10-kv-test-implementation

November 19, 2024

1 Demonstration Overview: Models in Action

1.1 Setting everything up

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import sys

sys.path.append("../../")
```

[3]:
```
from PIL import Image
import numpy as np
```

[4]:
```
from image_processing.invoker import Invoker

request_dict = {}
invoker = Invoker()
```

```
GPU not available. Using CPU instead.
```

[5]:
```
def display_images_side_by_side(image1, image2):
    """
    Displays two PIL images side by side in a Jupyter notebook.

    Args:
    - image1: The first PIL image.
    - image2: The second PIL image.
    """
    total_width = image1.width + image2.width
    max_height = max(image1.height, image2.height)

    new_image = Image.new("RGB", (total_width, max_height))

    new_image.paste(image1, (0, 0))
    new_image.paste(image2, (image1.width, 0))

    display(new_image)
```

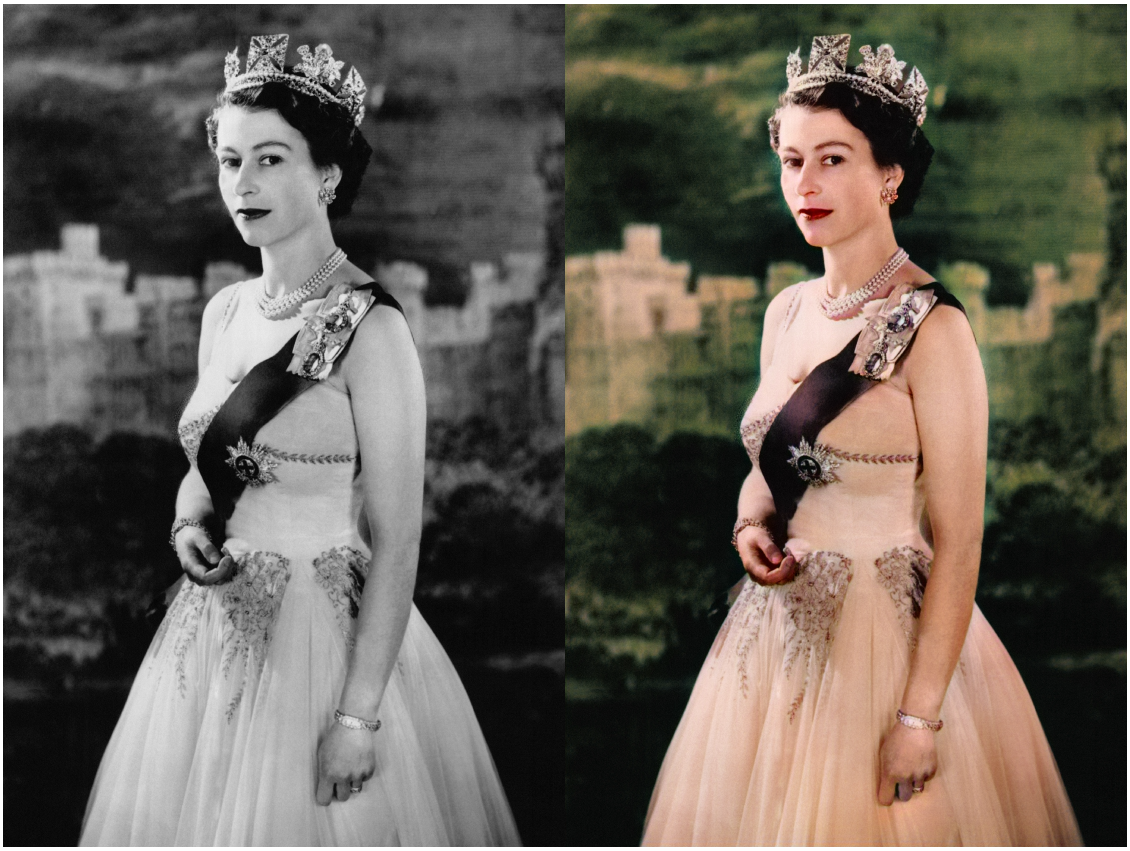
1.2 Image Colorization

[6]:
```
images = []
for i in range(1, 4):
    images.append(Image.open(f"../example_images/colorize_image/image{i}.jpg").
 ↪convert('RGB'))
```

[7]:
```
request_dict["type"] = "colorize_image"
request_dict["params"] = {}
```

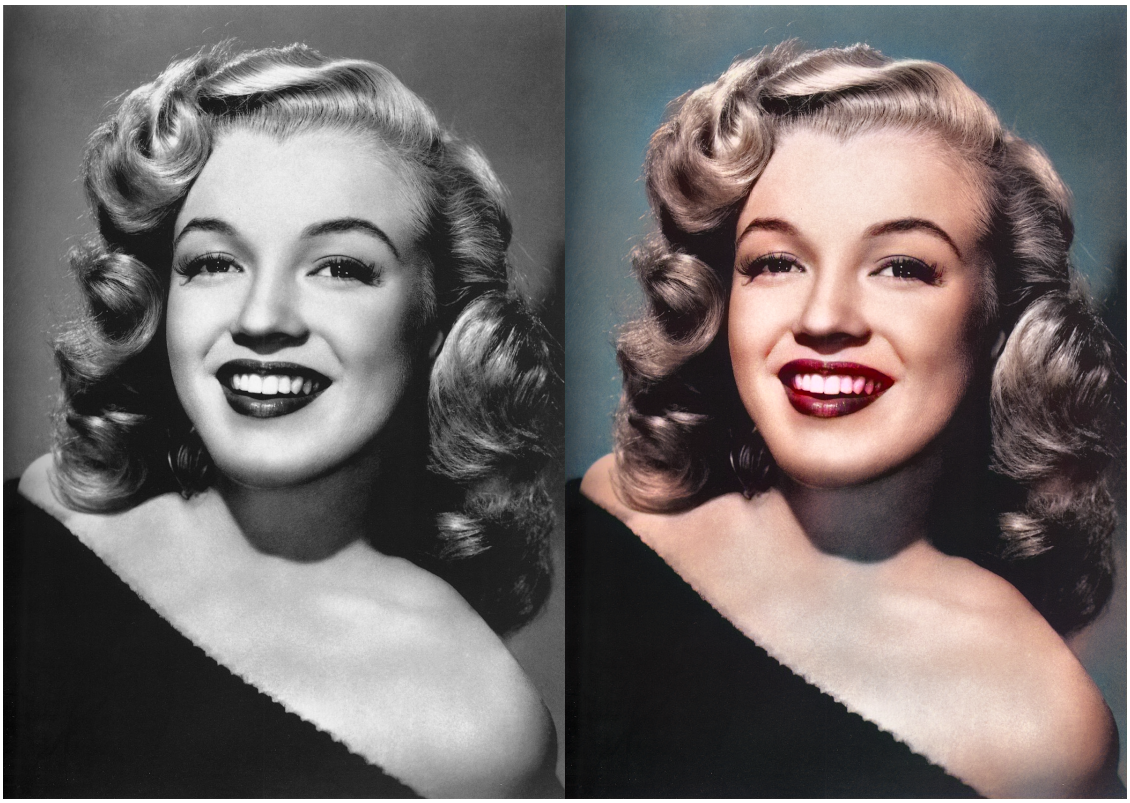
[8]:
```
colorization_results = []
for image in images:
    request_dict["params"]["image"] = image
    colorization_results.append(invoker.process_request(request_dict))
```

[9]:
```
display_images_side_by_side(images[0], colorization_results[0])
```



[10]:
```
display_images_side_by_side(images[1], colorization_results[1])
```

[11]:
```
display_images_side_by_side(images[2], colorization_results[2])
```



1.3 Object Removal

[12]:
```
images, masks = [], []
for i in range(1, 5):
    images.append(Image.open(f"../example_images/object_deletion/image{i}.jpg").
 ↪convert('RGB'))
    masks.append(Image.open(f"../example_images/object_deletion/mask{i}.jpg").
 ↪convert('L'))
```

[13]:
```
request_dict["type"] = "delete_object"
request_dict["params"] = {}
```

[14]:
```
object_deletion_results = []
for image, mask in zip(images, masks):
    request_dict["params"]["image"] = image
    request_dict["params"]["mask"] = mask
    object_deletion_results.append(invoker.process_request(request_dict))
```

[15]:
```
display_images_side_by_side(images[0], object_deletion_results[0])
```

[16]:
```
display_images_side_by_side(images[1], object_deletion_results[1])
```

[17]:
```
display_images_side_by_side(images[2], object_deletion_results[2])
```

1.4 Style Transfer

[18]:
```
styles = ["vangogh"]*3
```

[19]:
```
images = []
for i in range(1, 4):
```

```
    images.append(Image.open(f"../example_images/transfer_style/image{i}.jpg").
↪convert('RGB'))
```

[20]:
```
request_dict["type"] = "transfer_style"
request_dict["params"] = {}
```

[21]:
```
vangogh_results = []
for image, style in zip(images, styles):
    request_dict["params"]["image"] = image
    request_dict["params"]["style"] = style
    vangogh_results.append(invoker.process_request(request_dict))
```

[22]:
```
vangogh_results[0]
```

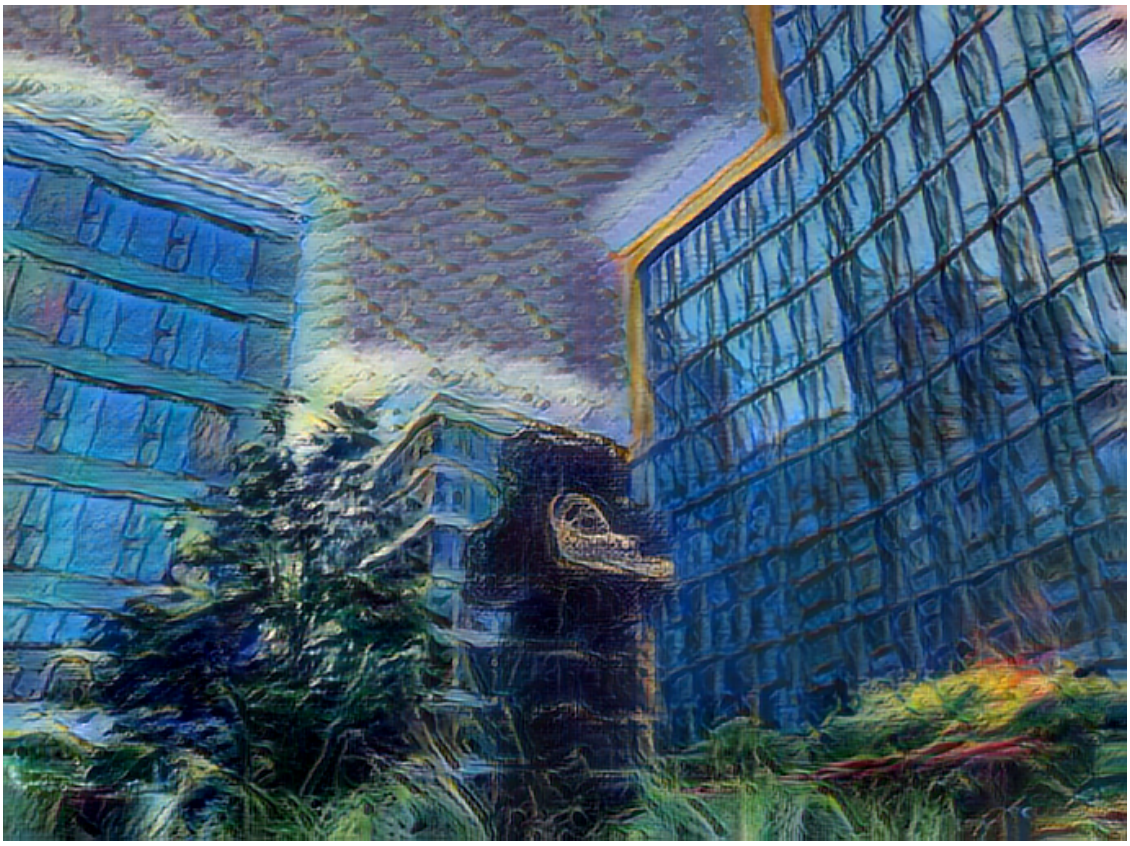
[22]:

[23]:
```
vangogh_results[1]
```

[23]:

[24]:
```
vangogh_results[2]
```

[24]:



[25]:
```
styles = ["matisse", "sketch", "mondrian"]
```

[29]:
```
styles_results = []
for image, mask, style in zip(images, masks, styles):
    request_dict["params"]["image"] = image
    request_dict["params"]["style"] = style
    styles_results.append(invoker.process_request(request_dict))
```

[30]:
```
styles_results[0]
```

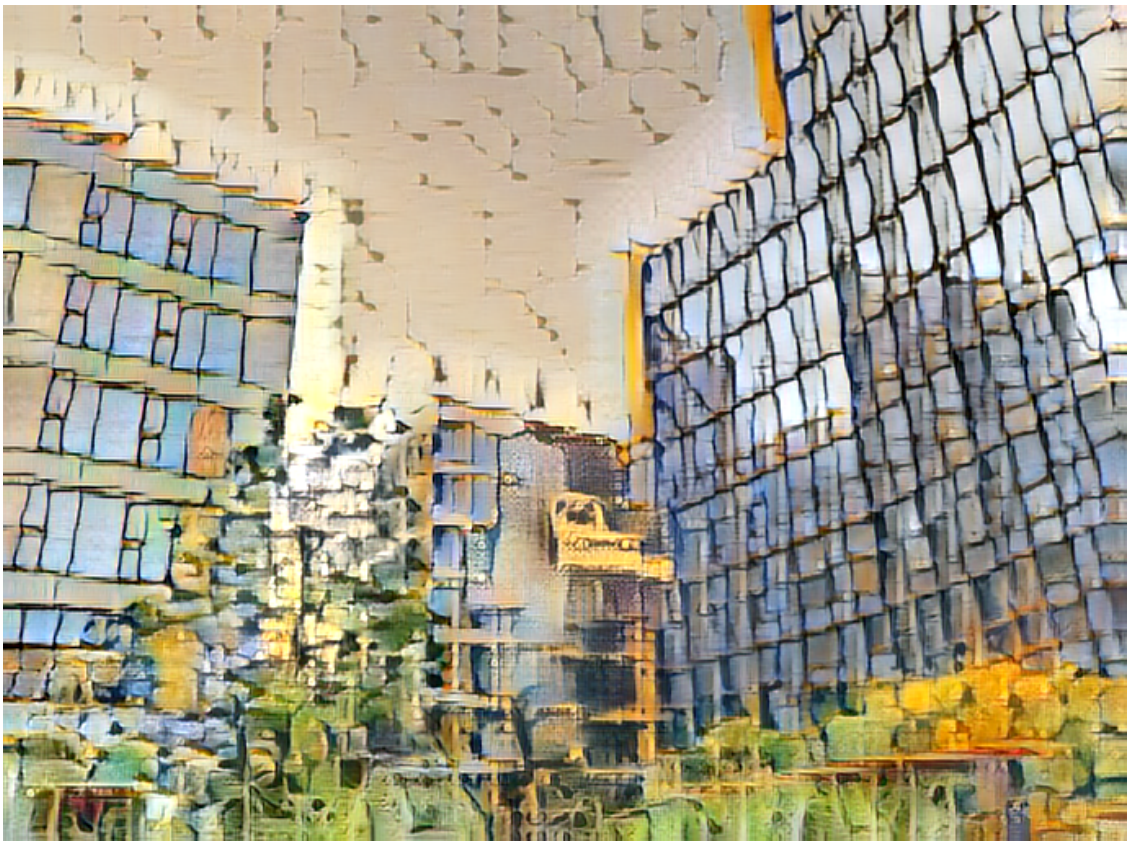
[30]:

[31]:
```
styles_results[1]
```

[31]:

[32]:
```
styles_results[2]
```

[32]:



1.5 Image Upscaling

[41]:
```
images = []
for i in range(1, 4):
    images.append(Image.open(f"../example_images/upscale_image/image{i}.jpg"))
```

[42]:
```
request_dict["type"] = "upscale"
request_dict["params"] = {}
```

[43]:
```
upscaling_results = []
for image in images:
    request_dict["params"]["image"] = image
    upscaling_results.append(invoker.process_request(request_dict))
```

[44]:
```
images[0]
```

[44]:

[45]:
```
upscaling_results[0]
```

[45]:

[46]:
```
images[1]
```

[46]:

[47]:
```
upscaling_results[1]
```

[47]:

[96]:
```
!jupyter nbconvert --to pdf --output models_demonstration.pdf␣
↪10-kv-test-implementation.ipynb
```

```
[NbConvertApp] Converting notebook 10-kv-test-implementation.ipynb to pdf
[NbConvertApp] Support files will be in models_demonstration_files/
[NbConvertApp] Making directory ./models_demonstration_files
[NbConvertApp] Writing 75667 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 72024713 bytes to models_demonstration.pdf
```

[ ]: