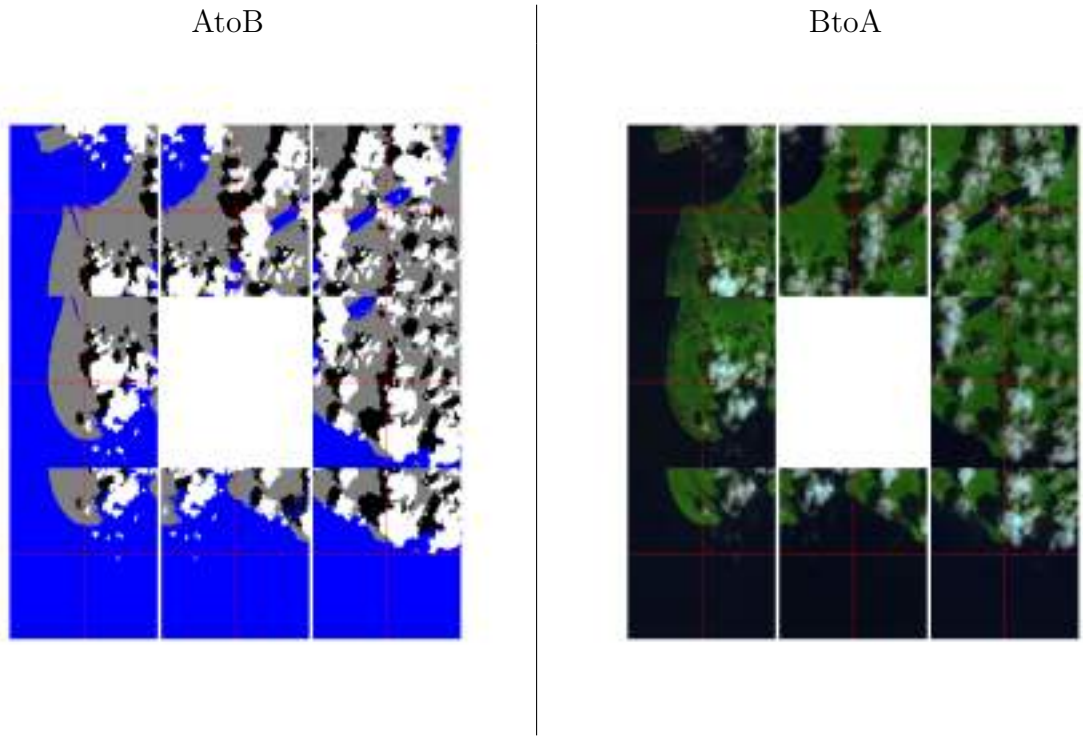


Figura 6.24: Generación de zona costera y nubes.

Para el siguiente caso (figura 6.25) sí podemos apreciar diferencias, como puede ser la anchura del río. Además las tonalidades de las fotografías cambian en algunos ejemplos. Esto no es de sorprender teniendo en cuenta que, como se ha visto en la sección anterior, un cambio de la rotación puede producir efectos similares. No obstante en la foto superior del caso BtoA se presenta un río acompañado de una pradera de tonos marrones, mientras que ese mismo río en la foto de la izquierda está rodeado de una pradera con tonos verdes, posiblemente porque a la derecha de este tenemos más cuerpos de agua. La red ha comprendido que cuanto mayor es la cantidad de agua, más tonos verdes usa en la clase pradera (gris).

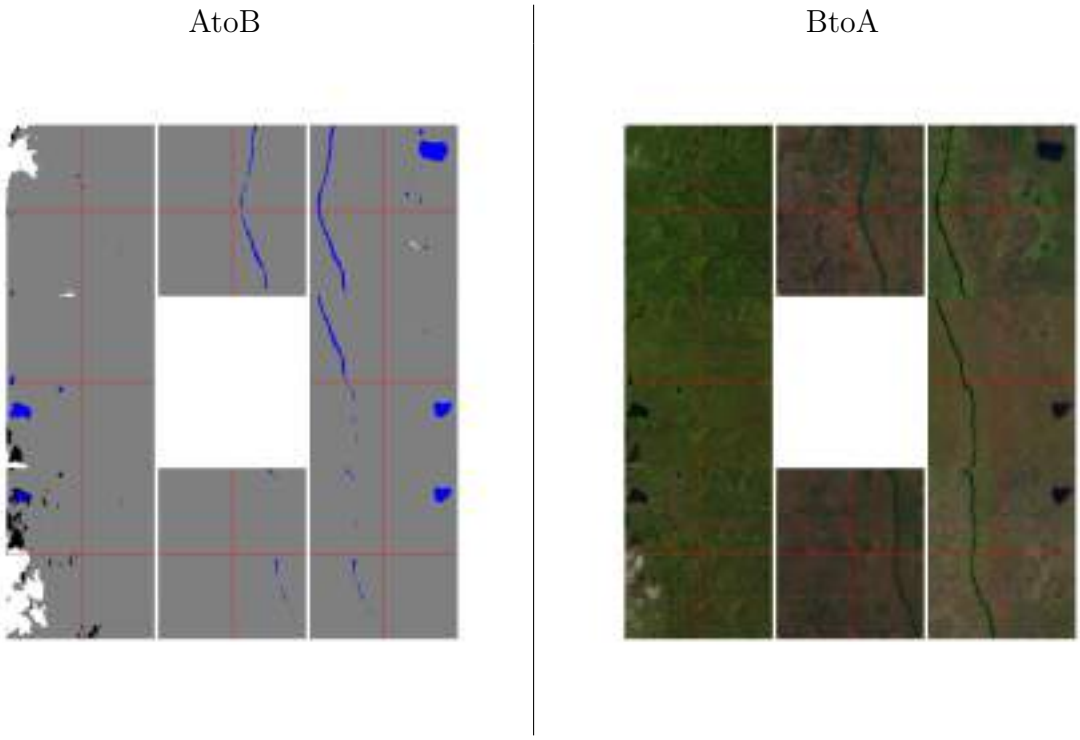


Figura 6.25: Generación de una pradera con un río.

El mismo efecto podemos verlo en la figura 6.26. En la fila superior las tonalidades cambian por cuadrantes dependiendo de la cantidad de agua en los alrededores.

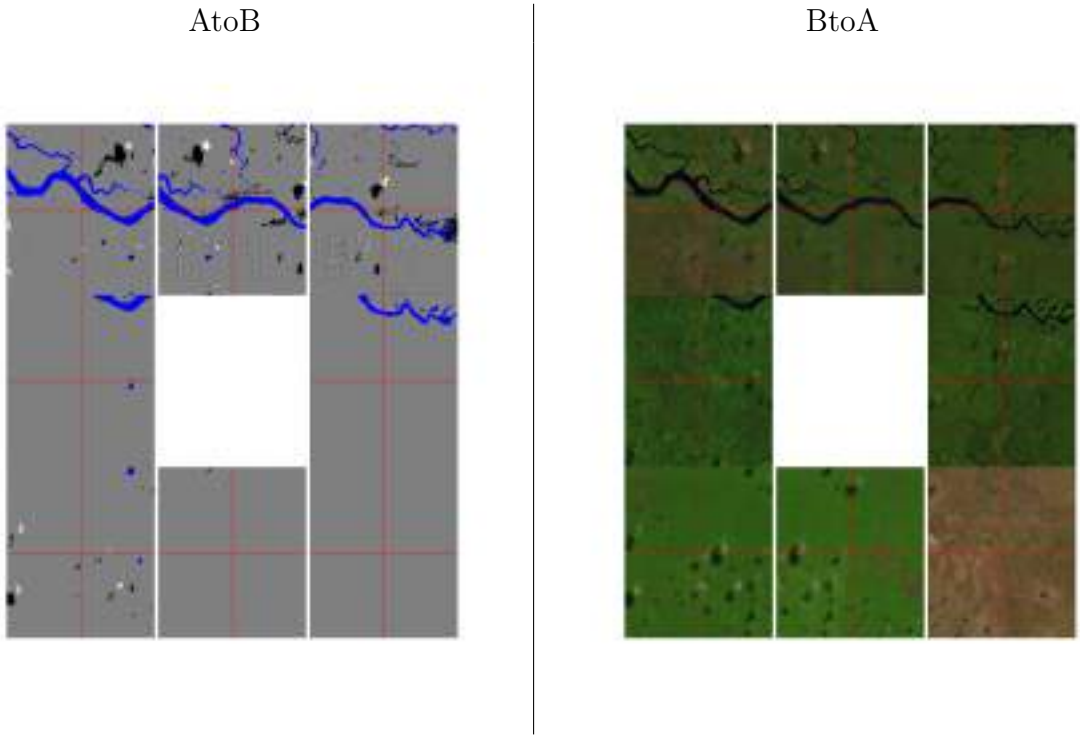


Figura 6.26: Generación de una pradera con agua y nieve.

En ocasiones las imágenes generadas sufren efectos diferentes como en el caso de

rotaciones, por ejemplo cambio de tonalidades sin un motivo evidente (figura 6.27).

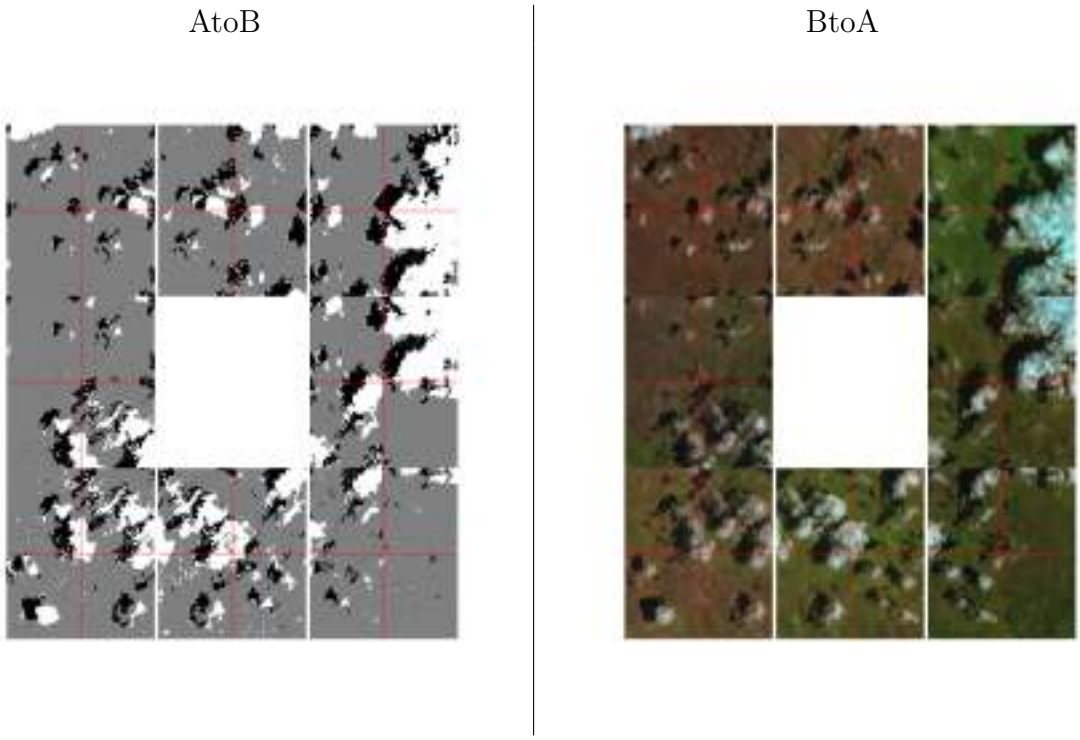


Figura 6.27: Generación de una pradera con agua y nieve.

Si observamos las figuras 6.28 y 6.29, podemos ver que las nubes también tienen tendencia a ir acompañadas de tonos más verdosos.

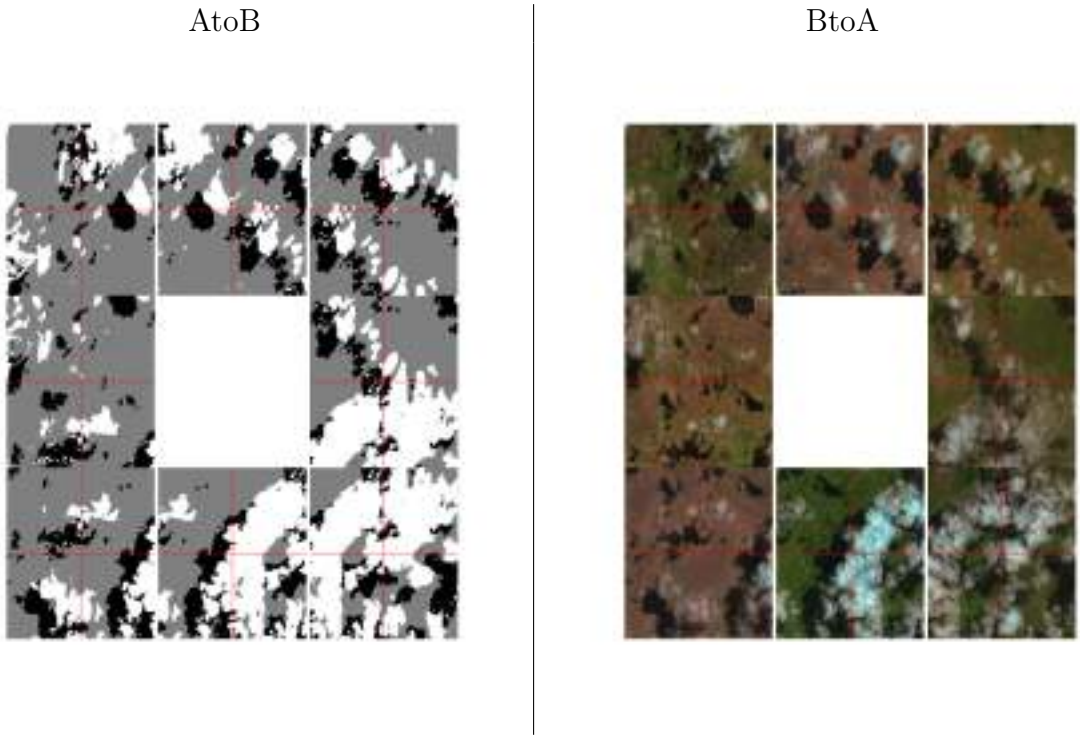


Figura 6.28: Generación de una pradera con nubes.

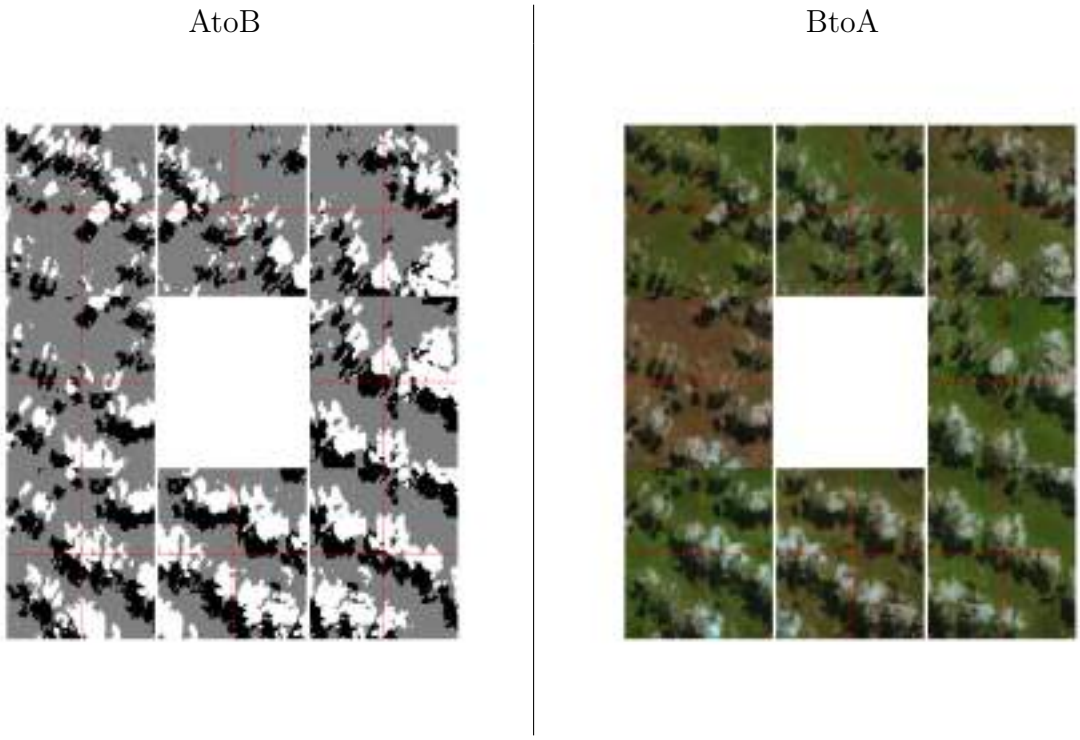


Figura 6.29: Generación de una pradera con nubes.

6.6. Pruebas con máscaras inventadas

Vamos a introducir máscaras inventadas con distintos colores para ver los resultados. Se ha empleado el isotipo de la Universidad de Valencia, *Xano*. En las figuras que se van a mostrar a continuación hay dos columnas, Máscara y fotografía. Máscara indica la máscara que se ha creado a mano mediante un editor de imágenes. Fotografía generada mantiene el significado empleado con anterioridad, se trata de la fotografía generada al introducir la máscara en la red.

En la figura 6.30 vemos un buen resultado. La red ha sido capaz de generar la estructura del isotipo sin problema. Por el contrario, en la figura 6.31 fracasó representar la estructura de las nubes sobre la pradera.

Máscara | Fotografía Generada

Figura 6.30: Pradera y agua.

Máscara | Fotografía Generada

Figura 6.31: Pradera y nubes.

Si se crea una máscara de nubes sobre agua (figura 6.32) obtenemos un resultado correcto en el cual se pueden apreciar los detalles de la máscara.

Máscara | Fotografía Generada

Figura 6.32: Agua y nubes.

Repetimos los procesos con una máscara un tanto diferente a la anterior. En la figura 6.33, igual que con el anterior caso de agua sobre pradera, se produce un resultado correcto.

Máscara | Fotografía Generada

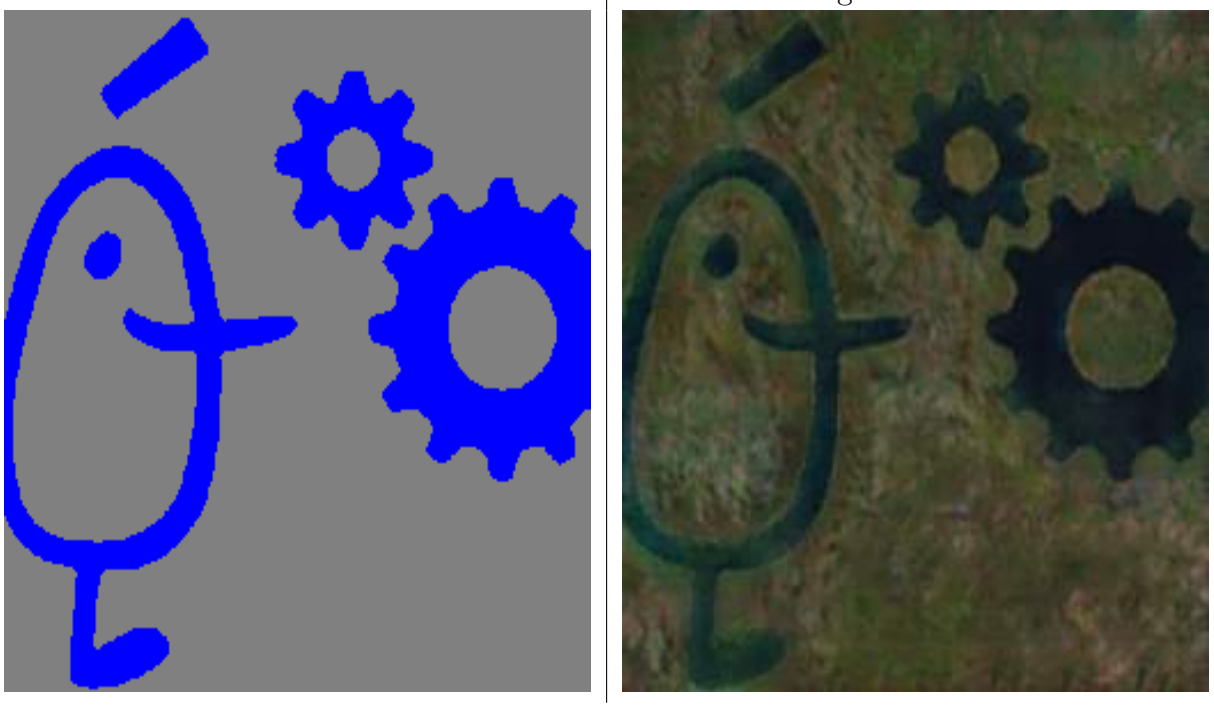
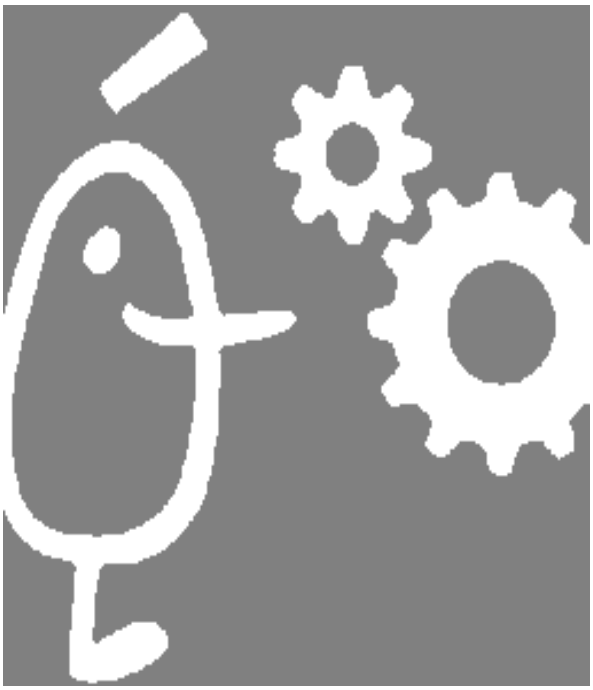


Figura 6.33: Pradera y nubes.

En la figura 6.34 observamos un resultado similar a la figura 6.31 ya que una vez más, una máscara con nubes sobre pradera genera una imagen poco natural. Finalmente el caso de nubes sobre agua de la figura 6.35 sí produce un resultado satisfactorio.

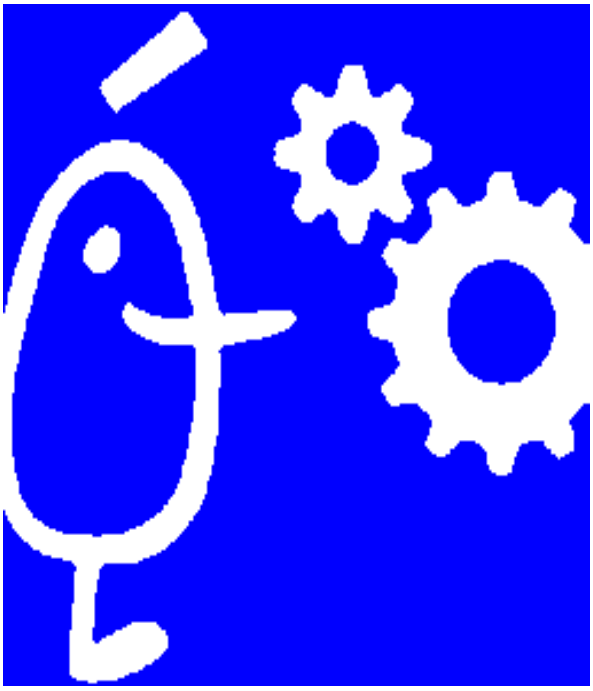
Máscara



Fotografía Generada

Figura 6.34: Pradera y nubes.

Máscara



Fotografía Generada

Figura 6.35: Agua y nubes.

6.7. Los peores casos

Vamos a mostrar diversos ejemplos en los que la red ha generado los peores resultados. La figura 6.36 corresponde a la generación de nieve con nubes y sombras.

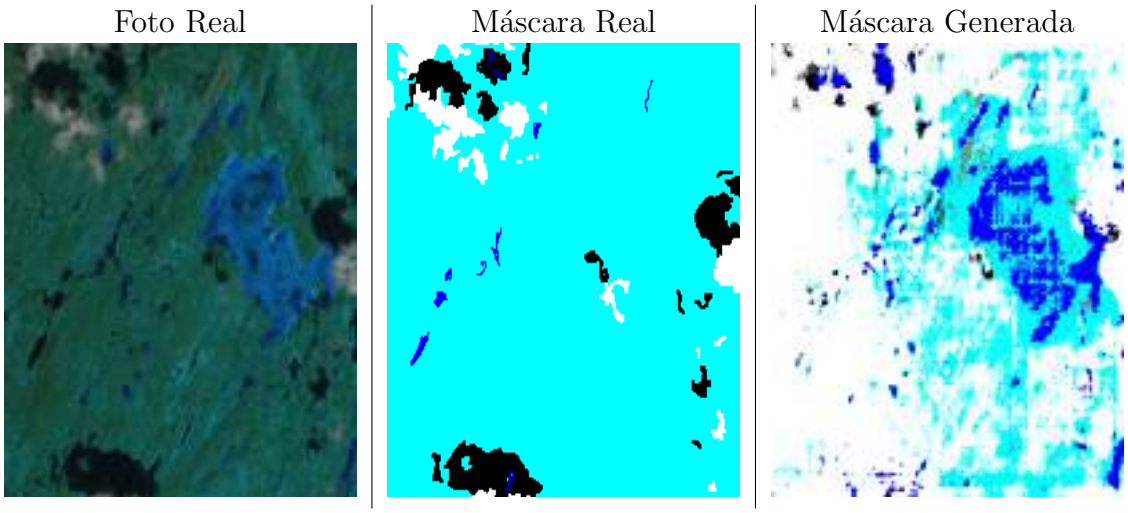


Figura 6.36: Generación de zona nevada con nubes y sombras.

En la figura 6.37 podemos ver los problemas que tiene la red para generar la clase de color cían, pues es confundida en muchos casos con nubes. Esta es la razón por la que, a pesar de que en general las nubes son correctamente clasificadas, el coeficiente Sørensen–Dice se ve afectada con una desviación tan grande.

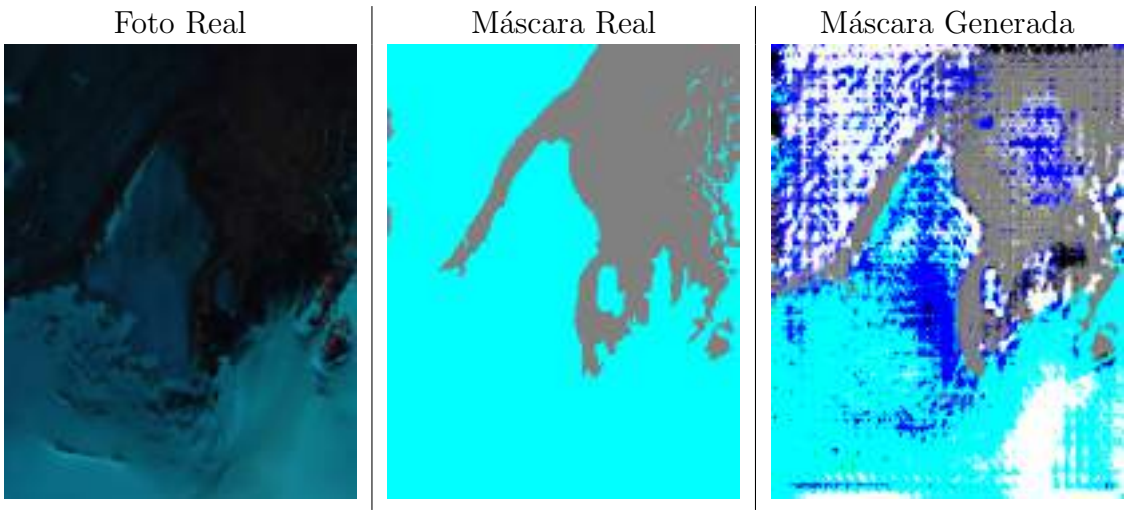


Figura 6.37: Generación zona montañosa nevada.

En el caso de BtoA con una máscara completamente gris (figura 6.38), podemos ver que se genera una fotografía que presenta un patrón irreal. Este es uno de los casos donde se observa un comportamiento propio de sobreentrenamiento.

Figura 6.38: Generación de pradera sin detalle.

6.8. Repitiendo el mismo caso

En esta sección vamos a mostrar qué pasa si generamos siempre la misma imagen de entrada, es decir, repetimos la misma imagen de test múltiples veces para ver si todas las imágenes generadas son idénticas o existen pequeñas diferencias debido a la aplicación de *dropout* en la fase de test.

6.8.1. AtoB

Partimos de la fotografía real que se puede ver en la figura 6.39 cuya máscara es generada seis veces (figura 6.40.

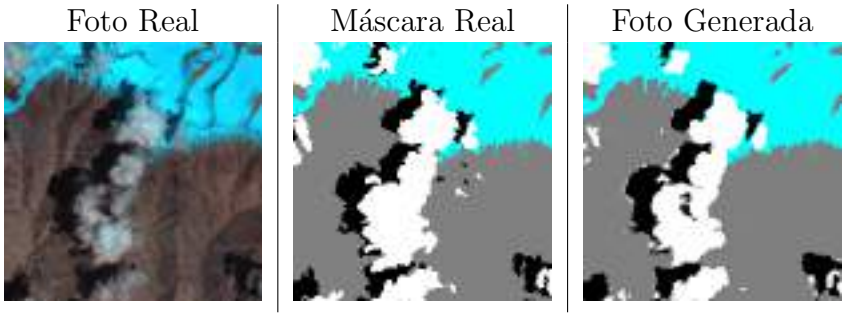


Figura 6.39: Pradera con nubes y nieve.

Observamos las diferencias entre las imágenes generadas son muy pequeñas, en ocasiones indistinguibles. Esto demuestra que a pesar de tener *dropout* en la fase de test, este afecta muy poco en los resultados finales para la red AtoB.

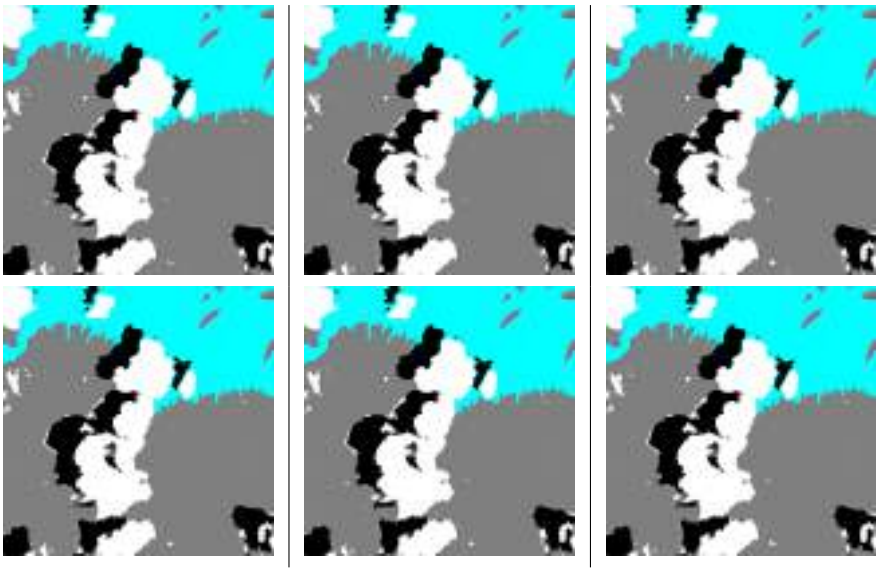


Figura 6.40: La misma máscara de 6.39 generada múltiples veces.

6.8.2. BtoA

Partimos de la máscara real que se puede ver en la figura 6.41 cuya fotografía es generada seis veces (figura 6.42.

Figura 6.41: Generación de río por una pradera.

La conclusión es la misma que la obtenida en la figura 6.40. Apenas se pueden observar diferencias visuales entre las imágenes.

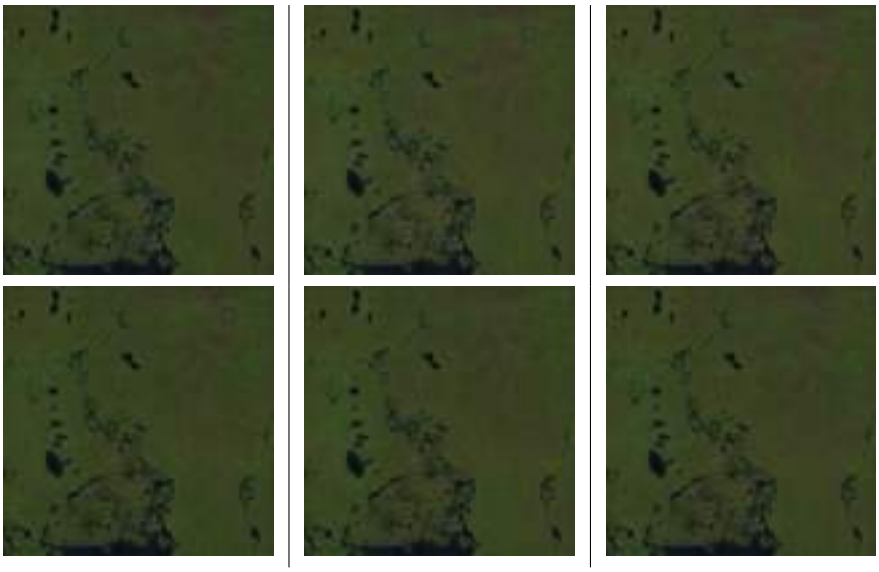


Figura 6.42: La misma fotografía de 6.41 generada múltiples veces.

Podemos afirmar una vez más para la red de los casos BtoA, que al aplicar *dropout* en la fase de test no se producen resultados diferentes para una misma entrada.

CAPÍTULO 7

Conclusiones finales

> I believe there is no deep difference between what can be achieved by a biological brain and what can be achieved by a computer. It, therefore, follows that computers can, in theory, emulate human intelligence — and exceed it
>
> Stephen Hawking

El entrenamiento de las redes neuronales adversarias (GAN) es más costoso que el de una red convolucional (CNN). Es por esto que con el mismo número de imágenes podemos realizar una mejor segmentación mediante el uso una red convolucional que solo identifique nubes [10], [12]. No obstante, este no era el principal objetivo del trabajo, pues la generación de imágenes sintéticas es uno de los mayores atractivos de las GAN. Por otra parte, no podemos olvidar que ciertas clases incluidas en los datos de entrenamiento eran francamente difíciles de identificar, provocando una mayor dificultad en el entrenamiento. Es posible que el entrenamiento mejorara enormemente si se usaran más casos donde las imágenes presentaran más comúnmente las clases que han sido generalmente mal identificadas, como es principalmente la nieve, sombra sobre agua e inundado. La clase correspondiente al agua no ha mostrado grandes resultados, en parte debido a que en muchas ocasiones la nieve era incorrectamente clasificada como agua. Esto provoca una disminución en su puntuación final.

Durante los diferentes entrenamientos y pruebas un fenómeno fue observado. La red alcanzó en muy pocas épocas un resultado aceptable, como se puede ver en las figuras 6.1 y 6.2, pero a continuación requería muchas épocas para disminuir la función de pérdida. Esto no se solucionaba cambiando el número de imágenes por lote, ni tampoco cambiando la tasa de aprendizaje (al menos en tiempos de entrenamiento similares). En todos los casos la función de pérdida alcanzó un resultado aceptable en unas pocas épocas, y a partir de ese momento mejorar la red suponía un esfuerzo muchísimo más alto. Una hipótesis del motivo, es el reparto de clases, pues en el conjunto de datos

existe una gran cantidad de casos de pradera, agua y nubes, mientras que los casos de nieve, terreno inundado y sombra sobre agua son mucho menos frecuentes.

Finalmente, podemos afirmar que el método de las redes cGAN funciona satisfactoriamente para la tarea que se ha propuesto resolver. Se han realizado evaluaciones positivas de los resultados tanto cuantitativamente como cualitativamente. Se han llevado a cabo diferentes pruebas de estrés donde se han mostrado peculiaridades en el proceso de generación de imágenes dando un poco luz para iluminar a la caja negra.

Bibliografía

[1] Rajendra Rana Bhat, Vivek Viswanath, and Xiaolin Li. Deepcancer: Detecting cancer through gene expressions via deep generative learning. *CoRR*, abs/1612.03211, 2016.

[2] Grigorios G. Chrysos, Jean Kossaifi, and Stefanos Zafeiriou. Robust conditional generative adversarial networks. *CoRR*, abs/1805.08657, 2018.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[4] Zheng Hui, Xiumei Wang, and Xinbo Gao. Fast and accurate single image super-resolution via information distillation network. *CoRR*, abs/1803.09454, 2018.

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017.

[6] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik G. Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. *CoRR*, abs/1712.00080, 2017.

[7] Simon Kohl, David Bonekamp, Heinz-Peter Schlemmer, Kaneschka Yaqubi, Markus Hohenfellner, Boris Hadaschik, Jan-Philipp Radtke, and Klaus H. Maier-Hein. Adversarial networks for the detection of aggressive prostate cancer. *CoRR*, abs/1702.08014, 2017.

[8] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[9] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014.

[10] Savas Özkan, Mehmet Efendioglu, and Caner Demirpolat. Cloud detection from RGB color remote sensing images with deep pyramid networks. *CoRR*, abs/1801.08706, 2018.

[11] Holger R. Roth, Hirohisa Oda, Xiangrong Zhou, Natsuki Shimizu, Ying Yang, Yuichiro Hayashi, Masahiro Oda, Michitaka Fujiwara, Kazunari Misawa, and Kensaku Mori. An application of cascaded 3d fully convolutional networks for medical image segmentation. *CoRR*, abs/1803.05431, 2018.

[12] Holger R. Roth, Chen Shen, Hirohisa Oda, Masahiro Oda, Yuichiro Hayashi, Kazunari Misawa, and Kensaku Mori. Deep learning and its application to medical image segmentation. *CoRR*, abs/1803.08691, 2018.

[13] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *CoRR*, abs/1710.08864, 2017.

[14] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.