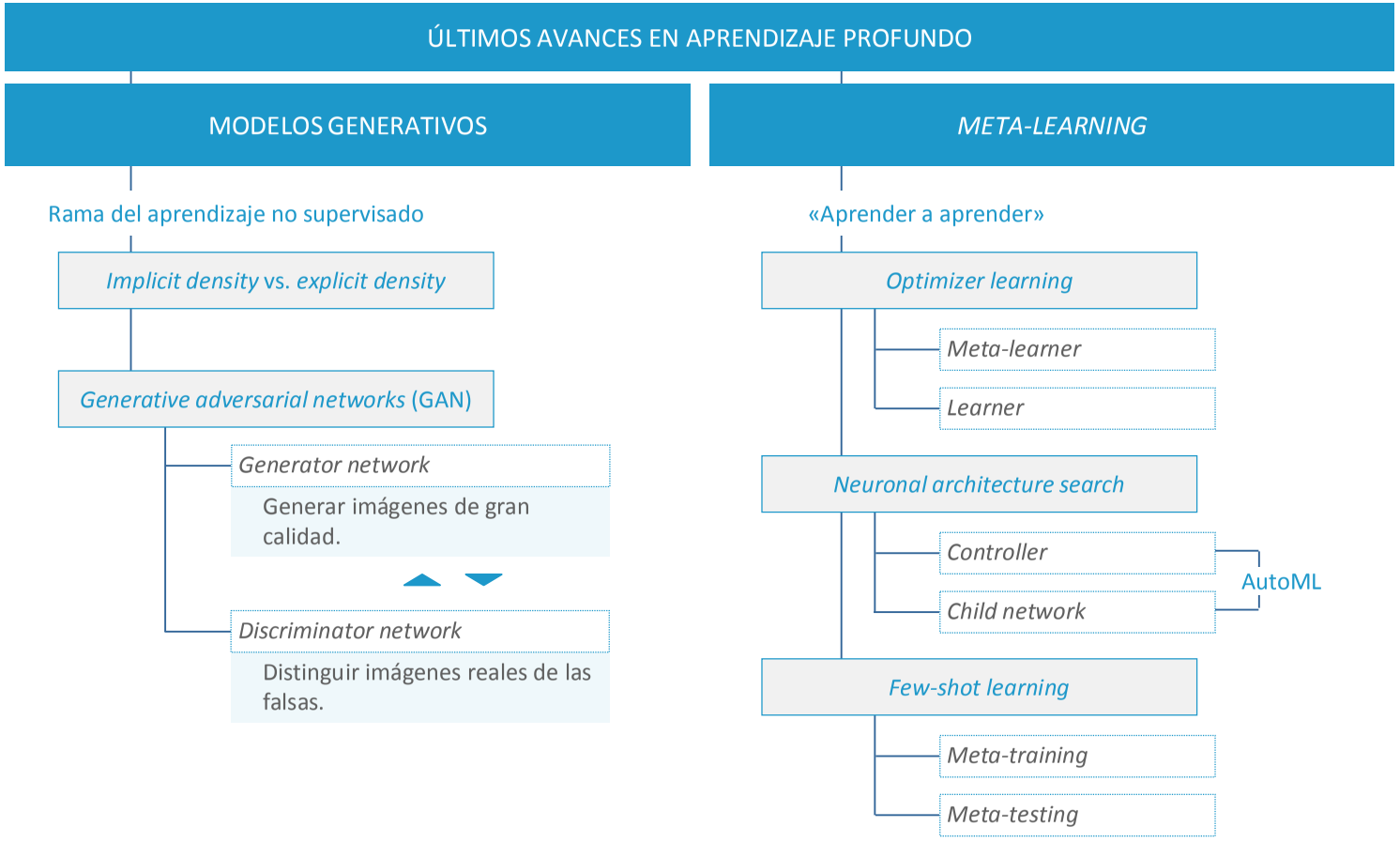
Sistemas Cognitivos Artificiales

Últimos avances en aprendizaje profundo

Índice

Esquema



Ideas clave

11.1. ¿Cómo estudiar este tema?

Para estudiar este tema deberás leer las **Ideas clave** que se desarrollan a continuación.

En este tema veremos algunos de los últimos avances y líneas de investigación en aprendizaje profundo. En particular, un nuevo tipo de red neuronal, las *generative adversarial networks* (GAN), y el concepto de *meta-learning*.

11.2. *Generative adversarial networks* (GAN)

Una de las áreas de investigación con más movimiento en la actualidad, dentro del mundo del aprendizaje profundo, es el de las ***generative adversarial networks* (GAN)**, introducidas por primera vez por Ian Goodfellow en 2014.

Las GAN son un tipo de **modelo generativo**. Los modelos generativos son una rama del aprendizaje no supervisado en la que, dado un conjunto de datos de entrenamiento, se intenta aprender la distribución de probabilidad de esos datos. A partir de esta distribución de probabilidad, podemos generar nuevos ejemplos similares a los de entrenamiento. Por ejemplo, si dado un conjunto de imágenes, nuestro modelo es capaz de aprender la distribución de probabilidad de esas imágenes, es posible generar nuevas imágenes parecidas a las originales mediante el muestreo (*sample*) de la distribución aprendida.

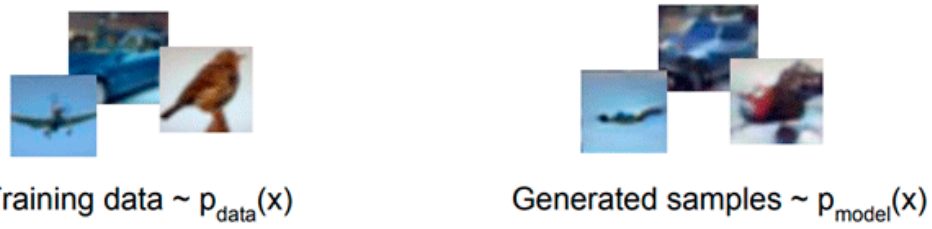


Figura 1. Modelos generativos.
Fuente: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture12.pdf

Esto es similar a lo que vimos en los **modelos del lenguaje**. En ellos considerábamos que el texto venía dado por una distribución de probabilidad en la que cada palabra depende de las palabras anteriores. Como vimos, la distribución de probabilidad aprendida puede ser usada para generar texto. En el caso de los modelos del lenguaje, el objetivo era aprender directamente una función de densidad. Esta estrategia para obtener un modelo generativo es usada también en otras técnicas actuales como:

- PixelRNN.
- PixelCNN.
- Variational Autoencoders (VAE).

En las GAN, a diferencia de estos últimos, no se intenta modelar una función de densidad (con la que posteriormente se pueden generar ejemplos), sino que directamente se aprende a generar ejemplos. Este tipo de modelo generativo se conoce como ***implicit density estimation*** (en oposición a *explicit density estimation*).

La popularidad de las GAN se debe principalmente a sus impresionantes resultados, con modelos capaces de generar imágenes de gran realismo. Se aproximan al problema de la generación desde el punto de vista de un juego para dos jugadores. Estos dos jugadores son dos redes neuronales conocidas como *generator network* y *discriminator network*.

La ***generator network*** es una aproximación a la obtención de una muestra (*sample*) de la distribución de los datos de entrenamiento. En otras palabras: se encarga de

generar imágenes parecidas al *training* set. Su *input* es un vector aleatorio, un vector que podemos cambiar a placer para generar imágenes distintas.

La ***discriminator network*** es una red que intenta distinguir entre imágenes reales (pertenecientes a los datos de entrenamiento) y falsas (generadas por la *generator network*).

El juego es el siguiente: la *generator network* tiene como objetivo engañar a la *discriminator network* mediante la generación de imágenes que pasen por ser reales a ojos de esta. Por su parte, la *discriminator network* intenta aprender la diferencia entre imágenes reales y falsas para no ser engañada. De este modo, al entrenar una GAN obtenemos una *generator network* que es capaz de producir réplicas de gran calidad, gracias a que intenta engañar a una red discriminadora que, a su vez, se vuelve cada vez más efectiva al ser entrenada. En definitiva, obtenemos una red generadora que ha aproximado el proceso de obtención de muestras a partir de la función de densidad de los datos de entrenamiento, pero sin modelar esta función directamente.

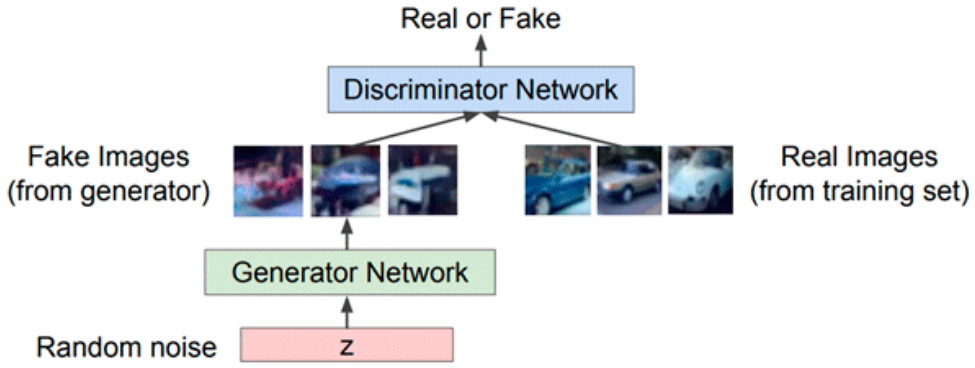


Figura 2. *Generative Adversarial Network*.
Fuente: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture12.pdf

Las dos redes son entrenadas a la vez. El entrenamiento o juego entre la *generator* y la *discriminator network* se plasma en la siguiente *objective function*:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_d} \left(G_{\theta_g}(z) \right) \right) \right]$$

Donde:

- *D* es la función del discriminador, con salida de un valor entre 0 y 1 (la probabilidad de que la imagen sea real).
- *G* representa al generador (con salida de una imagen).

De este modo**, el discriminador intenta maximizar el objetivo**, de manera que:

- $D(x)$ sea lo más cercano posible a 1 (imagen real).
- Y $D(G(z))$ sea lo más cercano posible a 0 (*fake*).

Por su parte, **el generador intenta minimizar el objetivo**, de manera que el discriminador piense que las imágenes generadas sean reales, esto es, que $D(G(z))$ sea cercano a 1.

Como podemos imaginar del hecho de que estemos entrenando dos redes a la vez, el proceso de entrenamiento de una GAN es algo complejo y relativamente inestable, por lo que hay una serie de trucos y mejores prácticas para conseguir entrenarlas con éxito. Esto hace que el entrenamiento de GAN sea un área de investigación muy candente en estos momentos.

El proceso de entrenamiento seguido en el artículo original de Ian Goodfellow (2014) se describe en el siguiente algoritmo.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figura 3. Proceso de entrenamiento seguido por Goodfellow.
Fuente: Goodfellow et al. (2014, p. 4).

11.3. *Meta-learning*

Meta-learning es otra área de investigación de gran interés para la comunidad de inteligencia artificial en la actualidad, si bien sus inicios datan de hace varias décadas.

La idea básica de *meta-learning* es aprender y mejorar el proceso del aprendizaje o, en otras palabras, «aprender a aprender».

Los humanos somos muy versátiles en aprender nuevas tareas recibiendo una mínima información. Por ejemplo, somos capaces de comprender cómo es un tipo de objeto sin necesidad de recibir miles de imágenes de distintas instancias de ese objeto, así como de utilizar nuestras experiencias pasadas para aprender nuevas tareas. A diferencia de nosotros, los algoritmos que hemos visto hasta ahora suelen ser muy efectivos a la hora de hacer una tarea específica, pero no son capaces de generalizar a nuevas tareas y, además, necesitan una gran cantidad de información específica (pensemos, por ejemplo, en todas las imágenes necesarias para un

detector de objetos o en las grandes cantidades de texto para entrenar un modelo del lenguaje).

De este modo, nos gustaría que los sistemas de inteligencia artificial fueran más versátiles y pudieran aprender una larga serie de habilidades, sin que cada habilidad requiriera de un arduo proceso de obtención de datos y de un largo entrenamiento posterior. En definitiva, nos gustaría que los sistemas de inteligencia artificial «aprendieran a aprender» nuevas tareas de manera rápida usando su experiencia de aprendizaje anterior, en vez de considerar cada tarea de manera aislada. Este es el objetivo del *meta-learning*, una compleja área de investigación, pero que está empezando a arrojar resultados prometedores. En este tema veremos algunos acercamientos al problema.

Es importante mencionar que *meta-learning* es un concepto distinto (aunque relacionado) del *hyperparameter tuning*. En este, como sabemos, se intenta encontrar la mejor combinación posible de hiperparámetros de un modelo en particular, de modo que obtengamos los mejores resultados posibles a la hora de entrenar ese modelo.

Veamos ahora algunas formas de definir el problema de *meta-learning*.

Optimizer learning

Como hemos visto en este curso, la elección del optimizador es de gran importancia a la hora de entrenar redes neuronales profundas. Existe una gran variedad de ellos y, pese a que sabemos que algunos suelen funcionar bien, no siempre está claro cuál es la mejor opción para nuestra arquitectura en concreto. De este modo, podríamos preguntarnos: ¿por qué no aprendemos también el optimizador a utilizar, de manera que sea óptimo para que nuestra red aprenda de manera más rápida la tarea a resolver?

En este tipo de *meta-learning*, hay una red neuronal auxiliar, conocida como *meta-learner,* que se encarga de aprender cómo hacer un *update* de los parámetros de la red principal (*learner*). La red *meta-learner* suele ser normalmente una RNN, lo cual facilita que esta recuerde cómo ha cambiado los parámetros en iteraciones anteriores. Esta red puede ser entrenada mediante aprendizaje supervisado o *reinforcement learning*.

Neural architecture search

Otro acercamiento al problema de *meta-learning* es la *neural architecture search*. Como hemos visto durante el curso, la búsqueda de arquitecturas de redes neuronales es una tarea compleja que requiere un elevado grado de conocimiento y experimentación. Una clara referencia son las complejas redes para *computer vision* que vimos en el tema de *convolutional networks*.

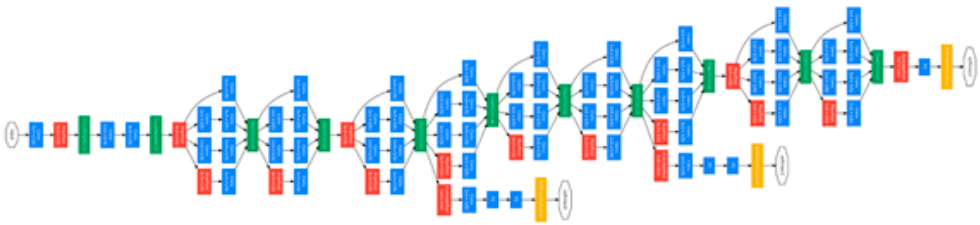


Figura 4. Arquitectura GoogleNet.
Fuente: https://ai.googleblog.com/2017/05/using-machine-learning-to-explore.html

Neural architecture search trata el problema de **diseñar una red** como un problema de *meta-learning*: se intenta aprender a diseñar redes neuronales de manera que aprendan de manera eficiente distintas tareas.

Un ejemplo de este tipo de aproximamiento al *meta-learning* es AutoML de Google. En este:

- Una red neuronal recurrente (*controller*) se encarga de proponer un modelo (*child network*), que es entrenado y evaluado para una tarea.
- El *feedback* recibido de esta evaluación se utiliza para informar al *controller* (siguiendo un marco de *reinforcement learning*) de cómo han afectado los cambios en la tarea.
- Este proceso se repite miles de veces, permitiendo al *controller* diseñar modelos de mayor calidad para la tarea en cuestión.

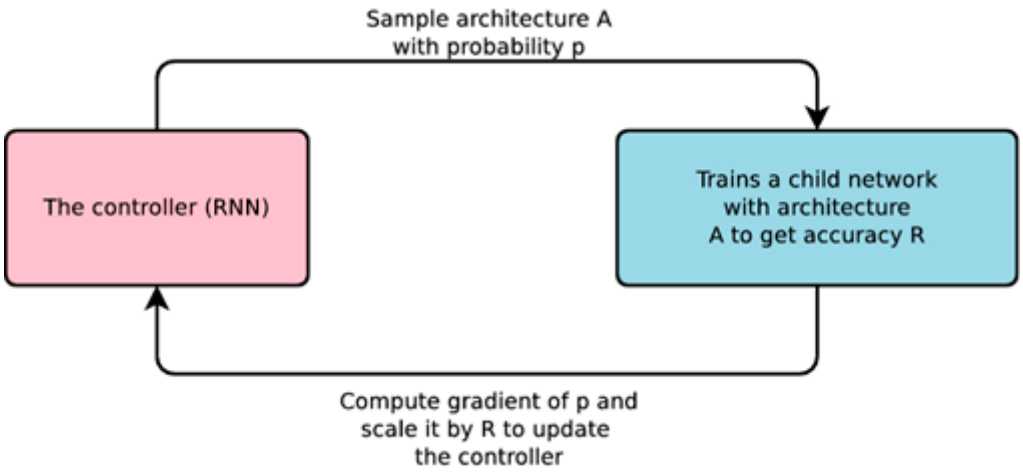


Figura 5. Ejemplo de *Neural Architecture Search*.
Fuente: https://ai.googleblog.com/2017/05/using-machine-learning-to-explore.html

Few-shot learning

En *few-shot learning*, el objetivo es conseguir que una máquina pueda generalizar conceptos y aprender a partir de un **número reducido de ejemplos**, tal y como haría un humano, y no a partir de datasets de gran tamaño. El problema se suele formular de manera que tenemos una serie de tareas distintas a aprender (cada una con sus correspondientes datos de entrenamiento y test) y queremos evaluar la capacidad de nuestro sistema de *meta-learning* para aprender nuevas tareas.

Figura 6. Ejemplo de *few-shot learning*.
Fuente: https://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/

En la imagen anterior, vemos un ejemplo con tareas de clasificación de imágenes (cada fila de imágenes es una tarea distinta). Durante el proceso de *meta-learning*, el modelo, (o más bien, meta-modelo), es entrenado para aprender tareas en el set de *meta-training*, y su capacidad de aprender nuevas tareas es evaluada en el set de *meta-testing*.

Para cada tarea se entrena un modelo diferente que se encarga de resolver dicha tarea. Por lo tanto, se manejan **dos niveles de optimización** a la vez: el *learner*, que aprende las tareas a realizar, y el *meta-learner*, que entrena al primero de modo que este sea efectivo aprendiendo a partir de la limitada cantidad de datos disponible.

11.4. Referencias bibliográficas

Goodfellow et al. (2014). Generative adversarial networks. En Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence y K. Q. Weinberger (Eds.). *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. Recuperado de https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

Lo + recomendado

Lecciones magistrales

Aplicaciones de las Generative Adversarial Networks (GAN)

Veremos una serie de aplicaciones de las GAN comentando su funcionamiento y cómo ha sido la evolución en la generación de imágenes.

Accede a la lección magistral a través del aula virtual

No dejes de leer

Using Machine Learning to Explore Neural Network Architecture

Le, Q. y Zoph, B. (17 de mayo de 2017). Using Machine Learning to Explore Neural Network Architecture [Blog post].

Post de Google AI describiendo el funcionamiento de AutoML que hemos visto brevemente en el apartado de *Neural architecture search.*

Accede al artículo a través del aula virtual o desde la siguiente dirección web:
https://ai.googleblog.com/2017/05/using-machine-learning-to-explore.html

No dejes de ver

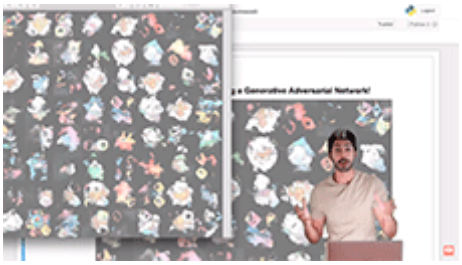
Progressive Growing of GANs for Improved Quality, Stability, and Variation

Impresionante vídeo que muestra los resultados de imágenes generadas con GAN, las cuales están dotadas de gran realismo.

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:
https://youtu.be/G06dEcZ-QTg

Generating Pokémon with a Generative Adversarial Network

Videotutorial sobre GAN en el que se explica tanto teoría como código a través de la generación de nuevos *pokémon*.



Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

https://youtu.be/yz6dNf7X7SA

Modelos generativos en CS231N

Conferencia que da una detallada descripción de los modelos generativos utilizados en aprendizaje profundo, incluyendo PixelRNN, PixelCNN, *Variational Autoencoders* y GAN.

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

https://youtu.be/5WoItGTWV54

+ Información

A fondo

Learning to Learn

Finn, C. (18 de julio de 2017). Learning to Learn [Blog post].

Análisis muy completo de *meta-learning*, con multitud de referencias. Incluye la descripción de *Model-Agnostic Meta-Learning* (MAML).

Accede al artículo a través del aula virtual o desde la siguiente dirección web:
http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/

Test

1. Con un modelo generativo (marca la respuesta correcta):

A. Podemos generar nuevos ejemplos de datos a partir de lo aprendido con los datos de entrenamiento.

B. Aprendemos la distribución de probabilidad de las *labels* «*y*» a partir de los datos «*x*», con lo cual podemos generar nuevos datos.

C. Ninguna de las anteriores.

2. Las *generative adversarial networks* (marca la respuesta correcta):

A. Modelan explícitamente una función de densidad.

B. Modelan implícitamente una función de densidad.

C. Tanto A como B son ciertas.

D. Ni A ni B son ciertas.

3. Al entrenar una GAN (marca la respuesta correcta):

A. Solo la *generator network* mejora con el tiempo. La *discriminative network* se mantiene constante y el proceso de entrenamiento termina cuando esta es engañada en la mayoría de ocasiones.

B. Solo la *discriminator network* mejora con el tiempo. El entrenamiento termina cuando esta es capaz de distinguir entre resultados reales y falsos.

C. Ambas, *generator* y *discriminator network*, mejoran con el tiempo. La *generator* aprende a obtener resultados más parecidos a los datos reales, mientras que la *discriminator* aprende a distinguir mejor, forzando a su vez a la otra red a mejorar con el tiempo.

D. Ninguna de las anteriores.

4. El resultado de interés al entrenar una GAN es principalmente (marca la respuesta correcta):

A. La *generator network*: es el modelo generativo resultante.

B. La *discriminator network*: nos permite distinguir *inputs* falsos.

C. La combinación de ambas: nos permite generar imágenes y luego comprobar si son suficientemente buenas.

5. La salida de $D(x)$ de la *discriminator network* (marca la respuesta correcta):

A. Es una imagen u otro contenido a generar, según la aplicación.

B. Es la probabilidad de que la imagen de entrada de la red sea verdadera o falsa.

C. Es la probabilidad de que el generador haya generado una imagen real.

C. Es la probabilidad de que el generador haya generado una imagen falsa.

6. En una *generator network*, podemos generar imágenes distintas (marca la respuesta correcta):

A. Cambiando los valores del *input*. Distintos valores aleatorios de entrada arrojan distintas imágenes de salida.

B. Suministrando una imagen distinta al *input* de la *generator network*.

C. Ninguna de las anteriores.

7. Los sistemas actuales de inteligencia artificial (marca la respuesta correcta):

A. Aprenden rápidamente cualquier tipo de tarea sin necesidad de una gran cantidad de datos y búsqueda de arquitecturas.

B. Necesitan una gran cantidad de datos para aprender una tarea, pero luego son capaces de aprender tareas similares con una efectividad cercana a las de los humanos.

C. Principalmente, son muy efectivos en tareas concretas para las que hemos encontrado buenos algoritmos y una buena fuente de datos.

8. ¿Cuáles de las siguientes son formas de aplicar *meta-learning*? (Marca las respuestas correctas):

A. Aprender el optimizador a utilizar para una tarea.

B. Realizar un *hyperparameter tuning* en nuestro modelo probando varios hiperparámetros distintos al azar.

C. Aprender cómo generar arquitecturas de redes neuronales más efectivas para una tarea.

D. Utilizar varios tipos de modelos distintos para ver cuál da mejores resultados en una tarea.

9. En AutoML (marca la respuesta correcta):

A. Se intenta aprender un optimizador óptimo para un problema en concreto.

B. Se intenta aprender una serie de hiperparámetros óptimos para un problema en concreto.

C. Se intenta generar una red neuronal óptima para un problema en concreto.

10. En *few-shot learning* (marca la respuesta correcta):

A. El objetivo es que un sistema de inteligencia artificial sea capaz de generalizar conceptos y aprender a partir de pocos ejemplos.

B. Se intenta que una máquina aprenda sin ningún dato de entrenamiento.

C. El objetivo es conseguir mejores arquitecturas de redes neuronales.