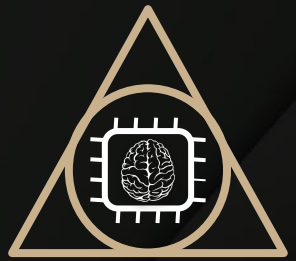
DaVincAI

Style transfer



Equipo 9

D. Valverde

Documentación

J. Acevedo

Programación

R. Cala

Supervisión

Temas a tratar

1. Introducción al proyecto
2. Requisitos del alto nivel
3. Objetivos medibles del proyecto y criterio de éxito asociado
4. Riesgos del Proyecto
5. Problemas encontrados
6. Requisitos y Objetivos alcanzados
7. Descripción técnica del funcionamiento e implementación
8. Descripción del funcionamiento
9. Prueba de ejecución y resultados

01
Introducción

Introducción

“When you take technology and mix it with art, you always come up with something innovative”

02
Requisitos

Requisitos

Transferencia de estilos	Velocidad
Interfaz de usuario	Accesibilidad

03

Objetivos medibles y Criterios de éxito

Objetivos medibles

Calidad

Velocidad

Magnitud

Criterios de éxito

Transferencia de estilos

Resultado agradable

Filtros predefinidos

Interfaz de usuario

Imágenes de Internet

04

Riesgos del proyecto

IMP: valor de impacto sobre 10
%: valor de probabilidad sobre 100

Riesgos del proyecto

IMP : 7
% : 20

Imágenes resultantes demasiado alteradas o irreconocibles

IMP : 3
% : 40

Tiempo de ejecución demasiado elevado

IMP : 5
% : 20

Resultados estéticamente desacertados

IMP : 9
% : 5

Imposibilidad de extraer los estilos de una imagen

IMP : 6
% : 15

Número limitado de imágenes a las que se les pueda aplicar el filtro

05

Problemas encontrados

Problemas encontrados

1. Cada ejecución requería de mucho tiempo.
2. La calidad del proyecto depende de diversos parámetros.
3. Adaptar la resolución de las imágenes en calidad/tiempo.
4. Adaptación al lenguaje Python, dado el poco uso anteriormente.

06

Requisitos y objetivos alcanzados

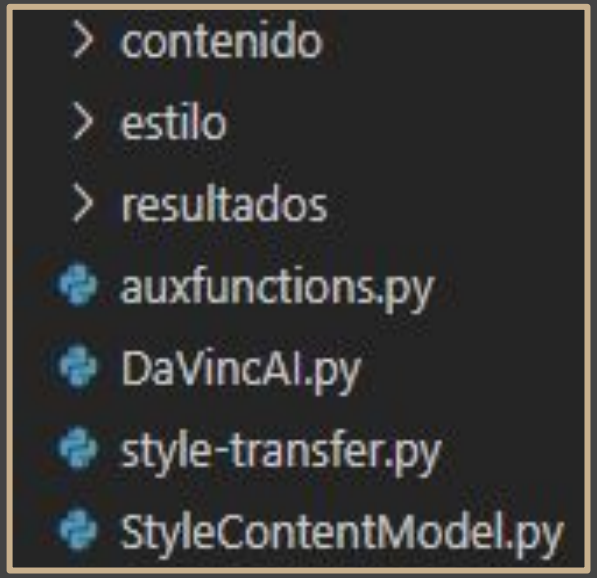
Requisitos y objetivos alcanzados

- Se tiene un programa funcional de transferencia de estilos
- Se han definido diferentes filtros por defecto
- Se ha conseguido generar *collages* con los resultados obtenidos
- Se permite tomar imágenes directamente de Internet
- Se ha implementado un menú en terminal que resulta amigable al usuario

07

Descripción técnica

Estructura de directorios



- Imágenes de contenido
- Imágenes de estilo
- Imágenes resultantes
- Fichero contenedor de funciones auxiliares
- Fichero con programa principal
- Fichero con gran parte de la carga de funciones implementadas
- Fichero con la definición del modelo de transferencia de estilos

StyleContentModel.py

```python
class StyleContentModel(tf.keras.models.Model):
  def __init__(self, style_layers, content_layers):
    super(StyleContentModel, self).__init__()
    self.vgg = vgg_layers(style_layers + content_layers)
    self.style_layers = style_layers
    self.content_layers = content_layers
    self.num_style_layers = len(style_layers)
    self.vgg.trainable = False
```

```python
def call(self, inputs):
  inputs = inputs*255.0
  preprocessed_input = tf.keras.applications.vgg19.preprocess_input(
      inputs)
  outputs = self.vgg(preprocessed_input)
  style_outputs, content_outputs = (outputs[:self.num_style_layers],
                                    outputs[self.num_style_layers:])

  style_outputs = [gram_matrix(style_output)
                   for style_output in style_outputs]

  content_dict = {content_name: value
                  for content_name, value
                  in zip(self.content_layers, content_outputs)}

  style_dict = {style_name: value
                for style_name, value
                in zip(self.style_layers, style_outputs)}

  return {'content': content_dict, 'style': style_dict}
```

Style-Transfer.py

```python
@tf.function()
def train_step(image, extractor, style_targets,
               style_weight, content_weight, num_style_layers, num_content_layers, content_targets, opt):
  # Una desventaja de esta implementación básica es que produce muchos artefactos de alta frecuencia.
  # Disminuya estos usando un término de regularización explícito en los componentes de alta frecuencia de la imagen.
  total_variation_weight = 30
  with tf.GradientTape() as tape:
    outputs = extractor(image)
    loss = style_content_loss(outputs, style_targets,
                              style_weight, content_weight, num_style_layers, num_content_layers, content_targets)
    loss += total_variation_weight*tf.image.total_variation(image)

  grad = tape.gradient(loss, image)
  opt.apply_gradients([(grad, image)])
  image.assign(clip_0_1(image))
  return loss
```

08

Descripción del funcionamiento

Menú de ejecución

```
---- Image Style Transfer ----
---- Content Image ----
1. Take from the Internet
2. Take from Local
Your choice:
```

```
---- Style Image ----
1. Take from the Internet
2. Take from Local
3. Take a Predefined Style
4. Use a chain of different Styles
Your choice:
```

```
---- Choose a Style ----
1. Abstract Style
2. El Grito Style
3. Gold Style
4. Guernica Style
5. Mona Lisa Style
6. Psychedelic Style
7. Puntillism Style
8. Waves Style
9. Yellow Thunder Style
Your choice:
```

```
---- Collage ----
Generate a Collage with the Result [y/n]:
```

09

Prueba de ejecución **en live streaming**

Gracias

¿Alguna pregunta?

David - alu0101100296@ull.edu.es
Rafa - alu0101121901@ull.edu.es
Jorge - alu0101123622@ull.edu.es

GitHub Repo:
https://github.com/daviddvg7/DaVincAI.git