Create_and_Art_with_Neural_style_transfer

April 2, 2025

[3]:
```
pip install tensorflow numpy matplotlib PILLOW
```

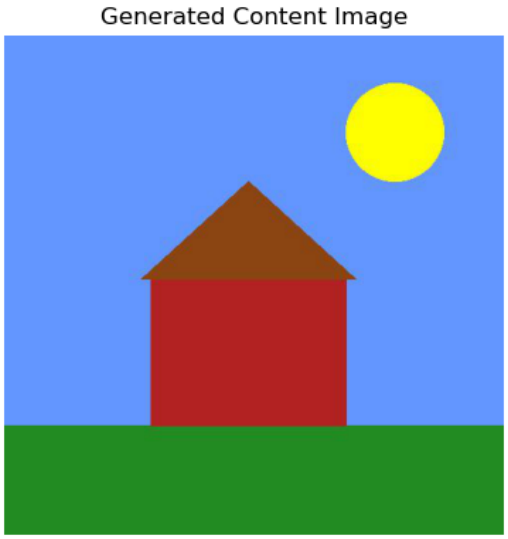
```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tensorflow in
c:\users\student\appdata\roaming\python\python311\site-packages (2.16.1)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (1.24.3)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-
packages (3.7.1)
Requirement already satisfied: PILLOW in c:\programdata\anaconda3\lib\site-
packages (9.4.0)
Requirement already satisfied: tensorflow-intel==2.16.1 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow) (2.16.1)
Requirement already satisfied: absl-py>=1.0.0 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.3.1 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (0.3.2)
```

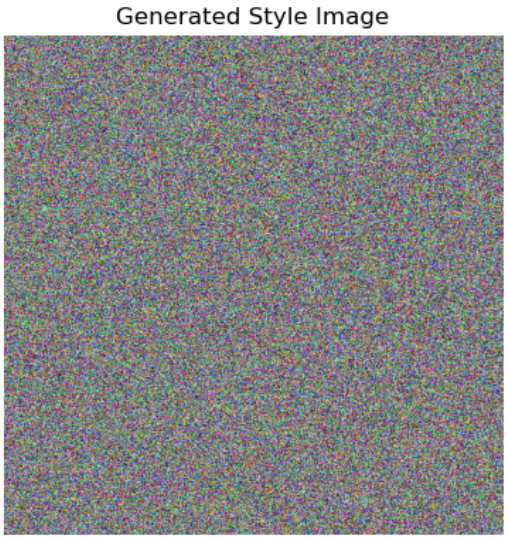
```
Requirement already satisfied: opt-einsum>=2.3.2 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-
packages (from tensorflow-intel==2.16.1->tensorflow) (23.0)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in
c:\programdata\anaconda3\lib\site-packages (from tensorflow-
intel==2.16.1->tensorflow) (2.29.0)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-
packages (from tensorflow-intel==2.16.1->tensorflow) (67.8.0)
Requirement already satisfied: six>=1.12.0 in c:\programdata\anaconda3\lib\site-
packages (from tensorflow-intel==2.16.1->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
c:\programdata\anaconda3\lib\site-packages (from tensorflow-
intel==2.16.1->tensorflow) (4.6.3)
Requirement already satisfied: wrapt>=1.11.0 in
c:\programdata\anaconda3\lib\site-packages (from tensorflow-
intel==2.16.1->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (1.63.0)
Requirement already satisfied: tensorboard<2.17,>=2.16 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (2.16.2)
Requirement already satisfied: keras>=3.0.0 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (3.3.3)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (0.31.0)
Requirement already satisfied: contourpy>=1.0.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
```

```
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
c:\programdata\anaconda3\lib\site-packages (from astunparse>=1.6.0->tensorflow-
intel==2.16.1->tensorflow) (0.38.4)
Requirement already satisfied: rich in
c:\users\student\appdata\roaming\python\python311\site-packages (from
keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (13.7.1)
Requirement already satisfied: namex in
c:\users\student\appdata\roaming\python\python311\site-packages (from
keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (0.0.8)
Requirement already satisfied: optree in
c:\users\student\appdata\roaming\python\python311\site-packages (from
keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (0.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\programdata\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
c:\programdata\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\programdata\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in
c:\programdata\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (2023.5.7)
Requirement already satisfied: markdown>=2.6.8 in
c:\programdata\anaconda3\lib\site-packages (from
tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
c:\programdata\anaconda3\lib\site-packages (from
tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in
c:\programdata\anaconda3\lib\site-packages (from
werkzeug>=1.0.1->tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow)
(2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in
c:\programdata\anaconda3\lib\site-packages (from rich->keras>=3.0.0->tensorflow-
intel==2.16.1->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
c:\programdata\anaconda3\lib\site-packages (from rich->keras>=3.0.0->tensorflow-
intel==2.16.1->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\programdata\anaconda3\lib\site-
packages (from markdown-it-py>=2.2.0->rich->keras>=3.0.0->tensorflow-
intel==2.16.1->tensorflow) (0.1.0)
Note: you may need to restart the kernel to use updated packages.
```

[5]:
```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw

# Function to create a simple content image
def create_content_image():
    img = Image.new("RGB", (512, 512), color=(100, 150, 255))  # Sky blue␣
 ↪background
    draw = ImageDraw.Draw(img)

    # Draw a sun
    draw.ellipse((350, 50, 450, 150), fill=(255, 255, 0))  # Yellow sun

    # Draw some grass
    draw.rectangle((0, 400, 512, 512), fill=(34, 139, 34))  # Green grass

    # Draw a house
    draw.rectangle((150, 250, 350, 400), fill=(178, 34, 34))  # Red house
    draw.polygon([(140, 250), (250, 150), (360, 250)], fill=(139, 69, 19))  #␣
 ↪Roof

    img.save("content.jpg")
    print("Content image saved as 'content.jpg'.")

# Function to create an abstract style image
def create_style_image():
    img = np.random.rand(512, 512, 3) * 255  # Random noise
    img = img.astype(np.uint8)
    img = Image.fromarray(img)
    img.save("style.jpg")
    print("Style image saved as 'style.jpg'.")

# Generate and save images
create_content_image()
create_style_image()

# Show generated images
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
ax1.imshow(Image.open("content.jpg"))
ax1.set_title("Generated Content Image")
ax1.axis("off")

ax2.imshow(Image.open("style.jpg"))
ax2.set_title("Generated Style Image")
ax2.axis("off")

plt.show()
```

```
Content image saved as 'content.jpg'.
Style image saved as 'style.jpg'.
```



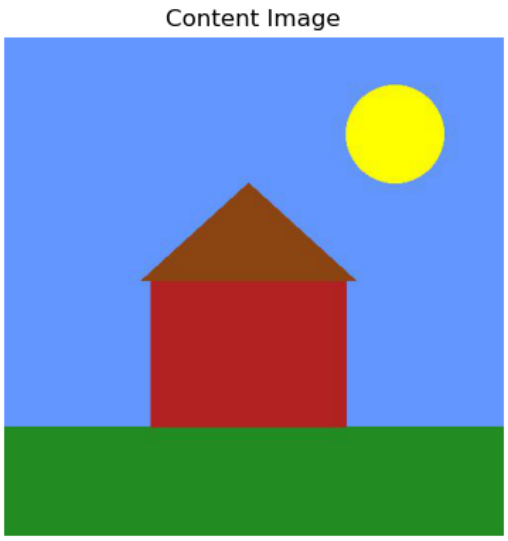


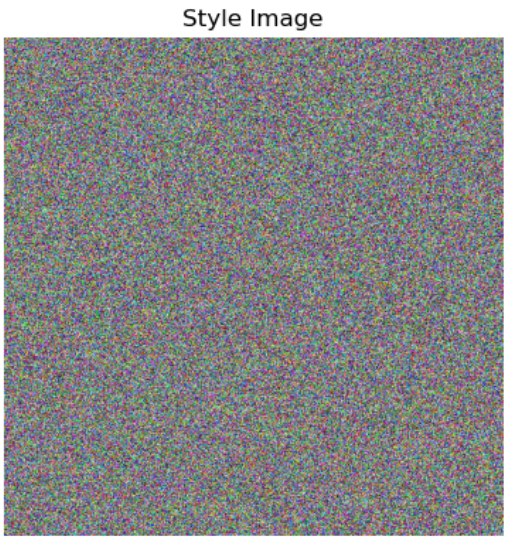
[ ]:
```python
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import PIL.Image
from tensorflow.keras.applications import vgg19

# Function to load and preprocess the image
def load_image(image_path, max_dim=512):
    img = PIL.Image.open(image_path)
    img = img.resize((max_dim, max_dim), PIL.Image.LANCZOS)
    img = np.array(img, dtype=np.float32)
    img = np.expand_dims(img, axis=0)
    img = tf.keras.applications.vgg19.preprocess_input(img)
    return img

# Function to deprocess image (convert back to RGB format)
def deprocess_image(img):
    img = img.reshape((img.shape[1], img.shape[2], 3))
    img[:, :, 0] += 103.939
    img[:, :, 1] += 116.779
    img[:, :, 2] += 123.68
    img = img[:, :, ::-1]  # Convert BGR to RGB
    img = np.clip(img, 0, 255).astype("uint8")
    return img

# Load content and style images
```

```
content_path = "content.jpg"
style_path = "style.jpg"

content_image = load_image(content_path)
style_image = load_image(style_path)

# Display the images
def show_images(content, style):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
    ax1.imshow(deprocess_image(content))
    ax1.set_title("Content Image")
    ax1.axis("off")

    ax2.imshow(deprocess_image(style))
    ax2.set_title("Style Image")
    ax2.axis("off")

    plt.show()

show_images(content_image, style_image)

# Define a VGG19 model for feature extraction
def get_model():
    vgg = vgg19.VGG19(weights="imagenet", include_top=False)
    vgg.trainable = False
    style_layers = ['block1_conv1', 'block2_conv1', 'block3_conv1',␣
 ↪'block4_conv1', 'block5_conv1']
    content_layers = ['block4_conv2']
    model_outputs = [vgg.get_layer(name).output for name in (content_layers +␣
 ↪style_layers)]
    return tf.keras.Model([vgg.input], model_outputs)

# Extract features
model = get_model()

# Compute content loss
def compute_content_loss(content, generated):
    return tf.reduce_mean(tf.square(content - generated))

# Compute style loss using Gram Matrix
def gram_matrix(tensor):
    channels = int(tensor.shape[-1])
    a = tf.reshape(tensor, [-1, channels])
    gram = tf.matmul(a, a, transpose_a=True)
    return gram

def compute_style_loss(style, generated):
```

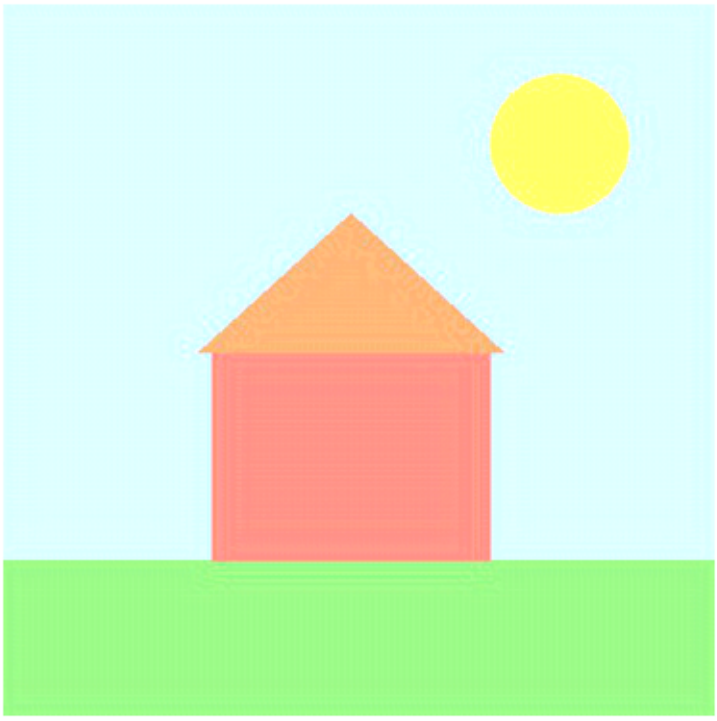
```
    return tf.reduce_mean(tf.square(gram_matrix(style) -␣
 ↪gram_matrix(generated)))

# Define total loss function
def compute_total_loss(model, content_image, style_image, generated_image,␣
 ↪content_weight=1e4, style_weight=1e-2):
    outputs = model(generated_image)
    content_outputs = outputs[:1]
    style_outputs = outputs[1:]

    content_loss = compute_content_loss(content_outputs[0],␣
 ↪model(content_image)[:1][0])
    style_loss = sum([compute_style_loss(style, gen) for style, gen in␣
 ↪zip(model(style_image)[1:], style_outputs)])

    total_loss = content_weight * content_loss + style_weight * style_loss
    return total_loss

# Perform Neural Style Transfer using optimization
generated_image = tf.Variable(content_image, dtype=tf.float32)

# Define optimizer
optimizer = tf.optimizers.Adam(learning_rate=5.0)

# Run the optimization
epochs = 500
for i in range(epochs):
    with tf.GradientTape() as tape:
        loss = compute_total_loss(model, content_image, style_image,␣
 ↪generated_image)

    gradients = tape.gradient(loss, generated_image)
    optimizer.apply_gradients([(gradients, generated_image)])

    if i % 50 == 0:
        print(f"Iteration {i}: Loss = {loss.numpy()}")
        output_img = deprocess_image(generated_image.numpy())
        plt.imshow(output_img)
        plt.axis("off")
        plt.show()

# Save the final image
output_image = deprocess_image(generated_image.numpy())
PIL.Image.fromarray(output_image).save("styled_output.jpg")

plt.imshow(output_image)
plt.axis("off")
```

```
plt.show()

print("Neural Style Transfer complete! Styled image saved as 'styled_output.
  ↪jpg'.")
```

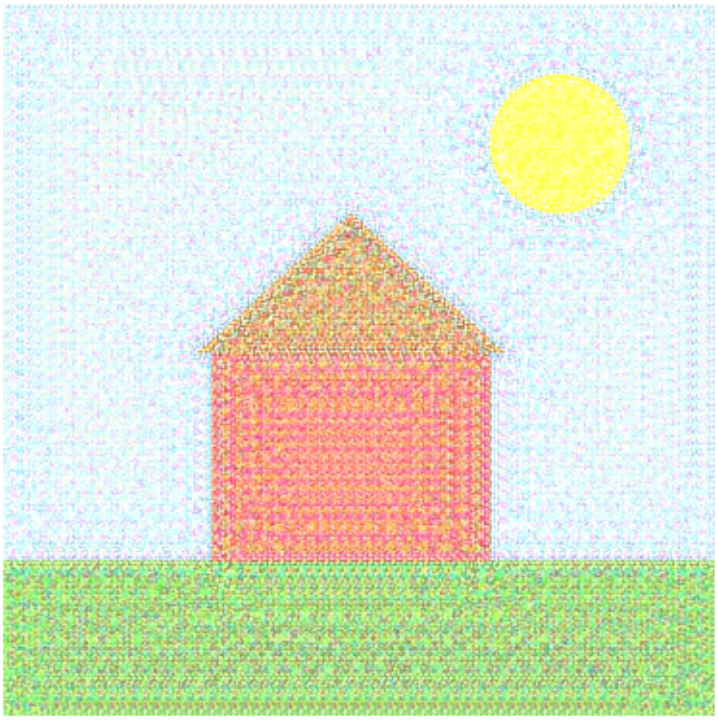




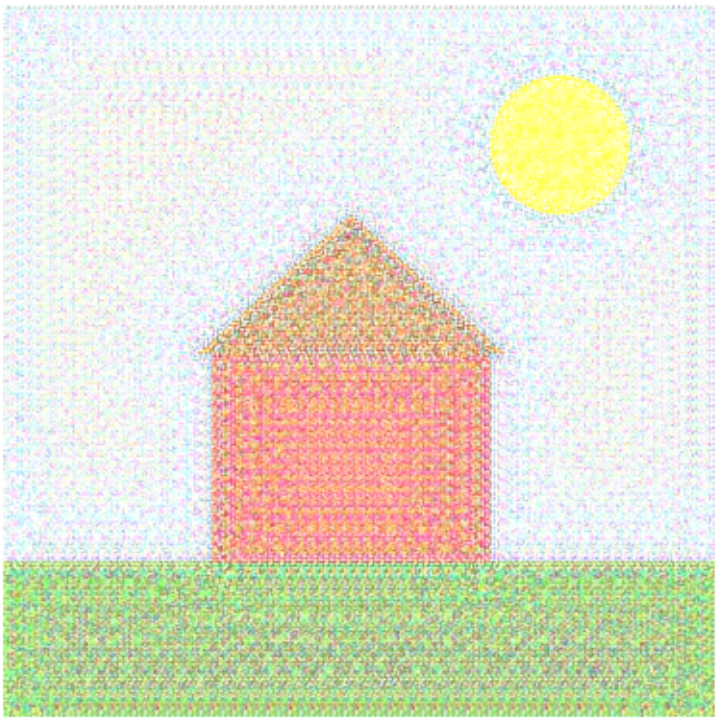

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624              15s
0us/step
Iteration 0: Loss = 8.10528017439785e+17
```



```
Iteration 50: Loss = 1.0067572993032192e+16
```



```
Iteration 100: Loss = 2542223623192576.0
```



[2]:
```
!jupyter nbconvert --to pdf "Create and Art with Neural style transfer on given␣
↪image using deep learning.ipynb"
```

```
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
```

```
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--coalesce-streams
    Coalesce consecutive stdout and stderr outputs into one stream (within each
cell).
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--CoalesceStreamsPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
```

```
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True
--TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the
system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful
for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook',
'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
```

```
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    Overwrite base name use for output files.
                Supports pattern replacements '{notebook_name}'.
    Default: '{notebook_name}'
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each notebook.
To recover
                                    previous default behaviour (outputting to the
current
                                    working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
```

```
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides',
'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX
includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]
```

```
            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.

[NbConvertApp] WARNING | pattern 'Create and Art with Neural style transfer on
given image using deep learning.ipynb' matched no files
```

[ ]: