**1) En base a lo visto en el paper ¿Qué significan los parámetros definidos en la siguiente celda?**

* Total\_variation\_weight: Es el valor por el cual se pesa la total\_variation\_loss, la cual actúa como un término regulador
* Style\_weight: Es el valor por el cual se pesa la style\_loss. En el paper se representa con α
* Content\_weight: Es el valor por el cual se pesa la content\_loss. En el paper se representa con β

**2) Explicar qué hace la siguiente celda. En especial las últimas dos líneas de la función antes del return. ¿Por qué?**

*def preprocess\_image(image\_path):  
 img = load\_img(image\_path, target\_size=(img\_nrows, img\_ncols))  
 img = img\_to\_array(img)  
 img = np.expand\_dims(img, axis=0)  
 img = vgg19.preprocess\_input(img)  
 return img*

* Primera línea: Carga la imagen en la variable img, utilizando el path provisto y le asigna las medidas img\_nrows para la altura y img\_ncols para el ancho.
* Segunda línea: Convierte la imagen en un array
* Tercera línea: Agrega una nueva dimensión al array de la imagen. Esta dimensión cumple la función del batch size.
  + Luego de esta línea, el shape de la variable img será (batch size, alto, ancho, canales).
* Cuarte línea: Aplica a la imagen el preprocesamiento requerido para los inputs, por la VGG19 en Keras.
* Ultima línea: Devuelve la imagen pre procesada

**3) Habiendo comprendido lo que hace la celda anterior, explique de manera muy concisa qué hace la siguiente celda. ¿Qué relación tiene con la celda anterior?**

*def deprocess\_image(x):  
 x = x.reshape((img\_nrows, img\_ncols, 3))  
 # Remove zero-center by mean pixel  
 x[:, :, 0] += 103.939  
 x[:, :, 1] += 116.779  
 x[:, :, 2] += 123.68  
 # 'BGR'->'RGB'  
 x = x[:, :, ::-1]  
 x = np.clip(x, 0, 255).astype('uint8')  
 return x*

En la primera línea de Código, se le saca al array X (la imagen), la dimensión agregada con el expand\_dims en la función anterior.  
Luego, se procede a revertir todos los cambios que realiza "vgg19.preprocess\_input" en la imagen.

En resumen, lo que obtengo como salida de la función es idéntico a lo que obtengo luego de utilizar las 2 primeras líneas de código de la función preprocess\_image

**4) En las siguientes celdas:**

* ¿Qué es la matriz de Gram? ¿Para qué se usa?
  + La matriz de Gram es la multiplicación de una matriz con su matriz transpuesta, donde el objetivo es que cada valor interno sea igual al cuadrado del original
  + Se utiliza para calcular la independencia lineal
* ¿Por qué se permutan las dimensiones de x?
  + Para poder realizar la matriz de Gram, necesito que las dimensiones de mi matriz sean (Cantidad de canales, (Alto \* Ancho))
  + La función K.batch\_flatten conserva la primera dimensión y aplica un “flatten” entre las restantes.
    - Por lo tanto, necesito que quede posicionada primera mi dimensión de la cantidad de canales y por último mis dimensiones de alto y ancho de la imagen.

**5) Losses:**

**Explicar qué mide cada una de las losses en las siguientes tres celdas.**

* Style loss: Minimiza la distancia media cuadrada entre la matriz de Gram de la imagen original y la matriz de Gram de la imagen generada
  + Esto lo hace evaluando las matrices en cada una de las capas definidas más adelante en la lista "feature\_layers"
* Content loss: Minimiza la distancia media cuadrada entre los “features” de la imagen generada menos los features de la imagen original.
* Total variation loss: Mide la cantidad de sonido que posee la imagen.
  + Se utiliza para reducir el sonido que posee la imagen
  + Actúa como un término de regularización

**6) Explique el propósito de las siguientes tres celdas. ¿Qué hace la función fmin\_l\_bfgs\_b? ¿En qué se diferencia con la implementación del paper? ¿Se puede utilizar alguna alternativa?**

* Función eval\_loss\_and\_grads:
  + Primero se hace un reshape de la entrada x
    - *x = x.reshape((1, img\_nrows, img\_ncols, 3))*
  + Se utiliza la K.function definida en la celda anterior, la cual devuelve en primera posición la loss y en la segunda, los gradientes
    - *outs = f\_outputs([x])*
  + Se guarda el valor de la loss obtenida al usar la función en el punto anterior.
    - *loss\_value = outs[0]*
  + Dependiendo del largo de outs, se hace el flatten sobre la salida de los gradientes de la función f\_outputs()
    - *if len(outs[1:]) == 1:*
      * *grad\_values = outs[1].flatten().astype('float64')*
    - *else:*
      * *grad\_values = np.array(outs[1:]).flatten().astype('float64')*
  + Se devuelve el valor de la loss y los gradientes
    - *return loss\_value, grad\_values*
* Clase Evaluator:
  + Permite calcular la loss y el gradiente en una sola pasada y las guarda en 2 funciones separadas.
  + Esto se hace ya que scipy.optimize necesita funciones separadas para la loss y el gradiente.
* Función Evaluator.loss
  + Primero se asegura de que el valor self.loss.value, sea nulo
  + Luego, utiliza la función eval\_loss\_and\_grads para calcular el valor de la loss y el gradiente.
  + Por último, los almacena en self.loss\_value y self.grads\_values y devuelve loss\_value
* Función Evaluator.grads
  + Primero comprueba que self.loss\_value, no sea nulo
  + Luego, copia los valores de self.grad\_values a grad\_values
  + Por último, limpia los valores self.loss\_value y self.grad\_values y devuelve grad\_values
* Tercera celda:
  + Primero se instancia la clase Evaluator() y se trae la imagen de base y se la pre procesa.
    - *evaluator = Evaluator()*
    - *x = preprocess\_image(base\_image\_path)*
  + Luego se realiza un for de n iteraciones, donde cada una realiza lo siguiente:
    - Muestra la iteración en la cual se encuentra
      * *print('Start of iteration', i)*
    - Se toma el tiempo inicial
      * *start\_time = time.time()*
    - Se calcula la loss utilizando fmin\_l\_bfgs\_b
      * *x, min\_val, info = fmin\_l\_bfgs\_b(evaluator.loss, x.flatten(), fprime=evaluator.grads, maxfun=20)*
    - Se muestra el valor de la loss
      * *print('Current loss value:', min\_val)*
    - Se desprocesa la imagen de la salida para poder visualizarla luego
      * *img = deprocess\_image(x.copy())*
    - Se guarda la imagen
      * *fname = result\_prefix / ('output\_at\_iteration\_%d.png' % i)*
      * *save\_img(fname, img)*
    - Se toma el tiempo final
      * *end\_time = time.time()*
    - Se imprime el nombre del archivo guardado y el tiempo que se tomo para realizar la iteración
      * *print('Image saved as', fname)*
      * *print('Iteration %d completed in %ds' % (i, end\_time - start\_time))*
* Función fmin\_l\_bfgs\_b:
  + Minimiza una función dada, utilizando el algoritmo L-BFGS-B
    - Algoritmo BFGS:
      * Es un método iterativo, que permite resolver problemas de optimización no lineales, determinando la dirección de descenso del gradiente.
      * Para calcularlo, utiliza la inversa de la matriz Hessiana y guarda una matriz densa de n x n, donde n es la cantidad de variables en el problema
    - Algoritmo L-BFGS (Limited Memory BFGS)
      * A diferencia de BFGS, L-BFGS utiliza para su matriz solo algunos vectores que representan una aproximación implícita. Para realizar esto, L-BFGS conserva historia de las ultimas m posiciones del gradiente de x.
    - Algoritmo L-BFGS-B
      * Sumado a lo destacado en los puntos anteriores, permite utilizar restricciones en las variables en formato de limites superiores e inferiores.
  + Dentro del Código, se utilizaron los siguientes valores para la función
    - Función para minimizar --> Evaluator.loss
    - Punto de partida / valor inicial --> preprocess\_image(imagen\_base).flatten() (x.flatten())
    - Gradiente de la función a minimizar --> fprime=evaluator.grads
    - Cantidad máxima de evaluaciones de la funcion --> maxfun=20
  + Esta funcion, devuelve 3 valores.
    - En la primera posición, devuelve un array con la posición estimada del mínimo, para este caso, la imagen modificada.
    - En la segunda posición, devuelve el valor de la funcion en dicho mínimo.
    - En la tercera posición, devuelve un diccionario informativo con
      * información si convergió, o si no lo hico debido a que pasaron muchas iteraciones u otra razón.
      * Si el gradiente se encuentra en el mínimo
      * La cantidad de llamados a la funcion realizados
      * Cantidad de iteraciones realizadas
* Diferencia entre el paper y la notebook.
  + La principal diferencia que se puede observar entre la implementación de esta notebook y el paper, es el uso de la total\_variation\_loss
    - En el paper, utilizan únicamente la relación entre el content\_weight y el style\_weight, para regular el ruido que se percibe en la imagen y el nivel de "combinación deseado"
  + Otra diferencia que se puede apreciar es que, en el paper, deciden utilizar average pooling en vez de max pooling, ya que con esto obtienen mejores resultados.

# 7) Ejecute la siguiente celda y observe las imágenes de salida en cada iteración.

# \*Ver imágenes adjuntas (Pesos-default.rar)

# 8) Generar imágenes para distintas combinaciones de pesos de las losses. Explicar las diferencias. (Adjuntar las imágenes generadas como archivos separados.)

En el primer ejemplo, decidí generar nuevas imágenes modificando únicamente la style\_weight, la cual paso de 10 a 10.000, dejando el valor en para el ratio entre α y β en 10^-4.   
La diferencia se puede distinguir fácilmente en la imagen 99 de ambas salidas. En las imágenes, se puede apreciar, que aquellas generadas utilizando un style\_weight de 10.000, poseen una mayor deformación de la imagen original, con mayores curvaturas y colores propios de la imagen de estilo utilizada.

El segundo ejemplo, opte por dejar la style weight en su valor original y modificar la content weight, la cual fue modificada de 1 a 100. Esto nos deja con un ratio entre el α y β de 10. En este caso, como era de esperarse, se puede distinguir rápidamente que los colores de la foto son mucho más similares a los de la imagen base y el estilo de la noche estrellada, se encuentra mucho menos presente.

# 9) Cambiar las imágenes de contenido y estilo por unas elegidas por usted. Adjuntar el resultado.

# \*Ver imágenes adjuntas (Estilo-e-imagen-propio.rar) Para esta punto, decidí utilizar varias combinaciones distintas. Decidí utilizar imágenes de base con bastante simples y otras un poco más complejas, así como también para el caso de las imágenes de estilo. En cuanto a los pesos de las losses, dejé los que estaban en la notebook por default.