PRÁCTICA 4 - VPC

Style Transfer

Cristian Villarroya Sánchez

01-06-2021

Índice

1. Descripción de la tarea

Se dispone de dos imágenes, una que hará de imagen contenedora y otra que hará de estilo. De tal manera que, el contenido de la primera imagen, será transformada según el estilo de la imagen de estilo.

Ilustración 1. Transferencia de estilo

Se han realizado dos versiones, una primera versión adaptada a las nuevas versiones de las diferentes librerías en Google Colab y una segunda haciendo un "downgrade" a las versiones, siendo esta segunda la que mejores resultados obtiene y por tanto, la que se considera solución a este ejercicio.

2. Versión 1 - Adaptada

Lo primero de todo hay que decir que, el código de referencia en el GitHub del profesor de la asignatura [1], contiene errores de versiones de Tensorflow y demás, ya que Google Colab cambió sus versies recientemente.

Sin embargo, en la página de Keras [2] existe un ejemplo de transferencia de estilo, que ha sido la guía de este trabajo. La idea en la página de Keras es la misma, realizada de manera diferente al GitHub.

Lo primero que utiliza es una función para pre procesar la imagen(preprocess_image). Este método básicamente se encarga de leer la imagen de un fichero, modificarle el tamaño al tamaño deseado y convertirla a un tensor. También posee el método inverso (deprocess_image), convertir un tensor en una imagen.

El método para calcular la matriz de gramm, la función de perdida en el estilo y la función de perdida en el contenido son similares en la página y en el GitHub, solamente tienen unas ligeras modificaciones solventando el problema de versiones, lo mismo ocurre para la función de pérdida total.

Después, utiliza una VGG19 como modelo pre entrenado y, por tanto, extractor de características.

Para la perdida de estilo, se utilizarán las primeras convoluciones de cada boque y para la perdida de contenido la 2 convolución del bloque 2.

Después, en la función de calcular la pérdida, una de las diferencias entre ambos es la inicialización de la perdida. Ahora, no se puede inicializar con una variable backend, si no que tiene que inicializarse con un tensor con todos los elementos a 0(tf.zeros(shape()). Esta función de perdida tiene en cuenta la perdida de estilo, de contenido y la variación total.

Después, es necesaria una Tensorflow function para calcular la perdida y calcular los gradientes, esta función es la llamada compute_loss_and_grads y necesita de la anotación @tf.function.

Finalmente, realiza un bucle de entrenamiento donde cada 100 iteraciones va sacando una imagen. Se han probado dos tipos de optimizador de gradiente, SGD y Adam.

Se han usado estas dos imágenes, la primera como imagen de contenido y la segunda como imagen de estilo:

Ilustración 2. Imagen de contenido

Ilustración 3. Imagen de estilo

El primer experimento ha consistido en darles el mismo peso a todo:

- total_variation_weight = 10
- style_weight = 10
- content_weight = 10

Tras 1000 iteraciones se ha obtenido el siguiente resultado:

Ilustración 4. Imagen mismos pesos

Como era de esperar, la imagen prácticamente no ha cambiado, se nota un poco el estilo de la imagen de estilo, sobre todo en el cielo, pero prácticamente es inapreciable.

El segundo experimento ha sido darle un peso mucho mayor al estilo:

- total_variation_weight = 1
- style_weight = 0.05
- content_weight = 50

Ilustración 5. Imagen contenido 1000 veces mayor

Finalmente, se ha probado a darle mucho mas peso al contenido que al estilo:

- total_variation_weight = 1
- style_weight = 50
- content_weight = 0.05

Ilustración 6. Imagen estilo 1000 veces mayor

Como se puede ver, la imagen de estilo aparece en la imagen, pero más que aplicarle el estilo, la imagen se va acoplando a la imagen original.

El fichero con este modelo es el llamado “styletransfer_adaptado.ipynb”

3. Versión 2 - Downgrade

En este punto, se ha “corregido” la versión del GitHub [1], ya que contenía algún error debido al cambio de versiones de Tensorflow en Google Colab. Esto se ha realizado gracias a la ayuda del usuario GPoleto27 en un pull request al GitHub original [3].

En este caso, se ha creado un evaluador que se encarga de evaluar tanto la pérdida como los gradientes. Después, realiza un bucle para minimizar la pérdida del evaluador.

Las modificaciones necesarias con respecto al GitHub de referencia [1] son las siguientes:

```
loss+=(style_weight/len(feature_layers))*sl
```

Ilustración 7. Función errónea

```
loss = loss + (style_weight/len(feature_layers))*sl
```

Ilustración 8. Función corregida

El operador += no esta permitido, por lo que es necesario separar la suma de esta manera y con estas únicas modificaciones, el notebook funciona correctamente.

Básicamente, se está ejecutando un notebook con las versiones de Tensorflow, scipy y demás anteriores en Google Colab, por lo que no es solucionar el error como tal, ya que se ha hecho un “downgrade”.

La metodología es la misma que en el apartado anterior, se quiere minimizar una función de perdida que contiene perdida de estilo, contenido y variación total.

Finalmente, se han realizado los siguientes experimentos, todos ellos tras 10 iteraciones:

- Experimento 1:
    - total_variation_weight = 1
    - style_weight = 50
    - content_weight = 0.05

Ilustración 9. Imagen estilo 1000 veces mayor

A diferencia de la versión anterior, esta es una combinación mucho mas limpia y queda una imagen mucho más bonita.

- Experimento 2:
    - total_variation_weight = 10
    - style_weight = 10
    - content_weight = 10

Ilustración 10. Imagen pesos iguales

Al igual que pasaba en la versión anterior, la imagen resultante es prácticamente igual.

- Experimento 3:
    - total_variation_weight = 1
    - style_weight = 0.05
    - content_weight = 50

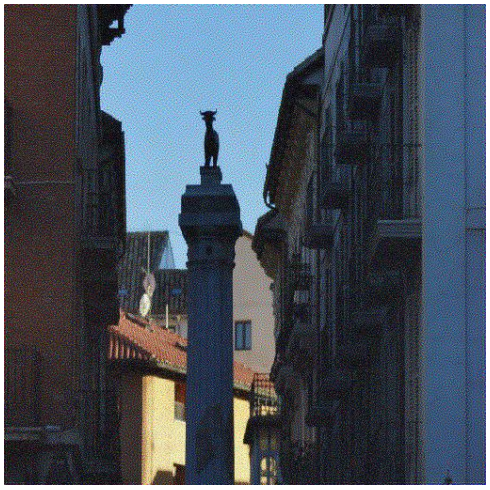

Ilustración 11. Imagen contenido 1000 veces mayor

La imagen es muy parecida a la original, pero se aprecia un poco una influencia del estilo, la imagen esta algo difuminada.

- Experimento 4:
    - total_variation_weight = 1
    - style_weight = 0.05
    - content_weight = 5

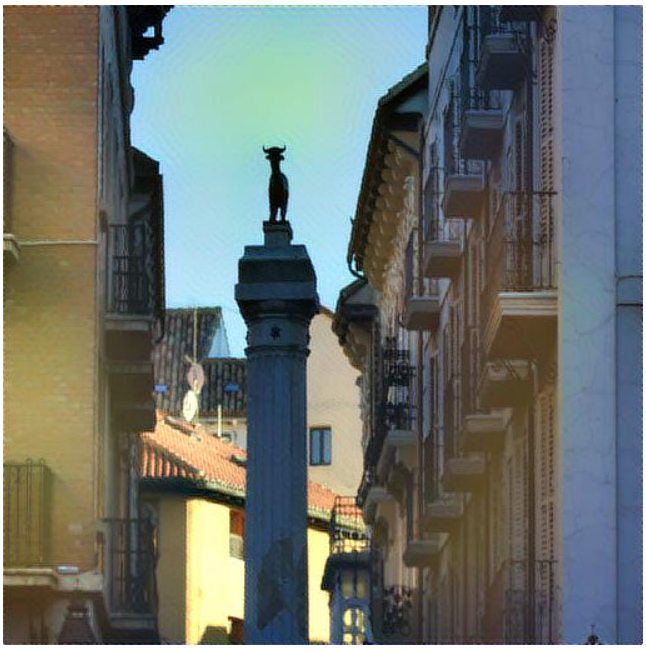

Ilustración 12. Imagen estilo 100 veces mayor

La imagen resultante tiene influencia de la imagen de estilo, pero no tan exagerada como en el caso anterior.

El fichero con este modelo es el llamado “styletransfer_downgrade.ipynb”

4. Conclusiones

Se han presentado dos versiones de transferencia de estilo en Keras, la versión 1 donde se ha adaptado el algoritmo a las nuevas versiones de Google Colab y una segunda donde se han mantenido las viejas versiones de Google Colab.

Viendo las imágenes generadas, se ve claramente que el mejor algoritmo es el presentado en la versión 2, con el “downgrade” y, por tanto, sería el propuesto como solución a este ejercicio.

Se ha visto una manera diferente de calcular una función de perdida a las vistas hasta este momento, ya que es una función dependiente de varios factores y además cada uno de los factores tiene un peso, que influirá más o menos en el resultado final.

5. Bibliografía

[1]. R. Paredes Palacios, “Computer vision lab.” https://github.com/RParedesPalacios/ComputerVisionLab , 2020.

[2]. F. Chollet, “Neural style transfer,”Keras, 2020. https://keras.io/examples/generative/neural_style_transfer/

[3]. Data Science Group, IIT Roorkee, “Neural-Style-Transfer” https://github.com/dsgiitr/Neural-Style-Transfer/pull/3 , 2021.