

X00144631_Ryan_Deering_Applied_AI_and_Deep_Learning_CA2

May 1, 2021

1 Applied Deep Learning and AI CA2 - Ryan Deering X00144631

2 Imports

```
[1]: import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
import glob

from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.datasets.samples_generator import make_classification
from sklearn.utils import shuffle
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.callbacks import ModelCheckpoint

from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import GaussianNoise
from google.colab import drive
drive.mount('/content/gdrive')
plt.rcParams['figure.figsize'] = [5,5]
```

/usr/local/lib/python3.7/dist-packages/scikit_learn/utils/deprecation.py:144:
FutureWarning: The sklearn.datasets.samples_generator module is deprecated in
version 0.22 and will be removed in version 0.24. The corresponding classes /
functions should instead be imported from sklearn.datasets. Anything that cannot
be imported from sklearn.datasets is now part of the private API.

```
warnings.warn(message, FutureWarning)  
Mounted at /content/gdrive
```

3 Image Collection

The image collection process was a fairly simple one for this type of classification problem. Using Google Images, and a Google Chrome extension called ‘Download All Images’, which allows a user to be able to easily download all images displayed within a web page, from their original individual sources.

In this case, I was able to Google Image search every Simpsons character I needed for this classification problem, that being Bart, Homer, Lisa, Maggie & Marge Simpson. With every individual search, I made sure to sort by the highest resolution to get the best possible accuracy and chance of identifying each class. When I downloaded the images, I sorted them into folders depending on classification.

The first step in each class/folder of images was to clear out any thumbnails or unnecessary data. The extension also downloaded images used by Google Images like a search bar icon, or my Google Account picture. It also downloaded thumbnails, which were low-resolution images of the images we need, and wholly unnecessary. These were deleted.

Next was identifying duplicate images and trimming down to 50 images per class, as with a search result on the internet you are likely to get duplicate links or similar ones. This was the case with the images too. I had to delete quite a few duplicate images and settle for the ones with the highest resolution or quality. There was also the case of unsuitable or irrelevant images popping up in our search, which were deleted.

The images were in different formats, so I used a third-party tool online to convert the images to one singular format – png. This gave us some consistency with our images, as they were pulled from different sources and resolutions on the internet. After that, I renamed them all into for example, homer (1), homer (2)... to have a consistent naming scheme for easy access from our code.

Converting all the images into one consistent format, as well as consistent naming scheme enabled us to import the images we sourced much easier as we can just use a wildcard and the file format to import our images into the various arrays in our Notebook.

4 Test

```
[ ]: myImage = Image.open("Homer/homer.png")  
grayscale = myImage.convert('L')  
  
plt.imshow(myImage)  
plt.show()  
  
plt.imshow(grayscale,cmap = 'gray')  
plt.show()
```



5 Image Pre-Processing

```
[ ]: homerImages = []

for filename in glob.glob('Homer/*.png'):
    im=Image.open(filename)
    homerImages.append(im)

bartImages = []

for filename in glob.glob('Bart/*.png'):
    im=Image.open(filename)
```

```

bartImages.append(im)

lisaImages = []

for filename in glob.glob('Lisa/*.png'):
    im=Image.open(filename)
    lisaImages.append(im)

maggieImages = []

for filename in glob.glob('Maggie/*.png'):
    im=Image.open(filename)
    maggieImages.append(im)

margeImages = []

for filename in glob.glob('Marge/*.png'):
    im=Image.open(filename)
    margeImages.append(im)

```

```

[ ]: print("Homer")

for item in homerImages:
    print(item)
    plt.imshow(item)
    plt.show()

print("Bart")

for item in bartImages:
    print(item)
    plt.imshow(item)
    plt.show()

print("Lisa")

for item in lisaImages:
    print(item)
    plt.imshow(item)
    plt.show()

print("Maggie")

for item in maggieImages:
    print(item)
    plt.imshow(item)
    plt.show()

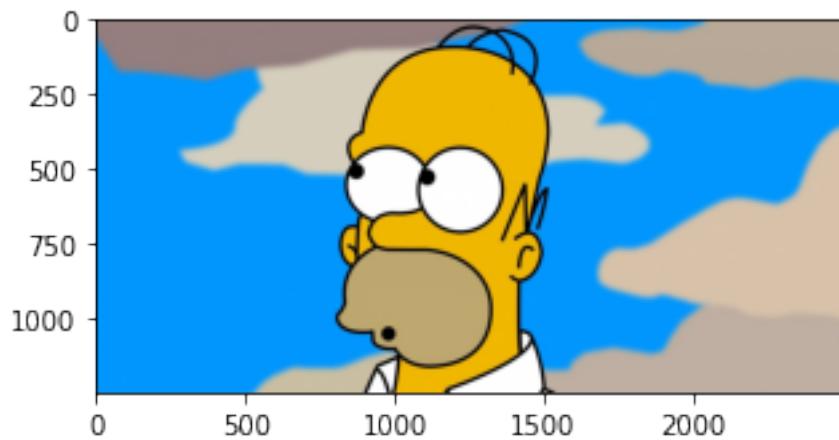
```

```
print("Marge")

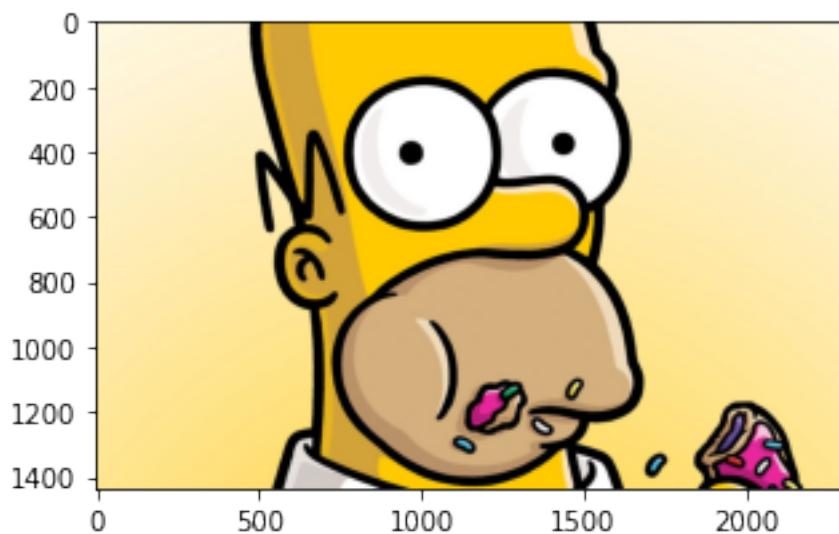
for item in margeImages:
    print(item)
    plt.imshow(item)
    plt.show()
```

Homer

<PIL.PngImagePlugin.PngImageFile image mode=RGB size=2500x1250 at 0x7FB119B84CD0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=2296x1435 at 0x7FB119B84610>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=2003x999 at 0x7FB119B84650>



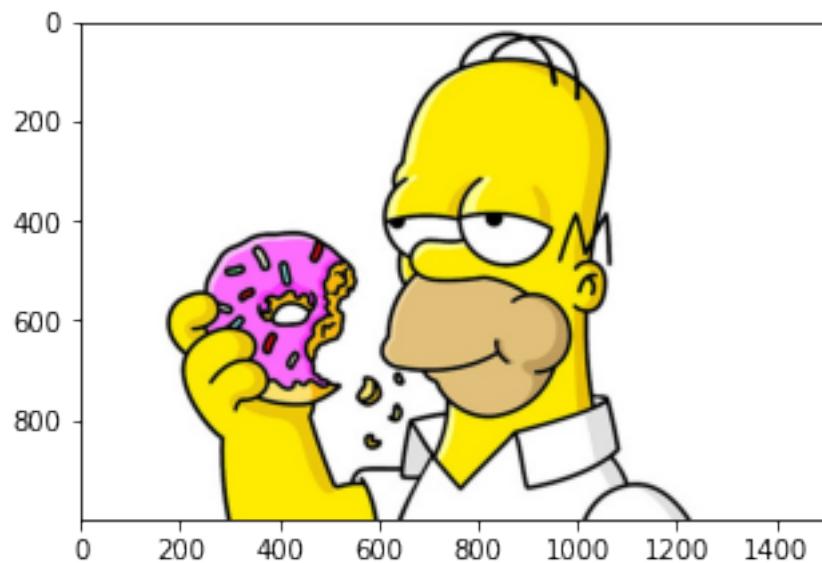
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=576x432 at 0x7FB119B84B90>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1765x1149 at 0x7FB119B84D90>



```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1500x1000 at 0x7FB119B84DD0>
```



```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=775x522 at 0x7FB119B849D0>
```



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=848x480 at 0x7FB119B84BD0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1200x900 at 0x7FB119B84590>



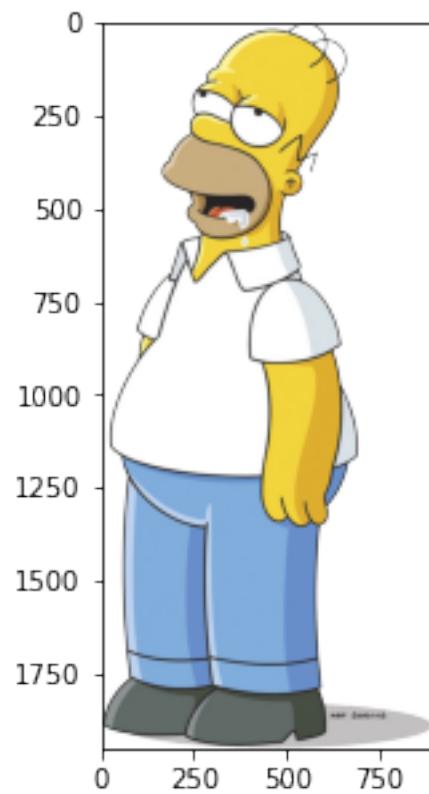
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=640x480 at 0x7FB119B84C10>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=4800x2700 at 0x7FB119B84AD0>



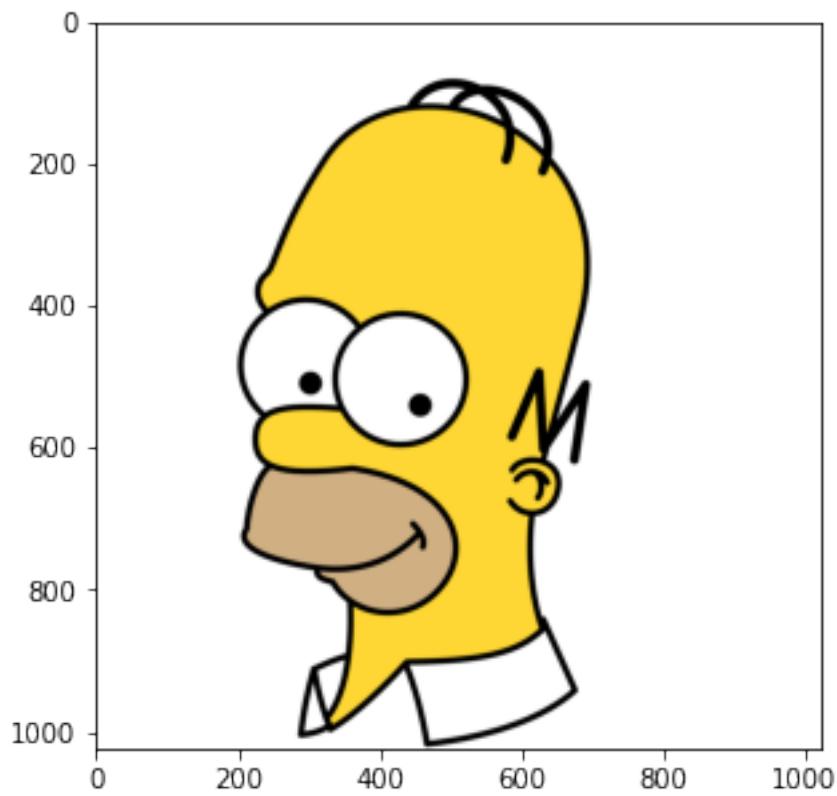
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=890x1951 at 0x7FB119B84290>
```



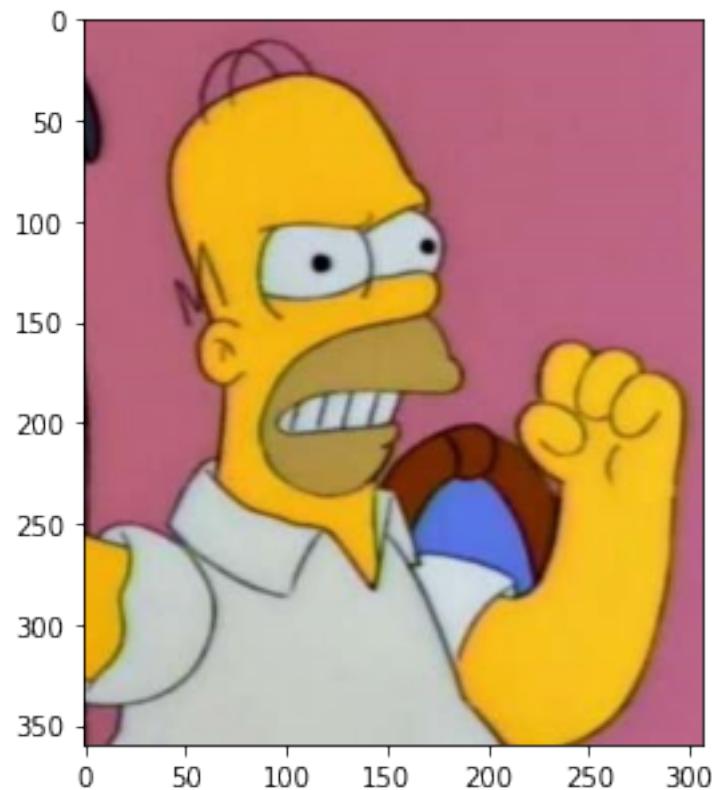
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=640x480 at 0x7FB119B84690>
```



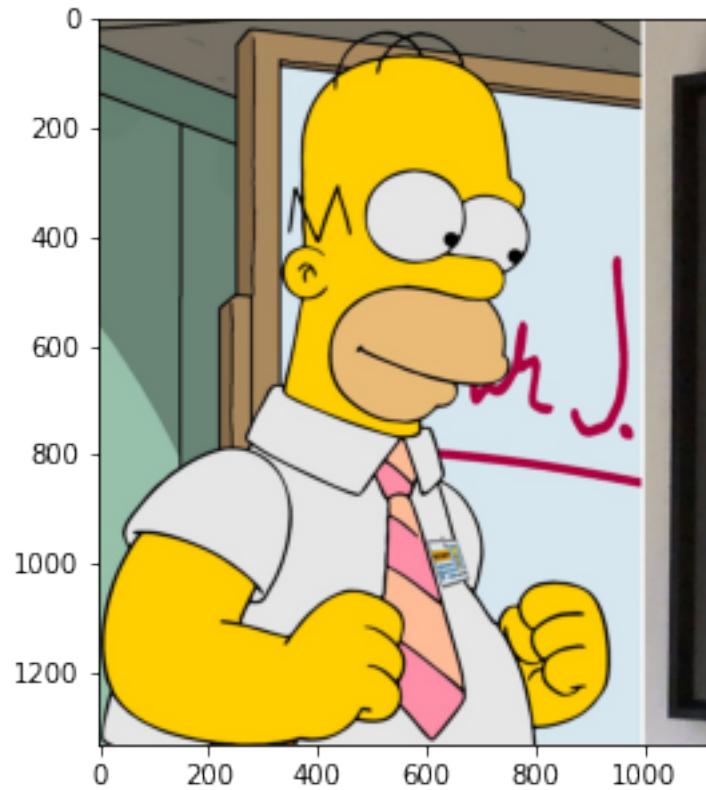
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1024x1024 at  
0x7FB119B84850>
```



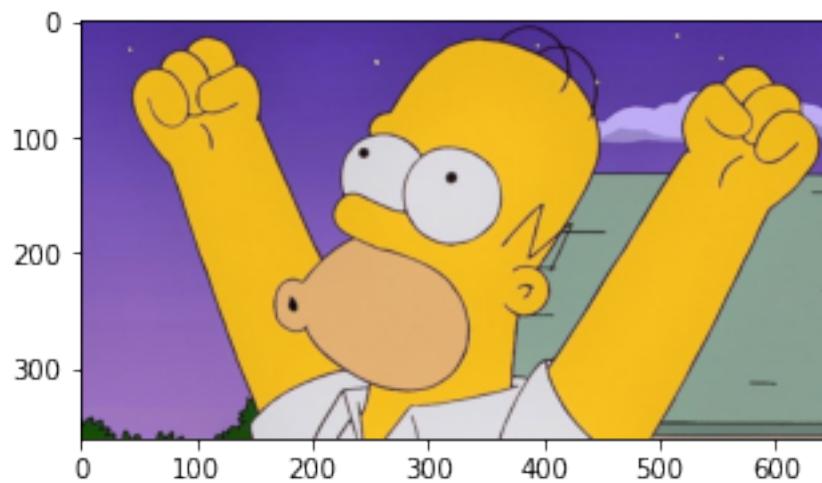
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=307x360 at 0x7FB119B84710>



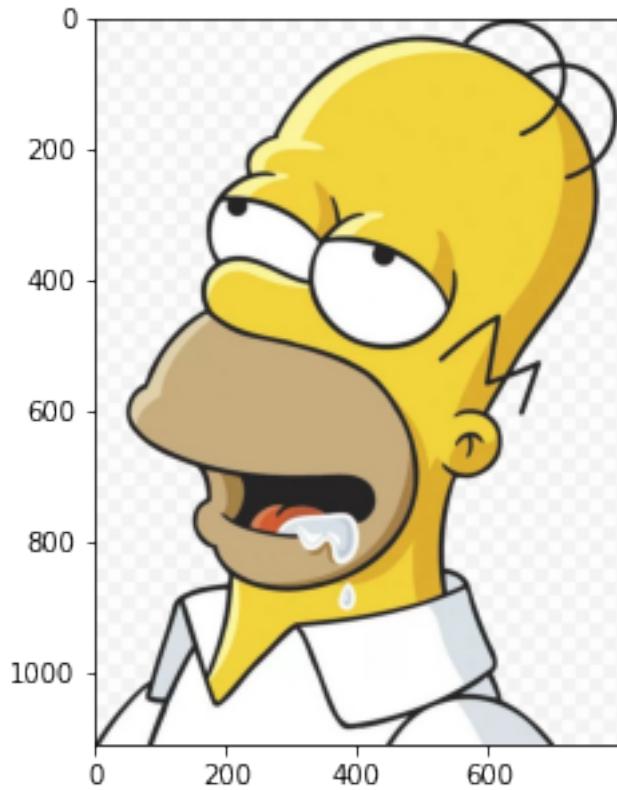
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1122x1332 at 0x7FB119B84490>



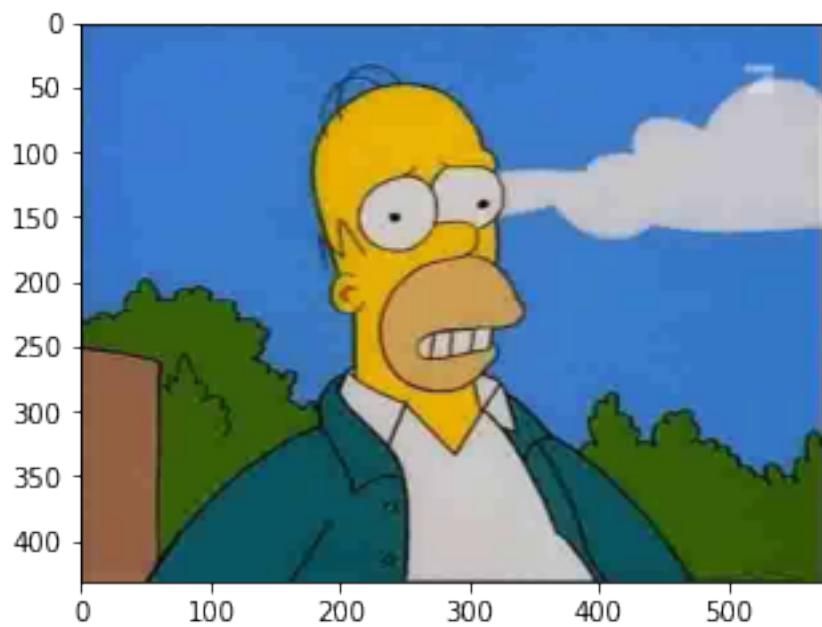
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=644x361 at 0x7FB119B84ED0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=800x1110 at 0x7FB119B848D0>



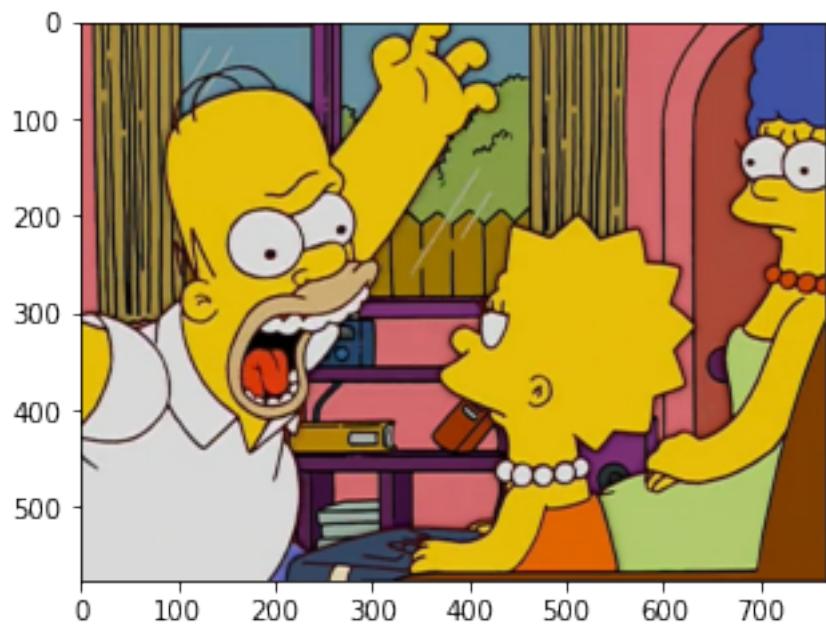
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=576x432 at 0x7FB119B7E2D0>



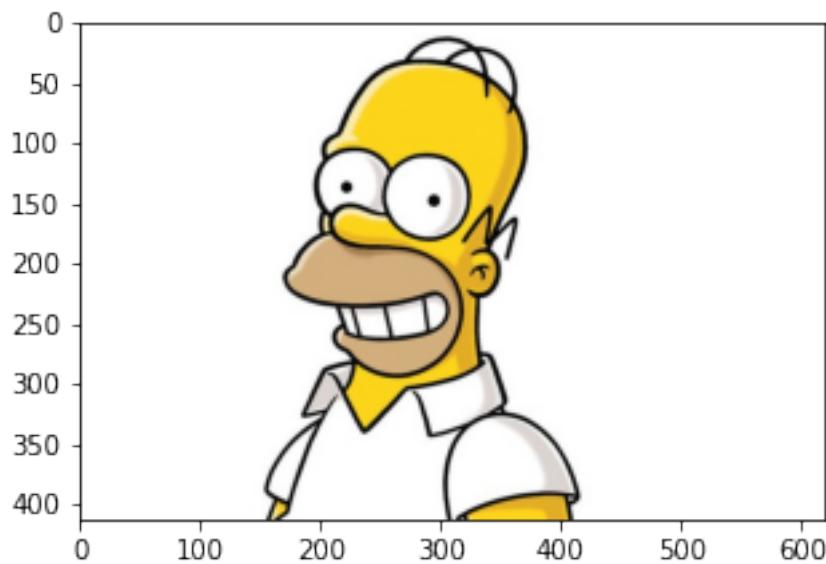
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=675x719 at 0x7FB119B7E690>



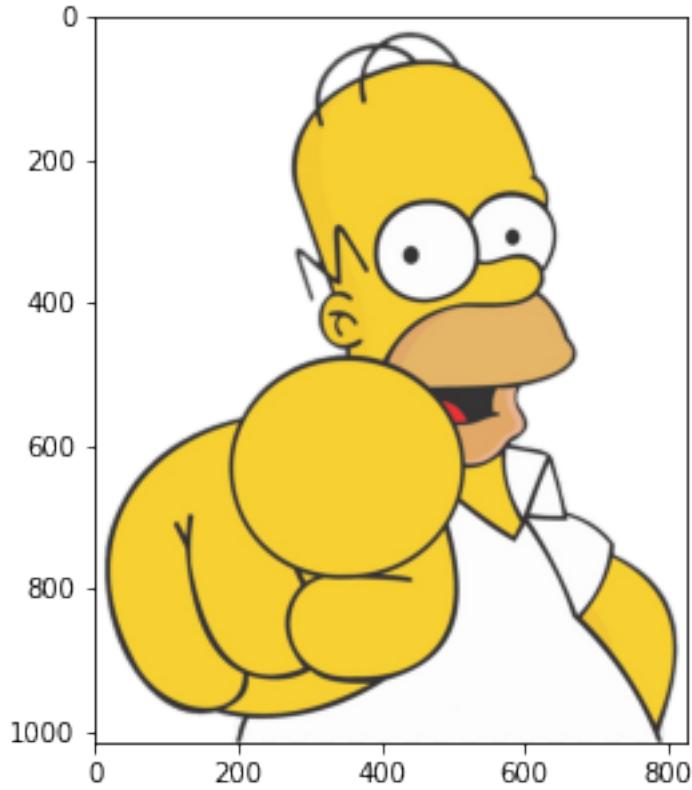
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=768x576 at 0x7FB119B7E250>



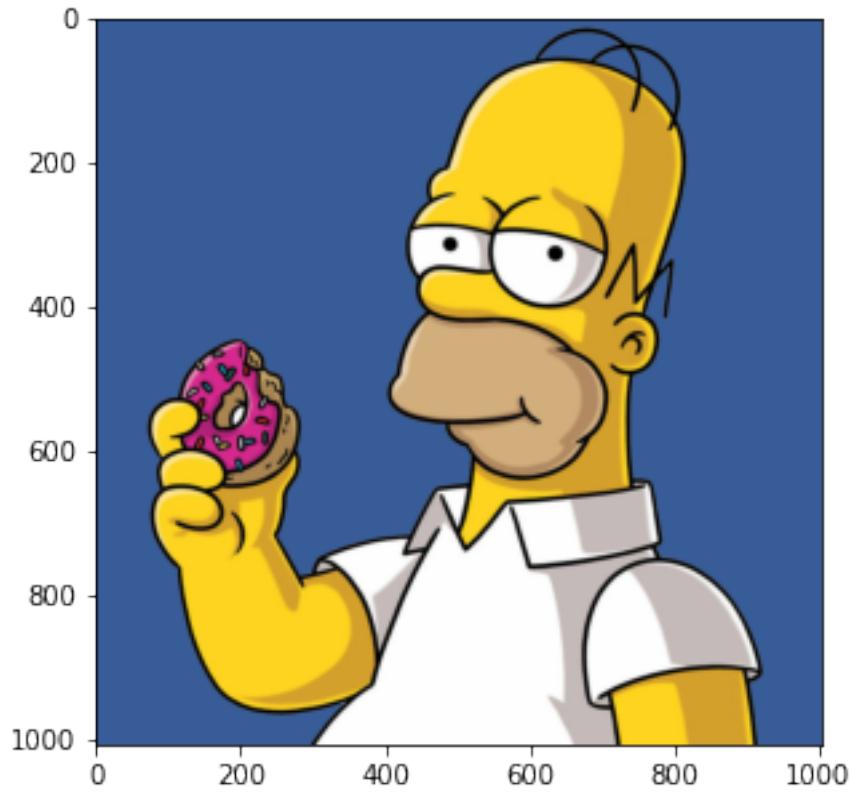
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=620x413 at 0x7FB119B84E50>



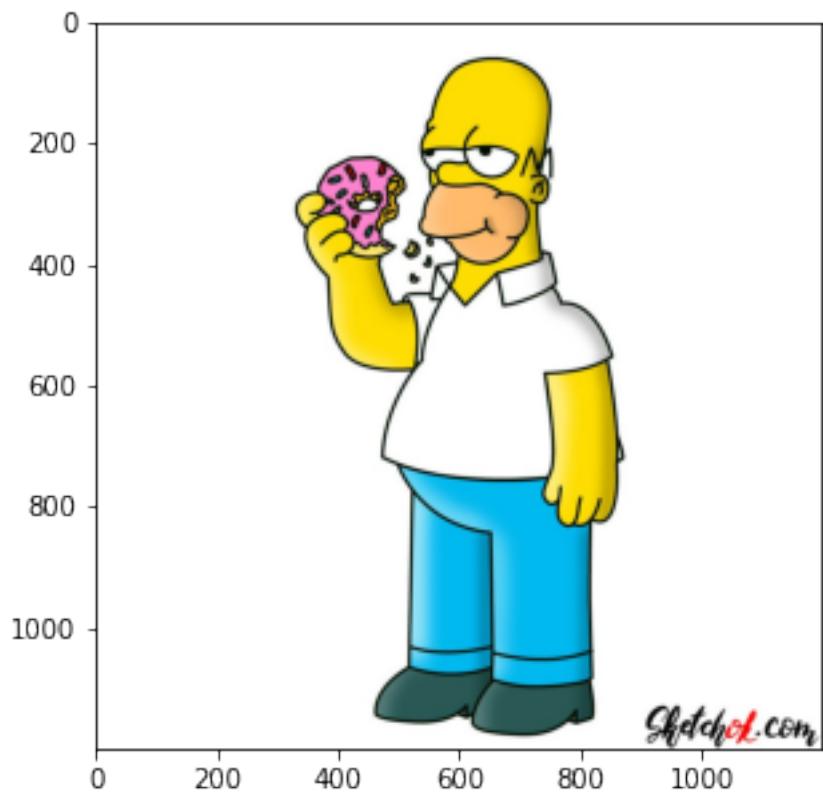
<PIL.PngImagePlugin.PngImageFile image mode=P size=828x1015 at 0x7FB119B7EC90>



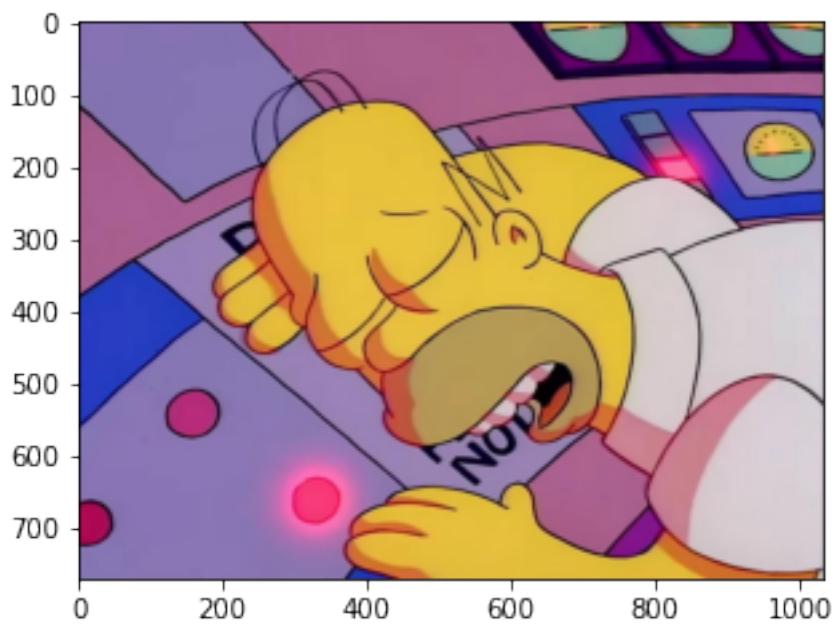
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1006x1006 at  
0x7FB119B7E350>
```



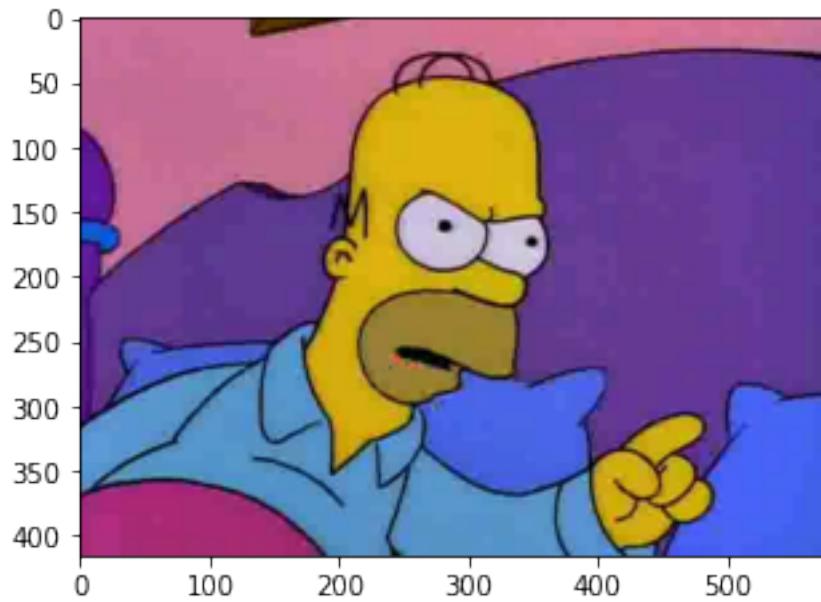
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1200x1200 at  
0x7FB119B84C50>
```



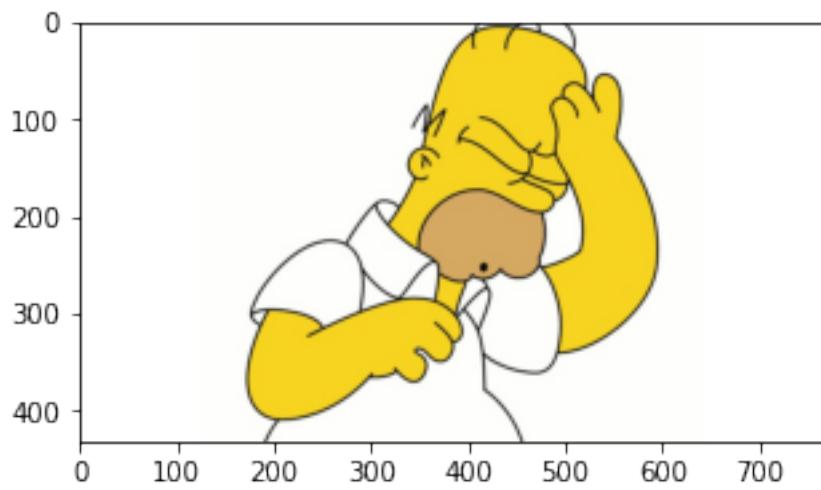
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1034x772 at 0x7FB119B7E050>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=576x416 at 0x7FB119B7E590>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=768x432 at 0x7FB119B7ED10>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1920x1280 at 0x7FB119B7EB50>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=689x699 at 0x7FB119B7EC50>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1420x947 at 0x7FB119B7E950>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=576x416 at 0x7FB119B7EA90>



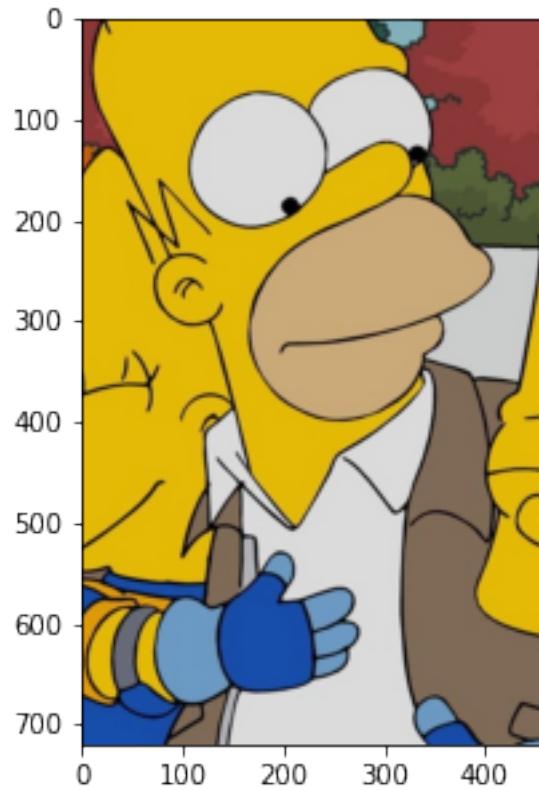
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=620x360 at 0x7FB119B7E7D0>



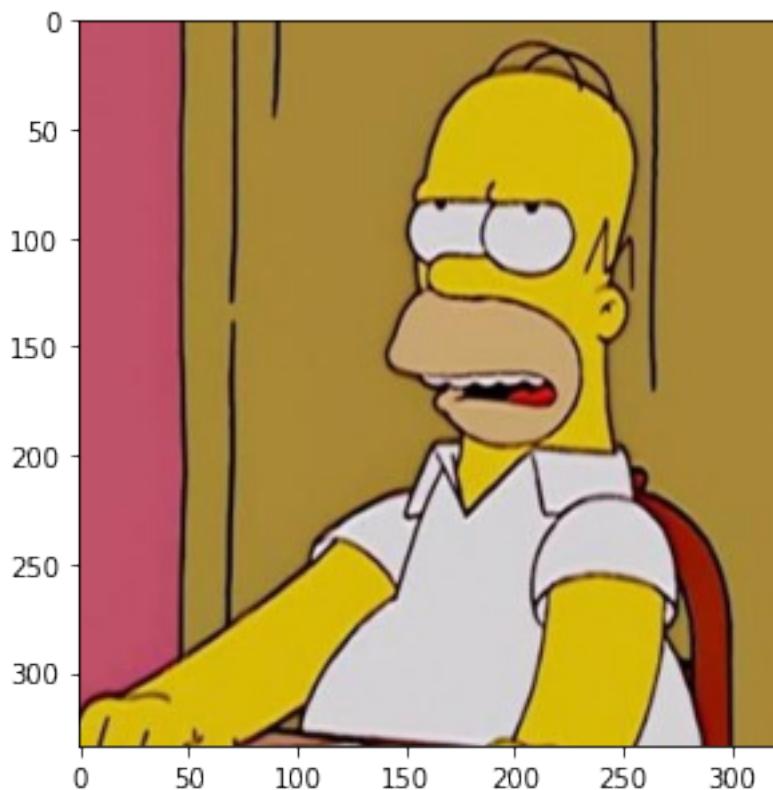
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1024x747 at 0x7FB119B7E290>



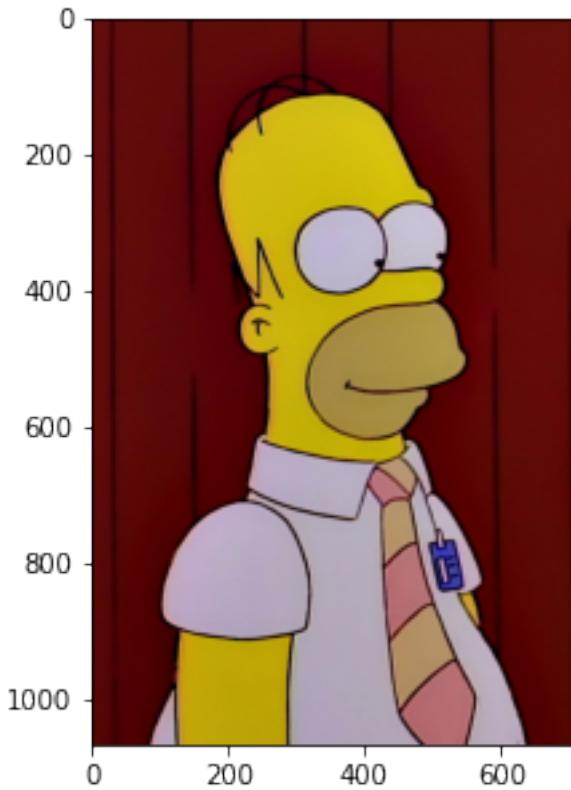
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=455x720 at 0x7FB119B7E390>



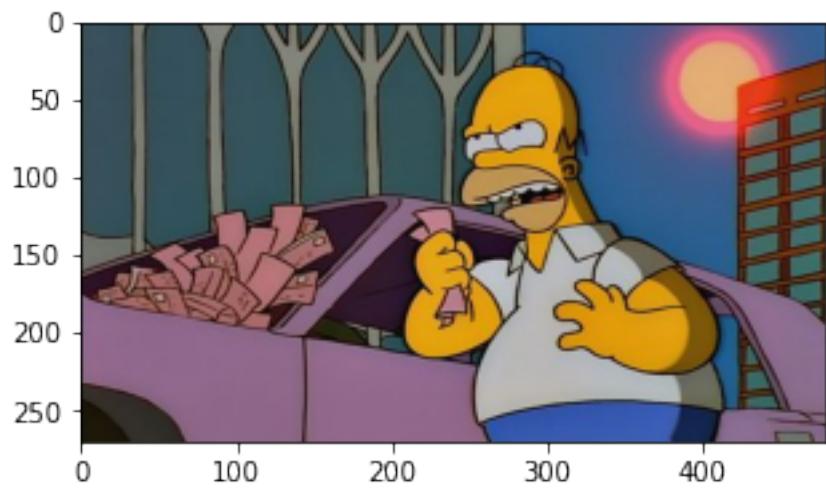
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=321x334 at 0x7FB119B7E150>



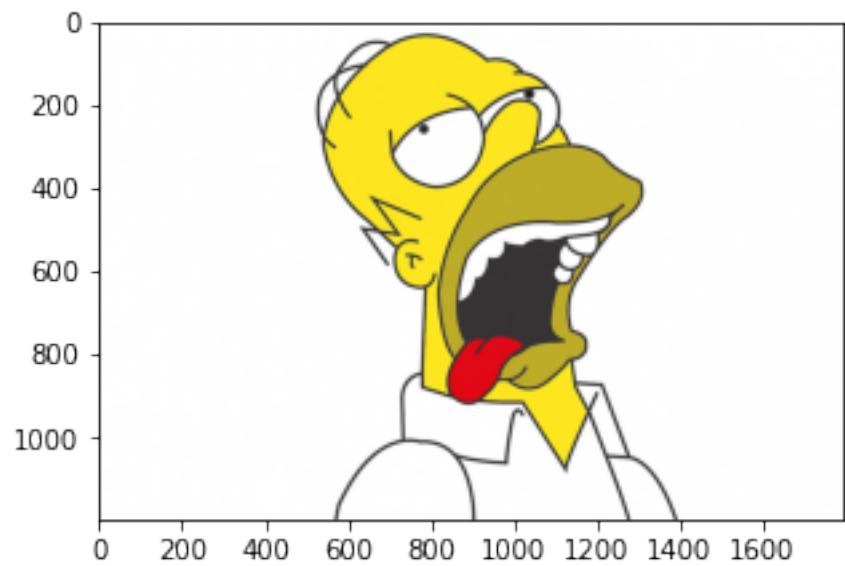
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=706x1067 at 0x7FB119B7E890>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=480x270 at 0x7FB119B9D050>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1796x1197 at 0x7FB119B9D0D0>



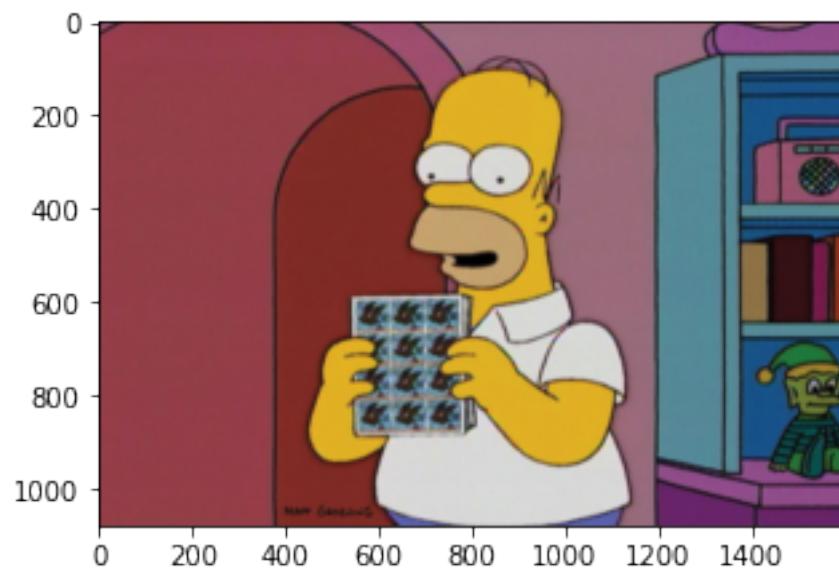
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=316x345 at 0x7FB119B9D150>



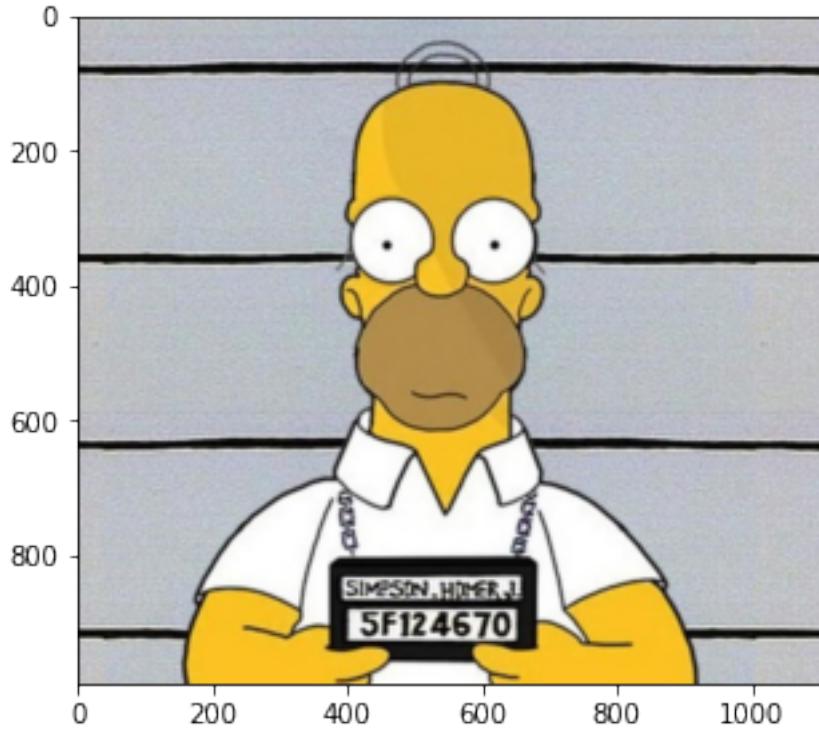
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1393x965 at 0x7FB11B485C10>



```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1600x1080 at  
0x7FB11B499A90>
```



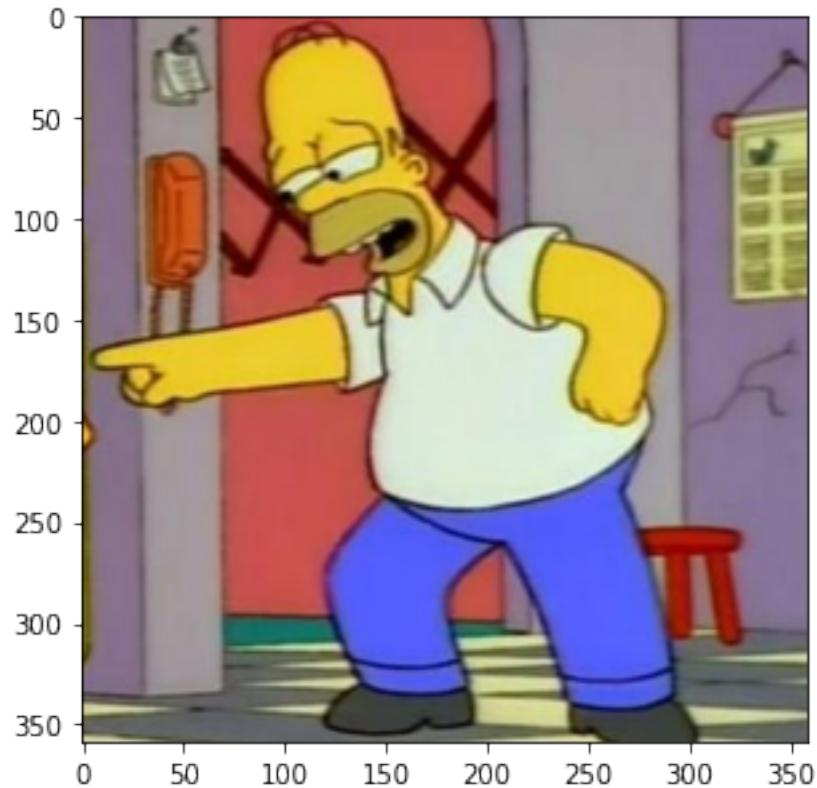
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1105x990 at 0x7FB11B485110>
```



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=615x410 at 0x7FB11B485E50>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=359x359 at 0x7FB11B4892D0>



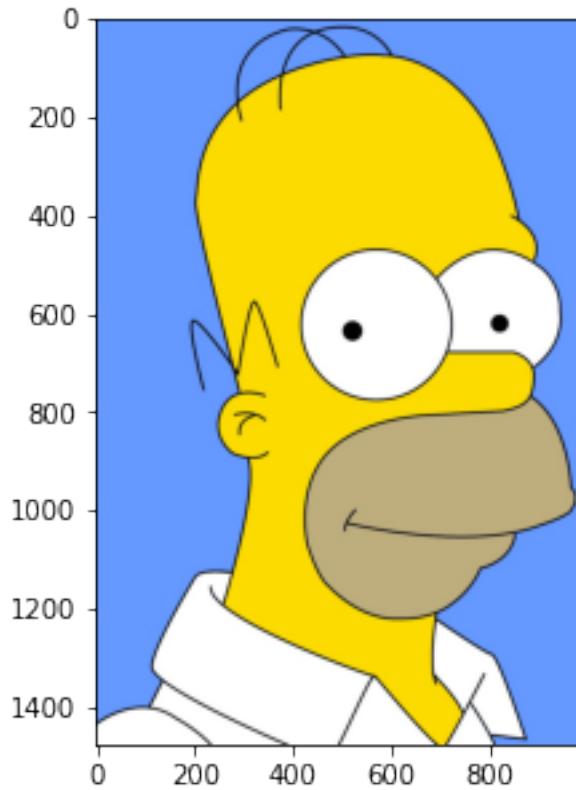
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=259x194 at 0x7FB11B489810>



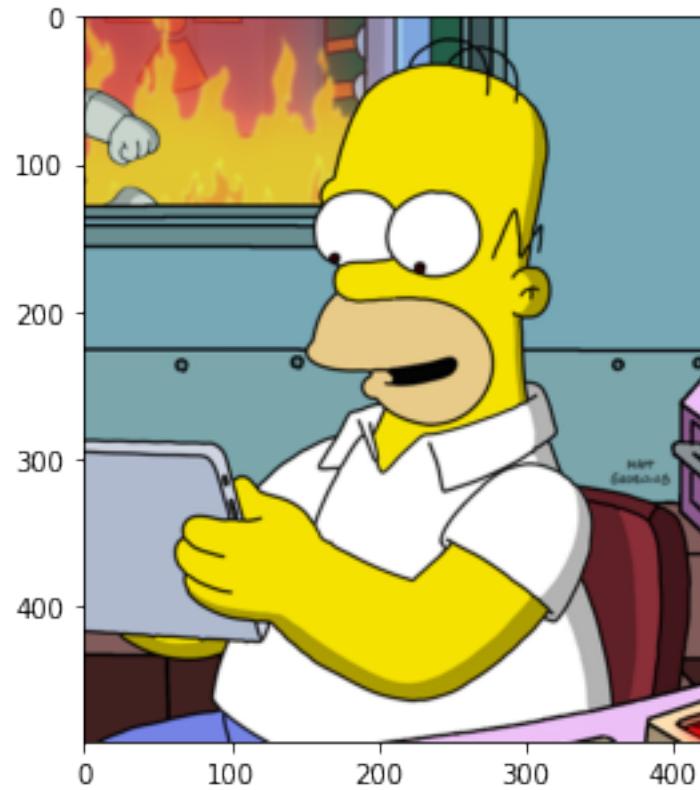
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=667x856 at 0x7FB11B489B90>



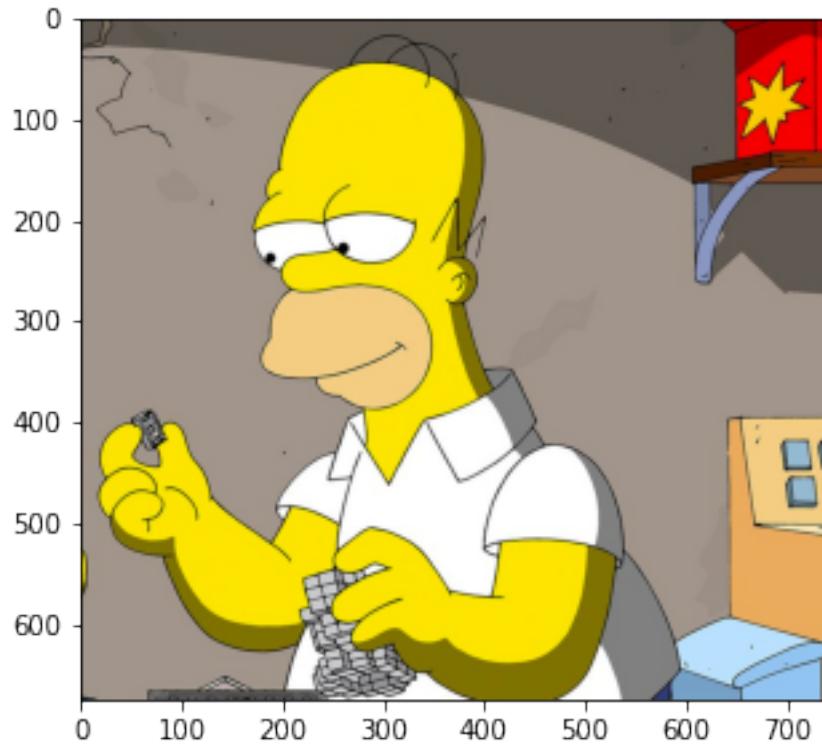
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=980x1476 at 0x7FB11B4E7A10>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=420x492 at 0x7FB11B4E7390>

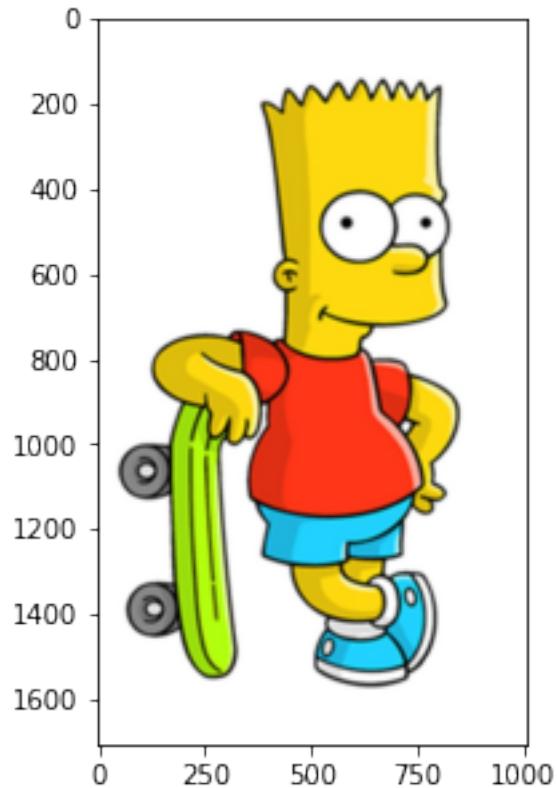


<PIL.PngImagePlugin.PngImageFile image mode=RGB size=739x675 at 0x7FB119B84950>

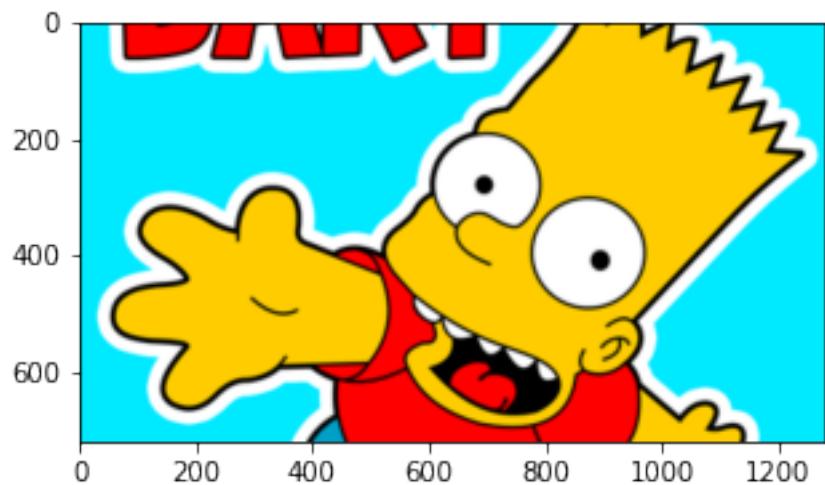


Bart

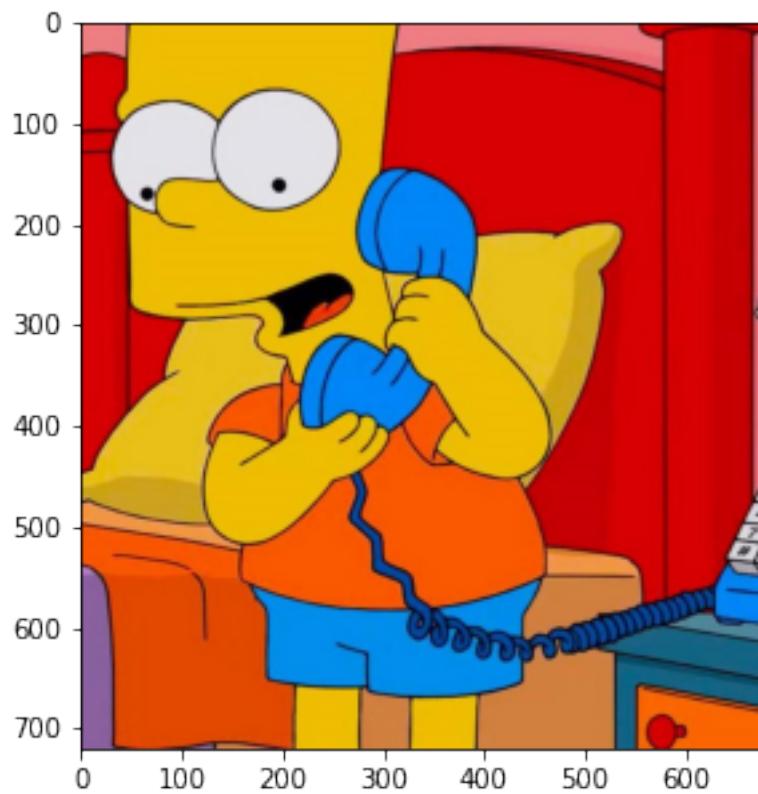
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1007x1706 at  
0x7FB119B7E110>
```



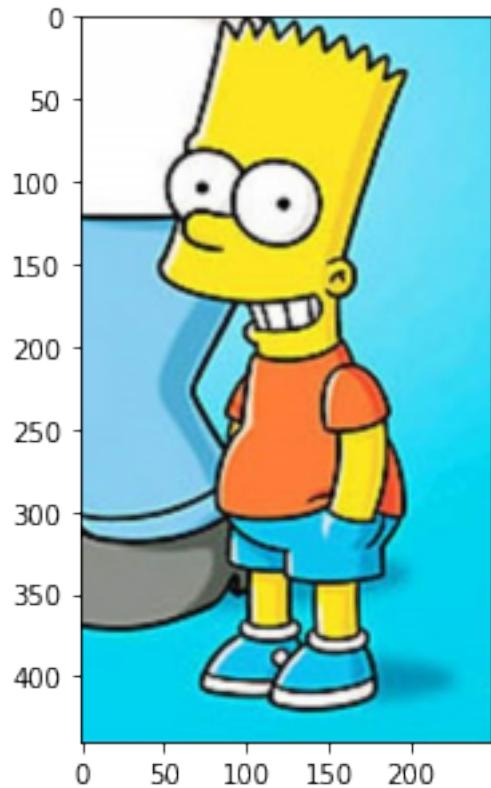
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1280x720 at  
0x7FB119B9D4D0>
```



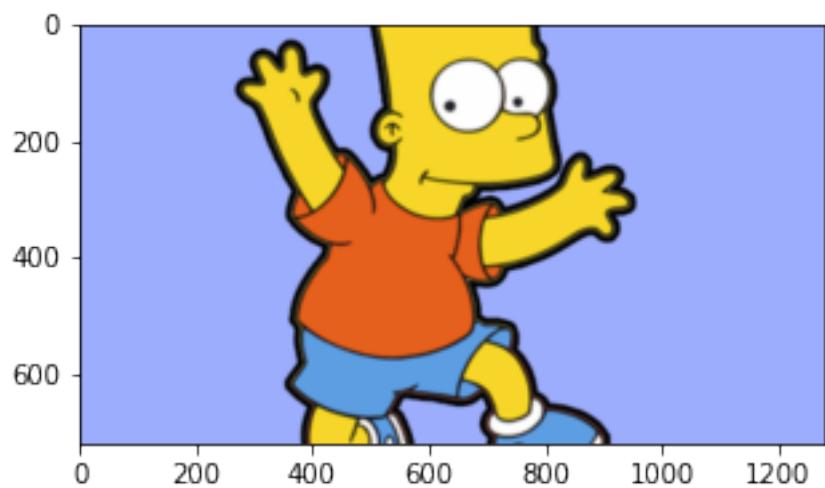
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=675x720 at 0x7FB119B9D510>
```



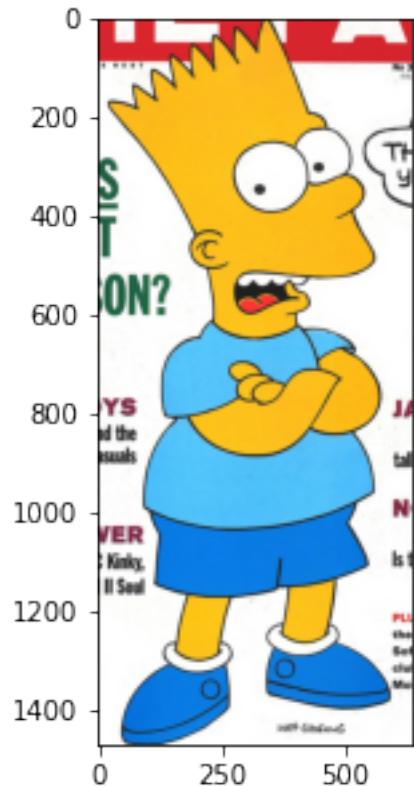
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=249x440 at 0x7FB119B9D590>



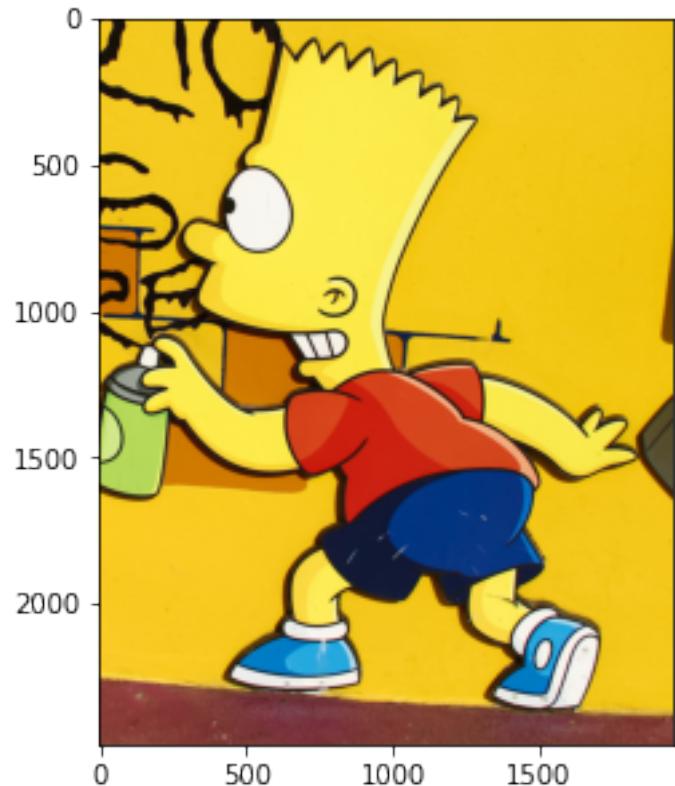
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1280x720 at 0x7FB119B9D610>
```



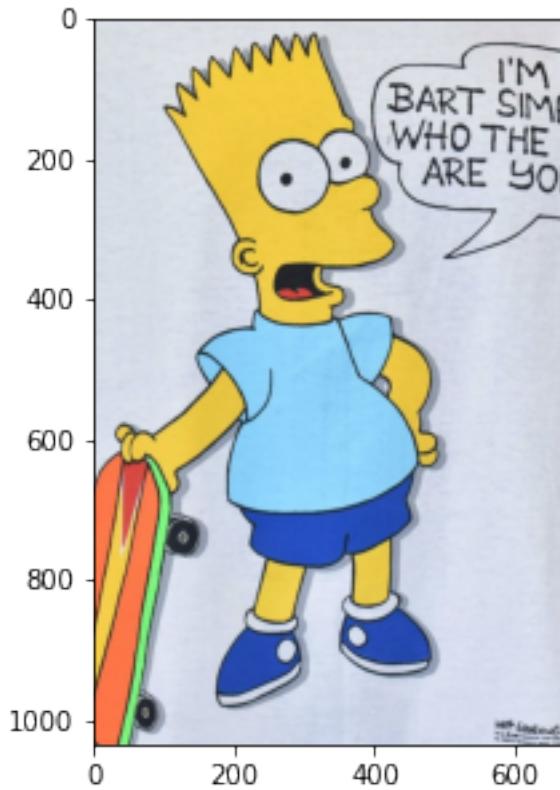
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=636x1468 at 0x7FB119B9D6D0>
```



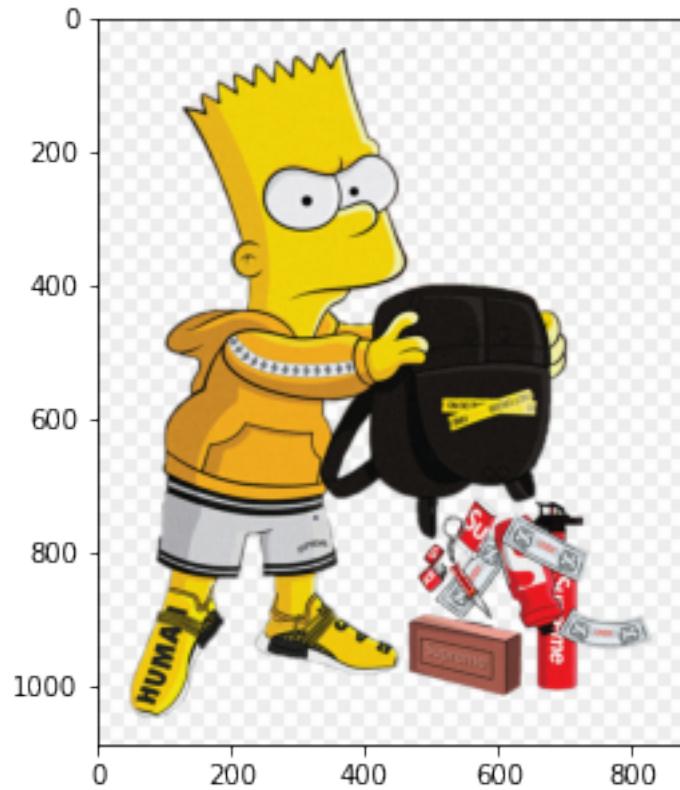
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1957x2480 at 0x7FB119B9D710>



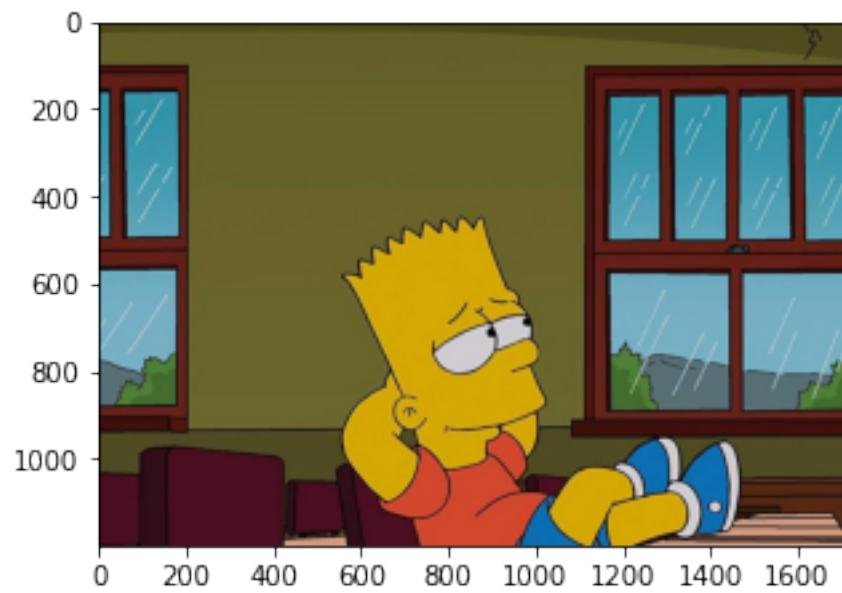
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=668x1035 at 0x7FB119B9D7D0>



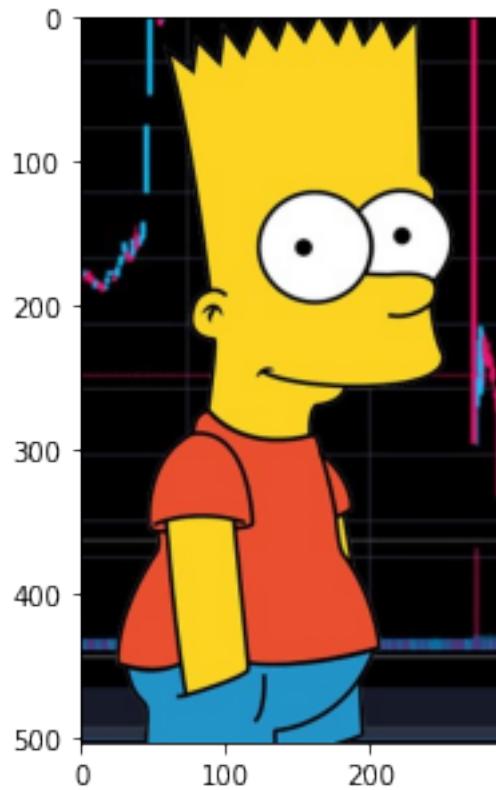
```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=880x1088 at  
0x7FB119B9D850>
```



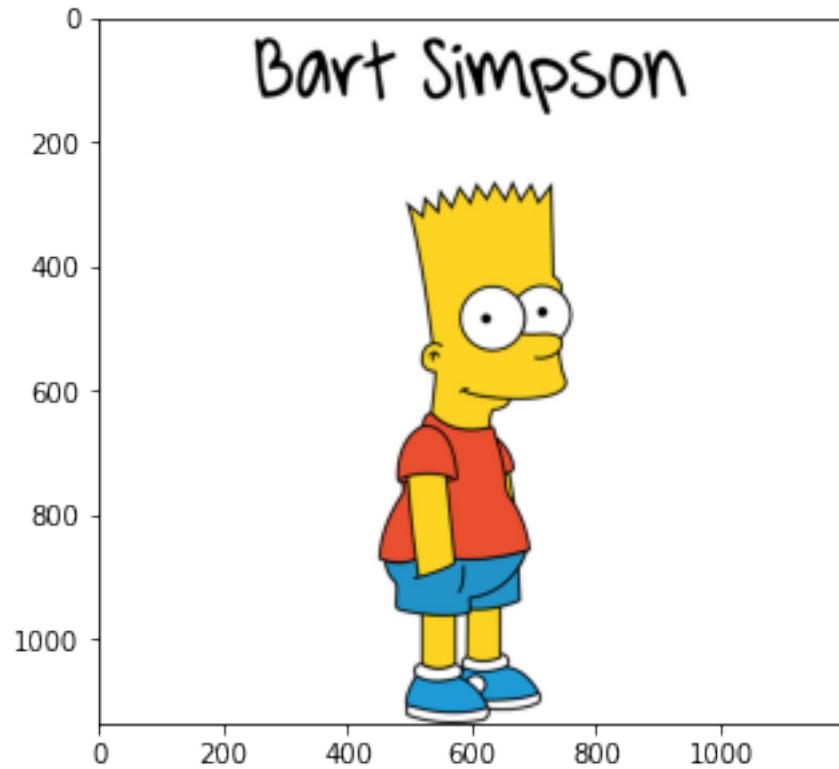
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1707x1200 at 0x7FB119B9D690>



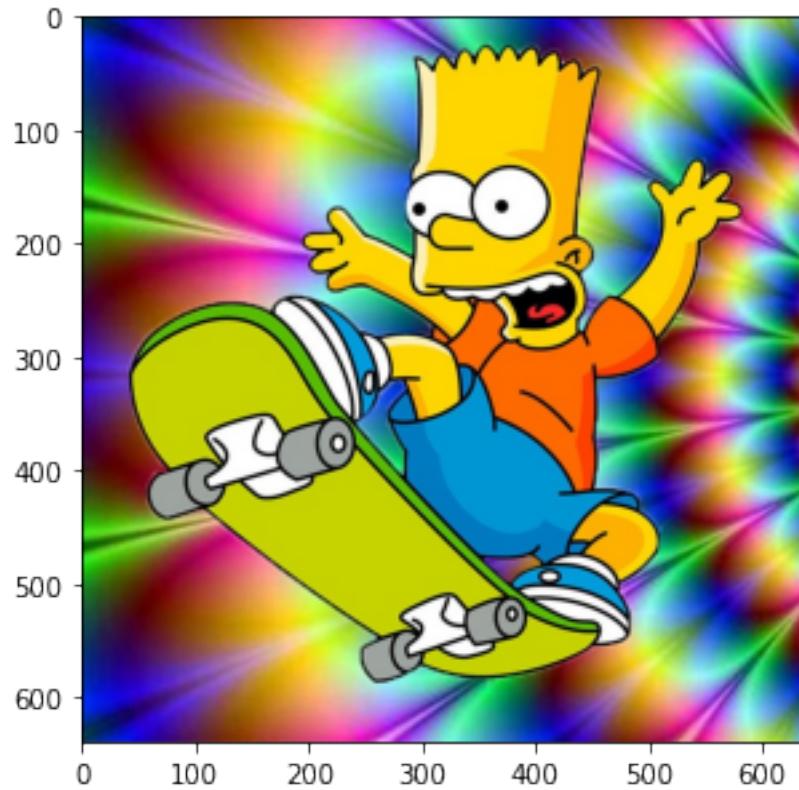
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=289x503 at 0x7FB119B9D8D0>
```



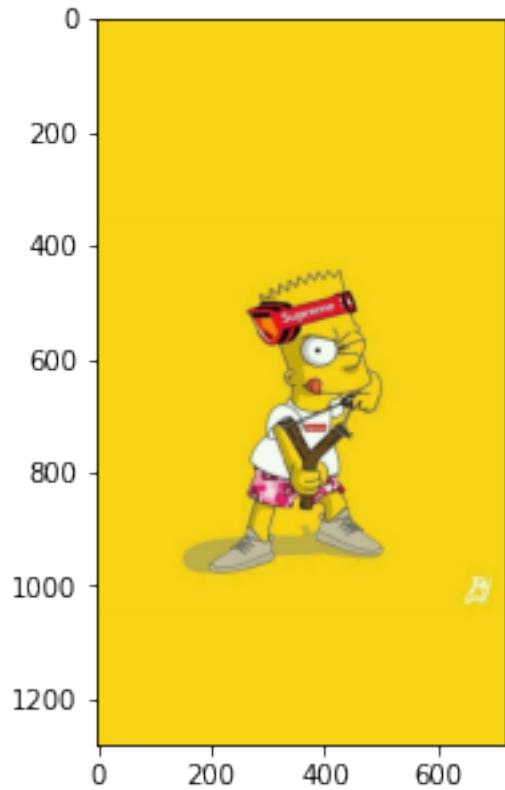
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1200x1136 at 0x7FB119B9DA90>
```



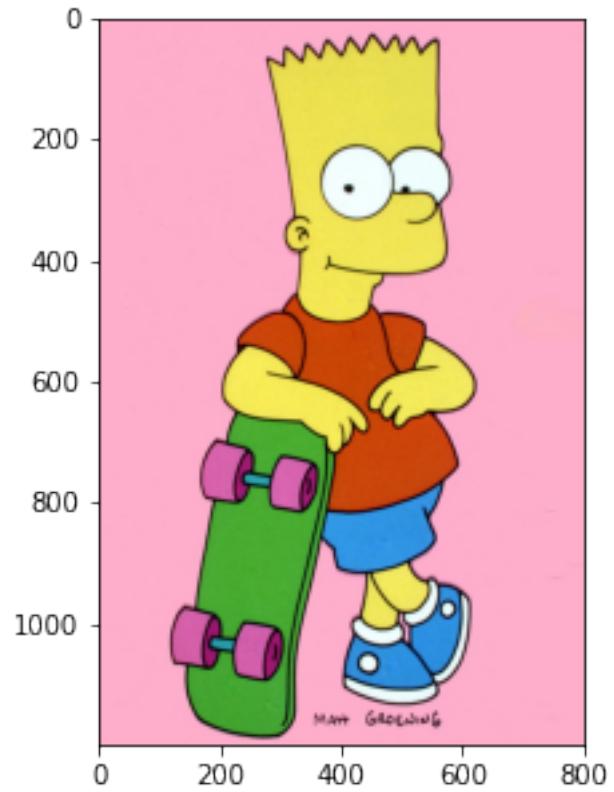
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=635x640 at 0x7FB119B9DB10>
```



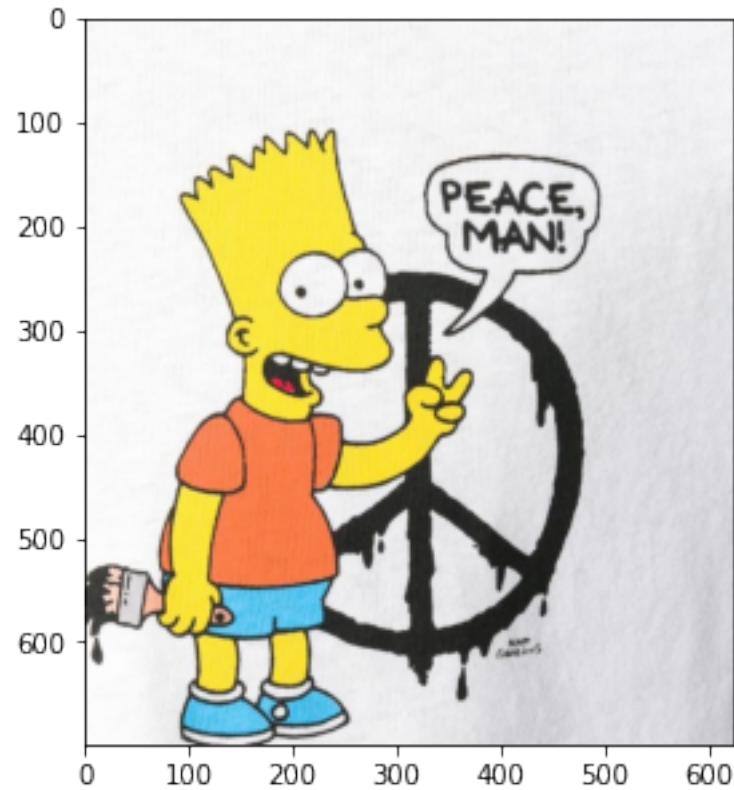
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=720x1280 at 0x7FB119B9DB90>



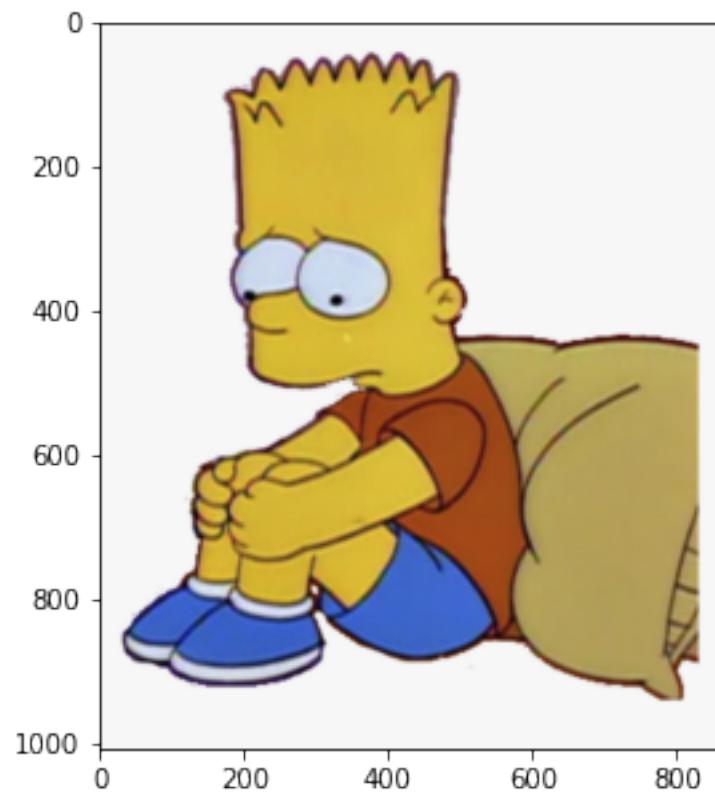
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=802x1200 at 0x7FB119B9DC10>



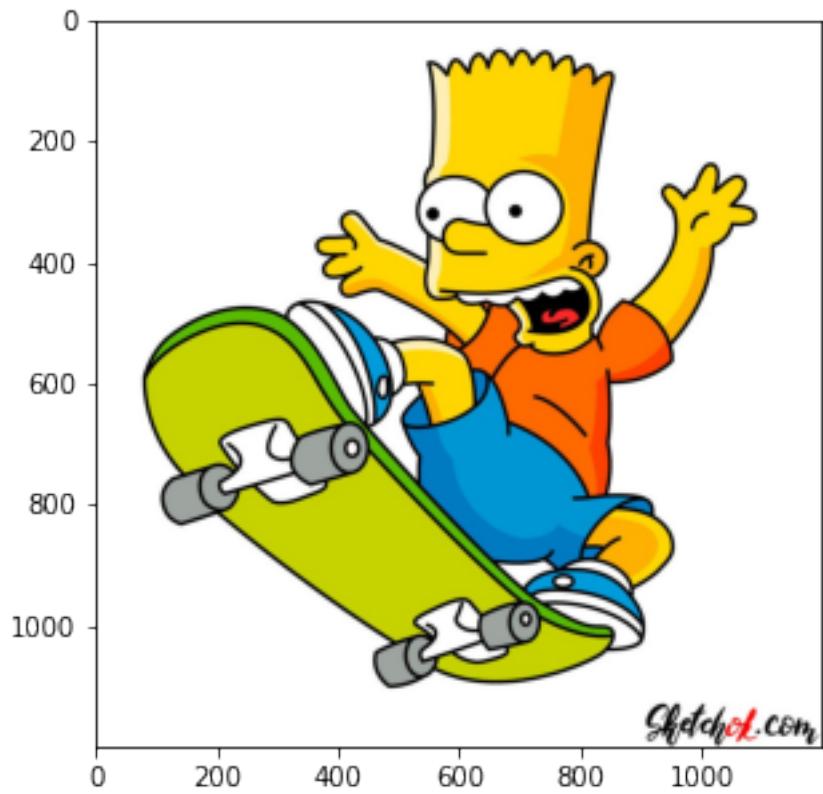
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=625x699 at 0x7FB119B9DC90>



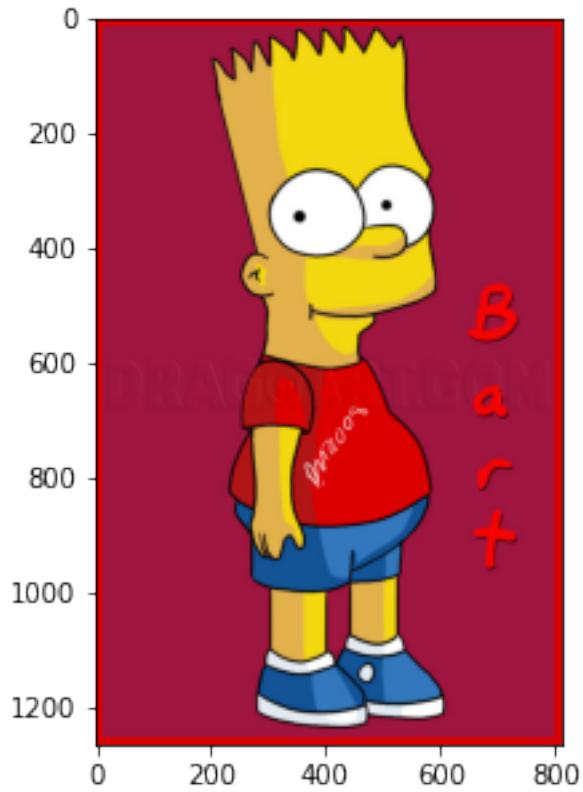
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=860x1007 at  
0x7FB119B9DD10>
```



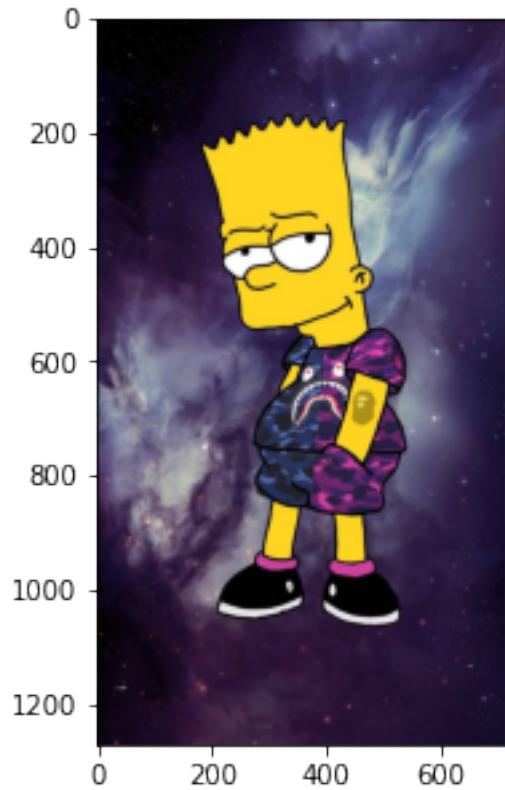
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1200x1200 at  
0x7FB119B9DD90>
```



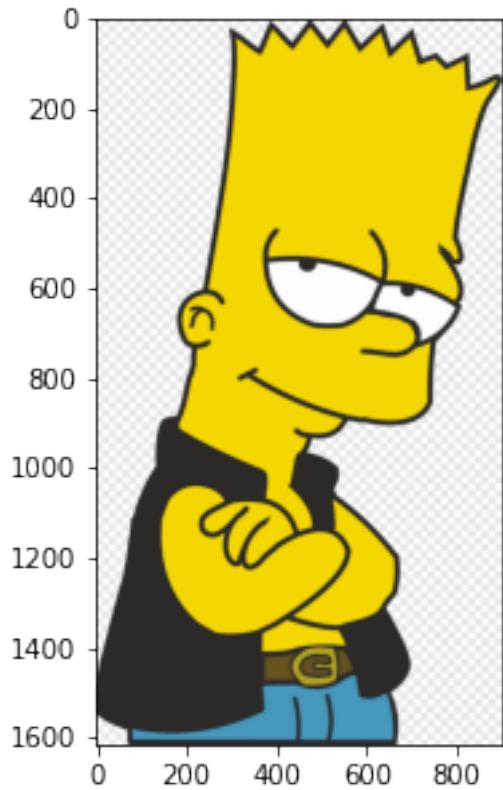
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=812x1266 at 0x7FB119B9DE10>



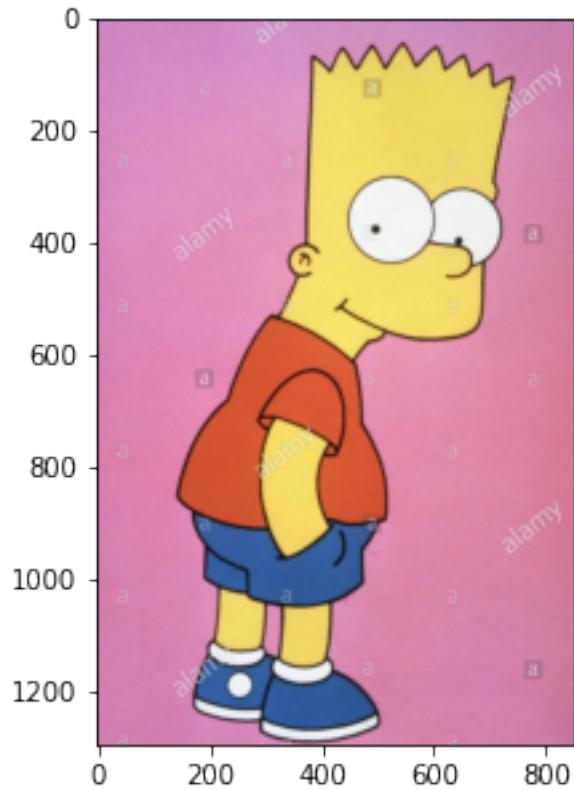
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=715x1271 at 0x7FB119B9DE90>



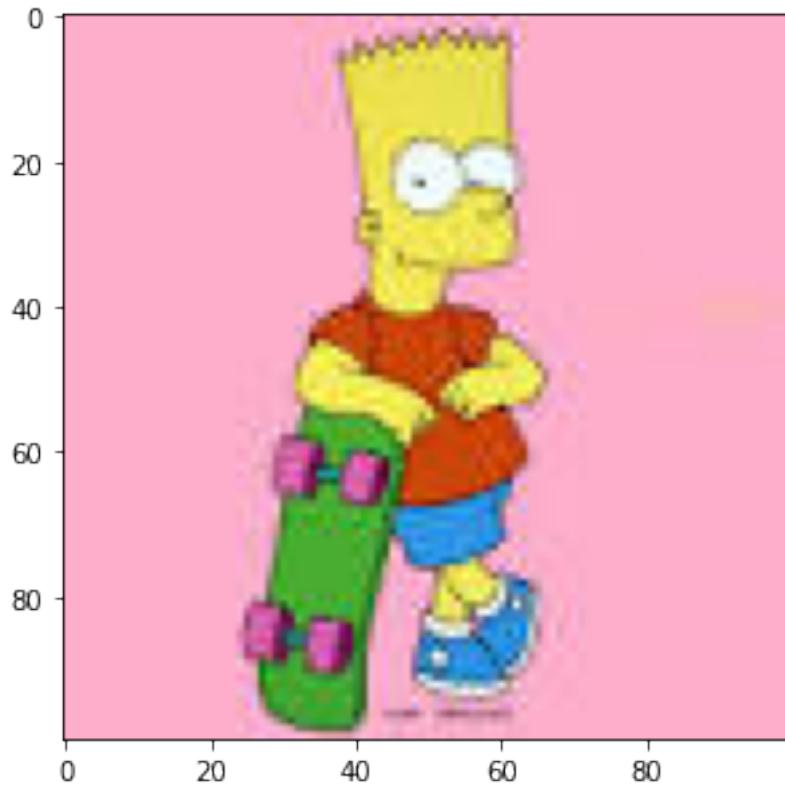
<PIL.PngImagePlugin.PngImageFile image mode=P size=900x1616 at 0x7FB119B9DF10>



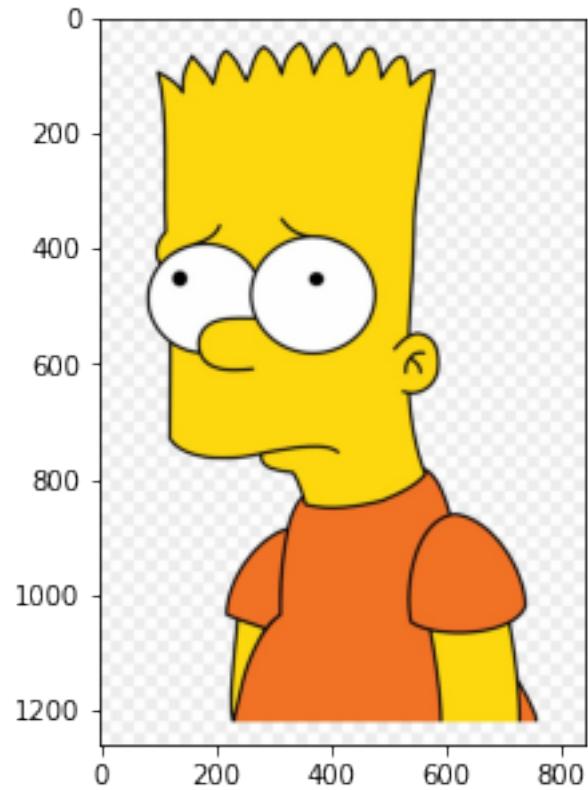
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=852x1295 at 0x7FB119B7EAD0>



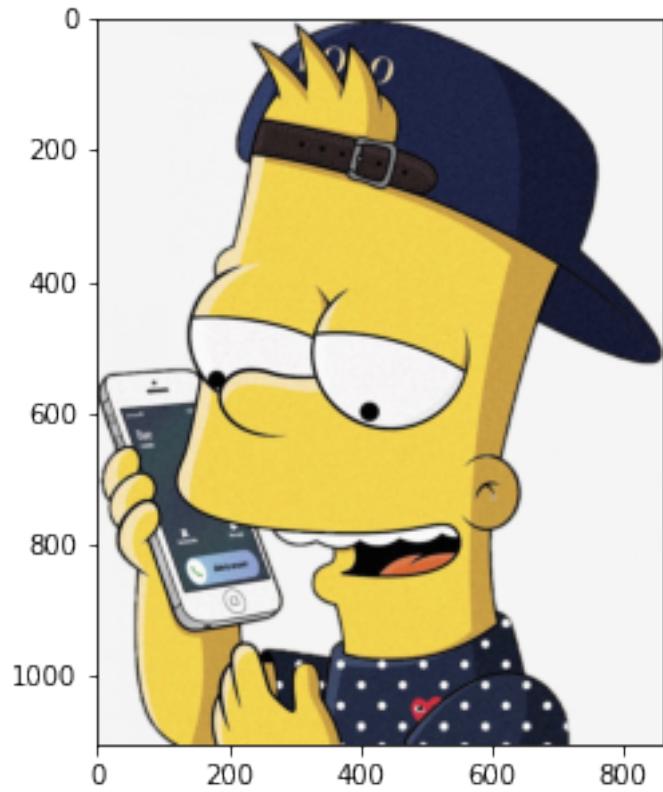
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=100x100 at 0x7FB119B9DFD0>



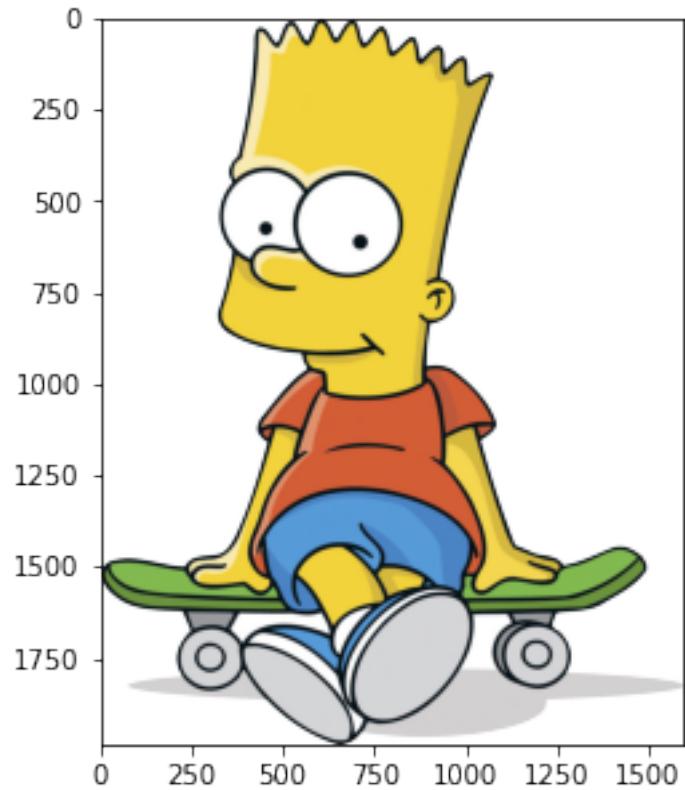
```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=840x1261 at  
0x7FB119B9D2D0>
```



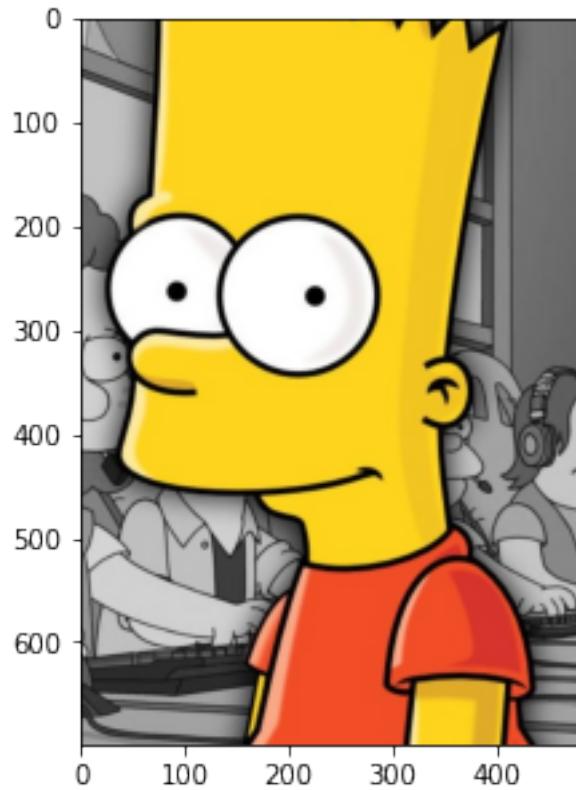
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=860x1105 at 0x7FB119BAB290>



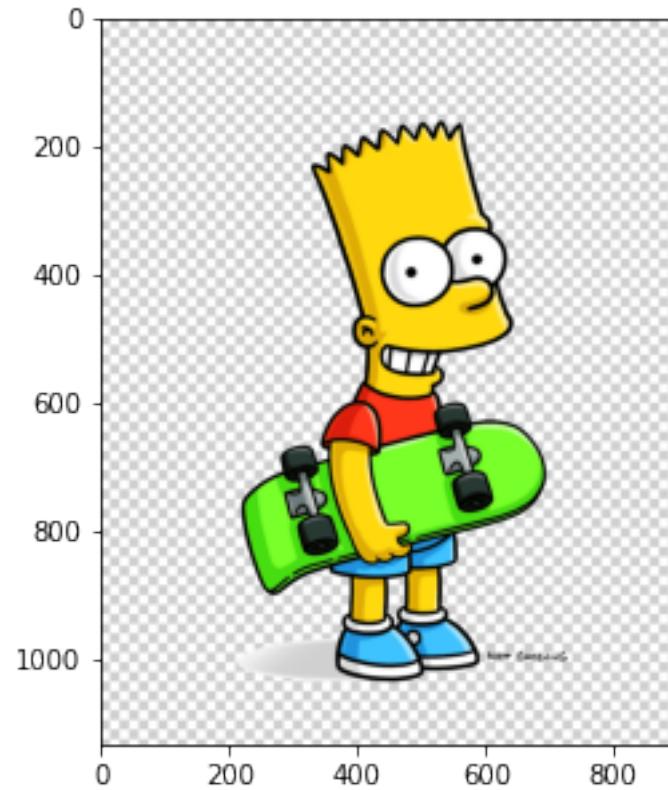
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1598x1986 at 0x7FB119B7EED0>



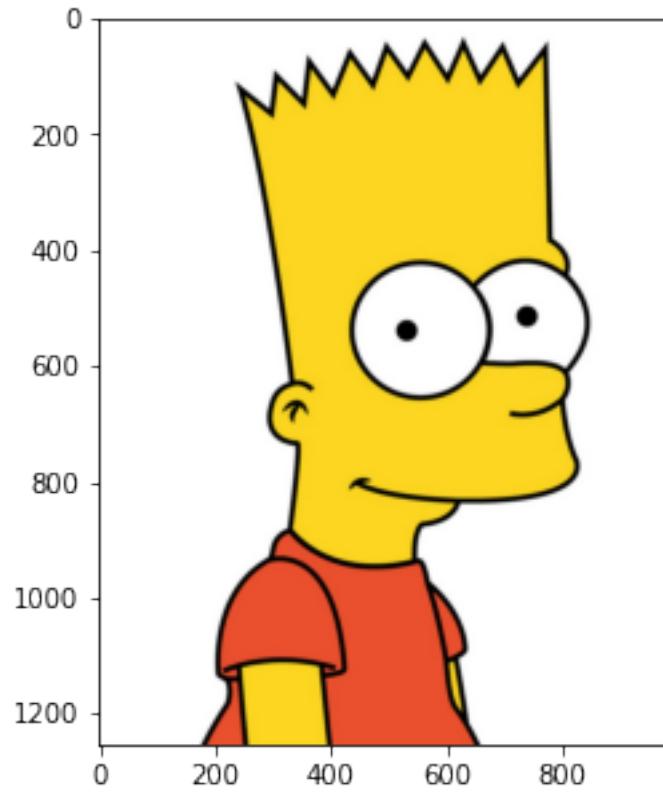
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=479x699 at 0x7FB119BAB1D0>



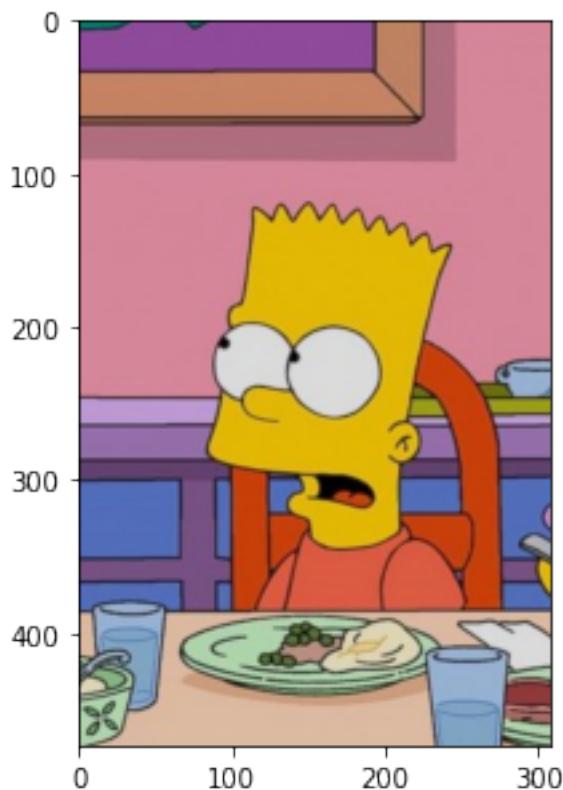
<PIL.PngImagePlugin.PngImageFile image mode=P size=890x1134 at 0x7FB119BAB390>



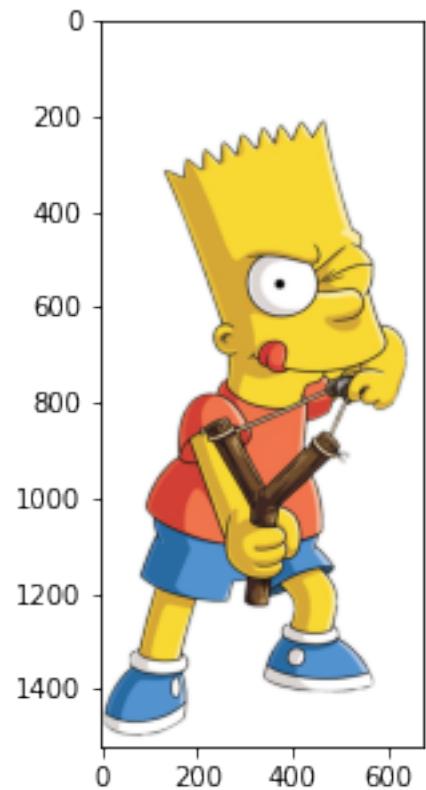
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=975x1254 at 0x7FB119BAB510>



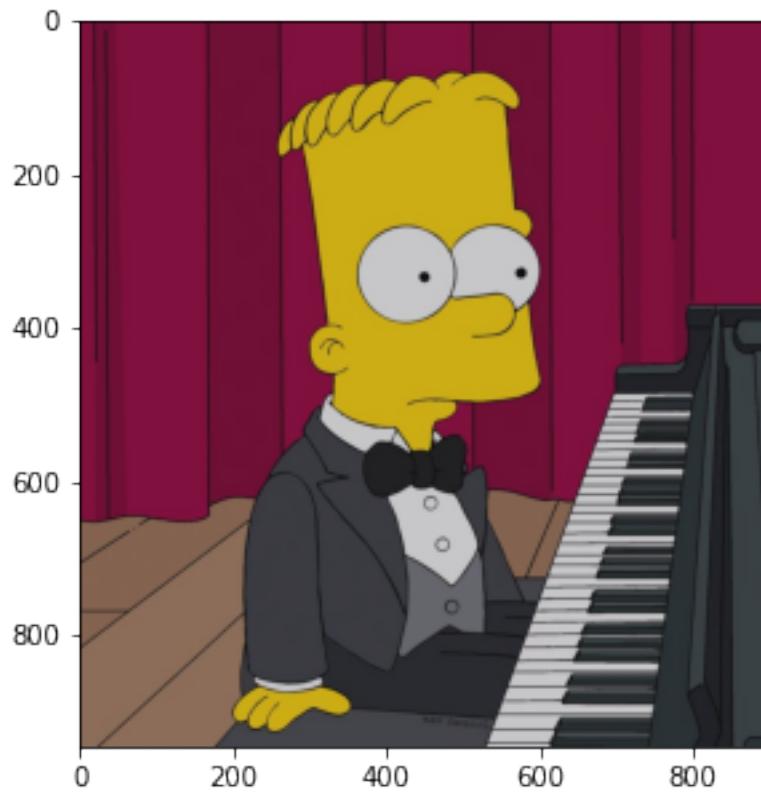
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=308x474 at 0x7FB119BAB450>



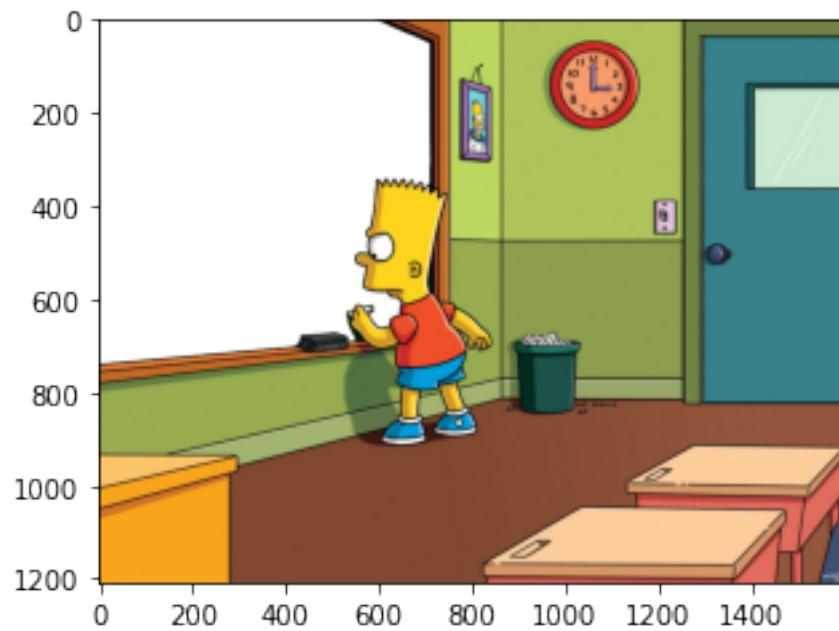
<PIL.PngImagePlugin.PngImageFile image mode=P size=677x1521 at 0x7FB119BAB550>



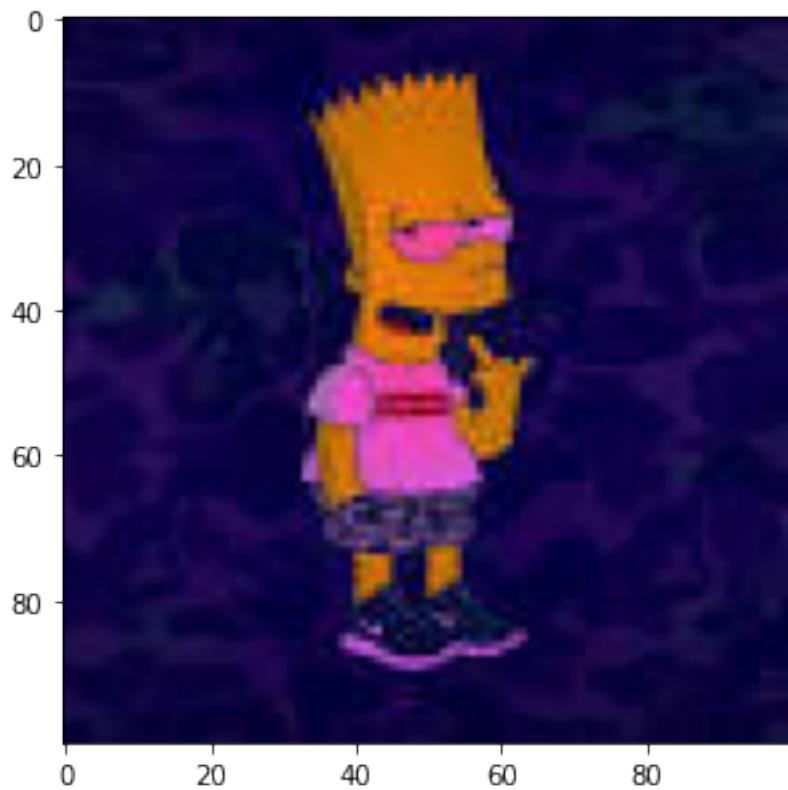
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=892x946 at 0x7FB119BAB6D0>



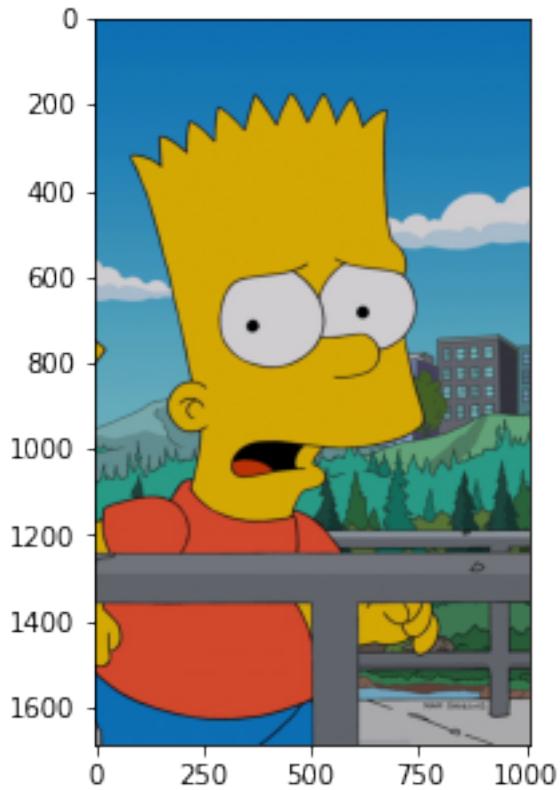
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1600x1208 at 0x7FB119B9D290>



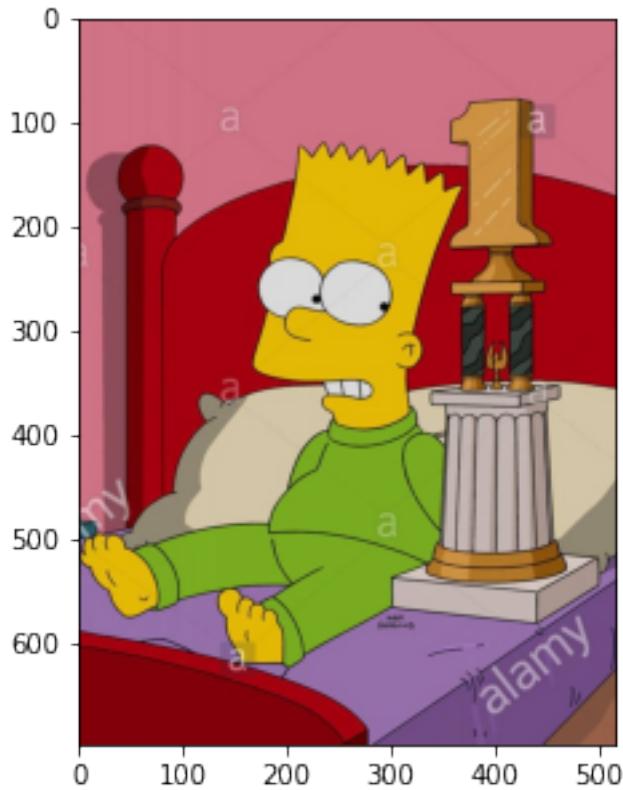
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=100x100 at 0x7FB119BAB710>



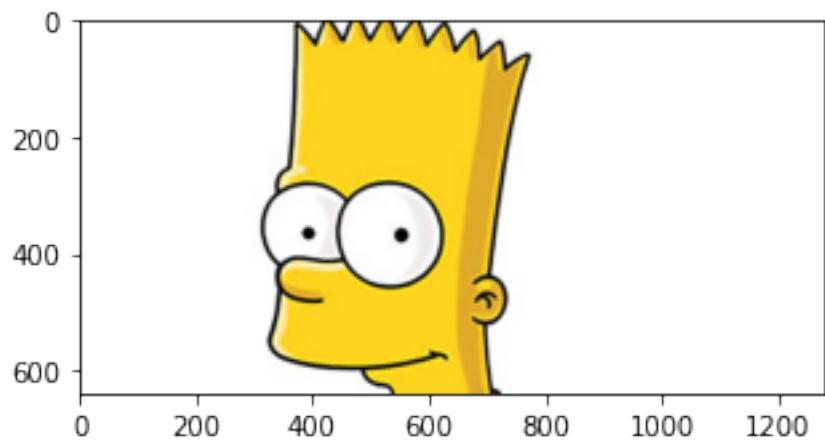
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1008x1687 at
0x7FB119BAB790>



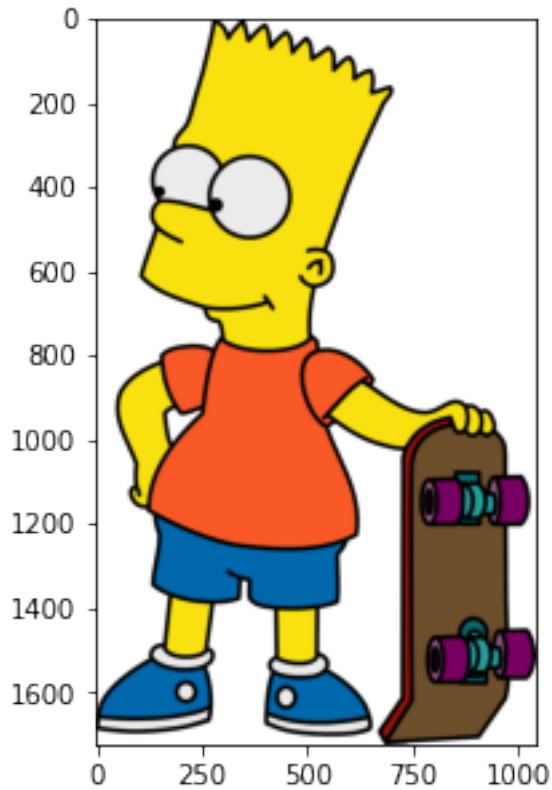
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=517x698 at 0x7FB119BAB810>



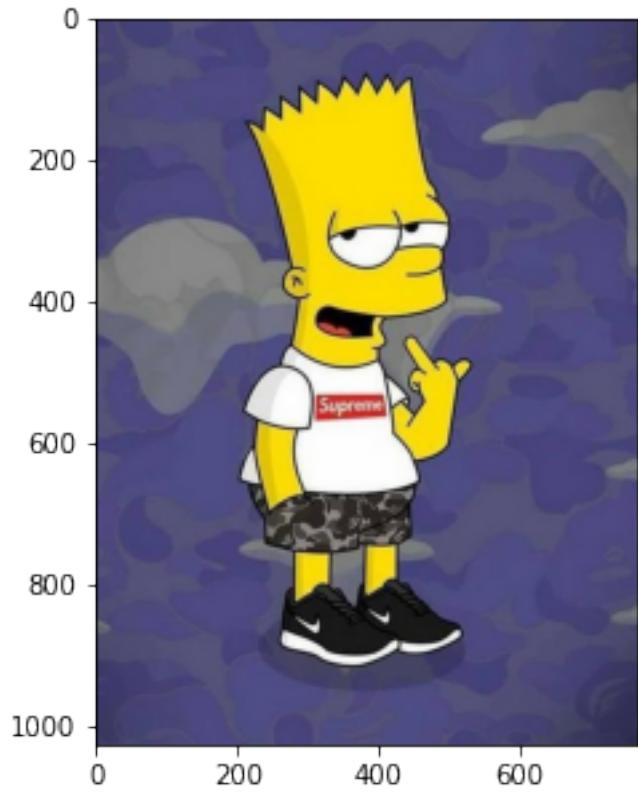
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1280x640 at 0x7FB119BAB890>



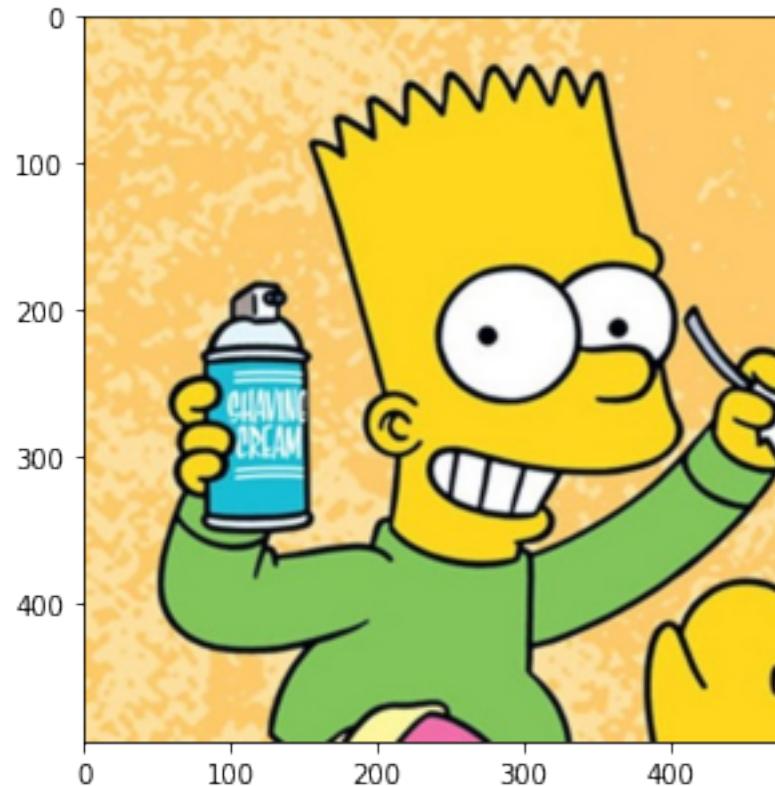
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1045x1723 at 0x7FB119BAB910>



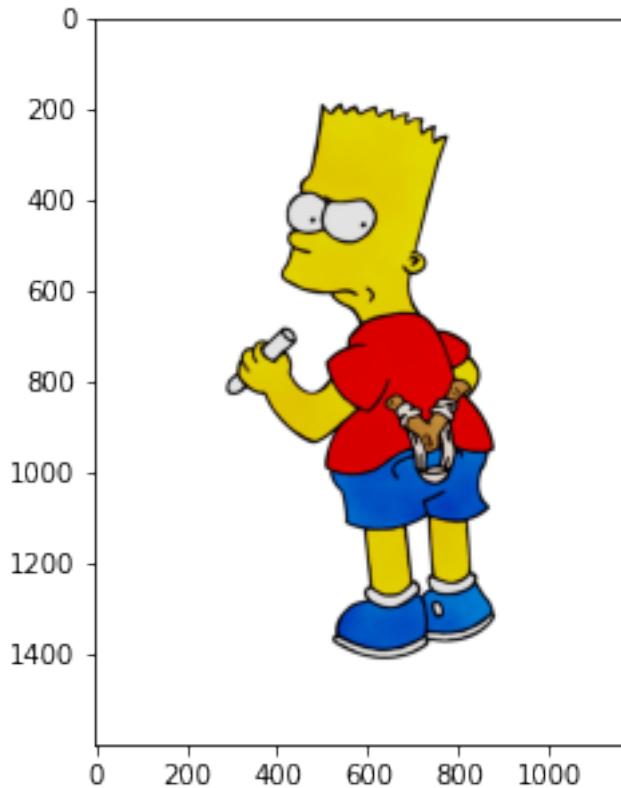
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=768x1028 at 0x7FB119BAB990>



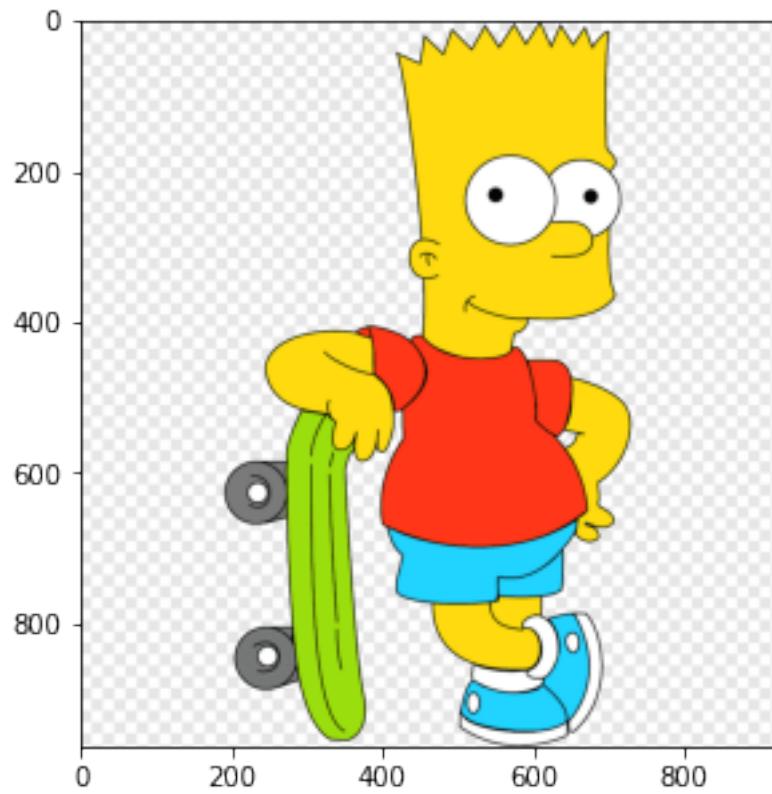
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=476x495 at 0x7FB119BABA10>



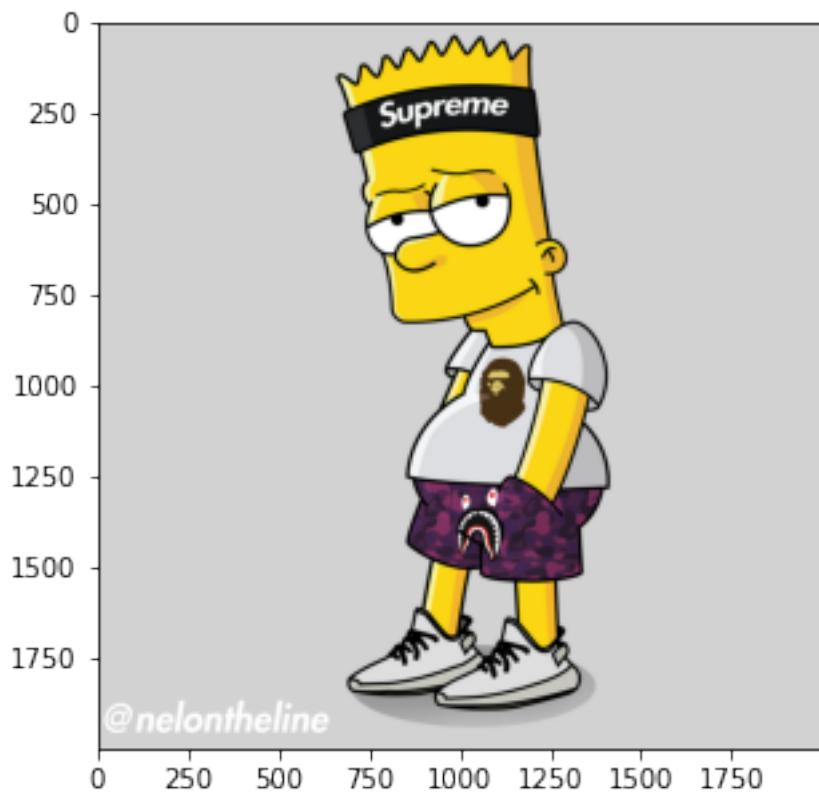
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1162x1600 at 0x7FB119BABA90>



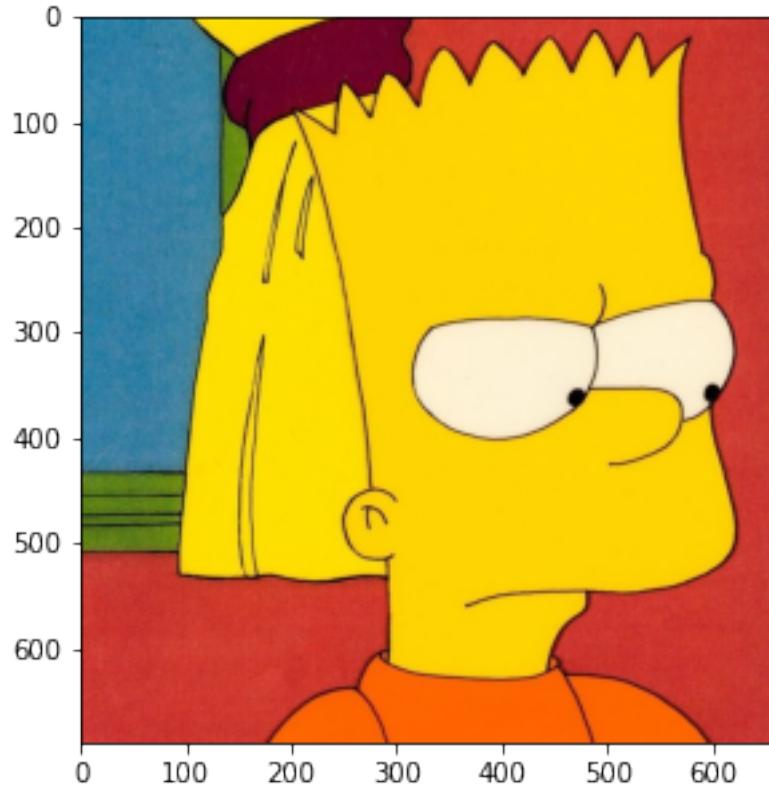
<PIL.PngImagePlugin.PngImageFile image mode=P size=920x963 at 0x7FB119B9D250>



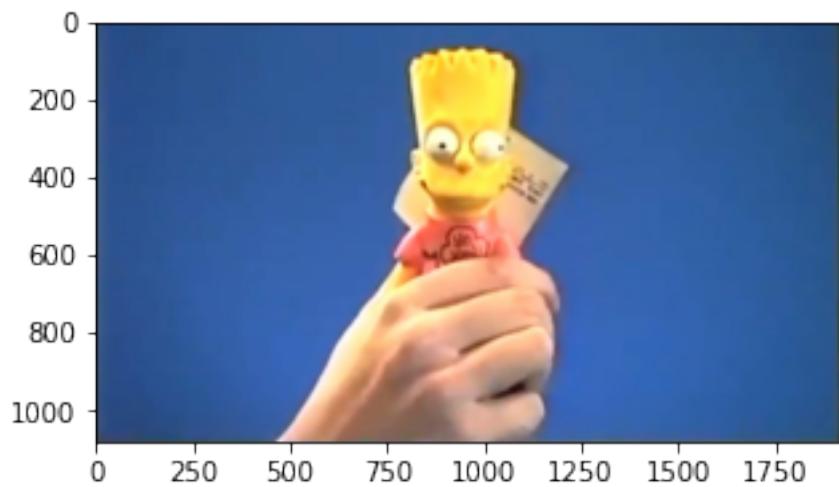
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=2000x2000 at 0x7FB119BACD0>



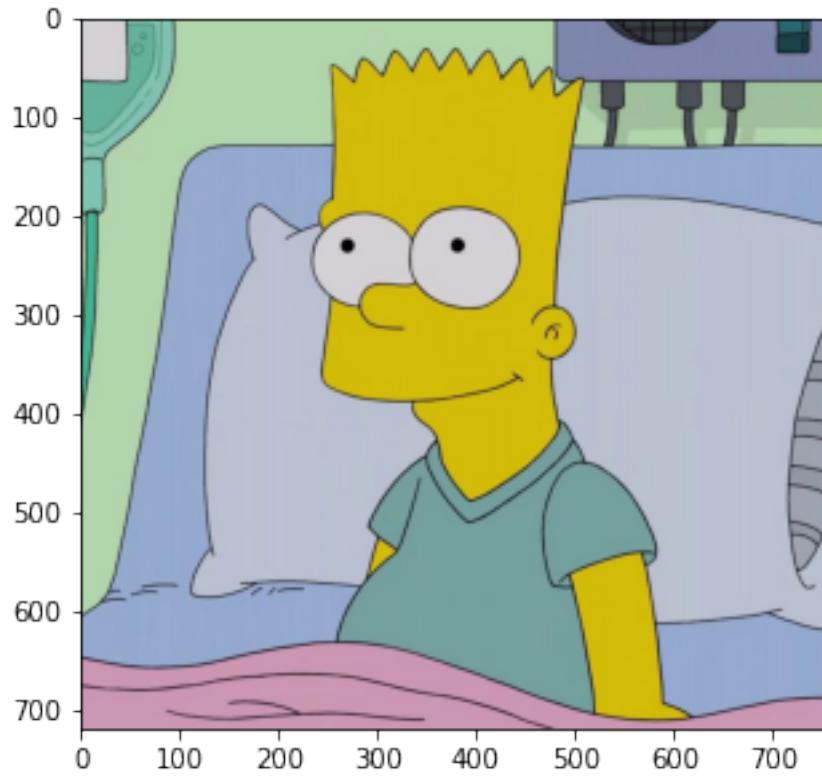
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=658x690 at 0x7FB119BABB0>



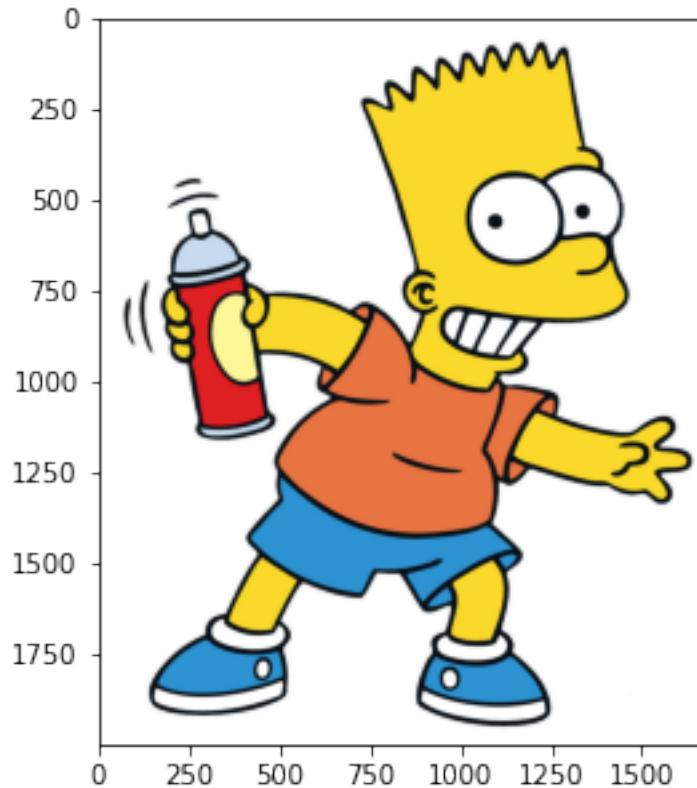
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1920x1080 at  
0x7FB119BABB50>
```



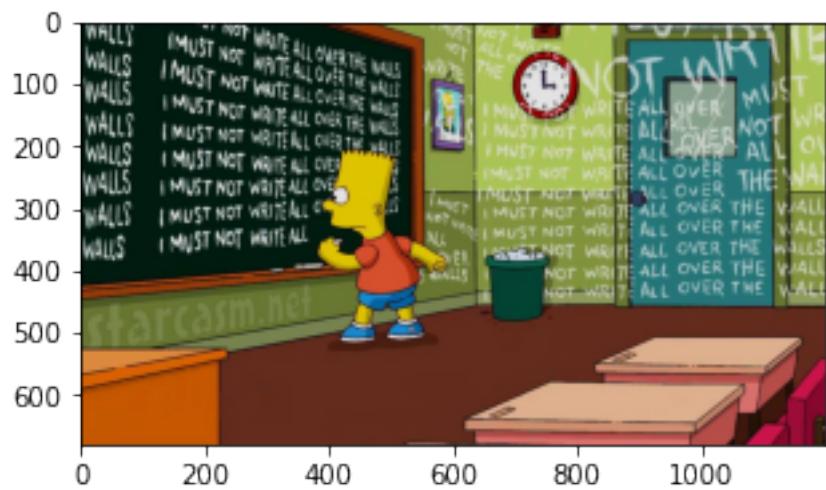
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=754x719 at 0x7FB119BABD90>
```



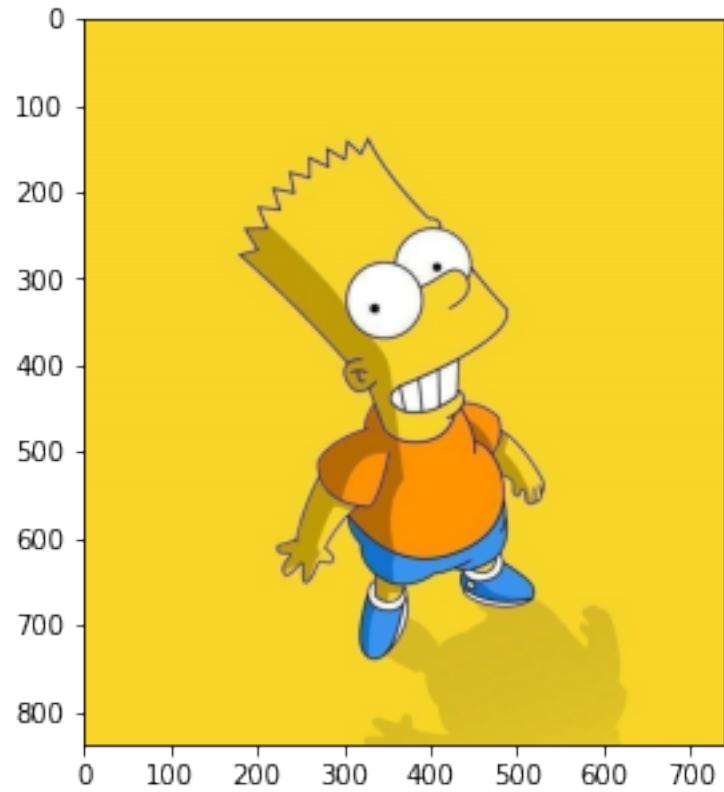
<PIL.PngImagePlugin.PngImageFile image mode=P size=1660x2000 at 0x7FB119B9D310>



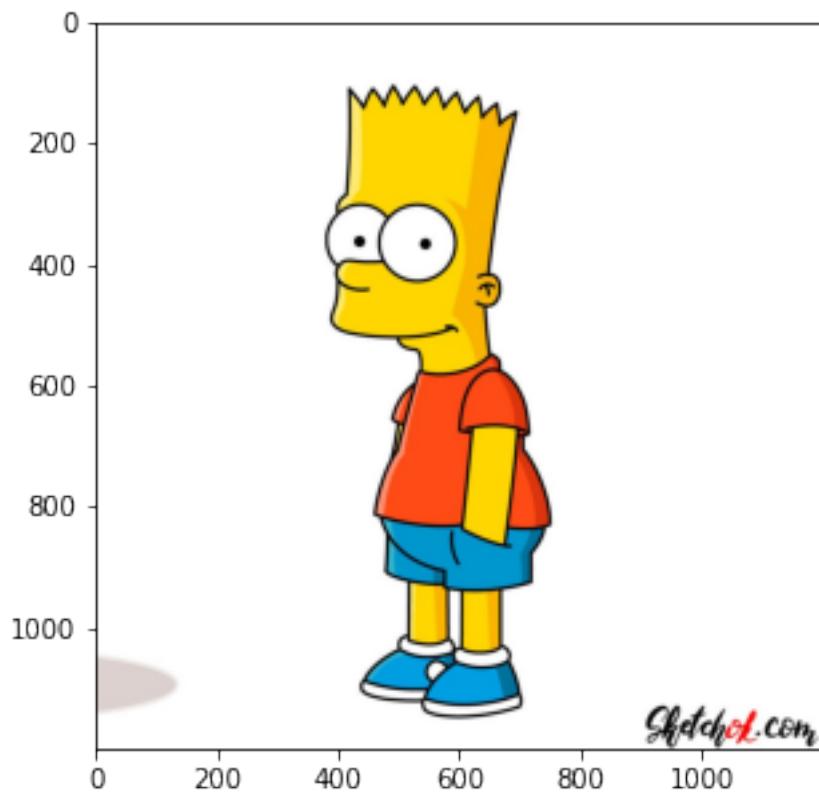
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1200x680 at 0x7FB119BABFD0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=741x838 at 0x7FB119BABC50>



```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1200x1200 at  
0x7FB119B3D050>
```

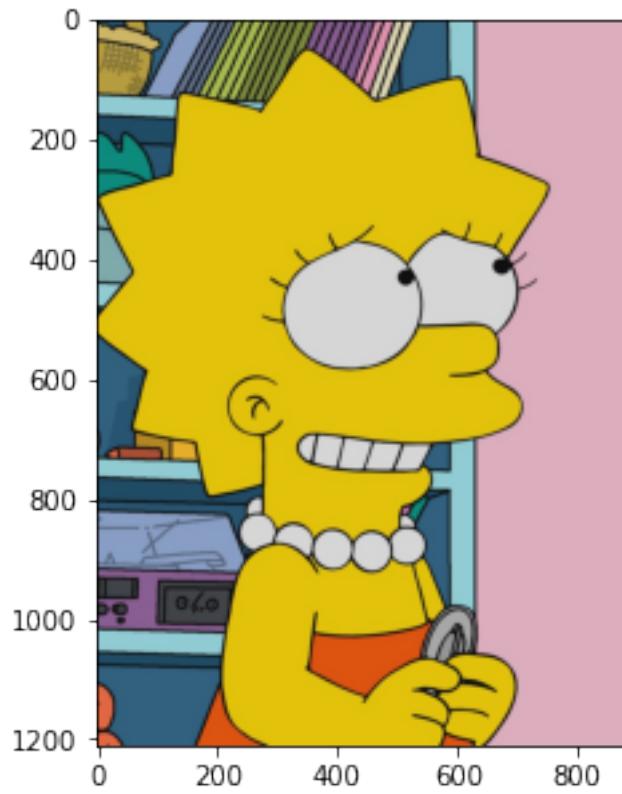


Lisa

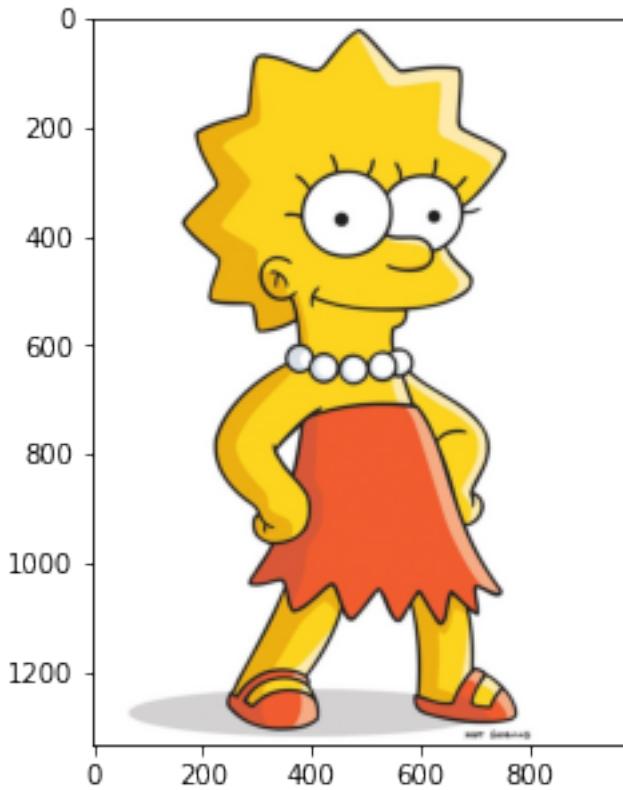
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=512x384 at 0x7FB119B3DD10>



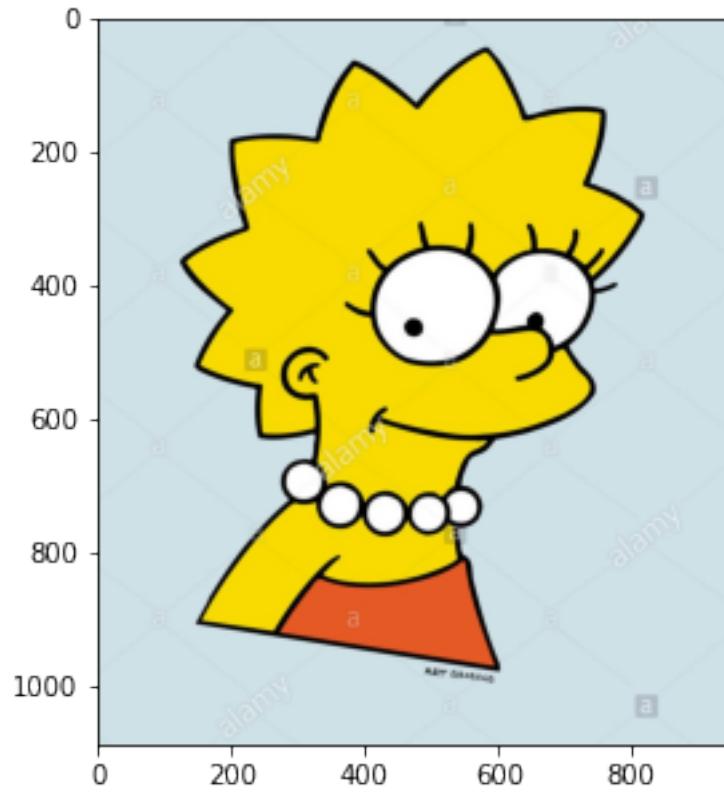
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=880x1211 at 0x7FB119B3D150>



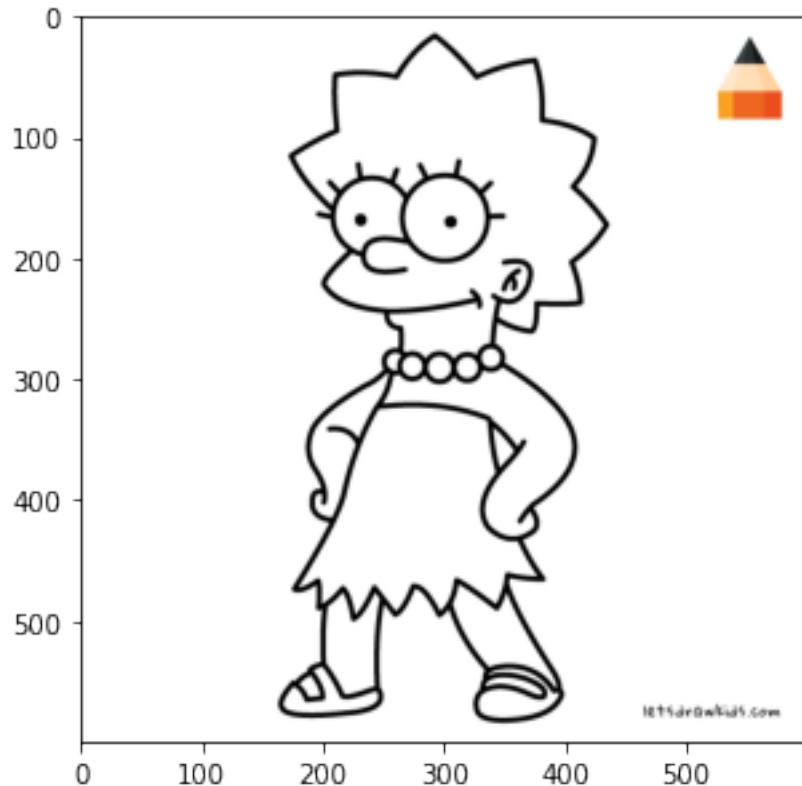
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=978x1333 at 0x7FB119B3D0D0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=945x1089 at 0x7FB119B3D1D0>



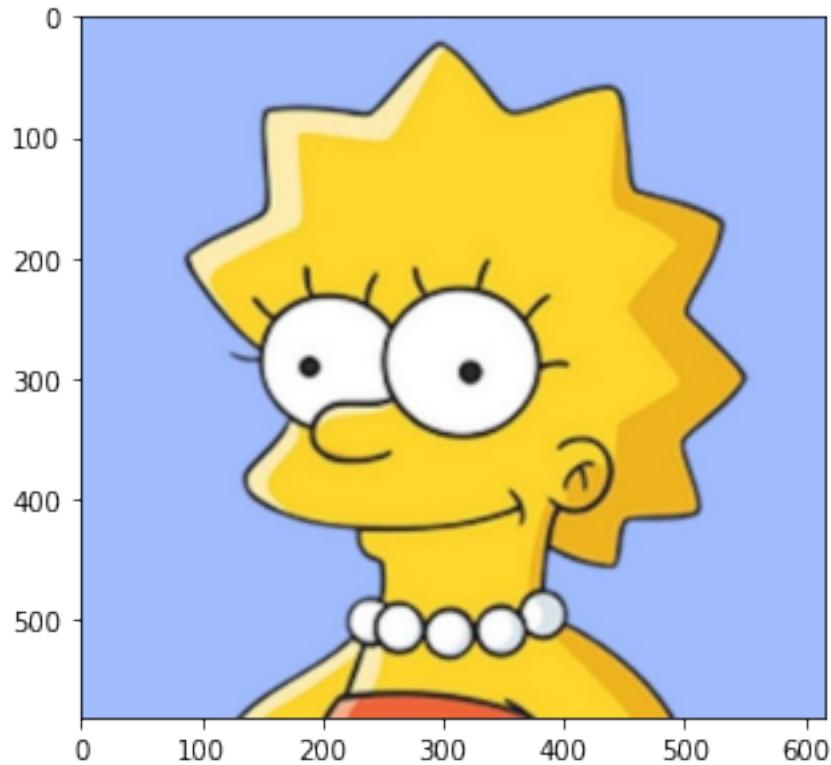
<PIL.PngImagePlugin.PngImageFile image mode=P size=600x600 at 0x7FB119B3D250>



```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=359x443 at 0x7FB119B3D2D0>
```



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=617x581 at 0x7FB119BABF50>



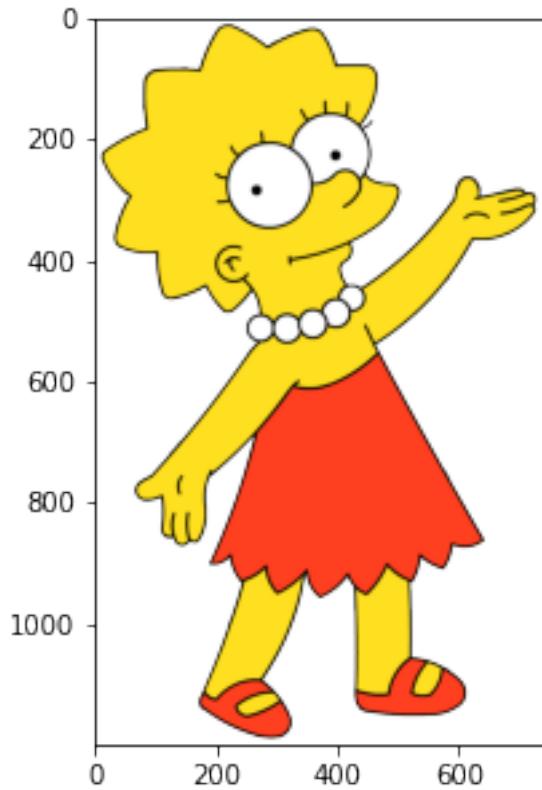
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=230x201 at 0x7FB119B3D510>



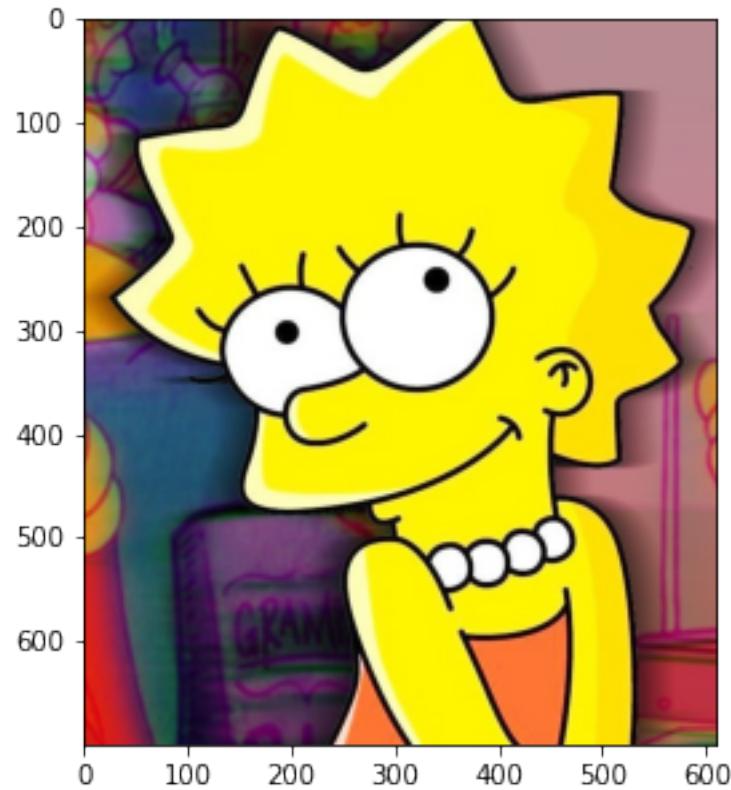
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=269x567 at 0x7FB119B3D550>



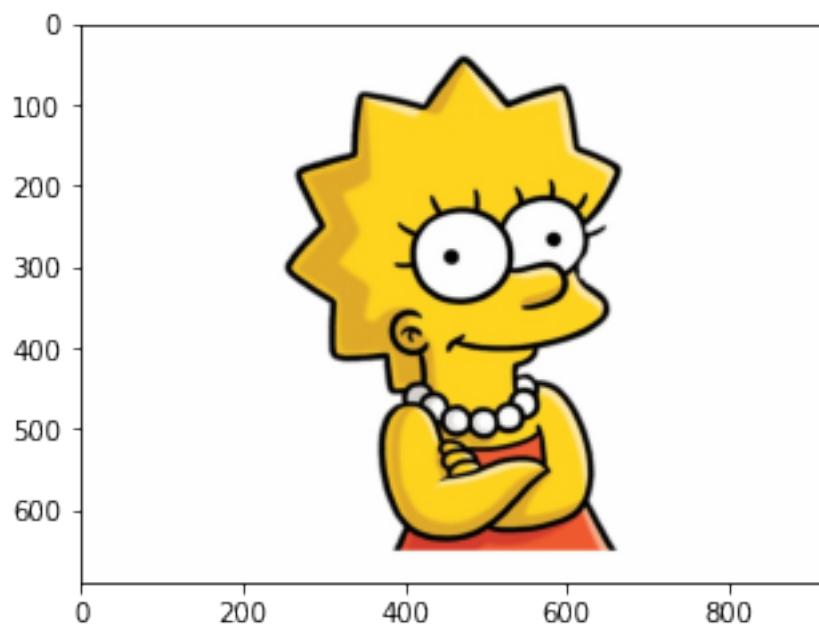
<PIL.PngImagePlugin.PngImageFile image mode=P size=740x1200 at 0x7FB119B1BE50>



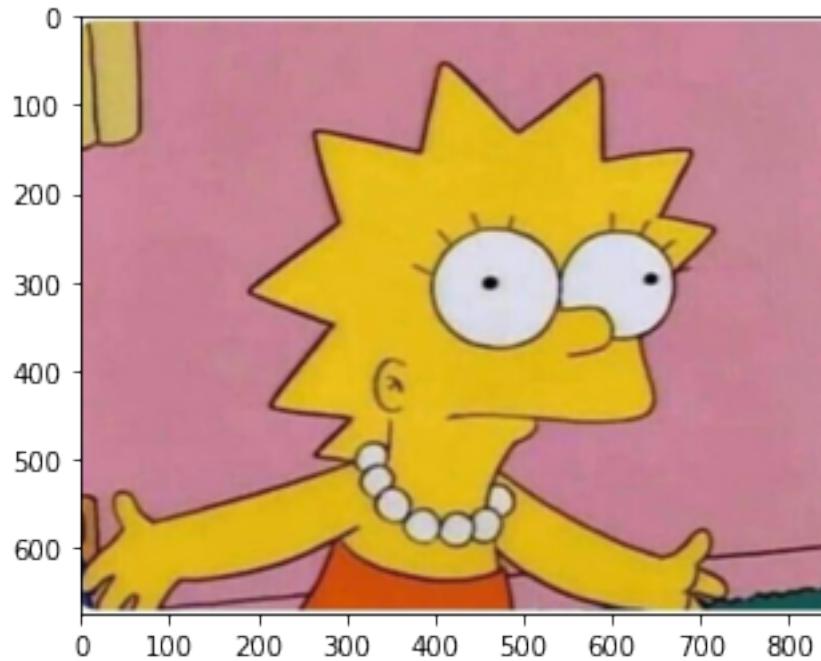
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=611x700 at 0x7FB119B3D750>



```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=920x690 at  
0x7FB119B3D710>
```



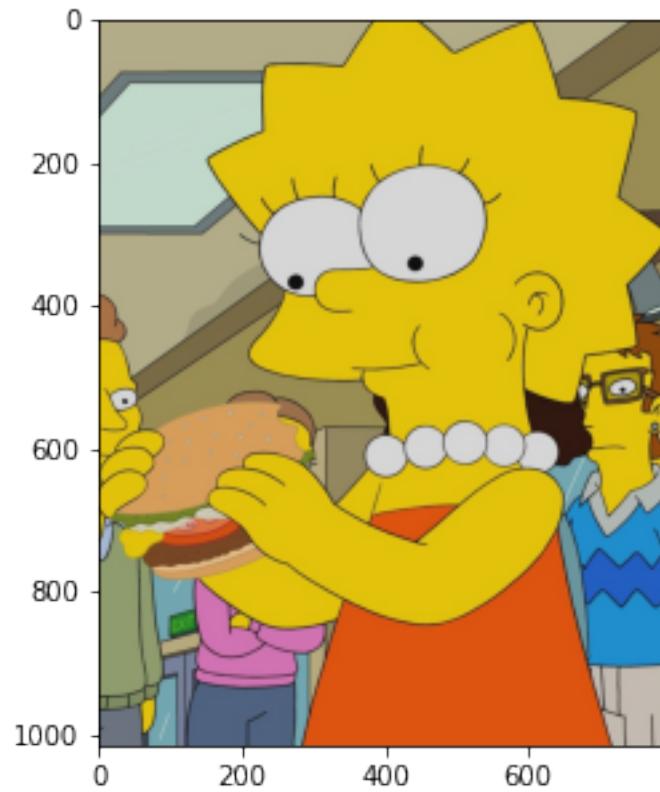
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=842x675 at 0x7FB119B3D890>



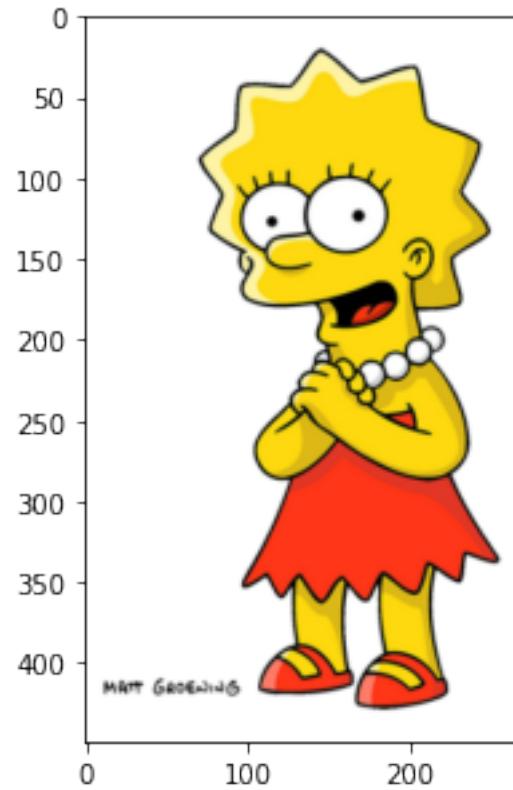
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1638x1121 at 0x7FB119B3D790>



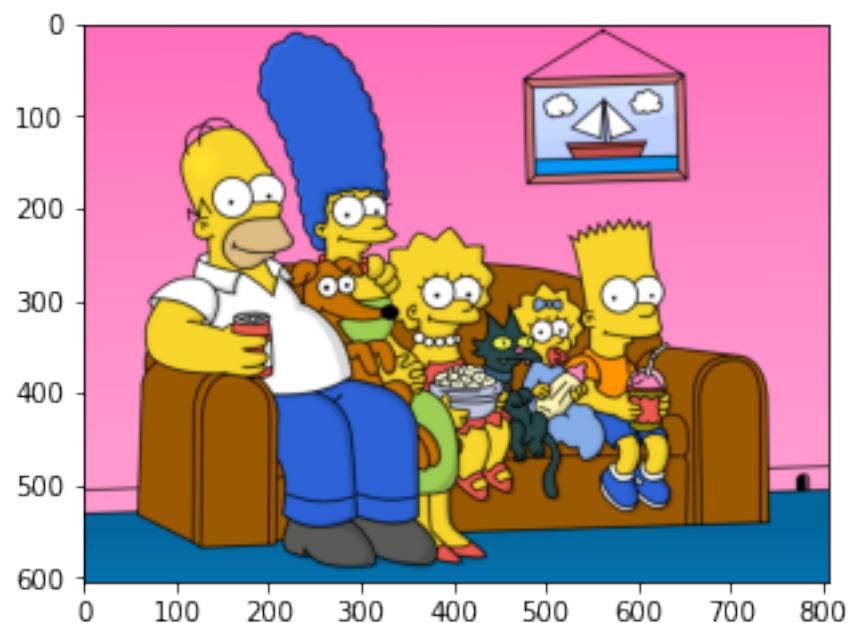
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=788x1015 at 0x7FB119B3D910>



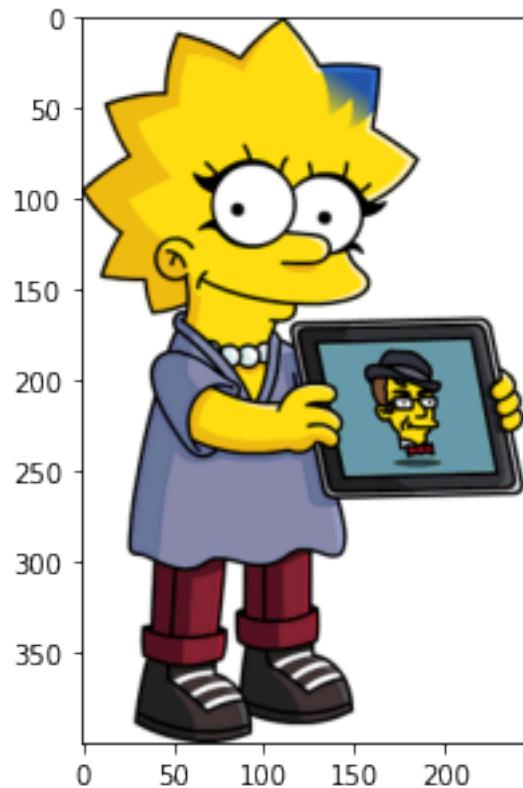
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=268x449 at 0x7FB119BAB650>



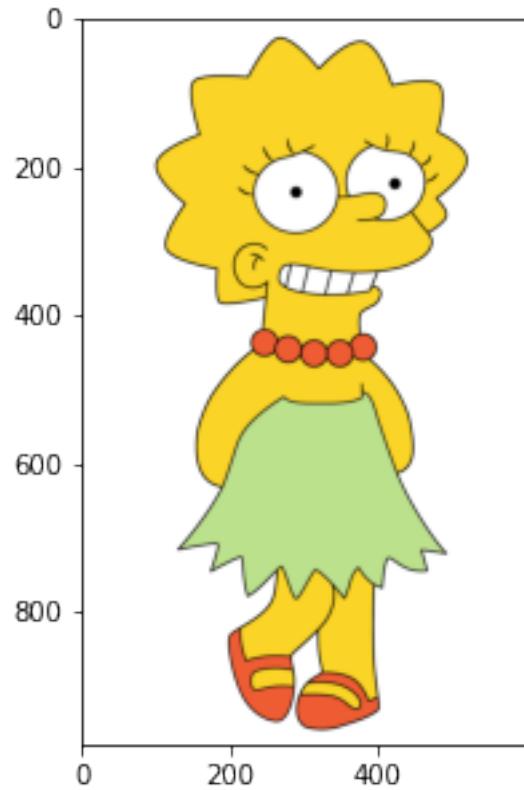
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=807x605 at 0x7FB119BABB90>



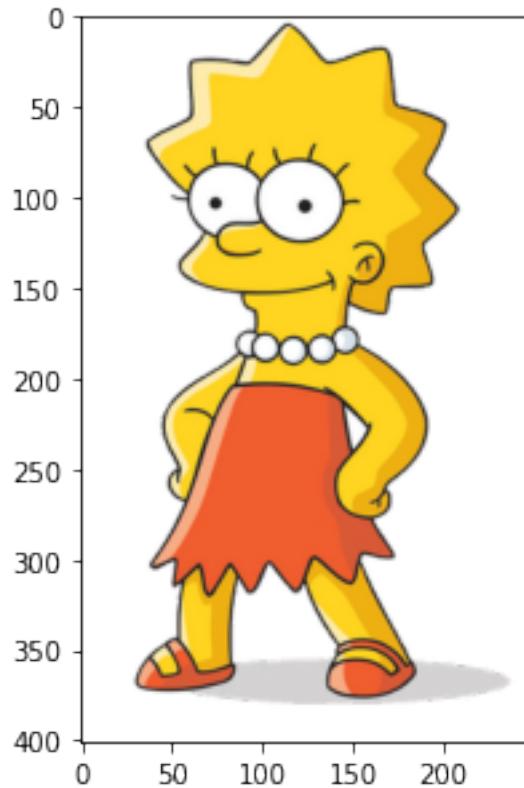
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=244x400 at 0x7FB119B3DA10>



<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=600x979 at 0x7FB119B3DA50>



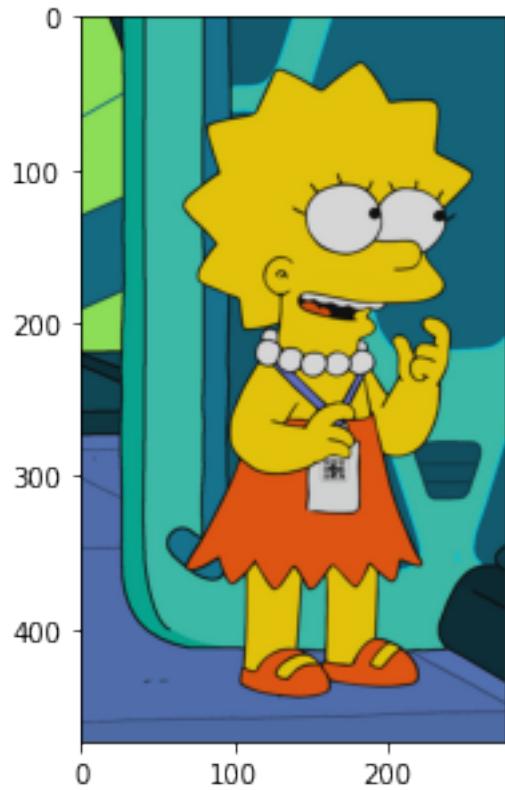
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=247x401 at 0x7FB119BAB150>



```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=495x630 at 0x7FB119B7E8D0>
```



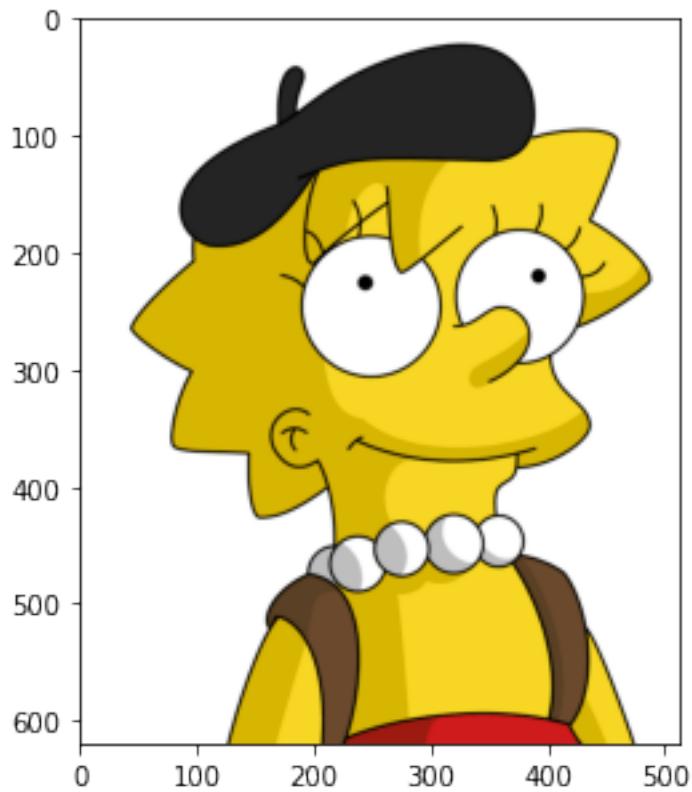
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=277x474 at 0x7FB119B3D9D0>



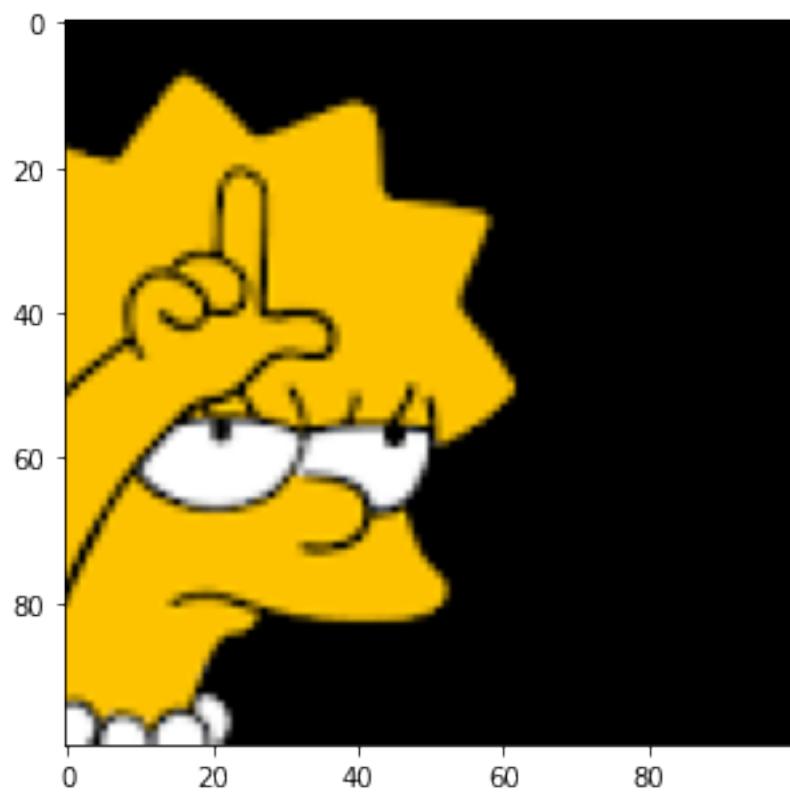
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=259x412 at 0x7FB119B3DD0>



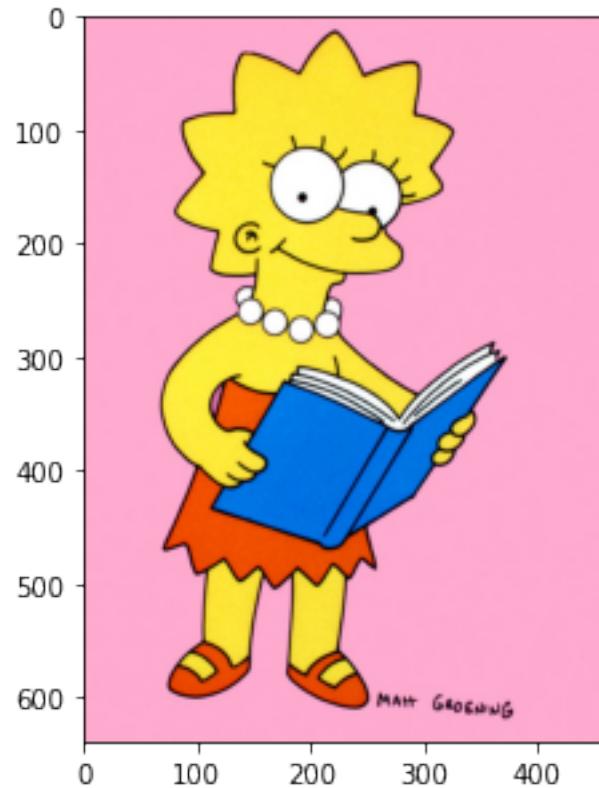
<PIL.PngImagePlugin.PngImageFile image mode=P size=514x620 at 0x7FB119B3DFD0>



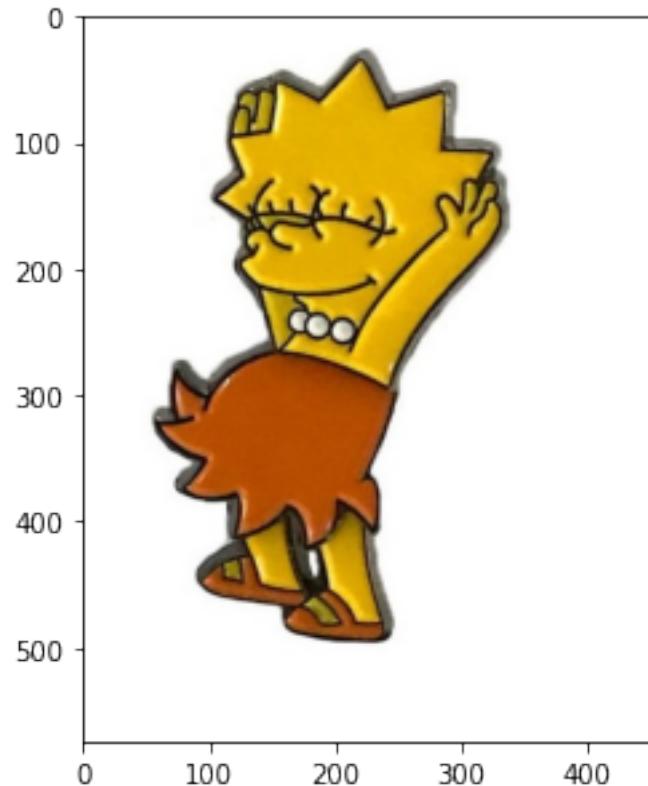
<PIL.PngImagePlugin.PngImageFile image mode=P size=100x100 at 0x7FB119B3D690>



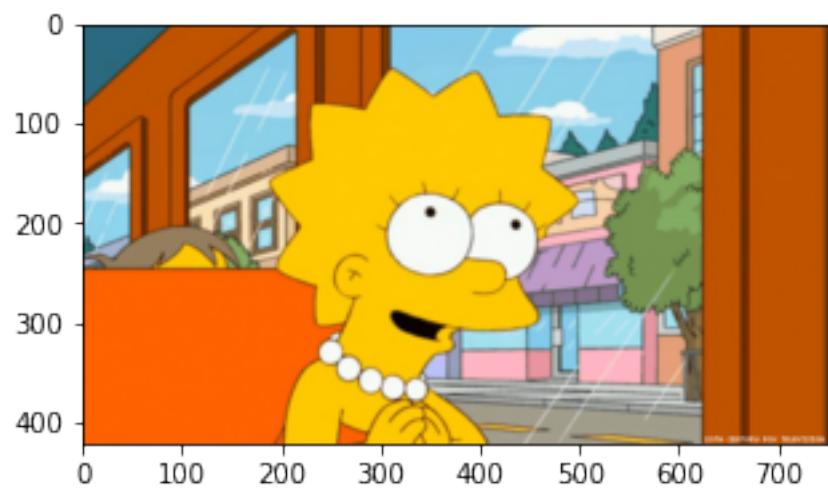
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=461x640 at 0x7FB119B4D410>



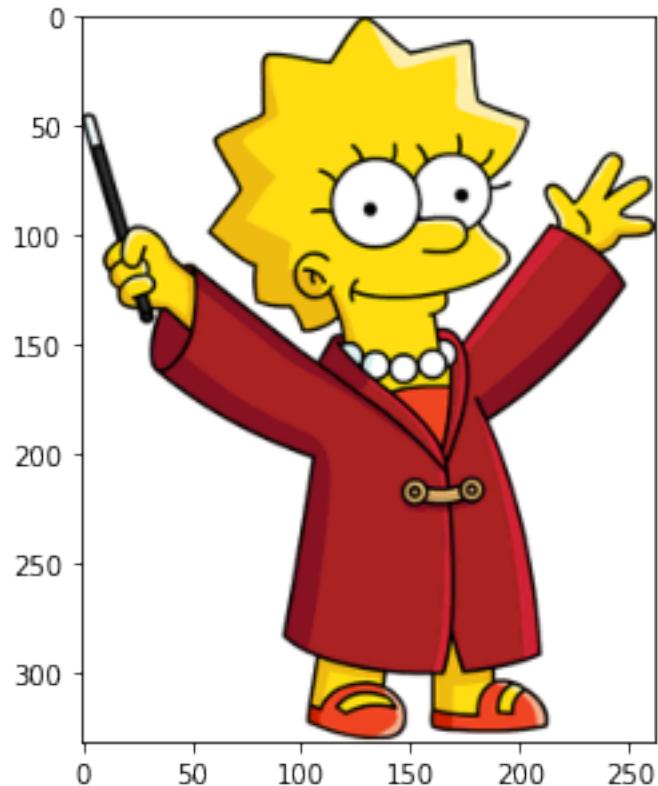
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=451x575 at 0x7FB119B4D310>



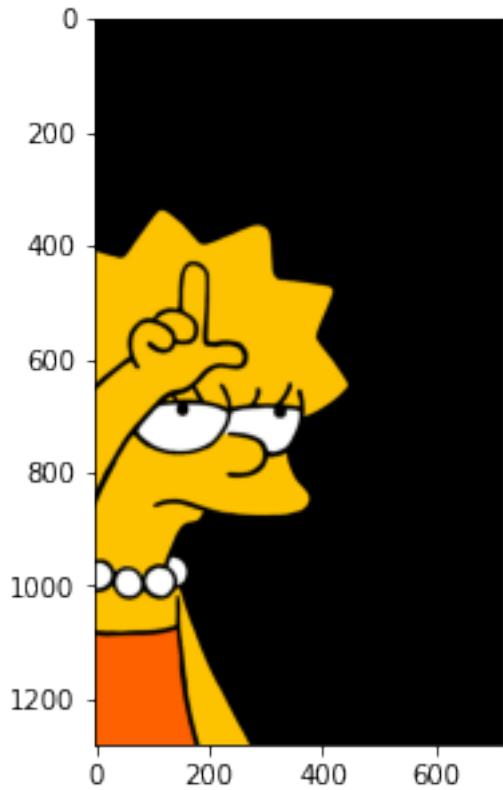
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=750x422 at 0x7FB119B4D450>



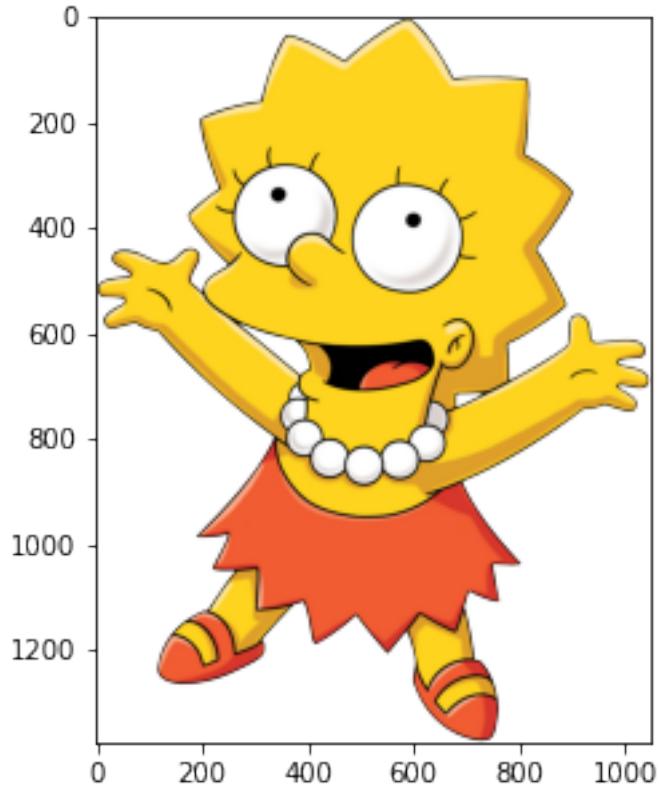
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=262x332 at 0x7FB119B4D4D0>



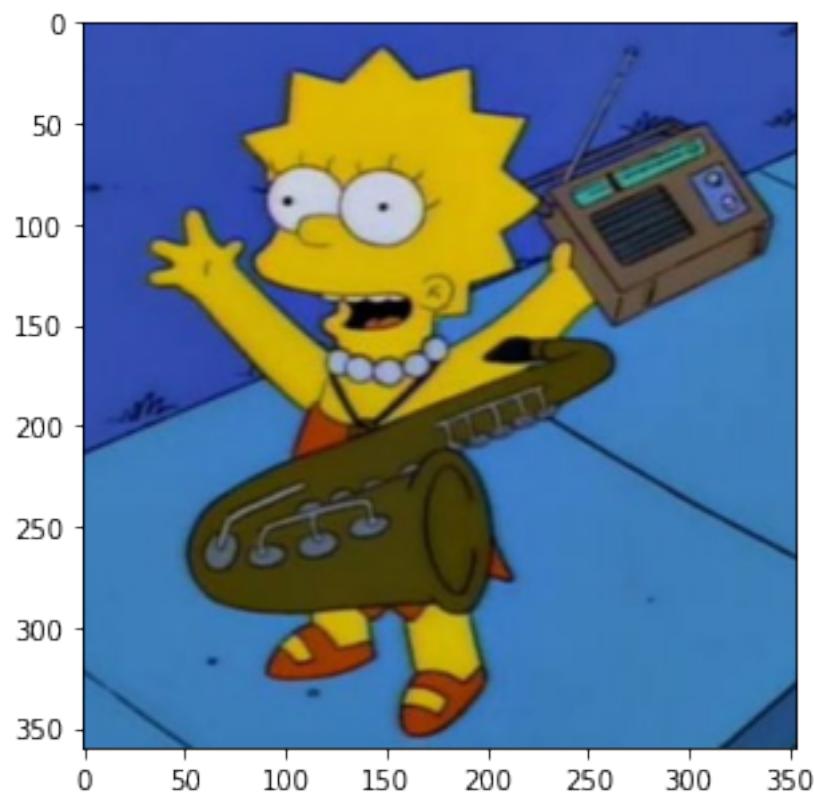
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=720x1280 at  
0x7FB119B4D550>
```



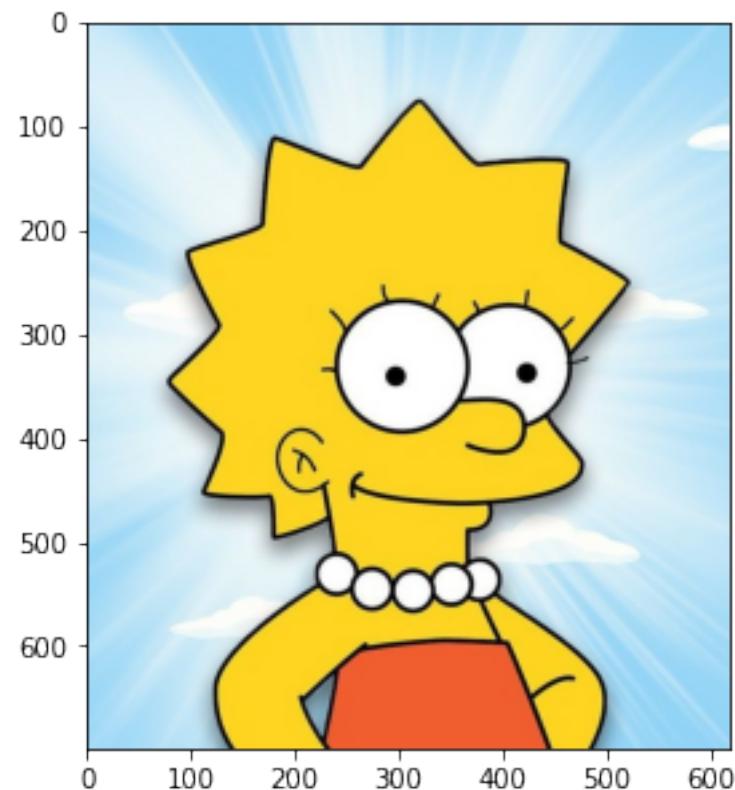
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1052x1375 at 0x7FB119B3DE50>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=354x360 at 0x7FB119B3DE90>



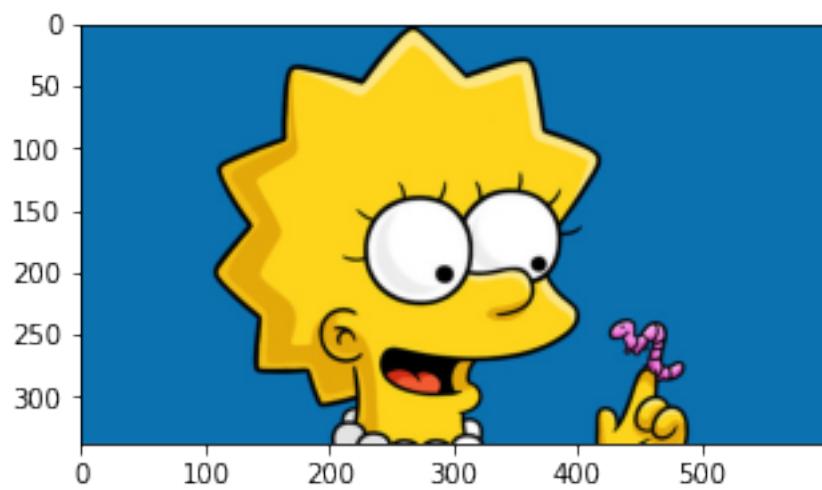
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=619x699 at 0x7FB119B4D6D0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=301x407 at 0x7FB119B4D750>



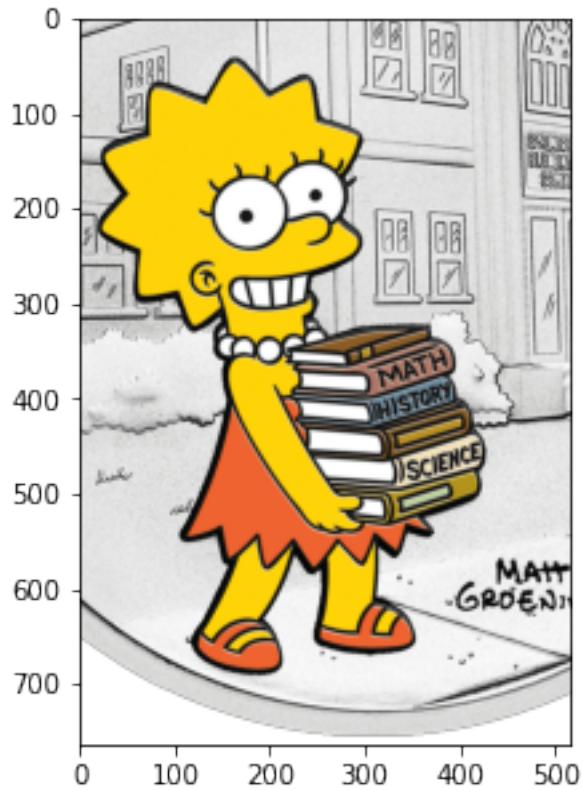
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=600x338 at 0x7FB119B4D7D0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=359x443 at 0x7FB119B4D850>



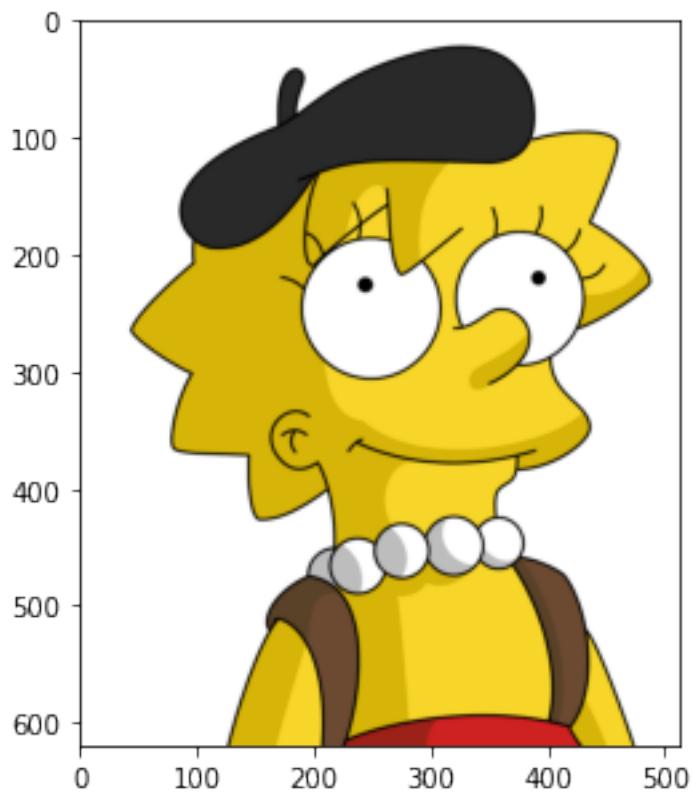
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=518x764 at 0x7FB119B4D8D0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=630x1200 at 0x7FB119B3DE10>



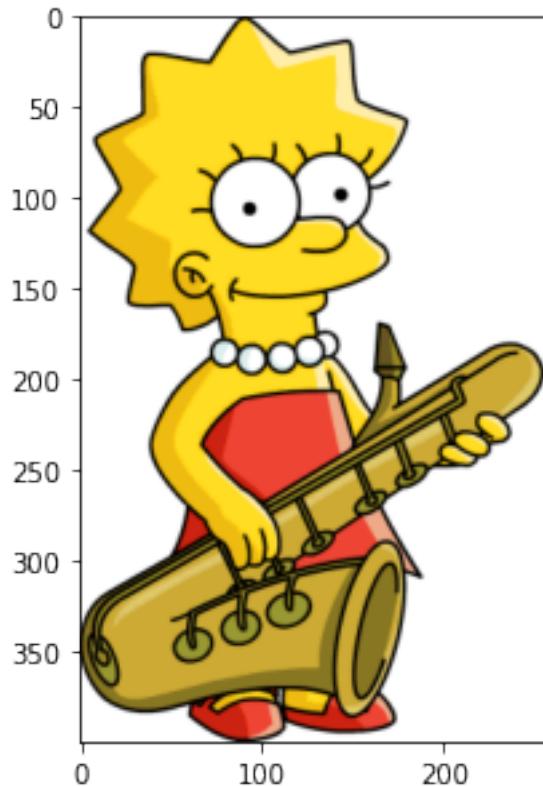
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=514x620 at 0x7FB119B4DA50>



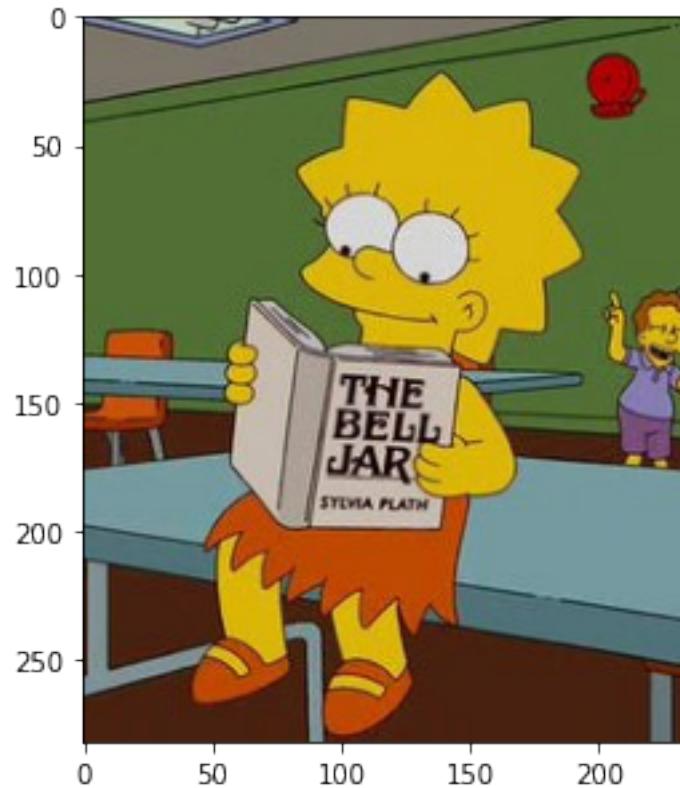
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1586x2139 at  
0x7FB119B4DA90>
```



<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=256x400 at 0x7FB119B4DB10>



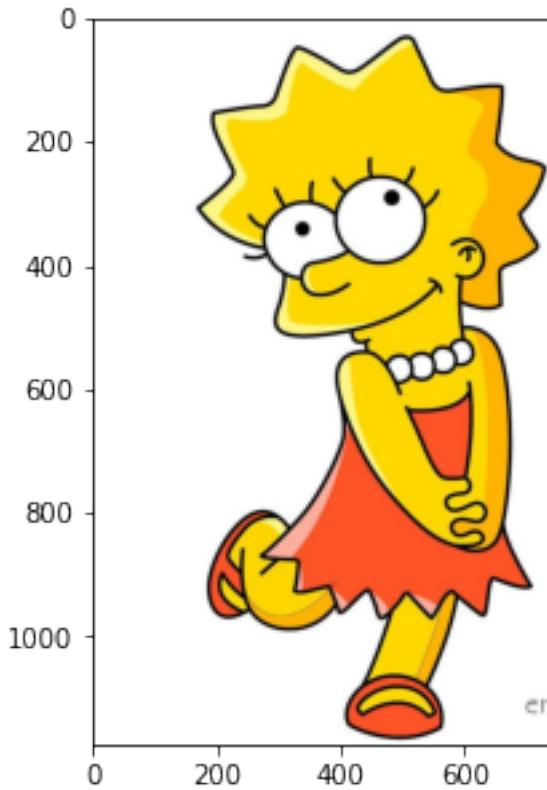
```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=234x282 at  
0x7FB119B3DC10>
```



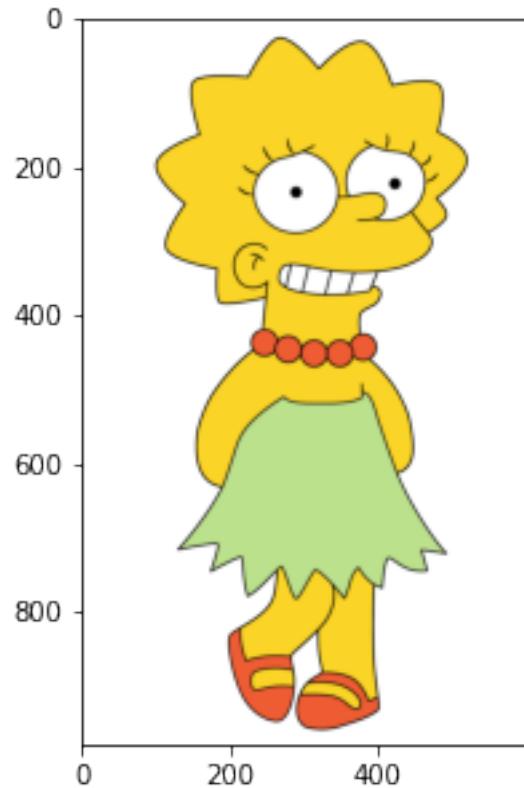
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=696x442 at 0x7FB119B4DD90>



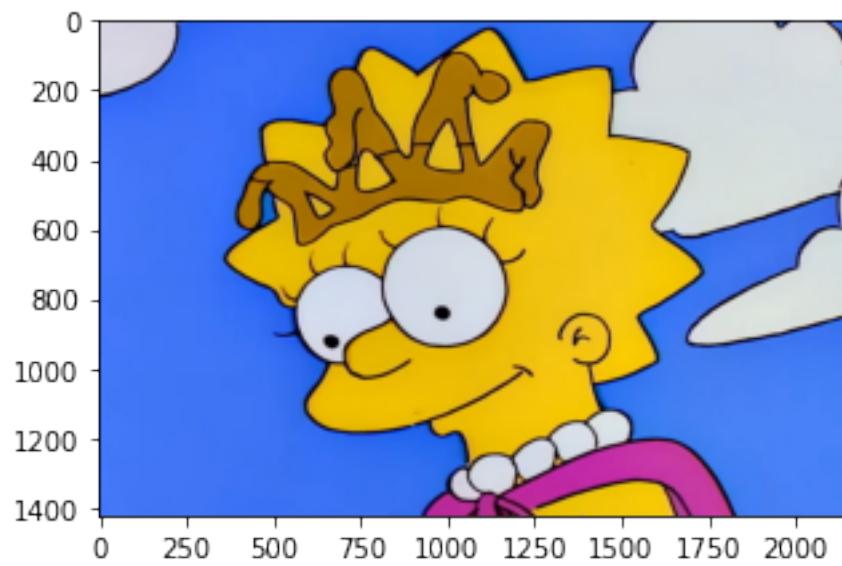
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=736x1175 at 0x7FB119B4D610>



<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=600x979 at 0x7FB119B4DE10>



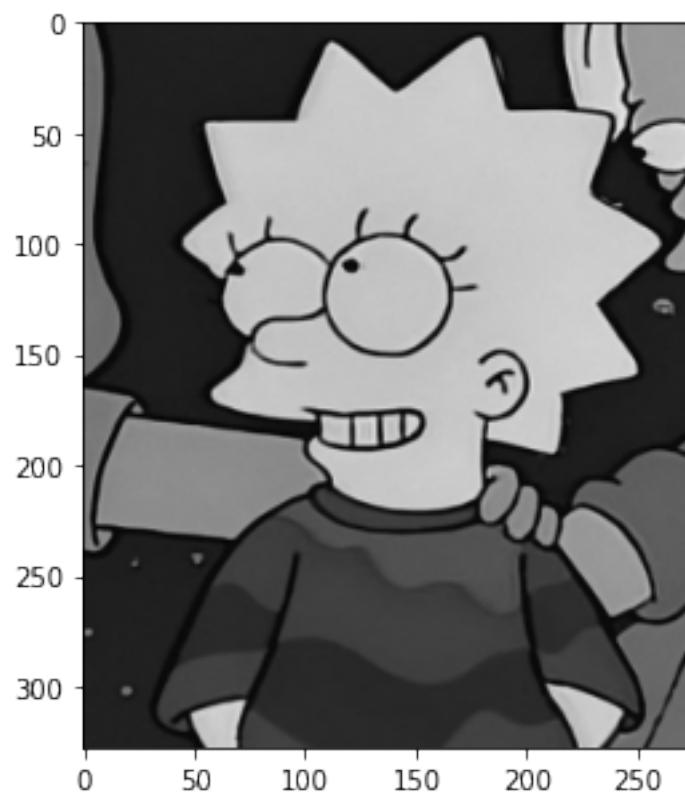
```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=2141x1423 at  
0x7FB119B4DE90>
```



<PIL.PngImagePlugin.PngImageFile image mode=P size=432x232 at 0x7FB119B5E0D0>



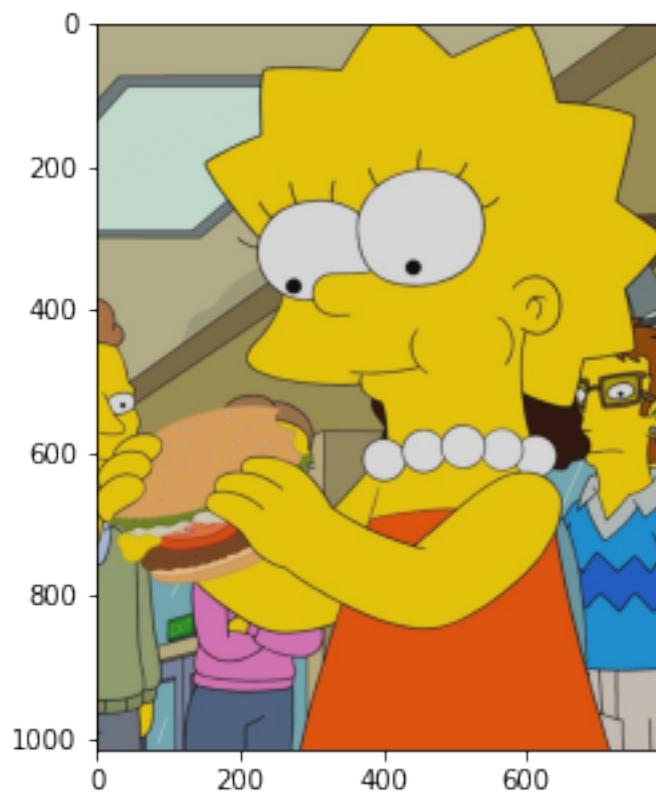
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=272x328 at 0x7FB119B4D390>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=594x484 at 0x7FB119B4DF50>

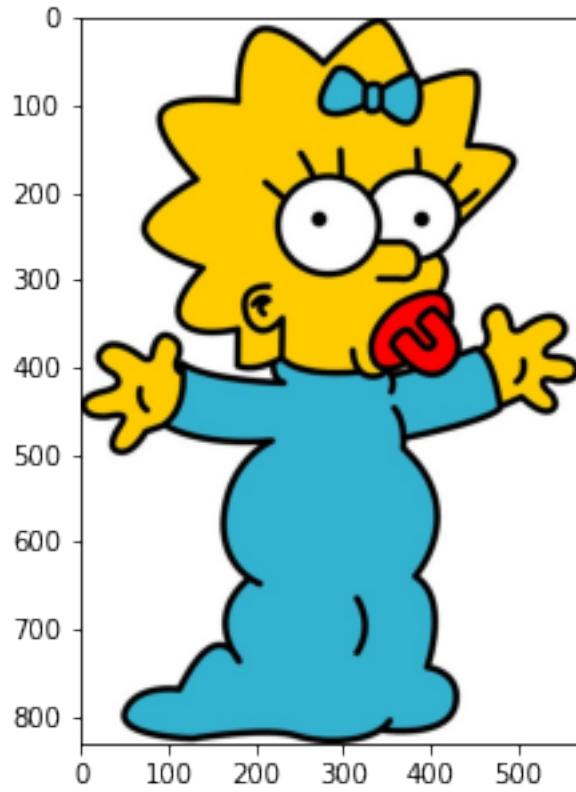


<PIL.PngImagePlugin.PngImageFile image mode=RGB size=788x1015 at 0x7FB119B5E290>

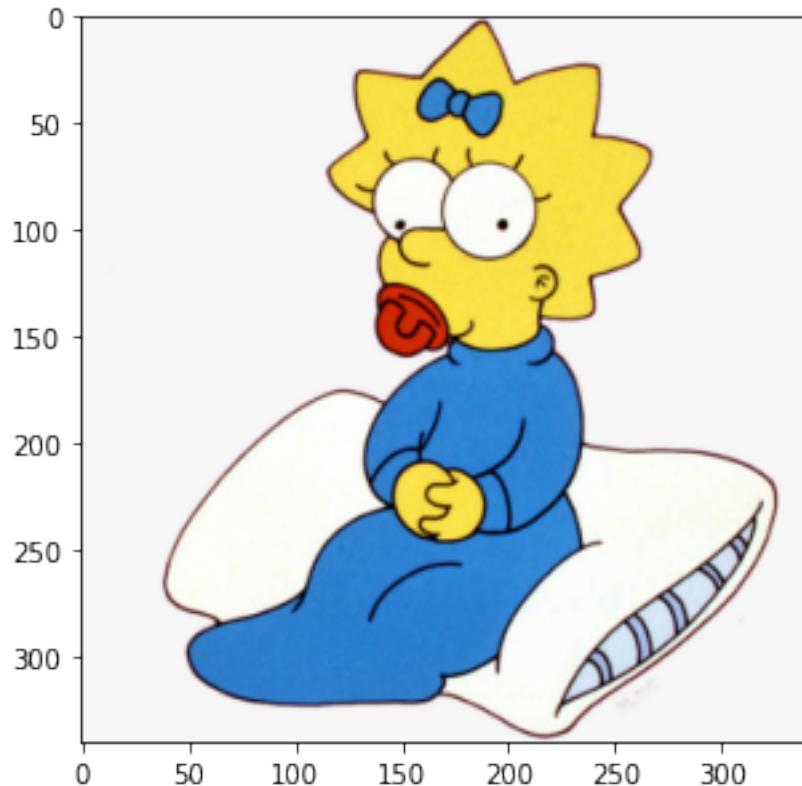


Maggie

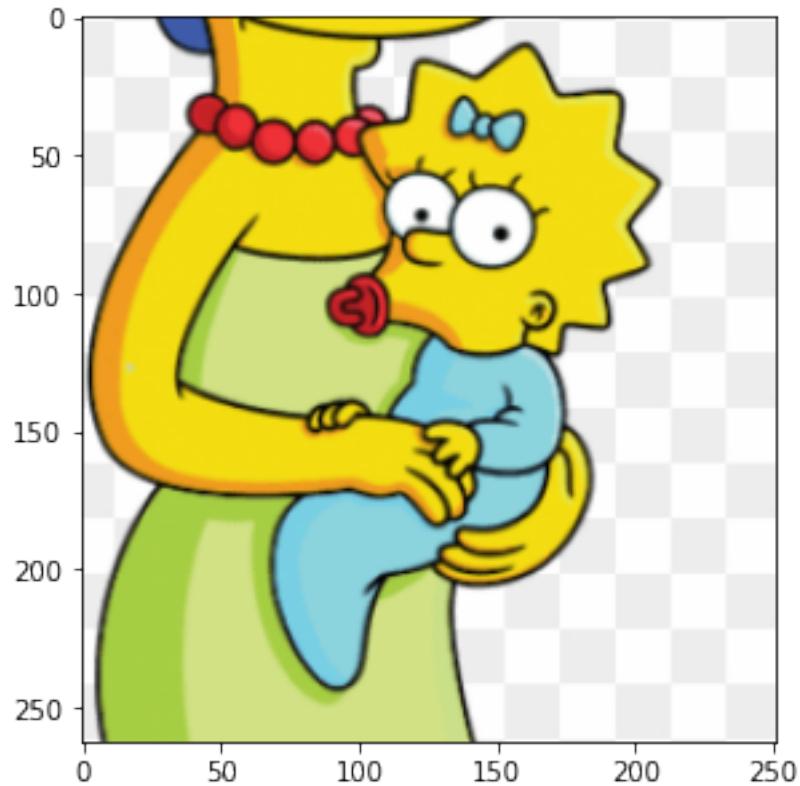
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=570x831 at 0x7FB119B5EF10>



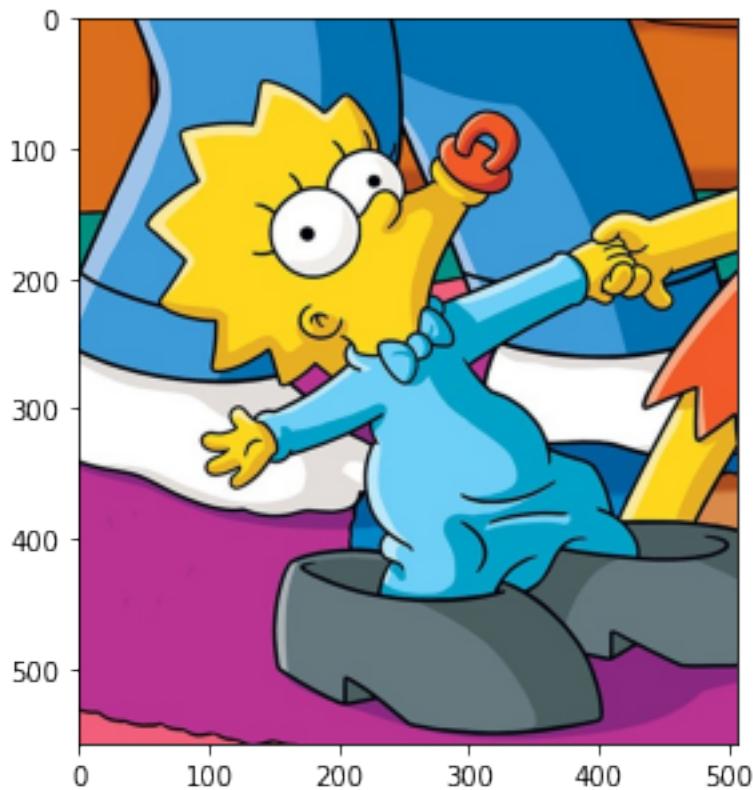
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=340x340 at 0x7FB119B5E350>



<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=251x263 at 0x7FB119B4DF10>



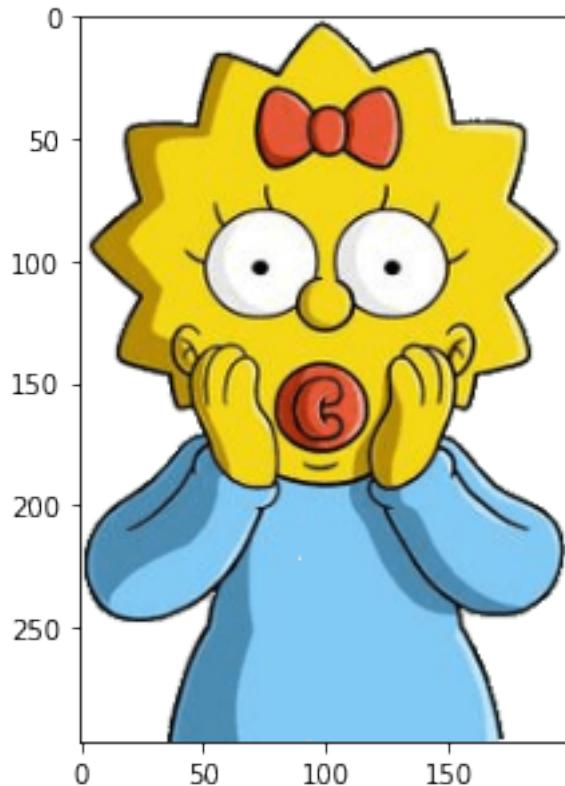
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=507x558 at 0x7FB119B4DC10>



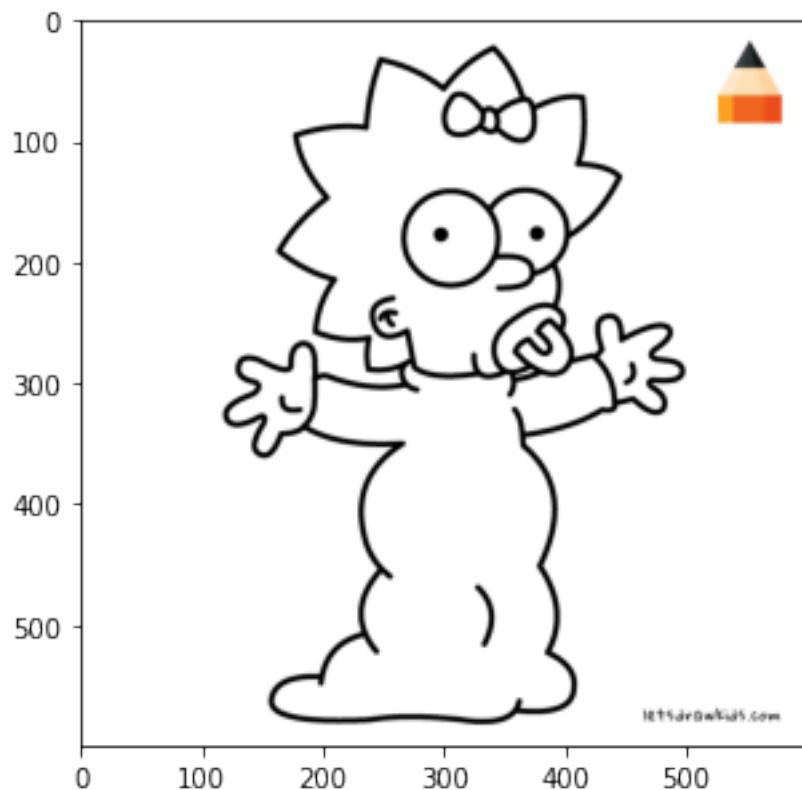
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1000x561 at 0x7FB119B5E4D0>



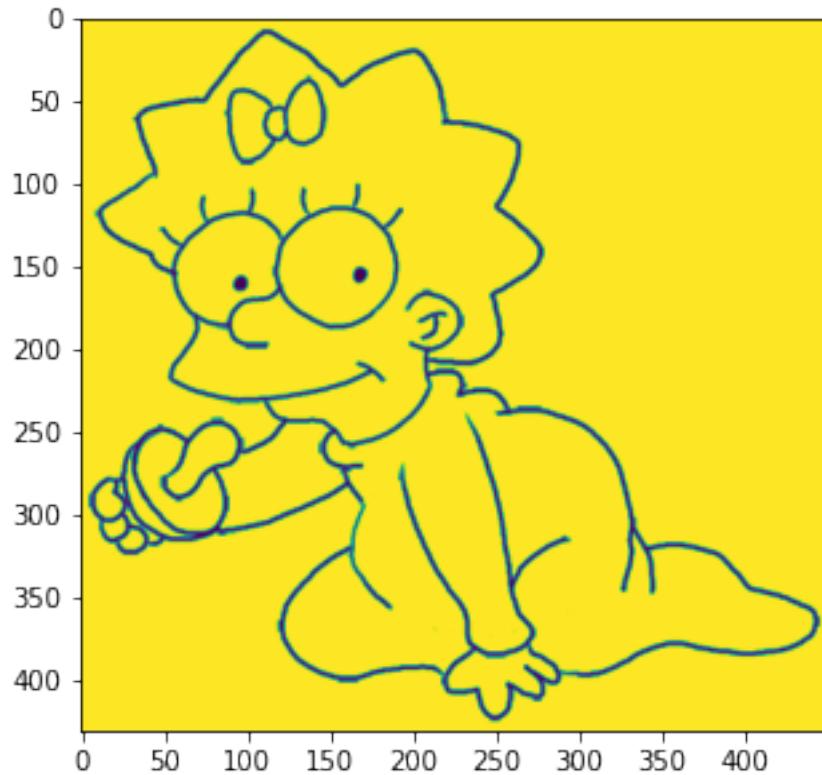
<PIL.PngImagePlugin.PngImageFile image mode=P size=200x297 at 0x7FB119B5E550>



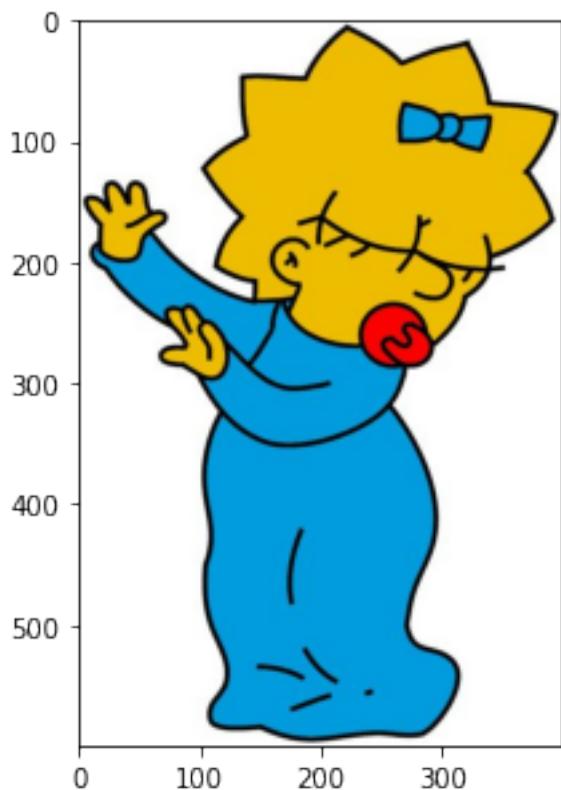
```
<PIL.PngImagePlugin.PngImageFile image mode=P size=600x600 at 0x7FB119B5E710>
```



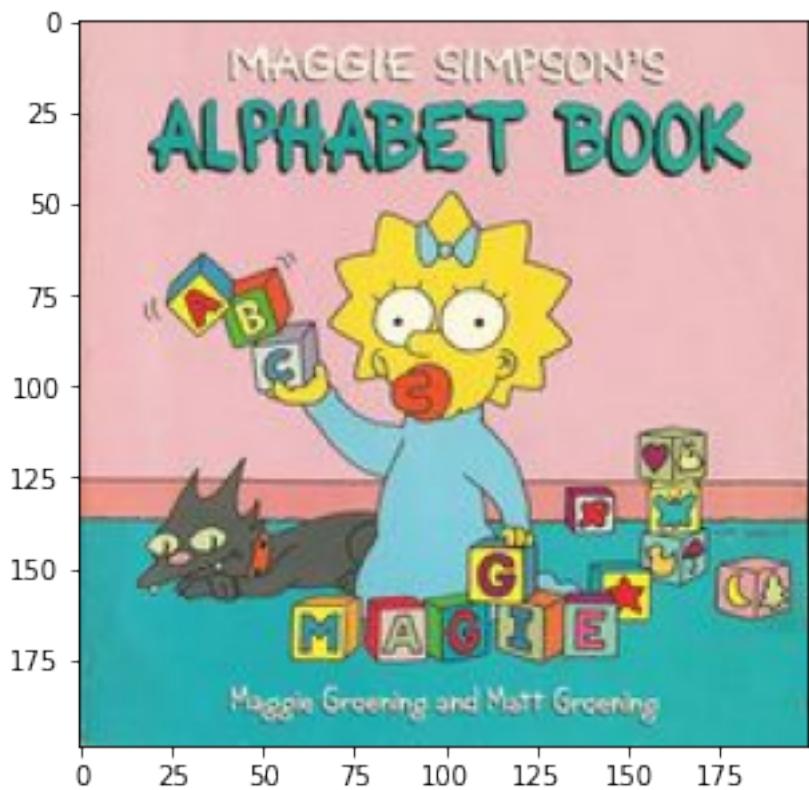
<PIL.PngImagePlugin.PngImageFile image mode=L size=450x431 at 0x7FB119B5E7D0>



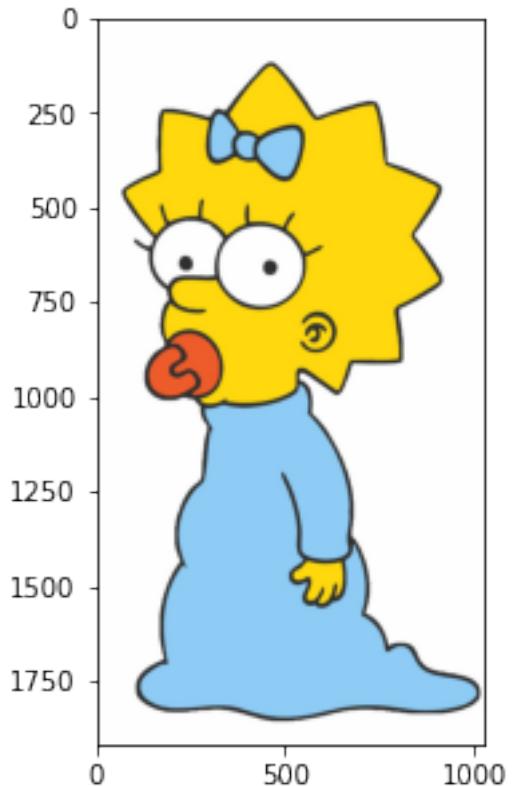
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=400x600 at 0x7FB119B5E650>



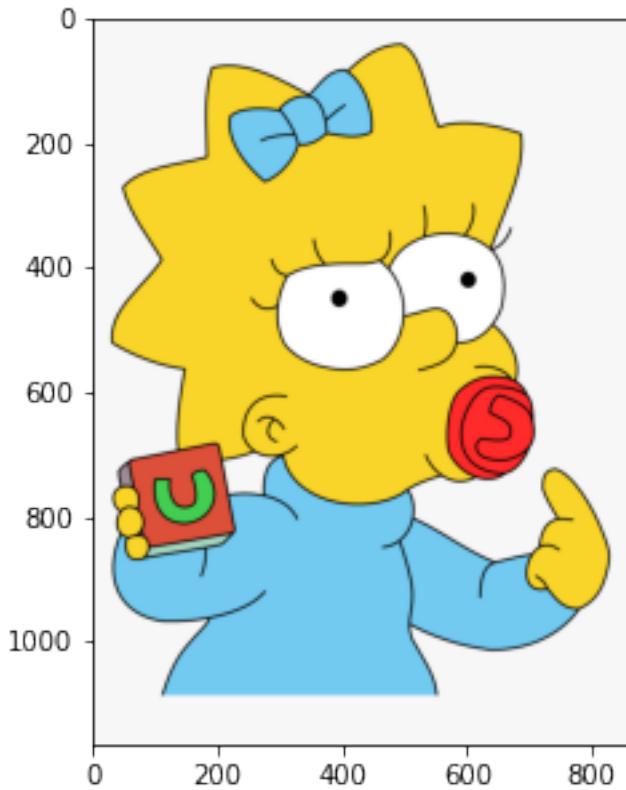
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=200x199 at 0x7FB119B5E510>



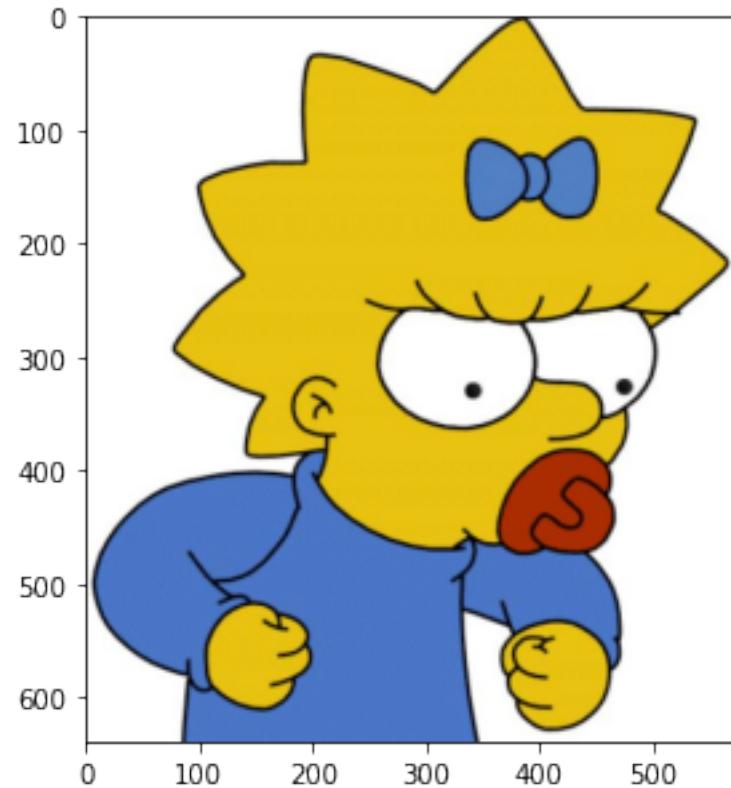
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1024x1919 at 0x7FB119B5E890>



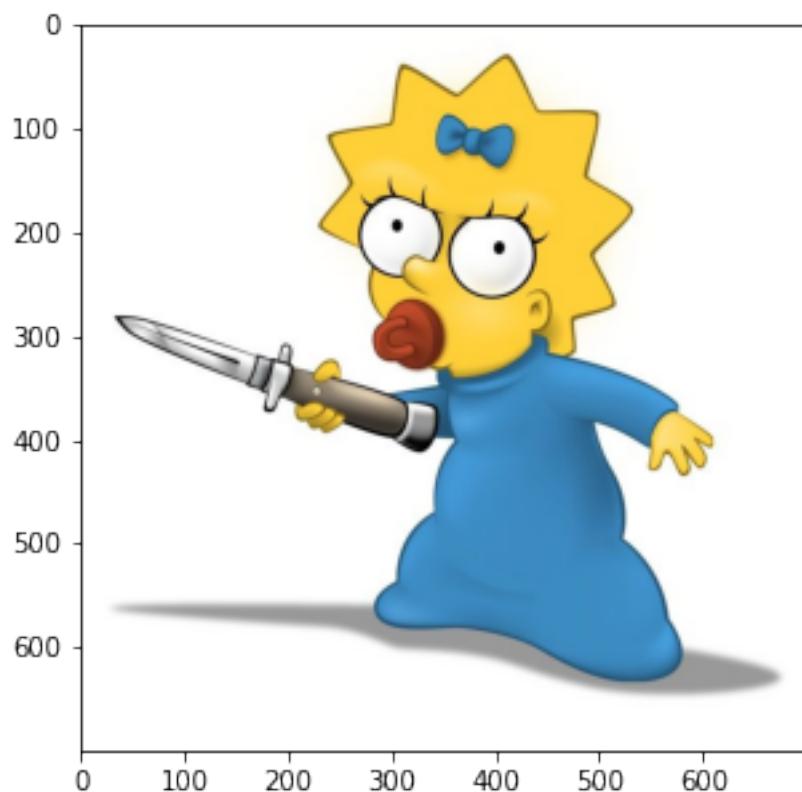
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=860x1166 at  
0x7FB119B5E850>
```



<PIL.PngImagePlugin.PngImageFile image mode=P size=576x640 at 0x7FB119B5E950>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=700x700 at 0x7FB119B5EAD0>



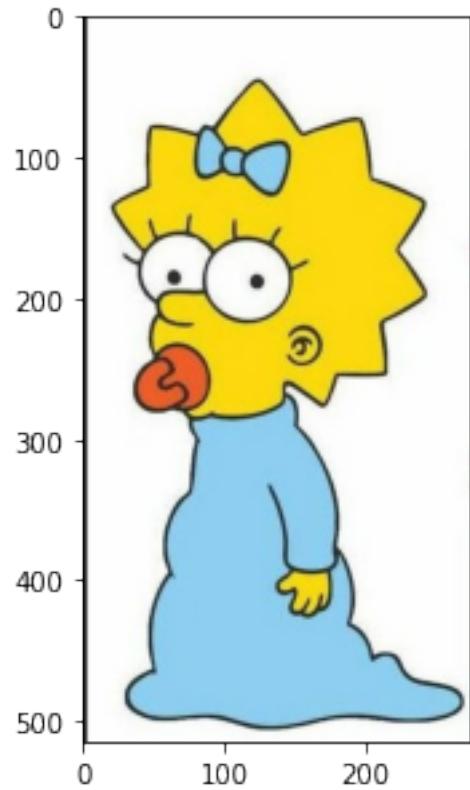
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=83x136 at 0x7FB119B5E210>



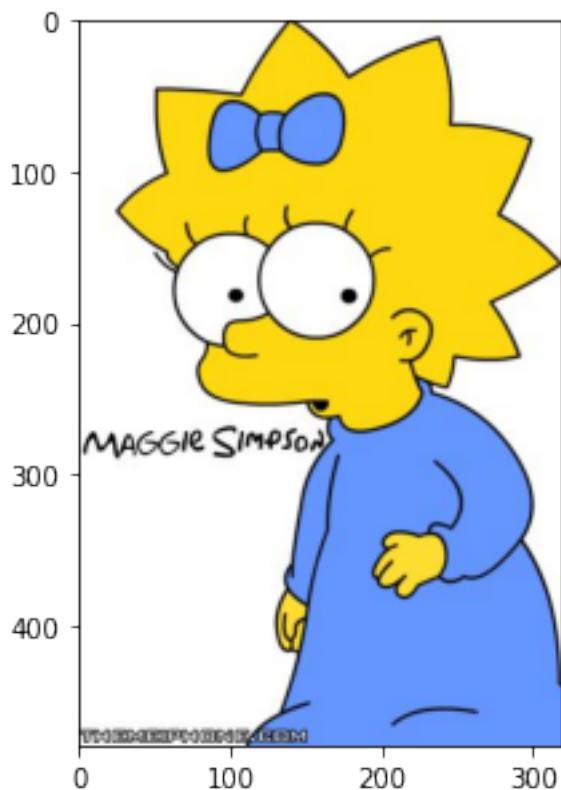
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=284x342 at 0x7FB119B5EB10>



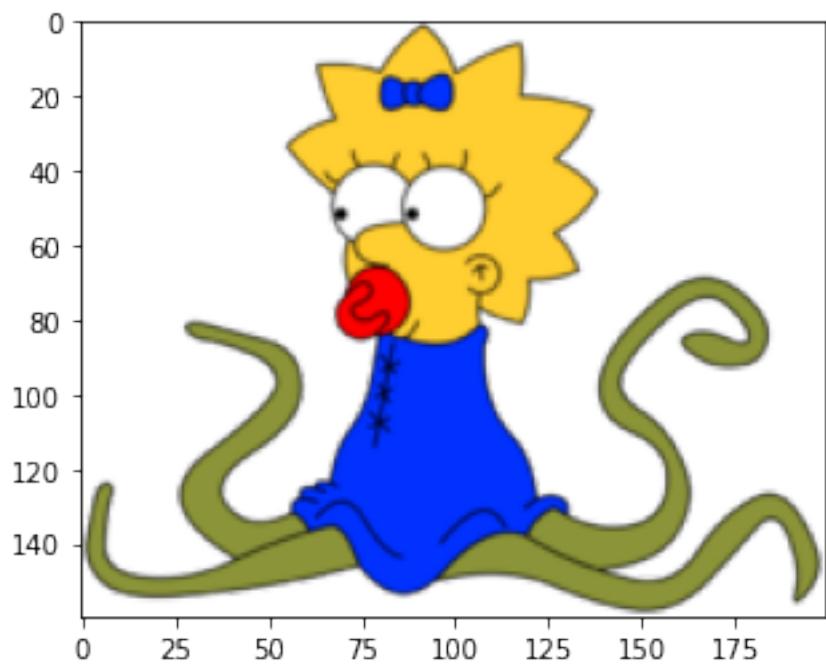
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=274x515 at 0x7FB119B5EB90>



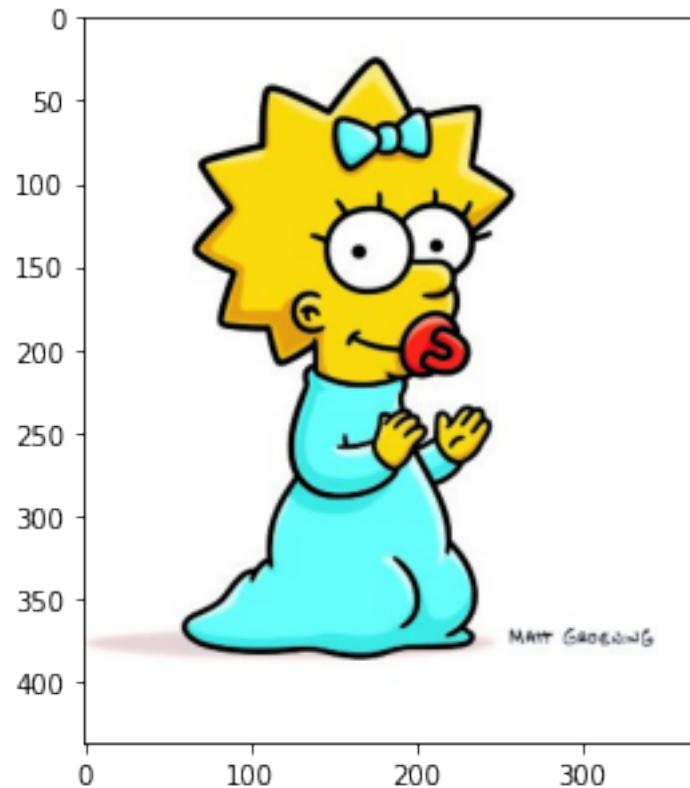
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=320x480 at 0x7FB119B5EC10>



<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=200x160 at 0x7FB119B5EC90>



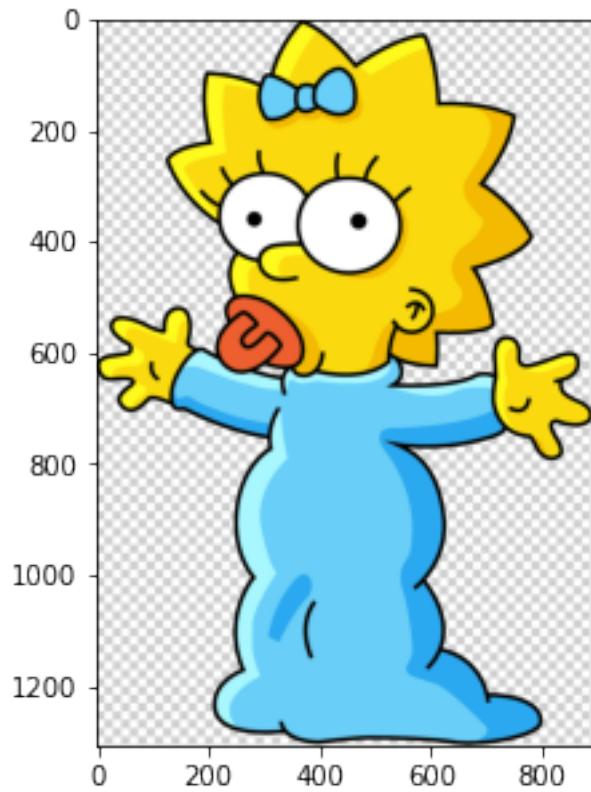
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=367x437 at 0x7FB119B4D290>



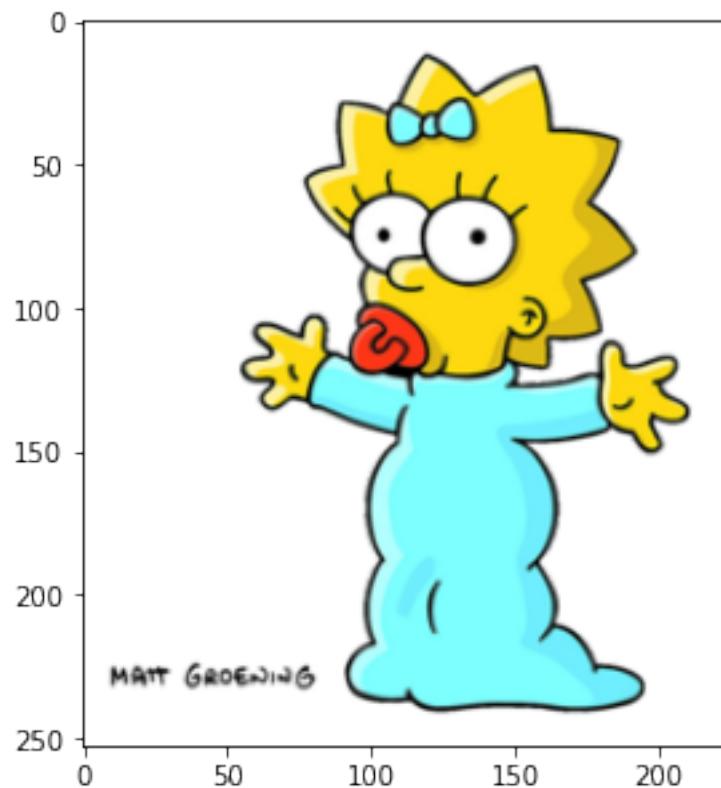
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1000x562 at 0x7FB119B4DED0>



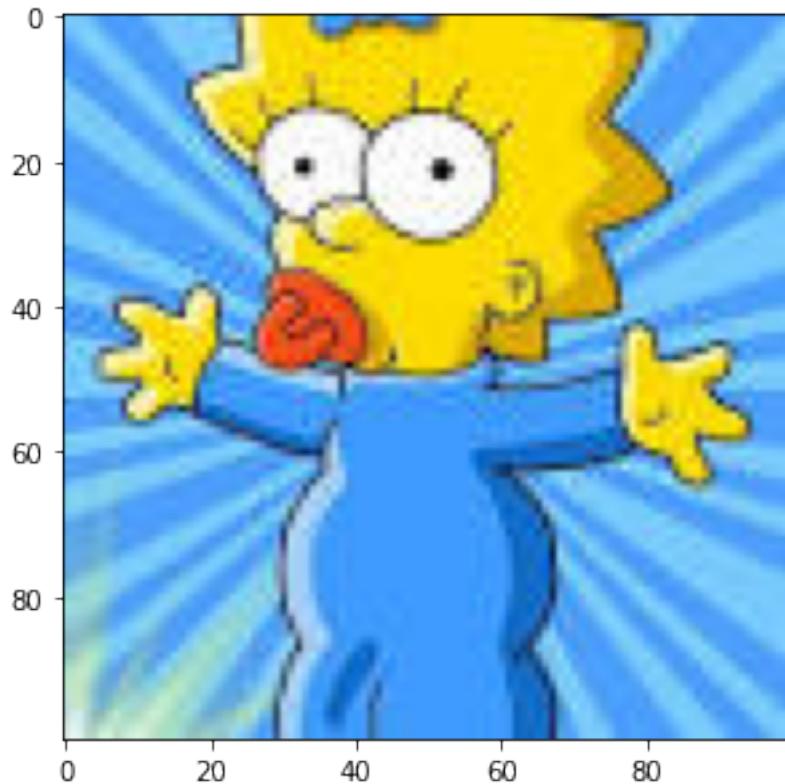
<PIL.PngImagePlugin.PngImageFile image mode=P size=890x1307 at 0x7FB119B5EE10>



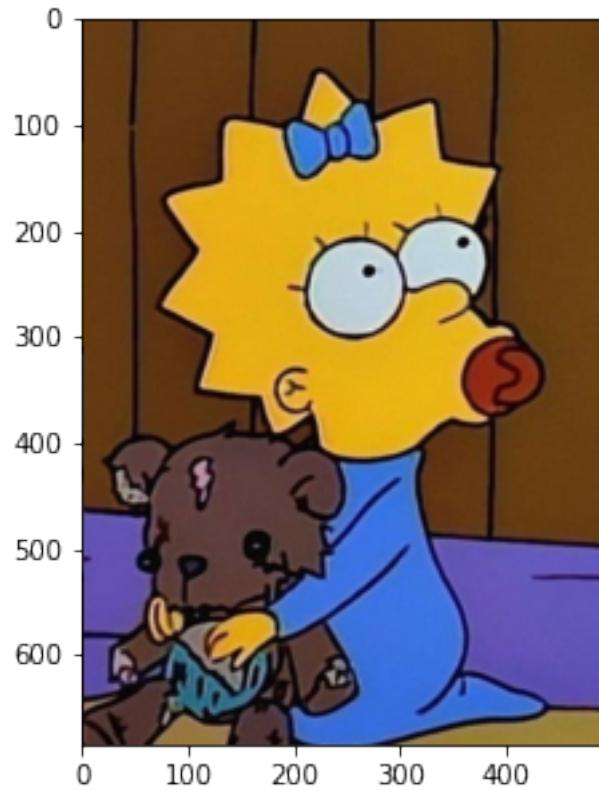
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=223x253 at 0x7FB119B5E5D0>



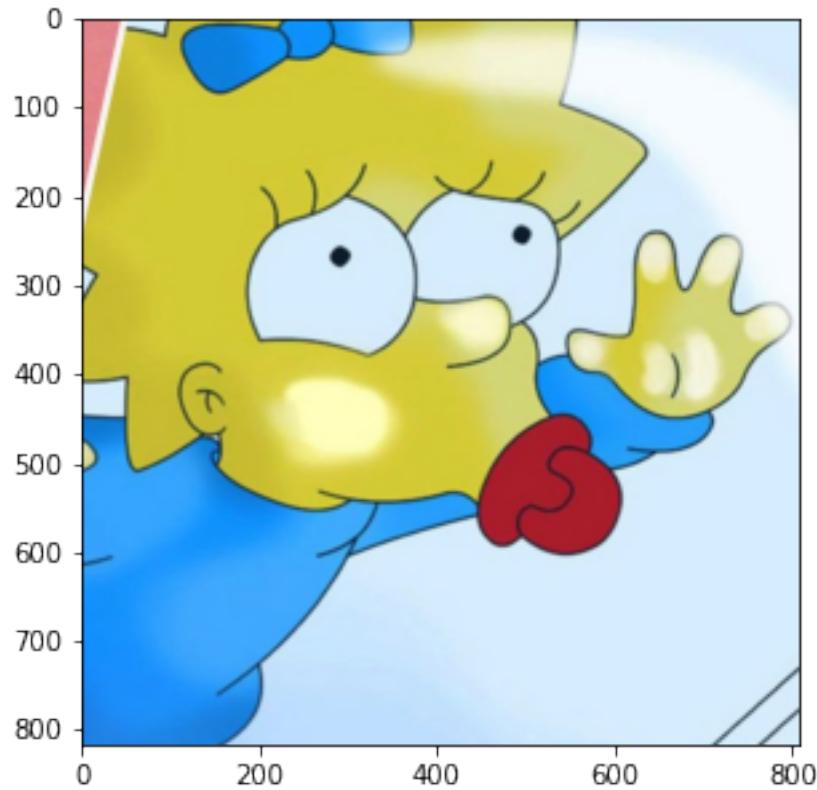
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=100x100 at 0x7FB119B5EED0>



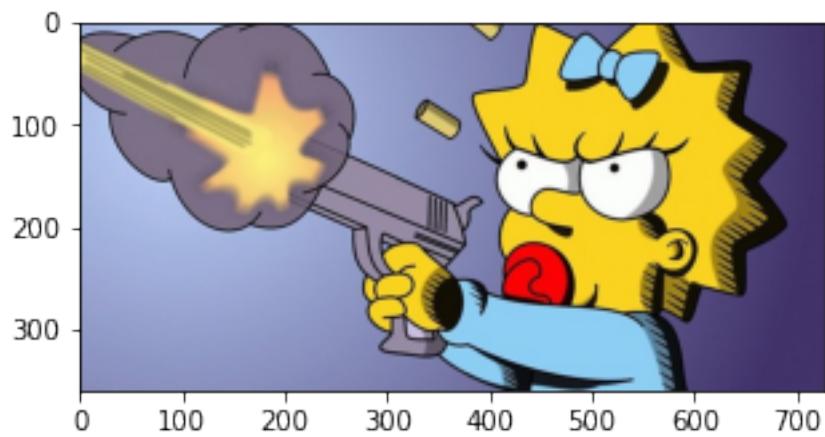
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=491x685 at 0x7FB119B5EF0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=808x817 at 0x7FB119B70110>



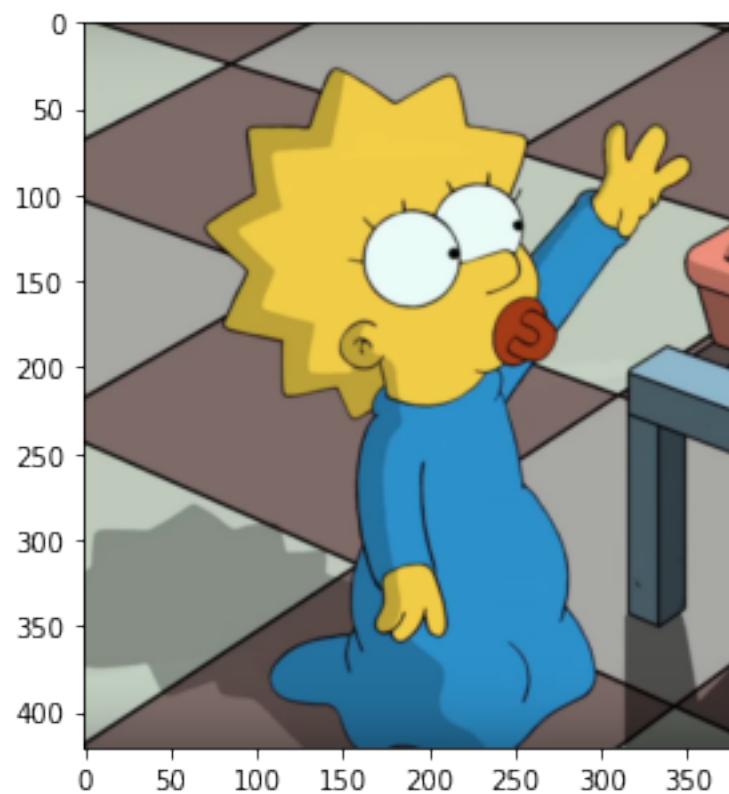
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=728x360 at 0x7FB119B70190>



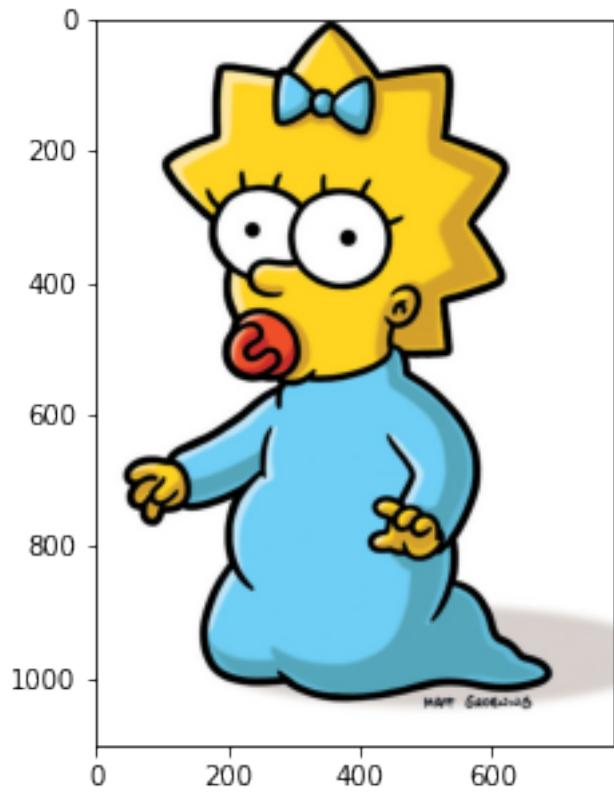
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=640x480 at 0x7FB119B70210>



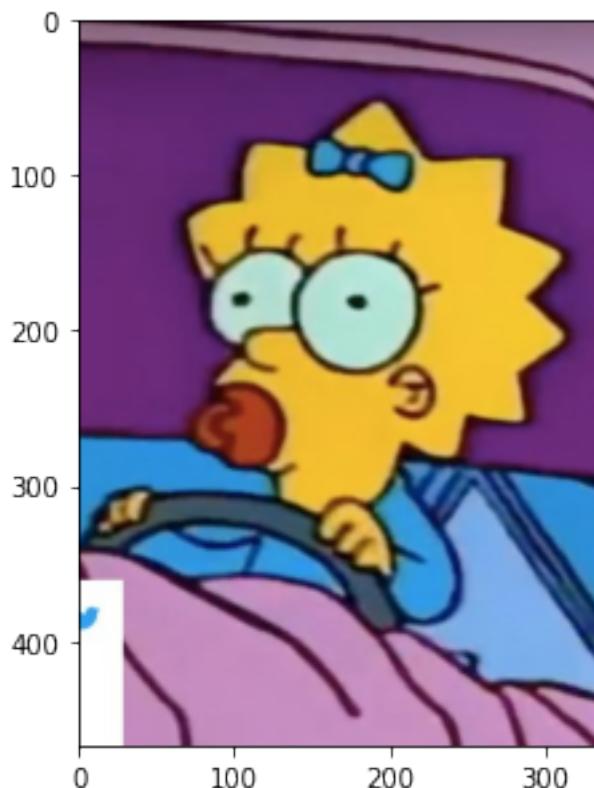
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=378x421 at 0x7FB119B70290>



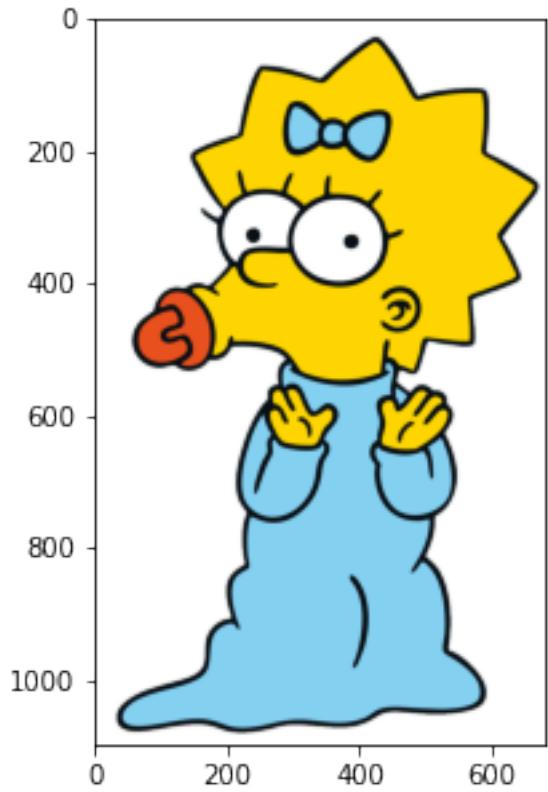
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=785x1103 at 0x7FB119B5E190>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=335x468 at 0x7FB119B704D0>



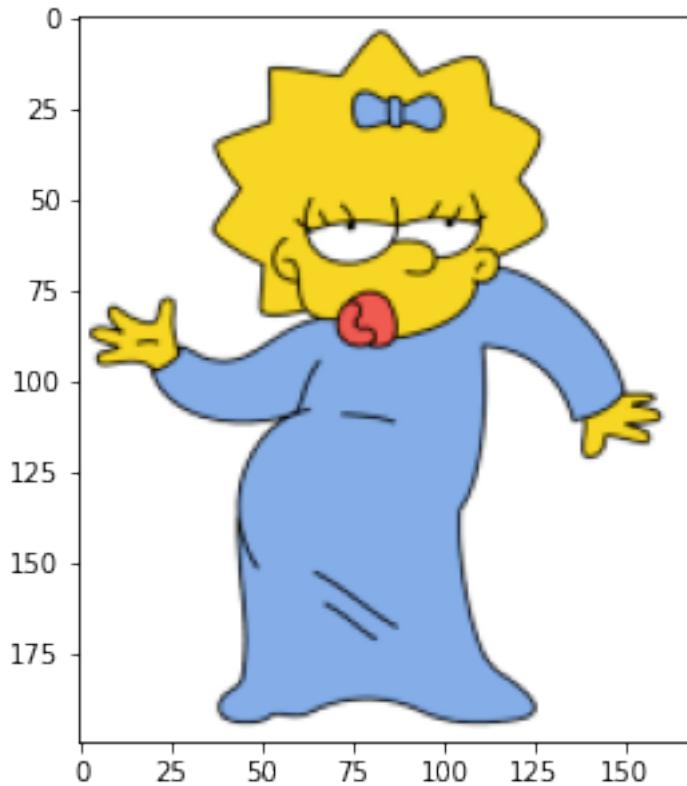
<PIL.PngImagePlugin.PngImageFile image mode=P size=683x1098 at 0x7FB119B70510>



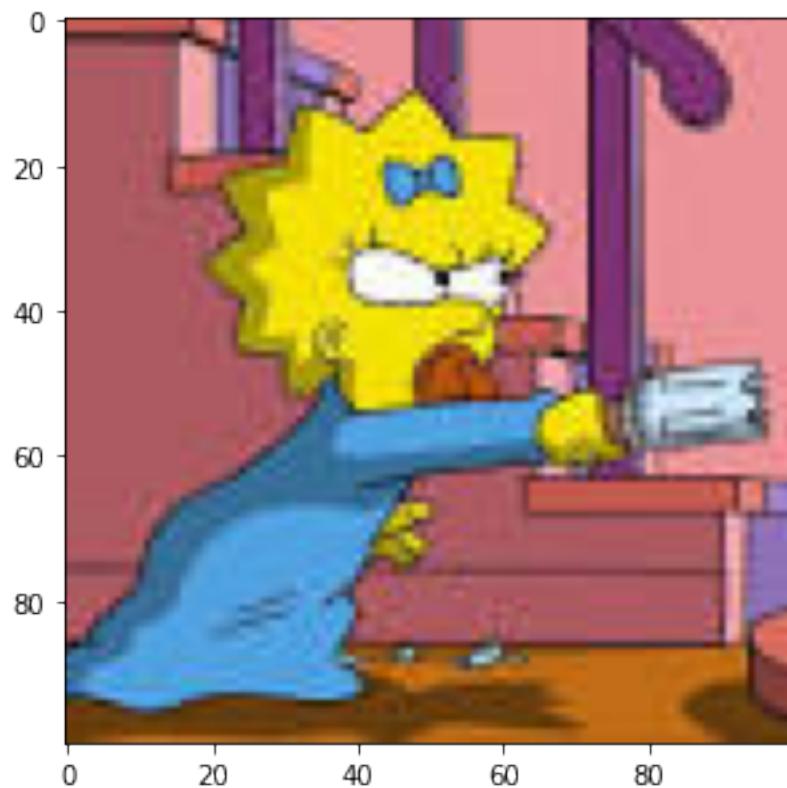
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=446x562 at 0x7FB119B70690>



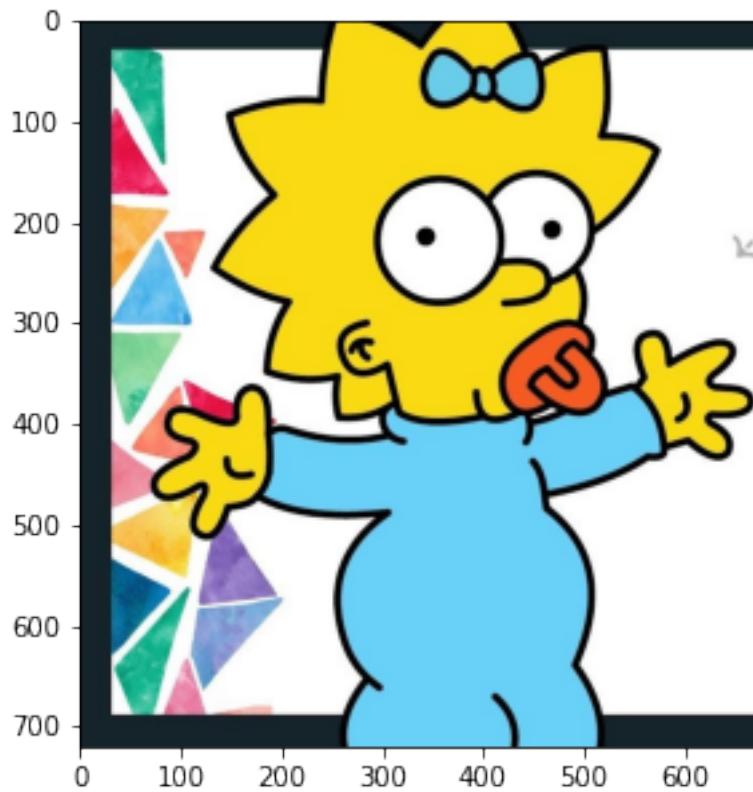
<PIL.PngImagePlugin.PngImageFile image mode=P size=168x200 at 0x7FB119B5EF90>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=100x100 at 0x7FB119B707D0>



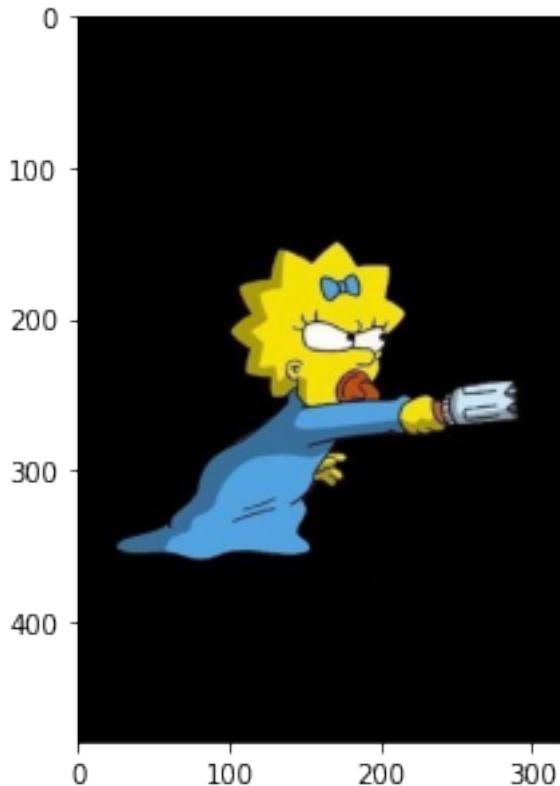
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=669x720 at 0x7FB119B70350>



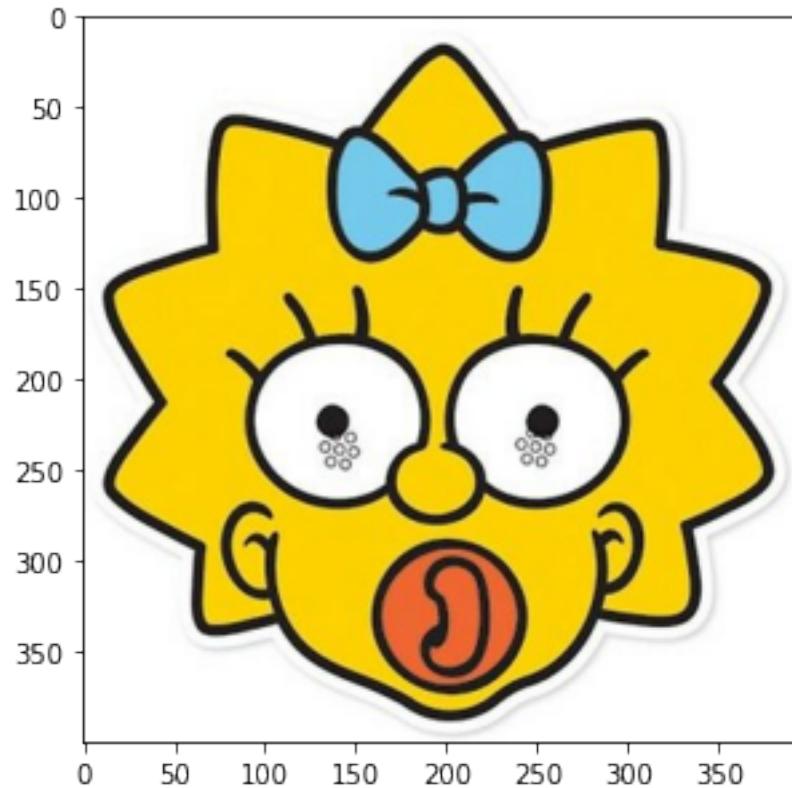
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1200x675 at 0x7FB119B70810>



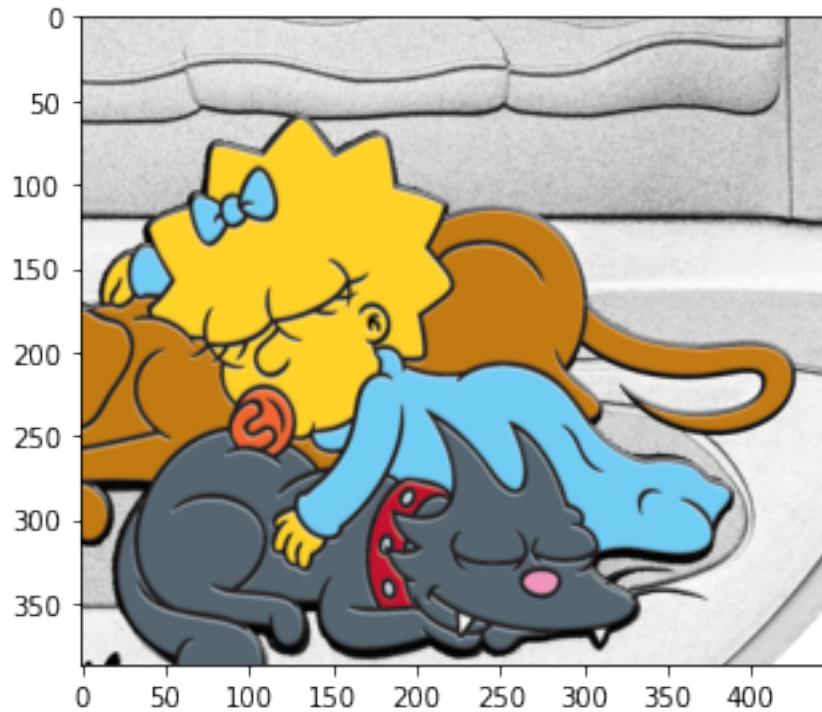
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=320x480 at 0x7FB119B70890>



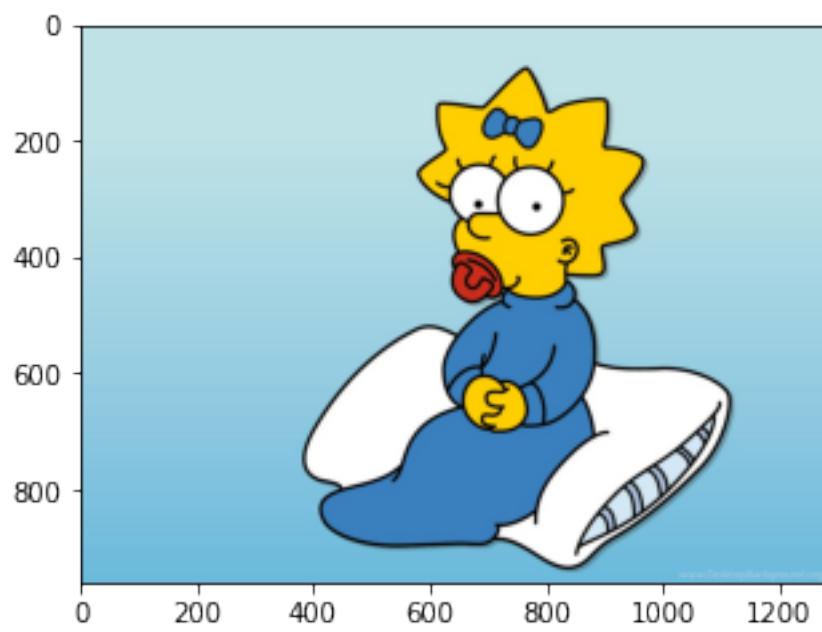
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=393x400 at 0x7FB119B70910>



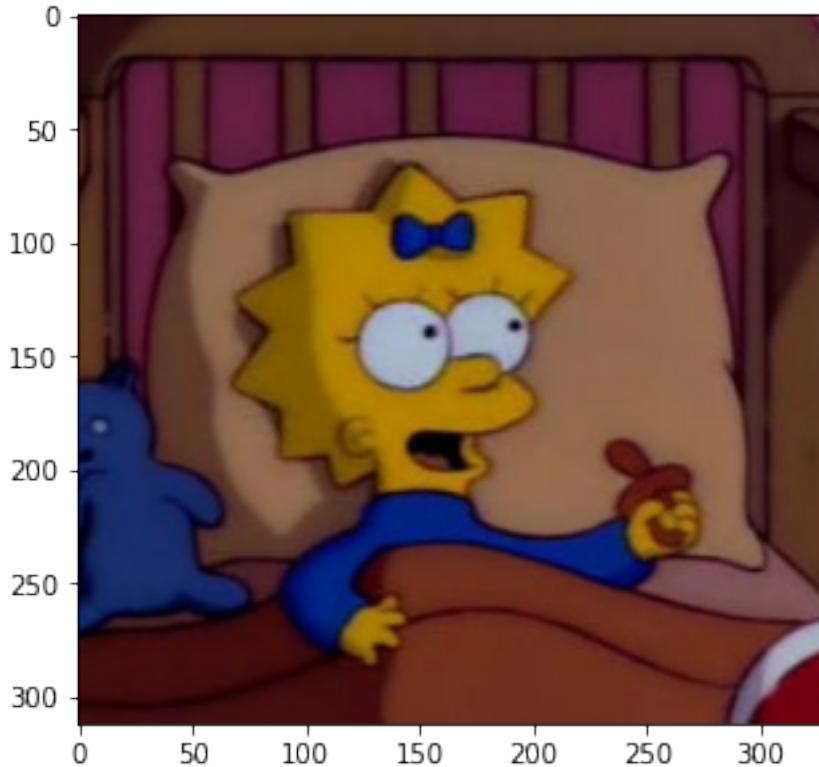
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=445x387 at 0x7FB119B70990>



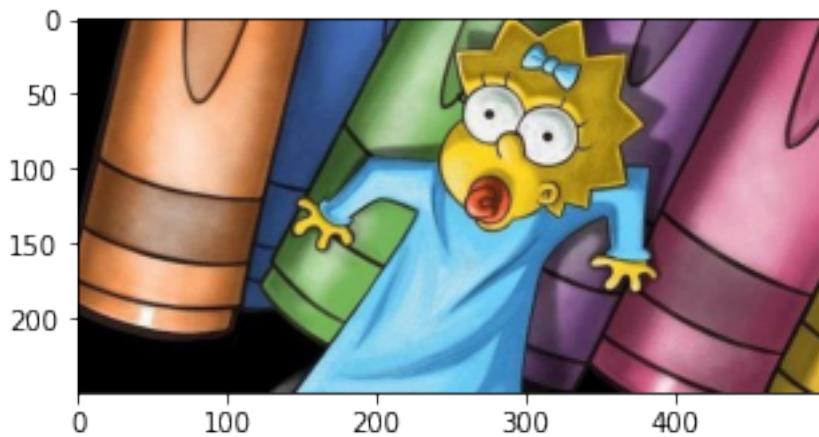
<PIL.PngImagePlugin.PngImageFile image mode=P size=1280x960 at 0x7FB119B70A10>



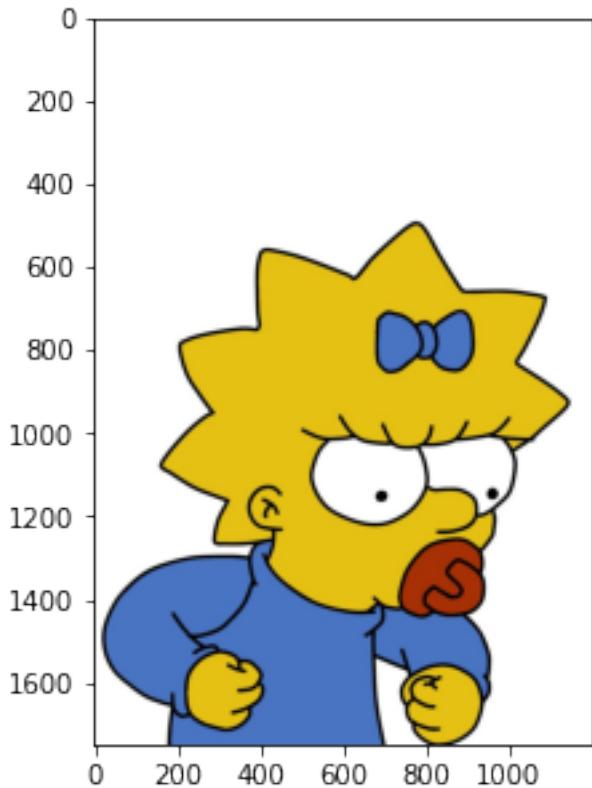
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=328x312 at 0x7FB119B70BD0>



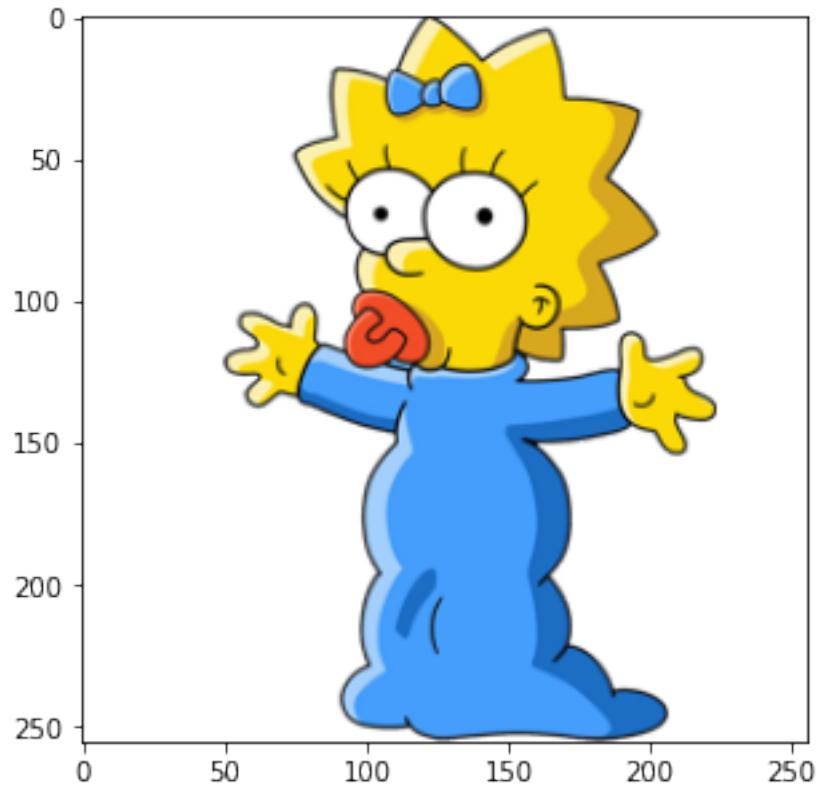
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=500x250 at 0x7FB119B70A90>



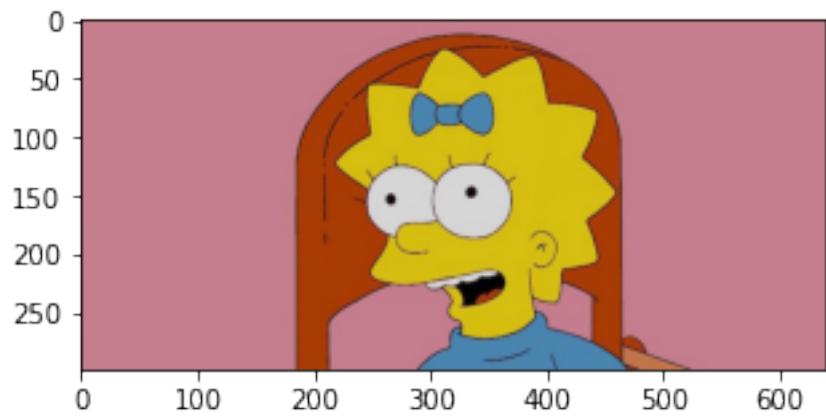
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1200x1750 at 0x7FB119B70C10>



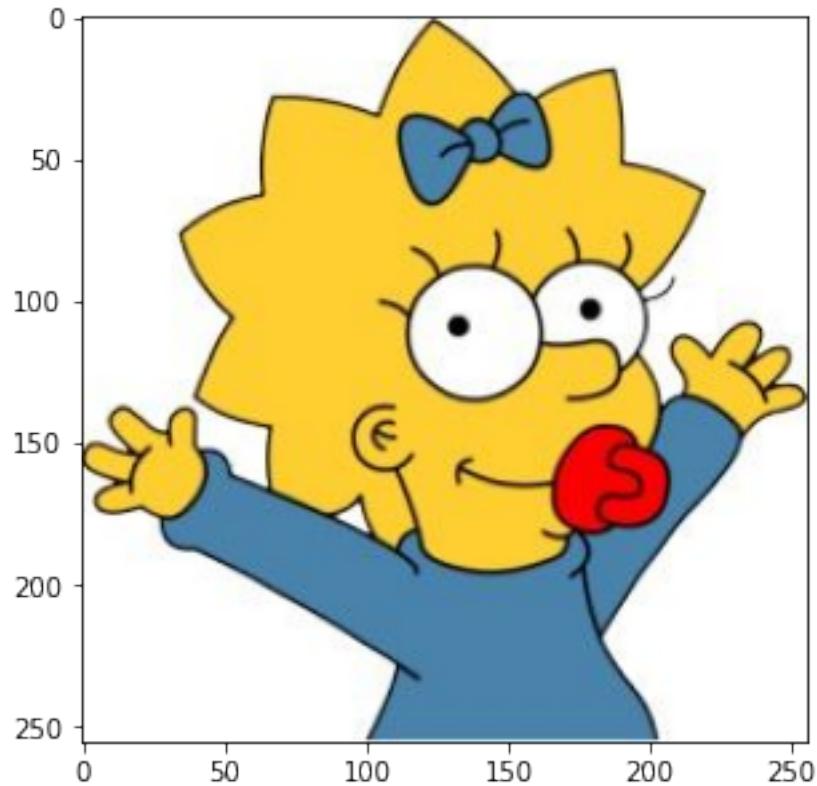
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=256x256 at 0x7FB119B70C90>



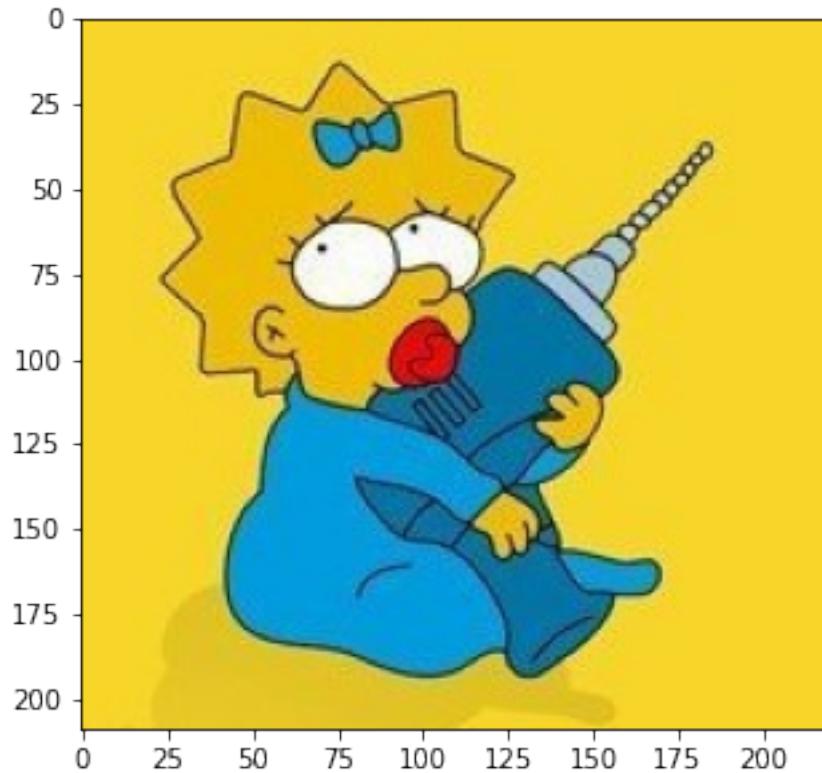
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=640x299 at 0x7FB119B70D10>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=256x256 at 0x7FB119B70D90>



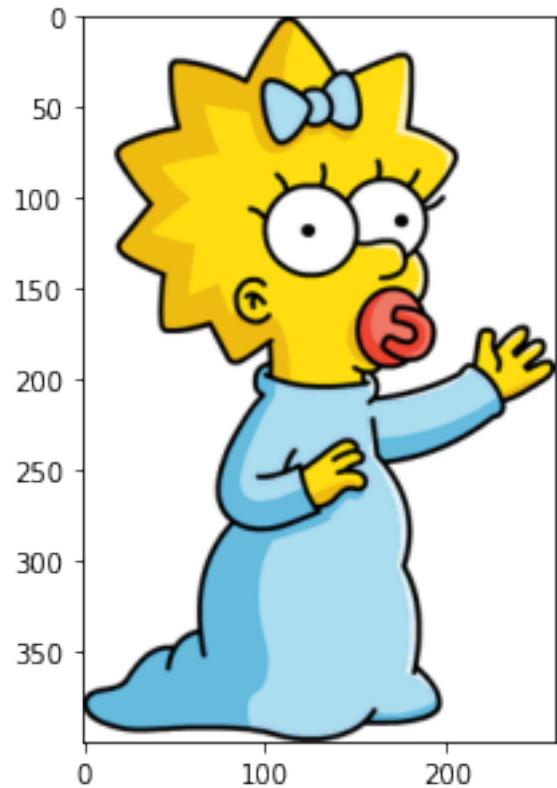
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=219x209 at 0x7FB119B70E10>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=136x198 at 0x7FB119B70E90>

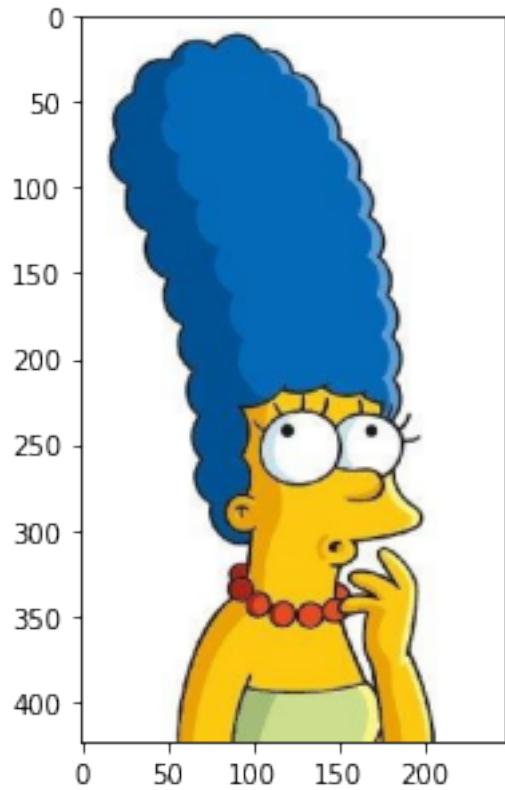


<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=260x400 at 0x7FB119B70F10>

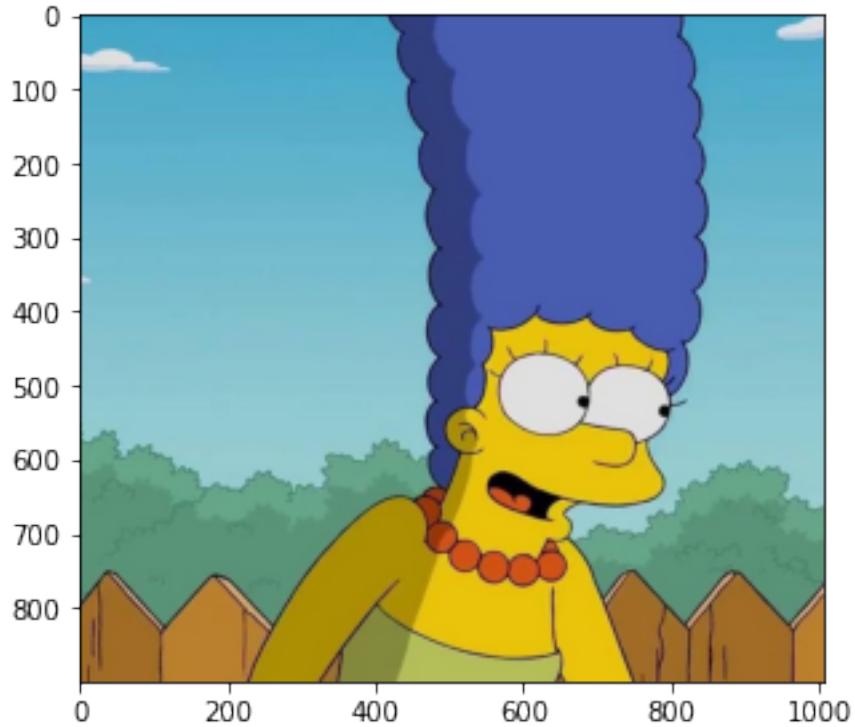


Marge

<PIL.PngImagePlugin.PngImageFile image mode=RGB size=248x424 at 0x7FB119B70F90>



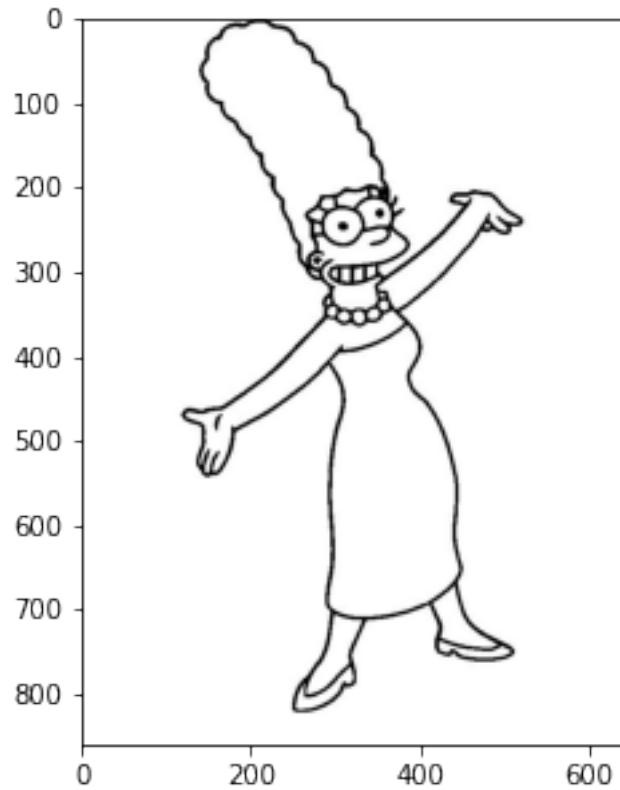
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1007x900 at 0x7FB119B70710>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=275x183 at 0x7FB119B00C10>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=640x860 at 0x7FB119B000D0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=182x268 at 0x7FB119B00150>



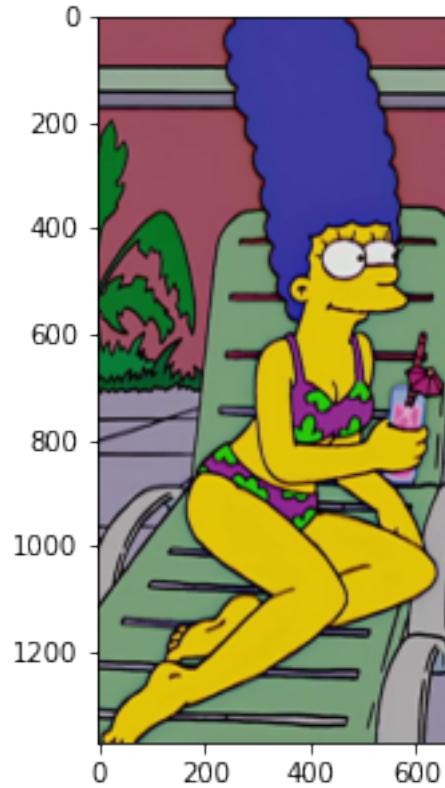
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=512x384 at 0x7FB119B00210>



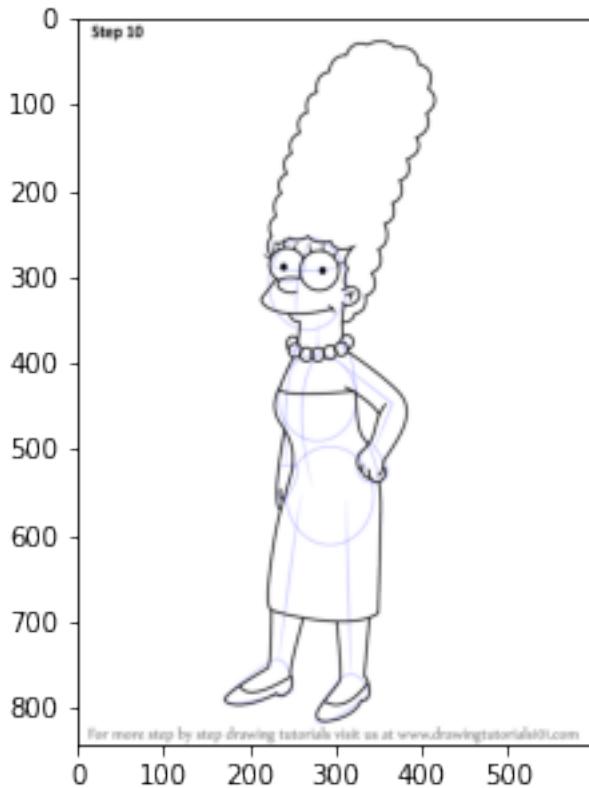
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=430x1463 at  
0x7FB119B00250>
```



```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=656x1371 at  
0x7FB119B001D0>
```



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=598x843 at 0x7FB119B70B50>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=640x362 at 0x7FB119B70090>



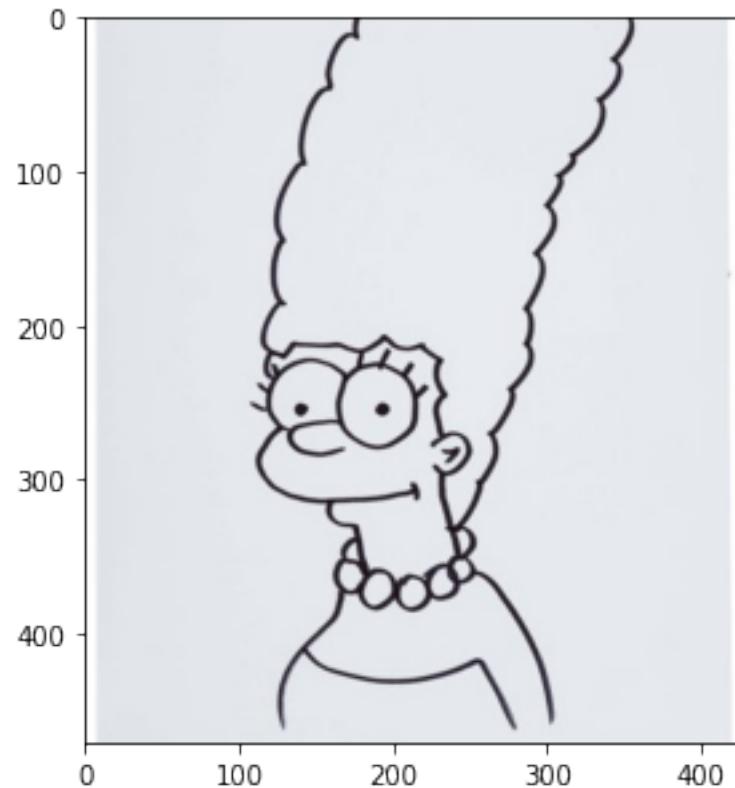
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=640x480 at 0x7FB119B00450>



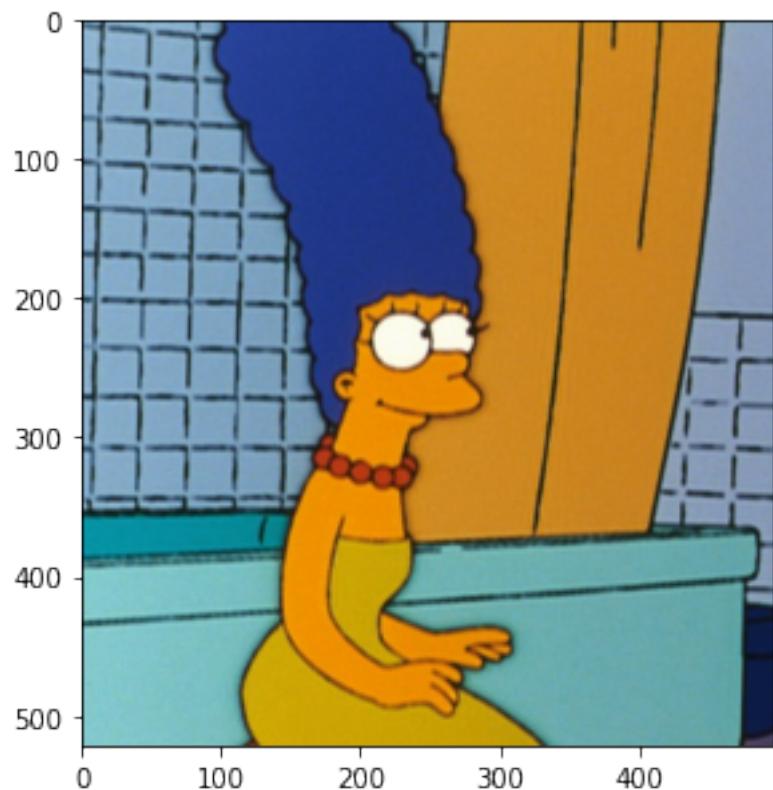
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=312x352 at 0x7FB119B00390>



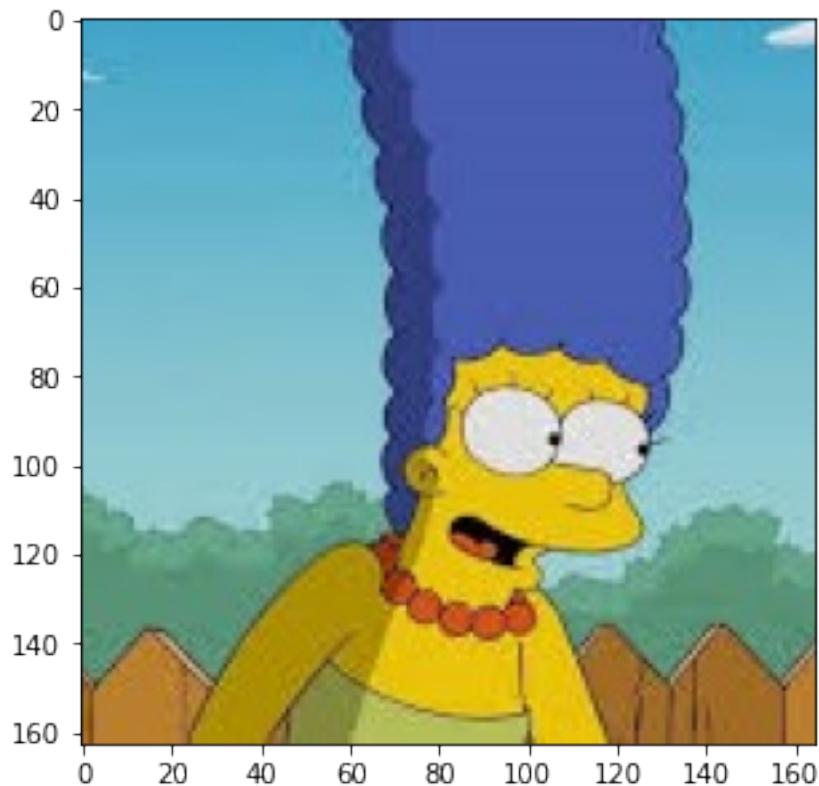
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=425x471 at 0x7FB119B00550>



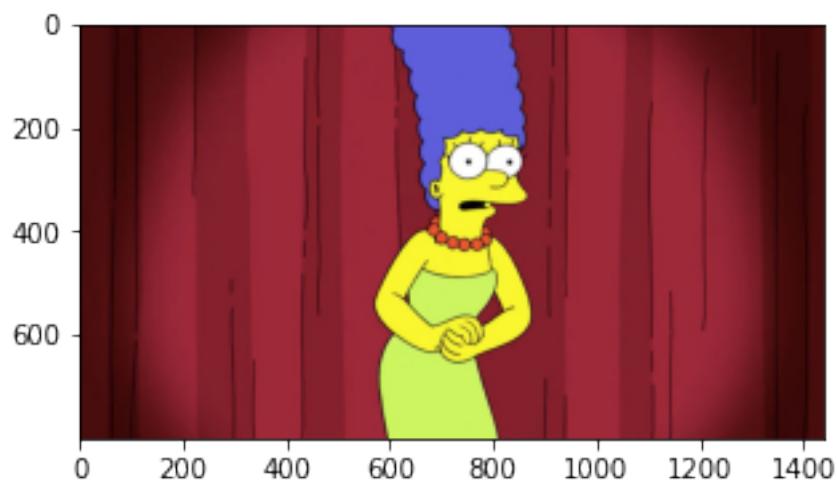
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=498x522 at 0x7FB119B005D0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=165x163 at 0x7FB119B00650>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1440x800 at 0x7FB119B006D0>



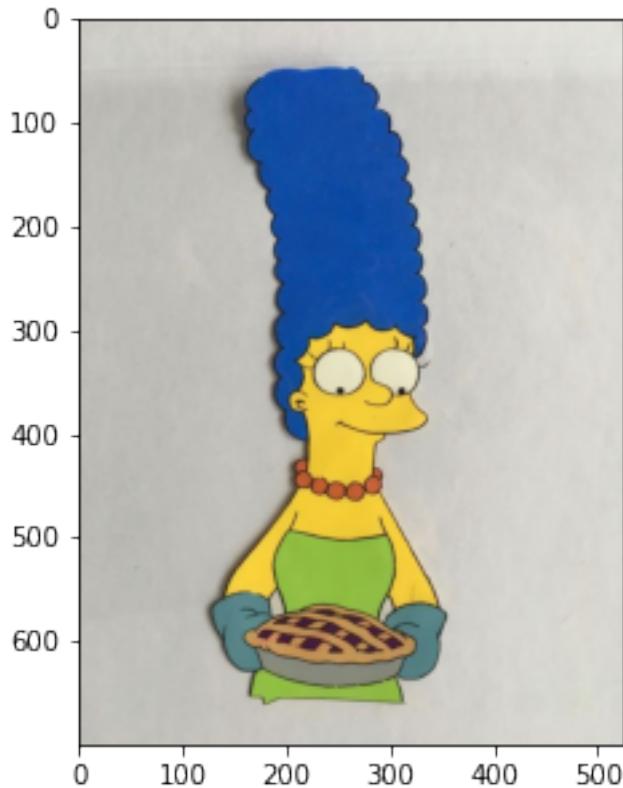
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1024x683 at 0x7FB119B00750>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=389x600 at 0x7FB119B007D0>



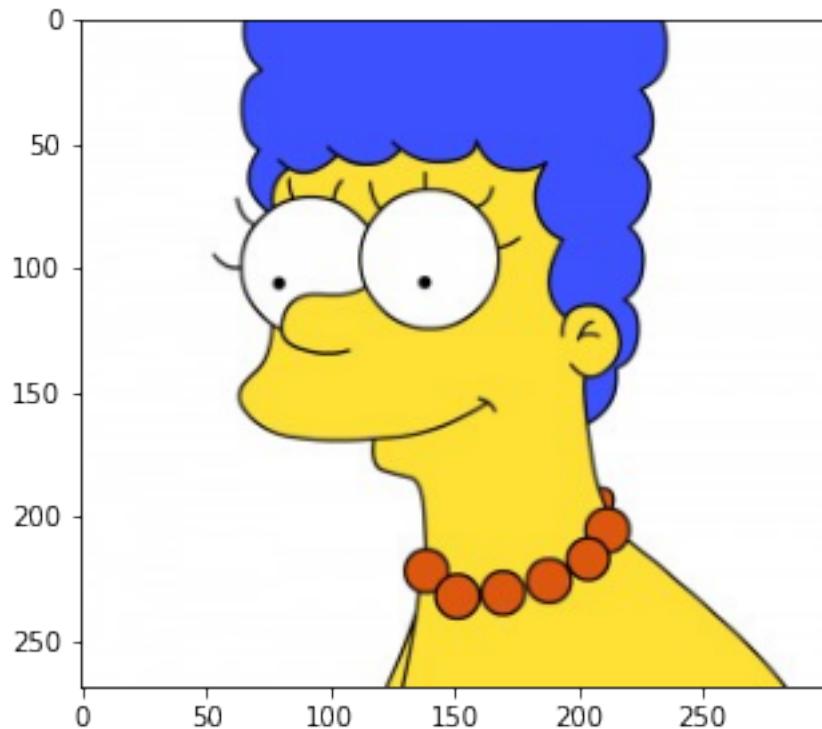
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=525x700 at 0x7FB119B00850>



```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=1043x734 at  
0x7FB119B008D0>
```



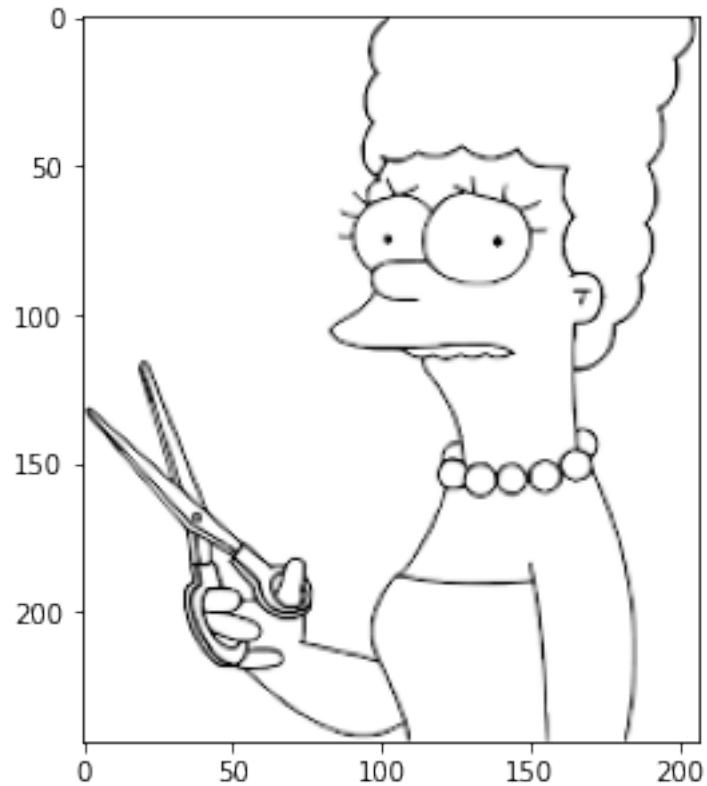
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=300x269 at 0x7FB119B4D9D0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=960x540 at 0x7FB119B3DC90>



<PIL.PngImagePlugin.PngImageFile image mode=P size=207x244 at 0x7FB119B00A50>



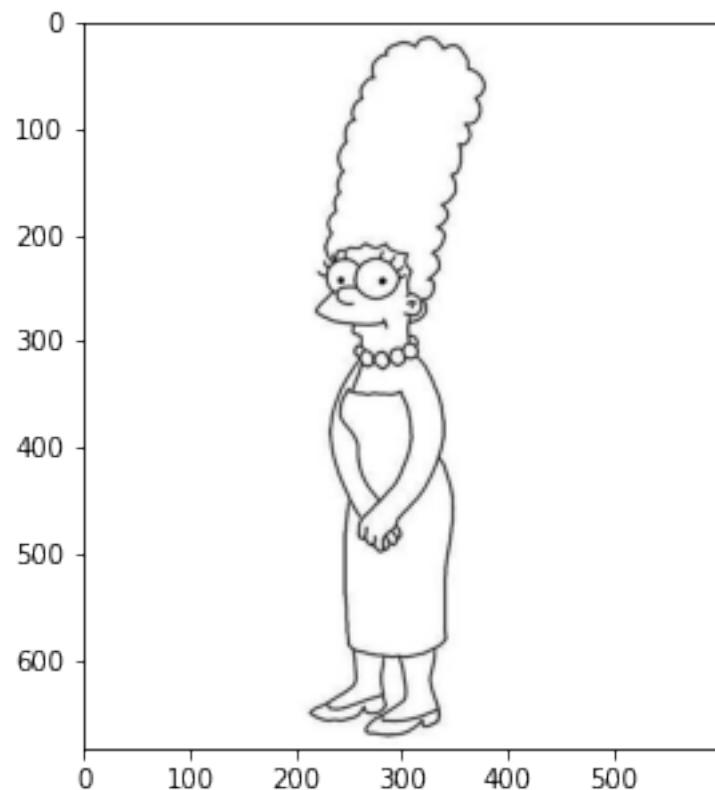
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=320x247 at 0x7FB119B00BD0>



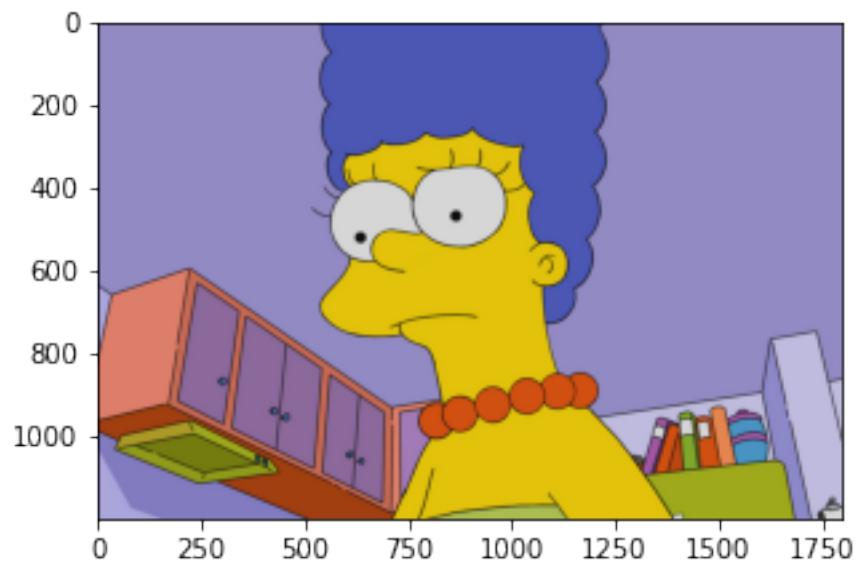
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=702x614 at 0x7FB119B70FD0>



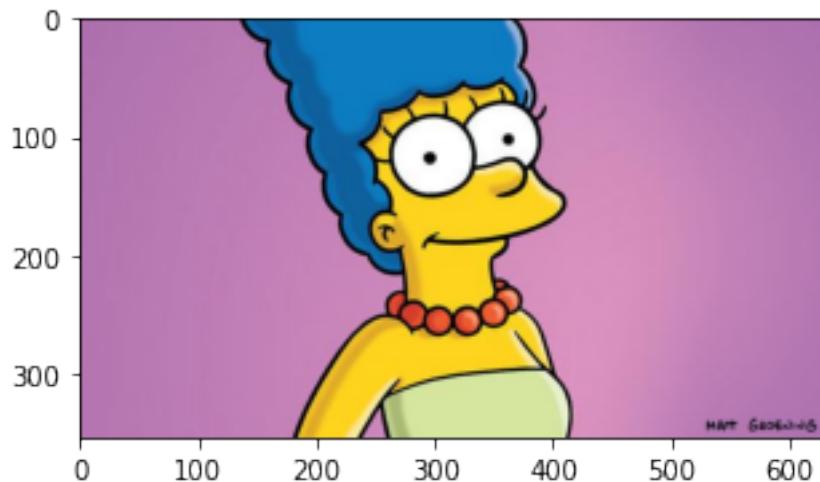
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=600x684 at 0x7FB119B00D10>



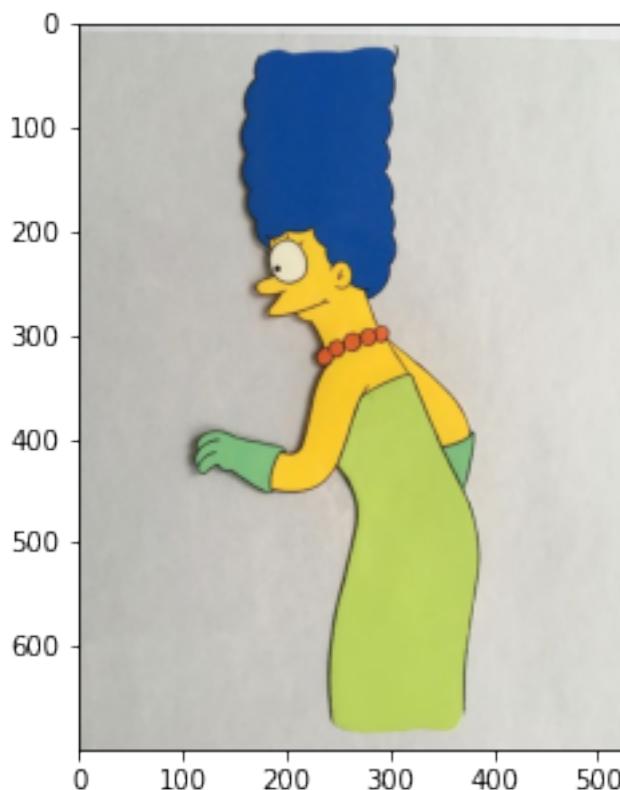
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1801x1200 at  
0x7FB119B00D50>
```



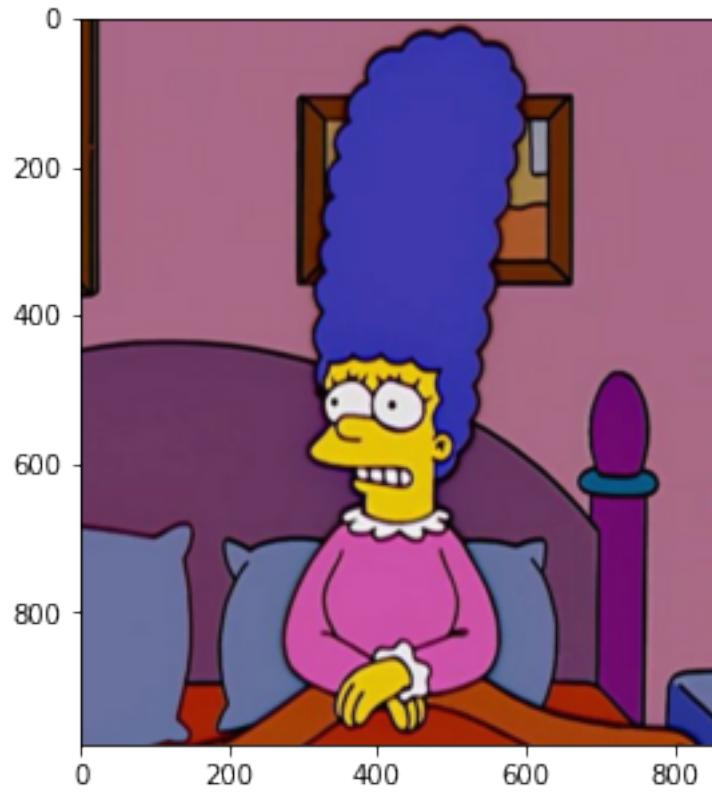
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=630x354 at 0x7FB119B00D90>



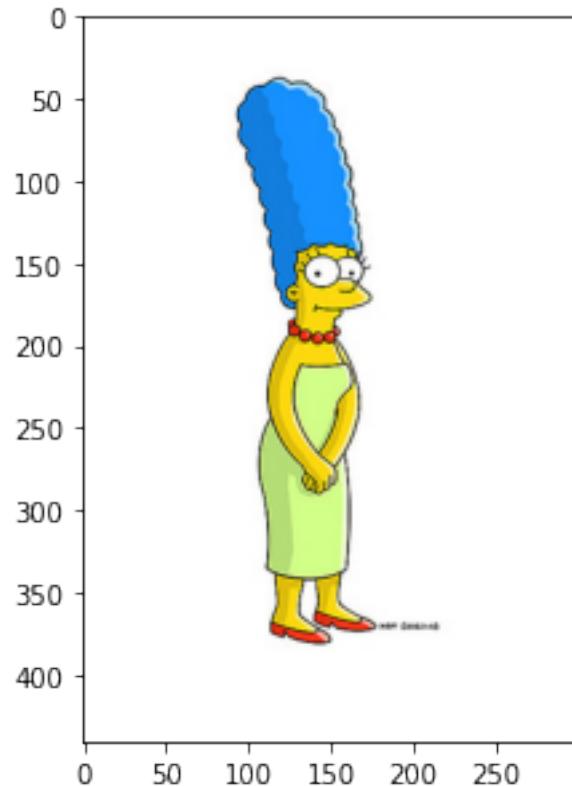
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=525x700 at 0x7FB119B00E50>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=857x978 at 0x7FB119B00ED0>



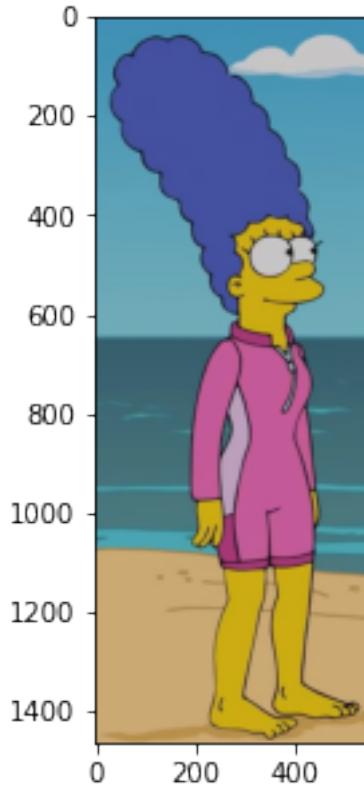
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=300x441 at 0x7FB119B00F50>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=556x715 at 0x7FB119B14090>



<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=546x1463 at 0x7FB119B140D0>

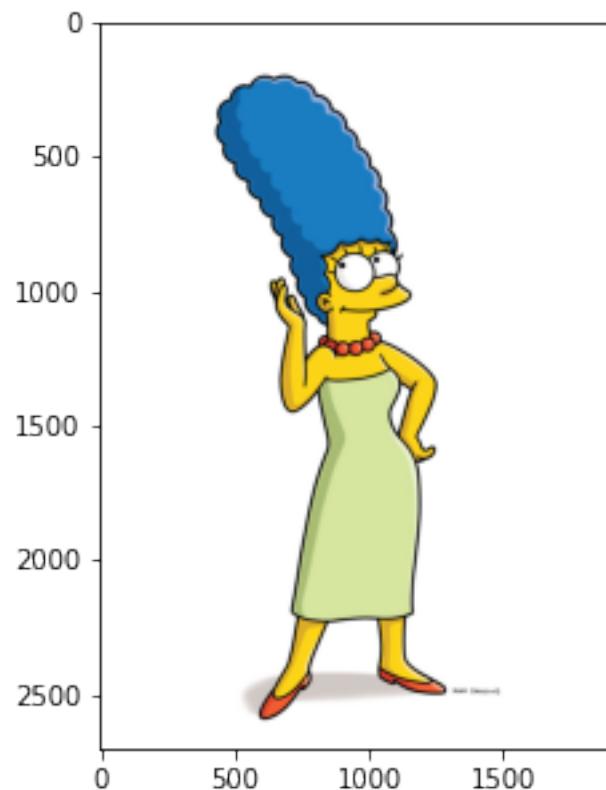


<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=993x655 at 0x7FB119B00B90>

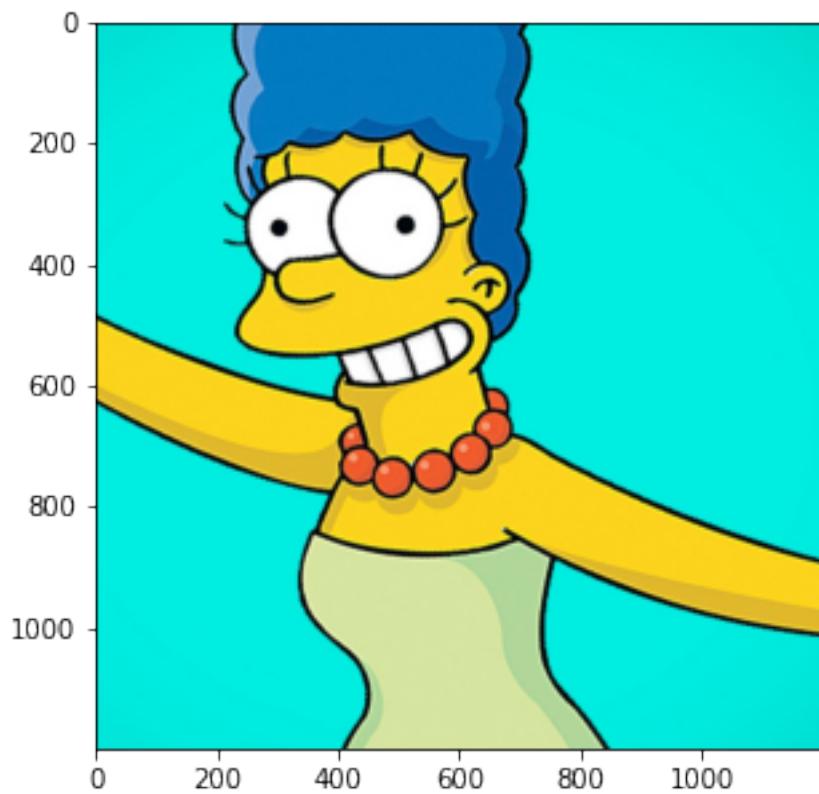


<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1890x2702 at

0x7FB119B00F90>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1200x1200 at
0x7FB119B009D0>



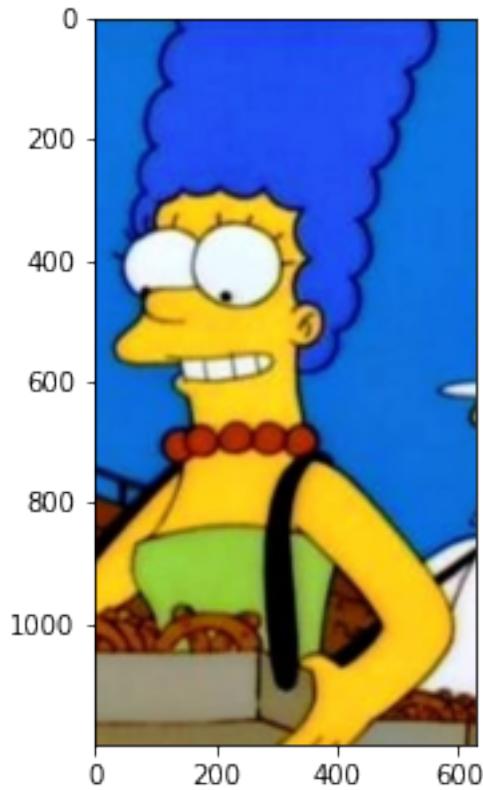
<PIL.PngImagePlugin.PngImageFile image mode=P size=640x480 at 0x7FB119B142D0>



<PIL.PngImagePlugin.PngImageFile image mode=P size=149x337 at 0x7FB119B14450>



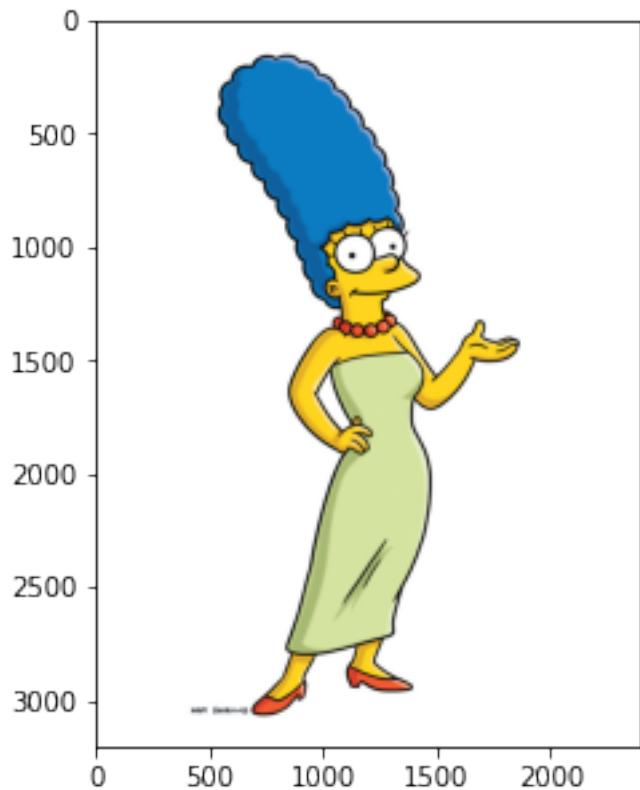
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=630x1200 at 0x7FB119B14510>



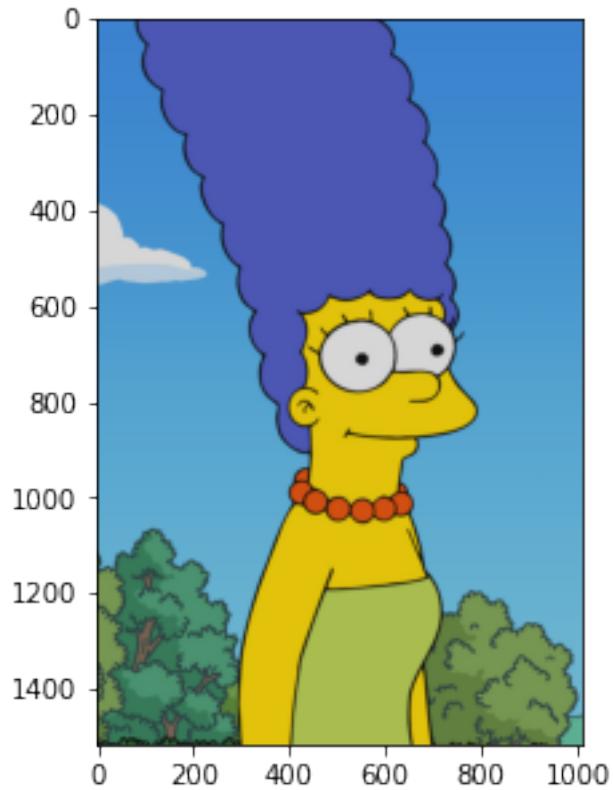
<PIL.PngImagePlugin.PngImageFile image mode=P size=269x187 at 0x7FB119B14190>



```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=2400x3201 at  
0x7FB119B14650>
```



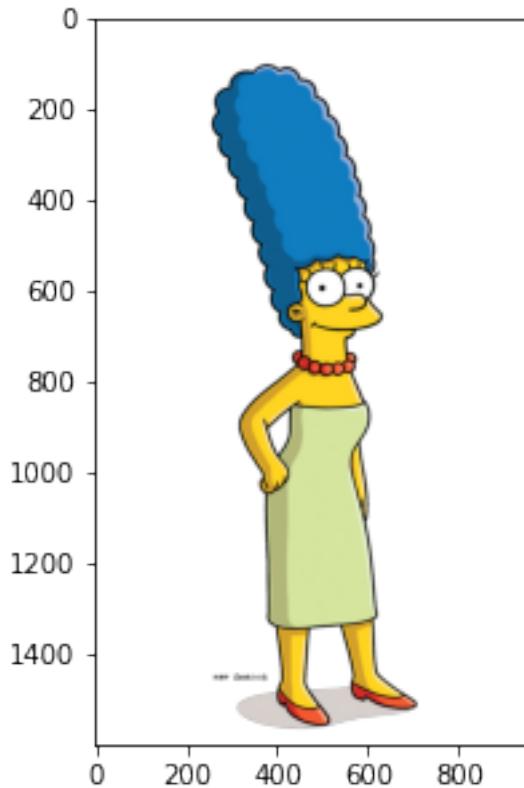
```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1010x1515 at  
0x7FB119B143D0>
```



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=476x522 at 0x7FB119B00C90>



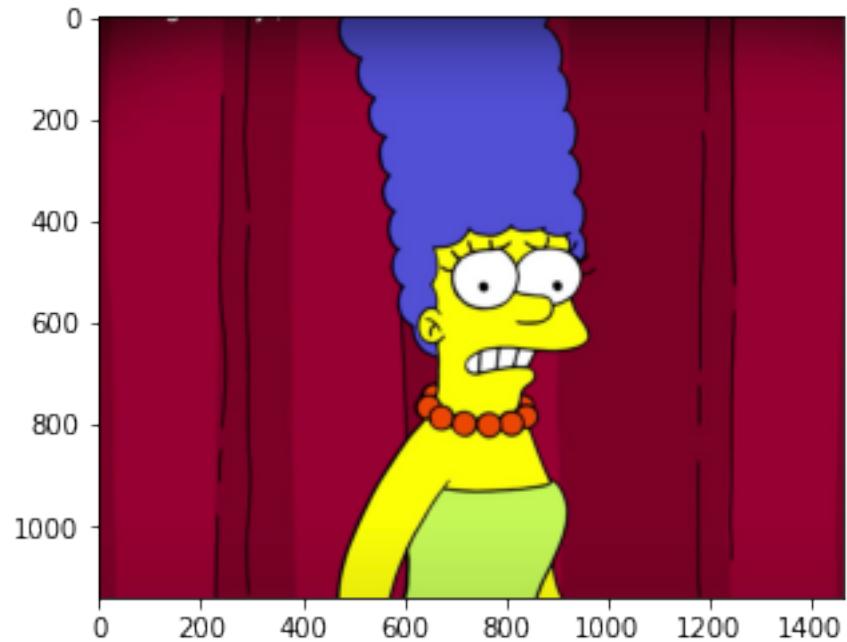
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=950x1600 at  
0x7FB119B14790>
```



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=640x480 at 0x7FB119B70310>



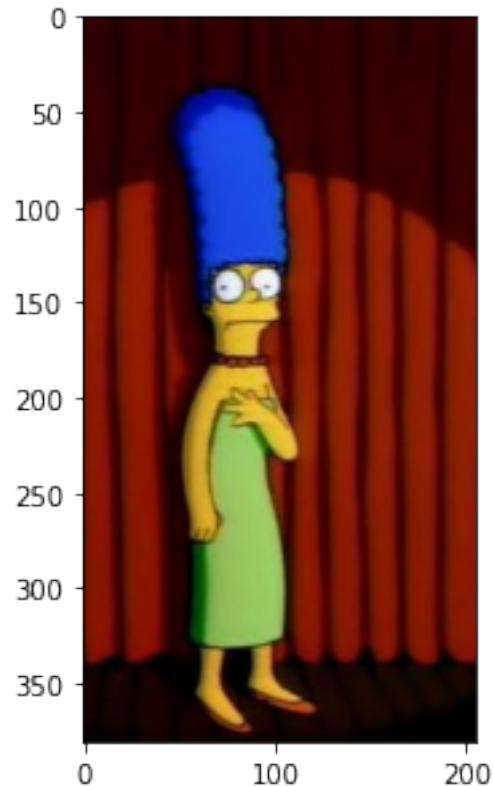
```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1466x1142 at  
0x7FB119B14810>
```



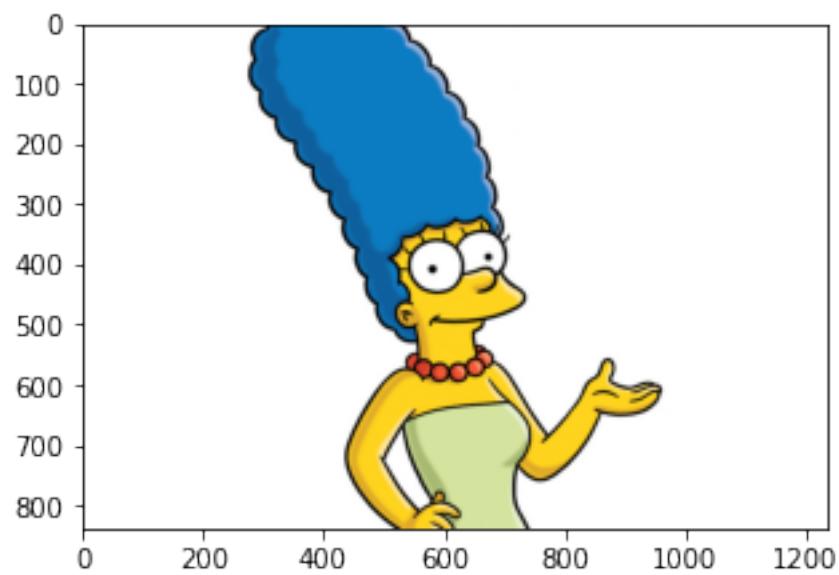
```
<PIL.PngImagePlugin.PngImageFile image mode=RGBA size=748x1687 at  
0x7FB119B14990>
```



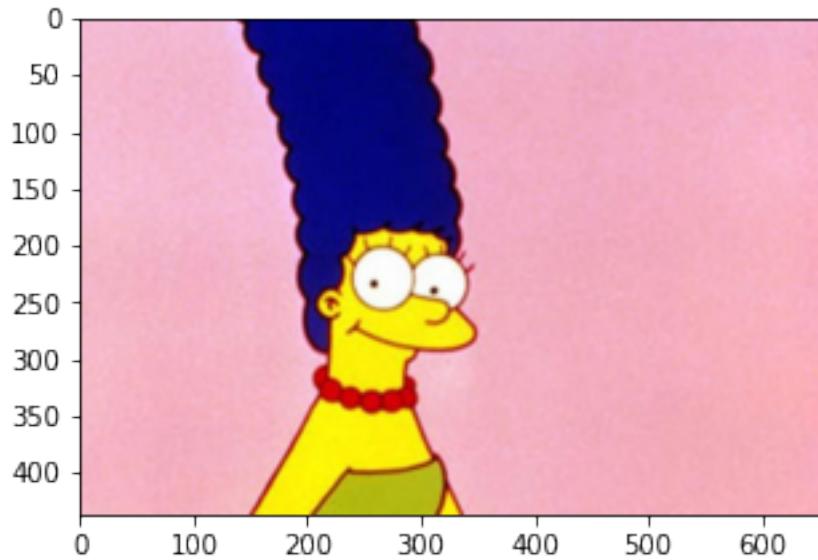
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=206x381 at 0x7FB119B148D0>



<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1237x839 at 0x7FB119B146D0>



```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=656x437 at 0x7FB119B14A90>
```



```
[ ]: w = 256
      h = 256

      rsHomerImages = []

      for item in homerImages:
          img = item.resize((w,h),Image.ANTIALIAS)
          rsHomerImages.append(img)

      rsBartImages = []

      for item in bartImages:
          img = item.resize((w,h),Image.ANTIALIAS)
          rsBartImages.append(img)

      rsLisaImages = []

      for item in lisaImages:
          img = item.resize((w,h),Image.ANTIALIAS)
          rsLisaImages.append(img)

      rsMaggieImages = []

      for item in maggieImages:
          img = item.resize((w,h),Image.ANTIALIAS)
          rsMaggieImages.append(img)
```

```

rsMargeImages = []

for item in margeImages:
    img = item.resize((w,h),Image.ANTIALIAS)
    rsMargeImages.append(img)

[ ]: print("Homer -- Resized")

for item in rsHomerImages:
    print(item)
    plt.imshow(item)
    plt.show()

print("Bart -- Resized")

for item in rsBartImages:
    print(item)
    plt.imshow(item)
    plt.show()

print("Lisa -- Resized")

for item in rsLisaImages:
    print(item)
    plt.imshow(item)
    plt.show()

print("Maggie -- Resized")

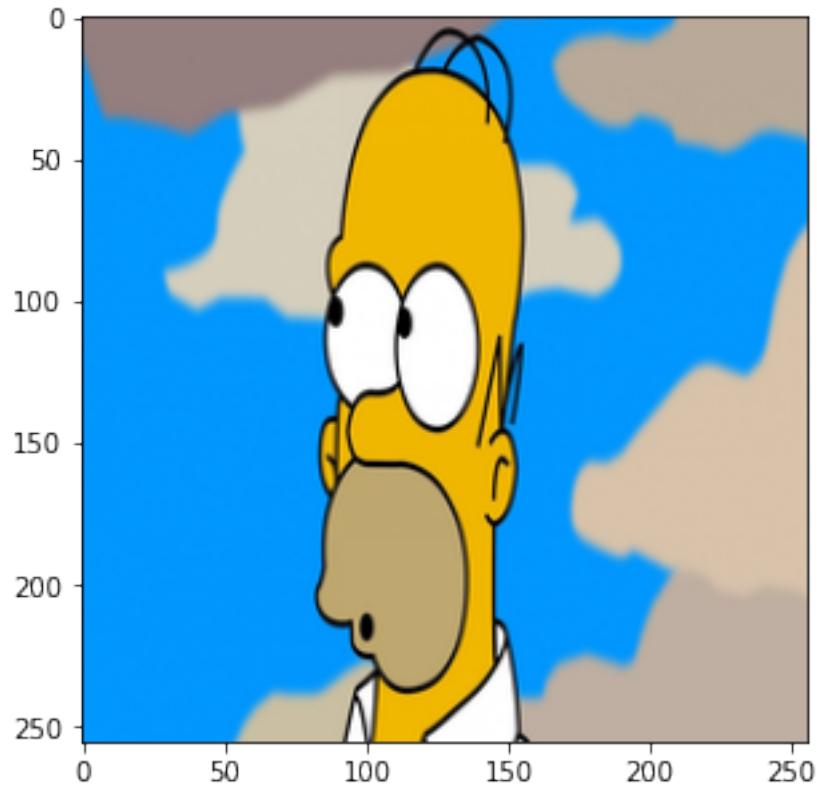
for item in rsMaggieImages:
    print(item)
    plt.imshow(item)
    plt.show()

print("Marge -- Resized")

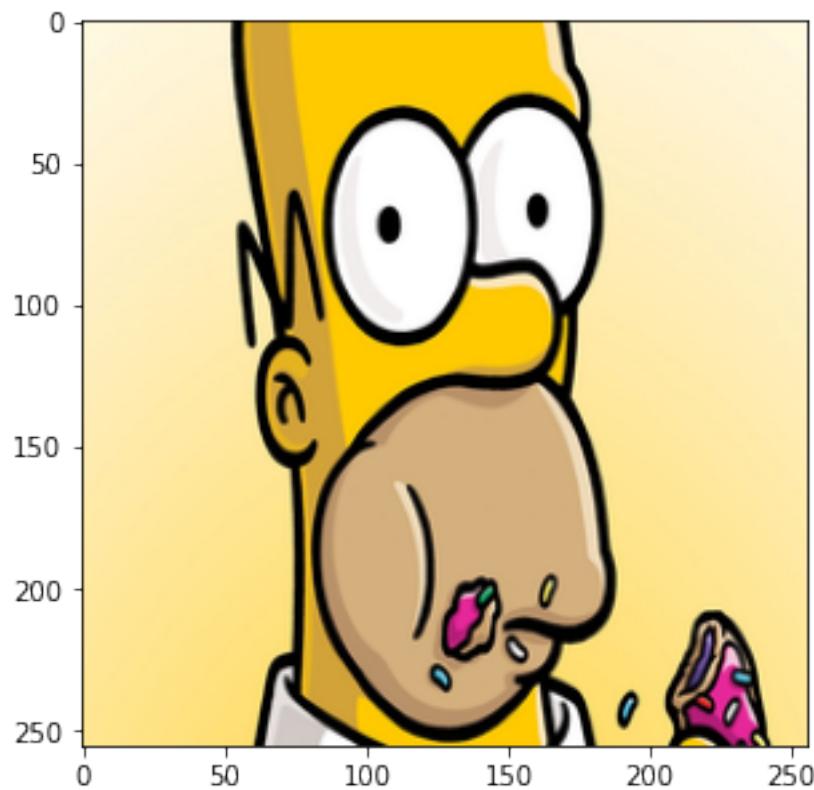
for item in rsMargeImages:
    print(item)
    plt.imshow(item)
    plt.show()

```

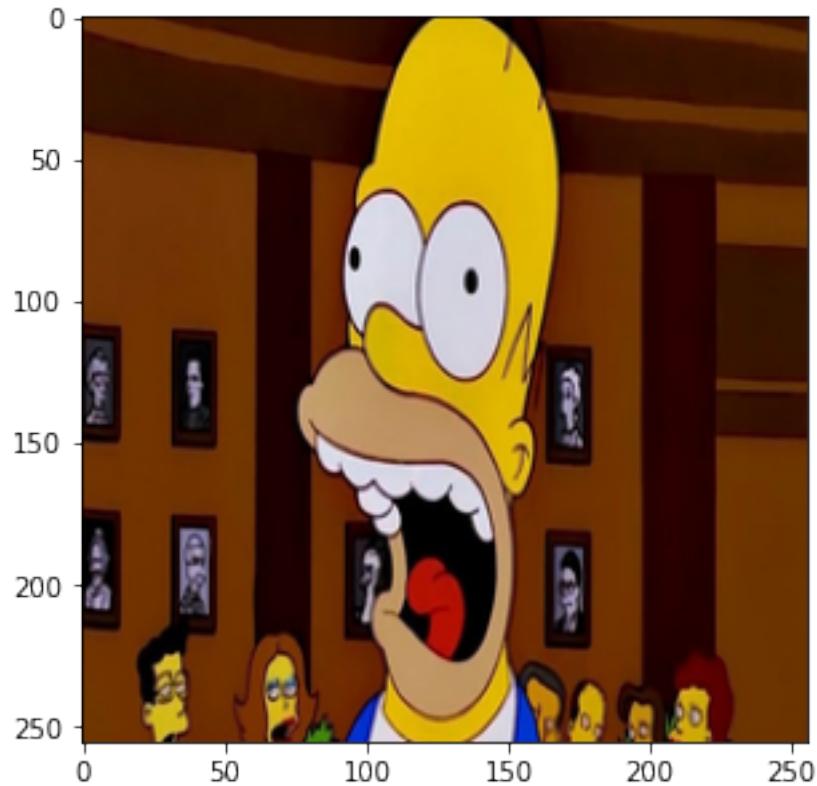
Homer -- Resized
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119460BD0>



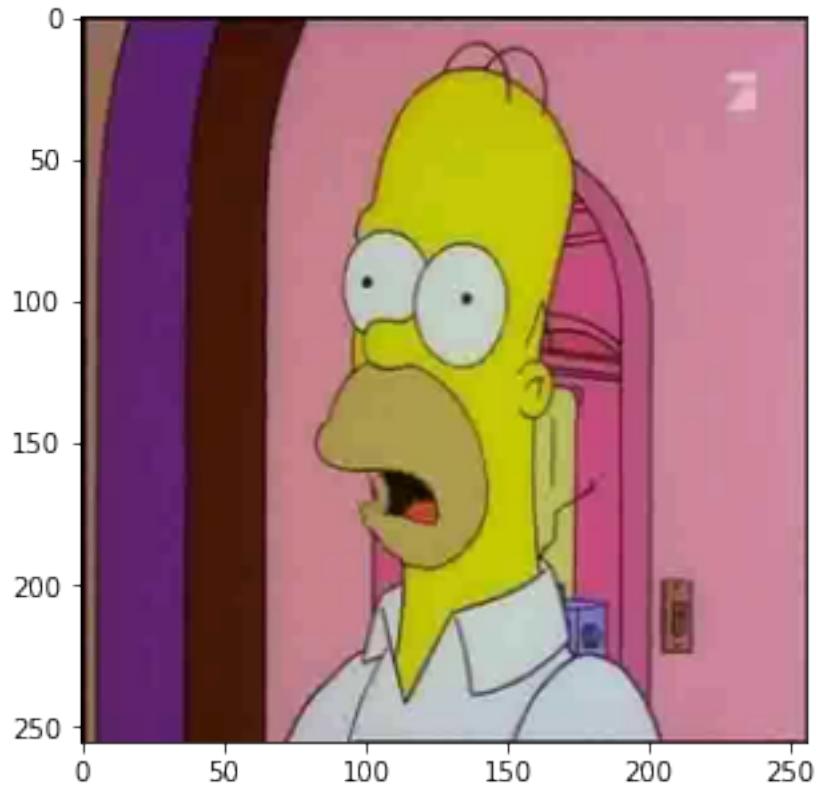
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119460E50>



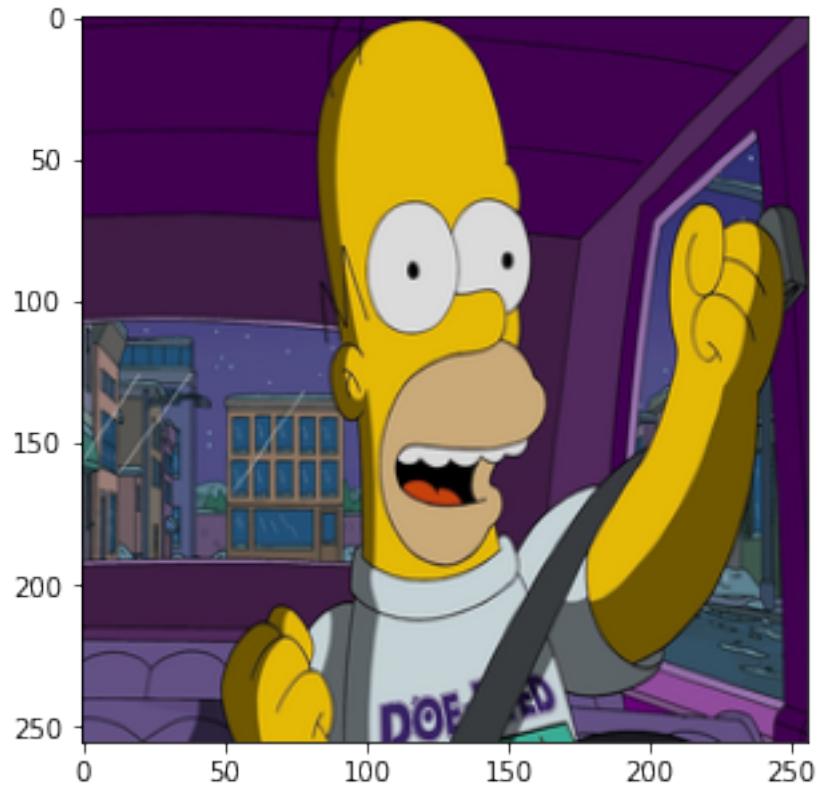
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119460F50>



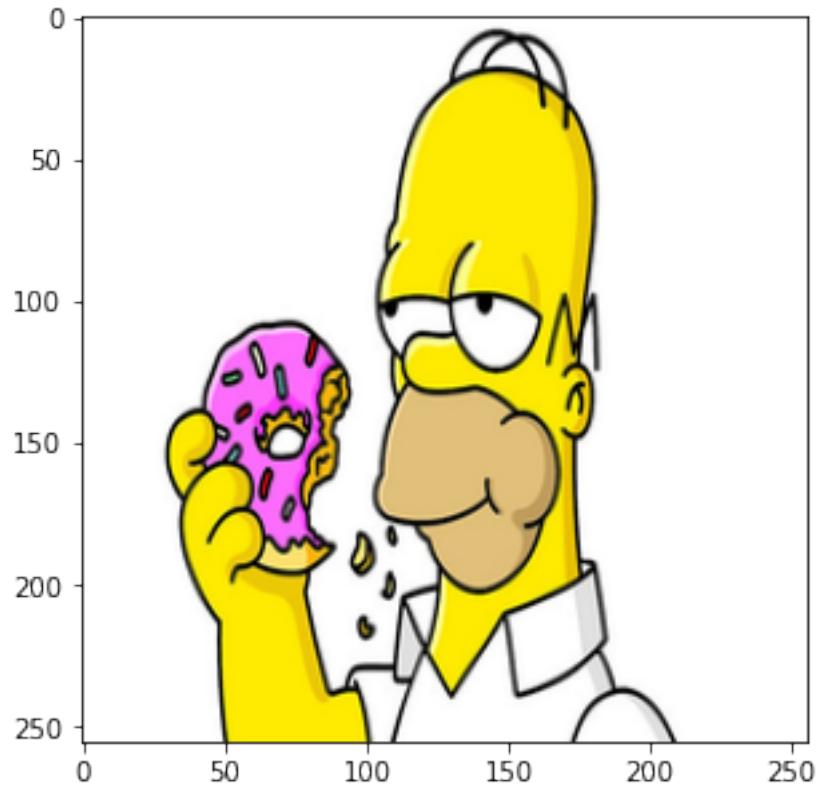
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119460ED0>



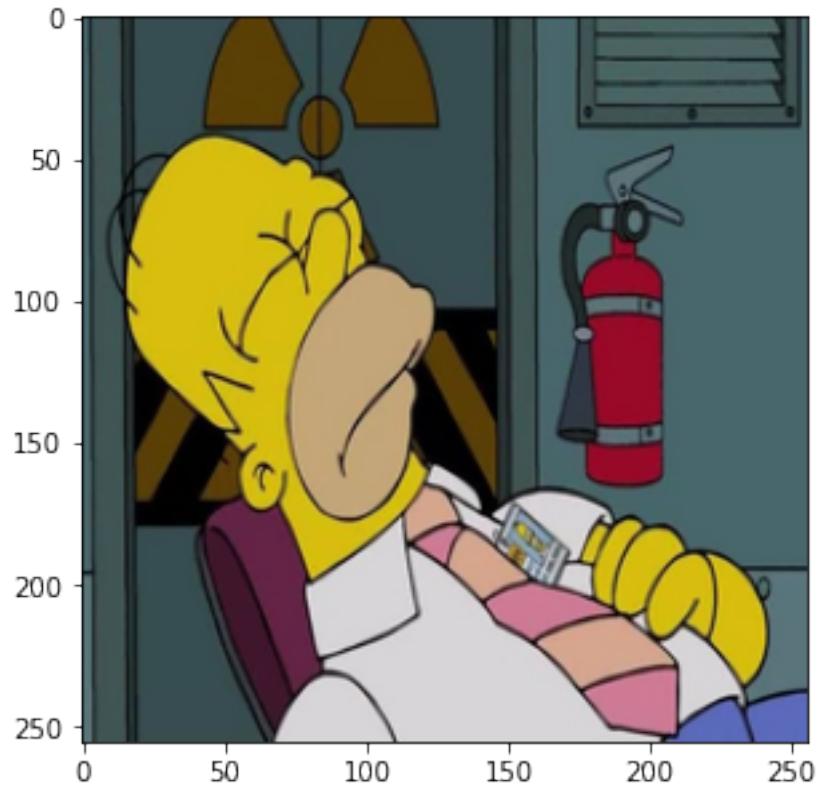
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119460B10>



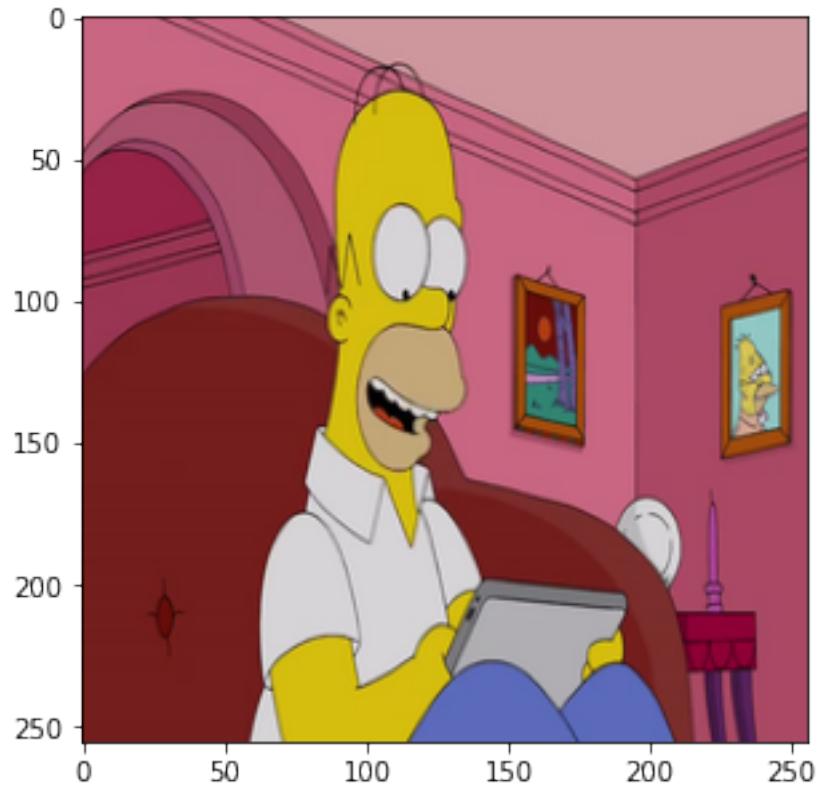
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1194490D0>



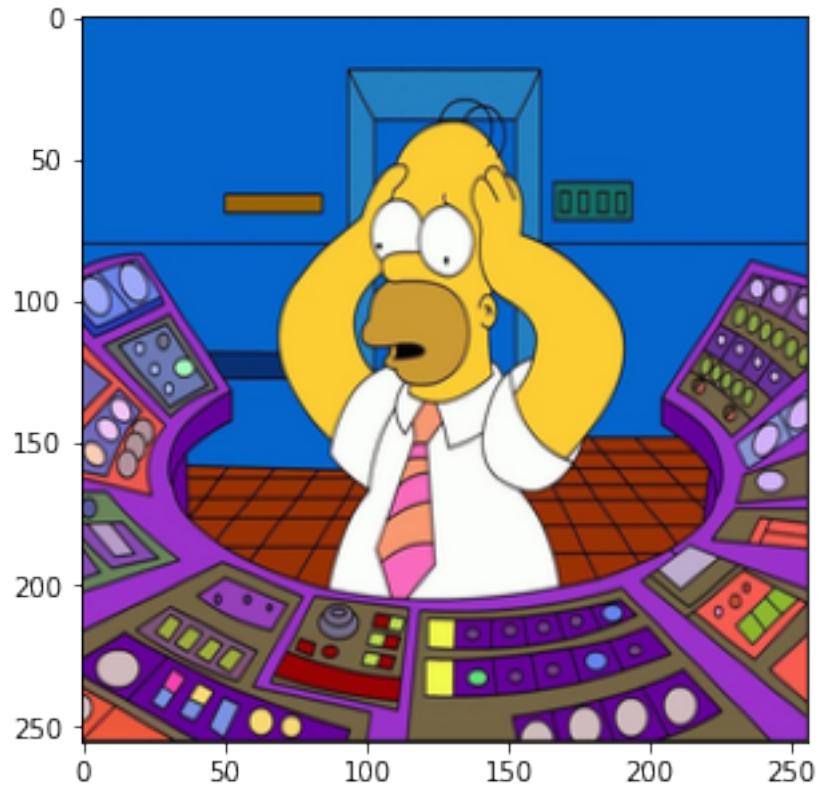
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119449150>



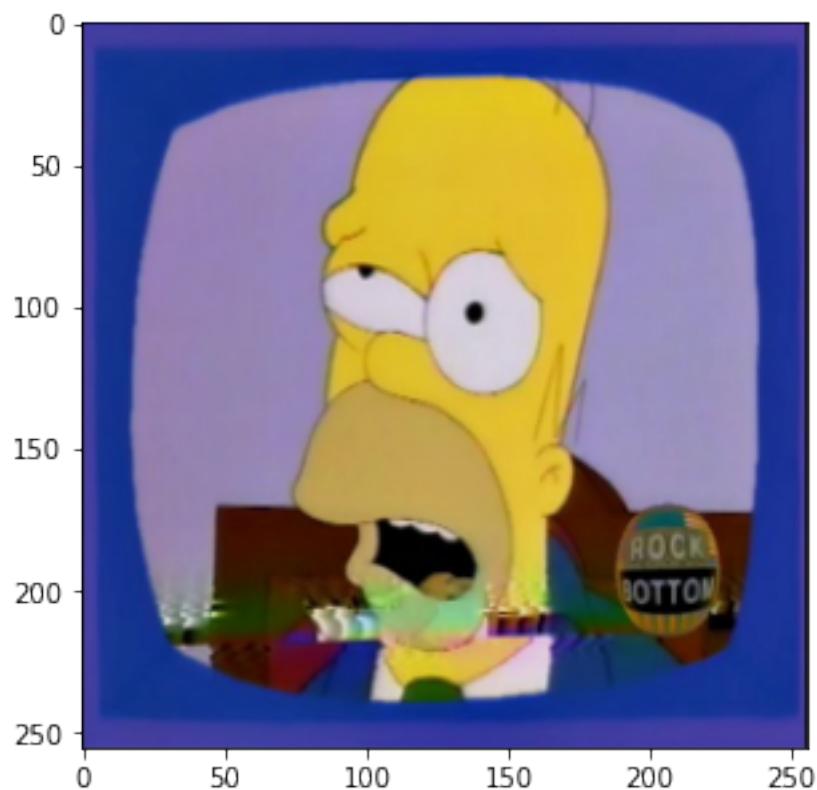
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119449B10>



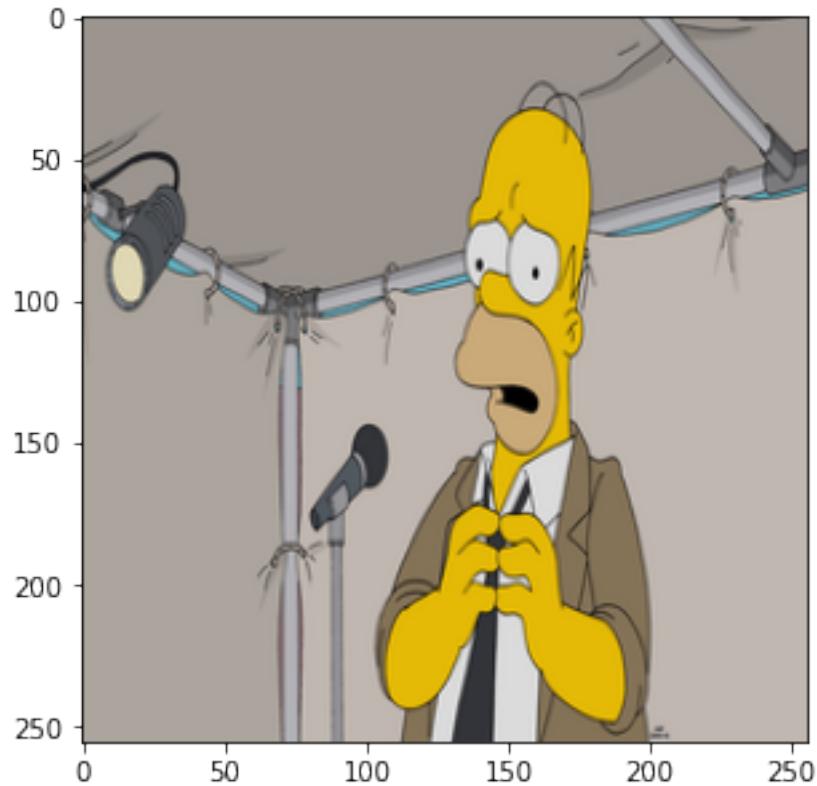
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119449190>



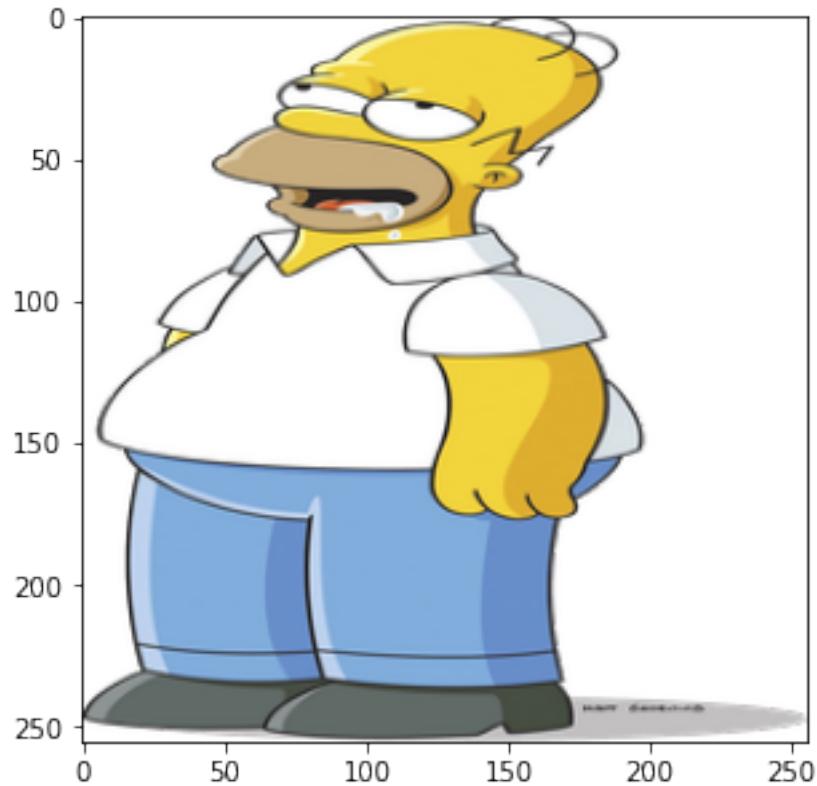
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119460D10>



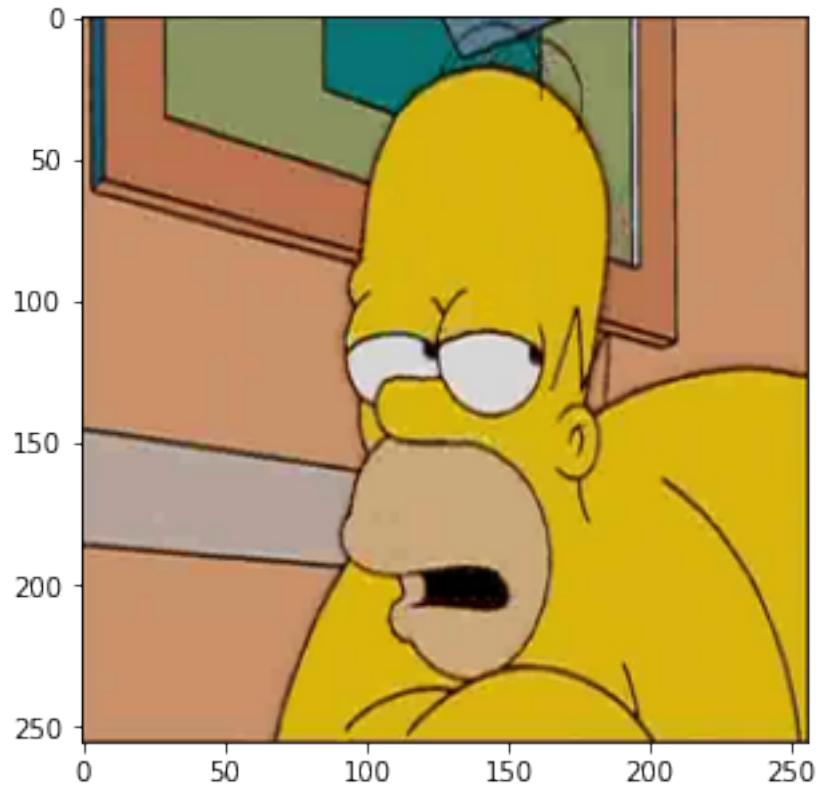
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195BD650>



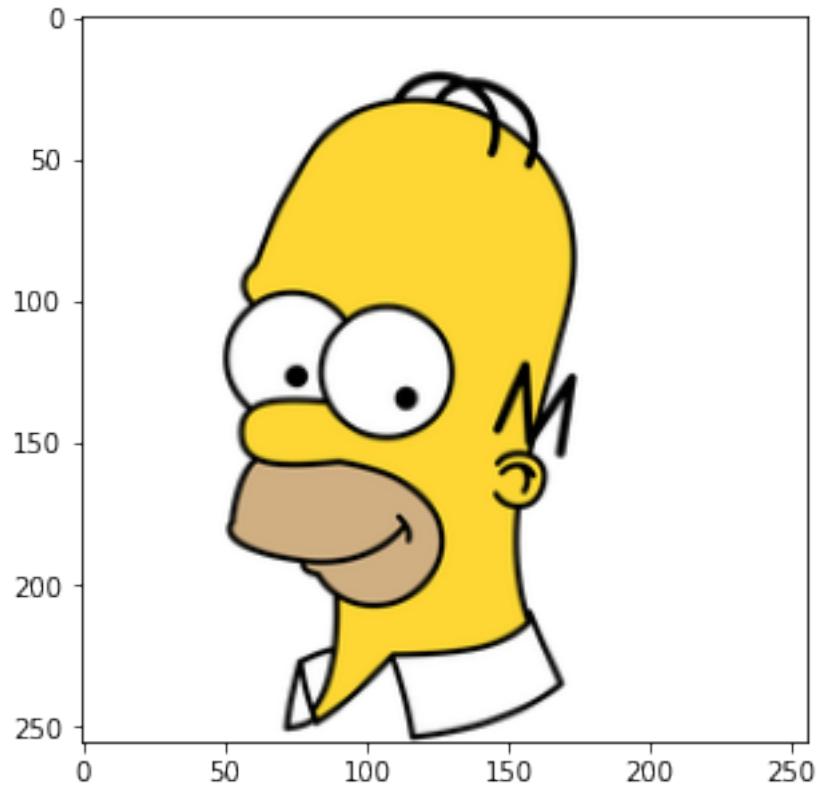
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119819250>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195BD450>



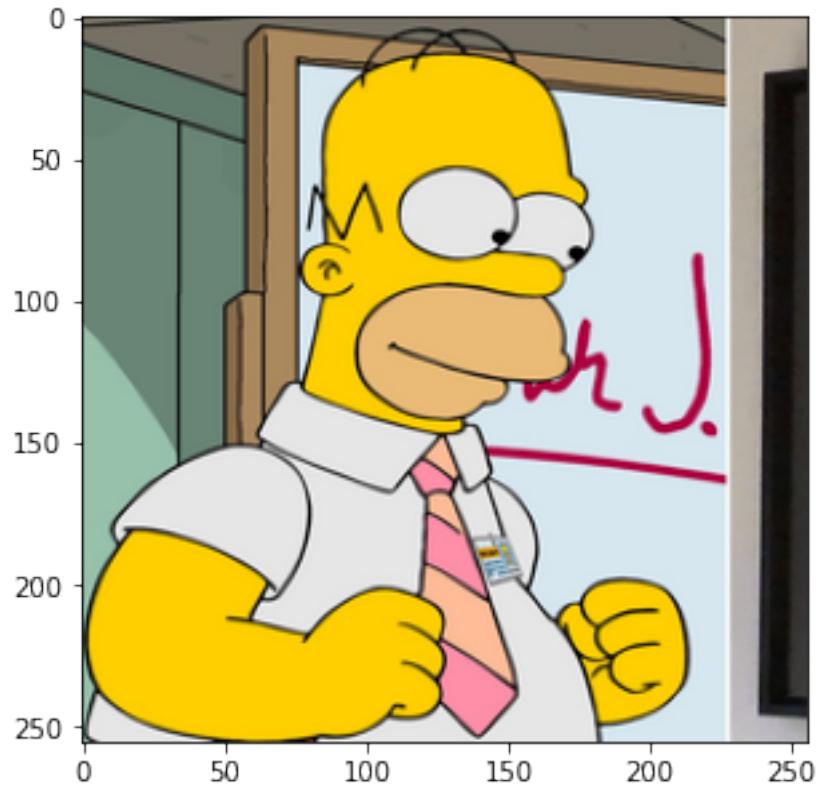
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119819090>



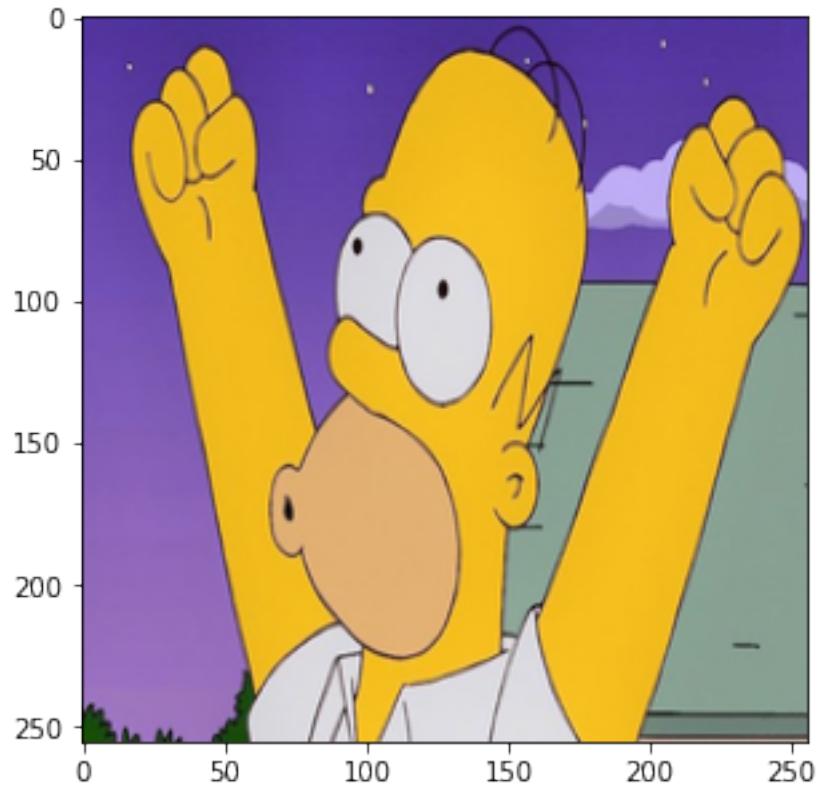
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1198198D0>



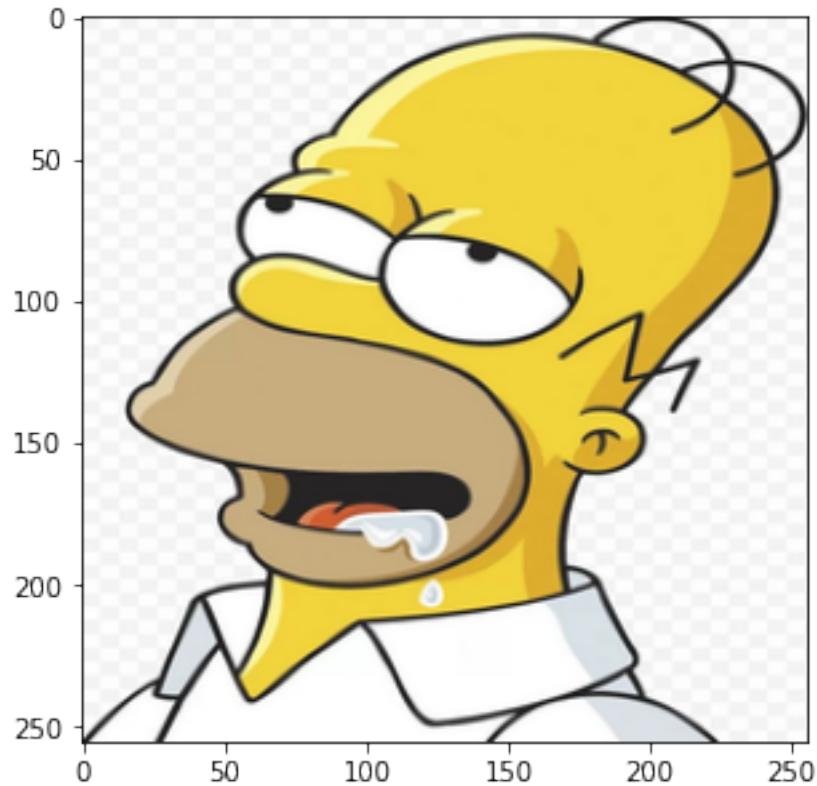
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119819790>



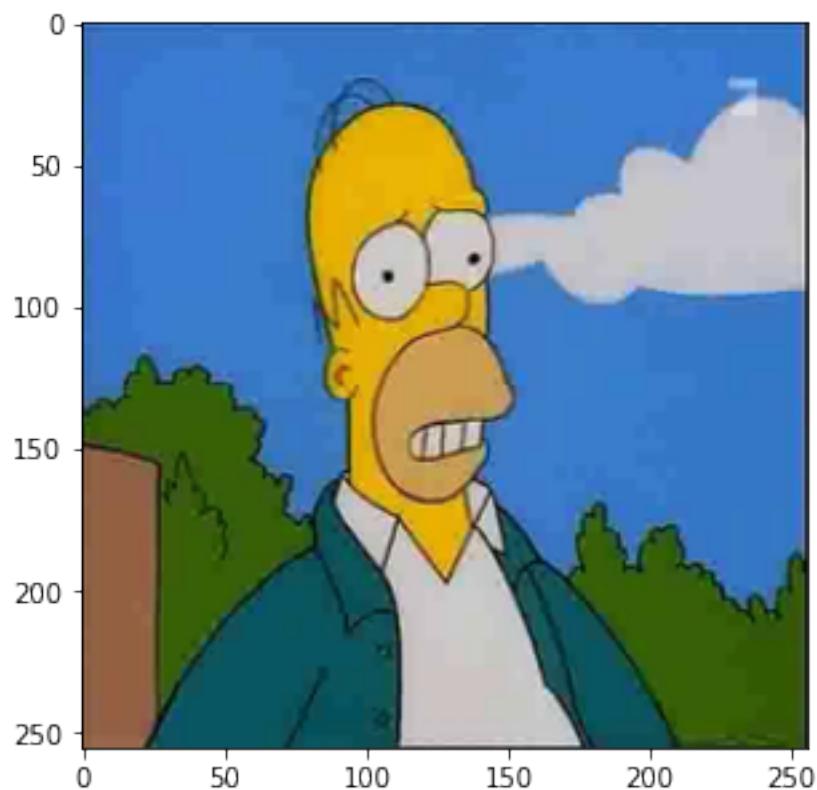
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119819450>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119819BD0>



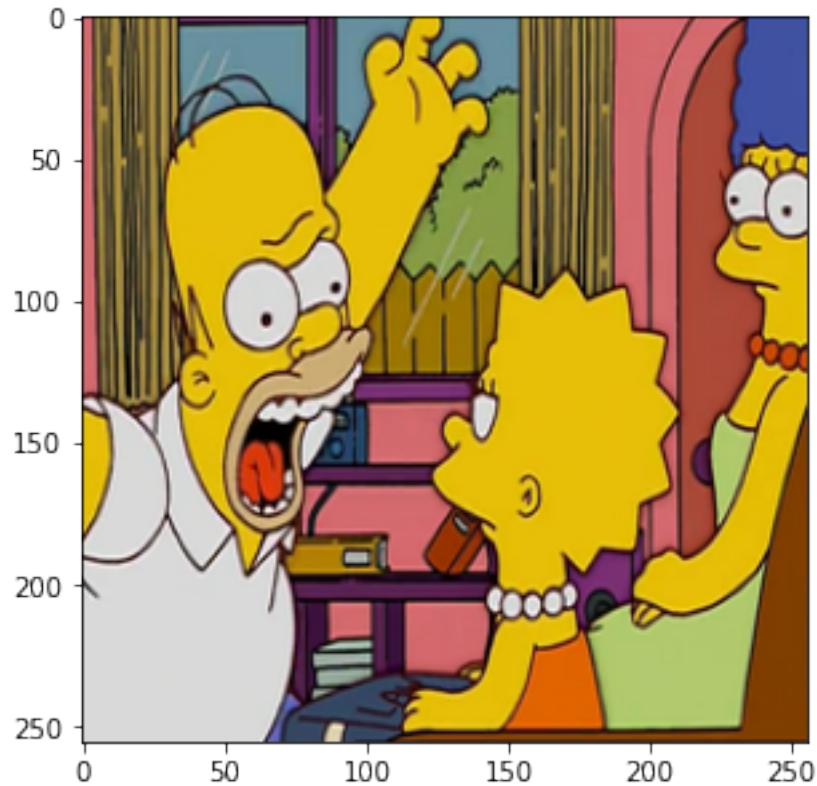
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F550>



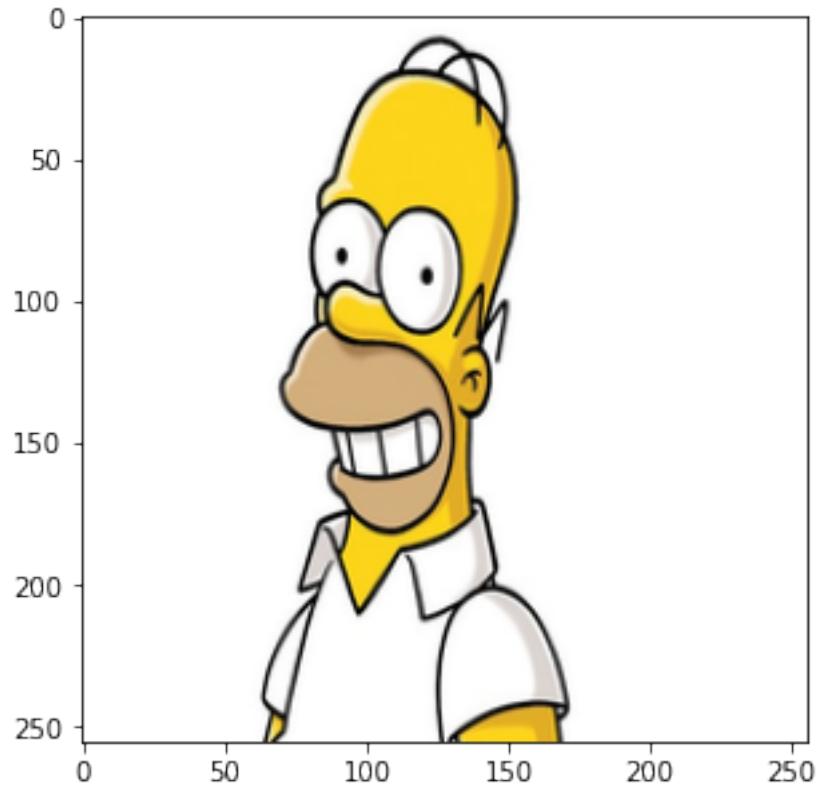
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943FD90>



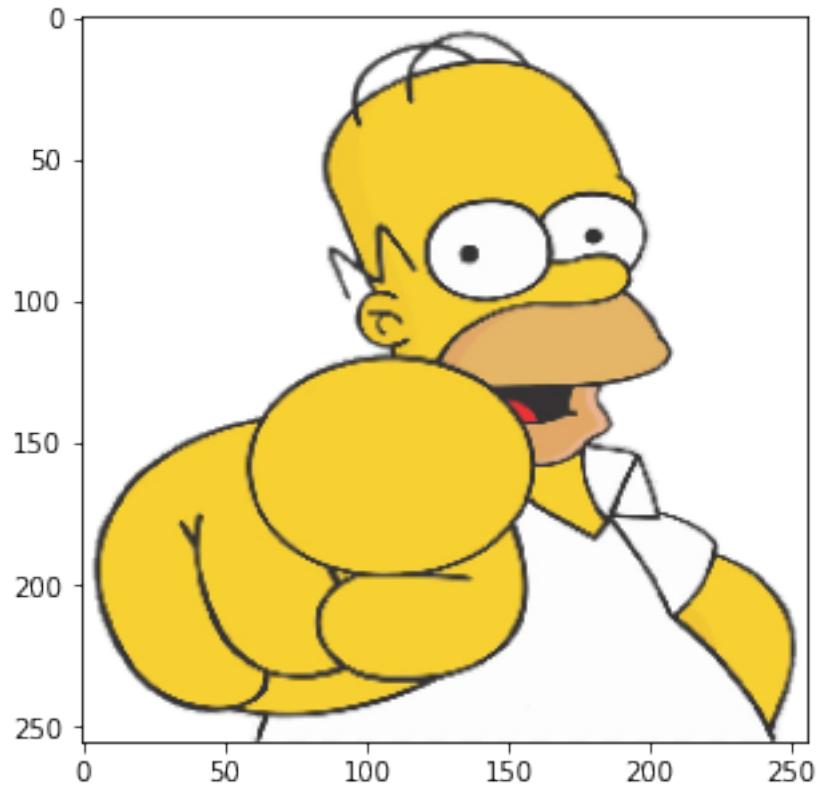
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F450>



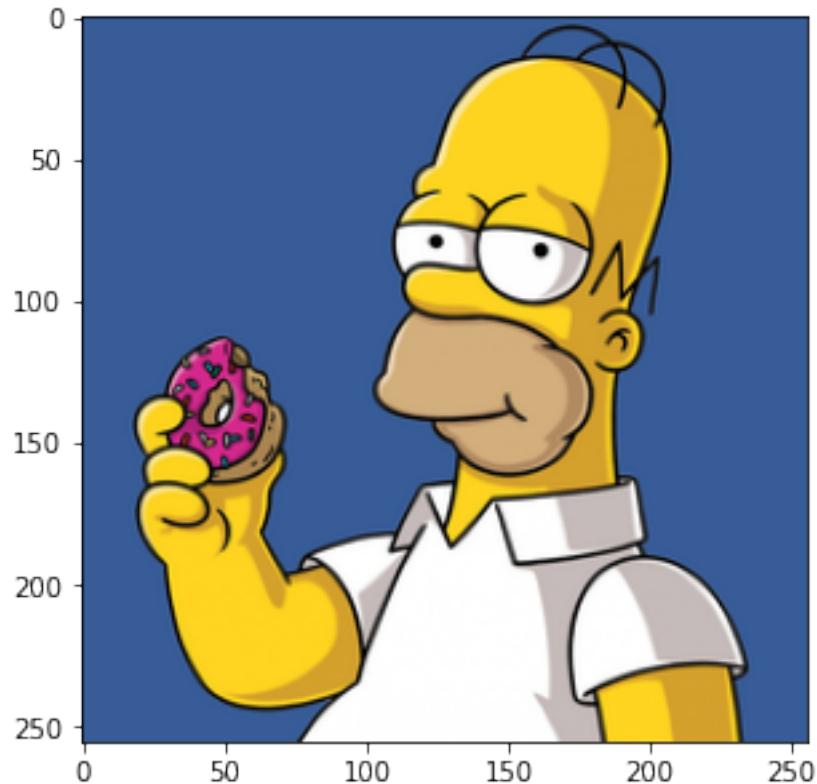
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F110>



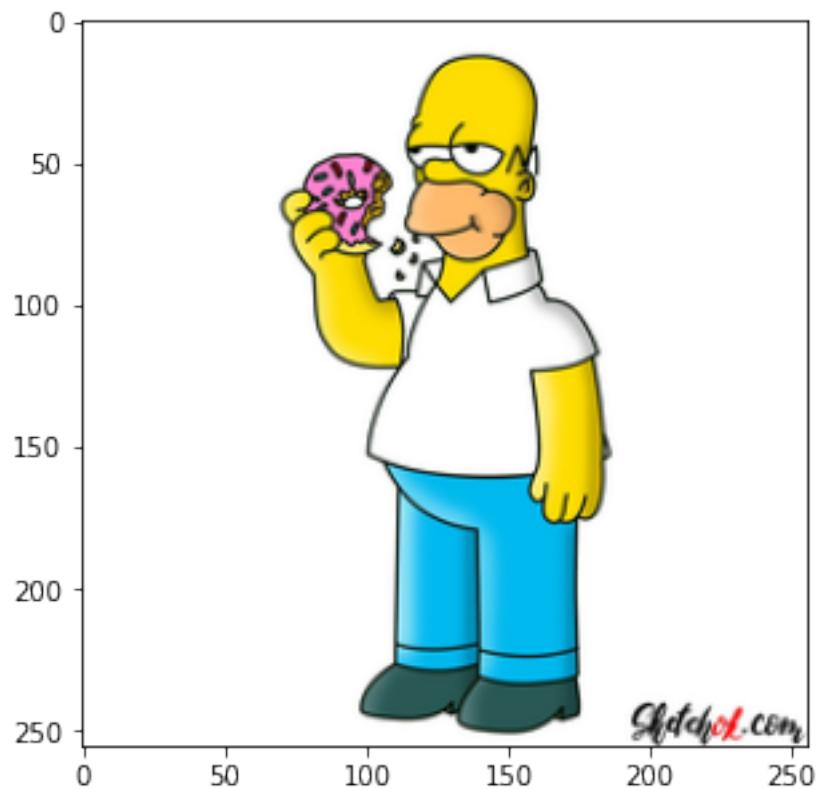
```
<PIL.Image.Image image mode=P size=256x256 at 0x7FB11943FB10>
```



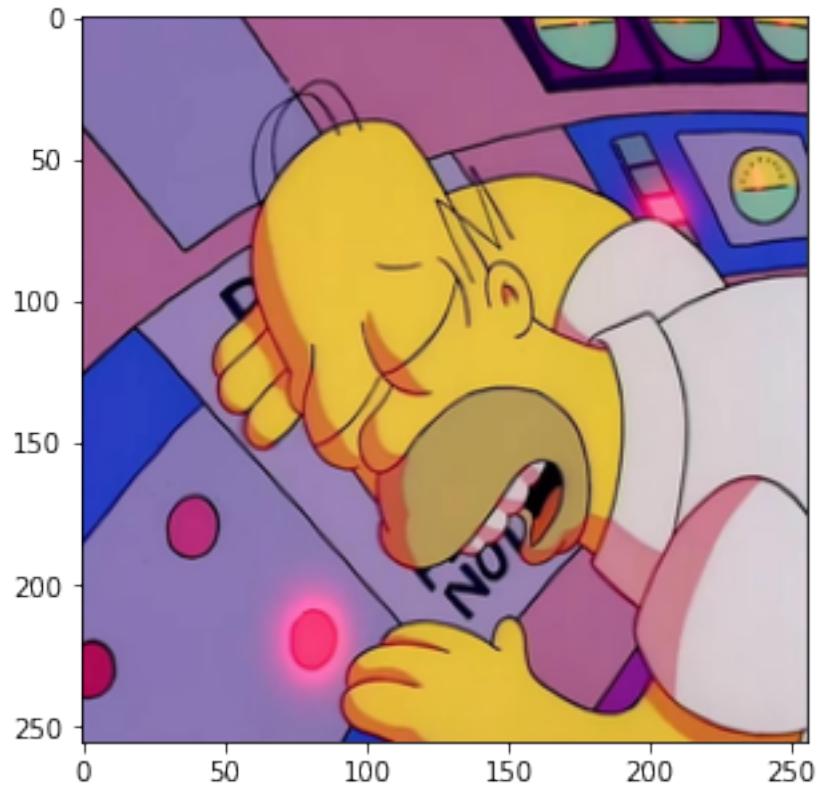
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB11943FED0>



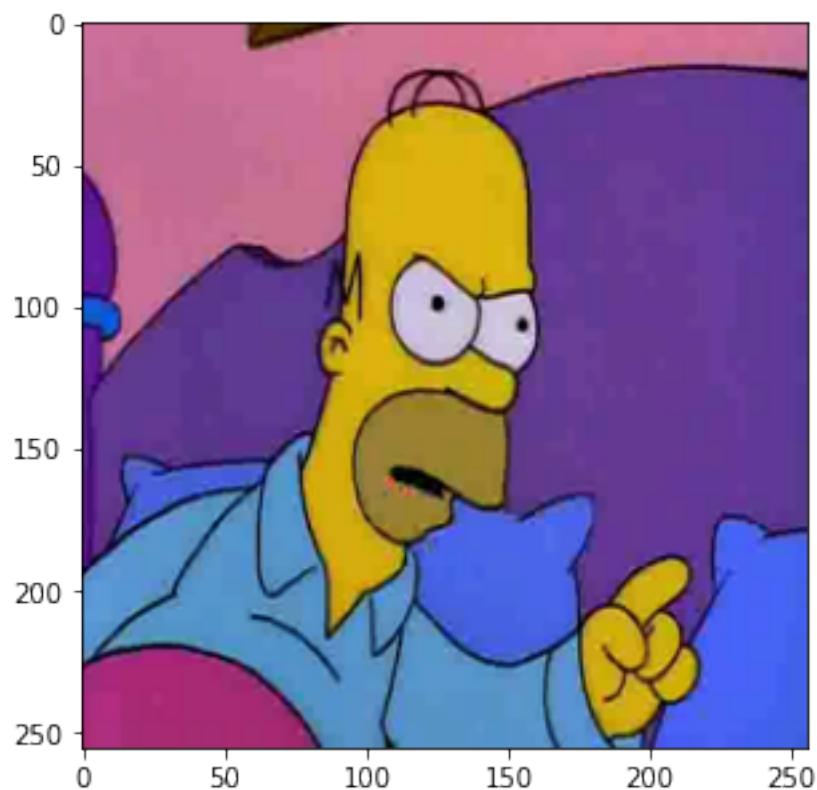
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943FF50>



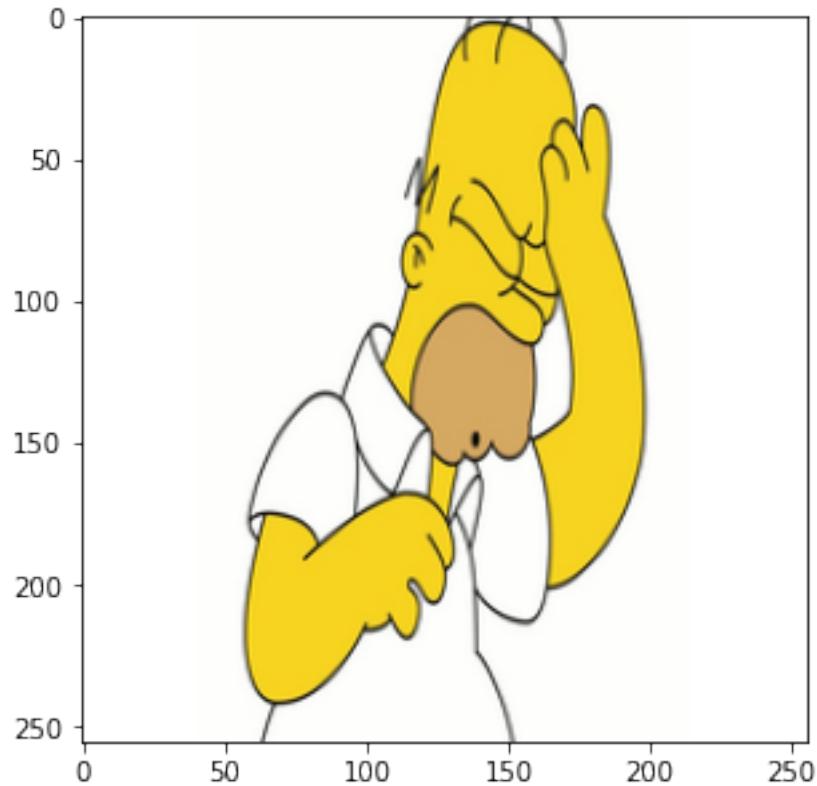
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F7D0>



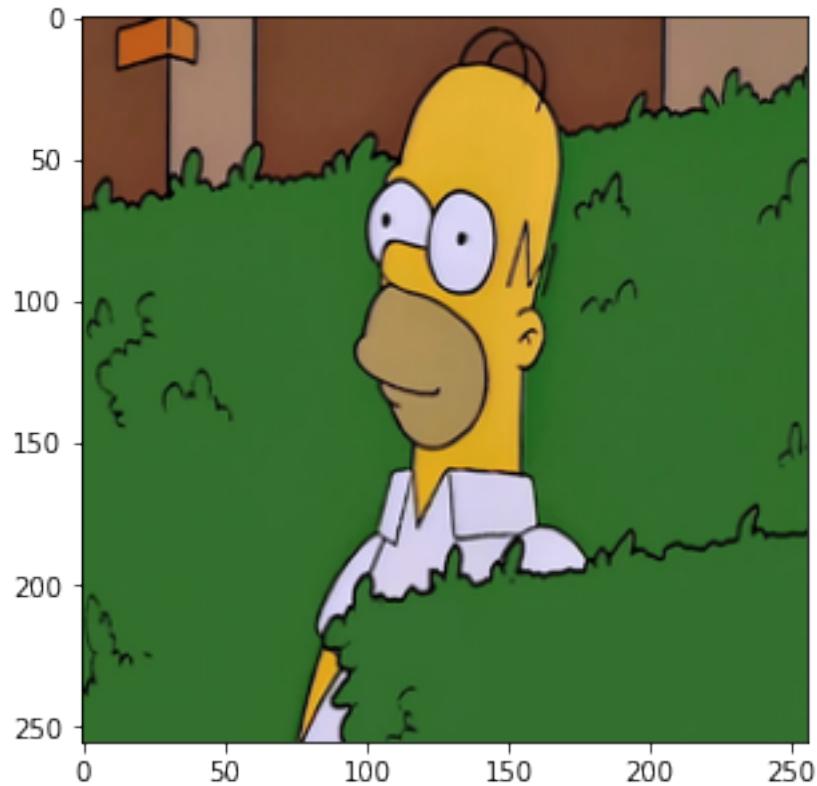
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F710>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F510>



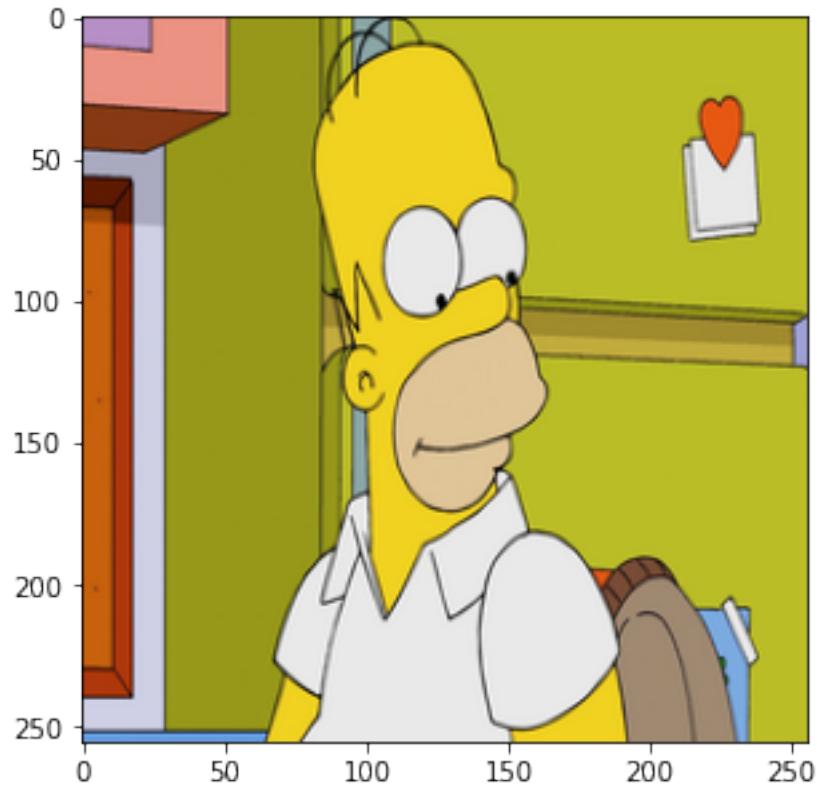
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F090>



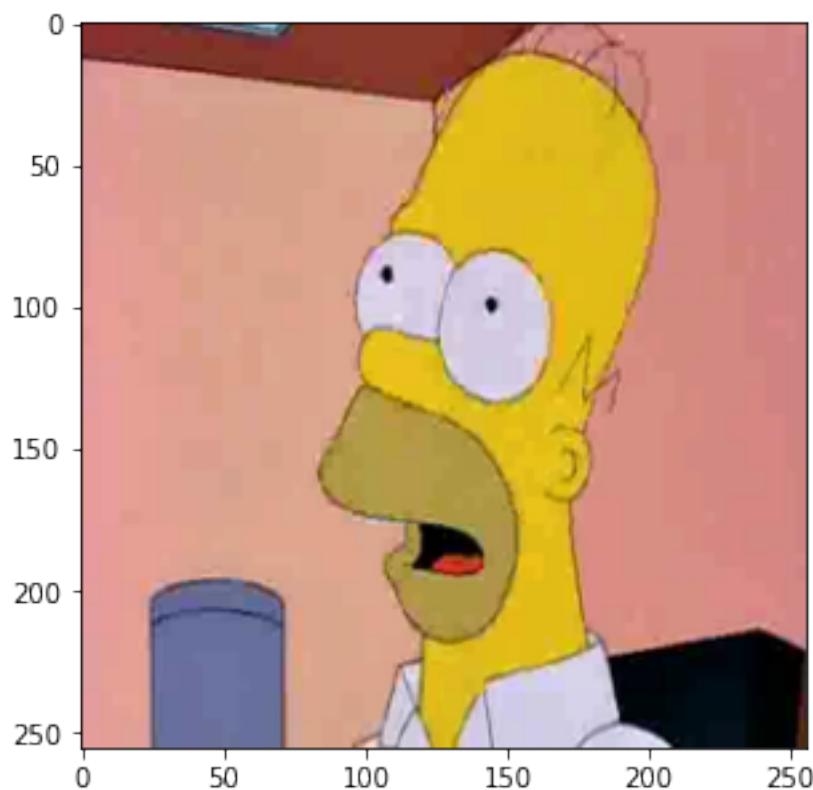
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F850>



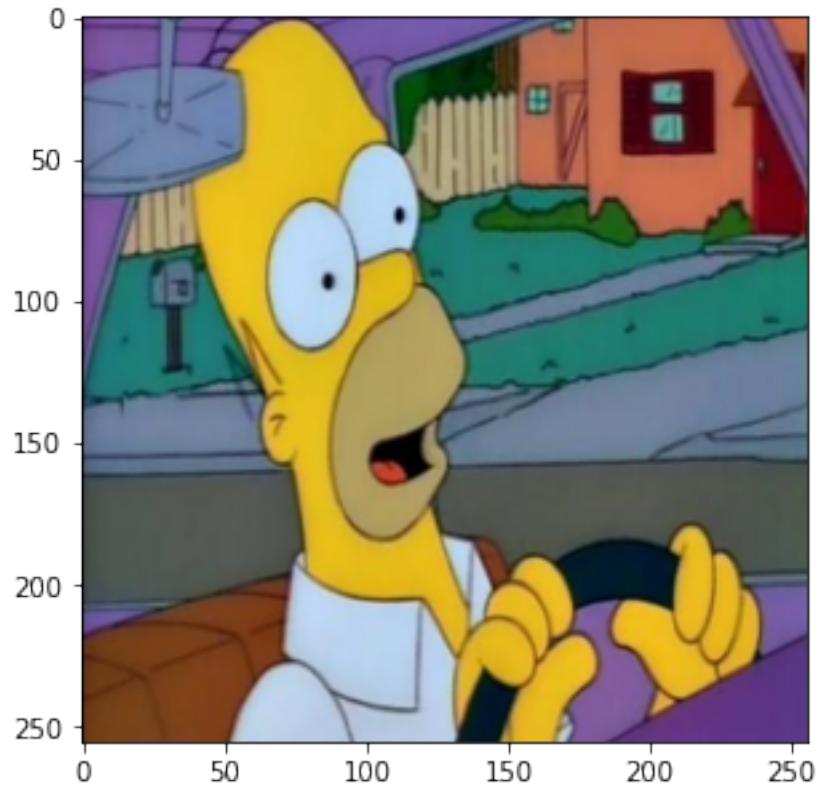
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195425D0>



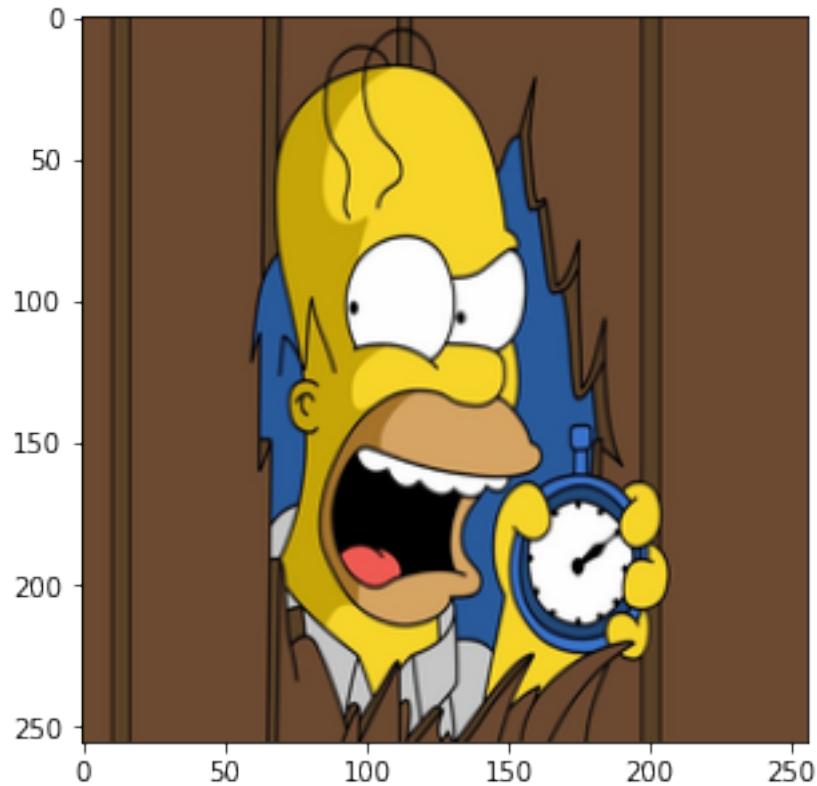
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11980DFD0>



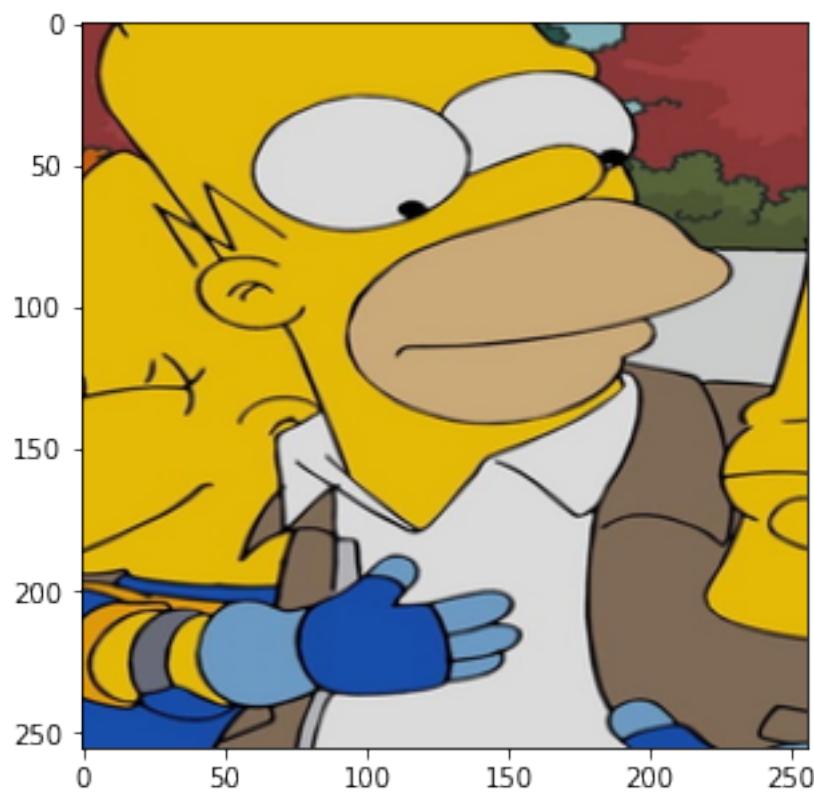
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119284DD0>



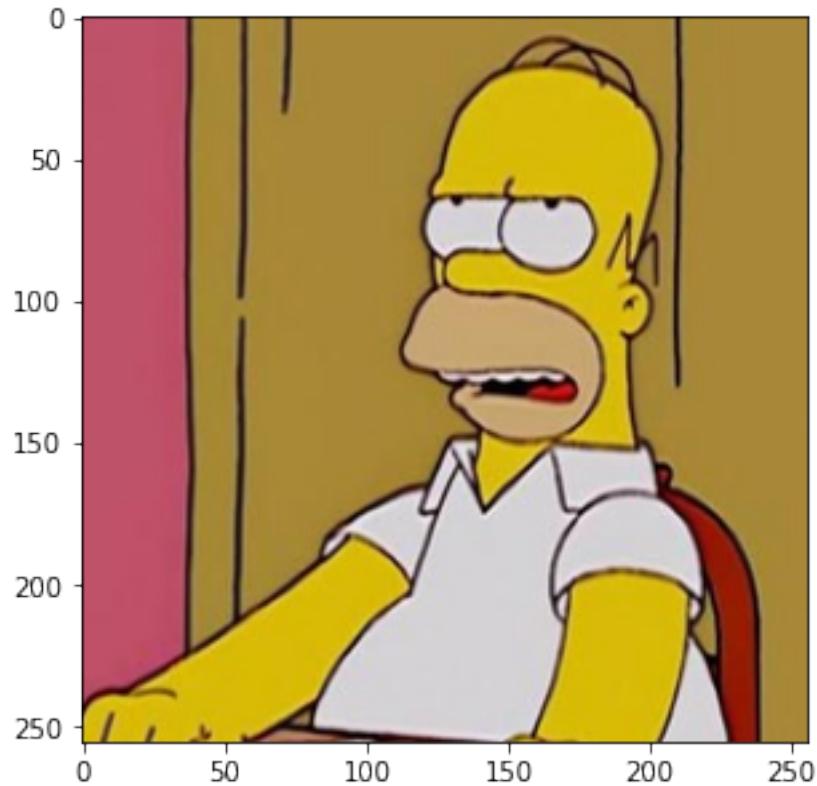
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119284F50>



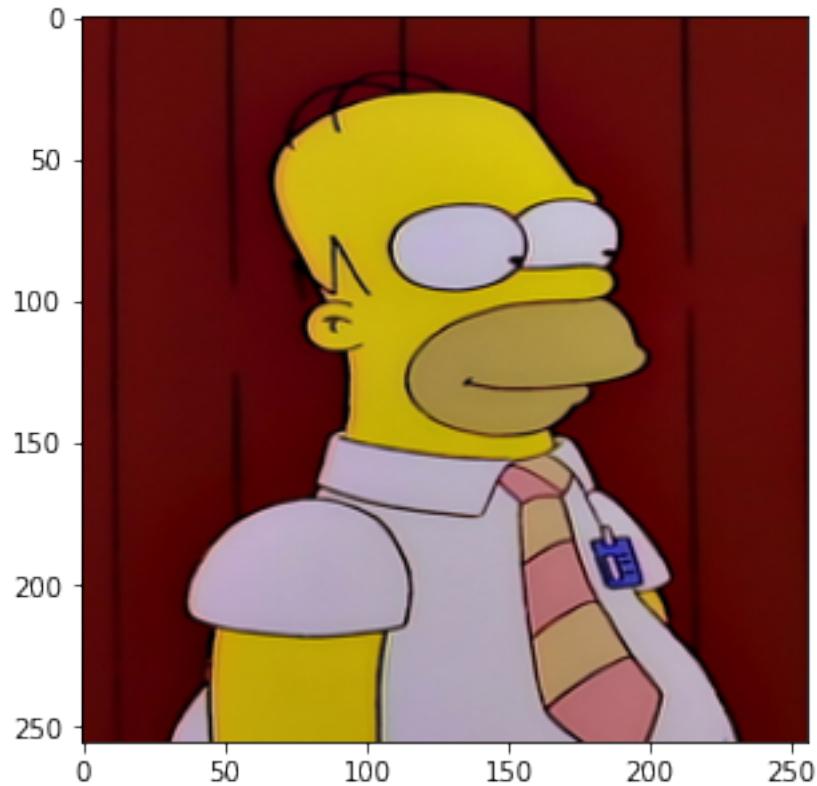
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119284FD0>



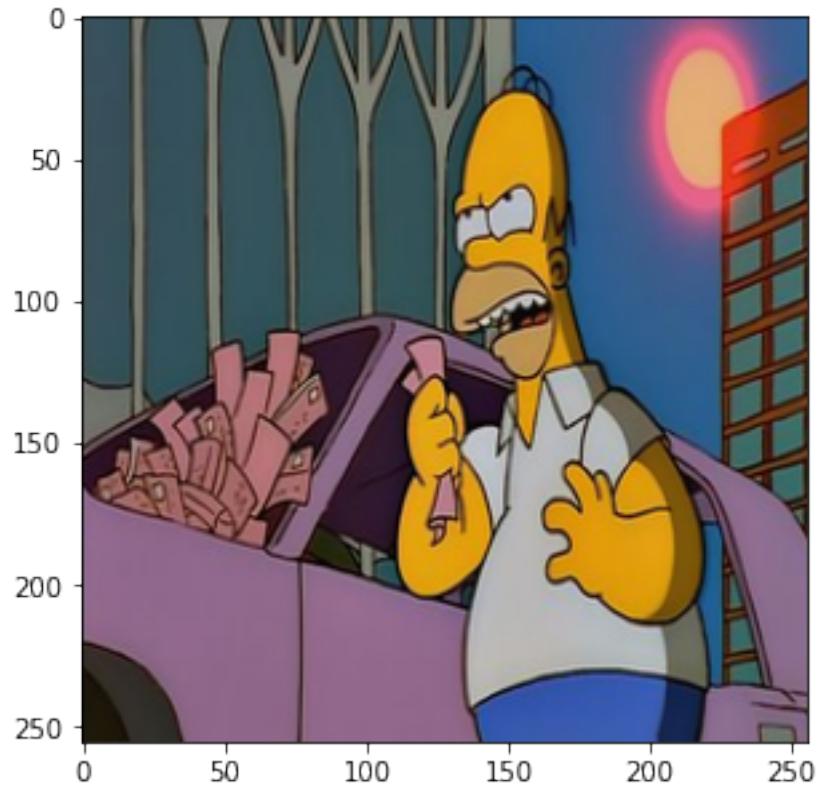
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1194553D0>



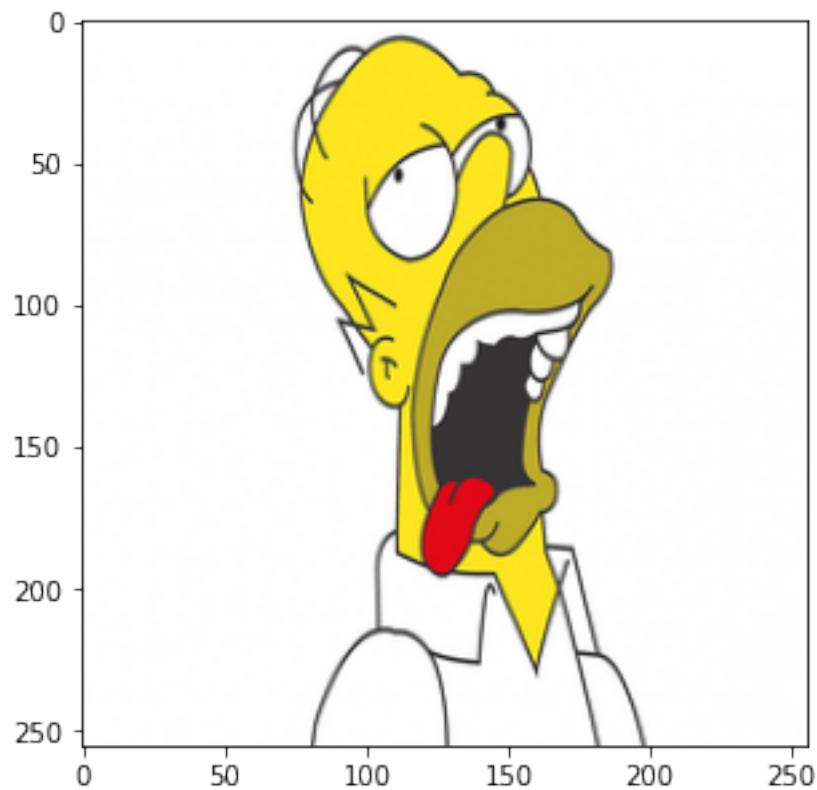
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119455650>



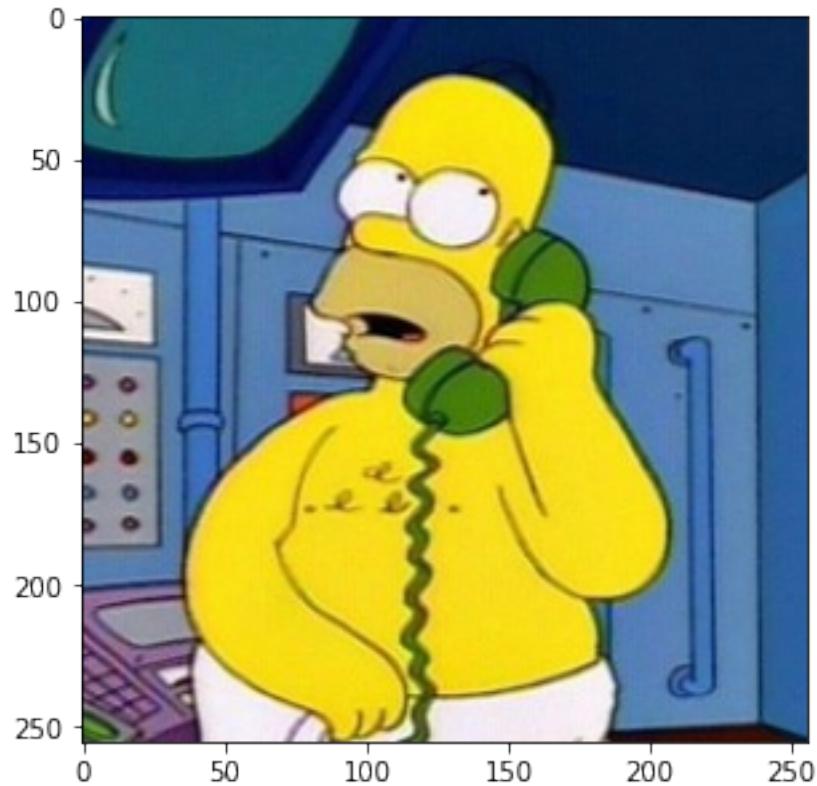
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119455590>



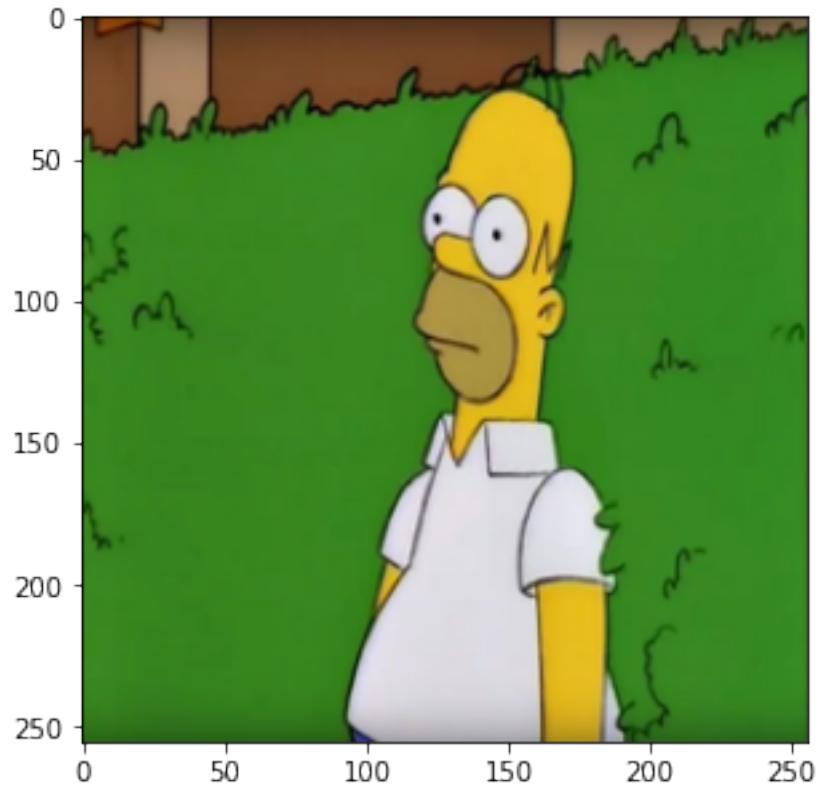
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1194550D0>



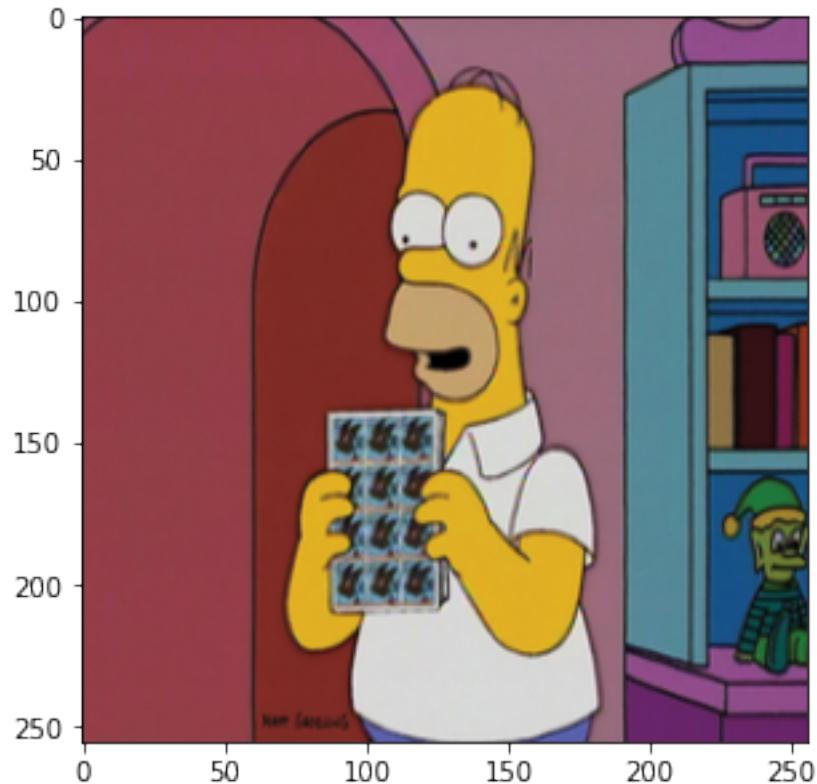
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1194551D0>



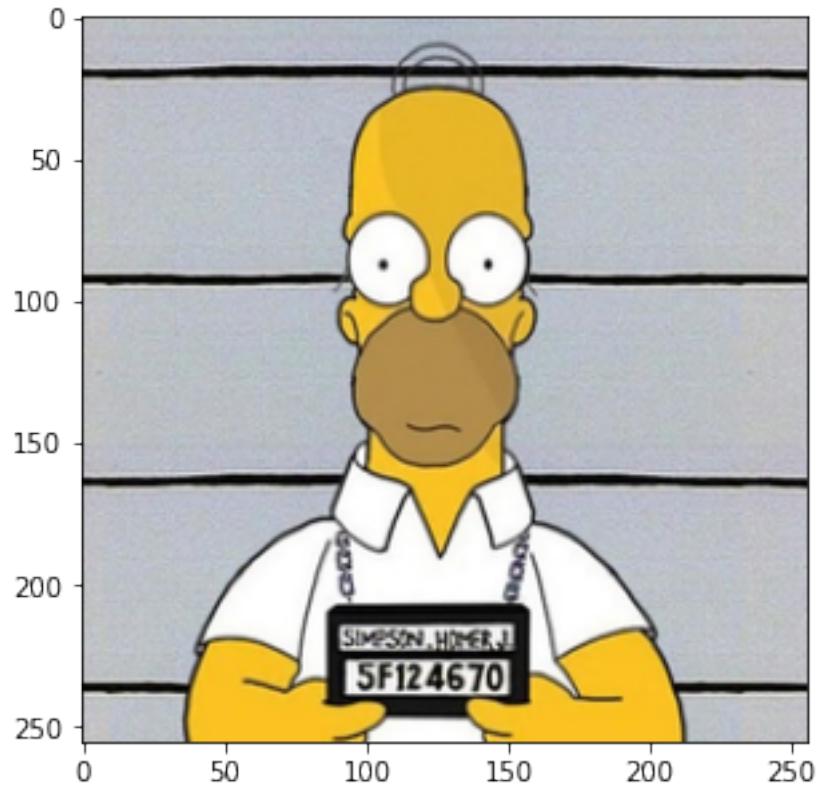
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119455350>



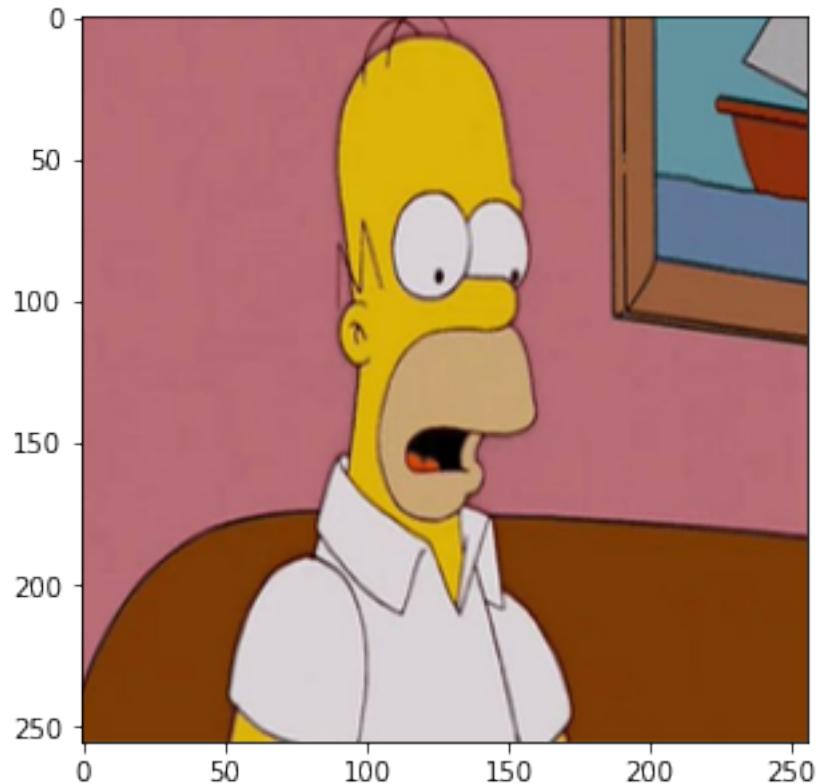
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119455DD0>



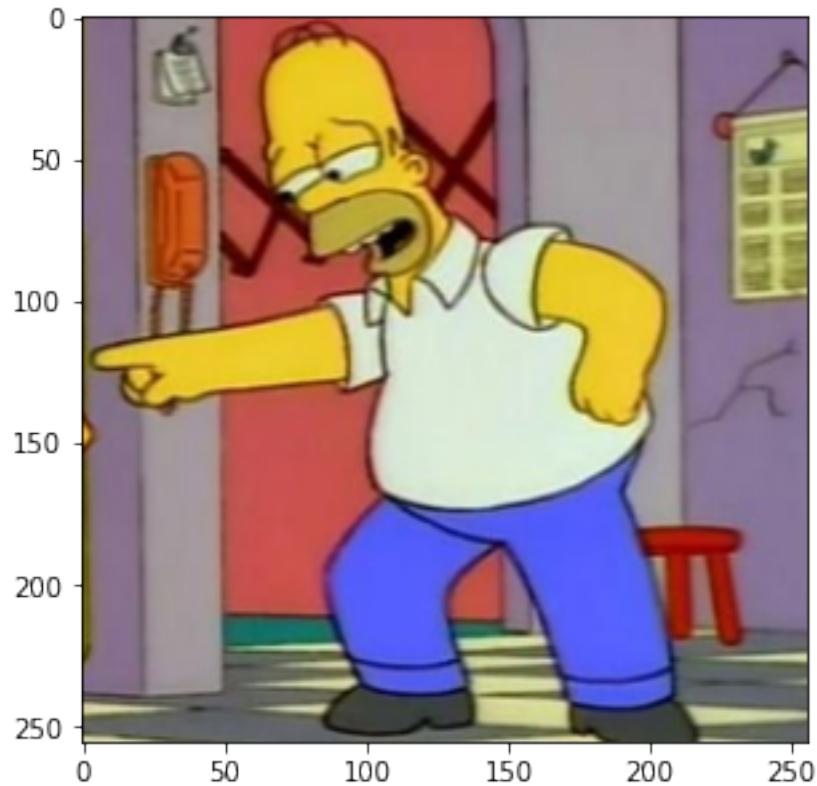
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119455F10>



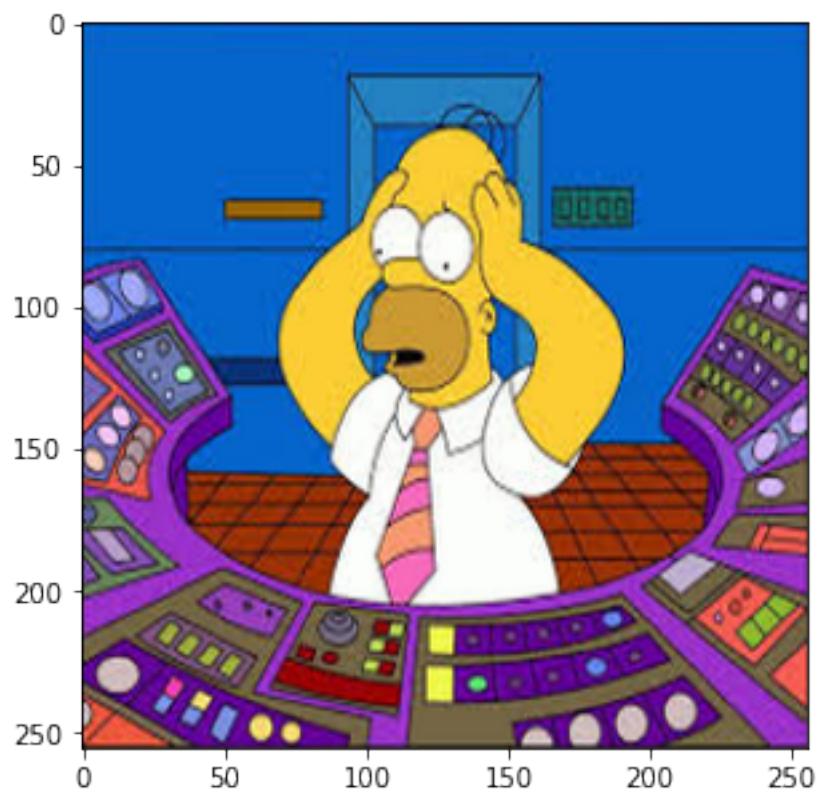
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119455410>



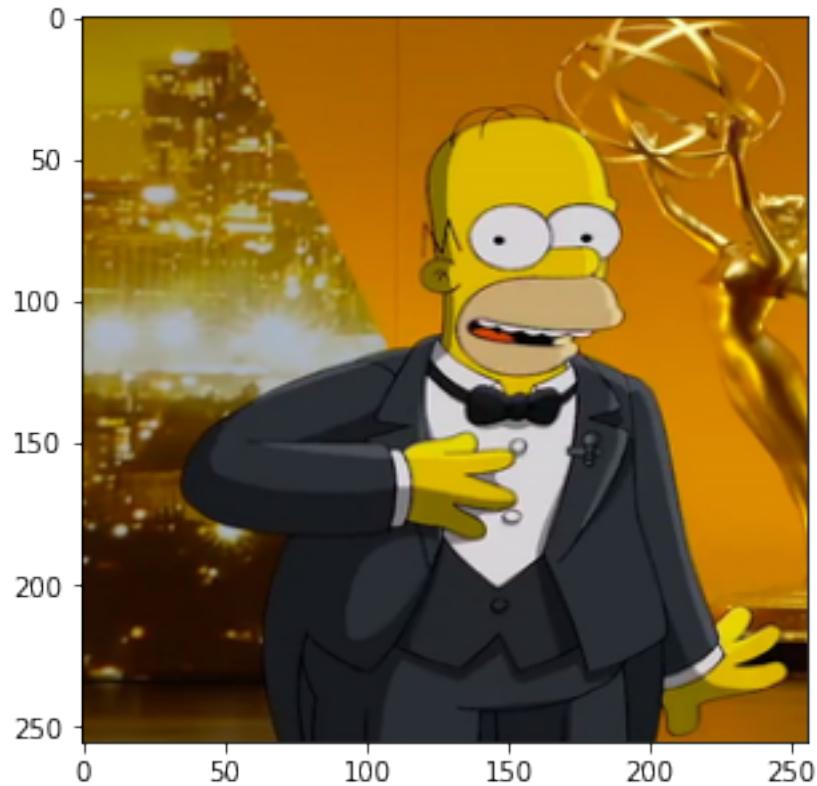
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119455C50>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119455A10>



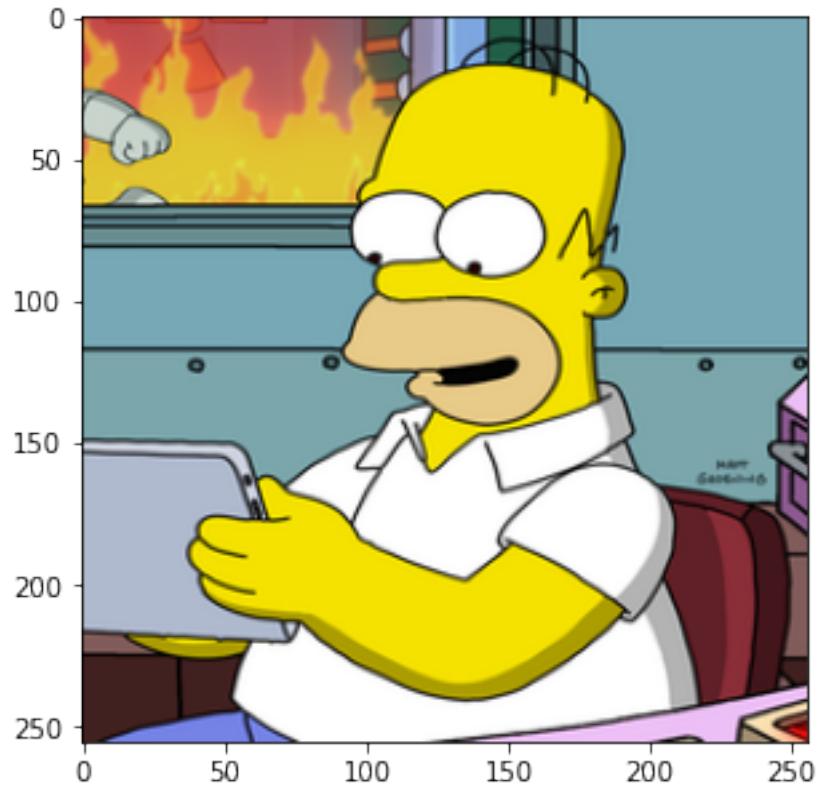
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119455890>



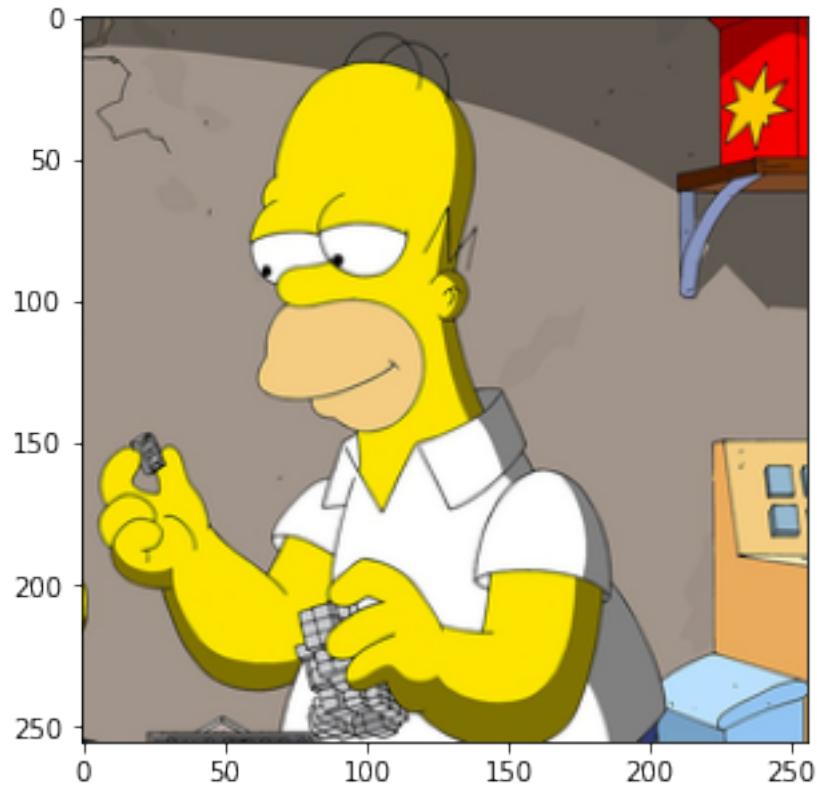
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F250>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F0D0>

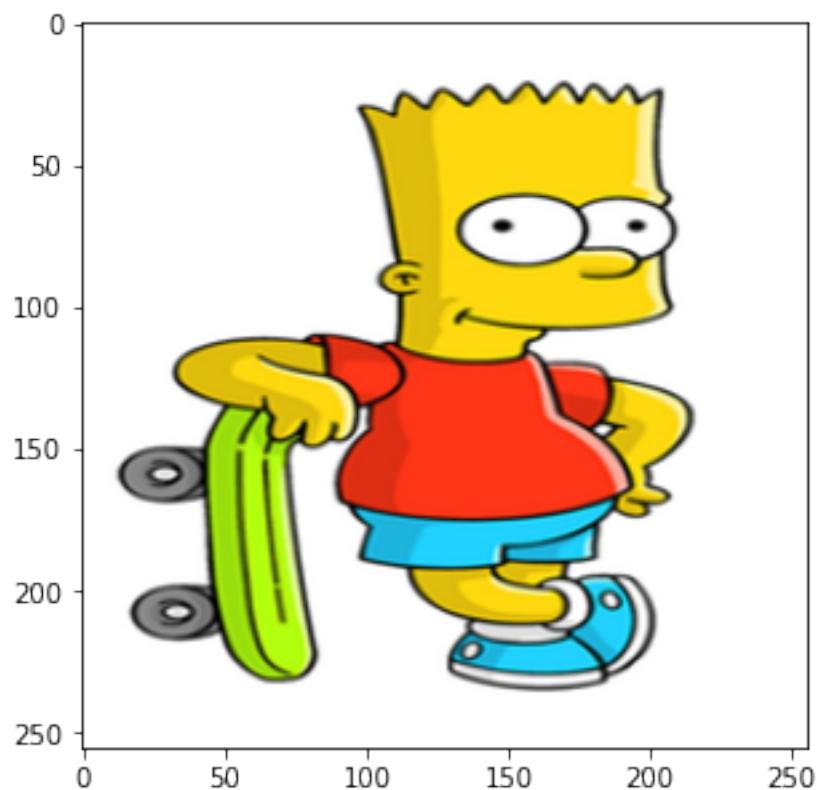


<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F310>

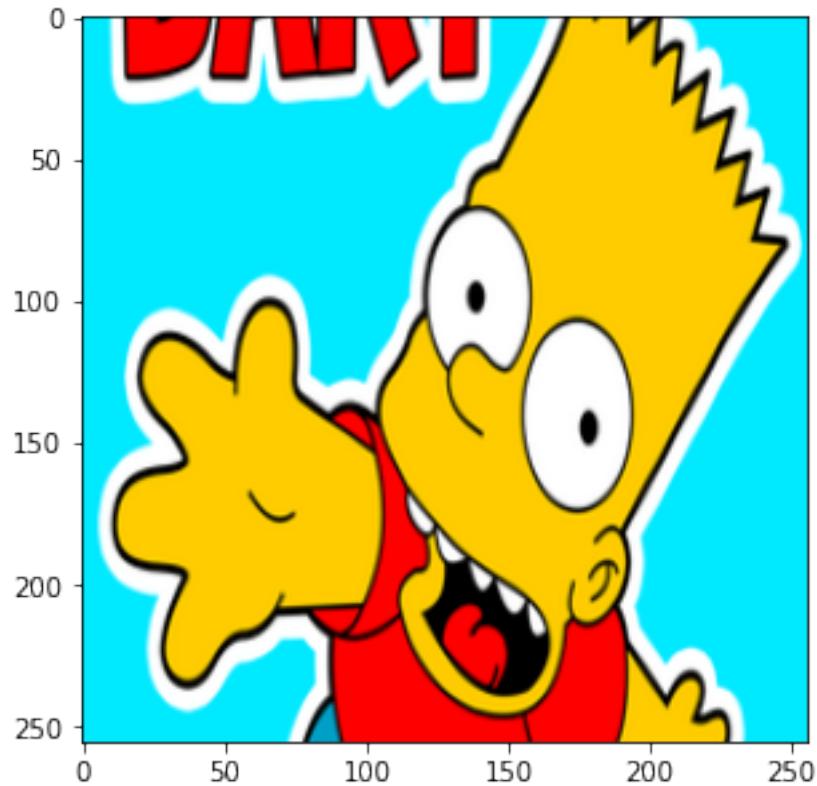


Bart -- Resized

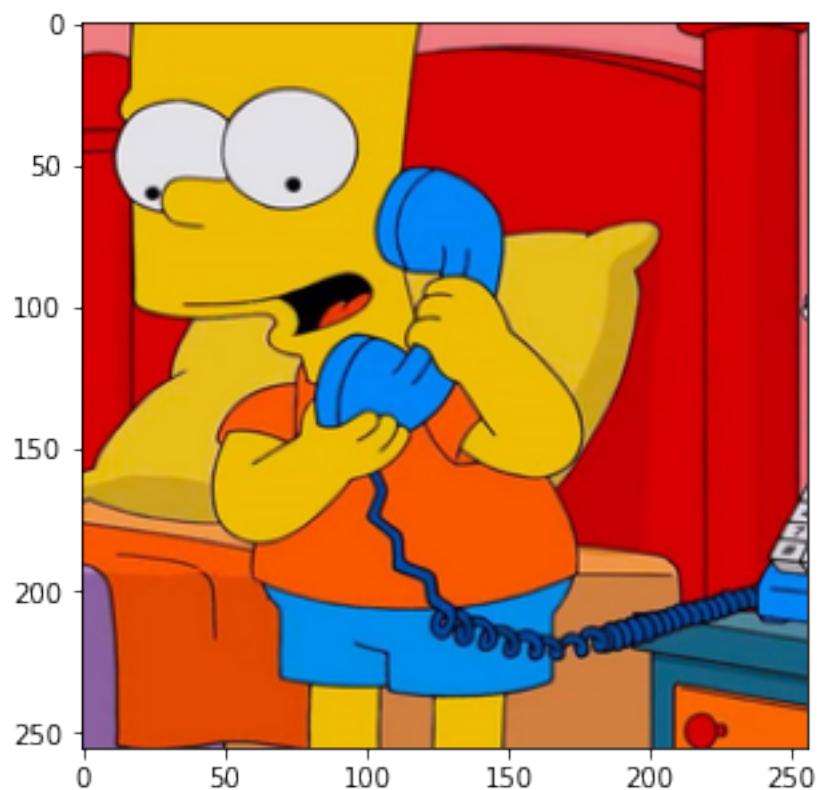
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F4D0>



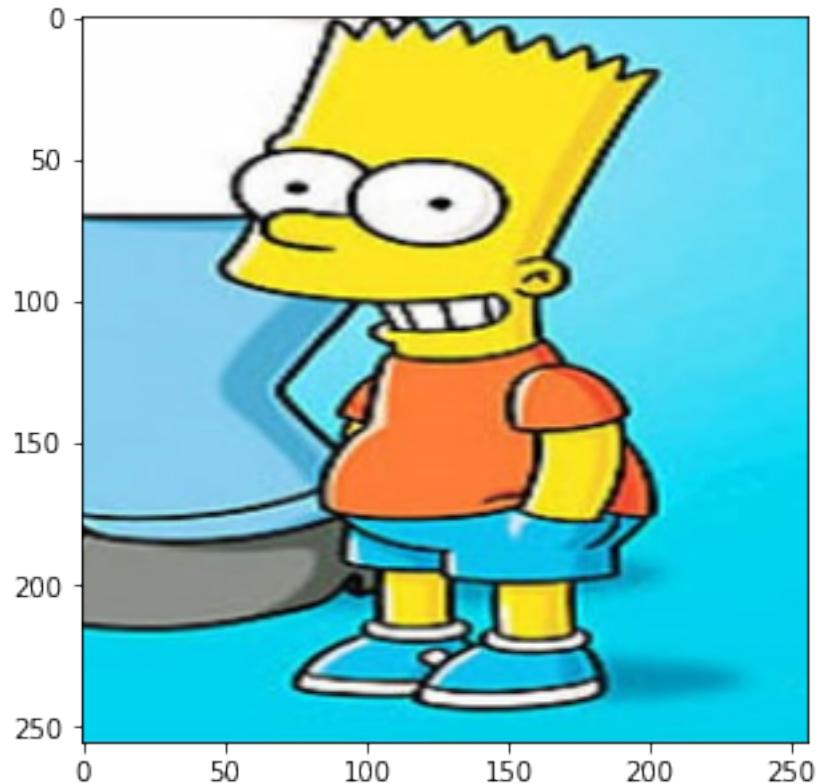
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1195FCF90>



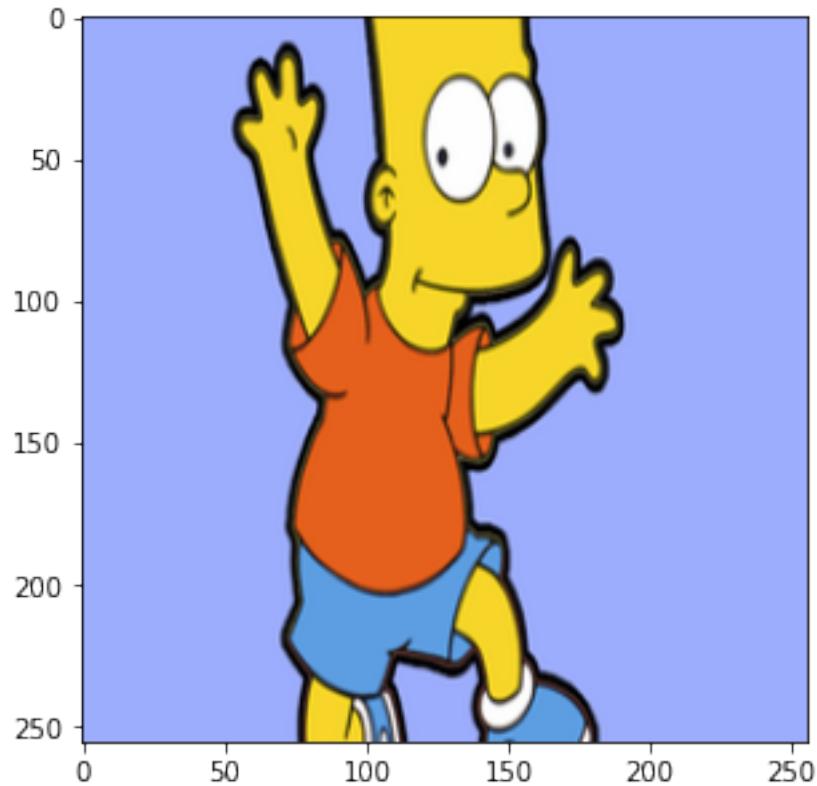
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F590>



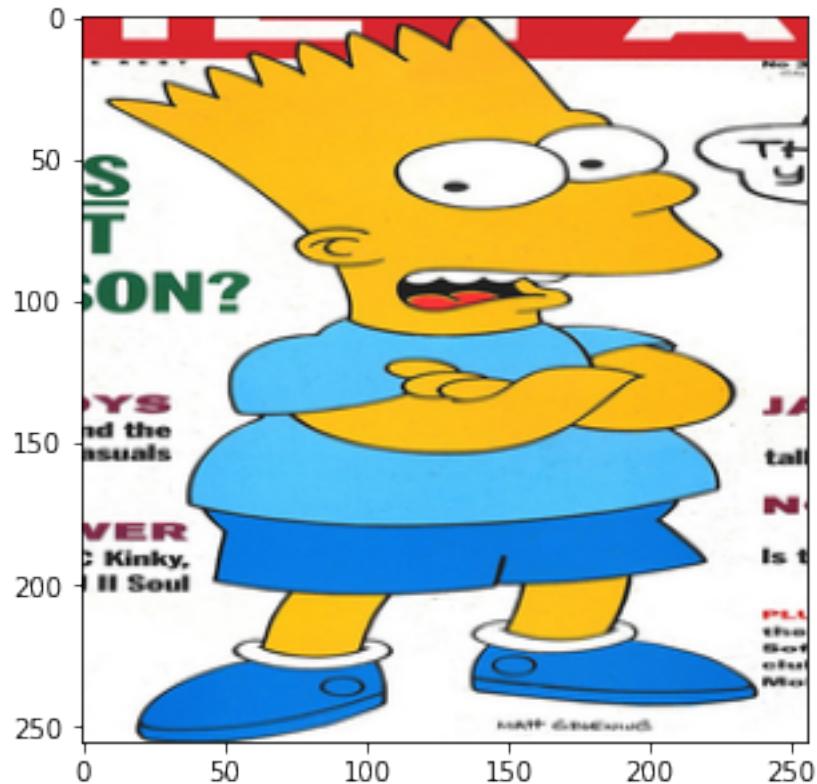
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC250>



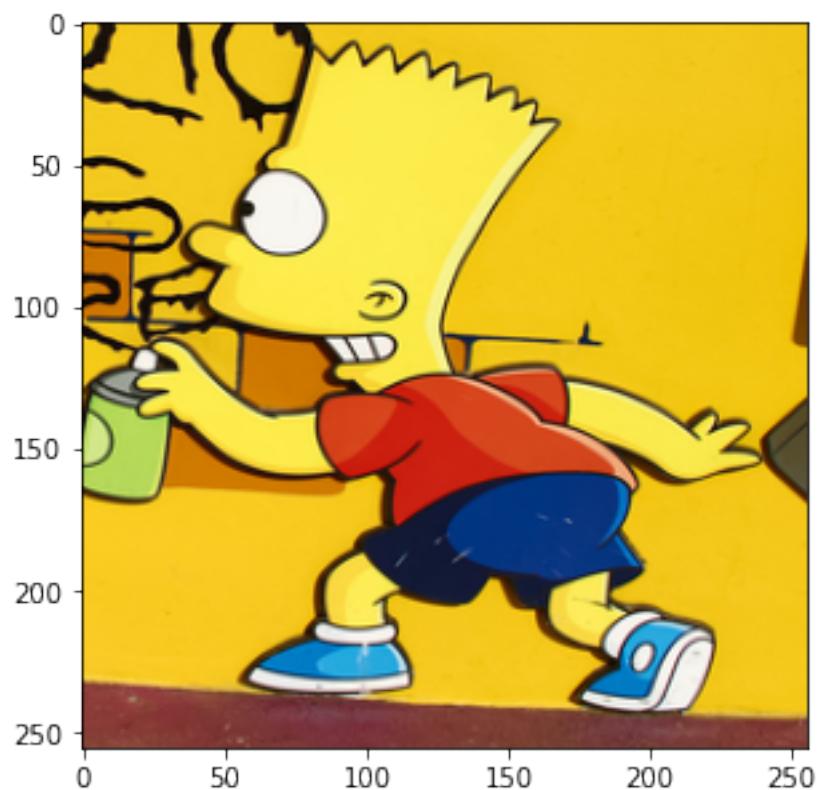
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1195FC210>



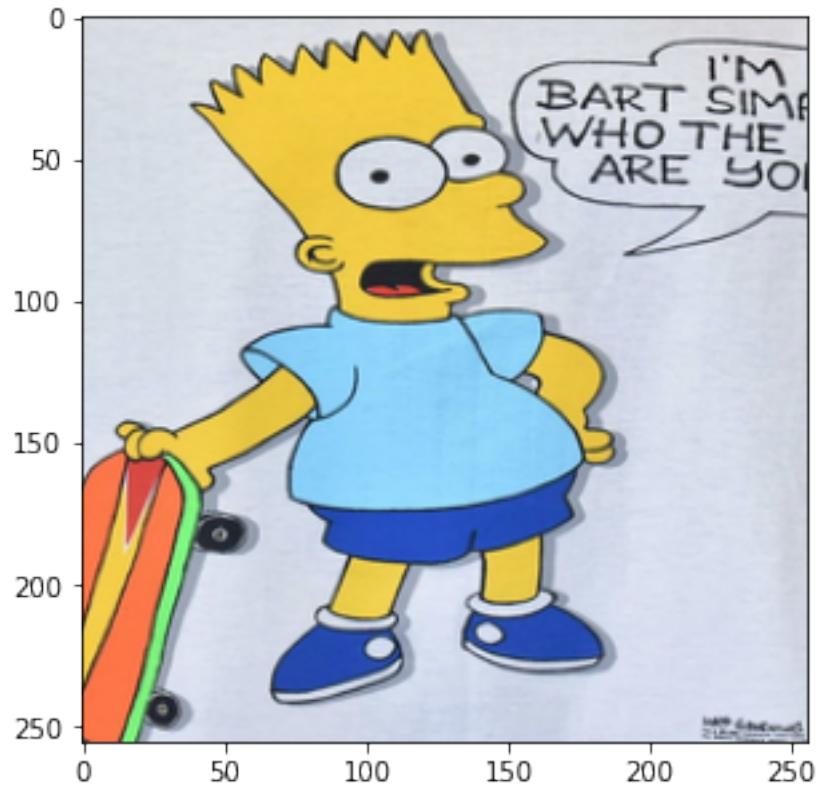
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FCBD0>



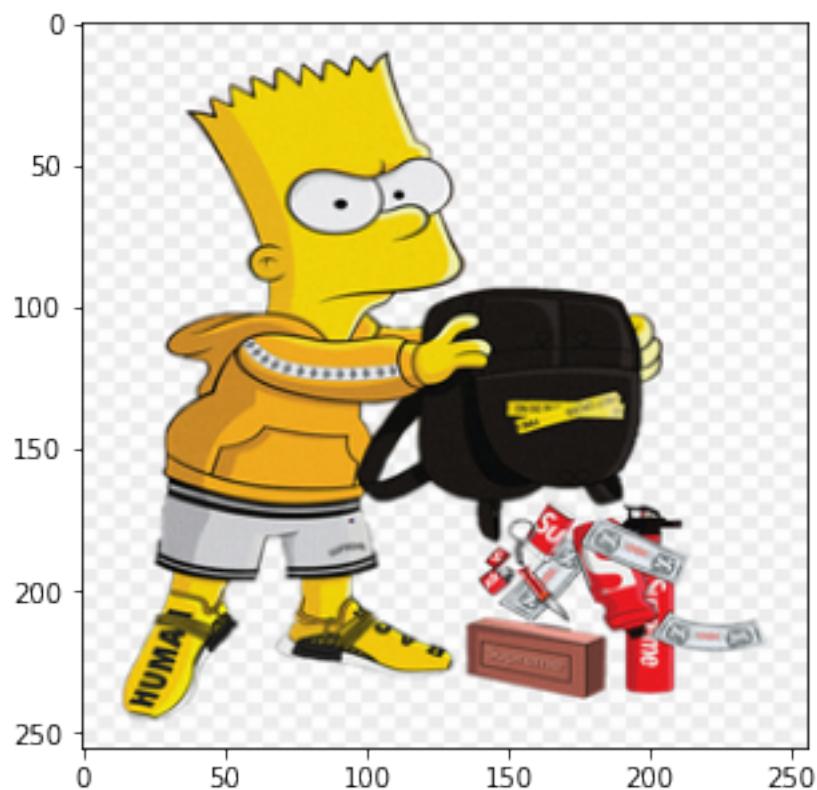
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC790>



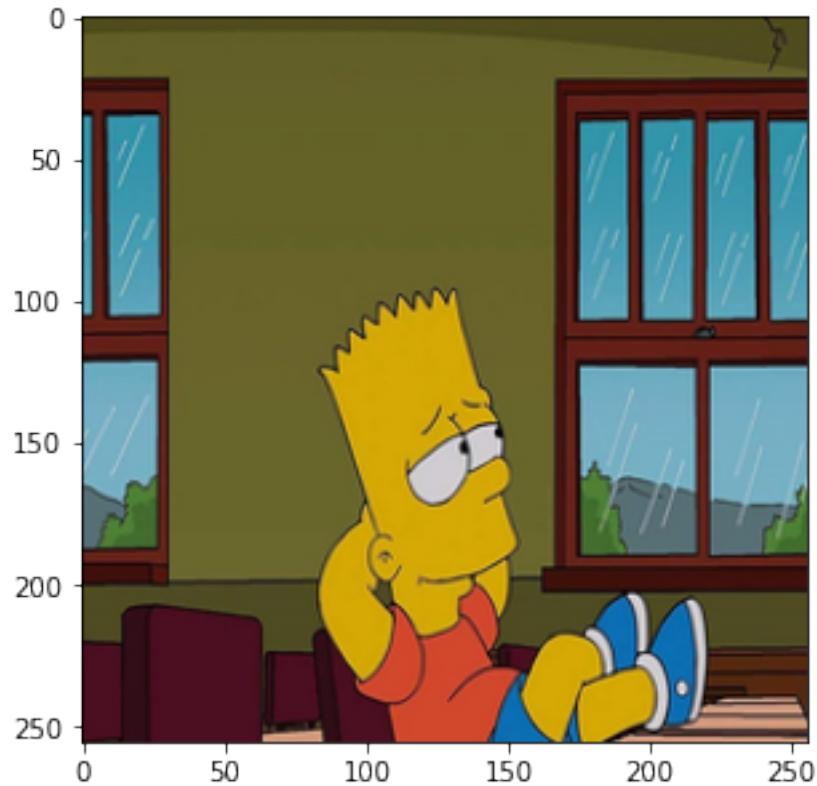
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FCCD0>



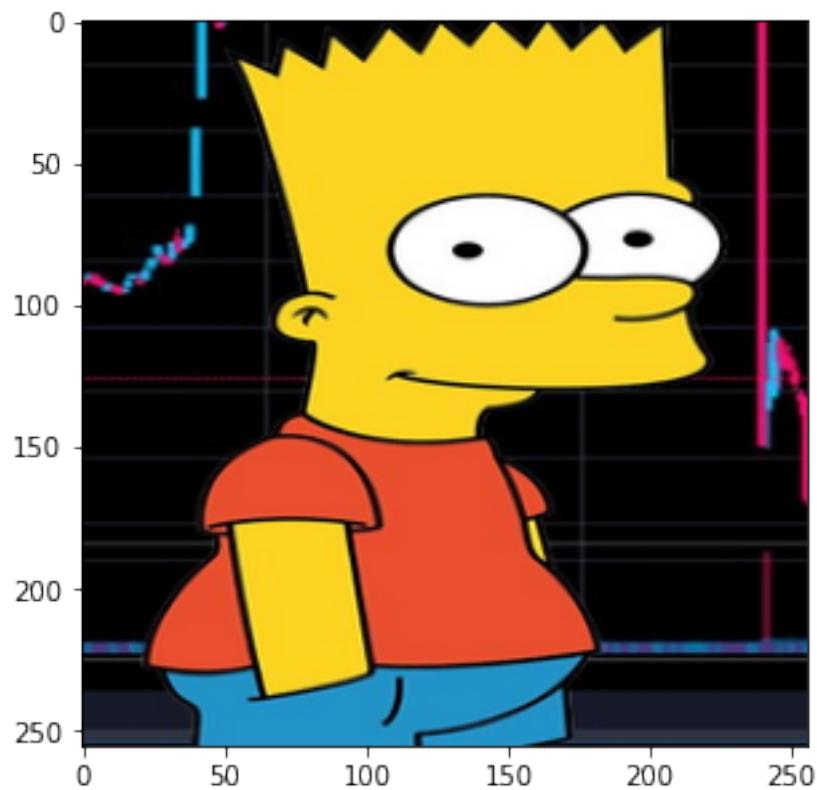
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FCB50>



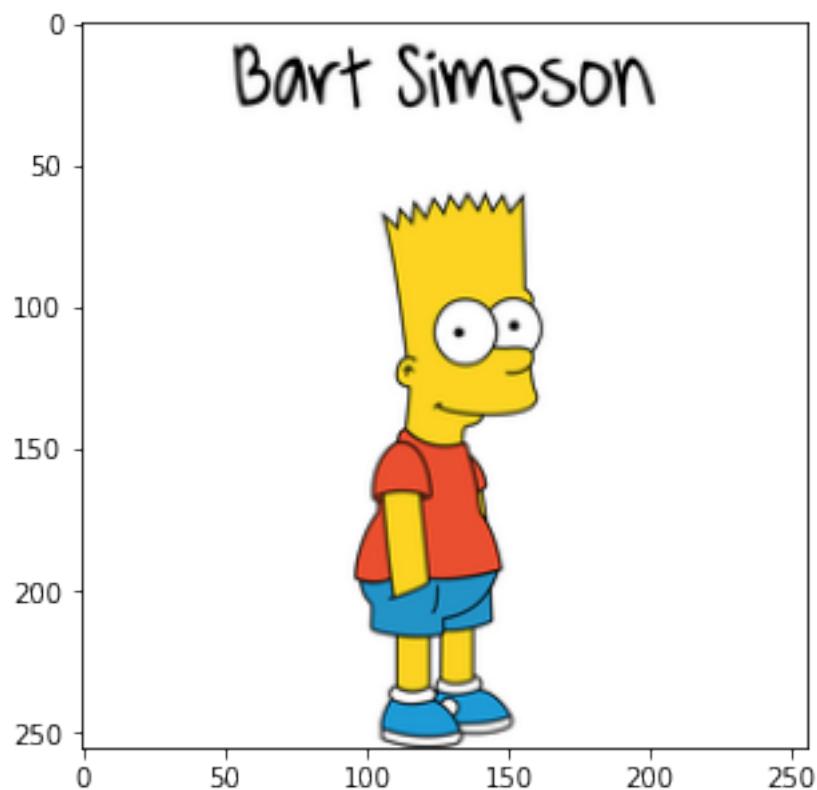
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FCE90>



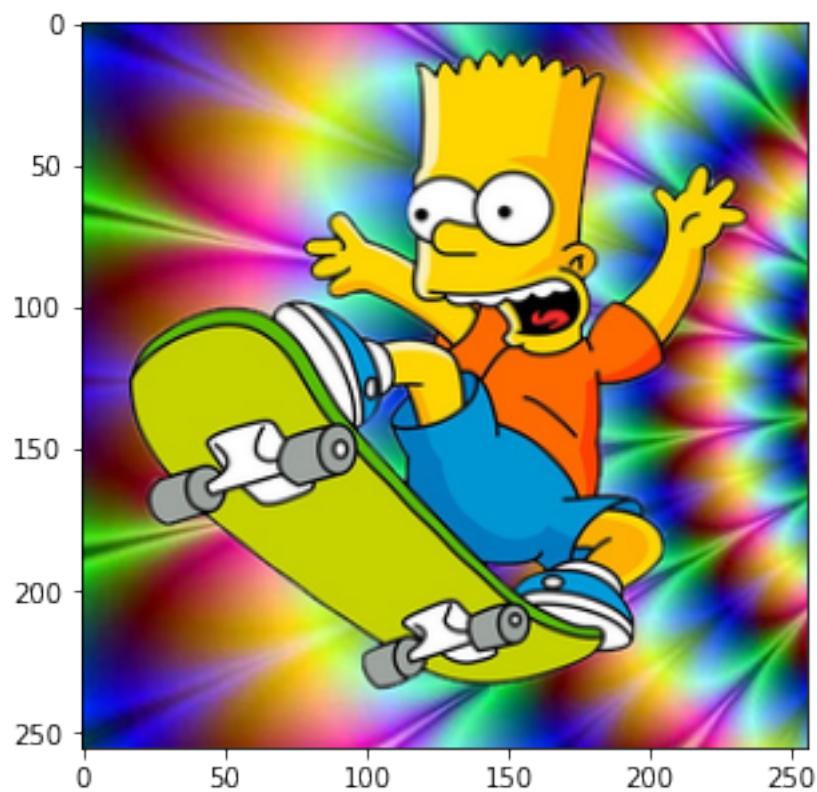
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FCAD0>



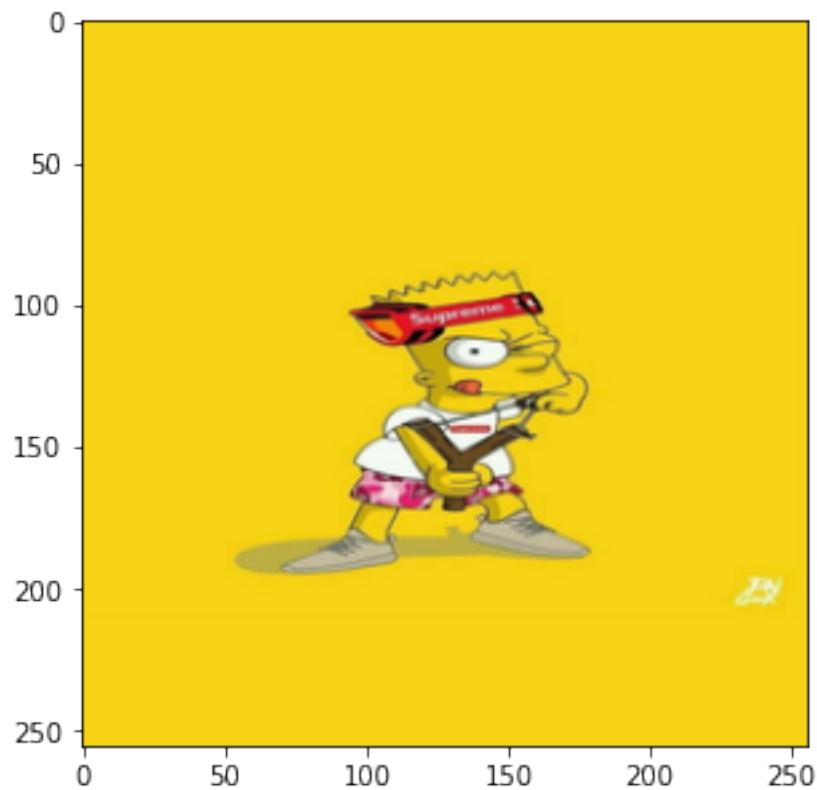
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC990>



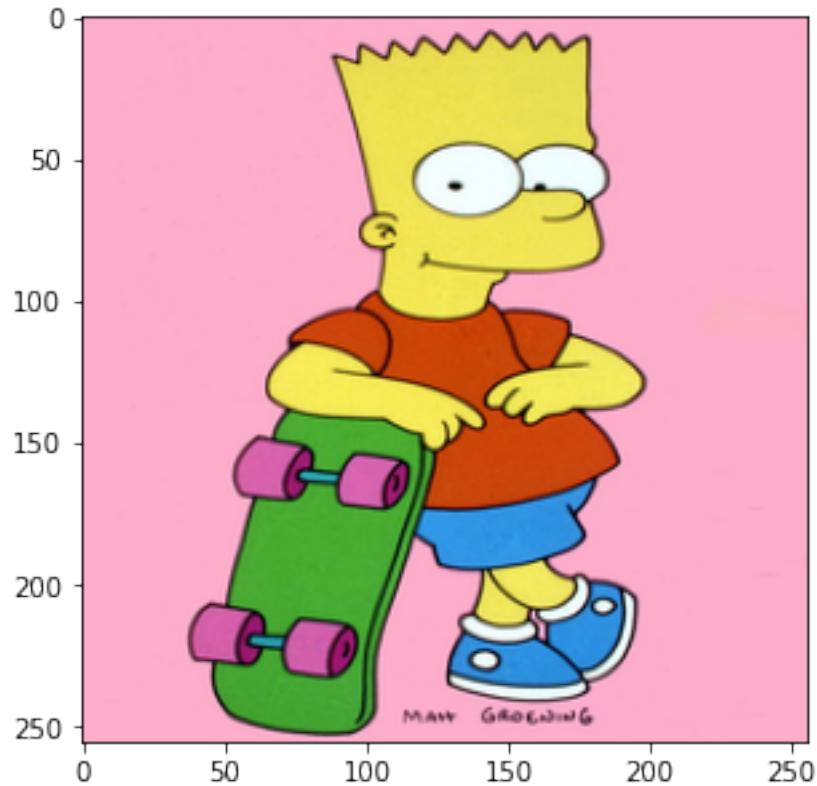
```
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC090>
```



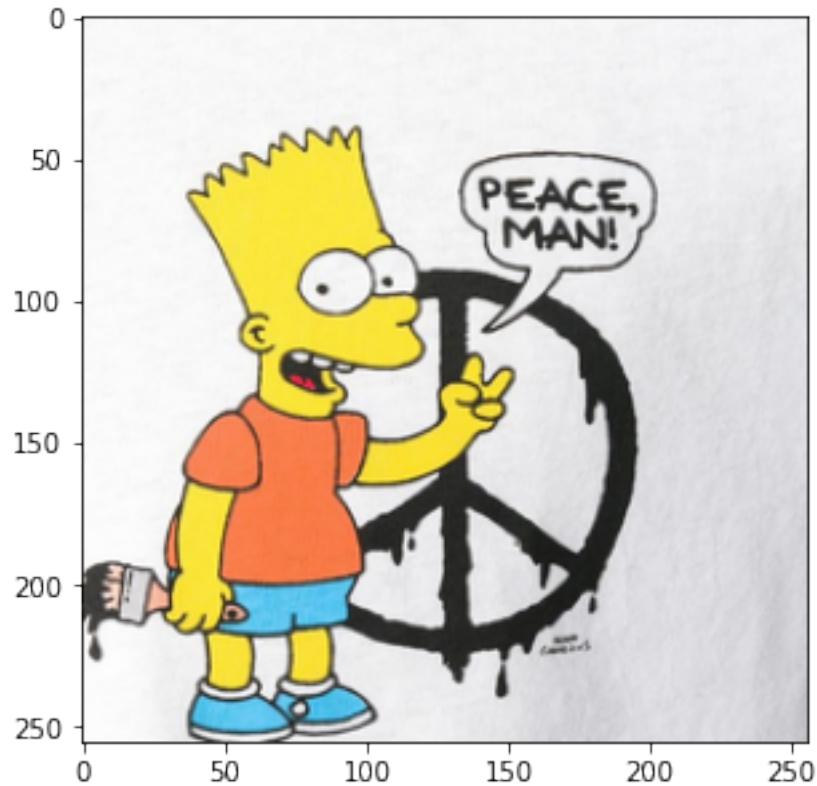
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC3D0>



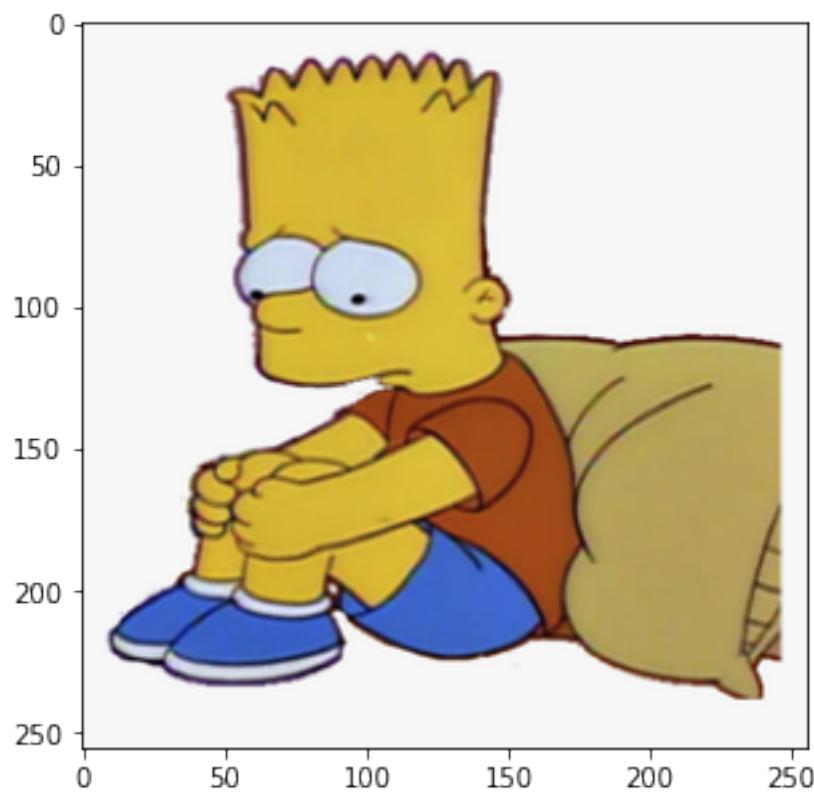
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC7D0>



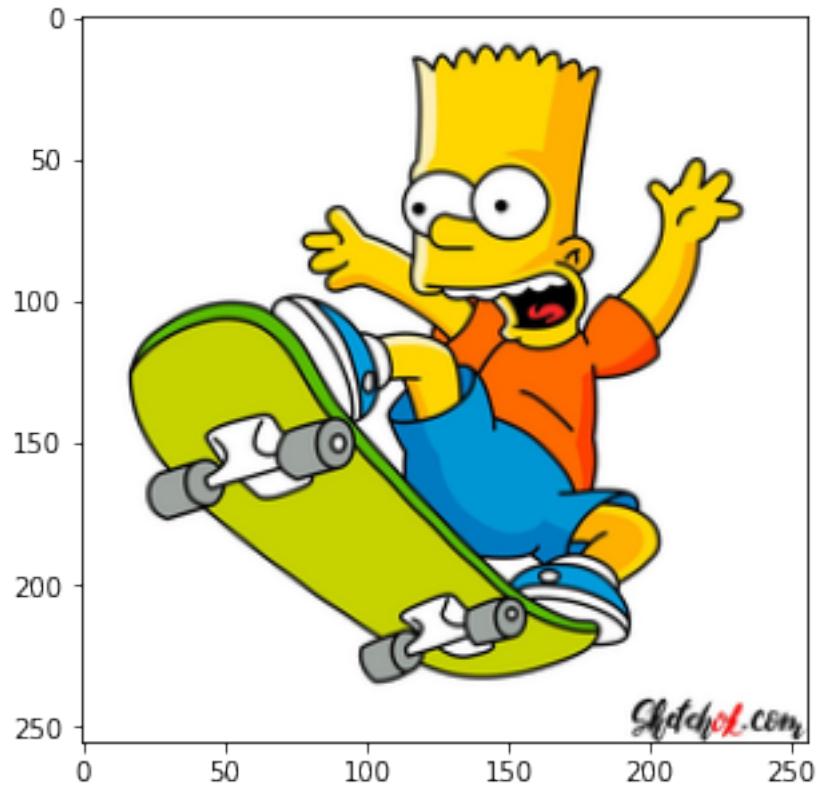
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC510>



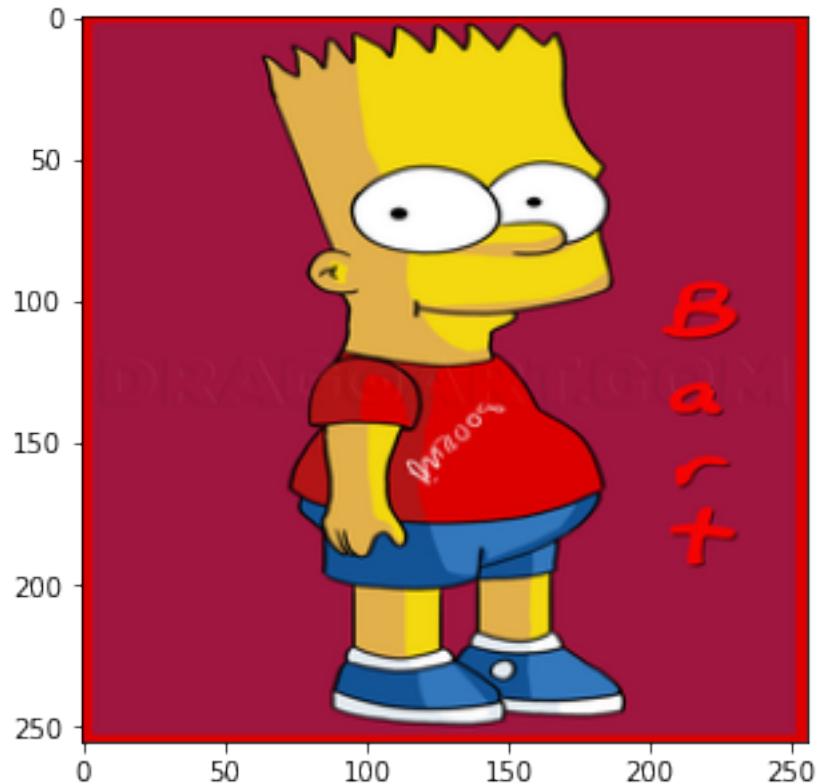
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119192110>



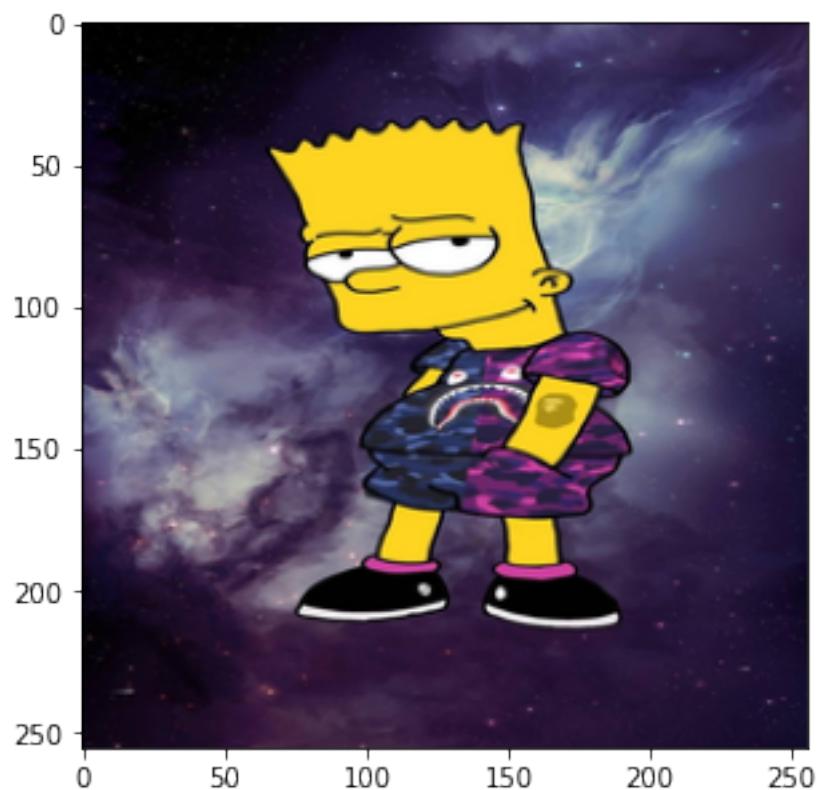
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11943F790>



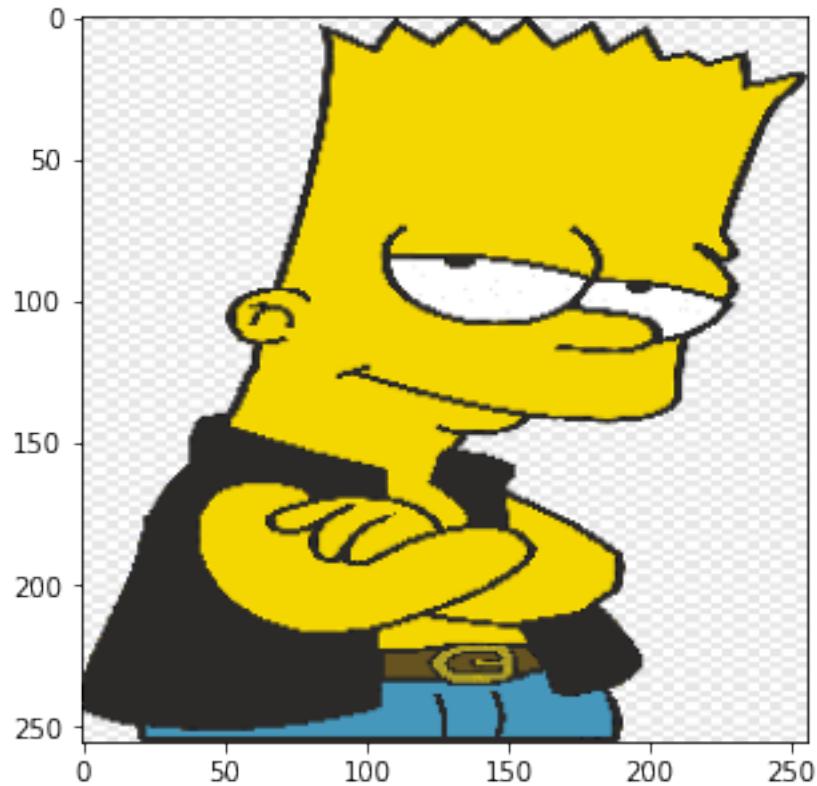
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11980D790>



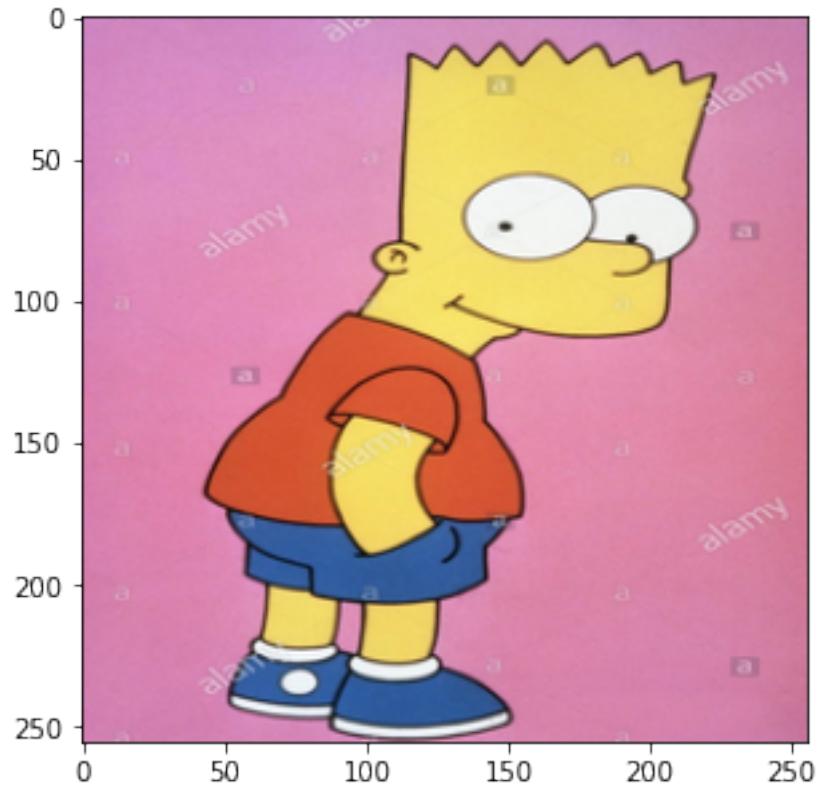
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192290>



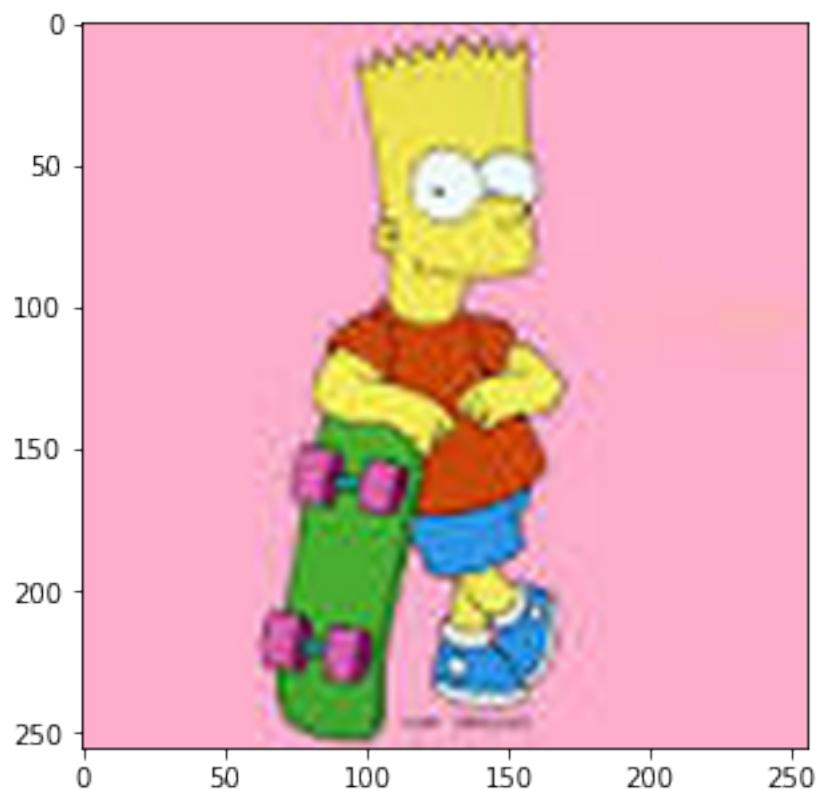
<PIL.Image.Image image mode=P size=256x256 at 0x7FB1191920D0>



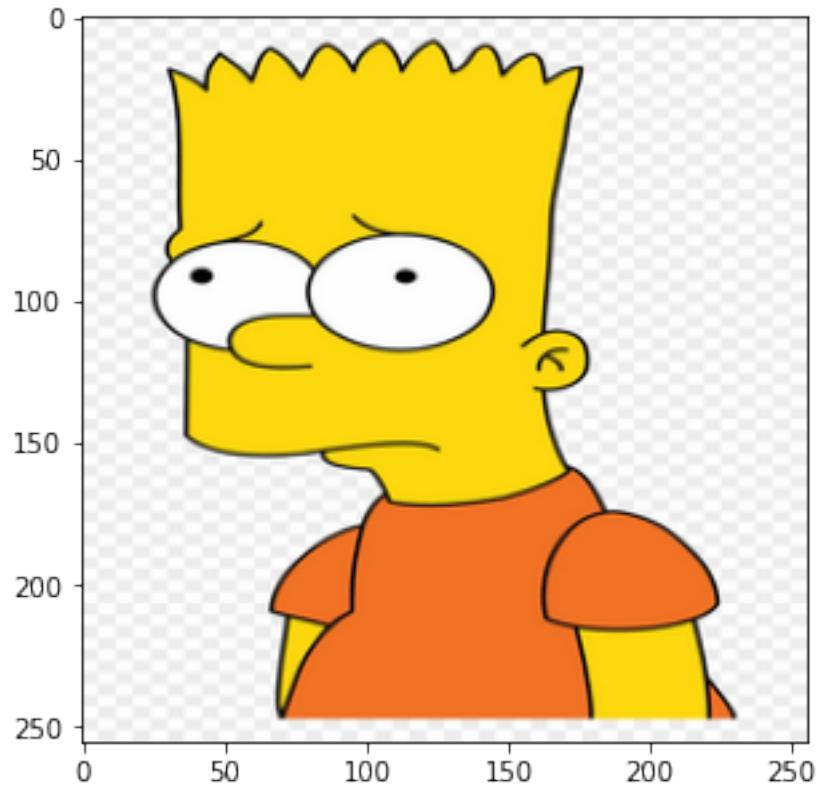
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192250>



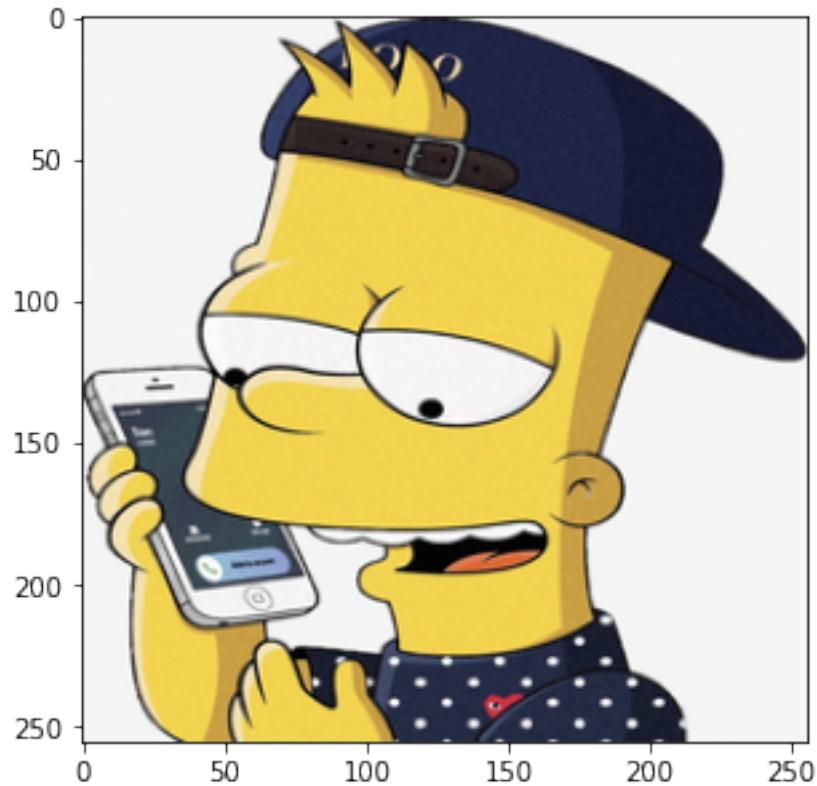
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192090>



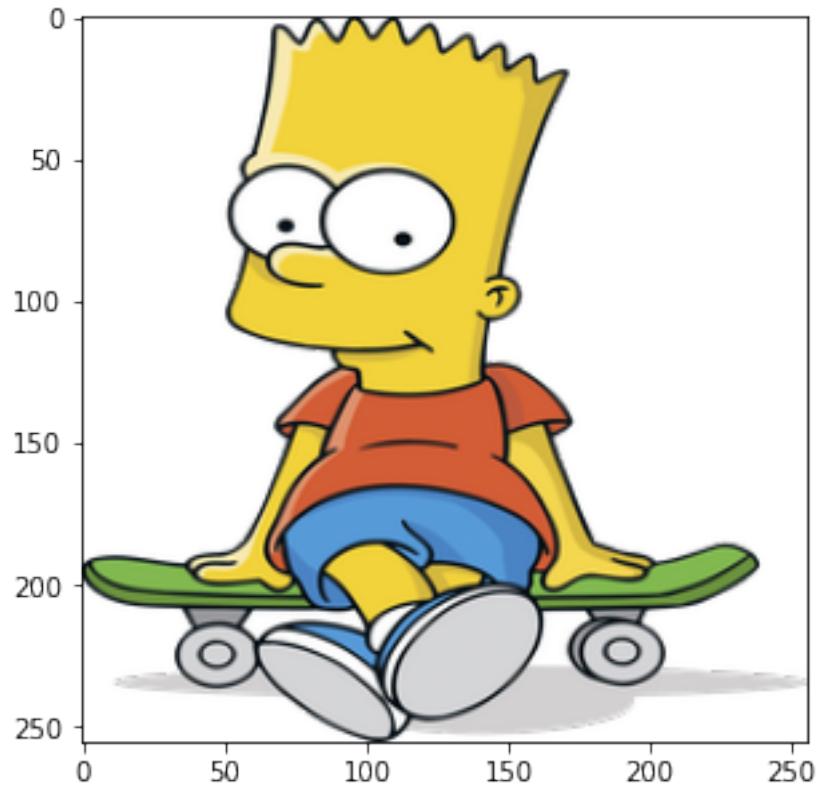
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192350>



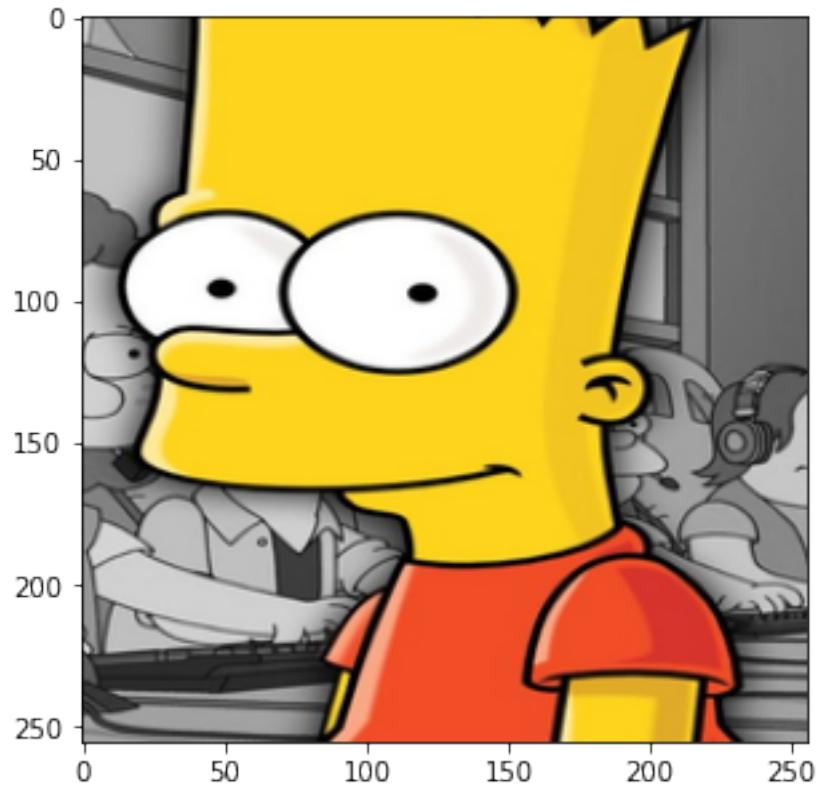
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192390>



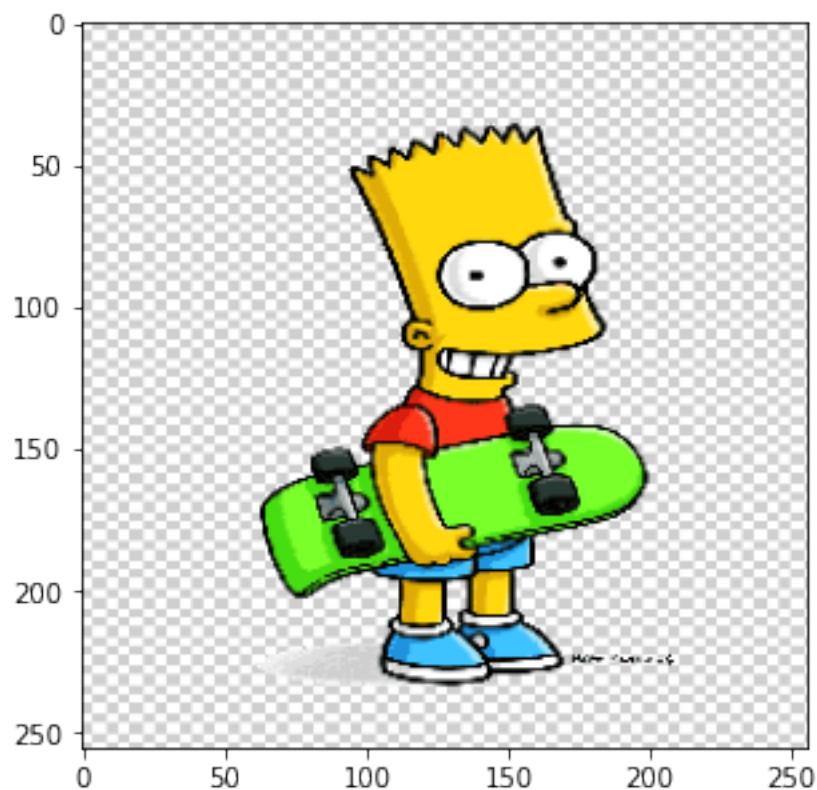
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119192490>



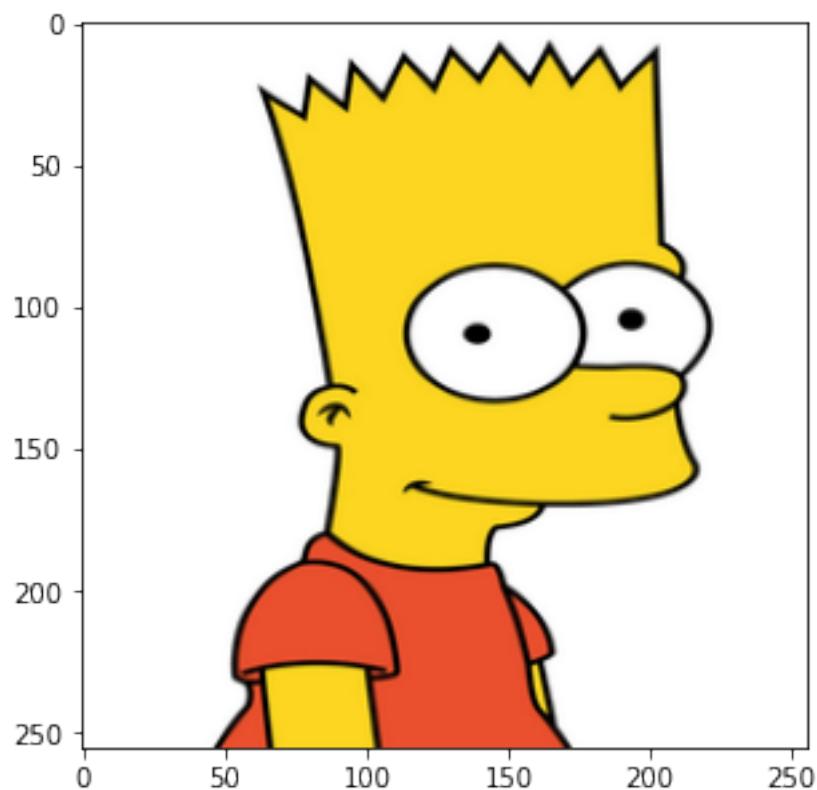
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192450>



<PIL.Image.Image image mode=P size=256x256 at 0x7FB1191924D0>



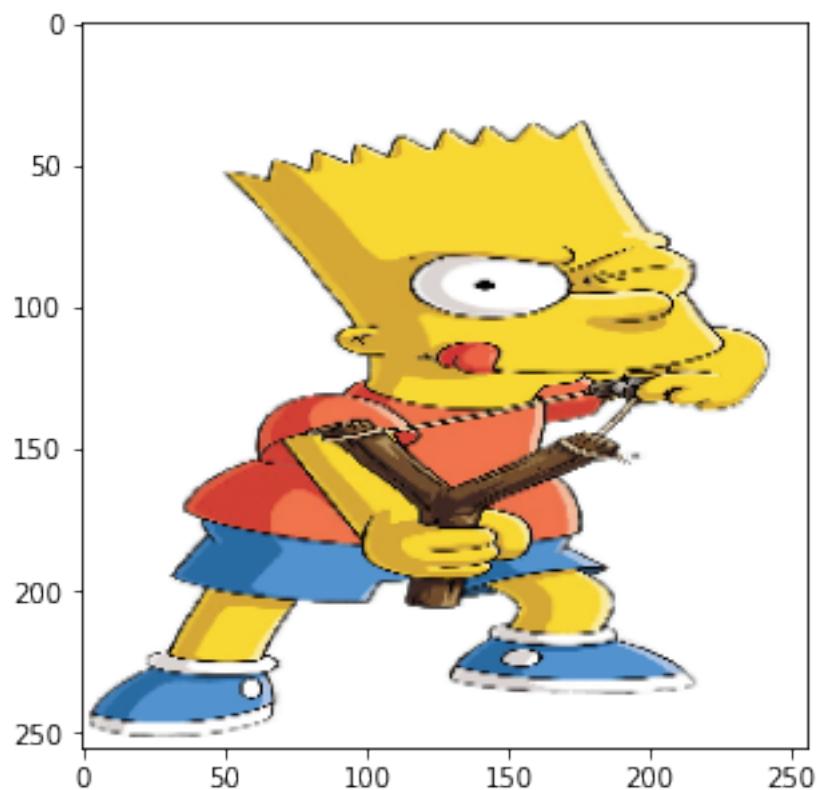
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192610>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192690>



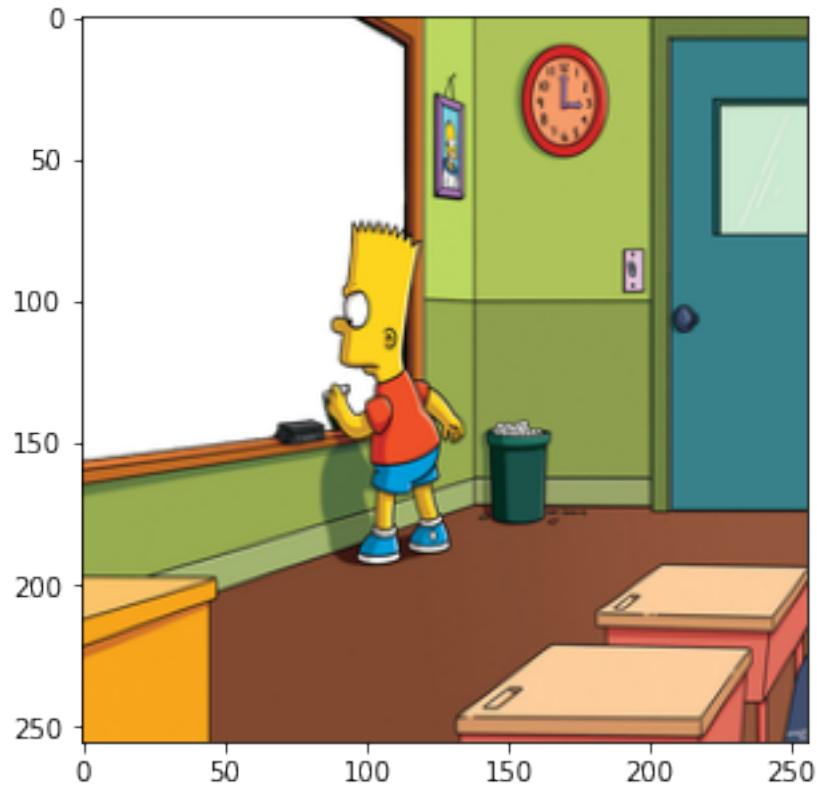
<PIL.Image.Image image mode=P size=256x256 at 0x7FB119192650>



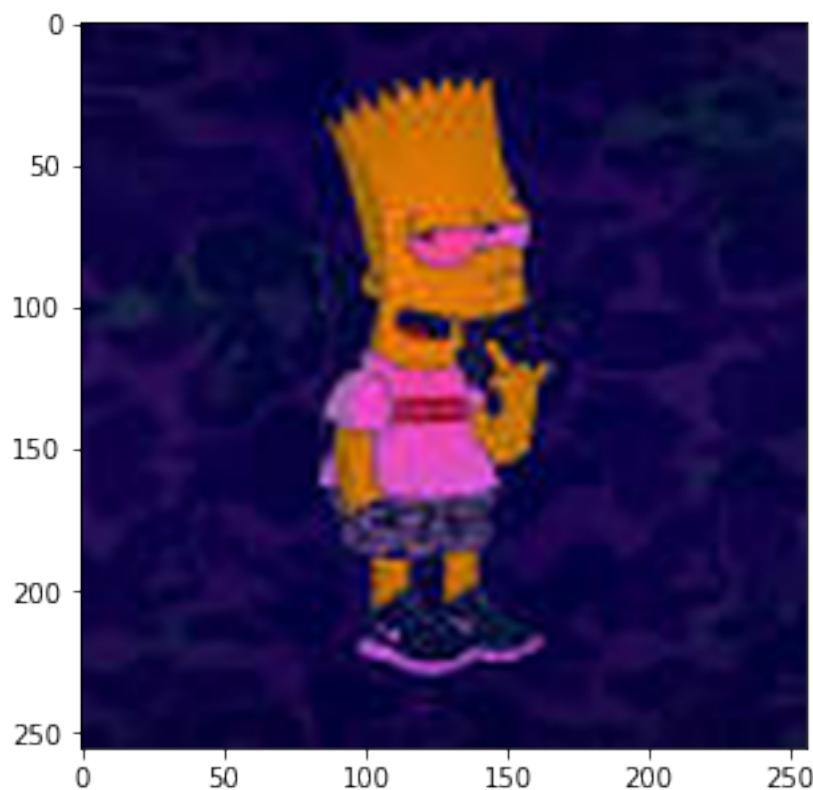
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192710>



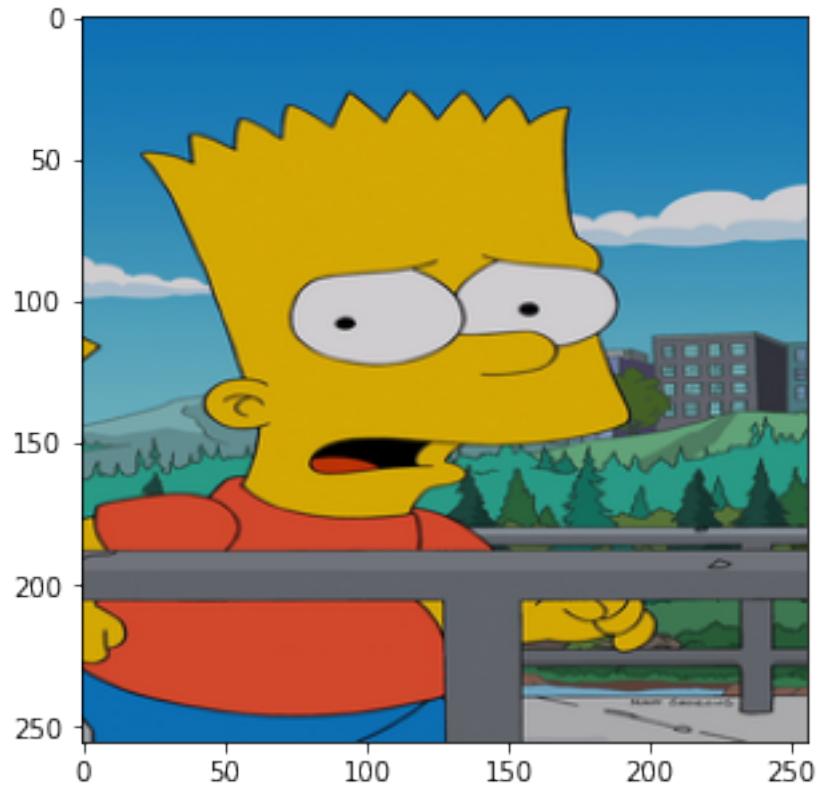
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191928D0>



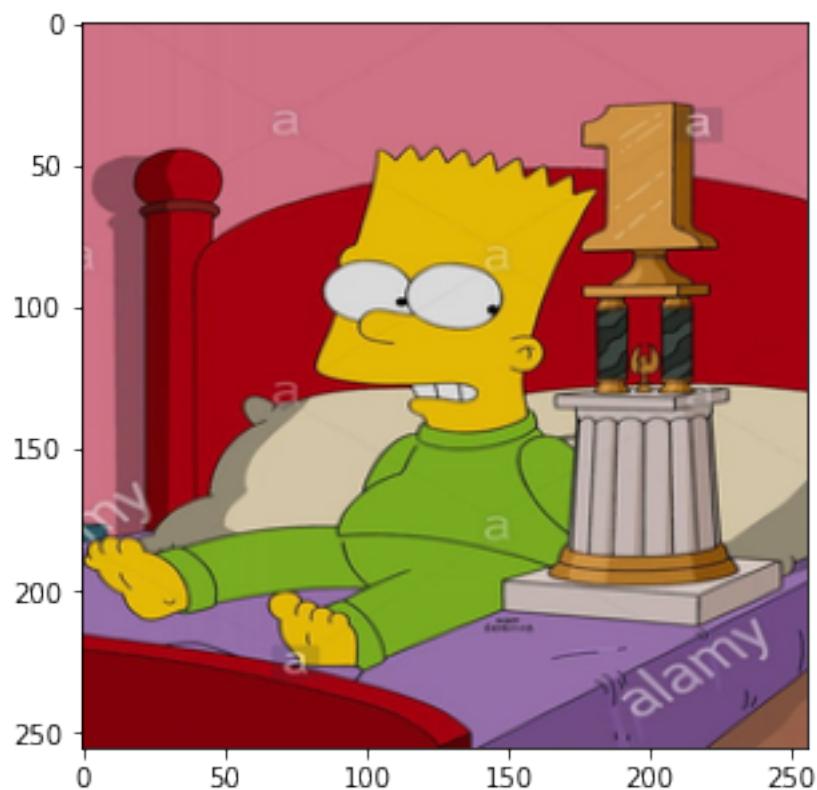
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192990>



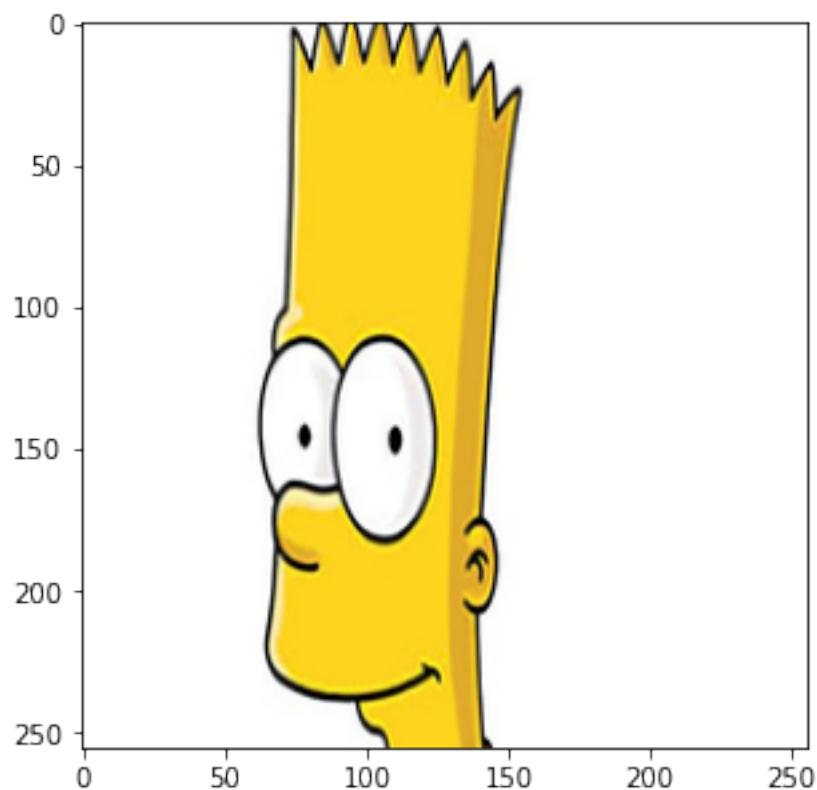
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191929D0>



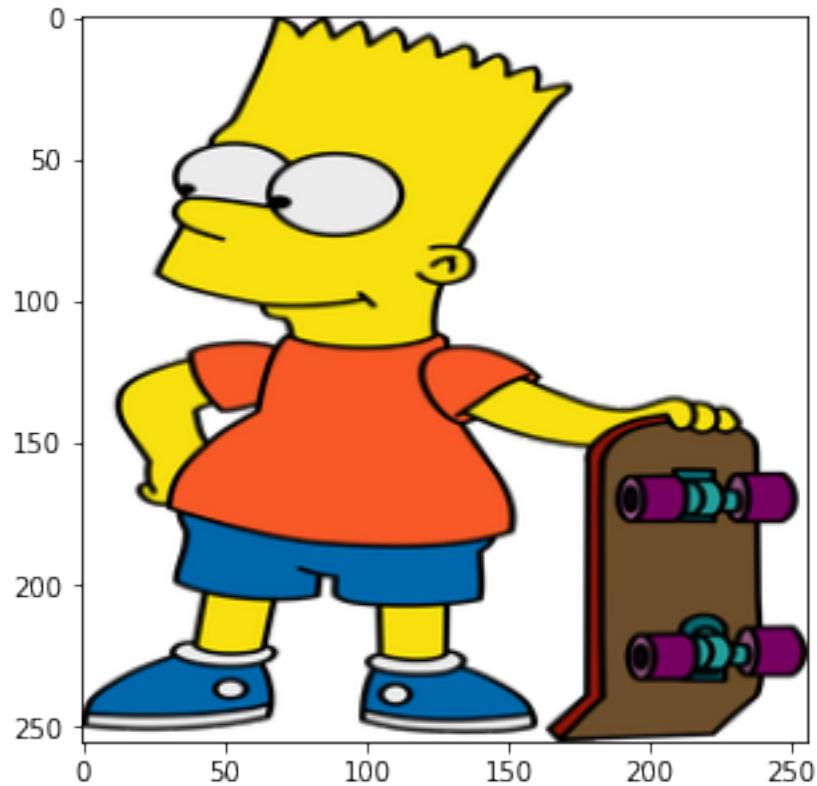
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192A10>



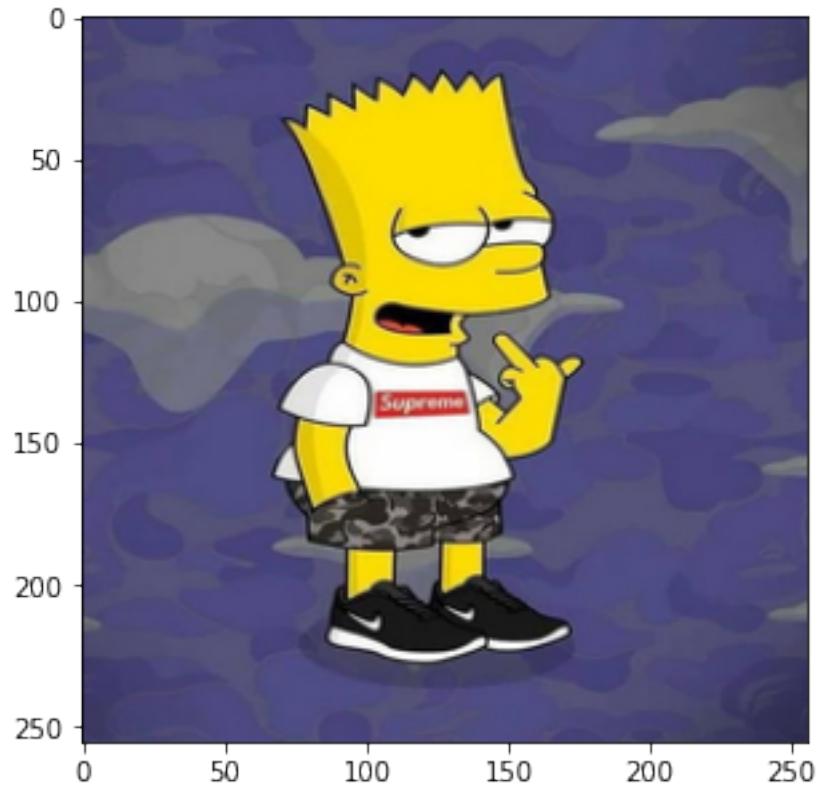
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192B10>



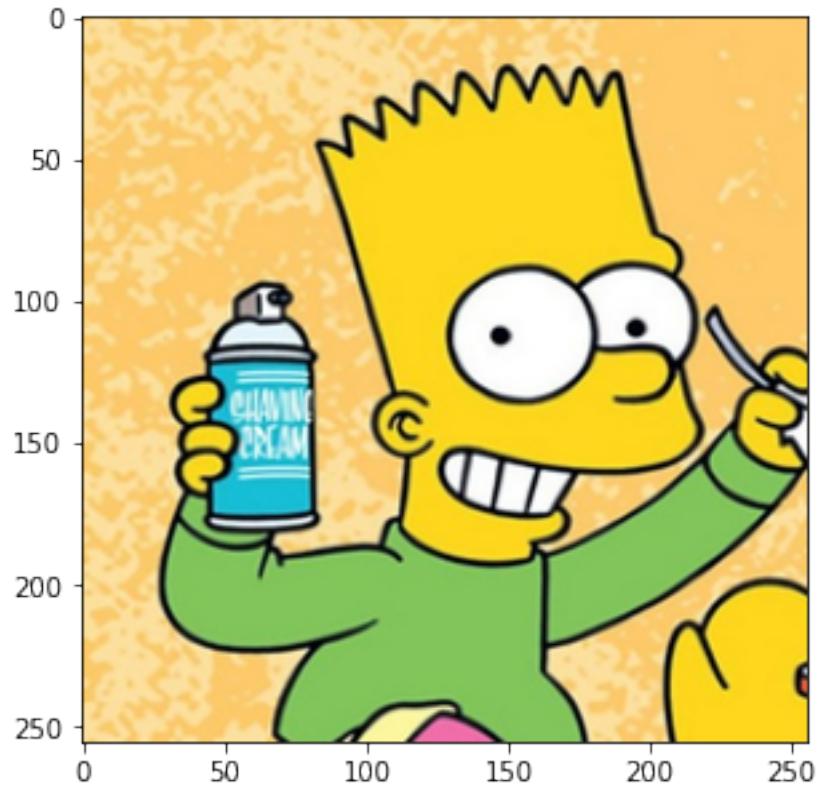
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119192C90>



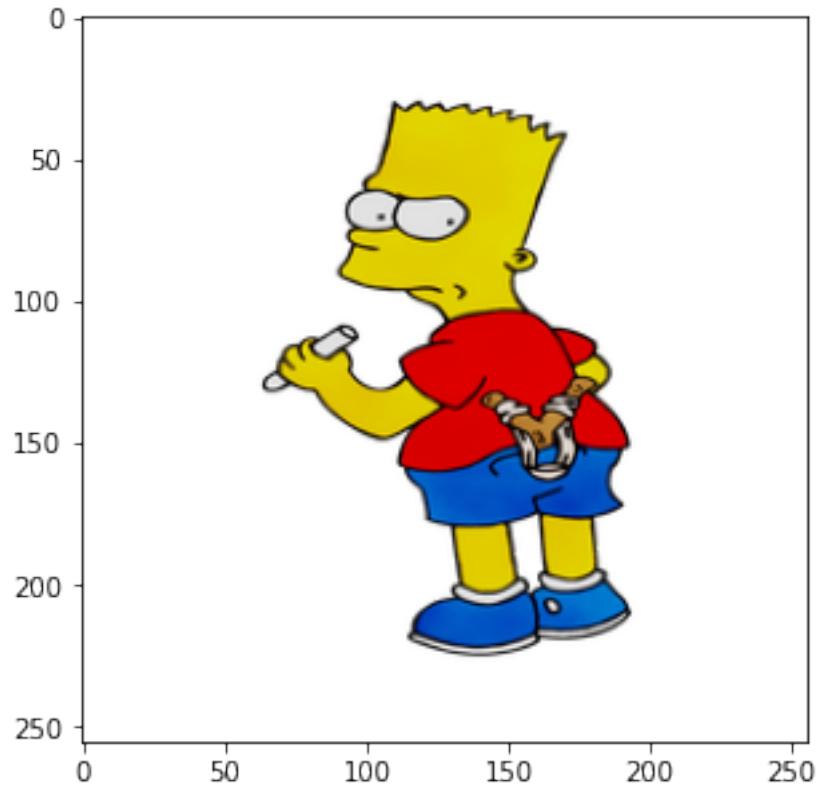
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192C10>



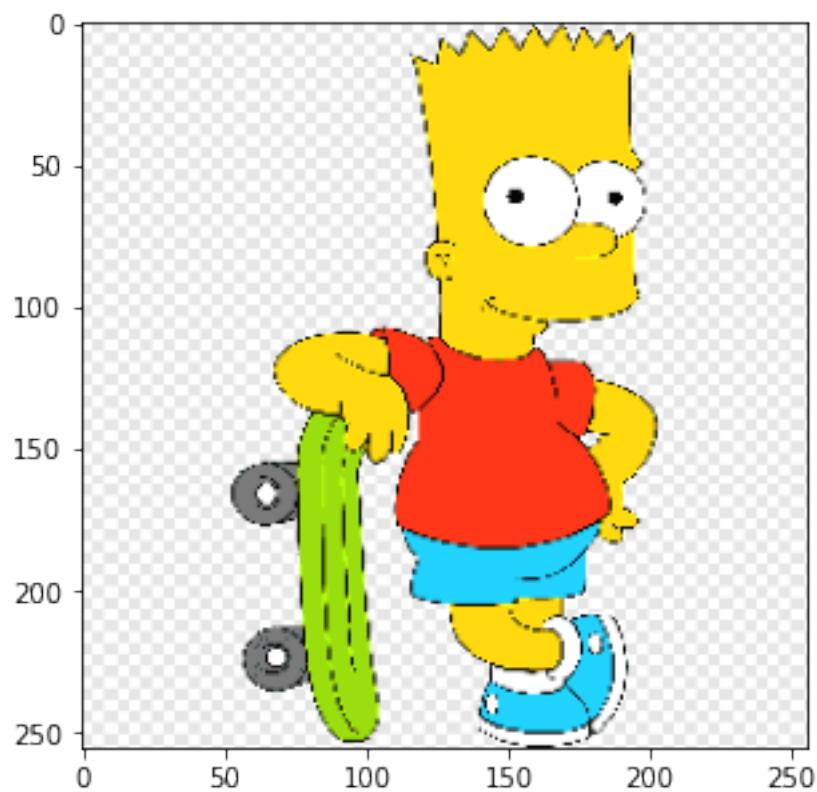
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192CD0>



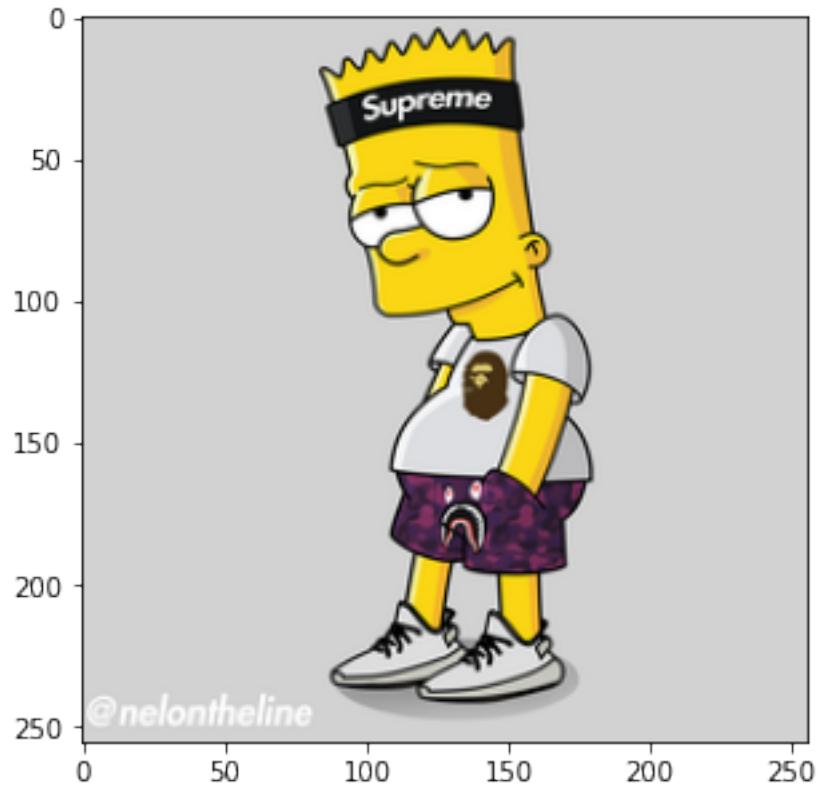
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119192DD0>



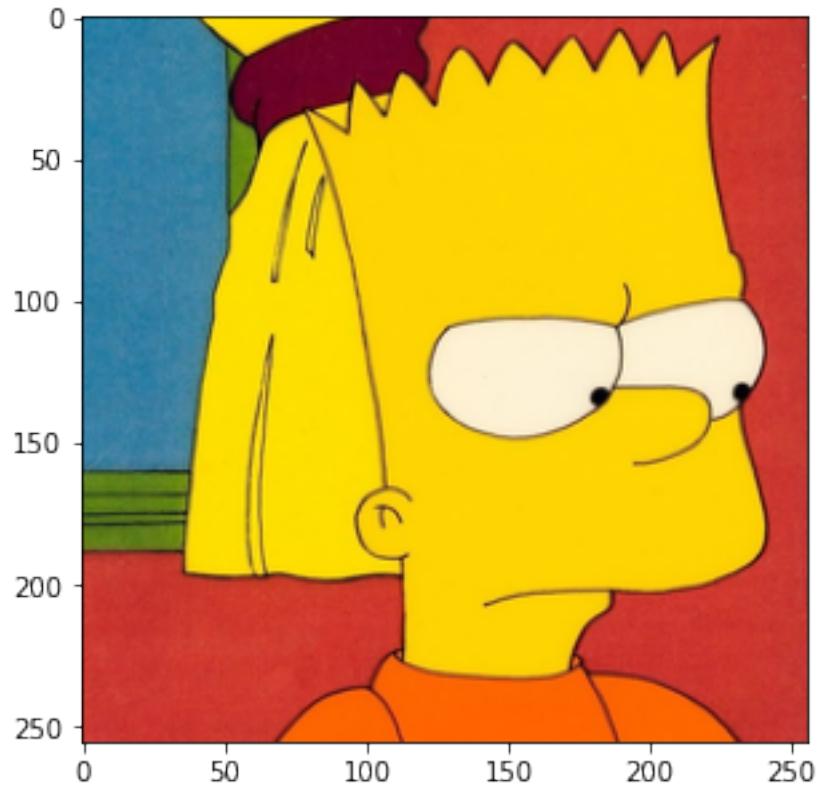
```
<PIL.Image.Image image mode=P size=256x256 at 0x7FB119192D10>
```



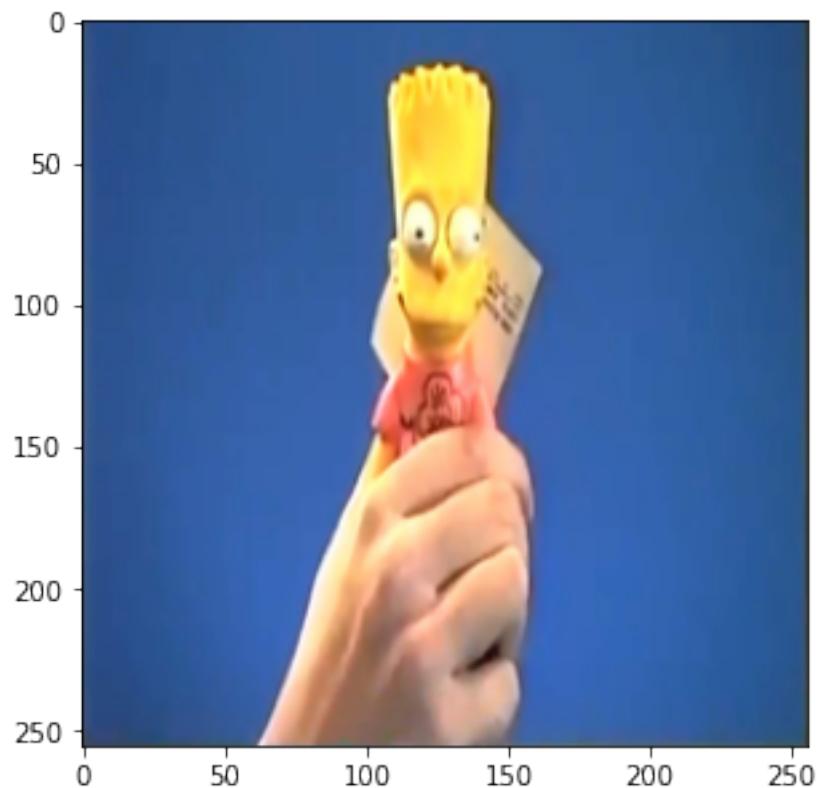
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192ED0>



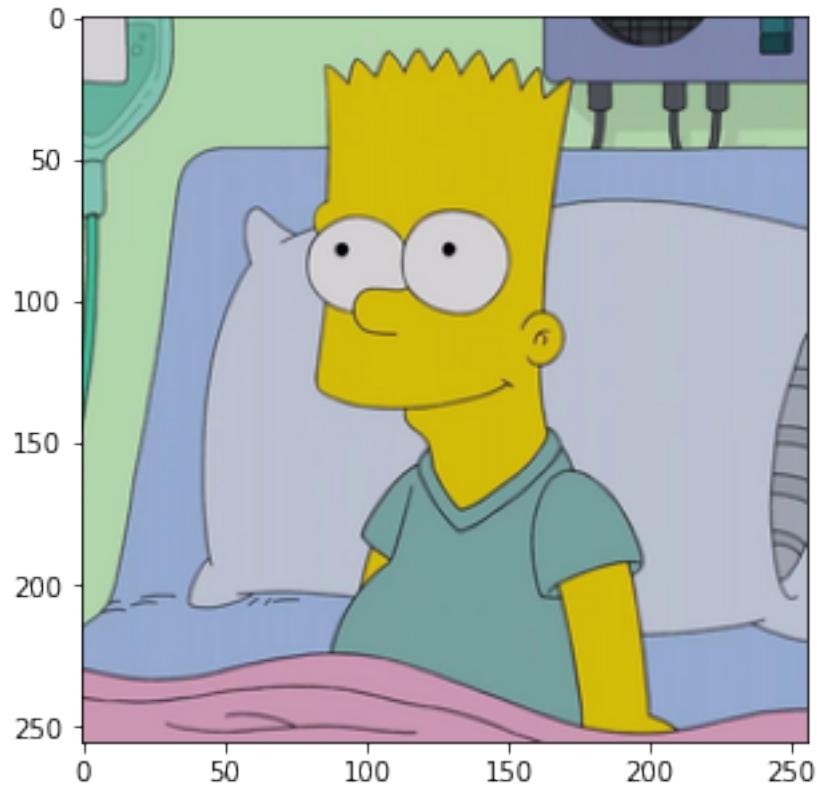
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192E10>



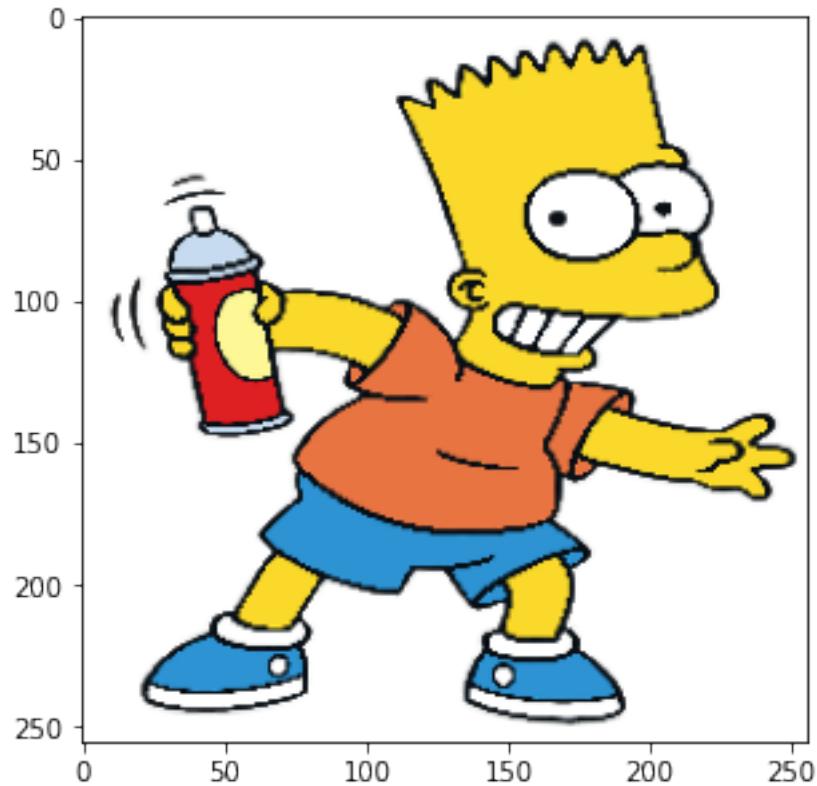
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192FD0>



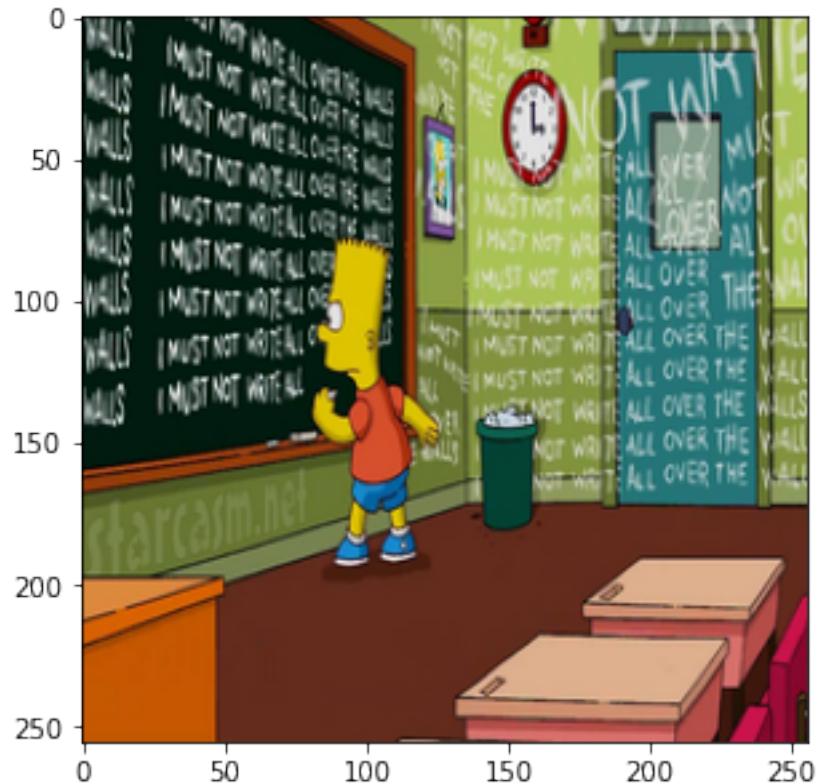
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192A50>



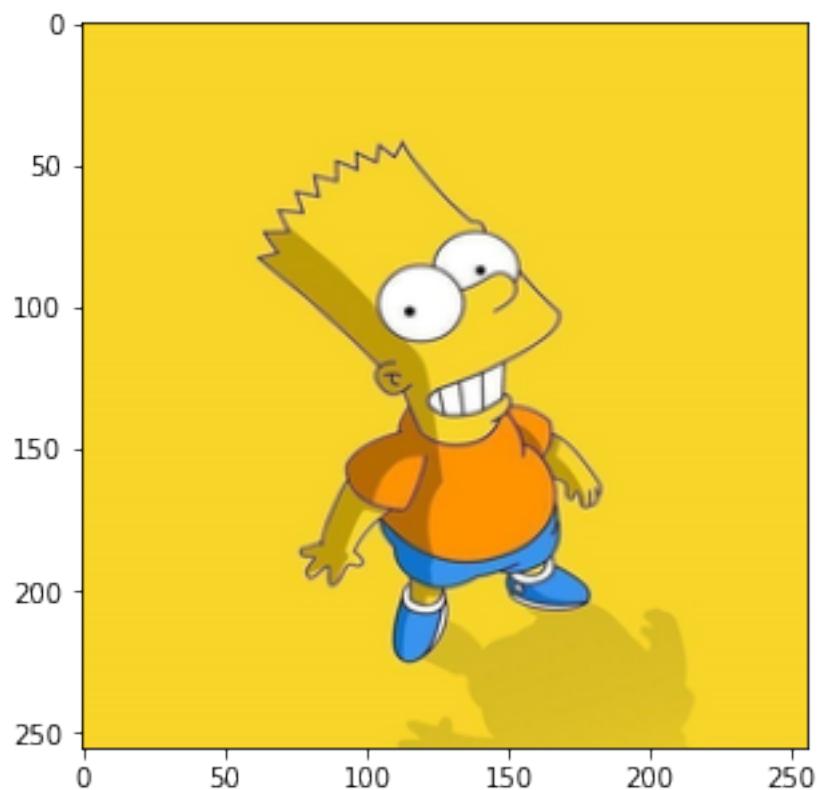
<PIL.Image.Image image mode=P size=256x256 at 0x7FB119192850>



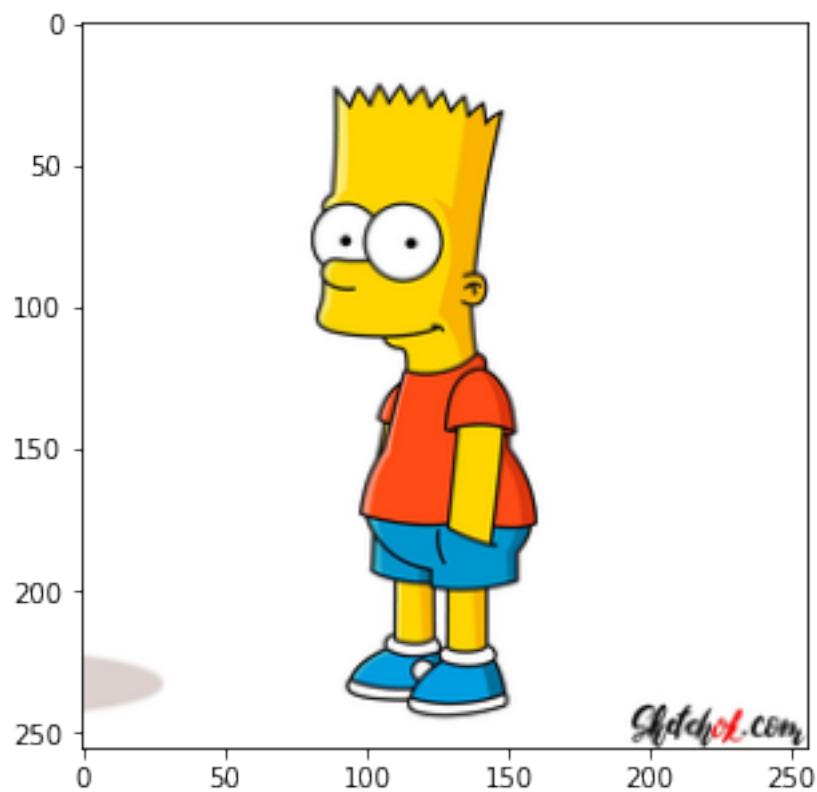
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479C50>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1194794D0>

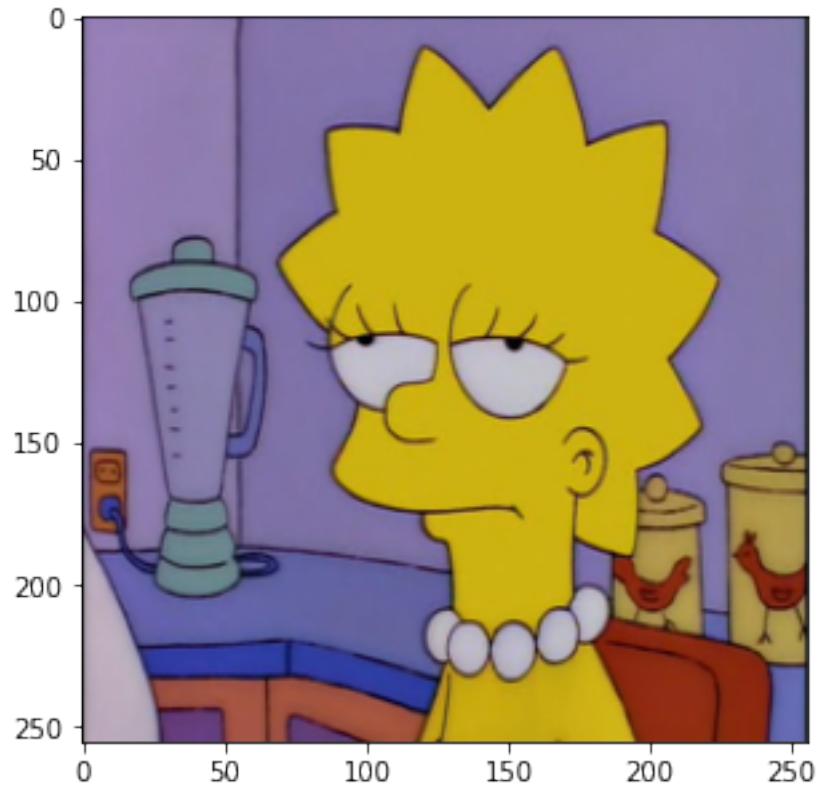


<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479550>

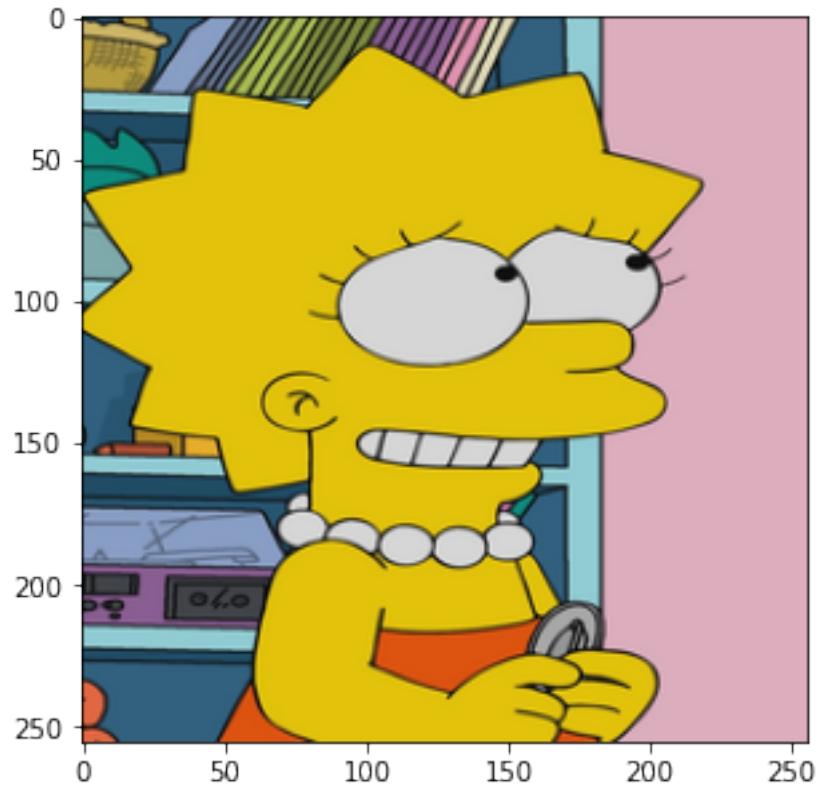


Lisa -- Resized

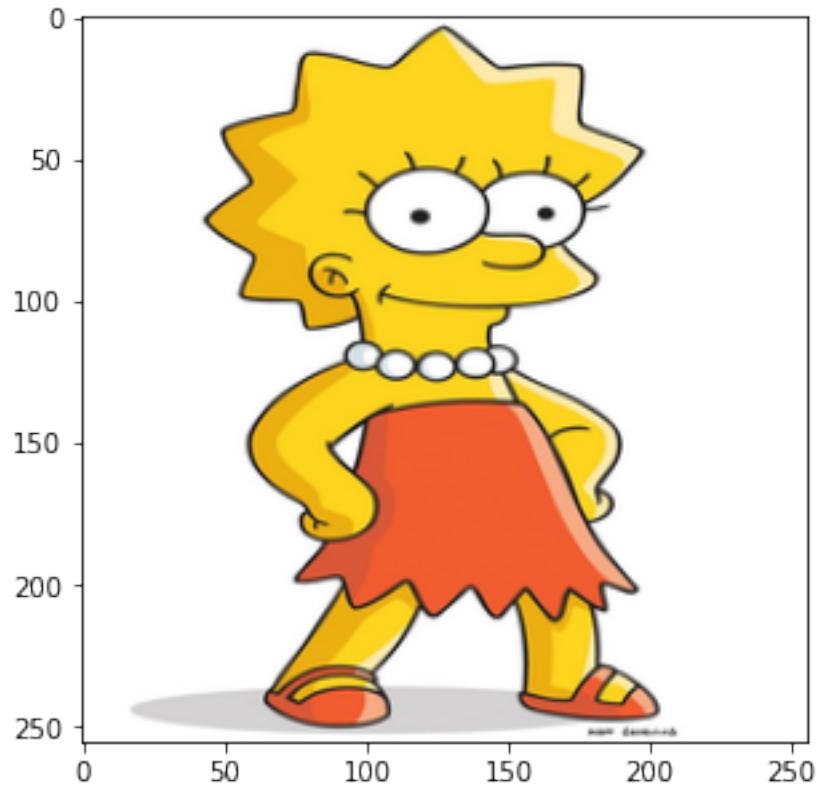
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119C06F90>



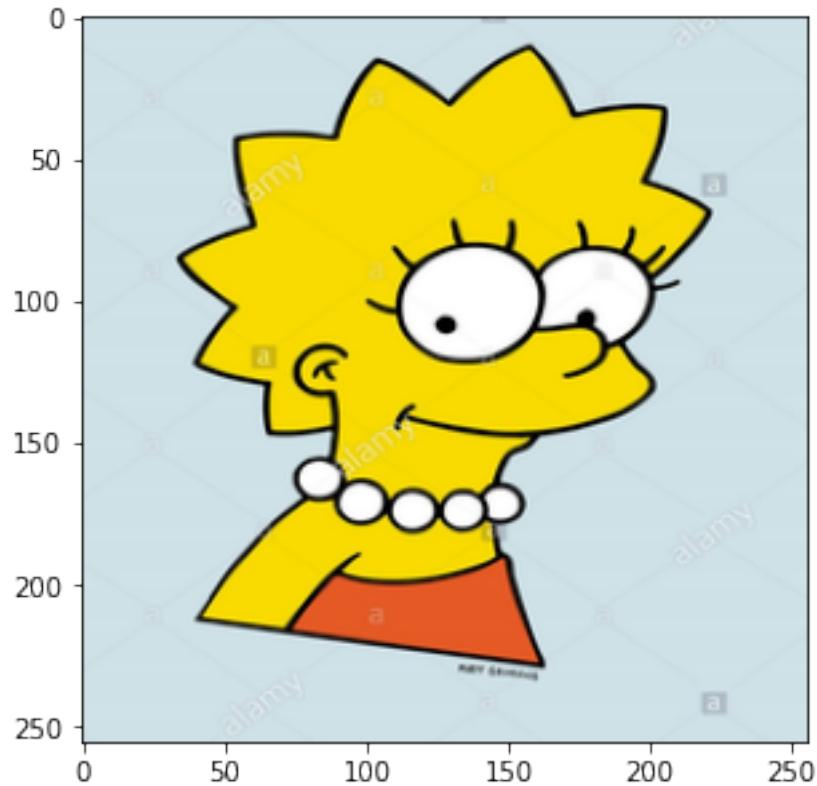
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119B87090>



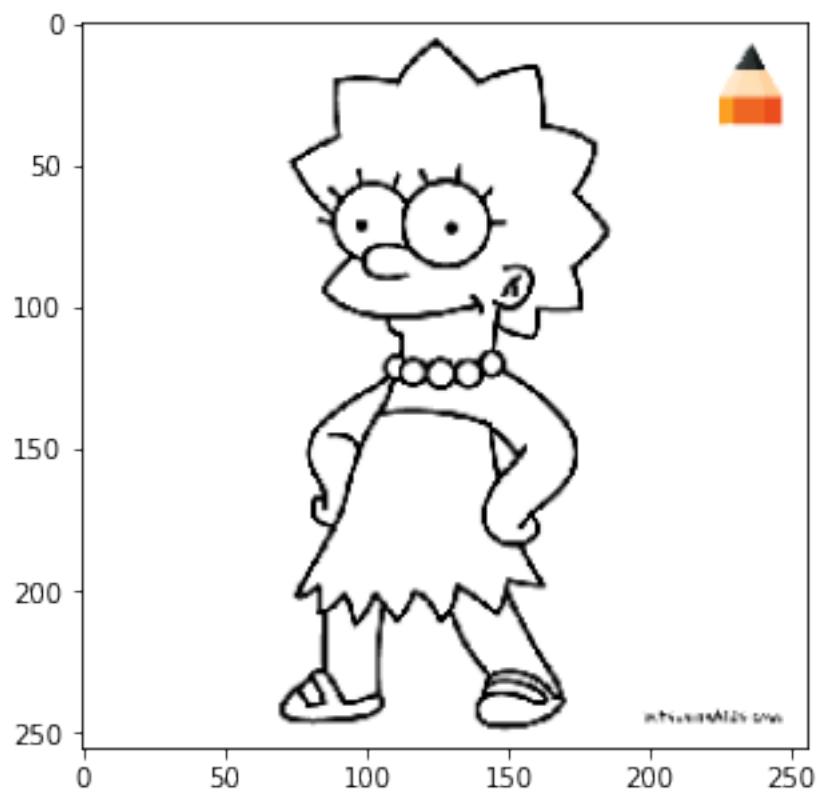
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119B87C10>



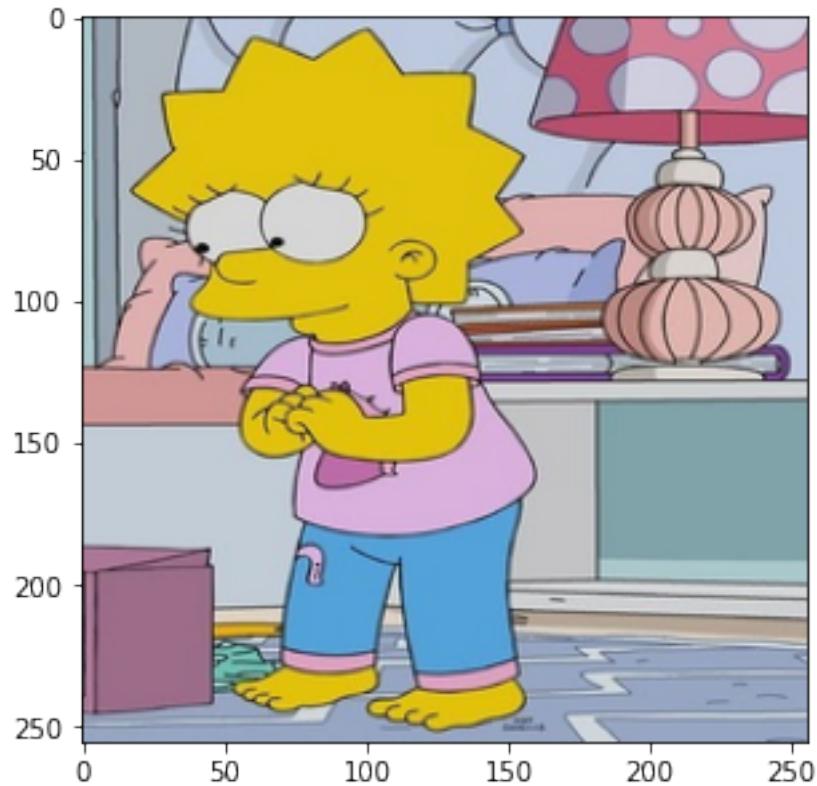
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119B87F10>



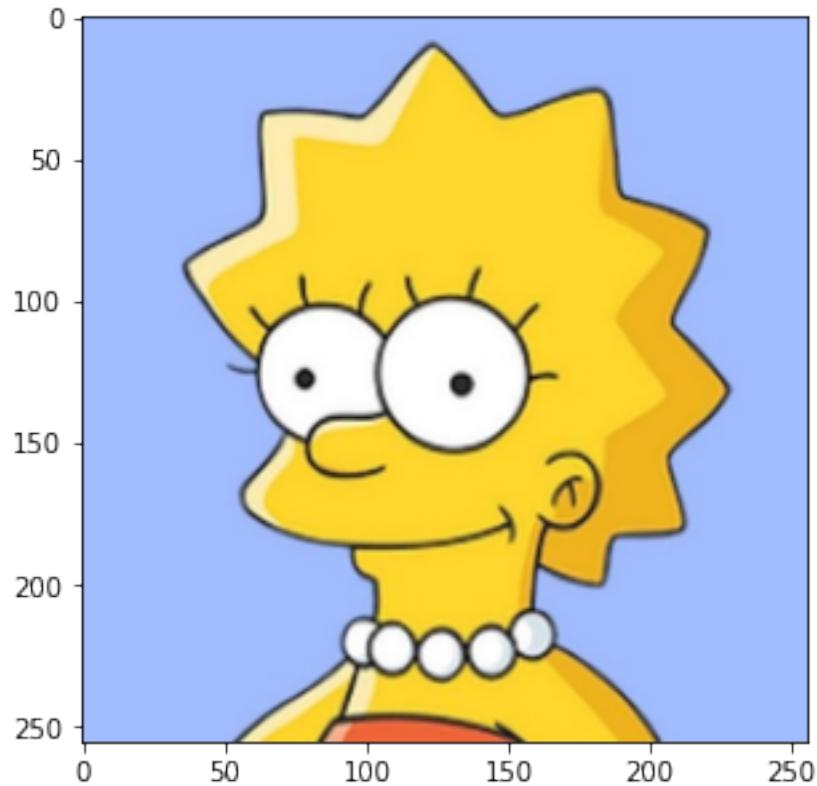
<PIL.Image.Image image mode=P size=256x256 at 0x7FB119B87C90>



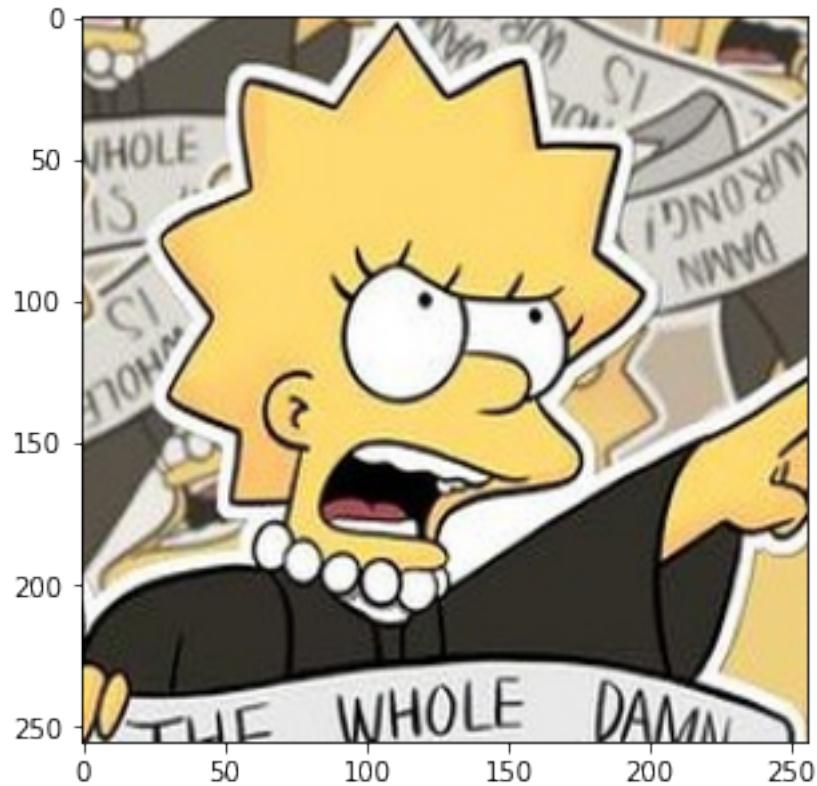
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119B87BD0>



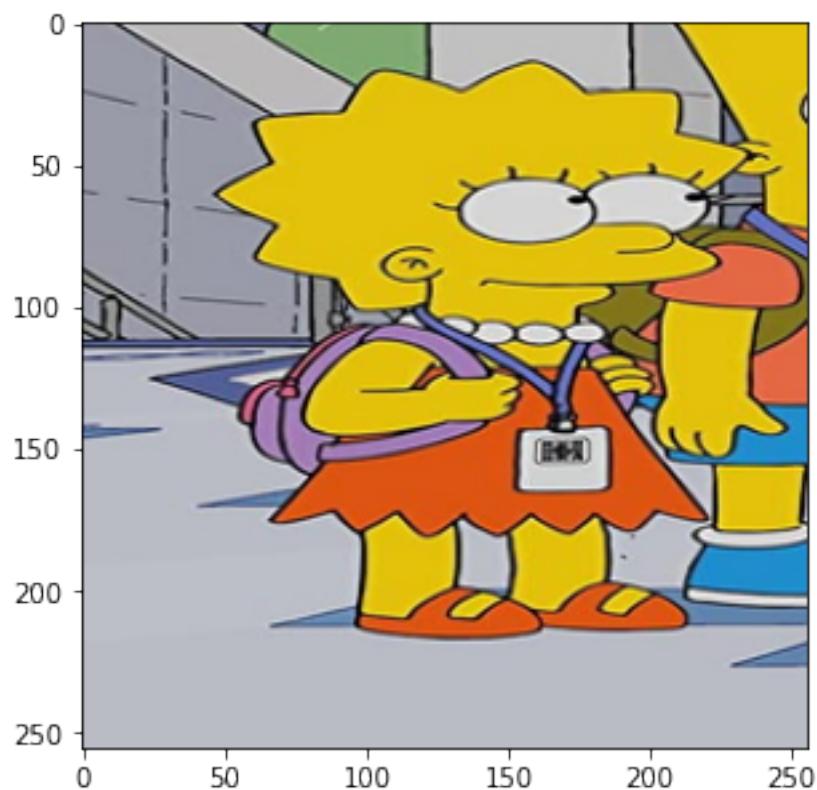
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192590>



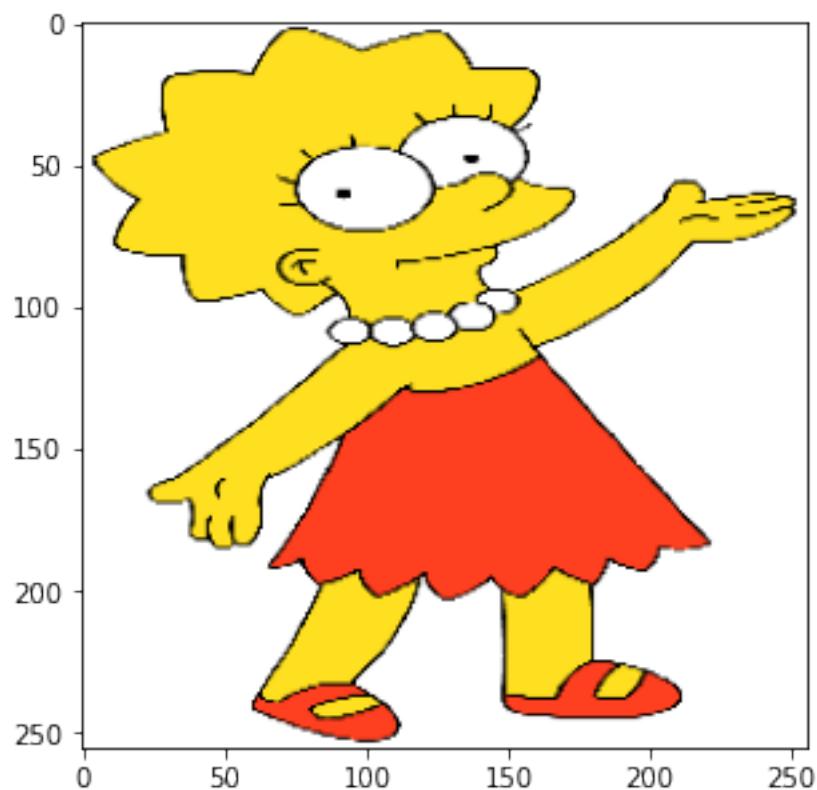
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119192910>



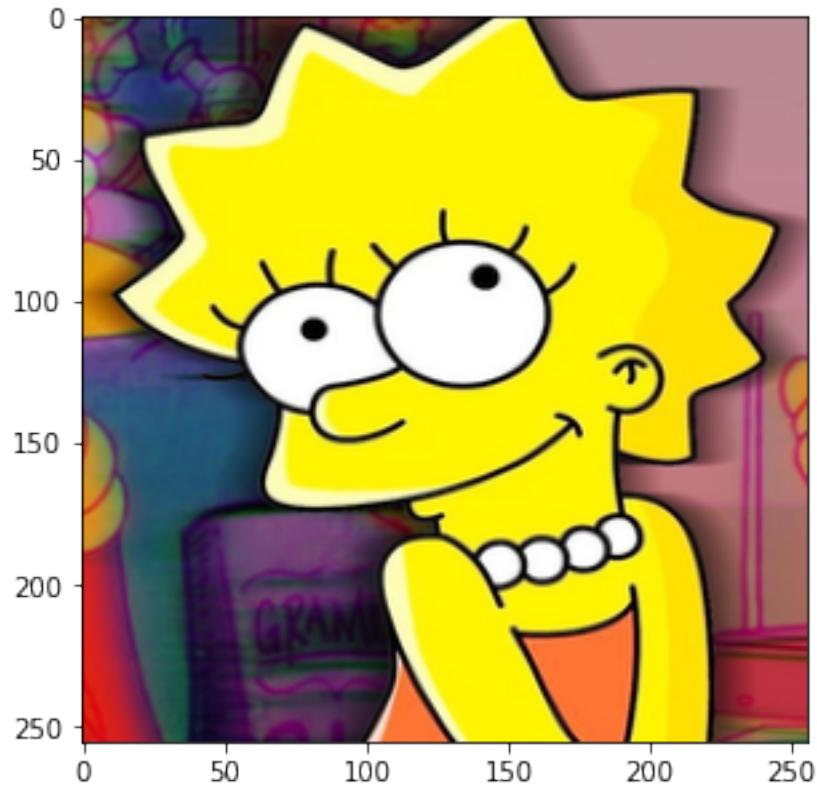
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119B87490>



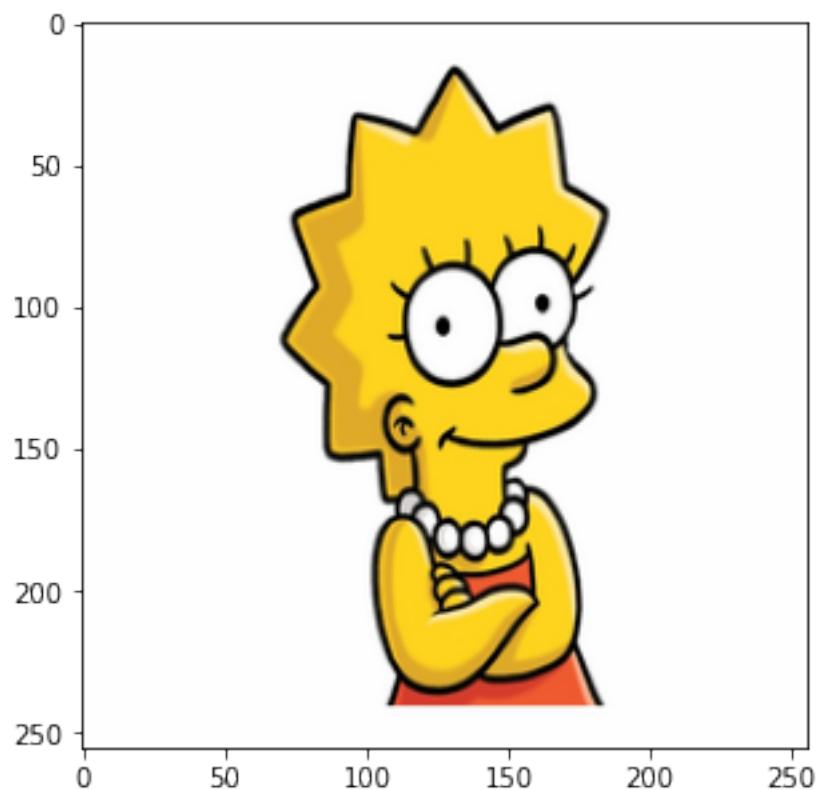
<PIL.Image.Image image mode=P size=256x256 at 0x7FB119B87290>



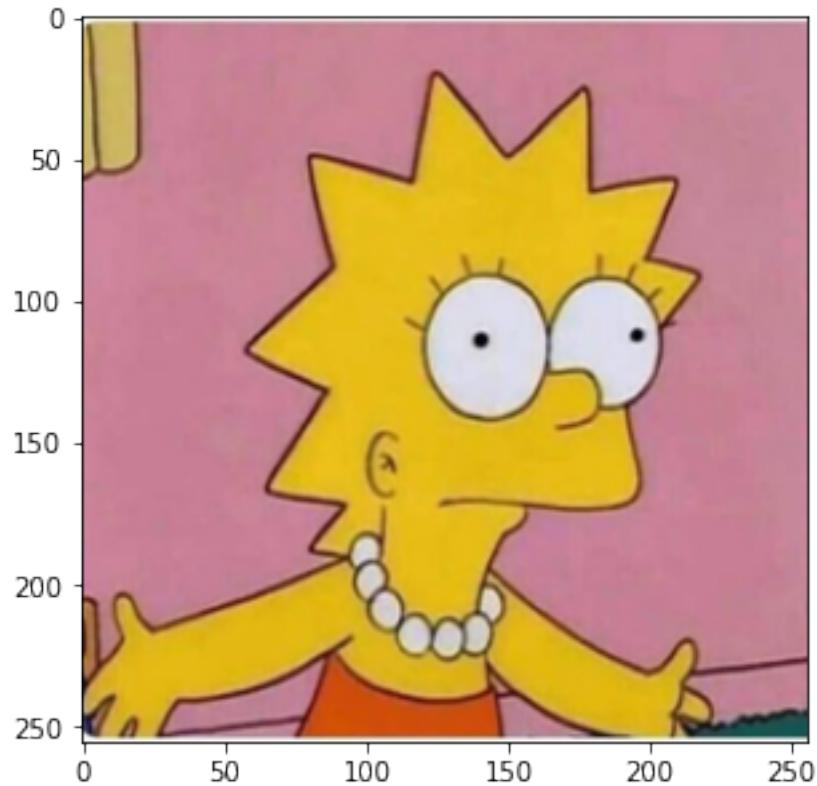
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119460B50>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119449550>



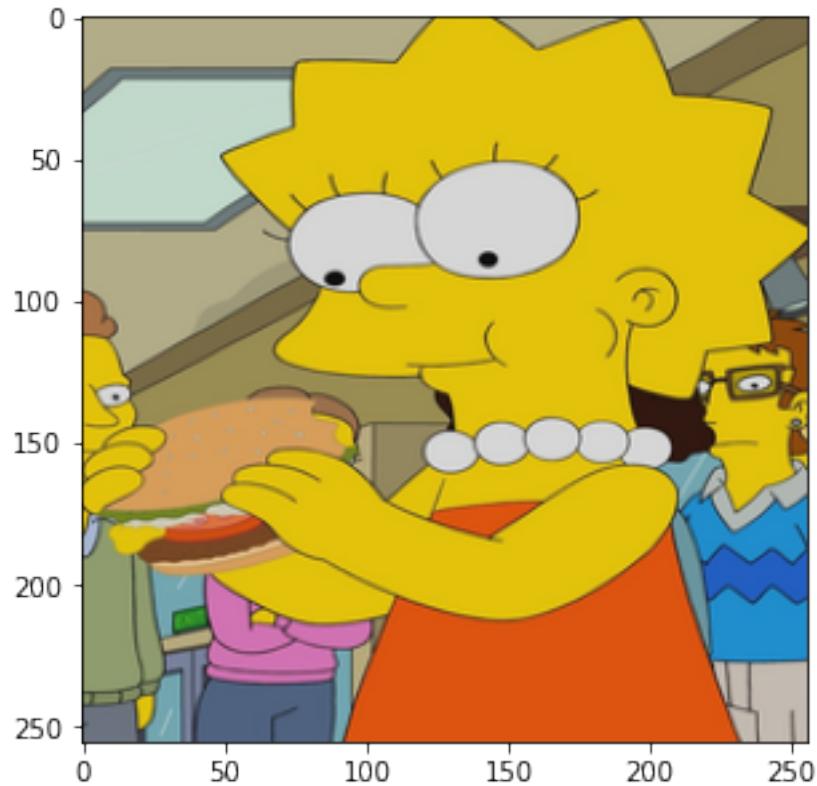
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119449B50>



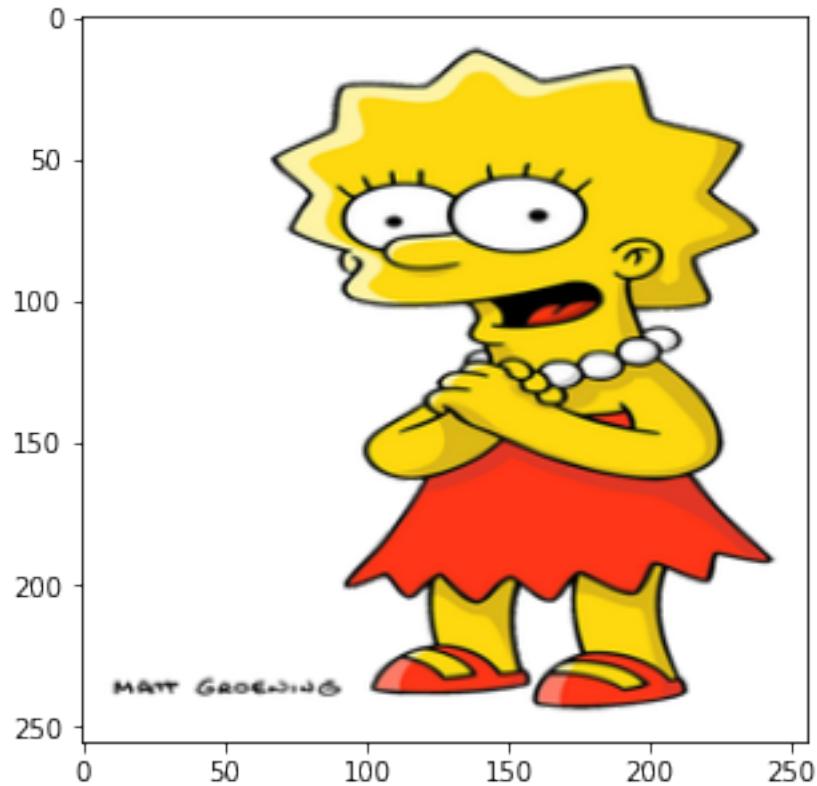
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119449350>



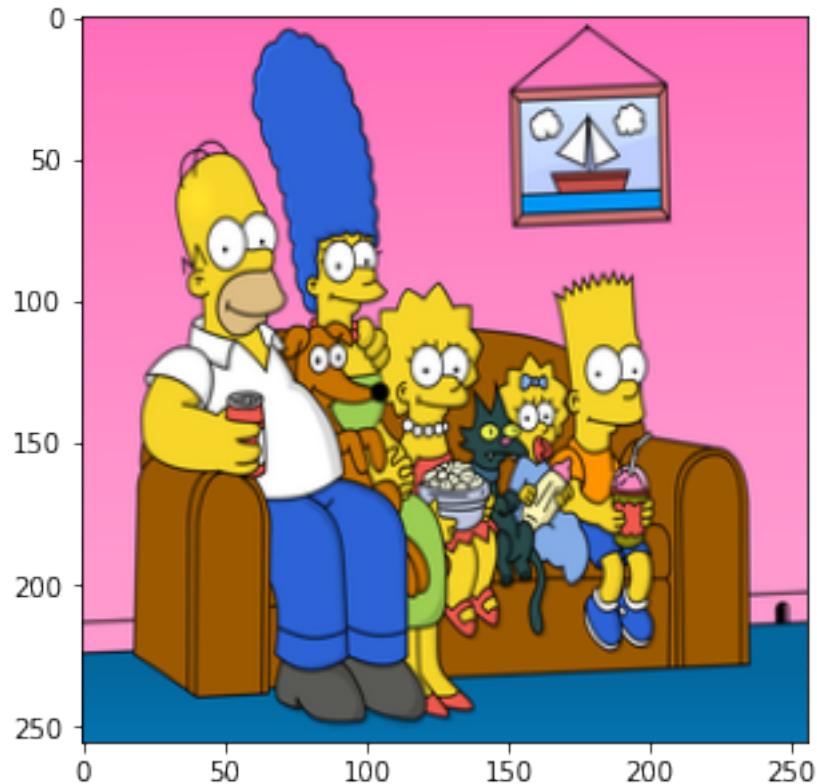
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119449790>



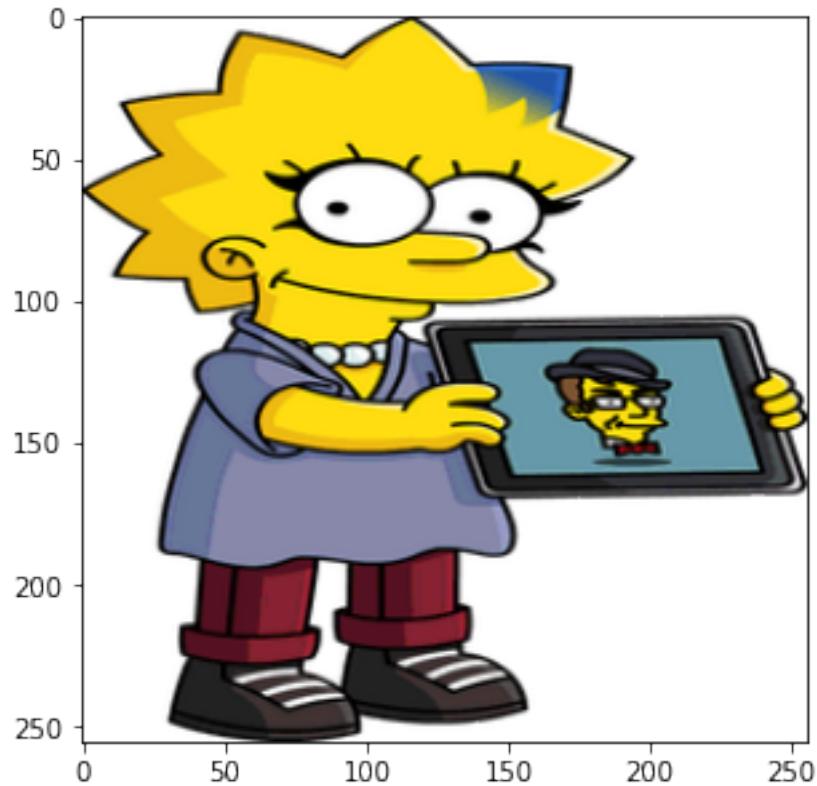
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1195FC2D0>



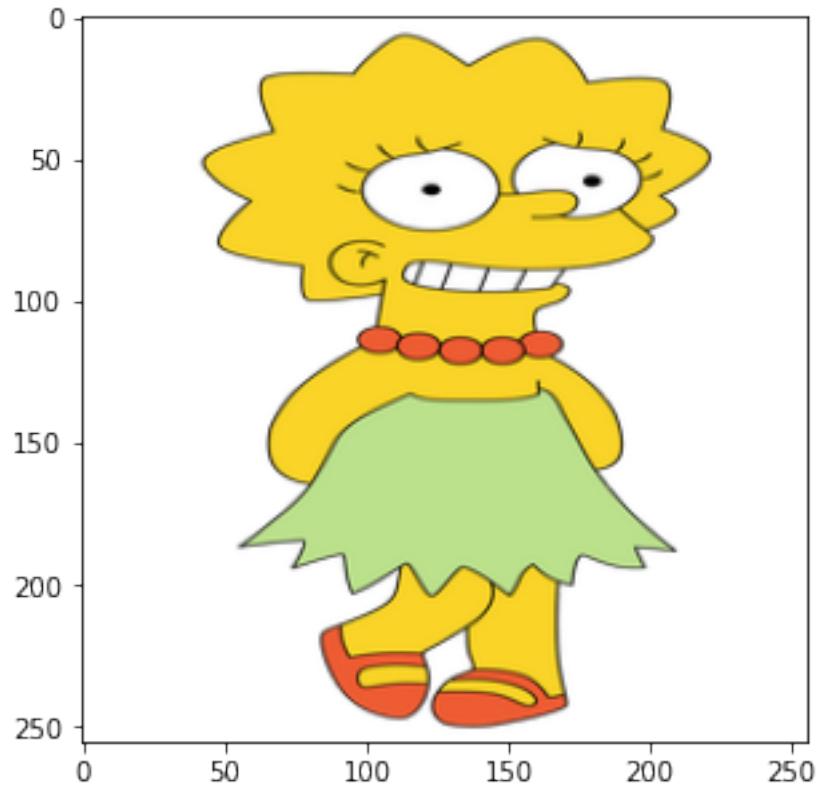
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119449ED0>



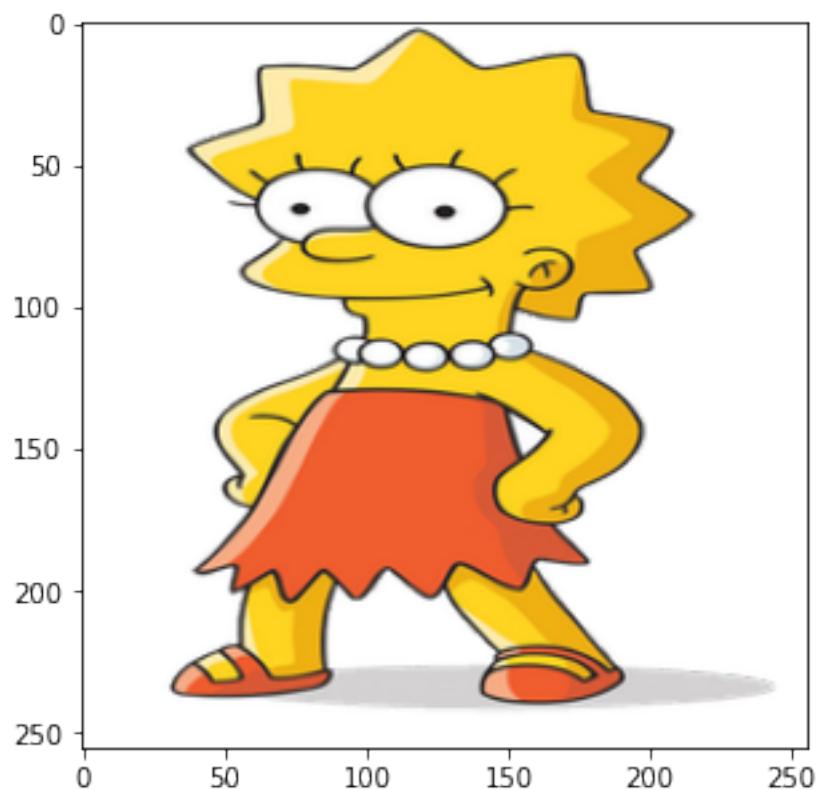
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1195FC5D0>



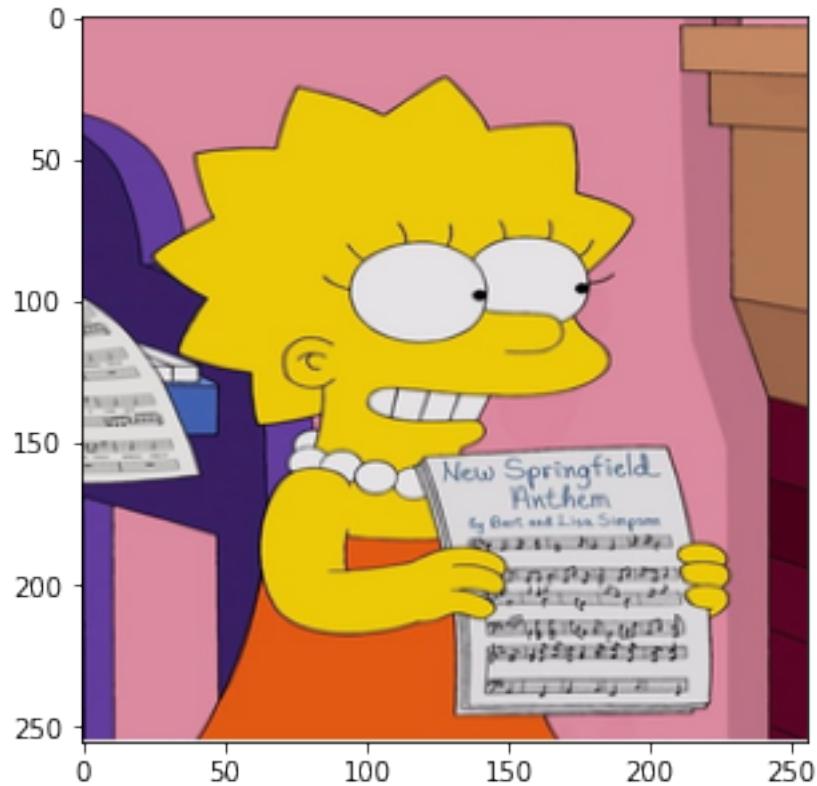
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1195FC4D0>



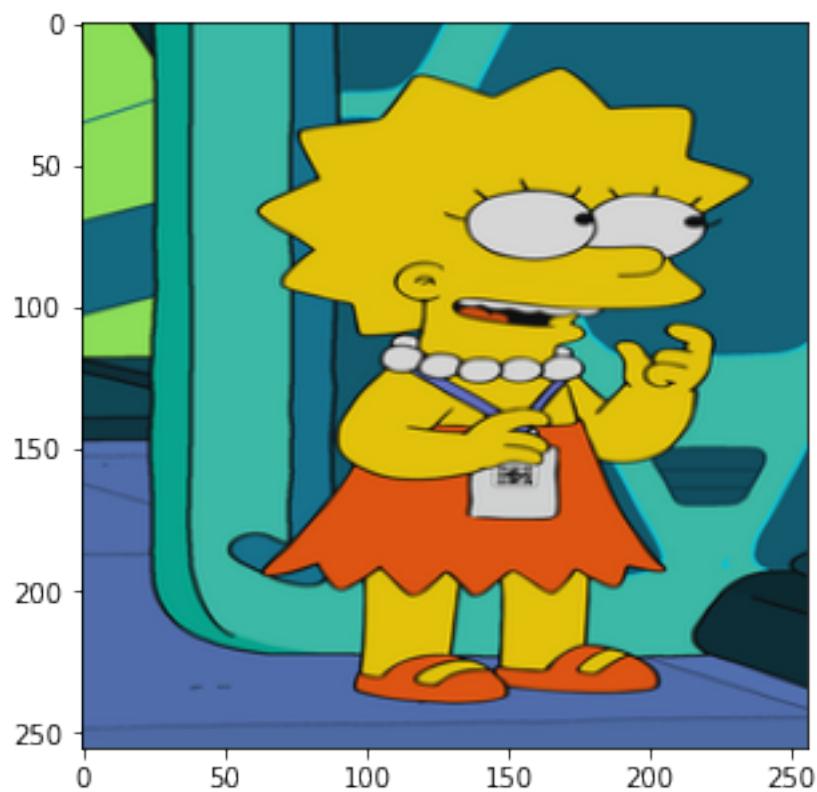
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1195FCD90>



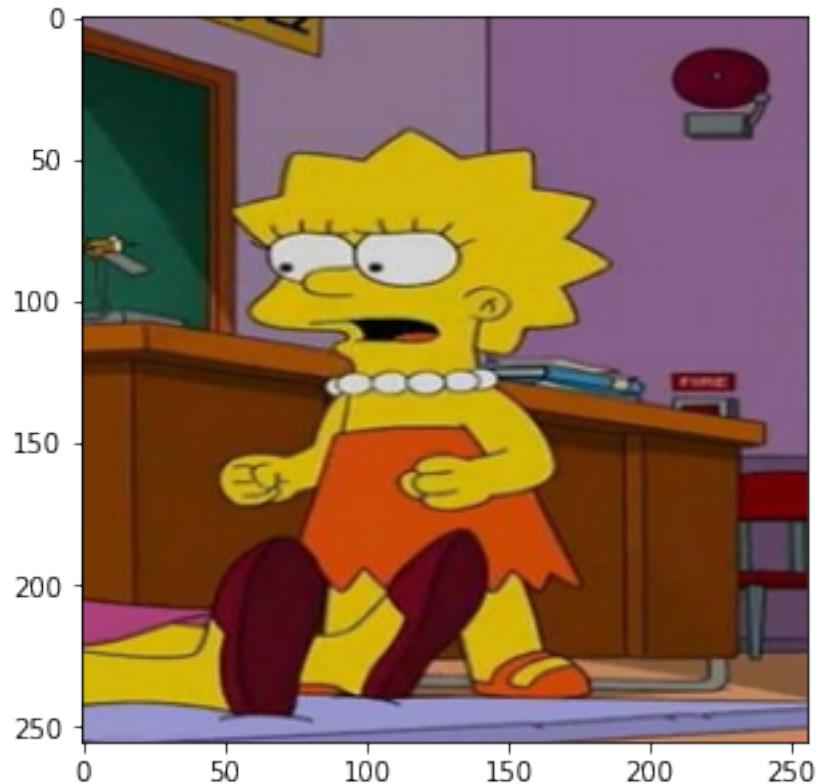
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC350>



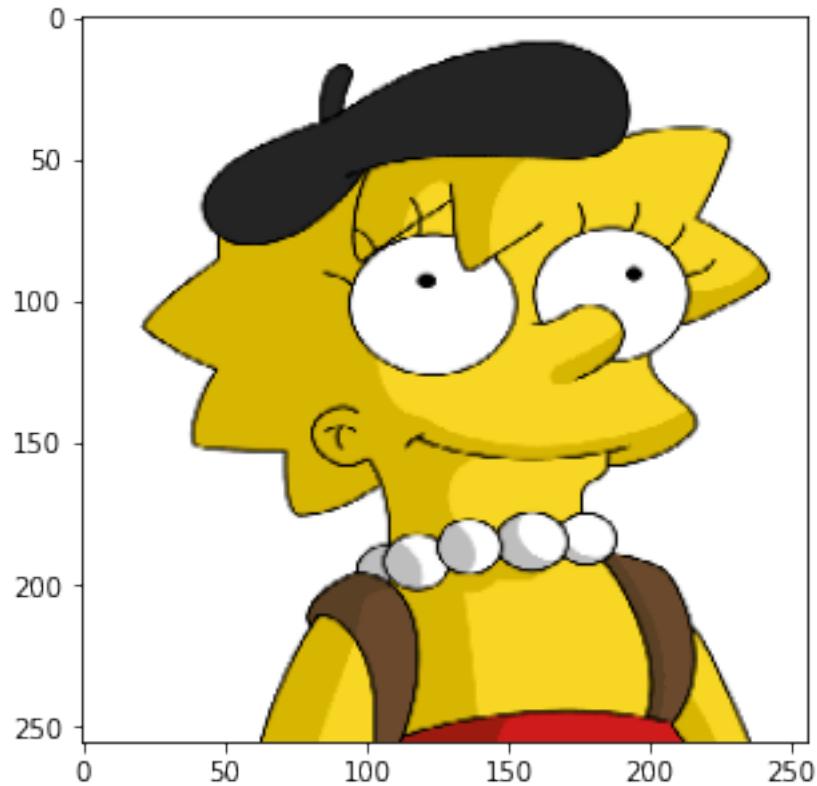
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1195FCE50>



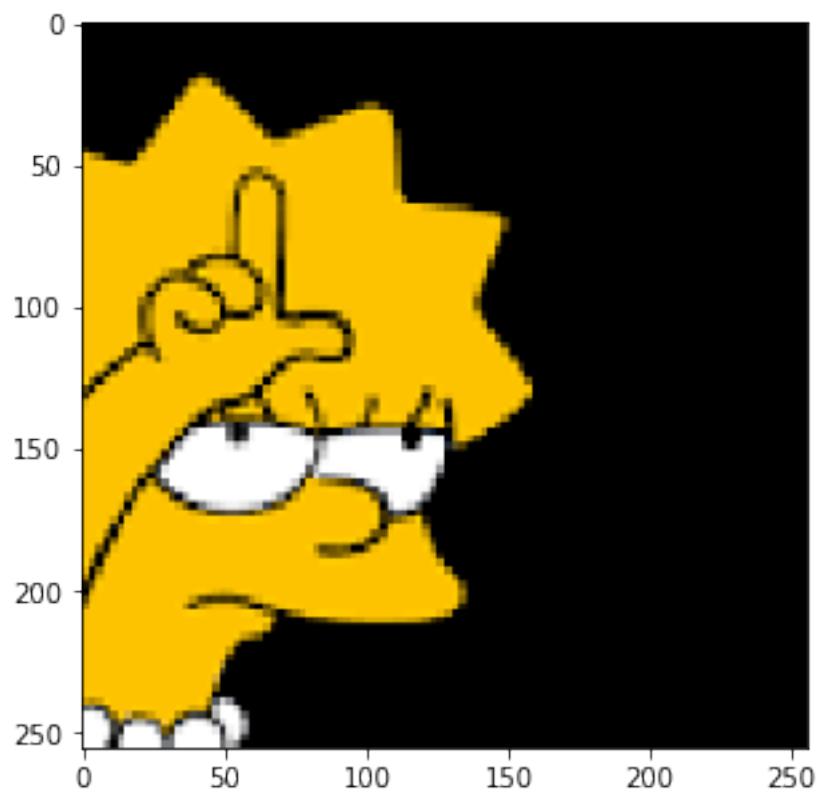
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC110>



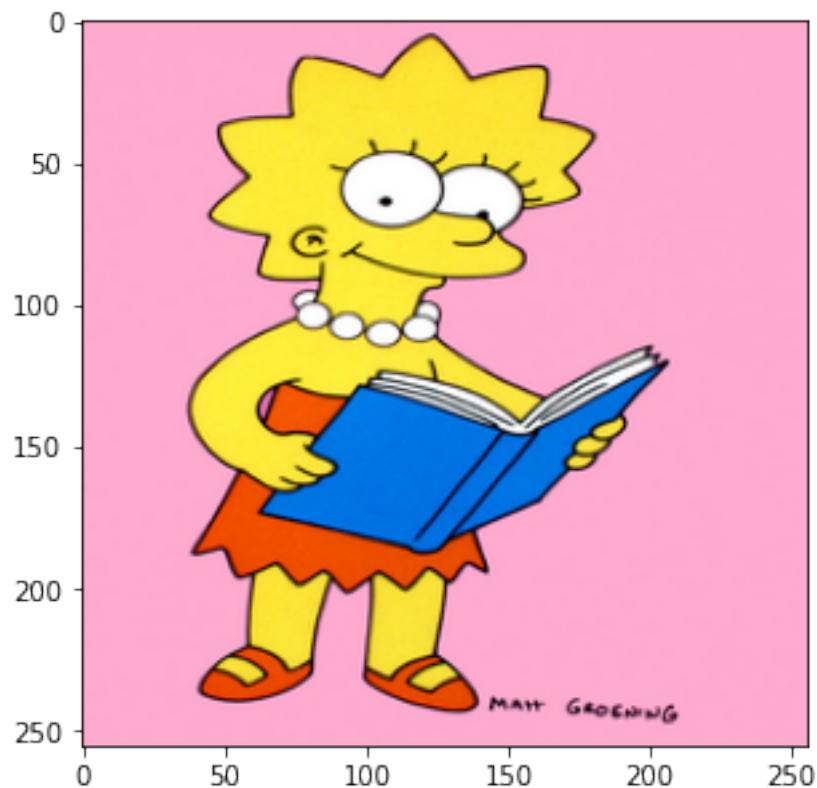
<PIL.Image.Image image mode=P size=256x256 at 0x7FB1195FCD10>



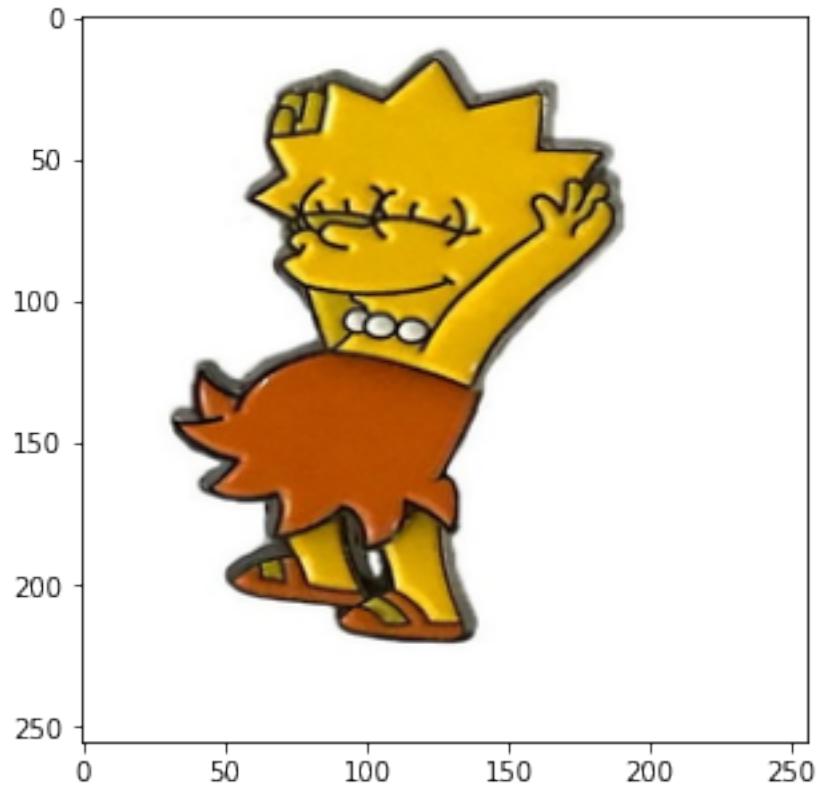
<PIL.Image.Image image mode=P size=256x256 at 0x7FB1195FC9D0>



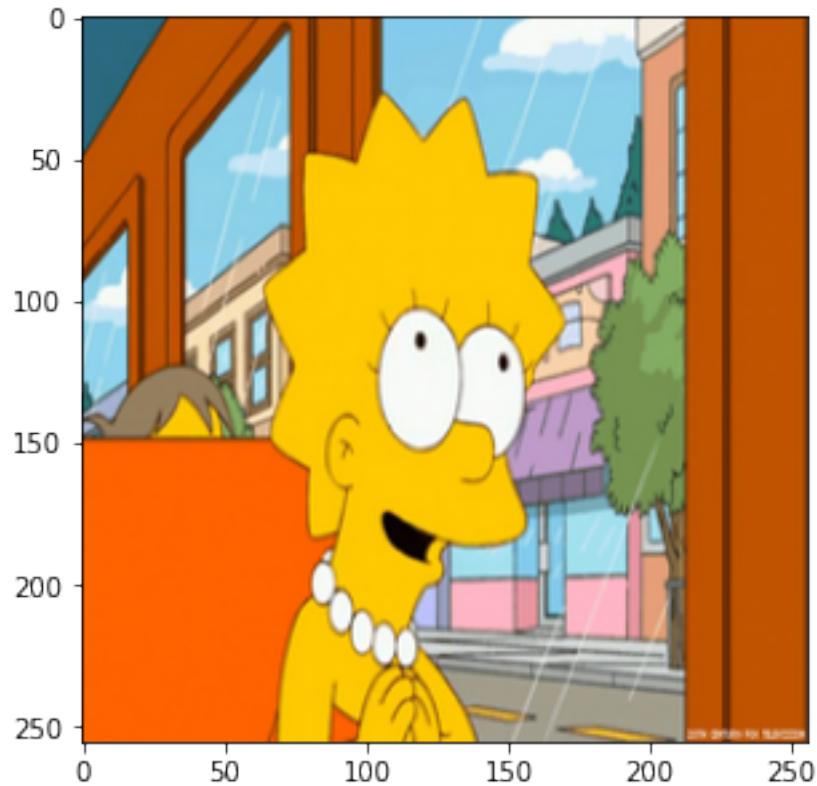
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FCC10>



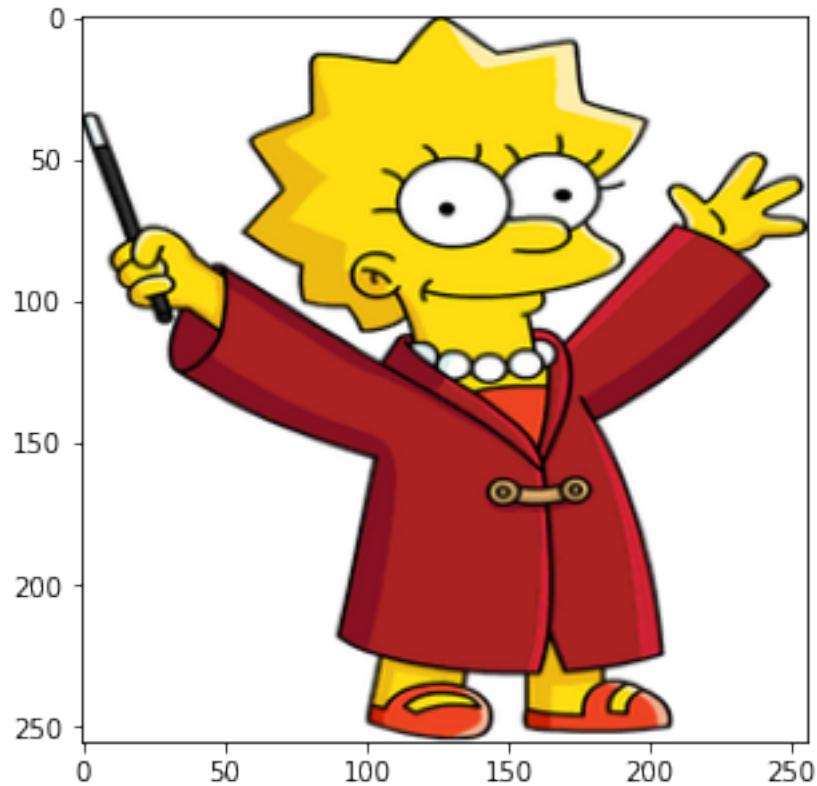
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC310>



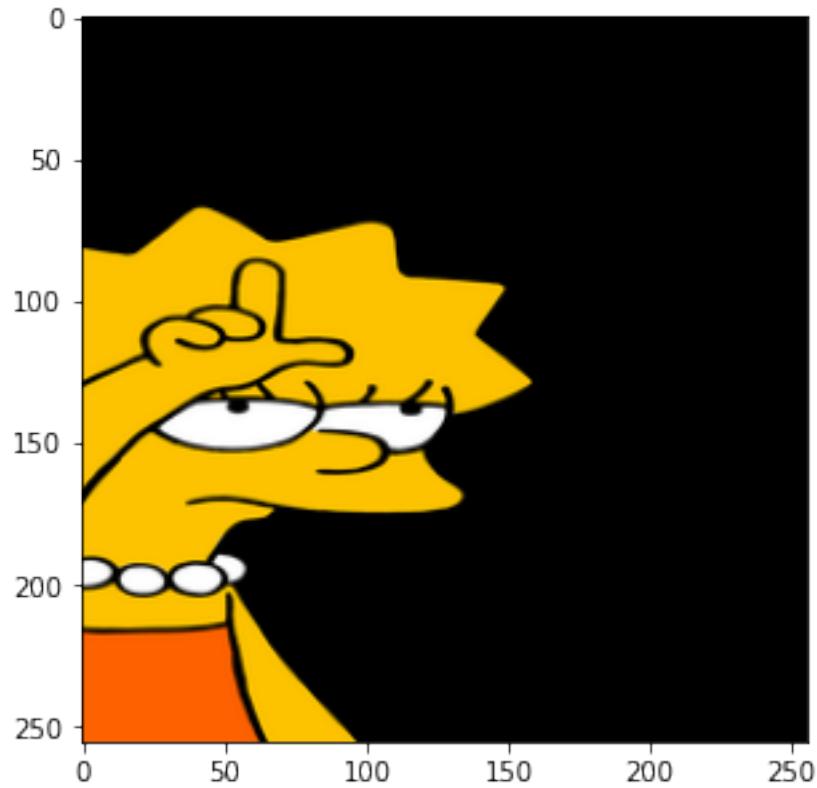
```
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FCB10>
```



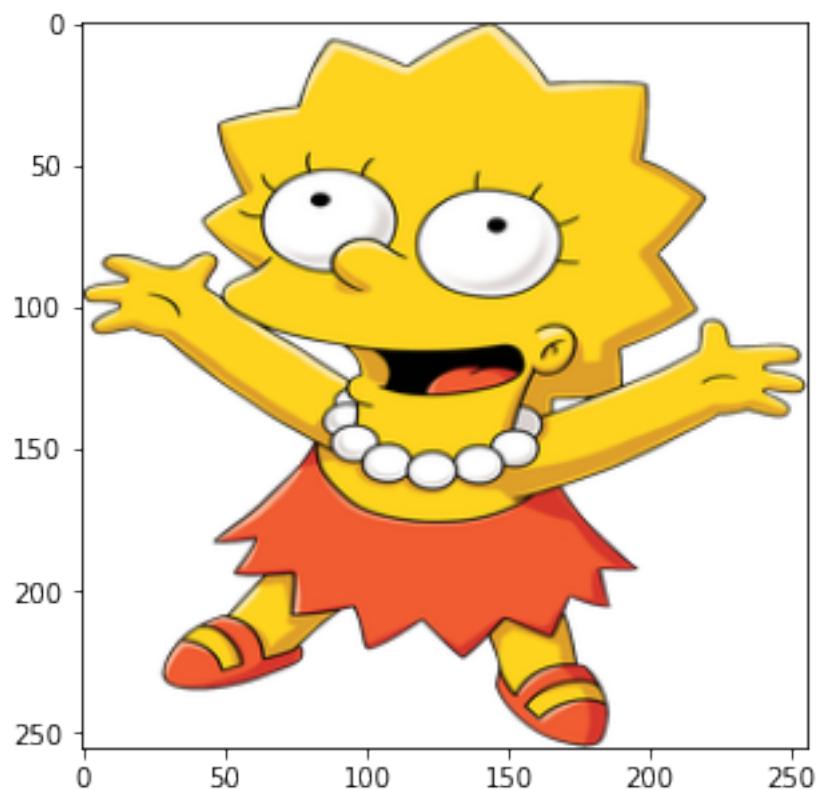
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1195FC910>



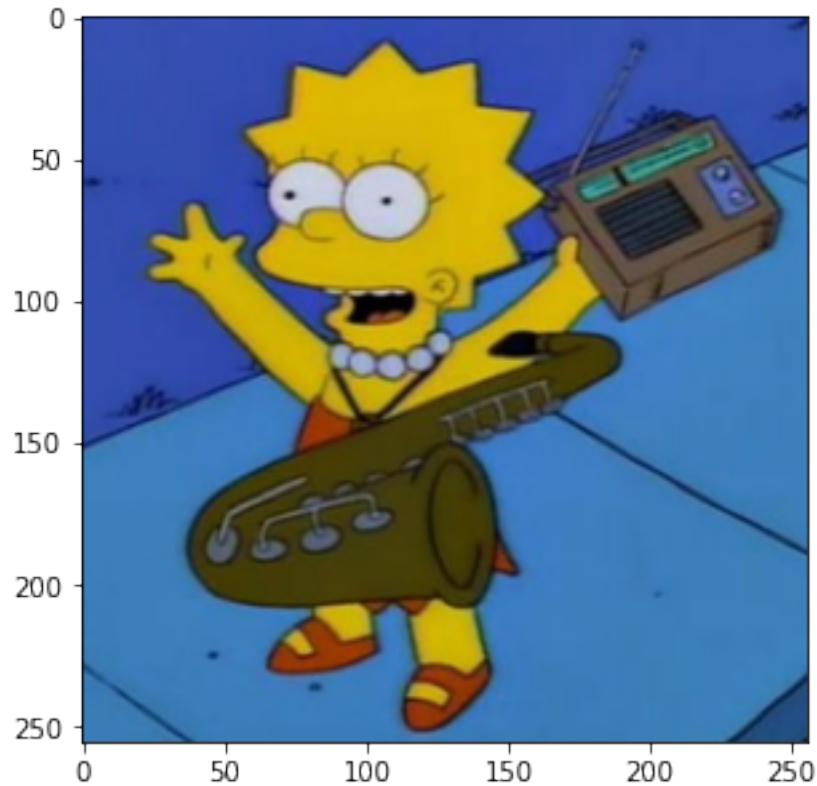
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119479690>



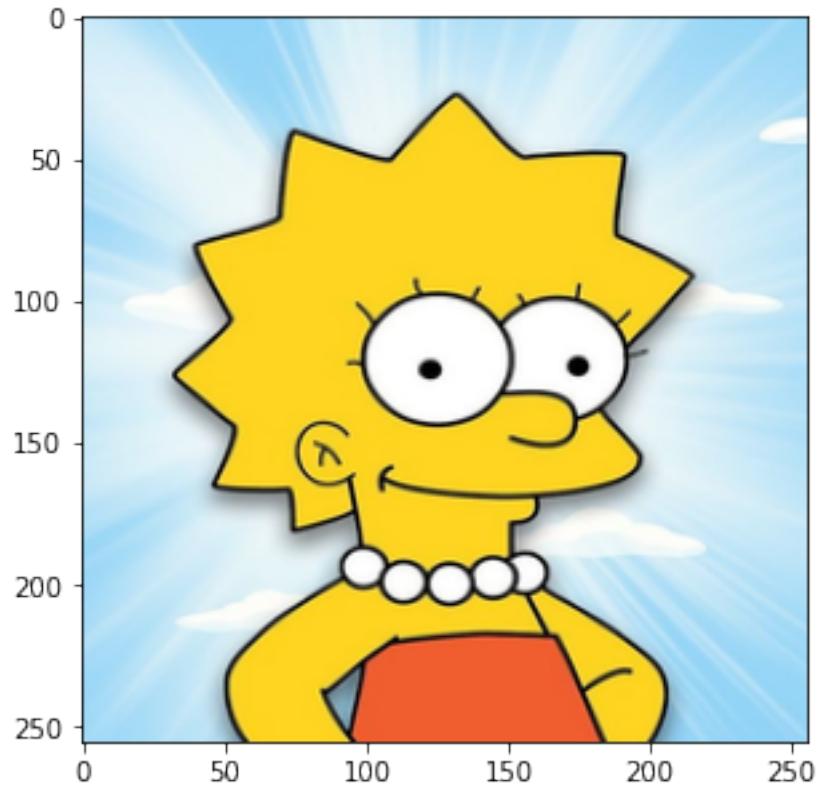
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119479490>



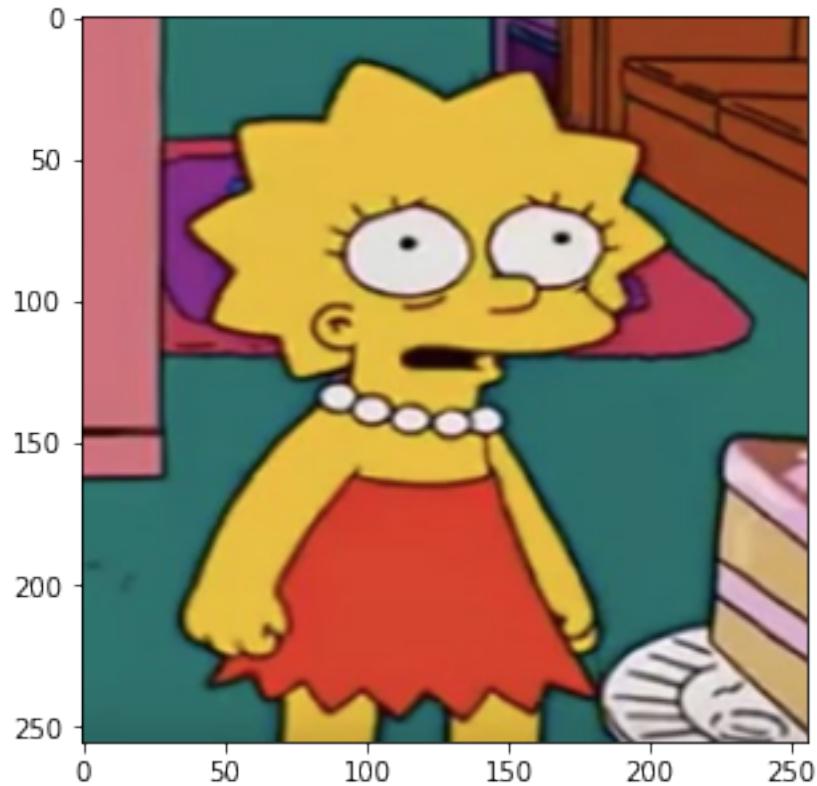
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FCA50>



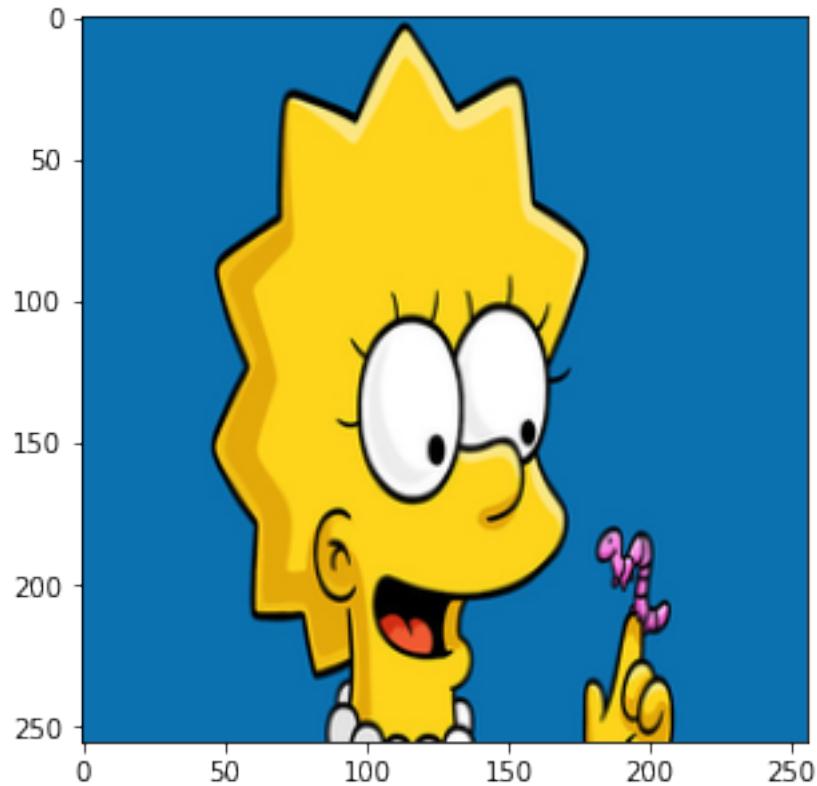
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FC190>



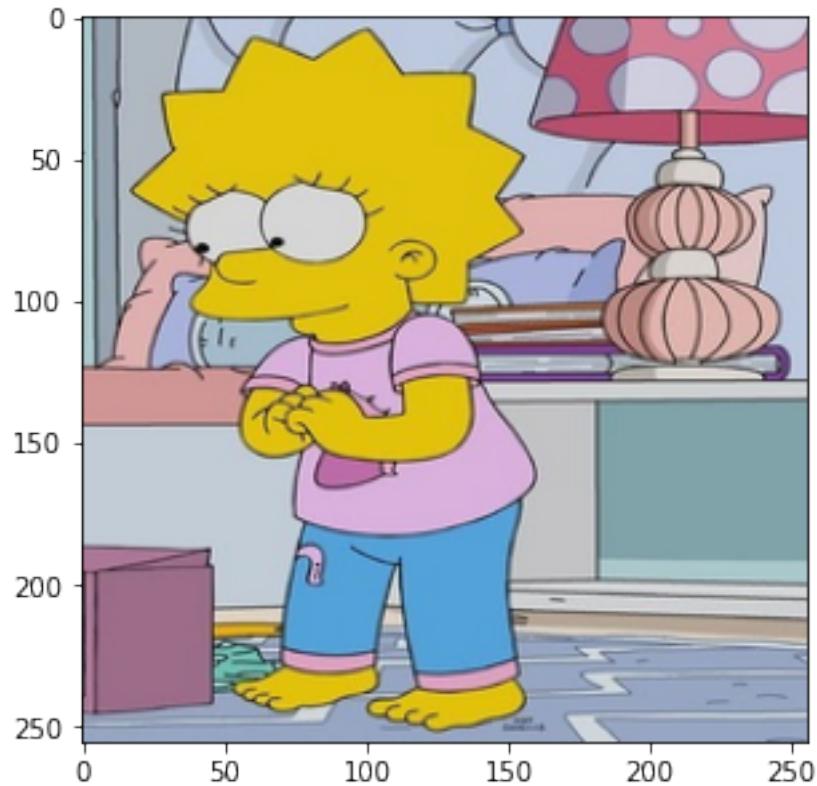
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1195FCDD0>



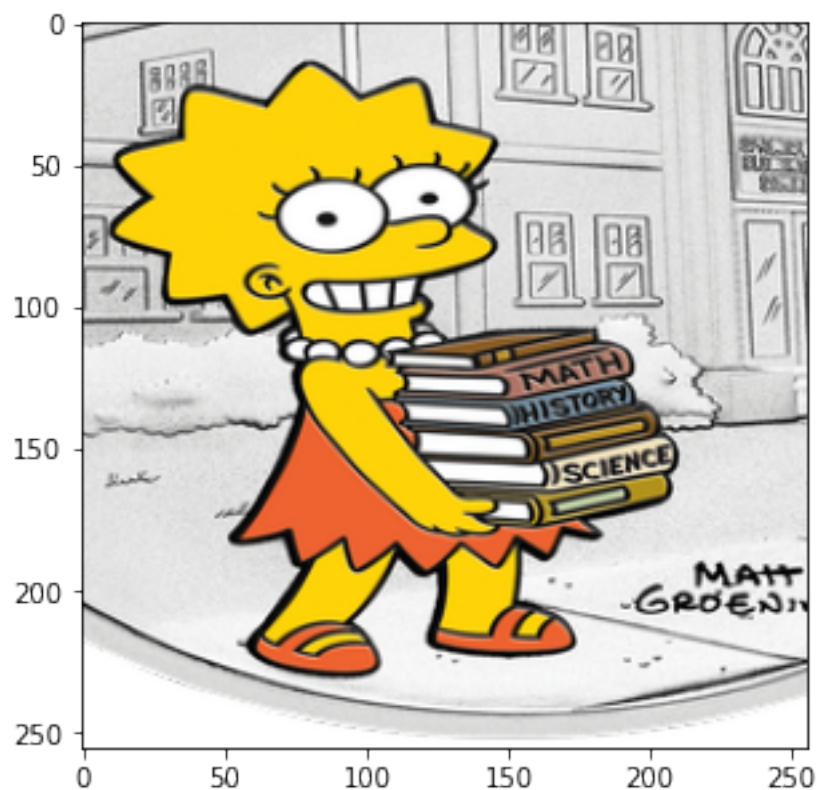
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479A50>



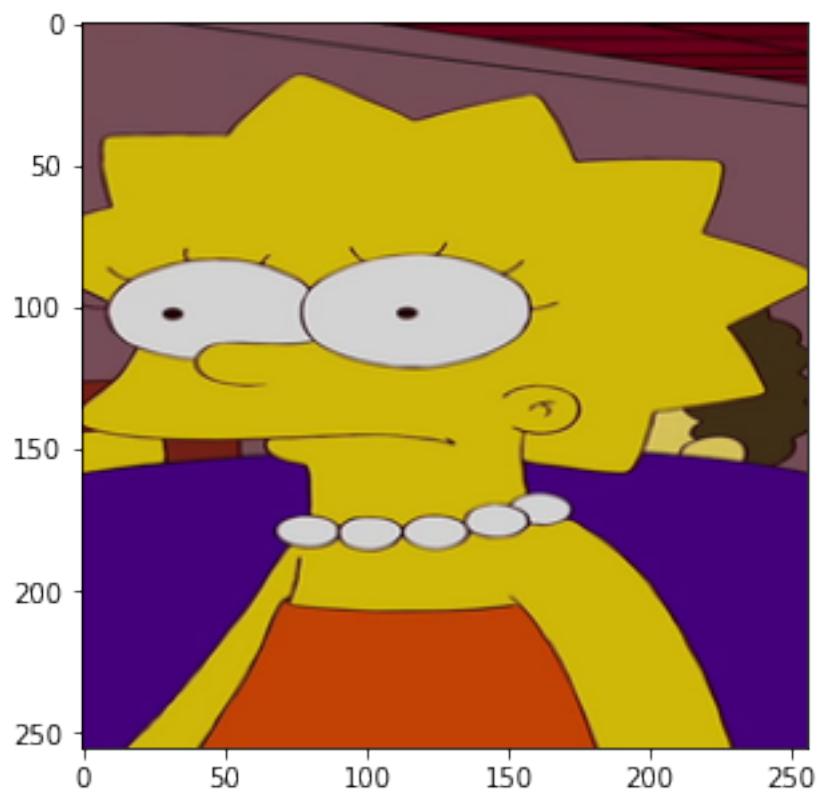
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479E50>



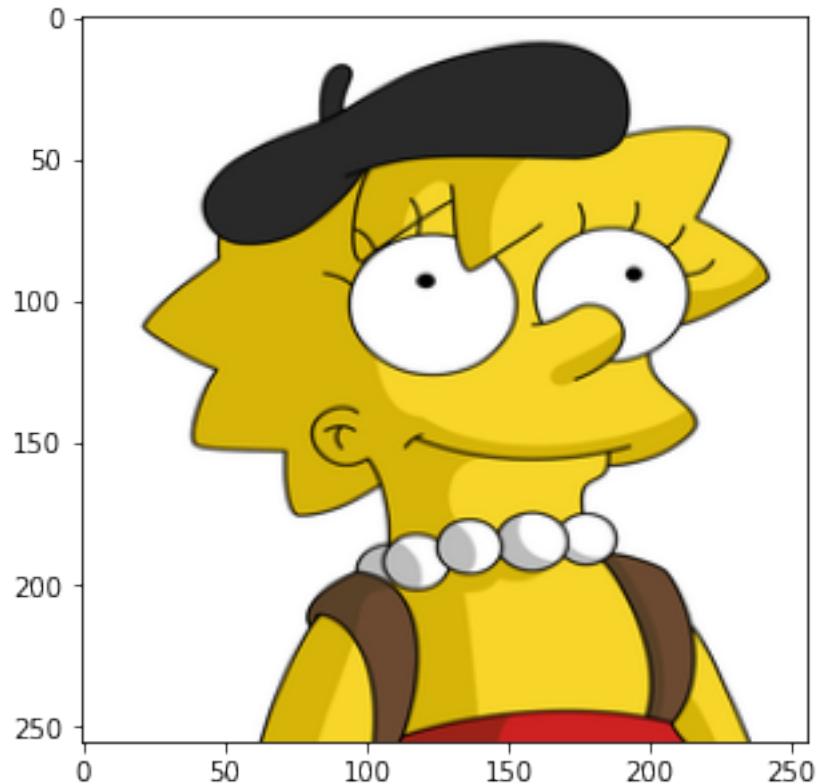
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119479650>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479CD0>



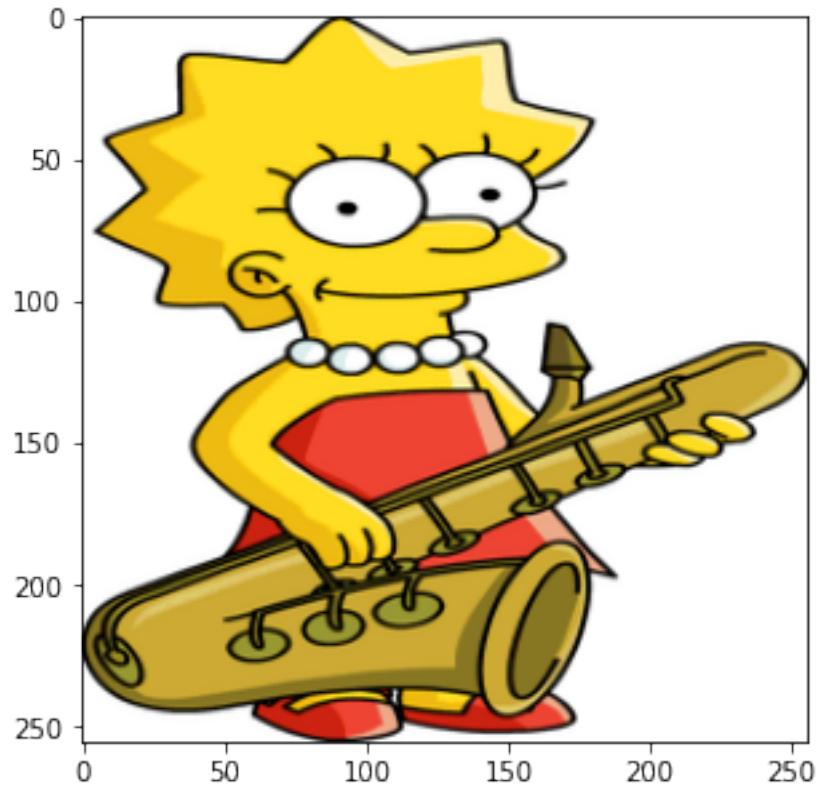
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479DD0>



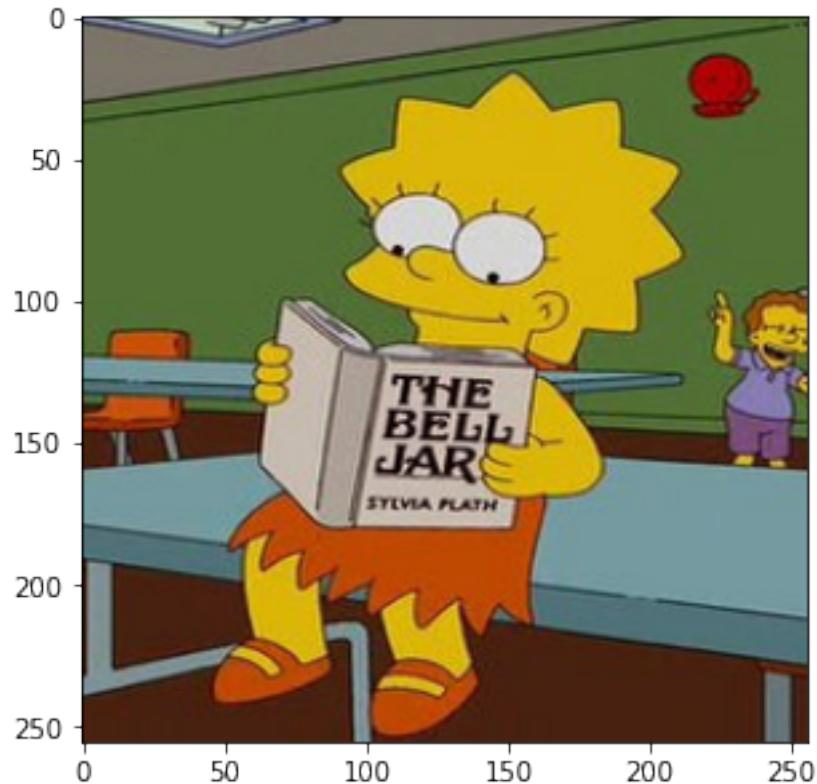
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119B879D0>



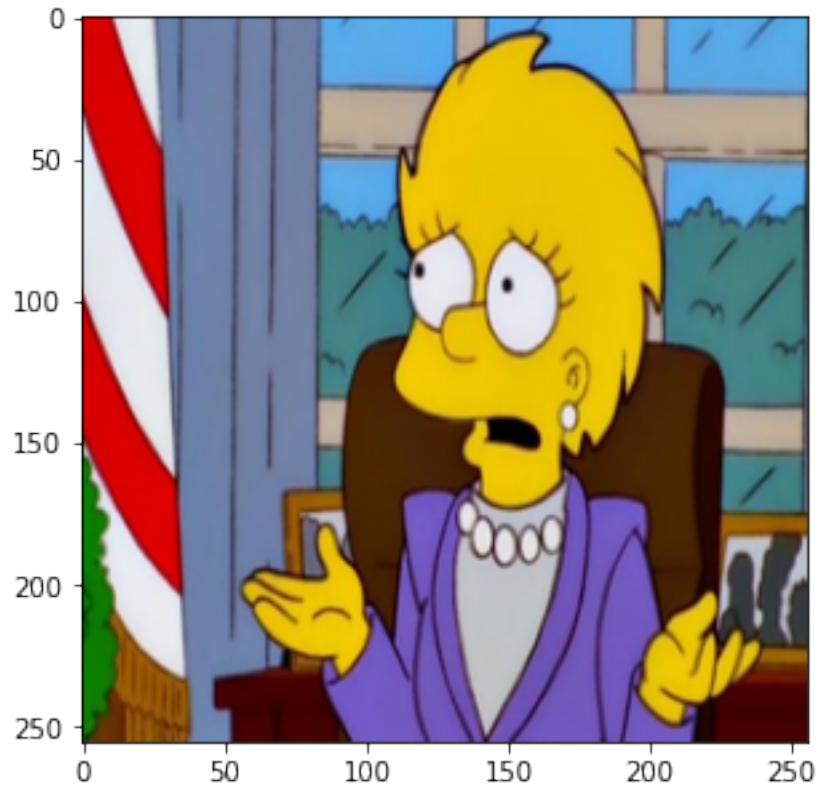
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119479410>



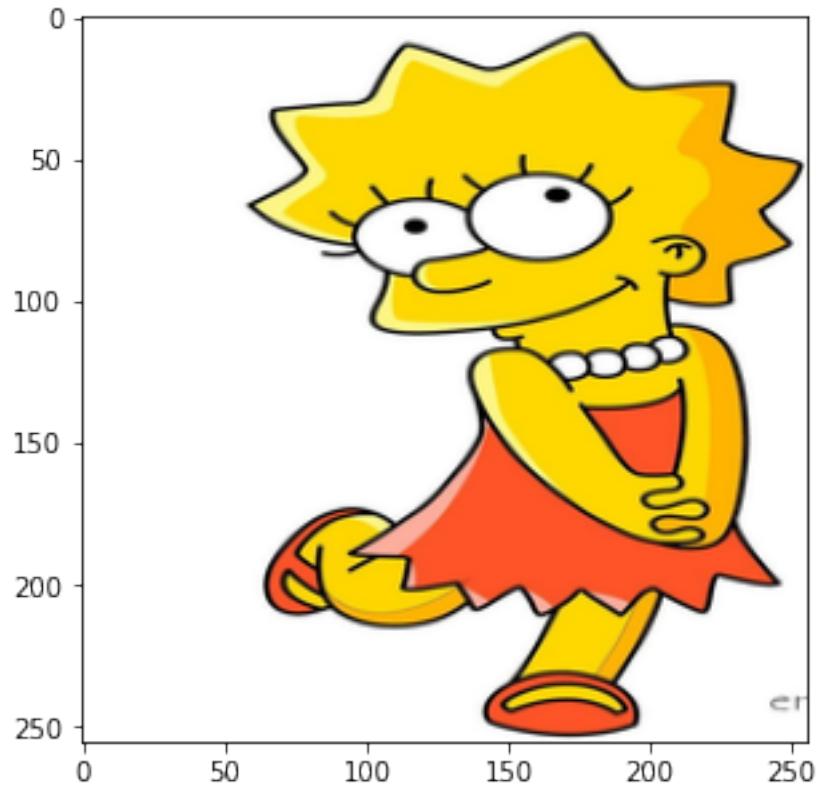
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479950>



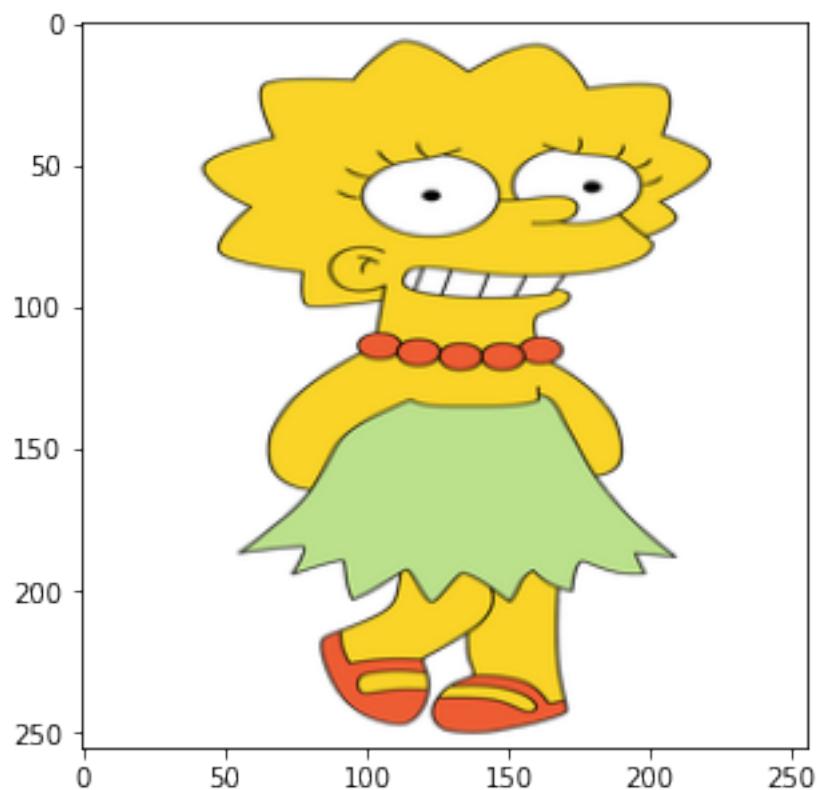
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479790>



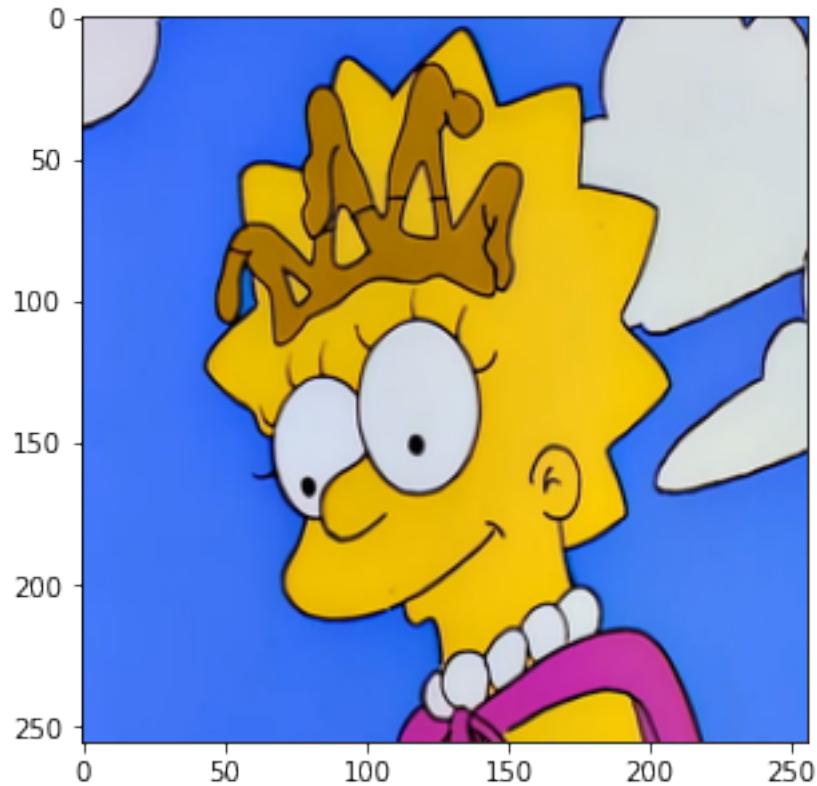
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479990>



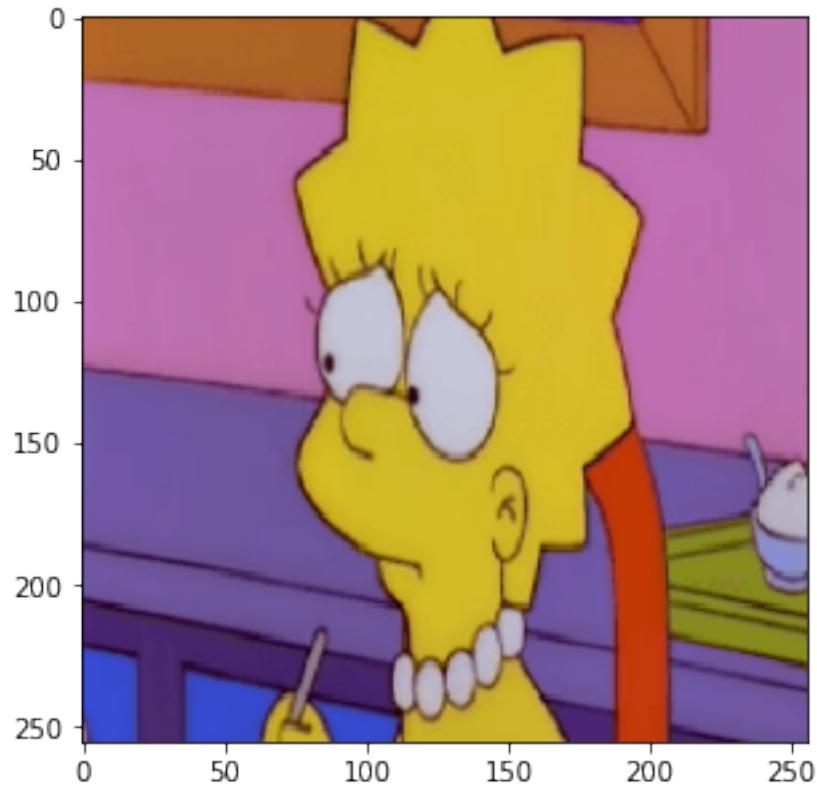
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB119479890>



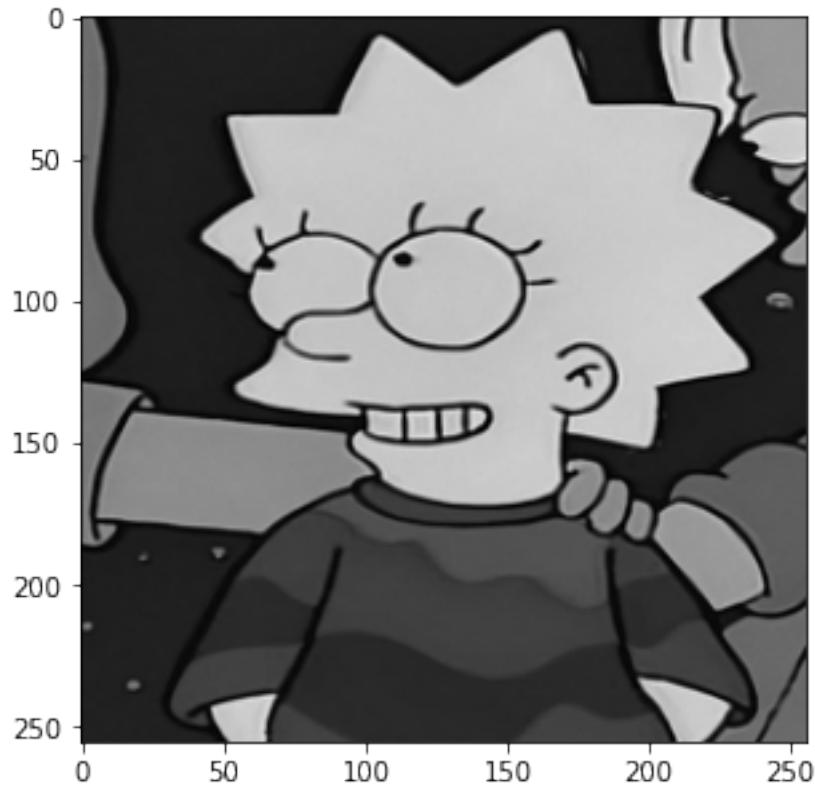
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479C10>



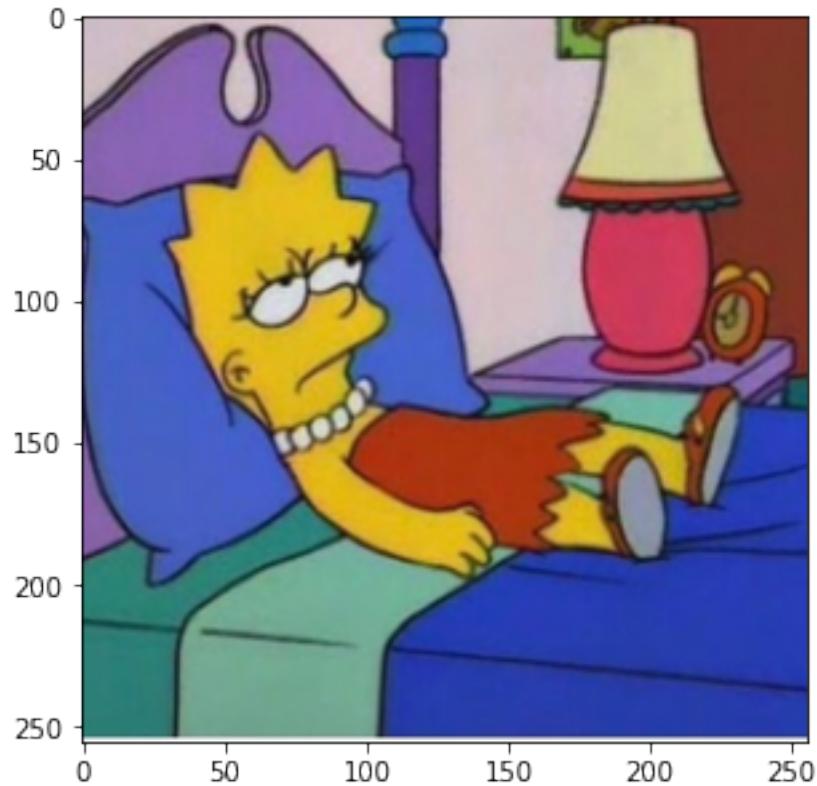
<PIL.Image.Image image mode=P size=256x256 at 0x7FB119479A90>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479C90>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479710>

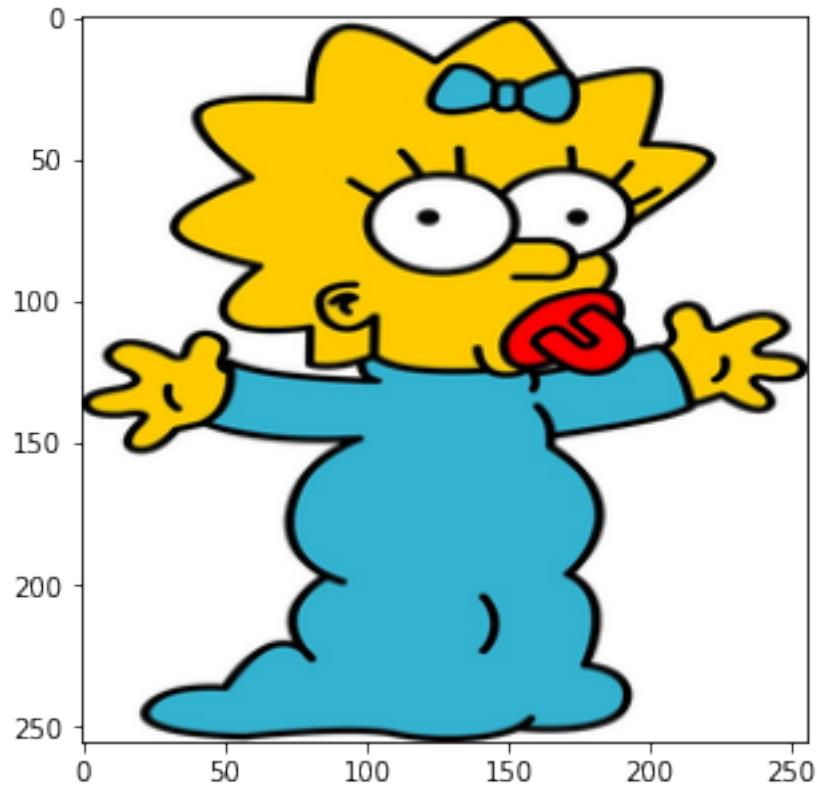


<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479D90>

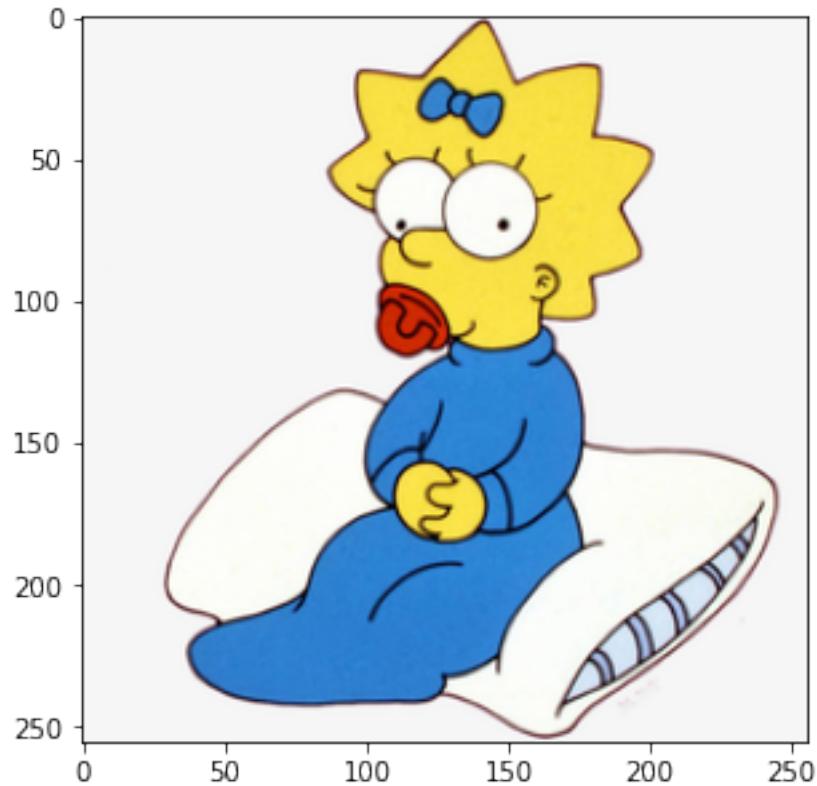


Maggie -- Resized

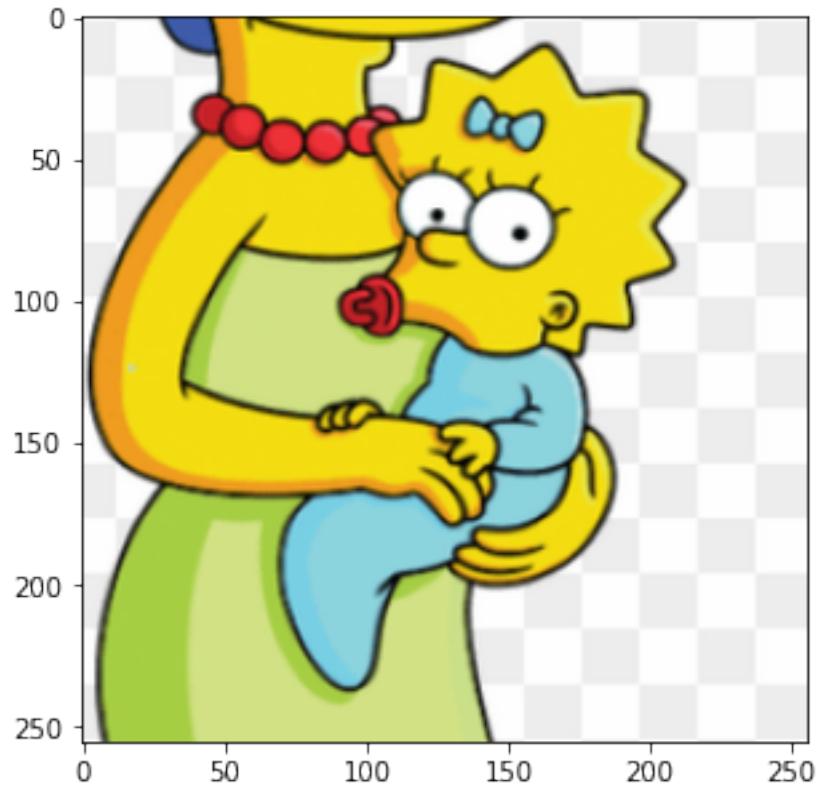
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479D50>



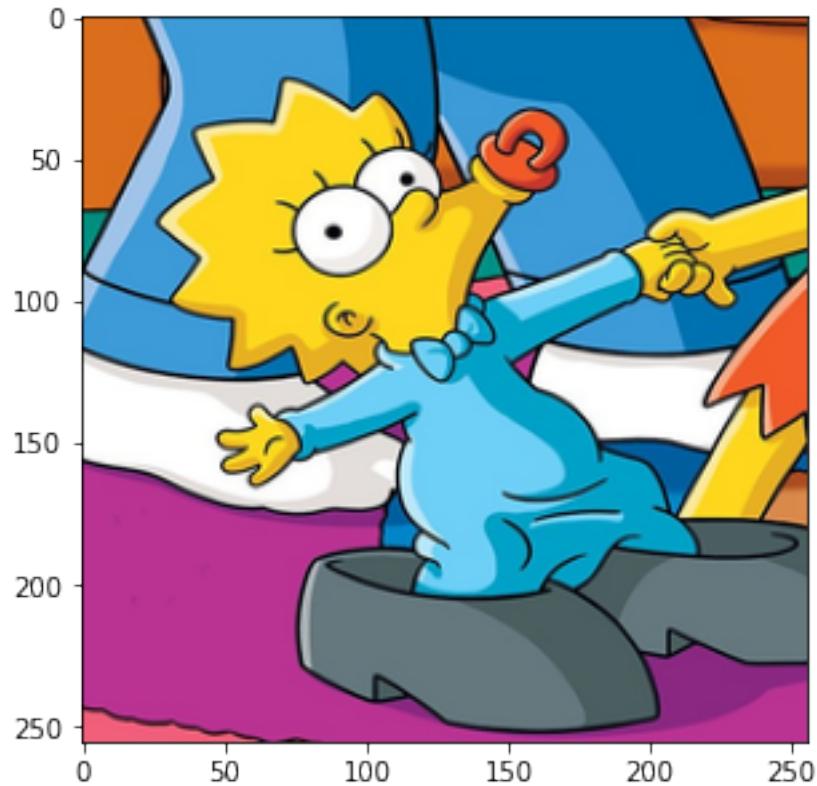
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479910>



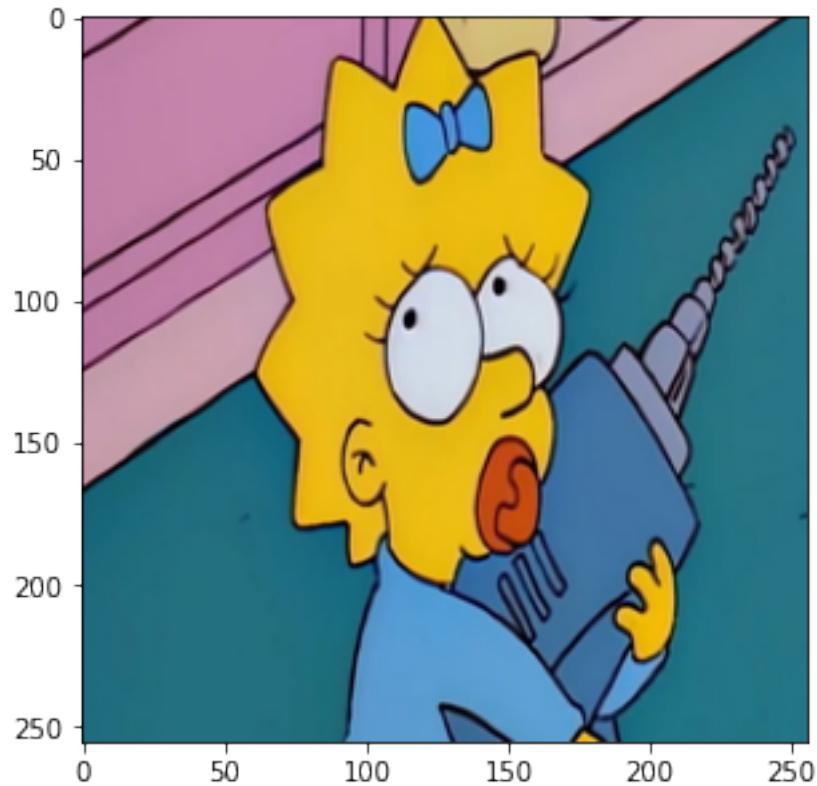
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB11B4B20D0>



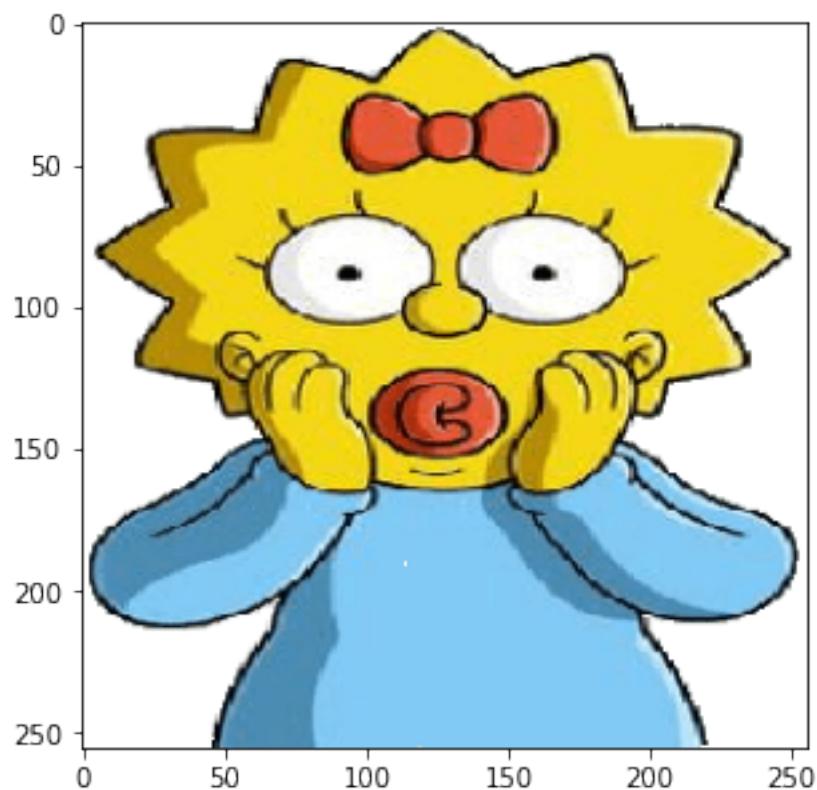
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479BD0>



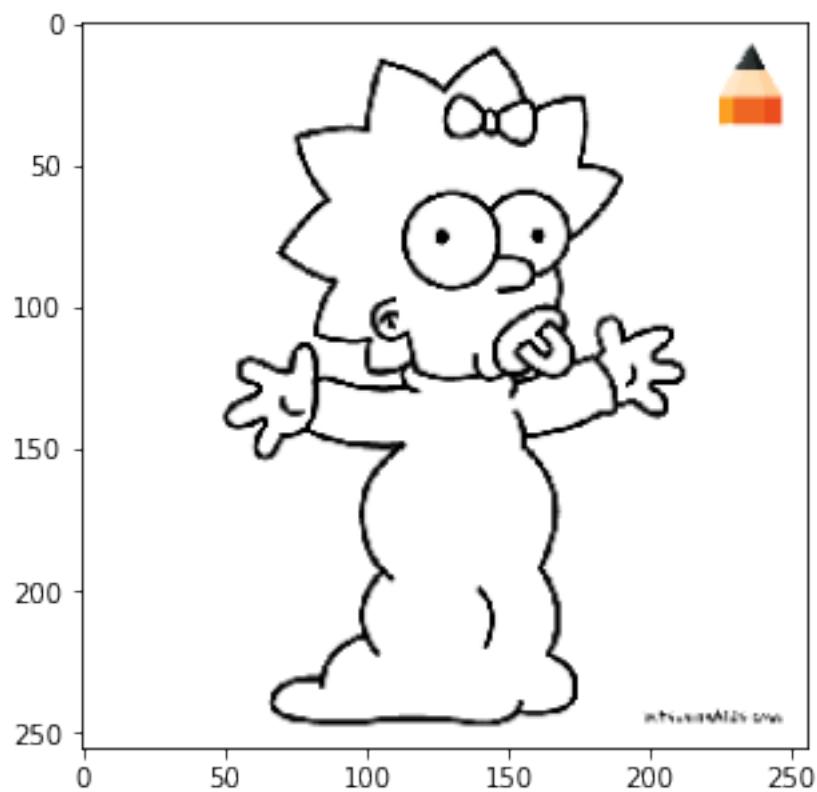
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479750>



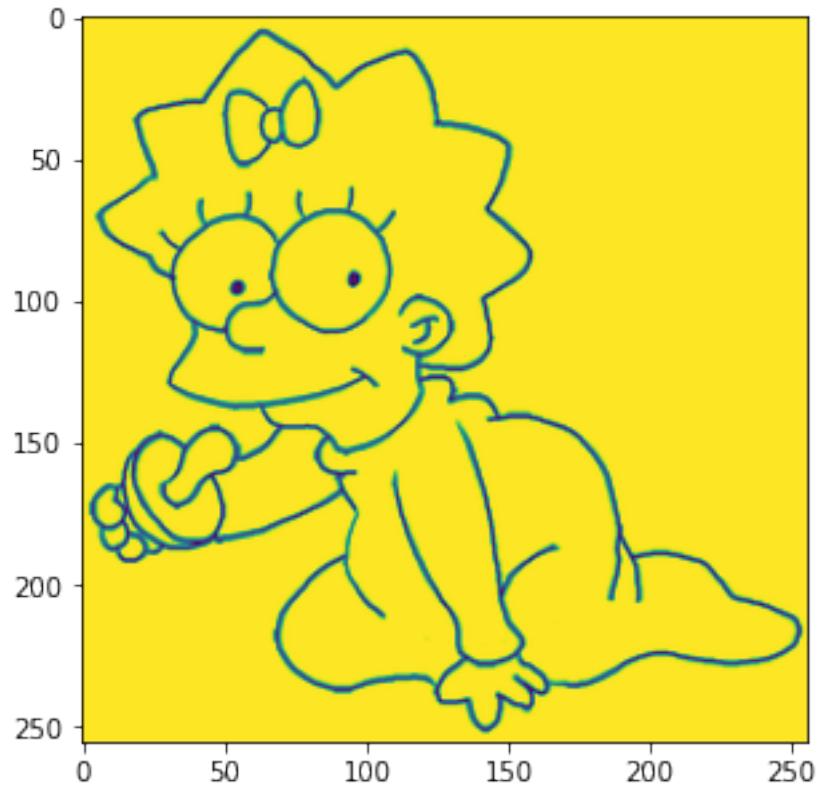
<PIL.Image.Image image mode=P size=256x256 at 0x7FB11B4B2150>



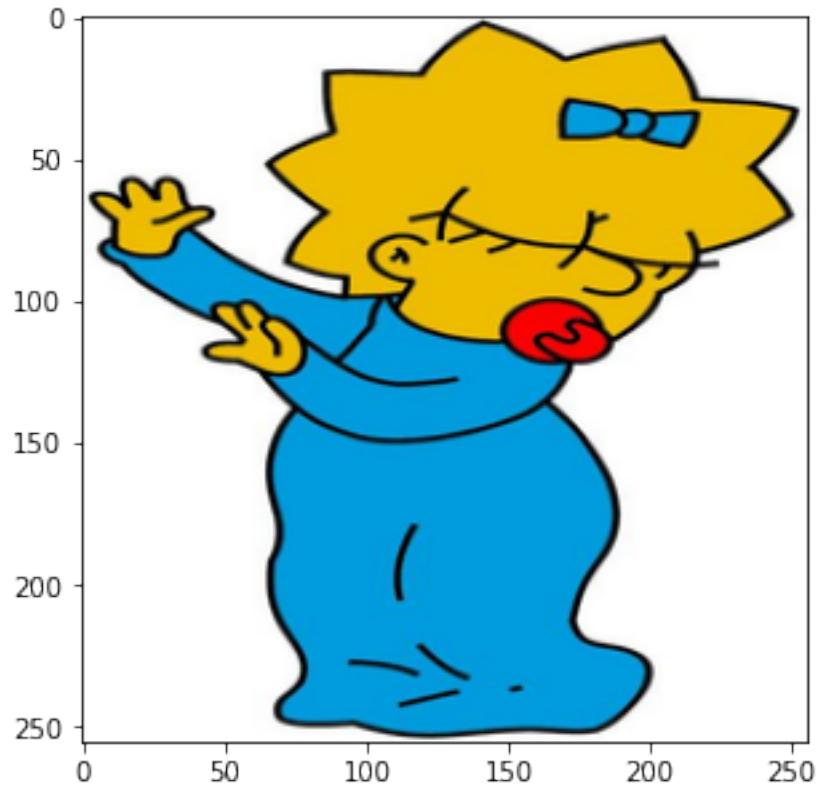
<PIL.Image.Image image mode=P size=256x256 at 0x7FB11B4B2210>



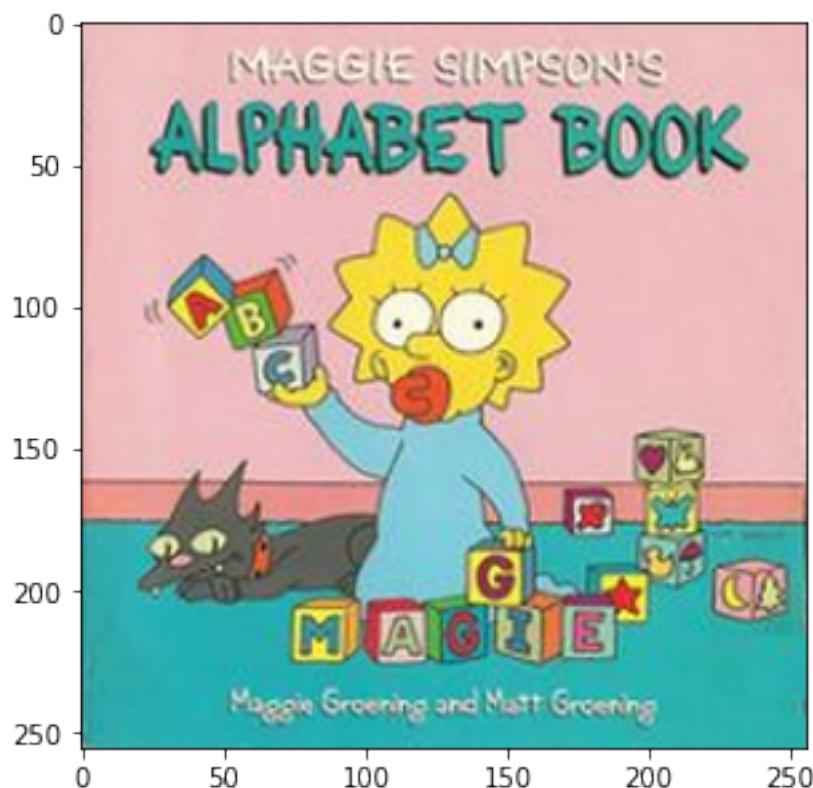
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11B4B2290>



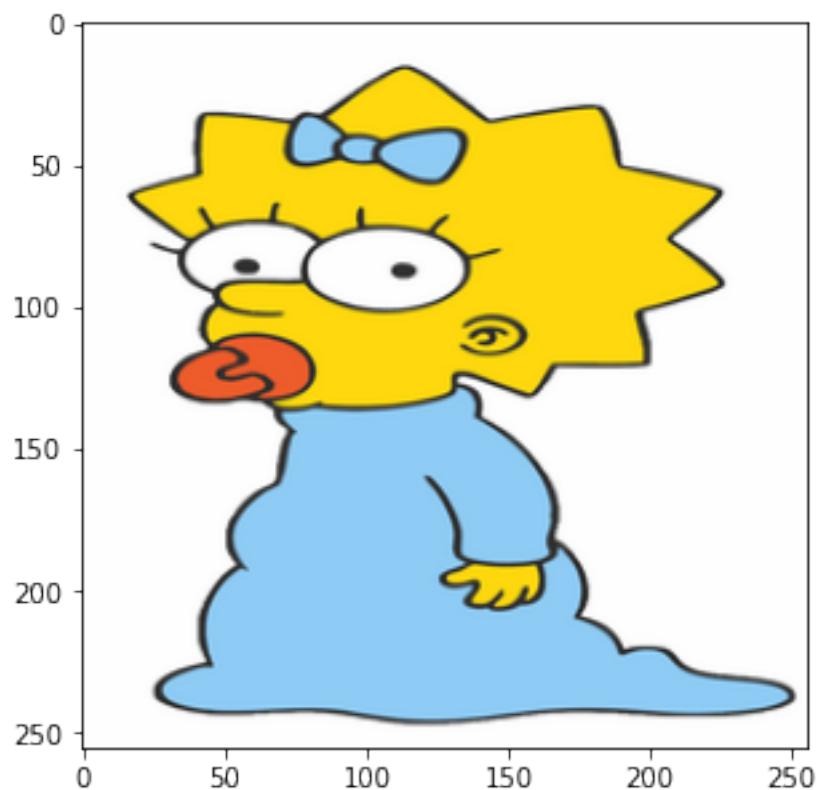
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2250>



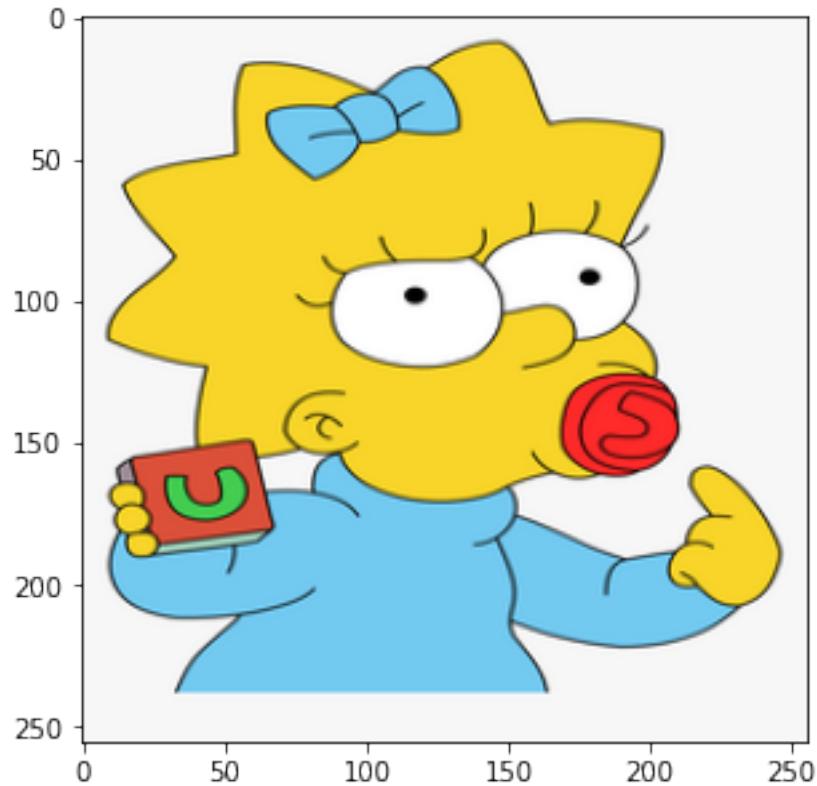
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2050>



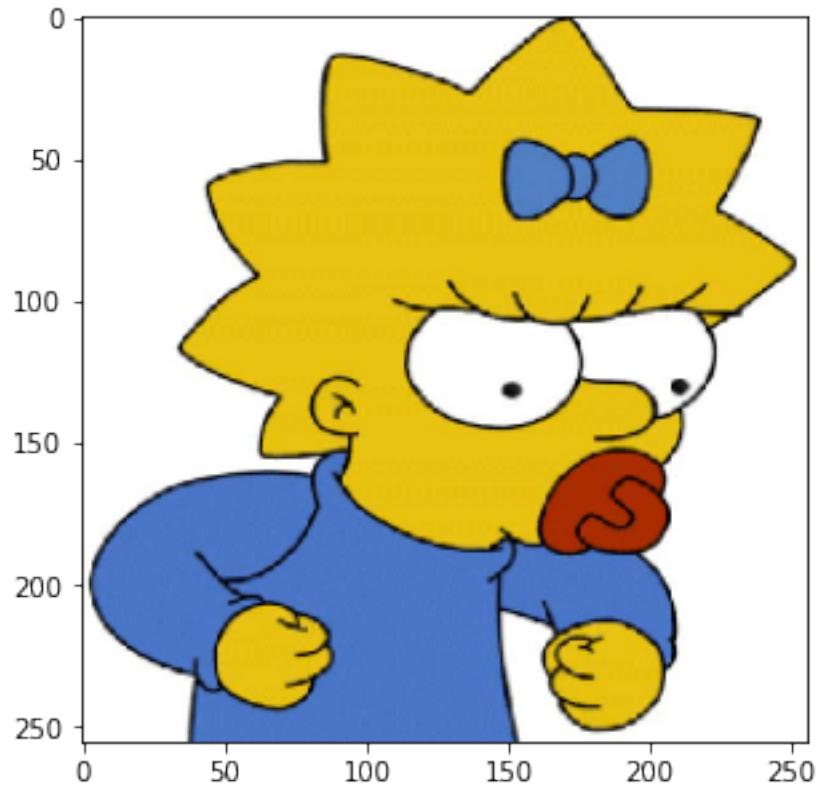
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2350>



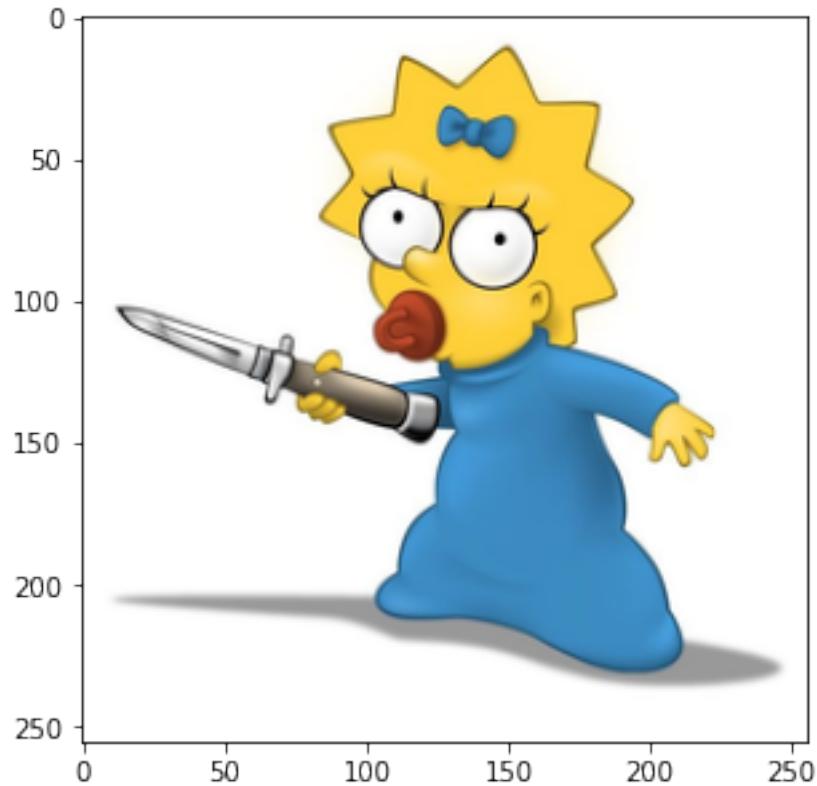
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB11B4B2410>



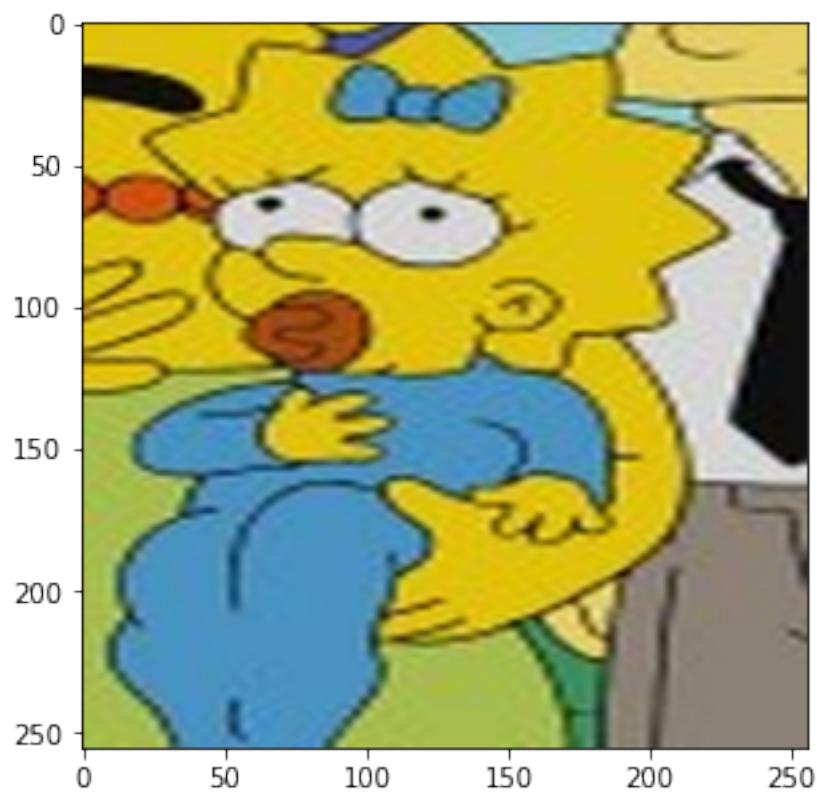
<PIL.Image.Image image mode=P size=256x256 at 0x7FB11B4B23D0>



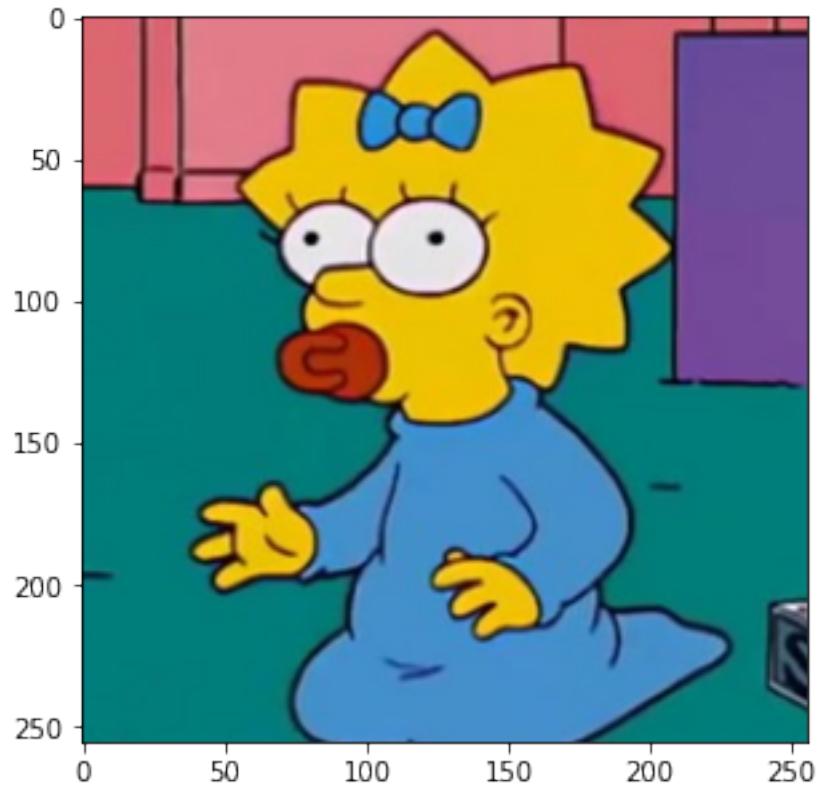
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2510>



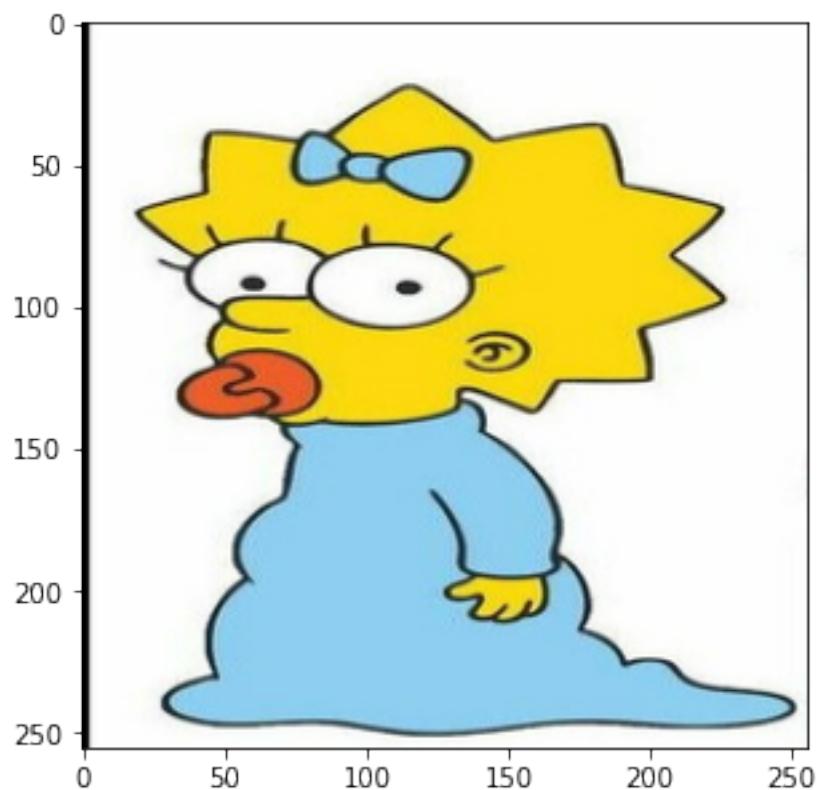
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B24D0>



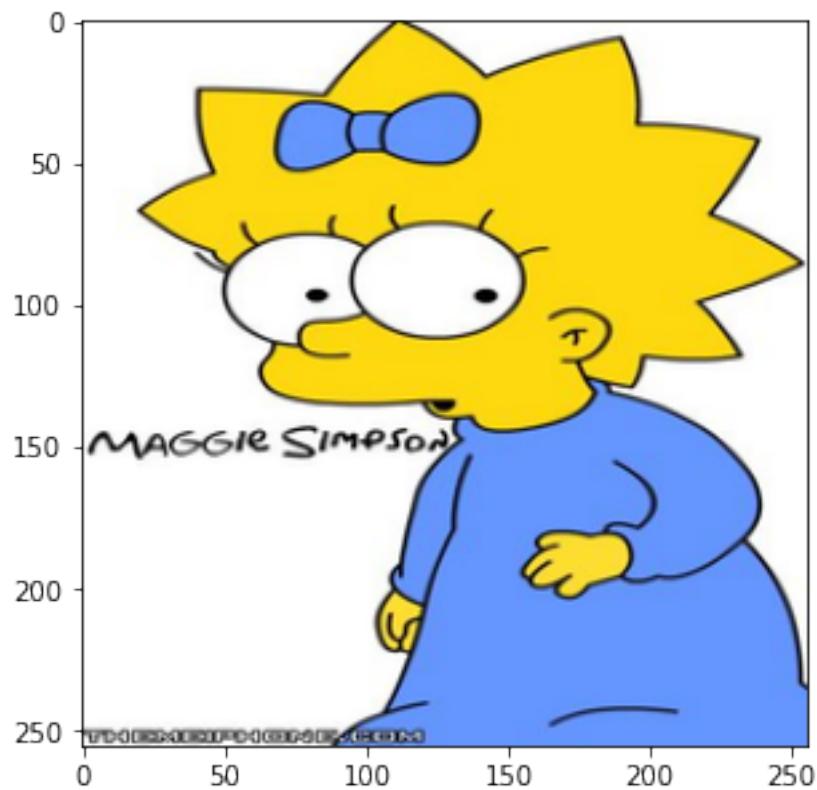
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B25D0>



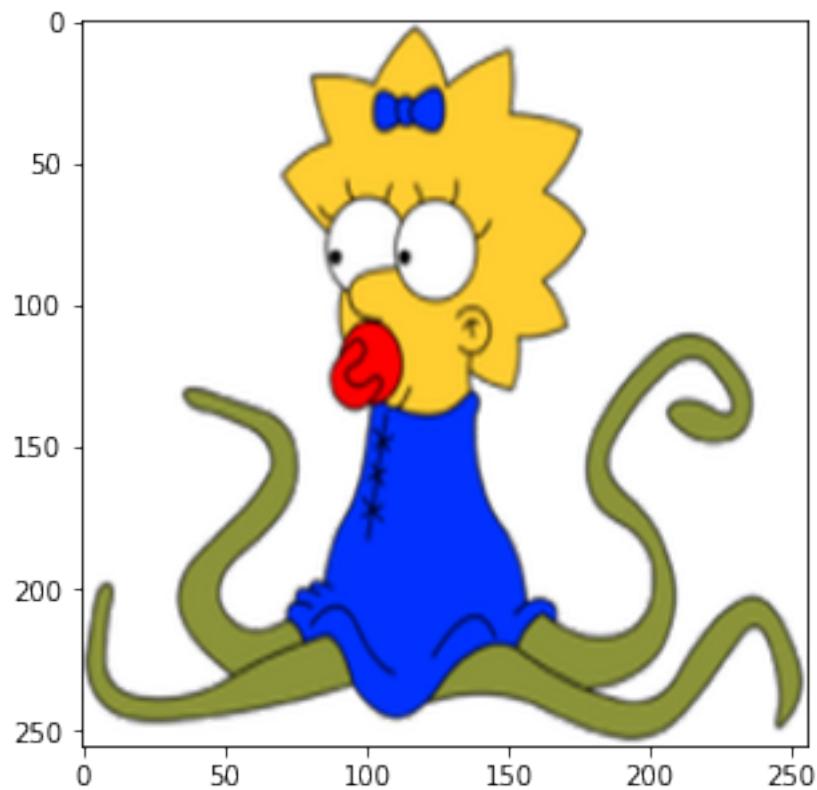
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2650>



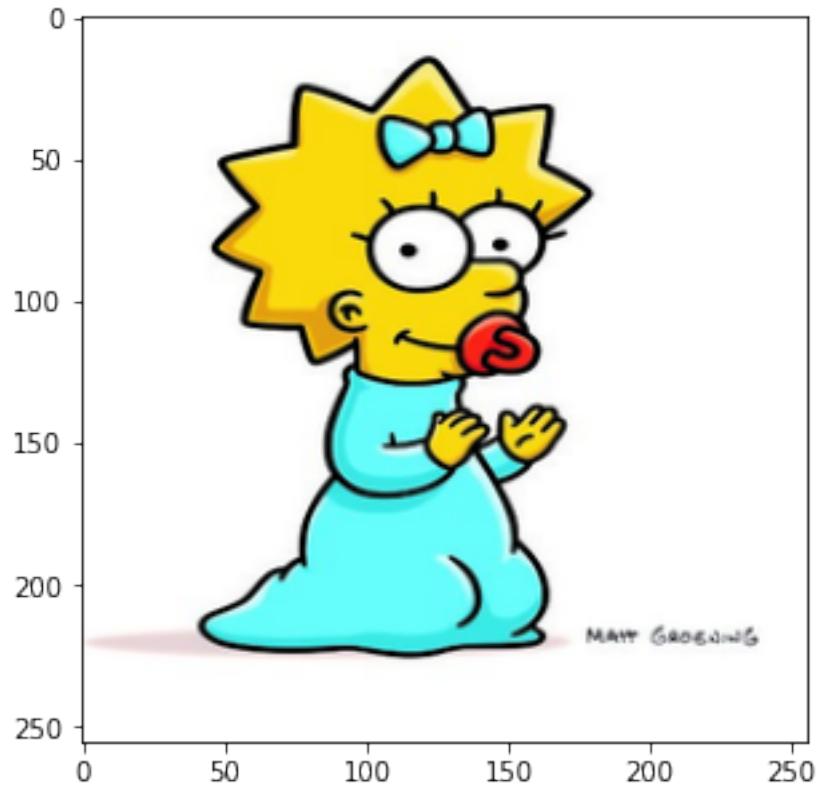
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B26D0>



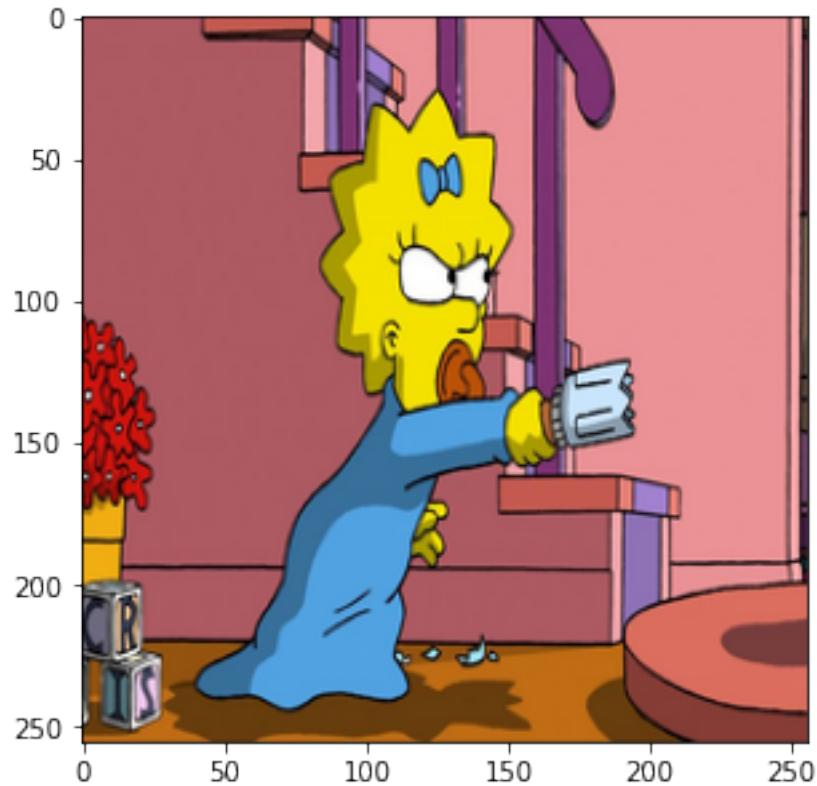
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB11B4B2790>



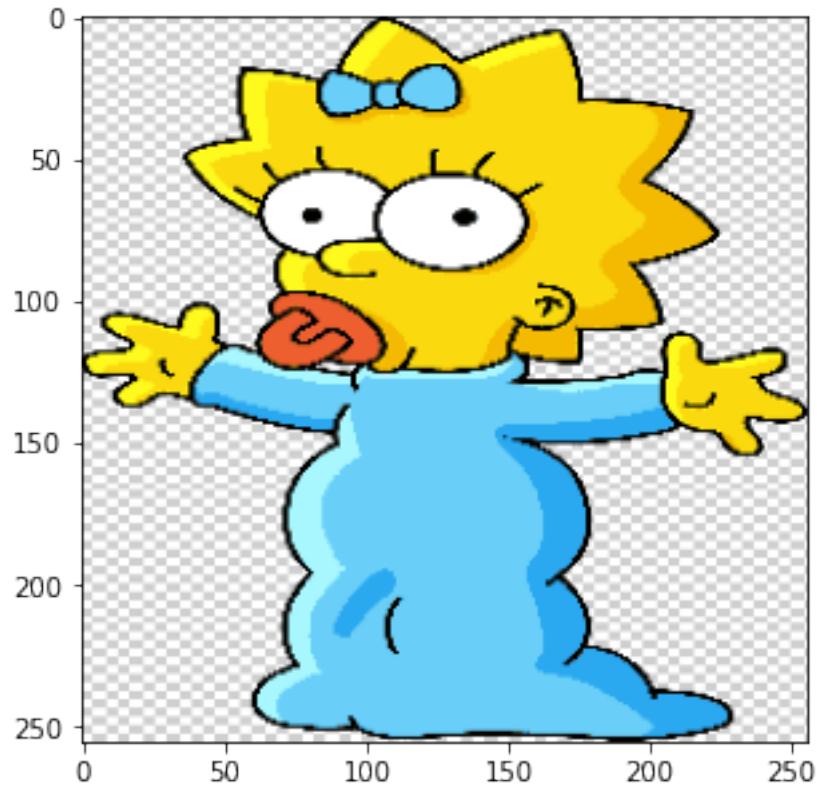
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2750>



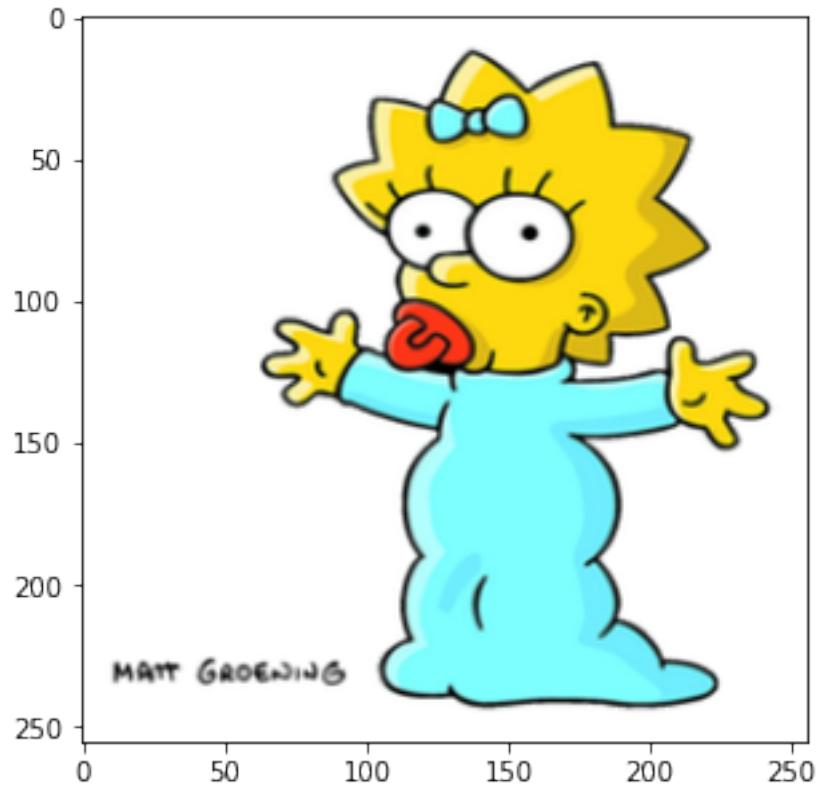
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B27D0>



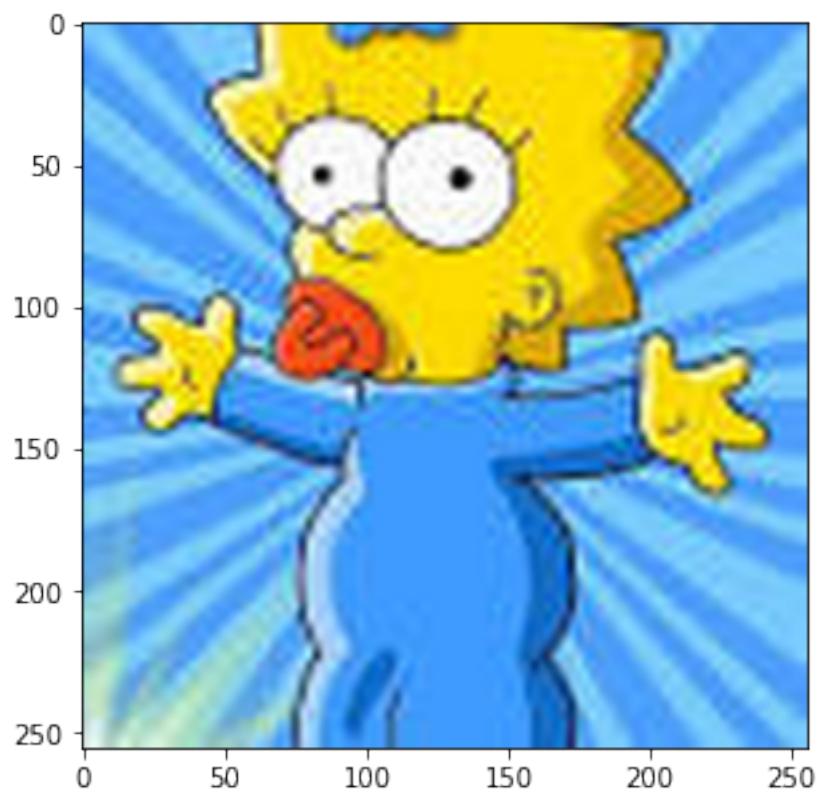
<PIL.Image.Image image mode=P size=256x256 at 0x7FB11B4B2850>



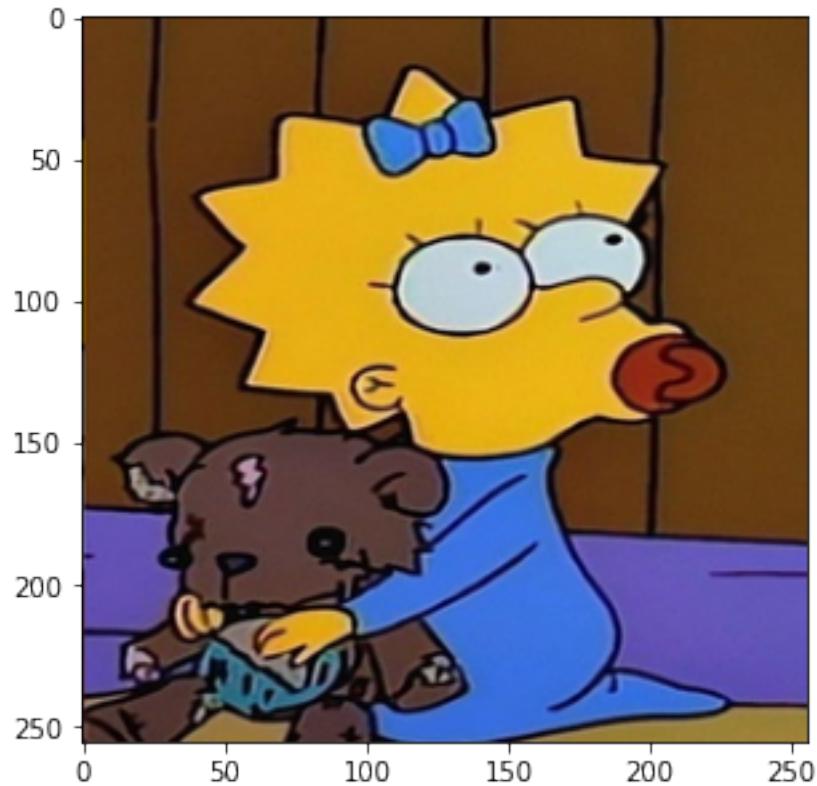
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB11B4B2950>



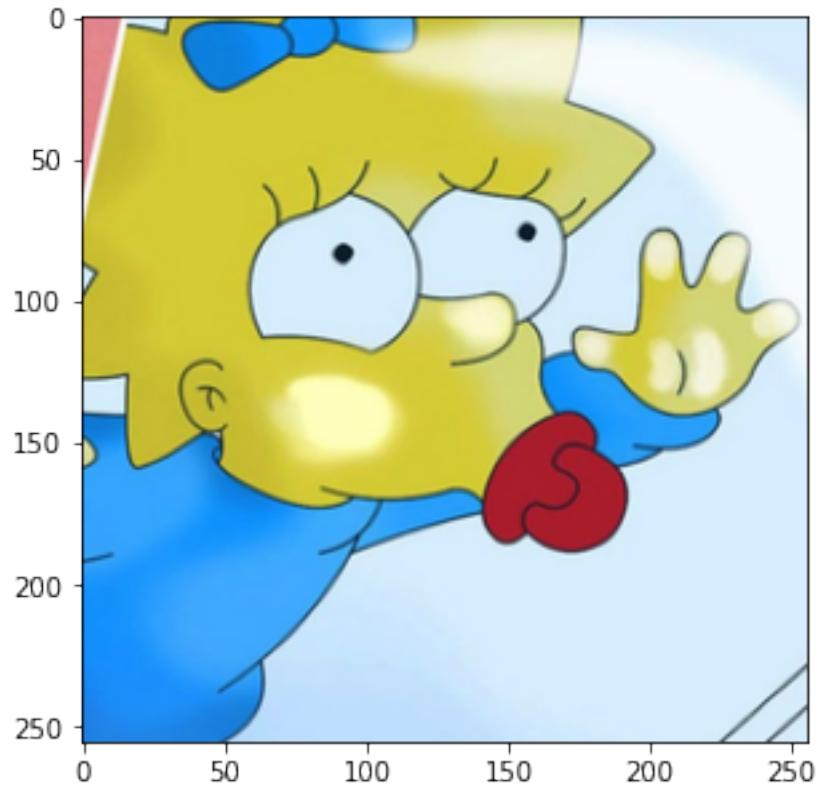
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2990>



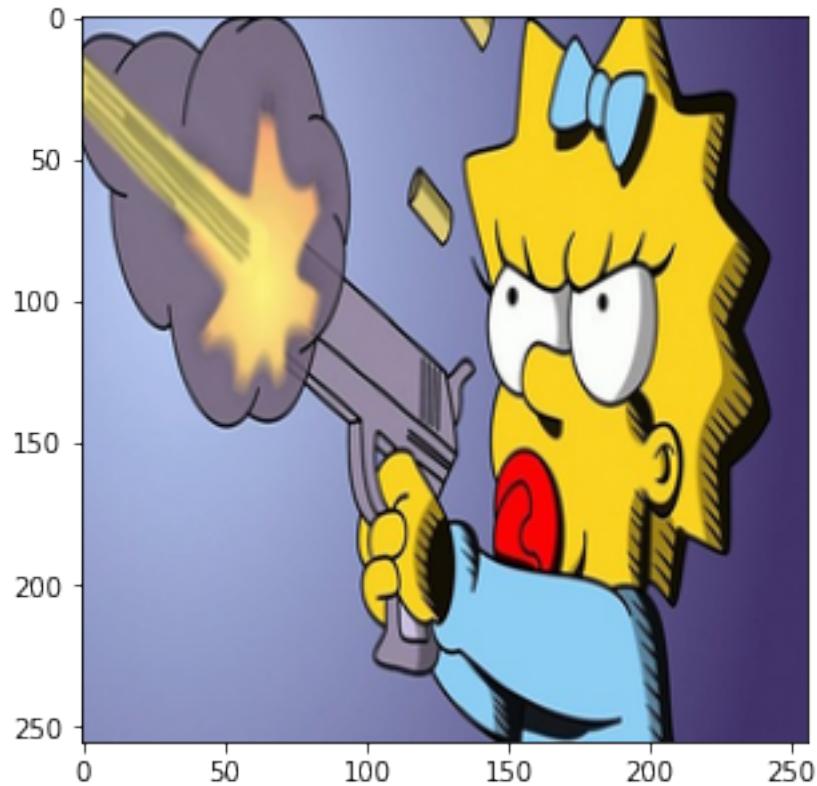
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B29D0>



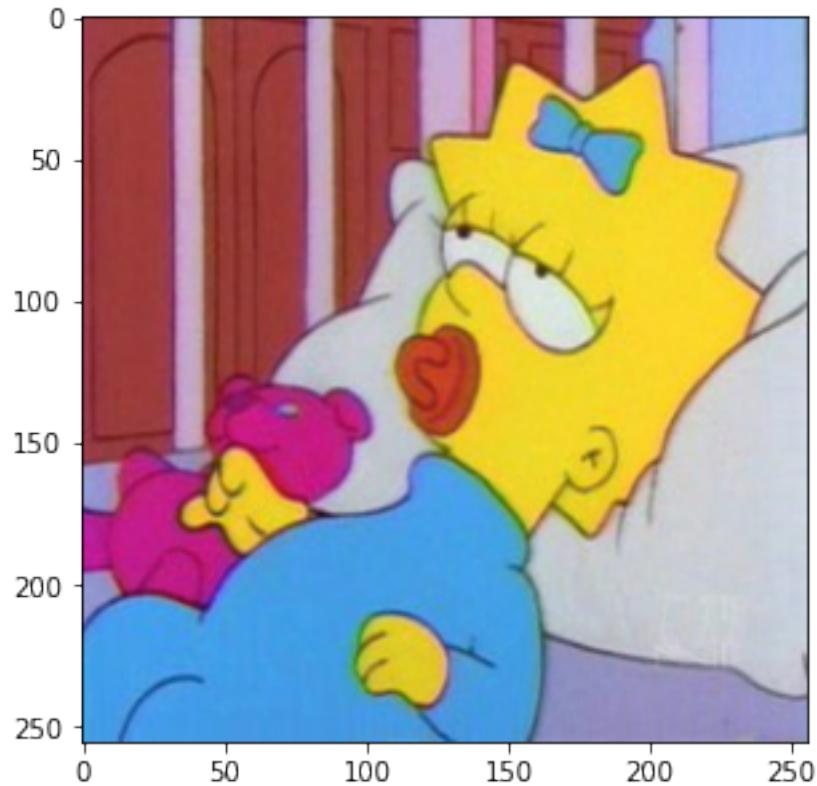
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2A50>



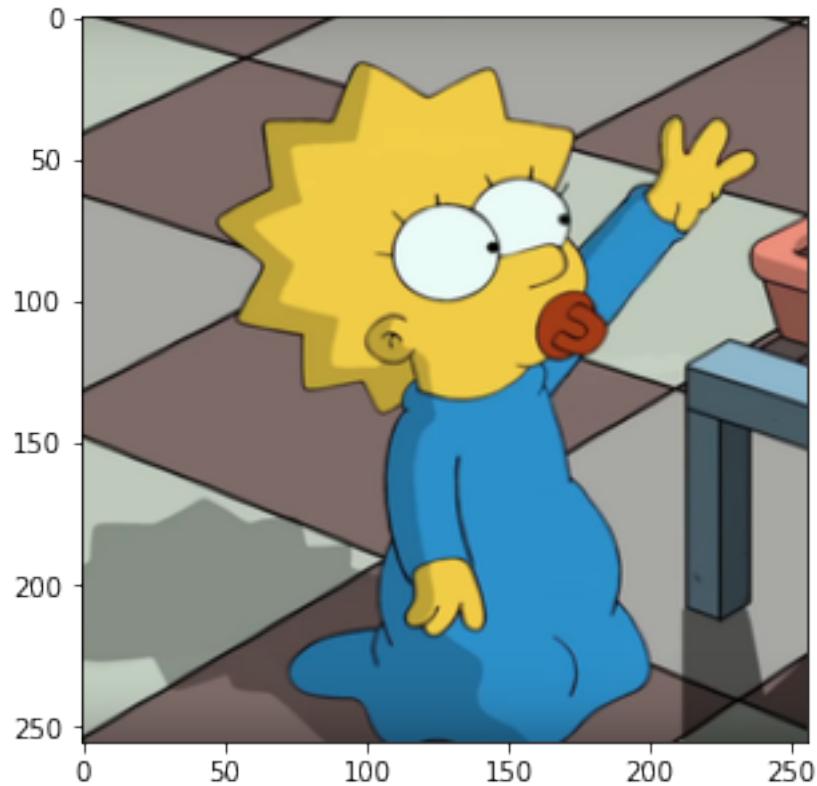
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2B50>



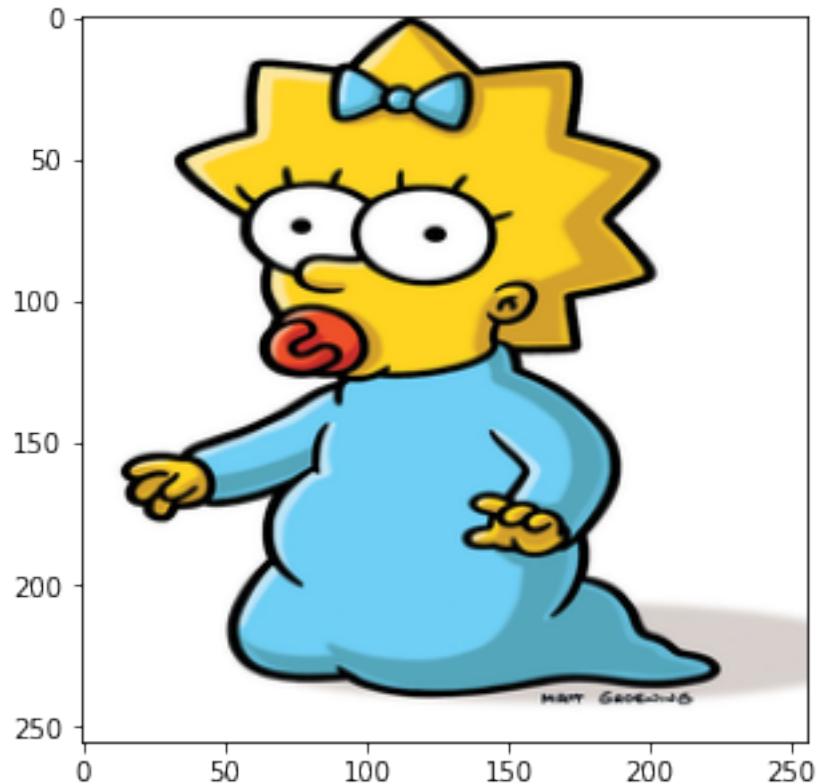
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2BD0>



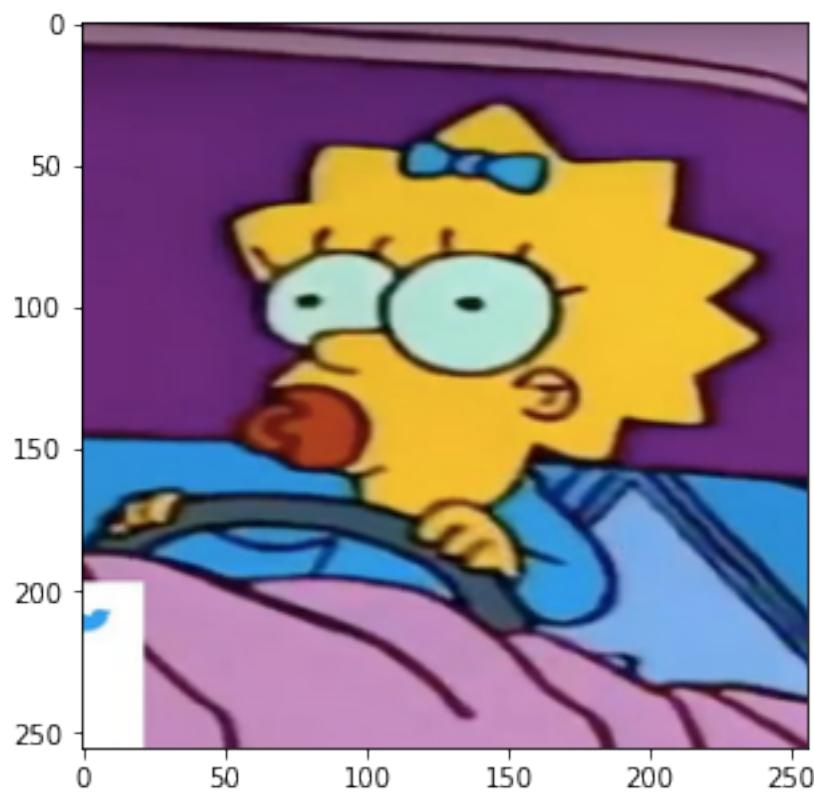
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB11B4B2C90>



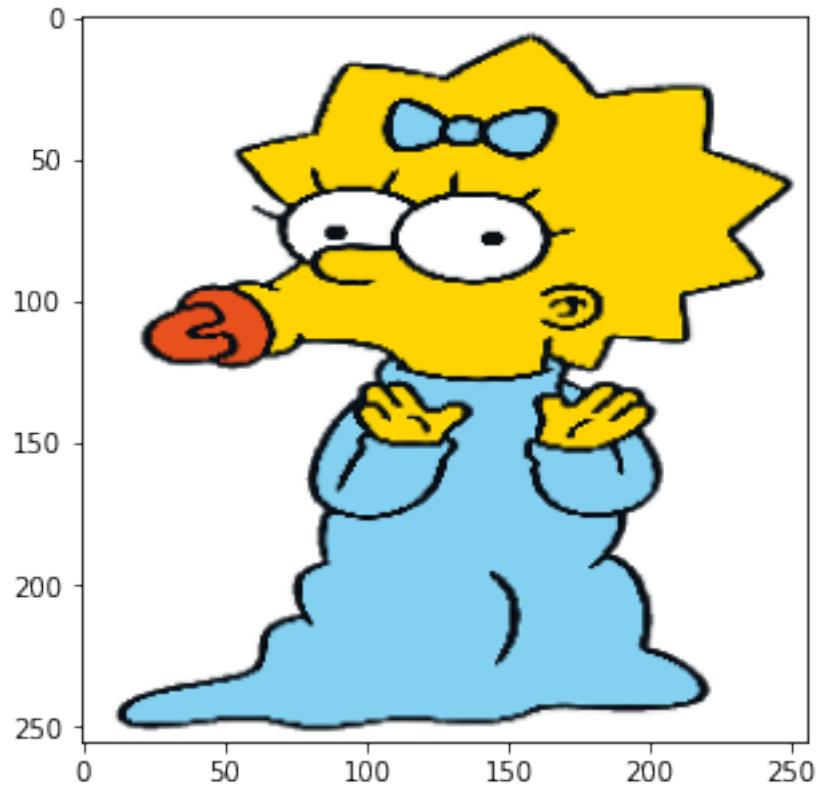
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2C50>



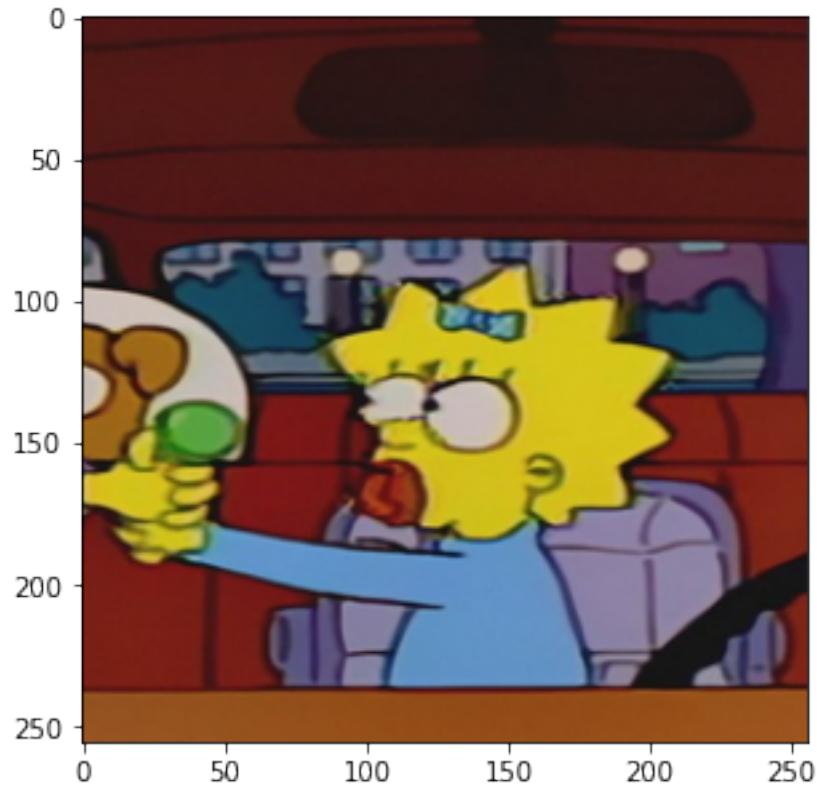
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2CD0>



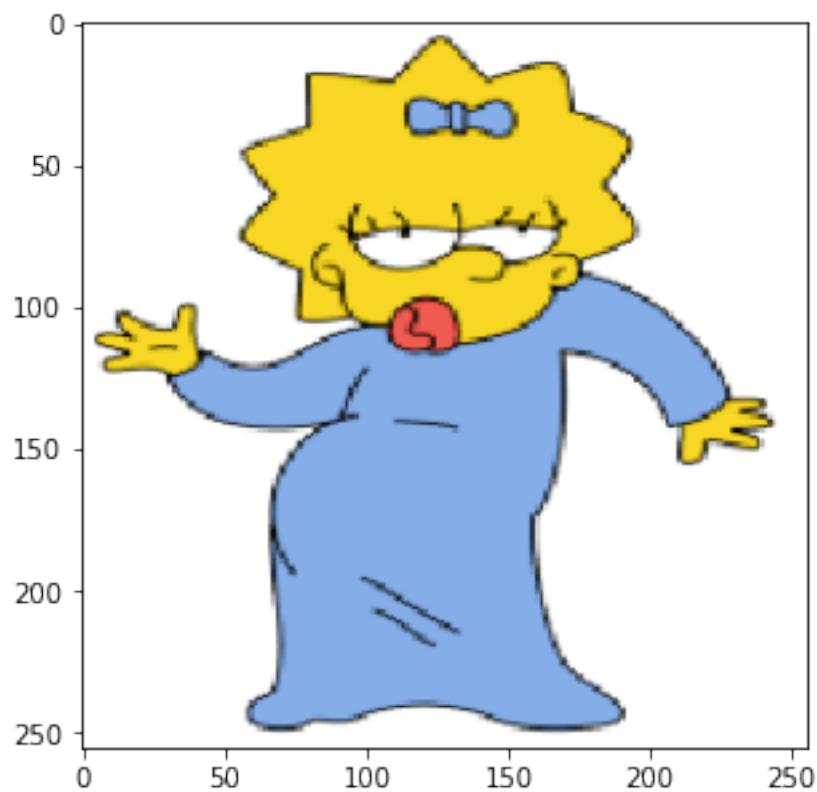
<PIL.Image.Image image mode=P size=256x256 at 0x7FB11B4B2D50>



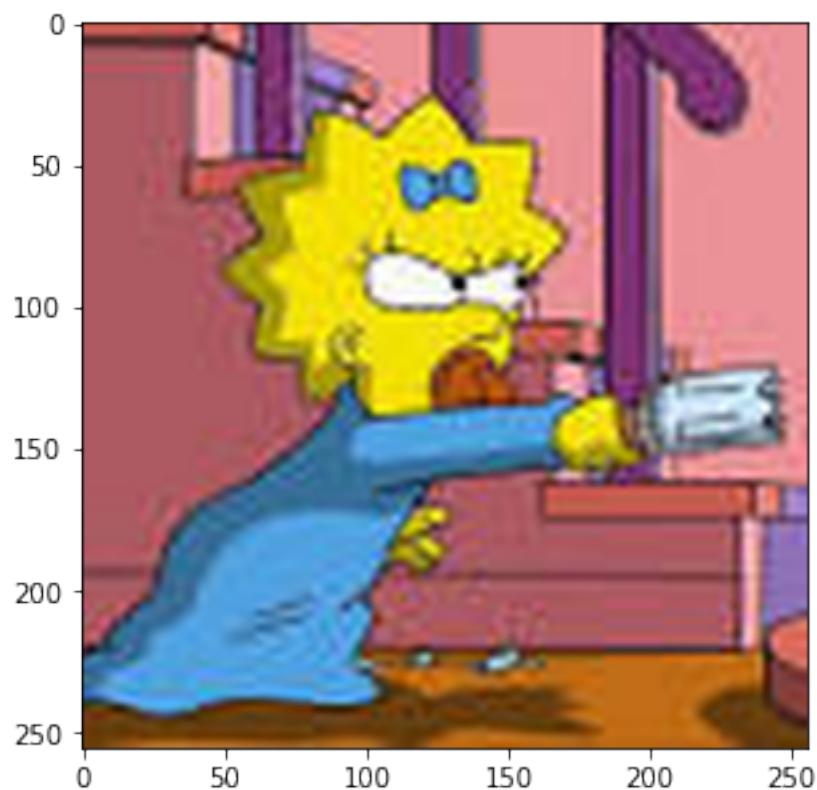
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2E90>



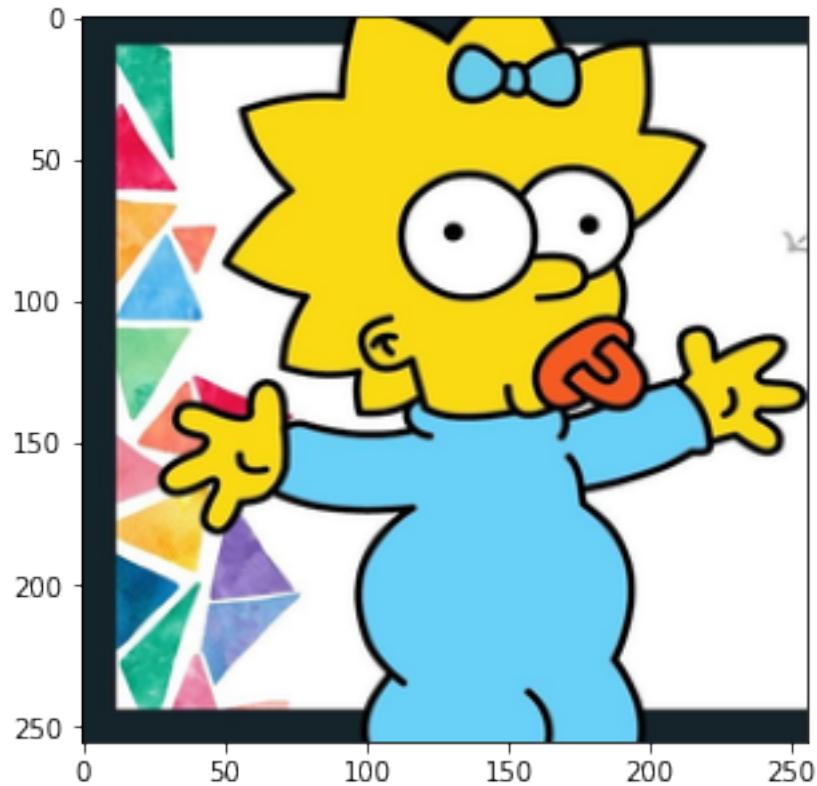
<PIL.Image.Image image mode=P size=256x256 at 0x7FB11B4B2ED0>



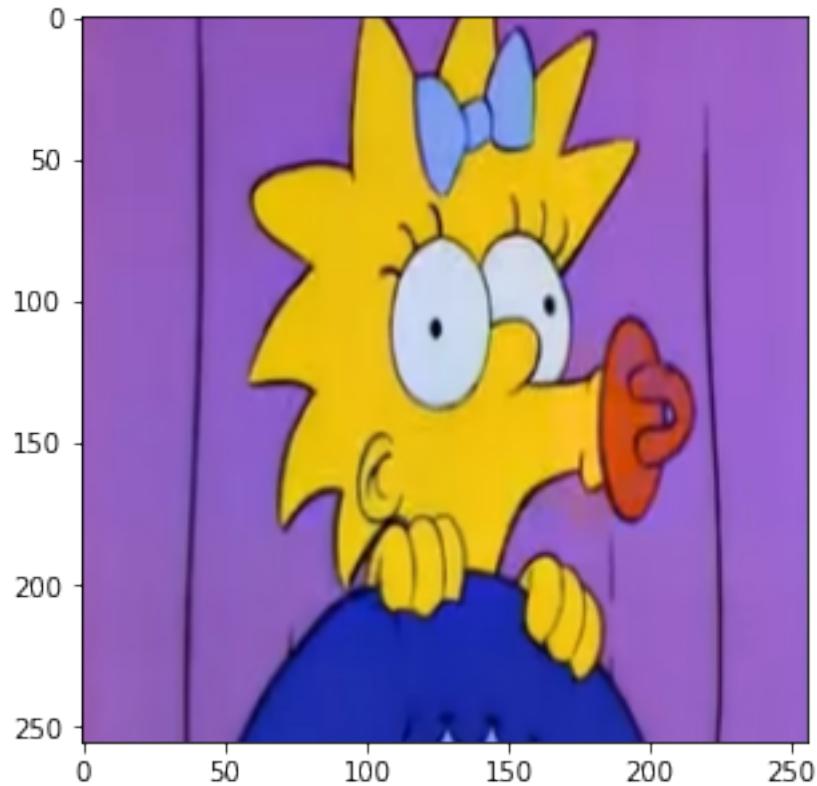
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2F90>



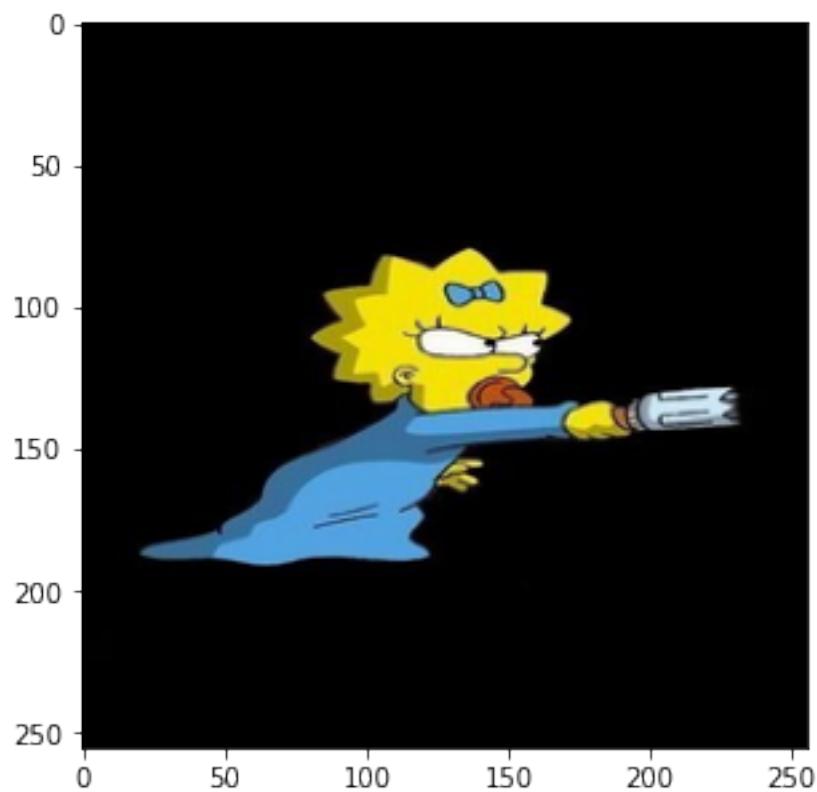
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB11B4B2FD0>



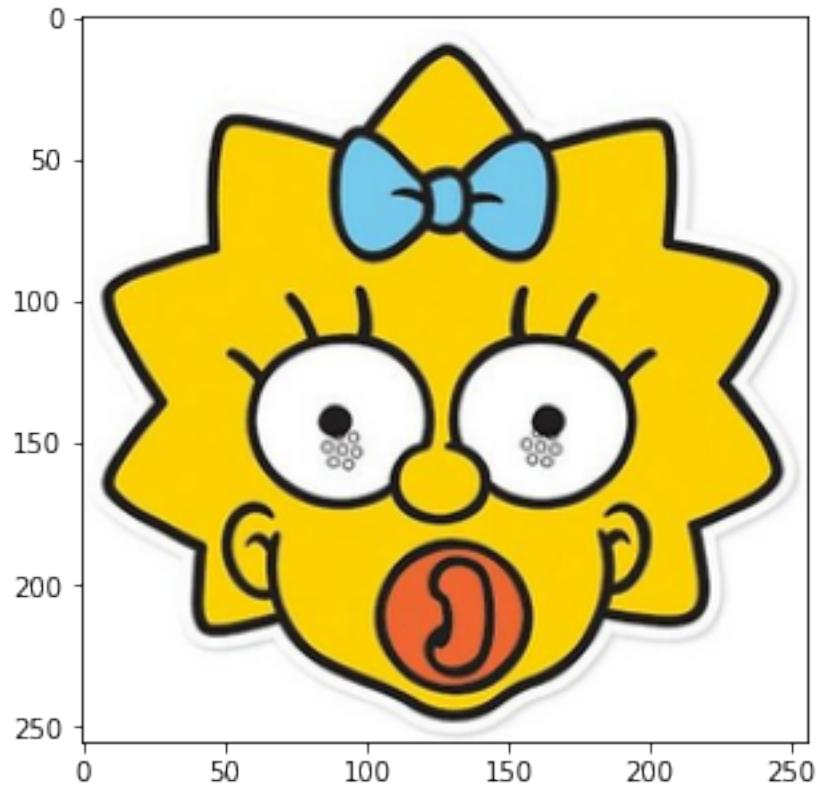
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0090>



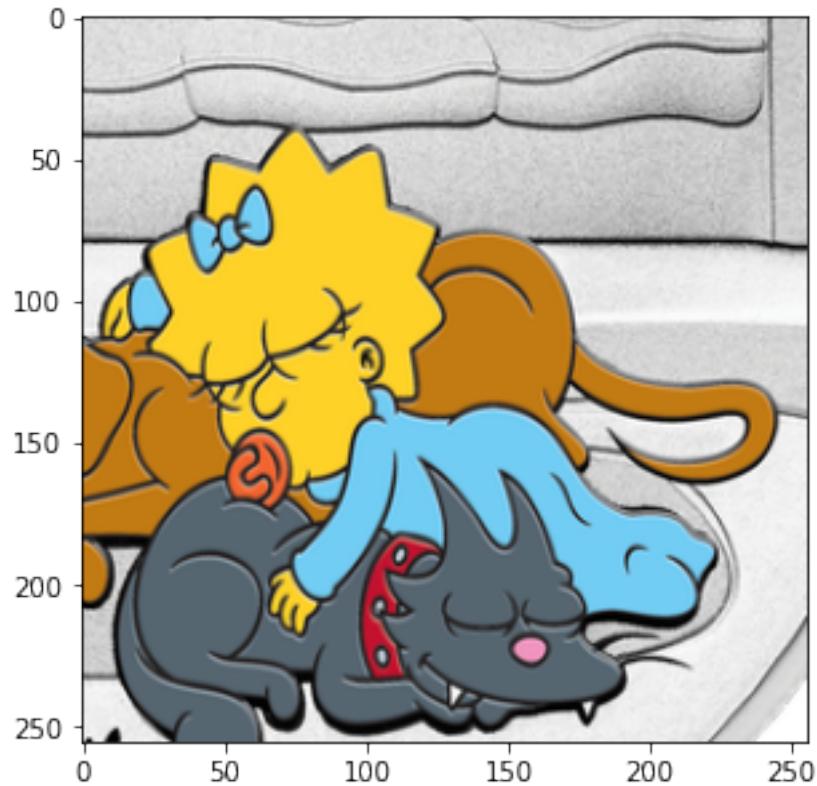
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0110>



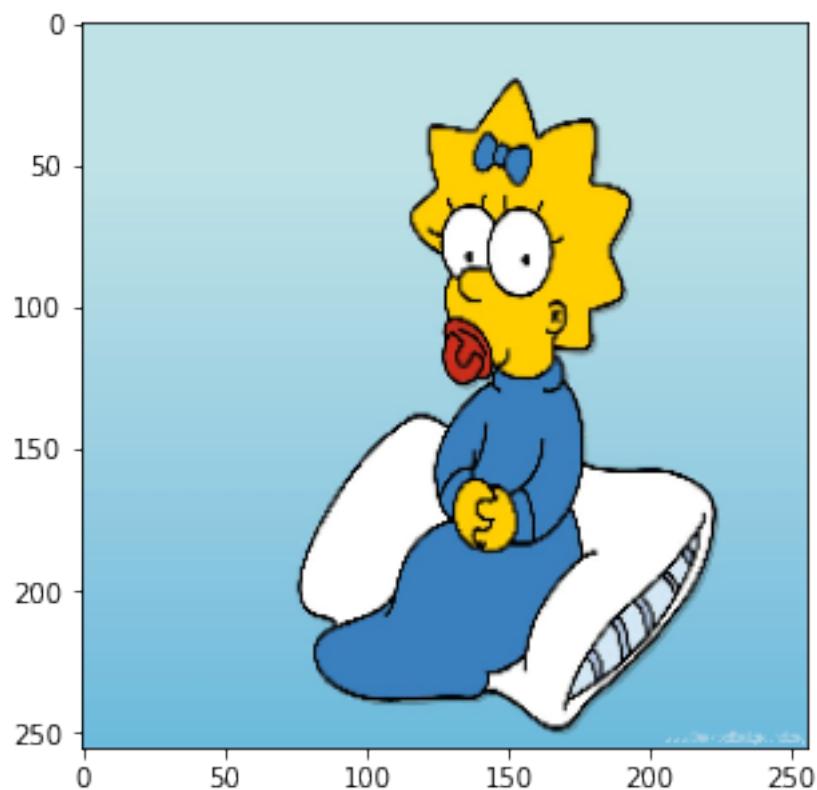
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB119479510>



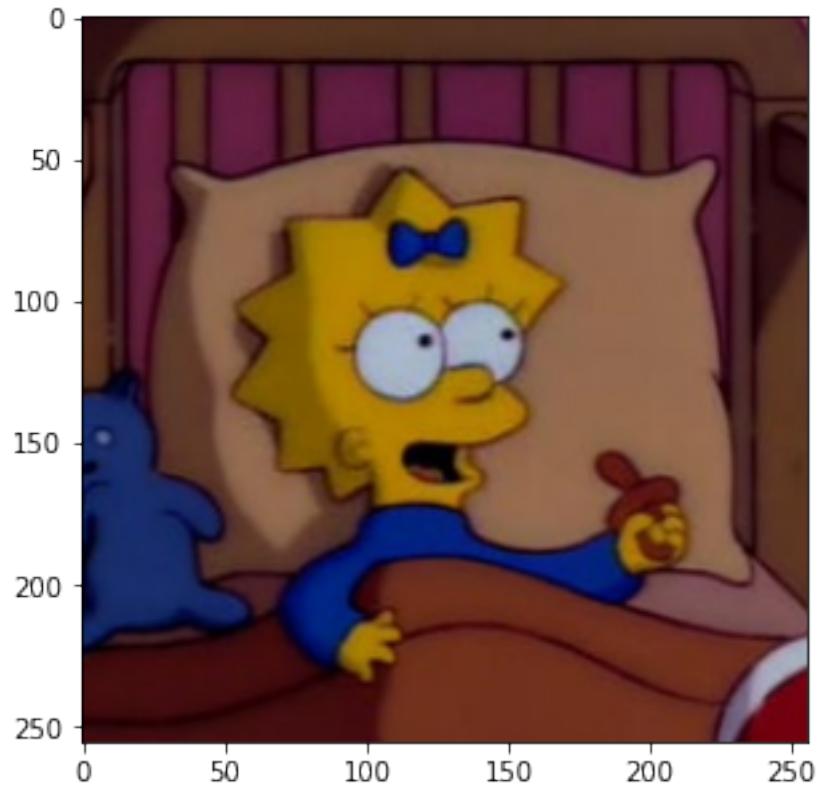
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191A0190>



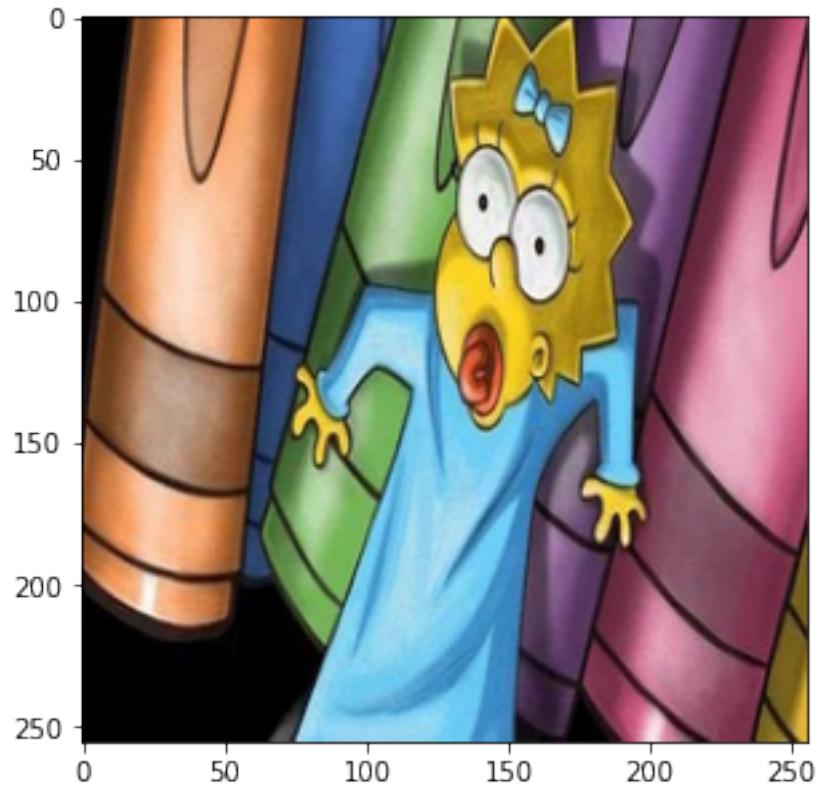
<PIL.Image.Image image mode=P size=256x256 at 0x7FB119C29990>



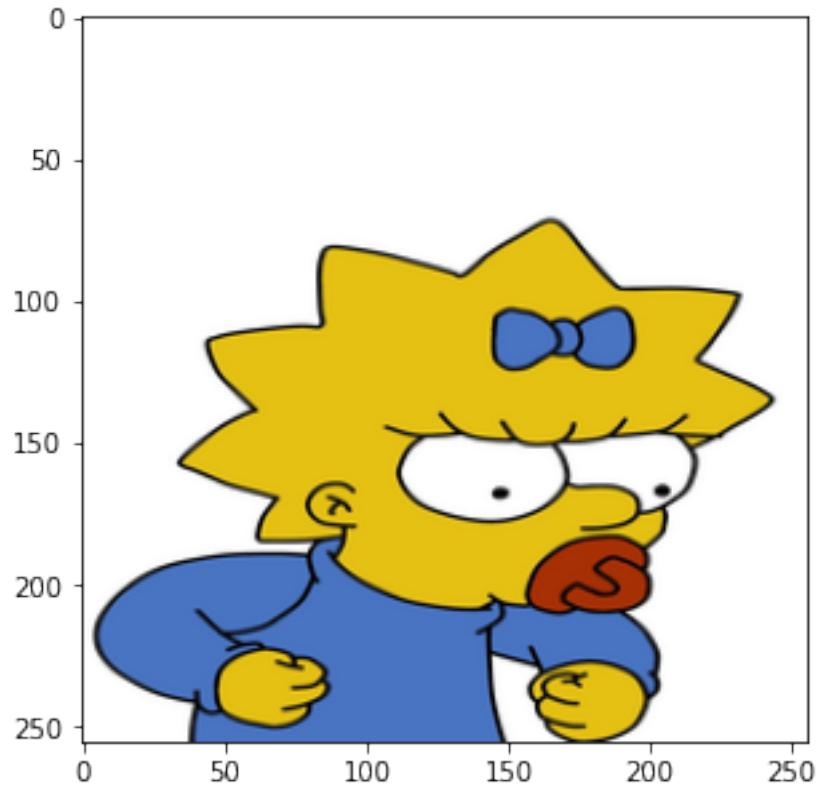
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0390>



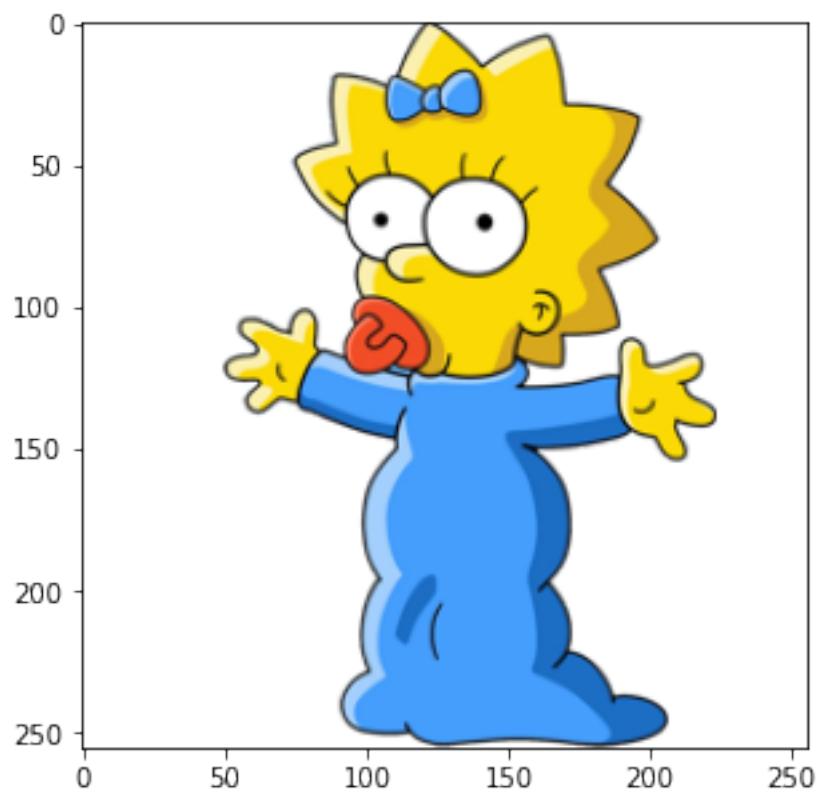
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0310>



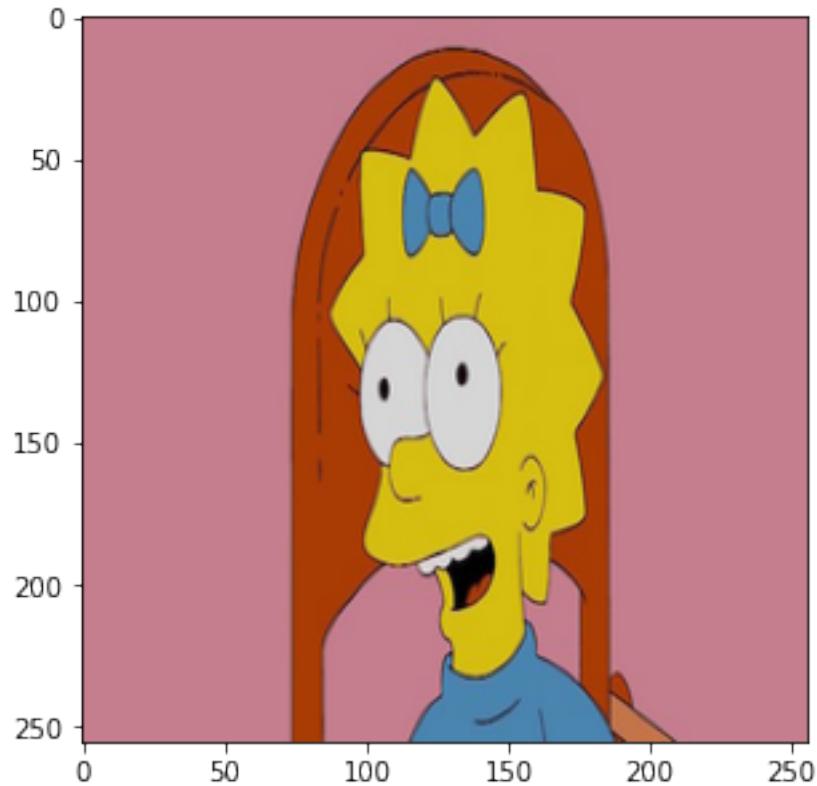
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0410>



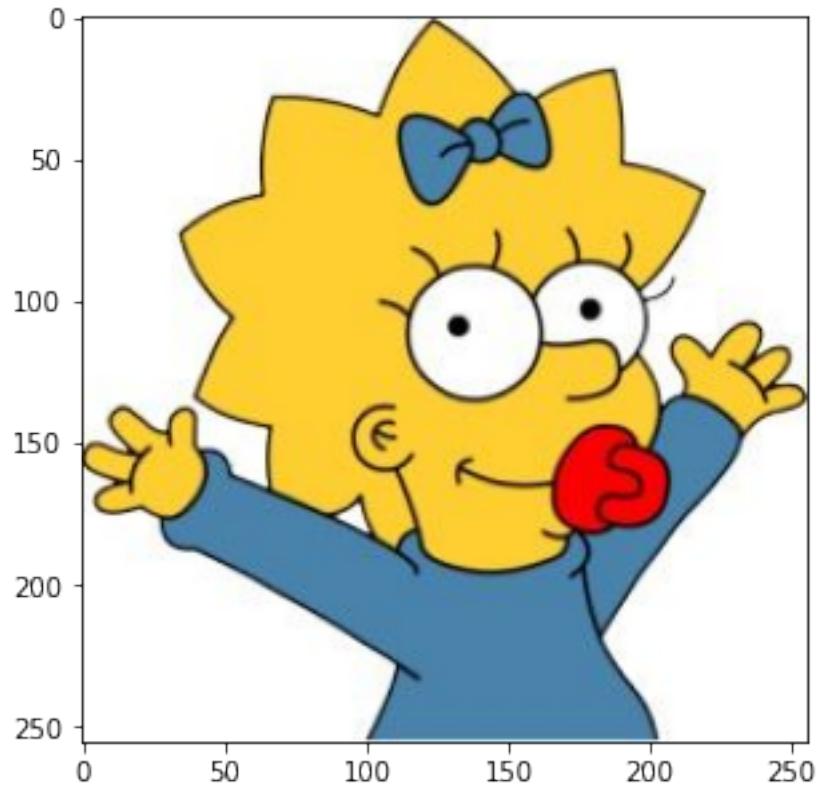
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191A0490>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0590>



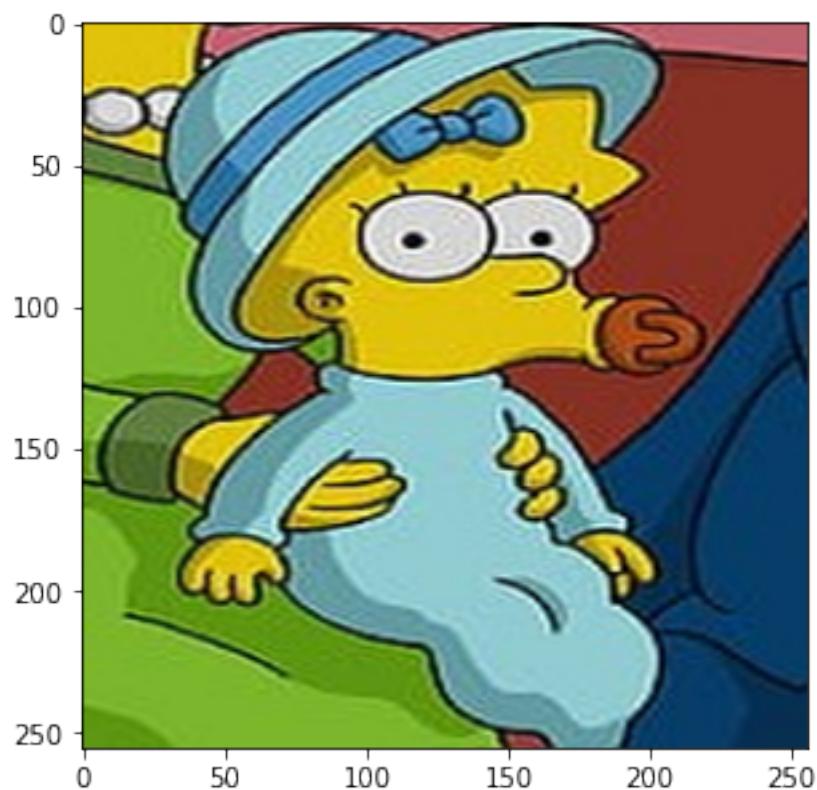
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0610>



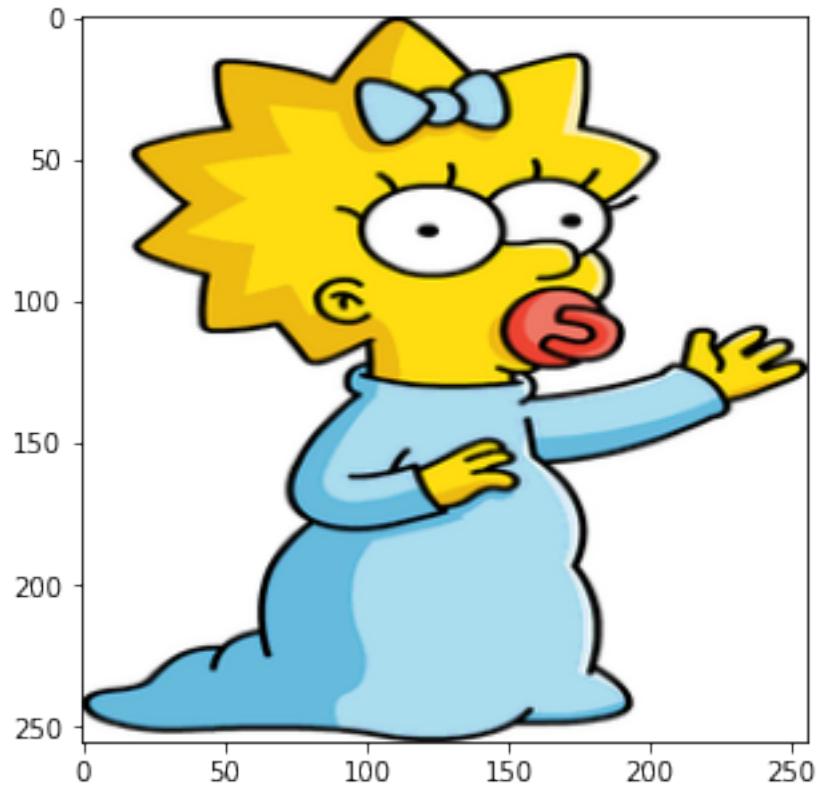
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0690>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0710>

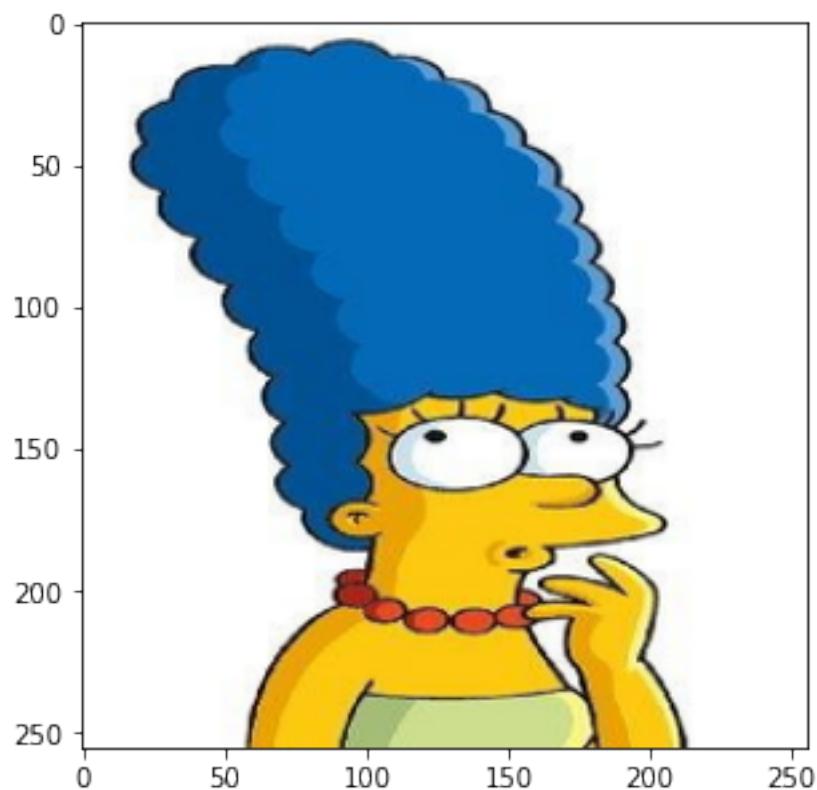


<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191A0810>

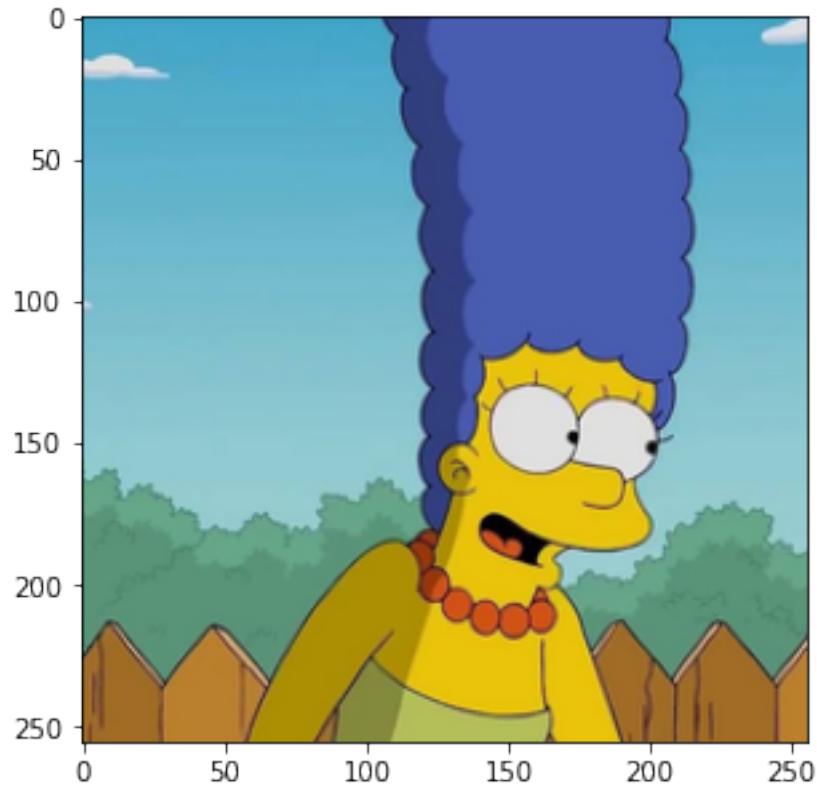


Marge -- Resized

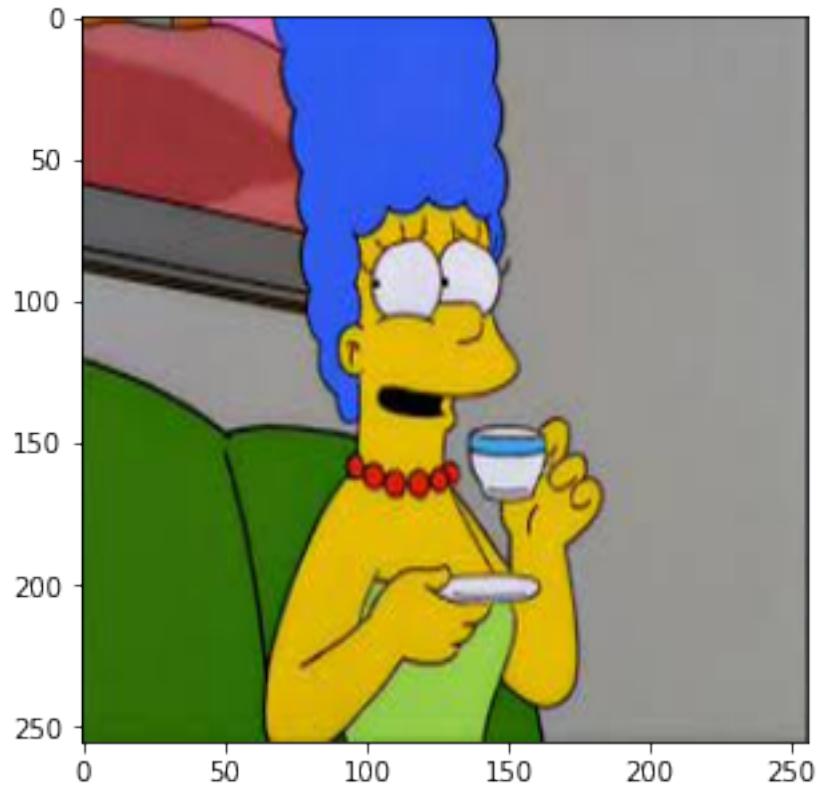
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0790>



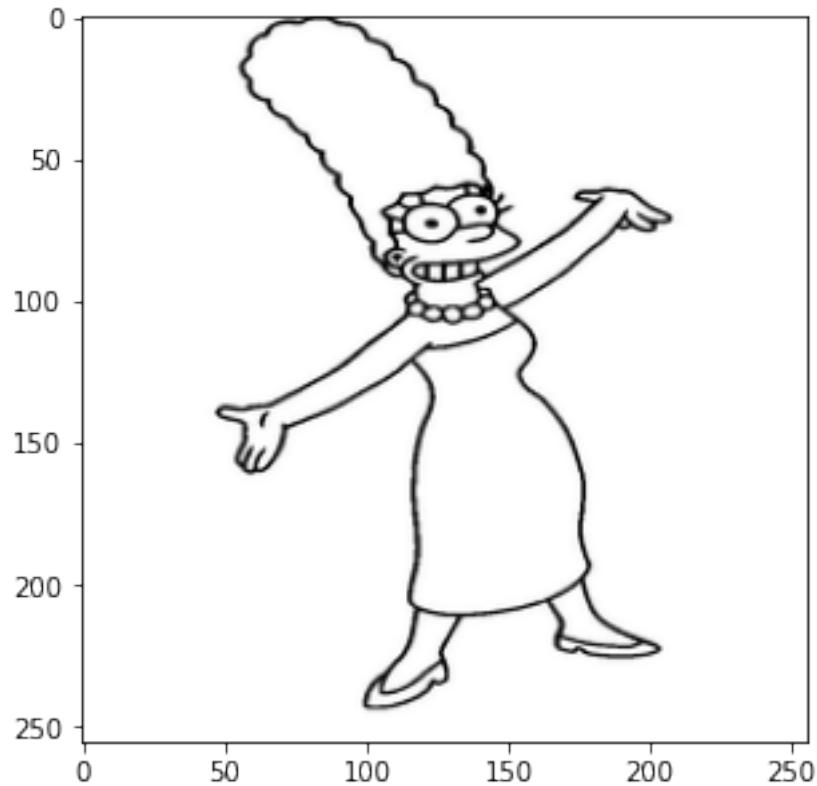
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0850>



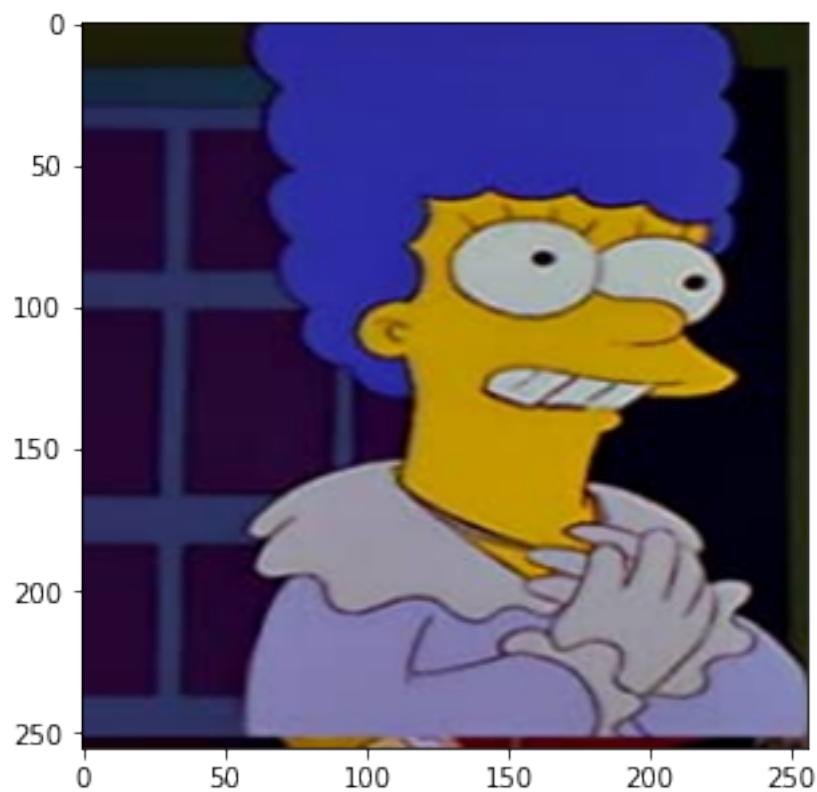
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0890>



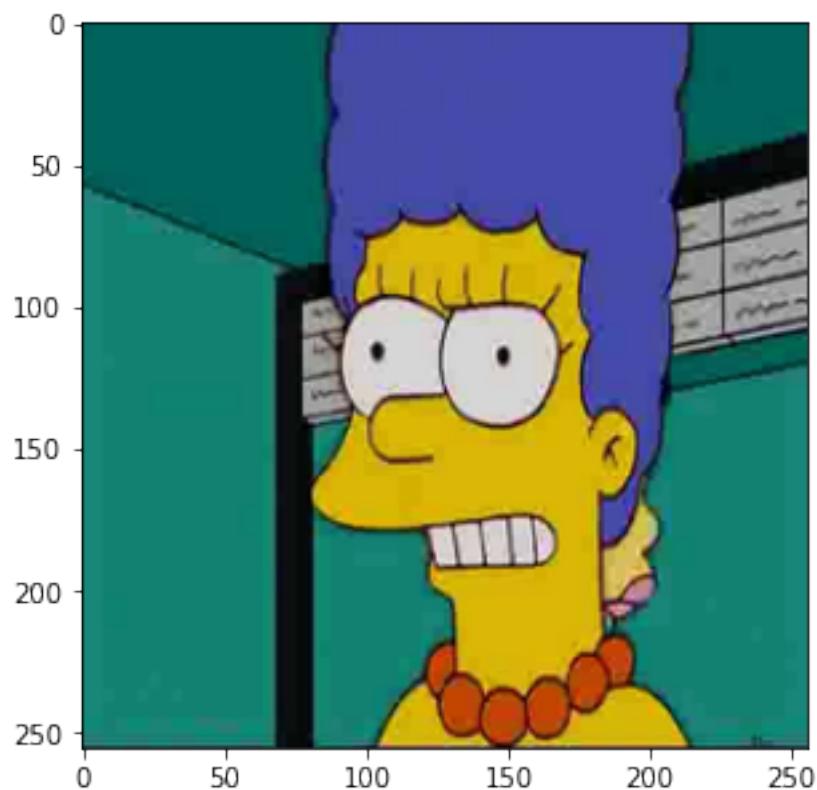
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0990>



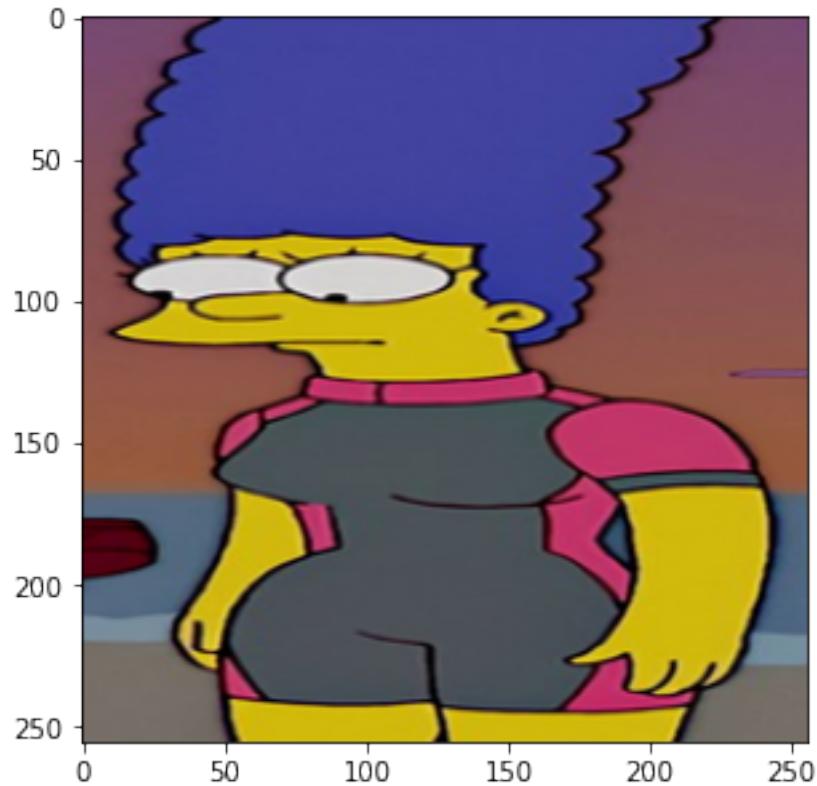
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0A10>



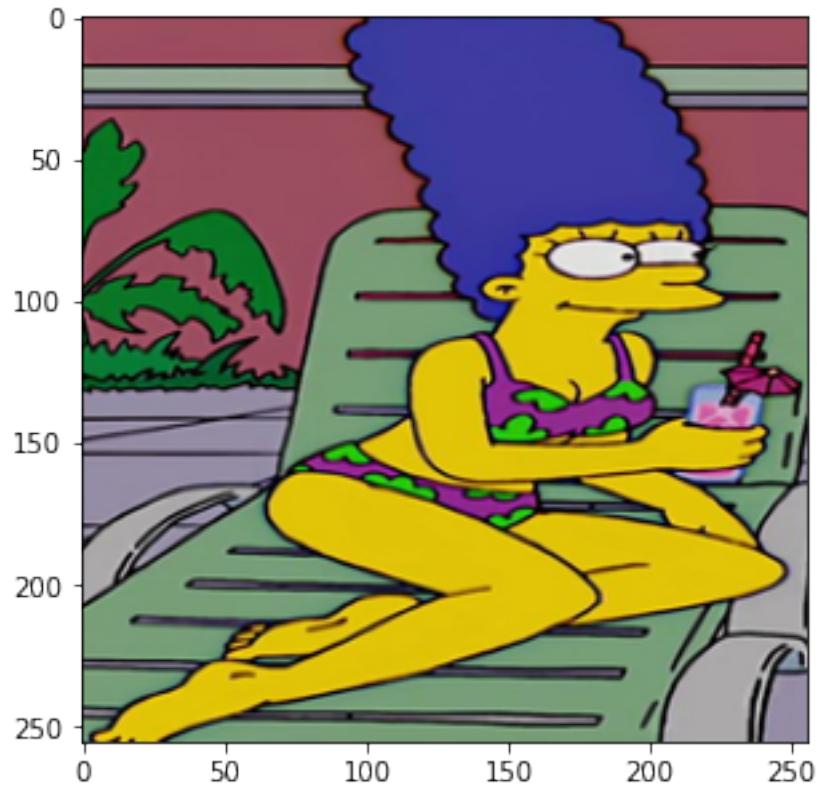
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0B10>



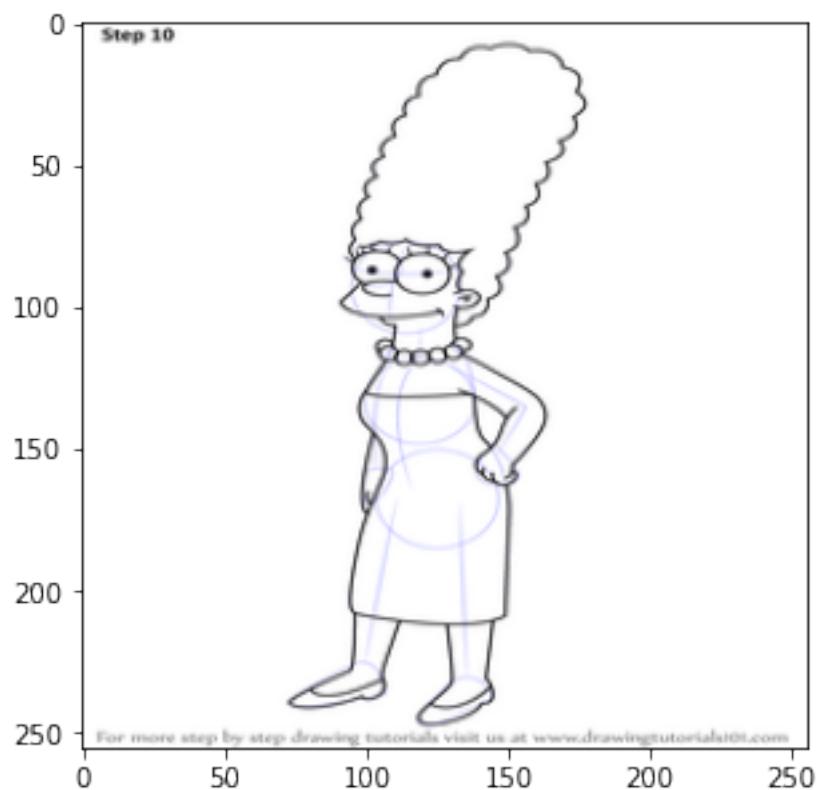
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191A0B90>



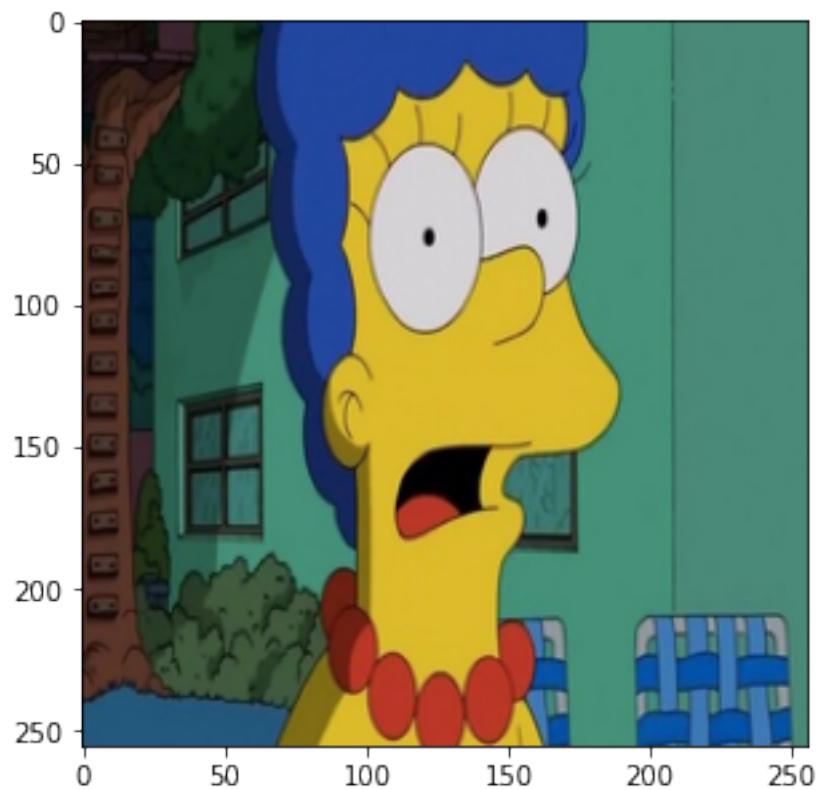
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191A0C10>



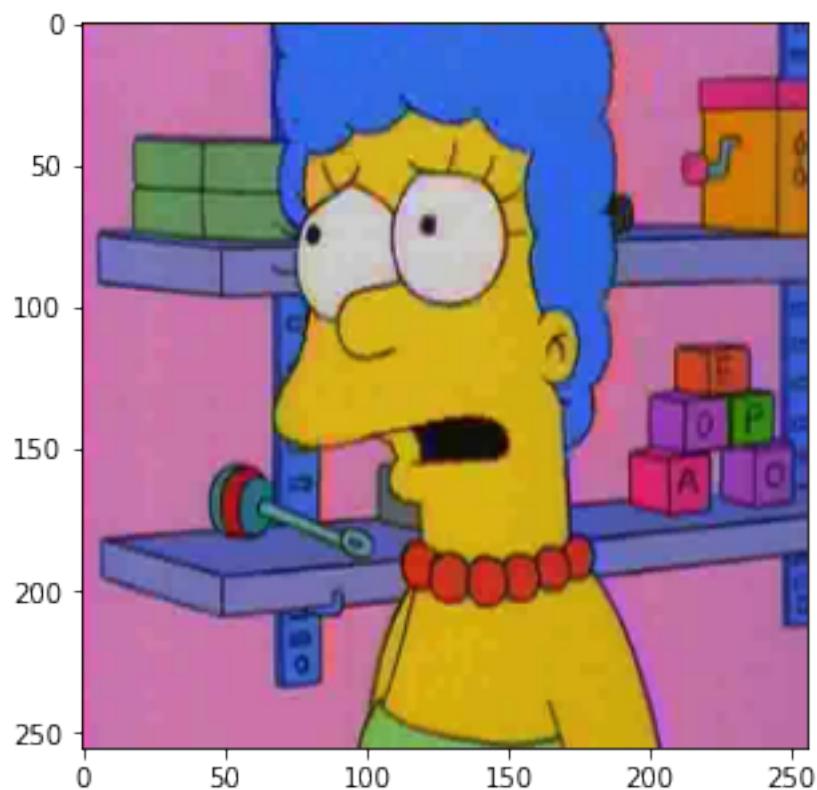
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0AD0>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0A90>



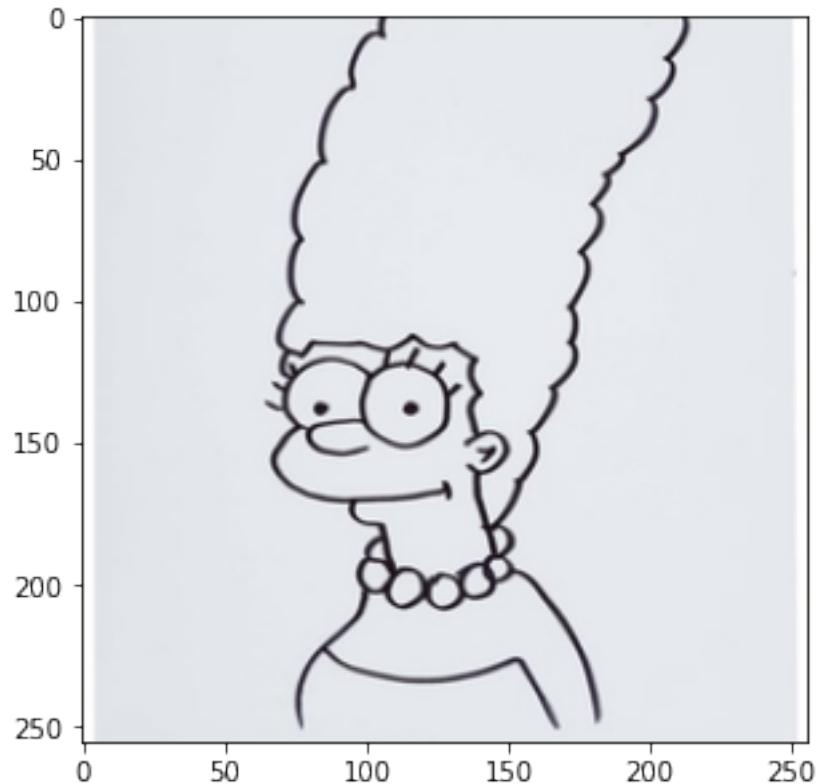
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0D10>



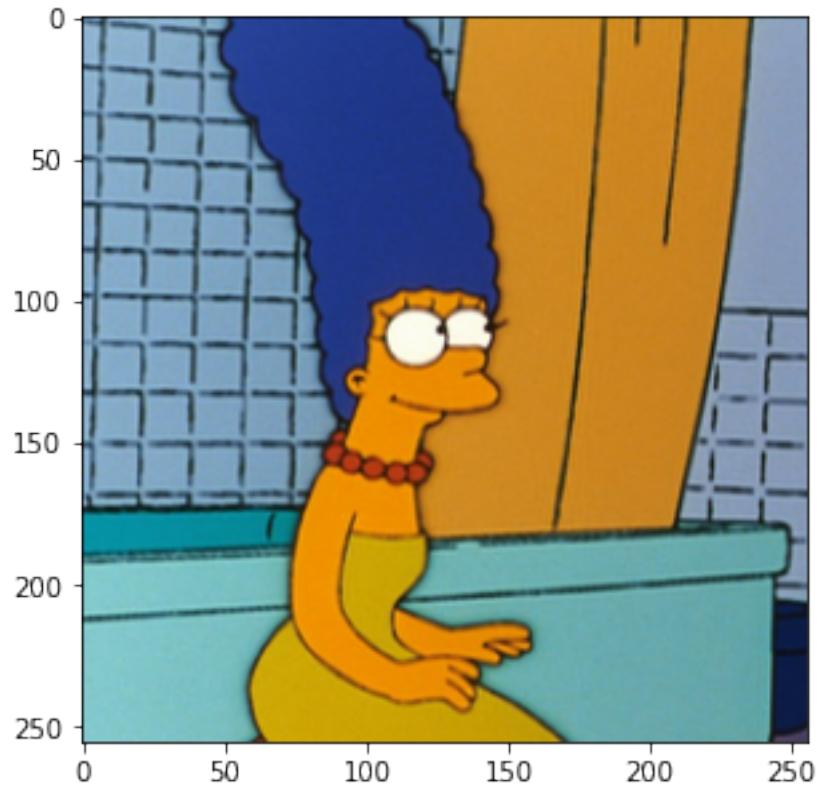
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0DD0>



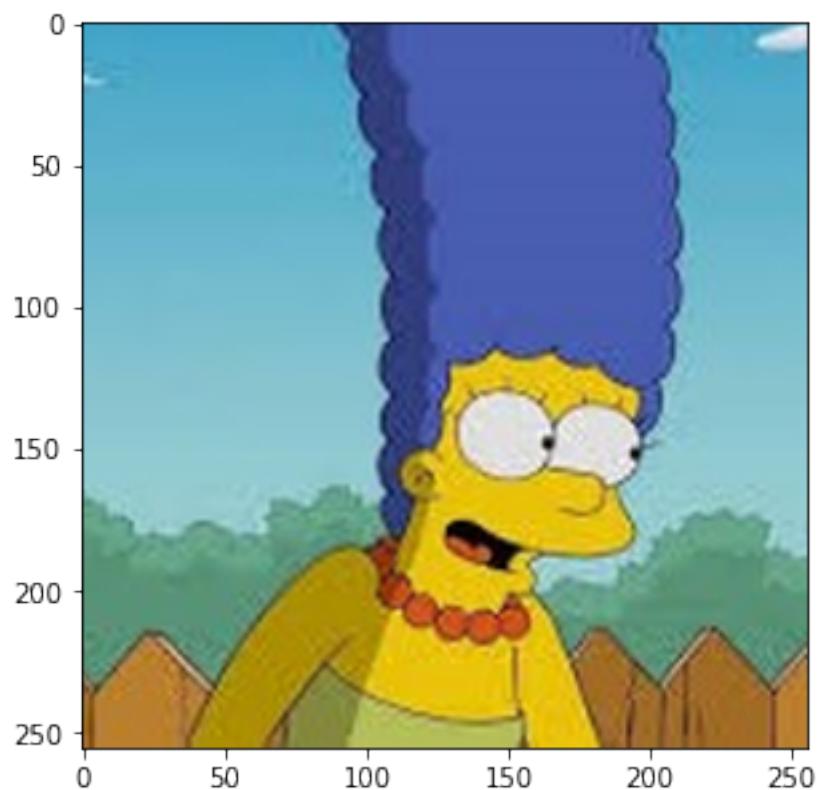
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0D90>



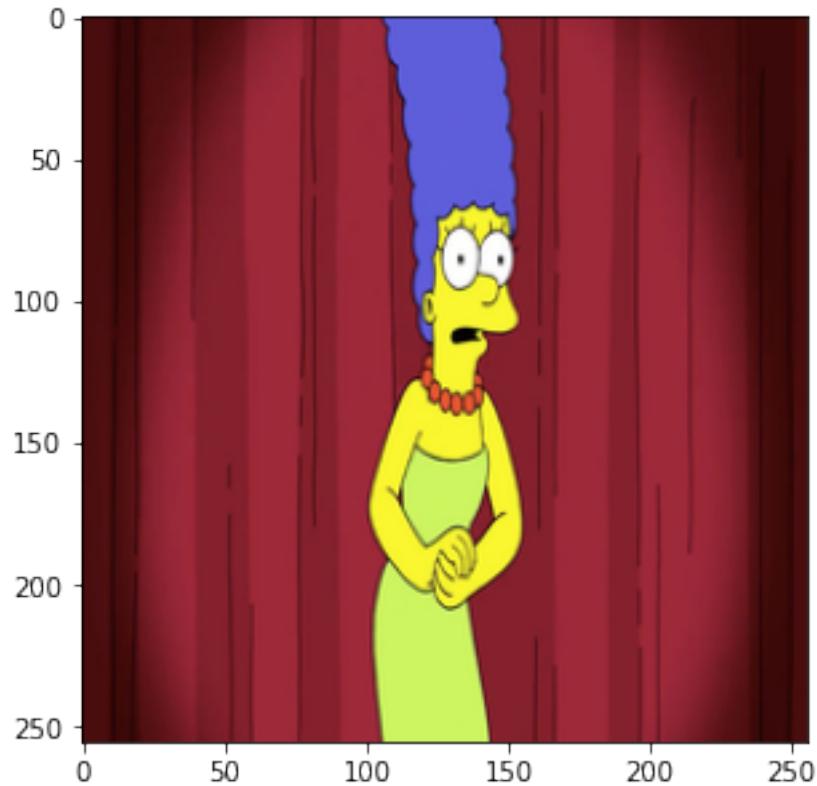
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0E90>



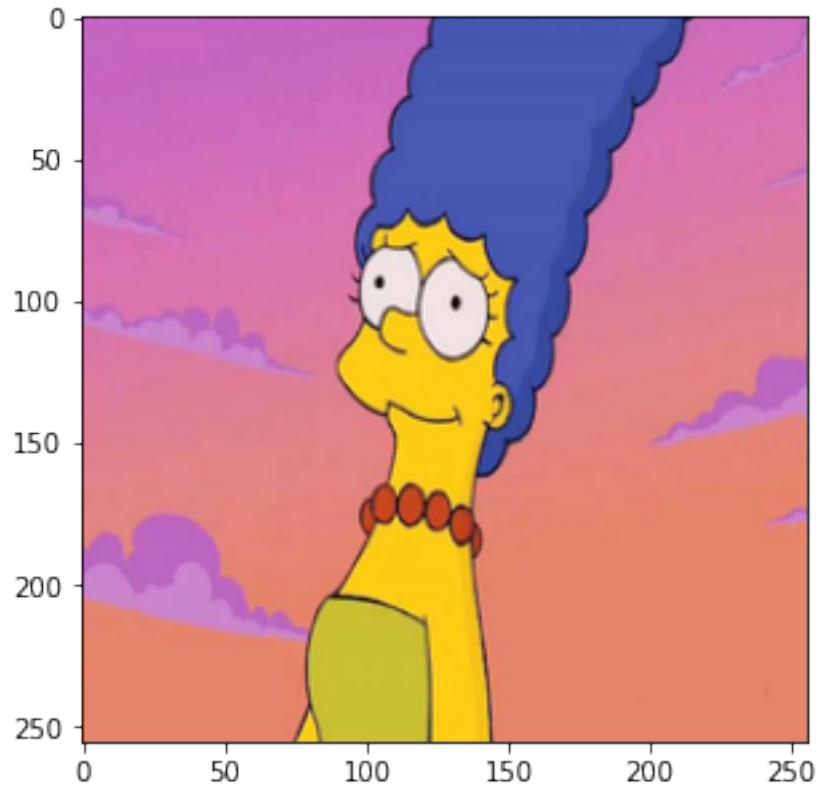
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0F10>



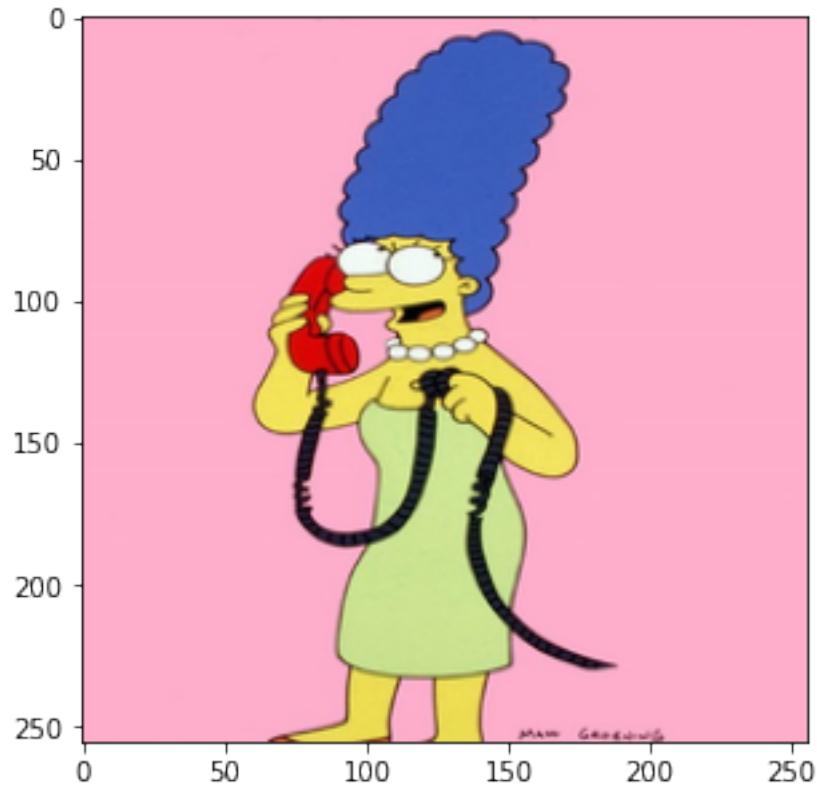
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0F90>



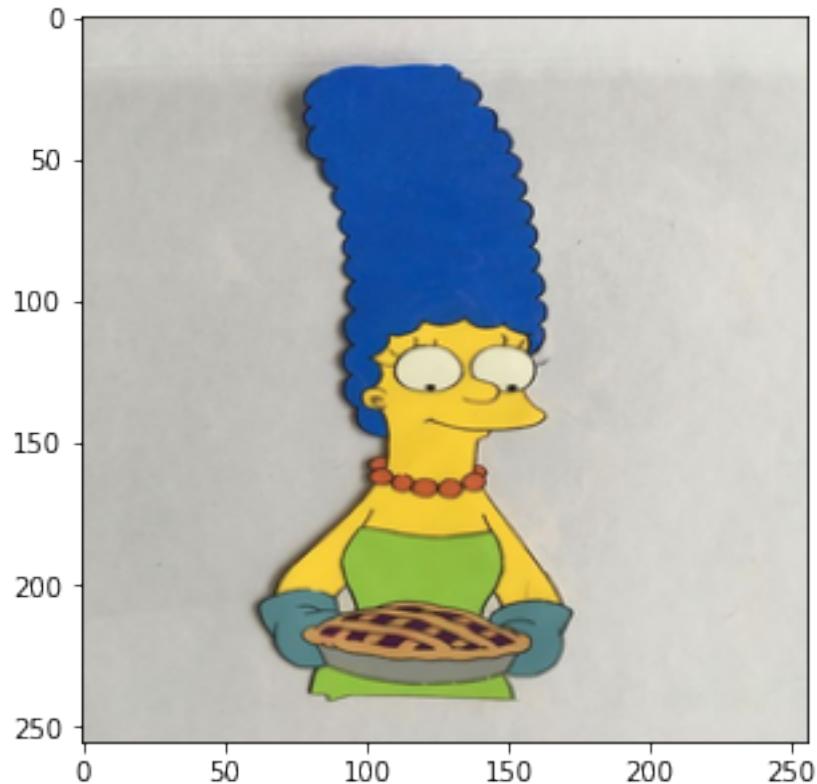
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0550>



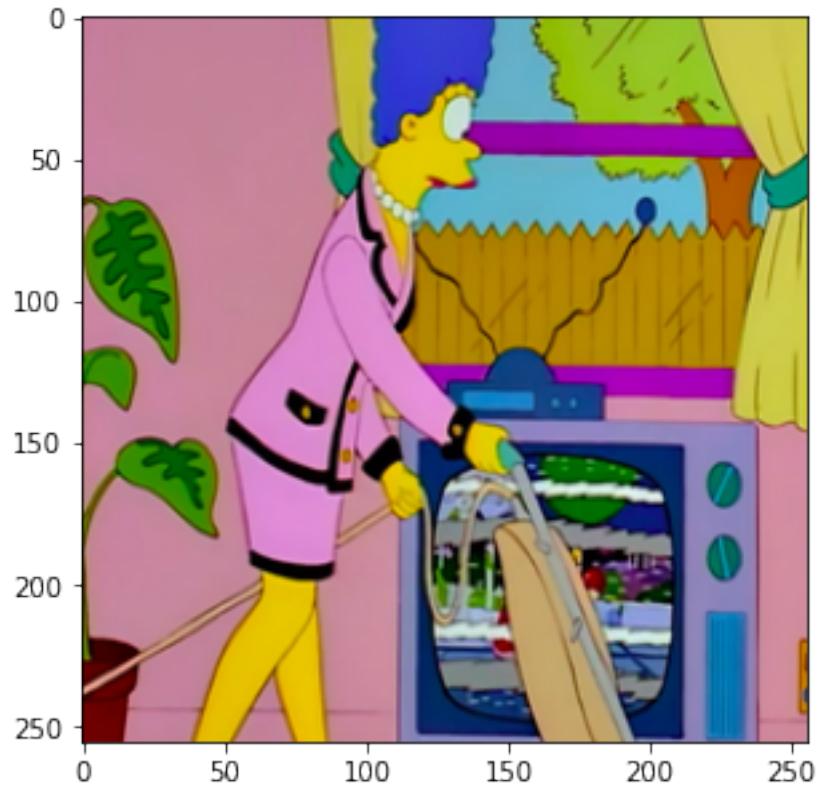
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0290>



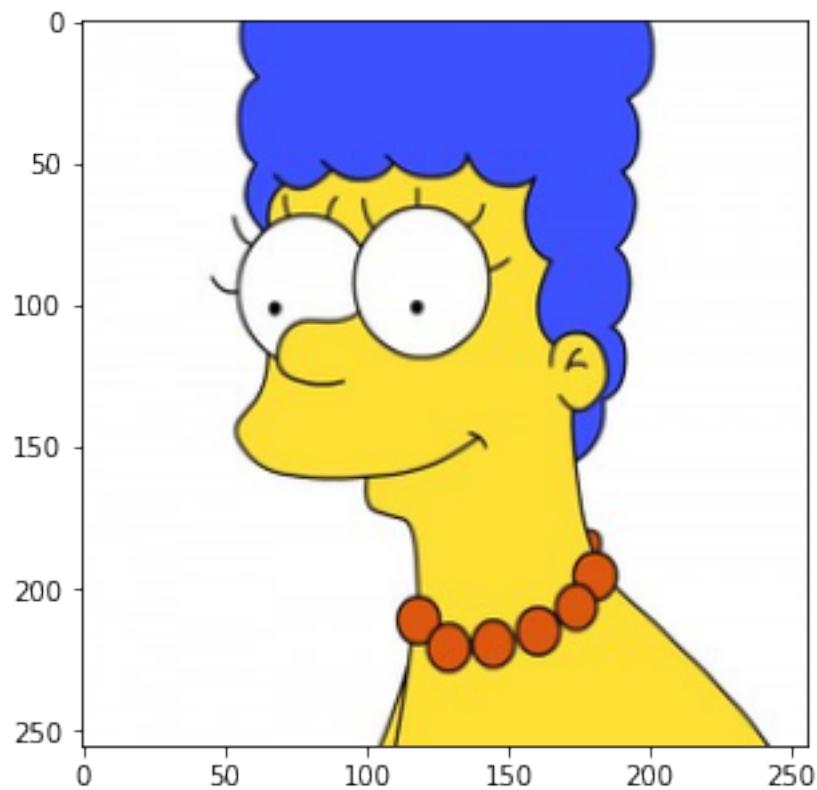
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE050>



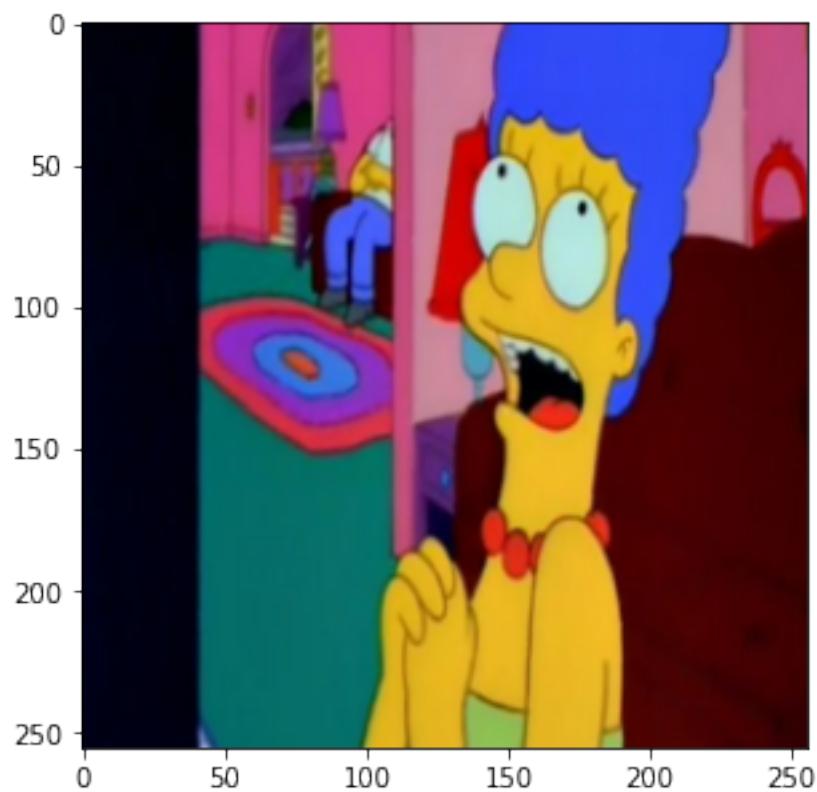
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191AE110>



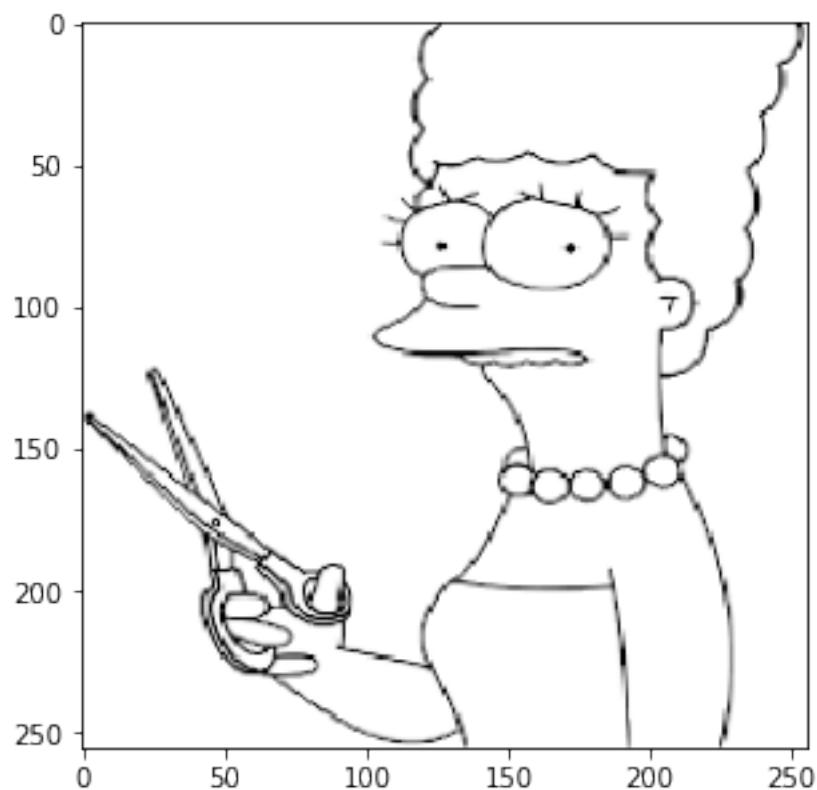
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE0D0>



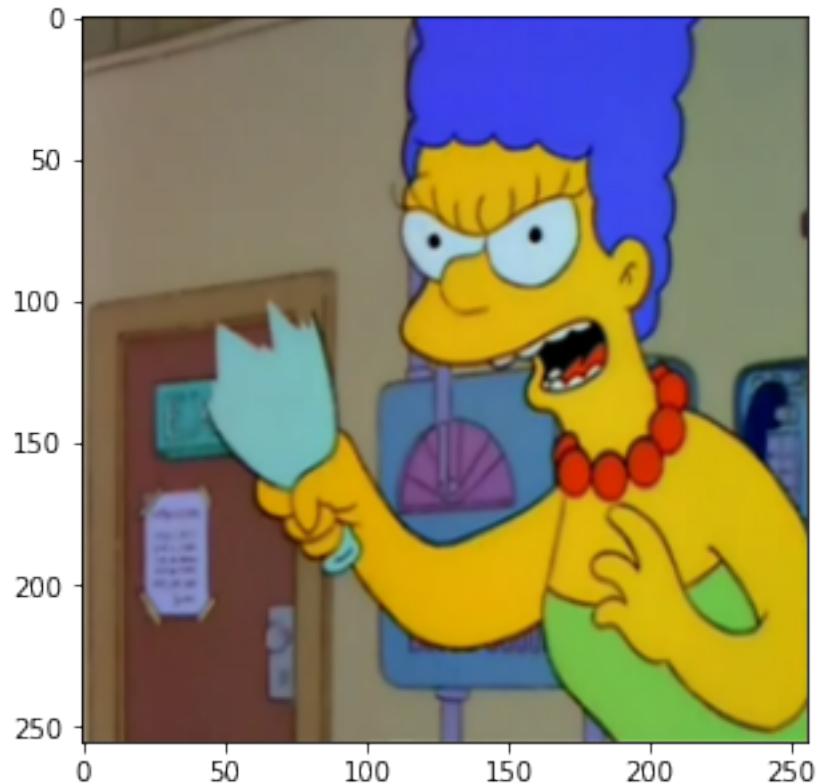
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE150>



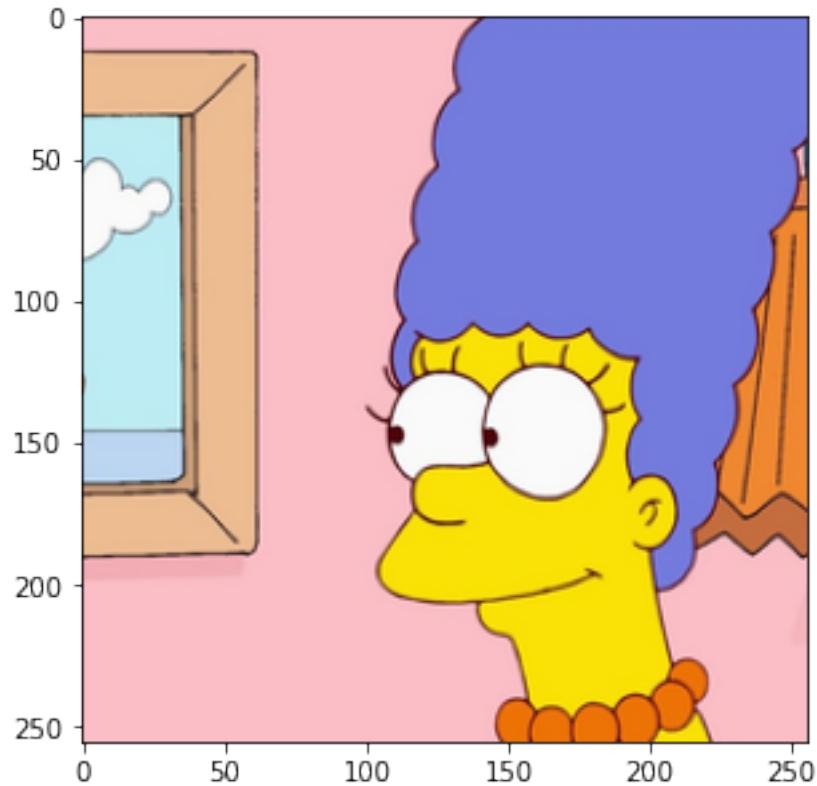
<PIL.Image.Image image mode=P size=256x256 at 0x7FB1191AE210>



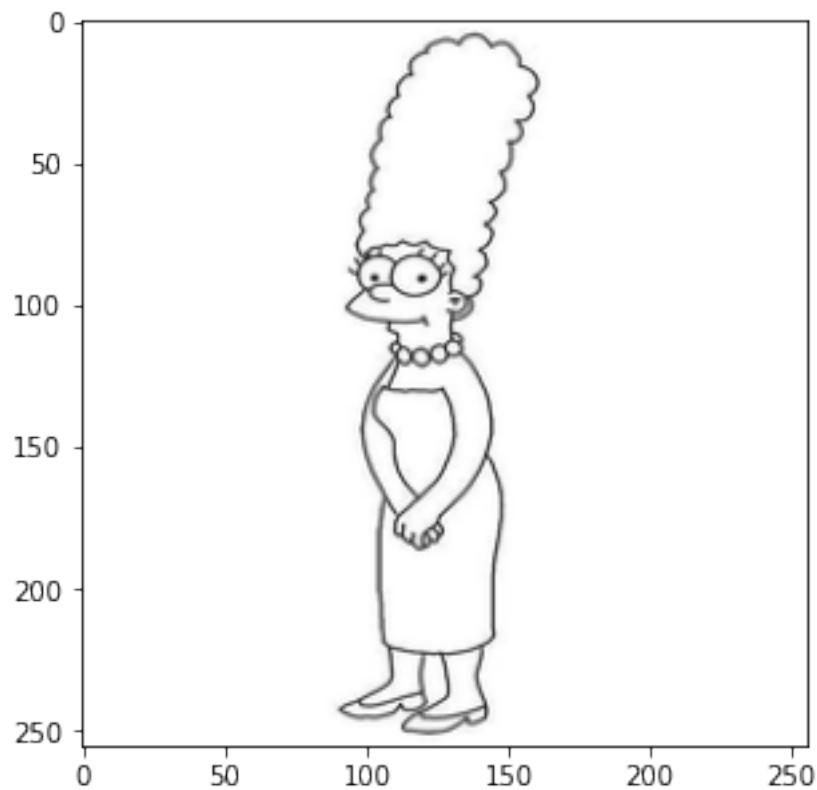
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191AE2D0>



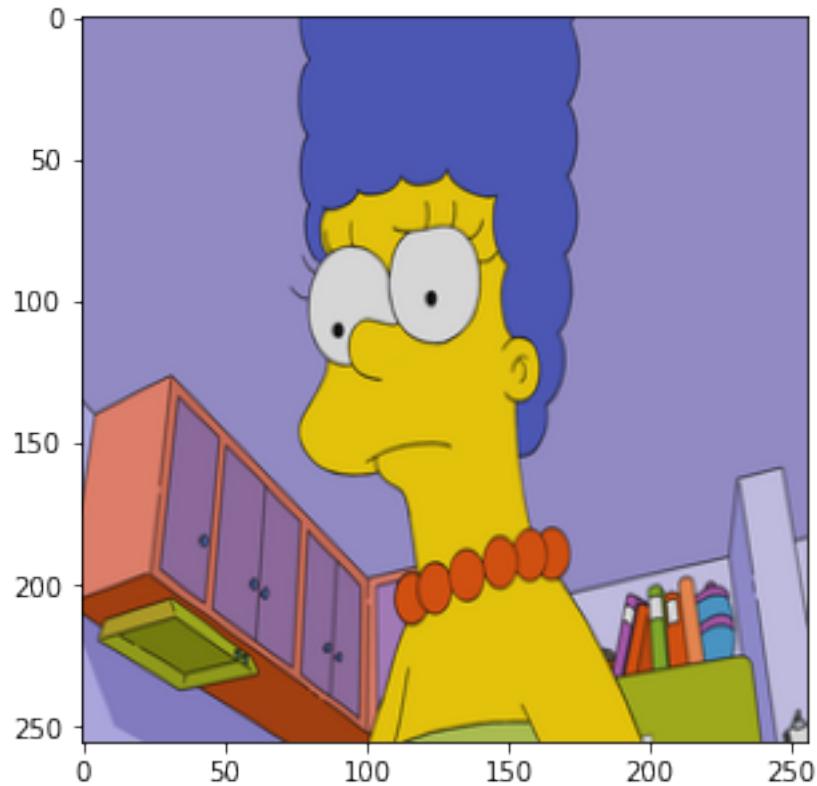
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE310>



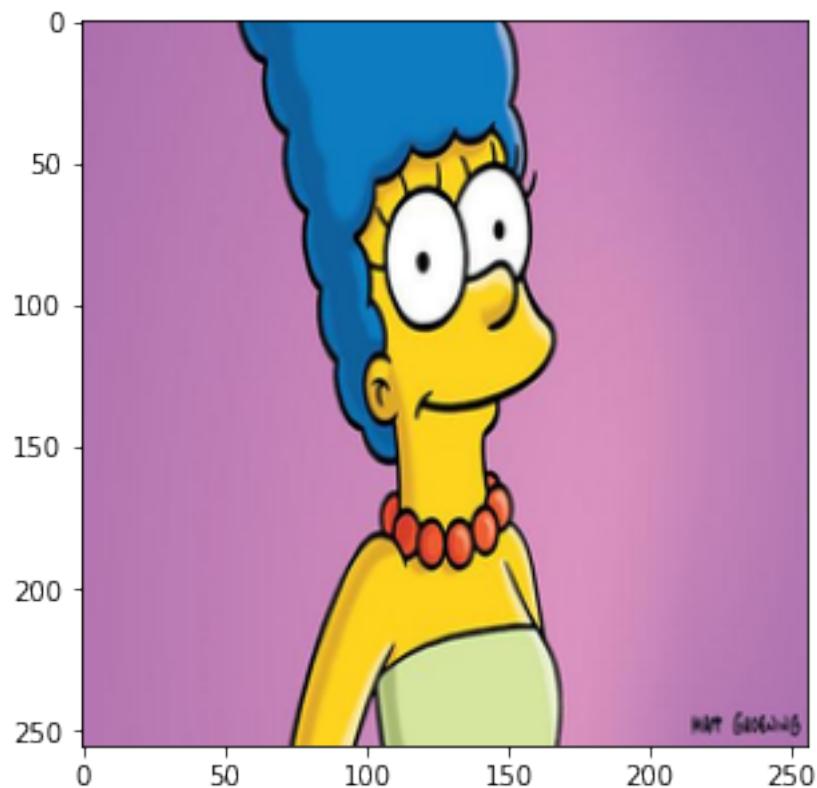
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191AE410>



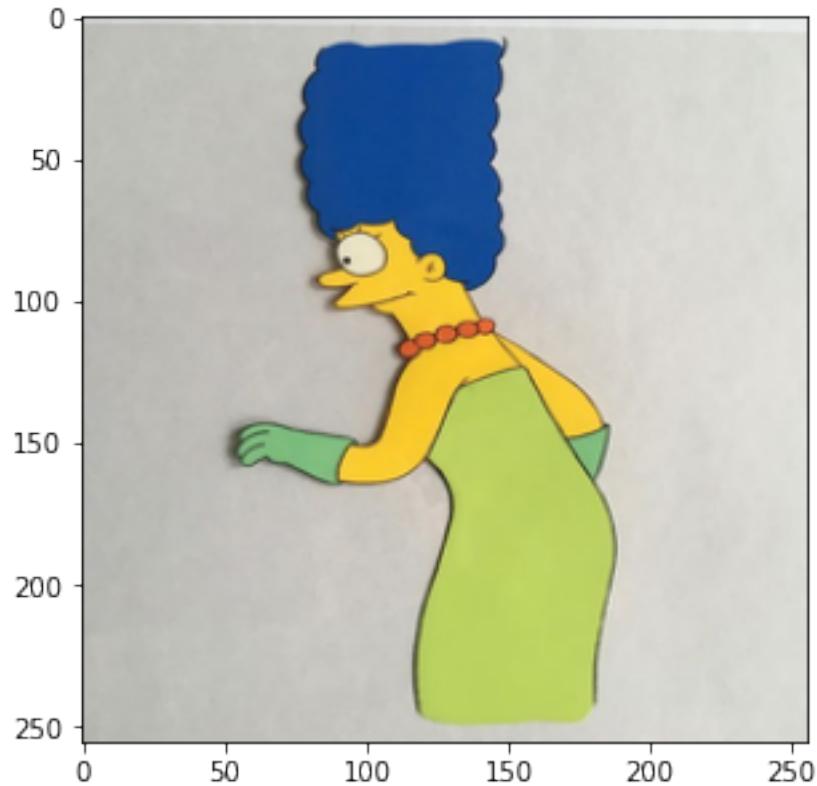
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE350>



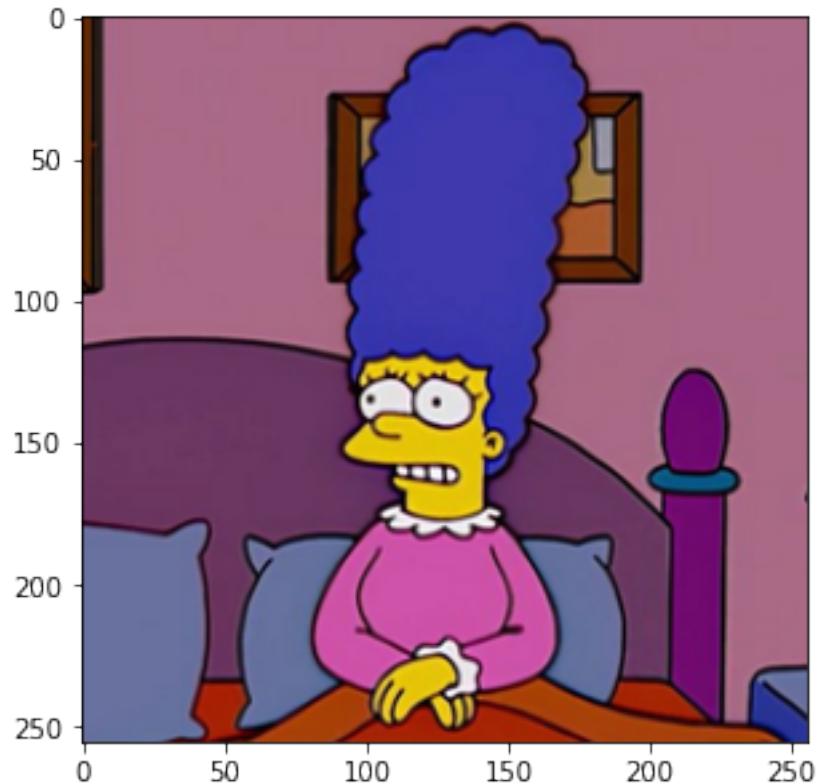
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE3D0>



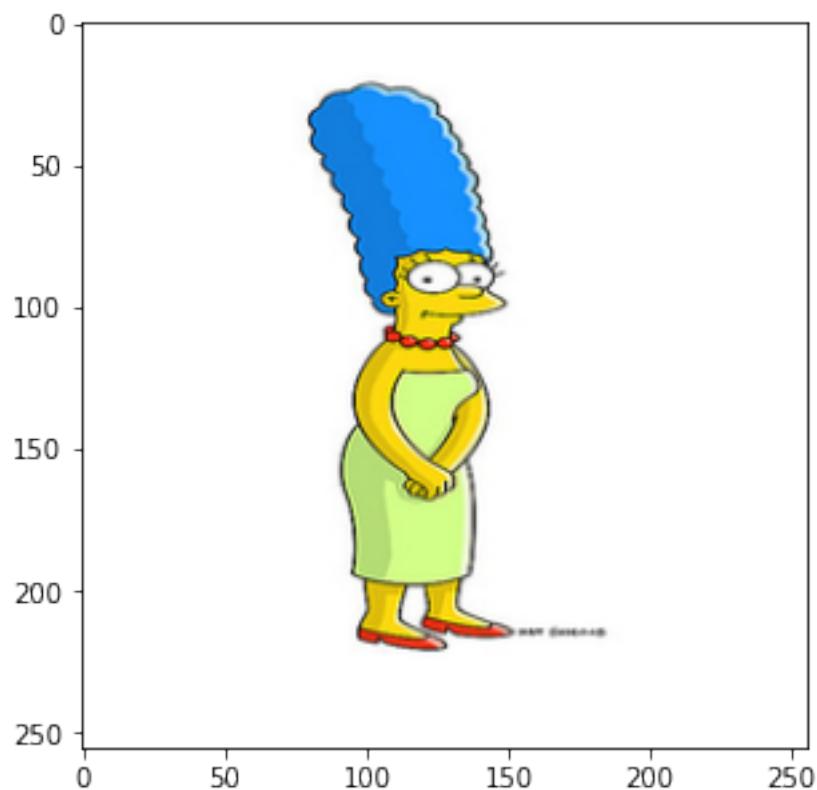
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE510>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE5D0>



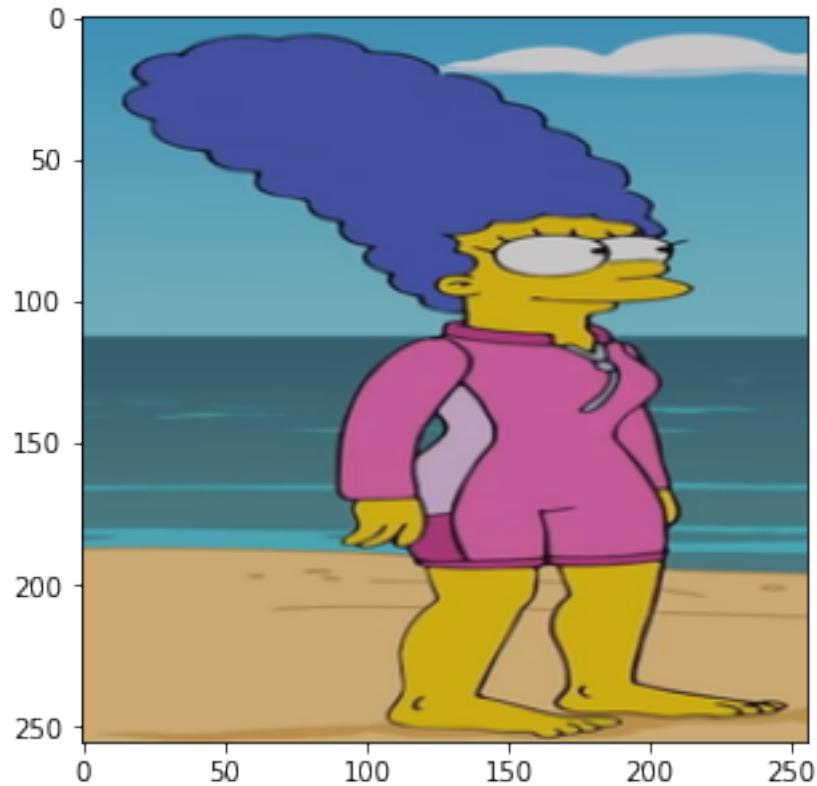
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE650>



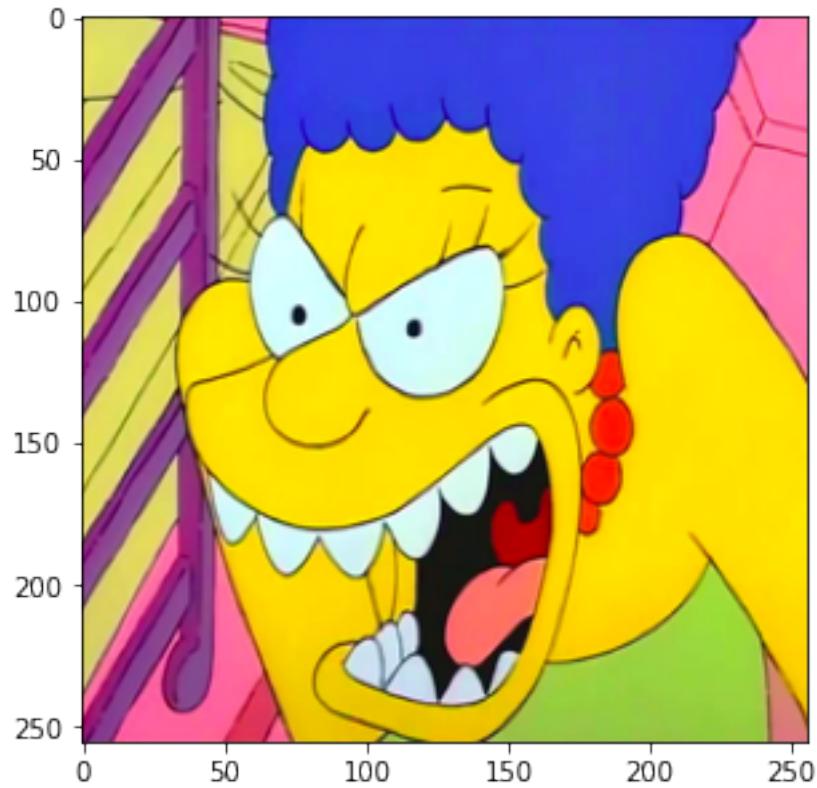
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE6D0>



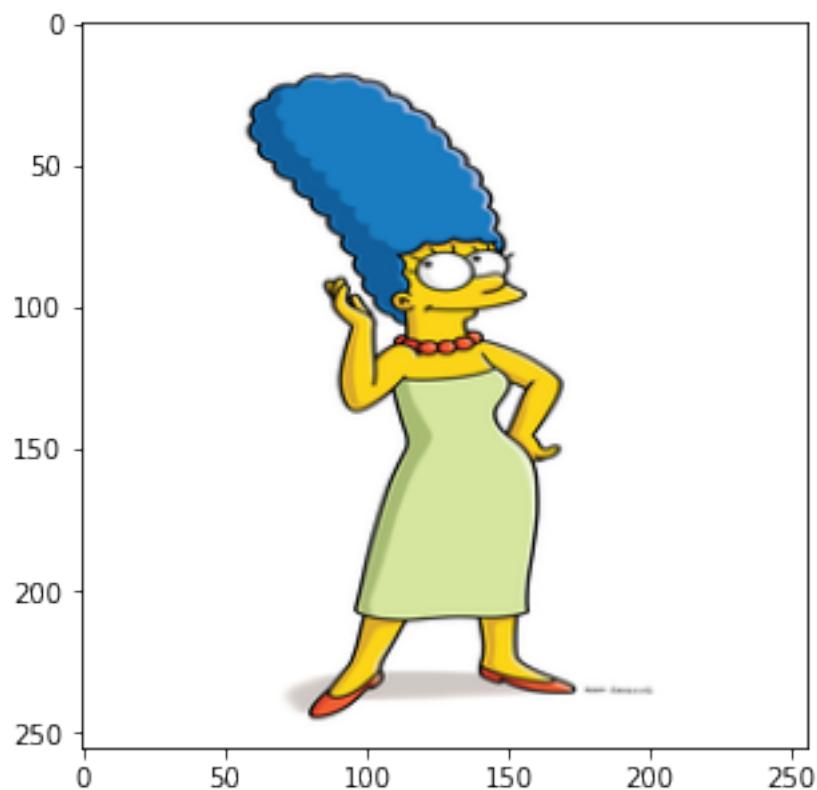
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191AE8D0>



<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191AE810>



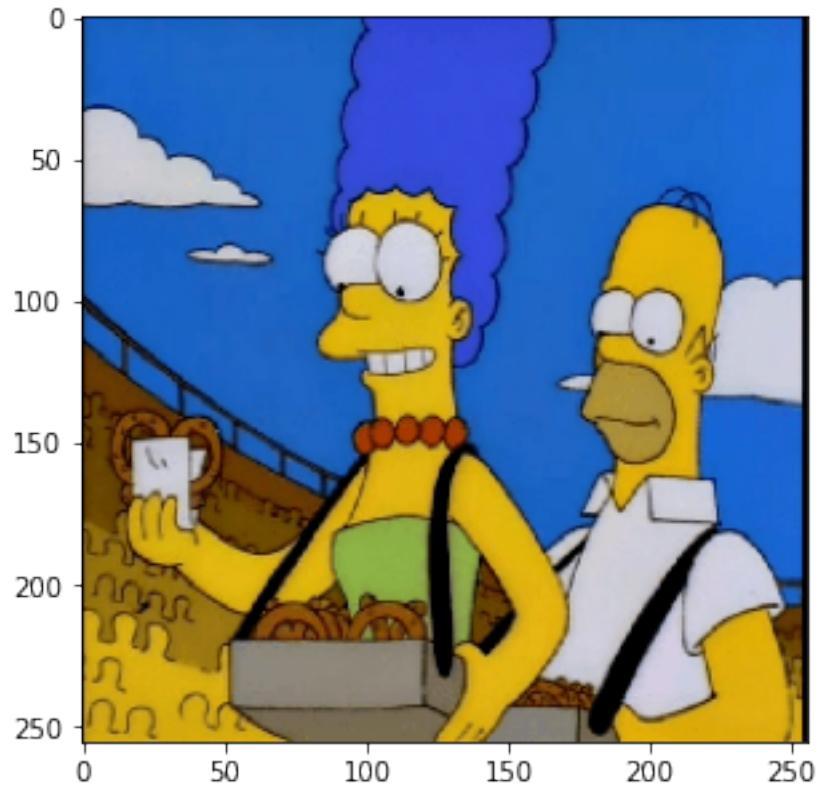
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191A0210>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE7D0>



<PIL.Image.Image image mode=P size=256x256 at 0x7FB1191AE850>



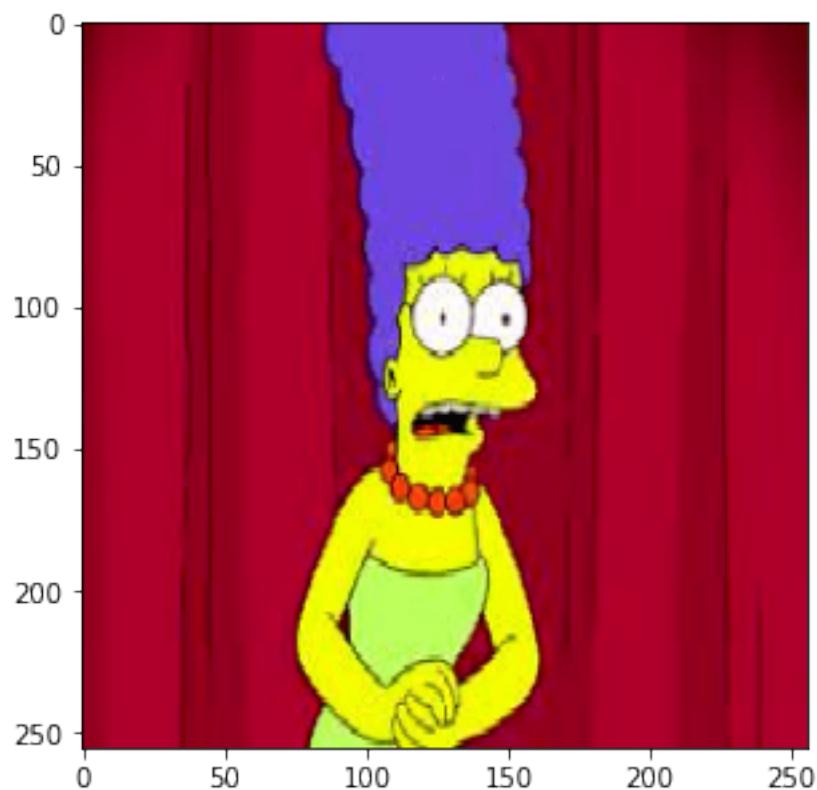
<PIL.Image.Image image mode=P size=256x256 at 0x7FB1191AEB50>



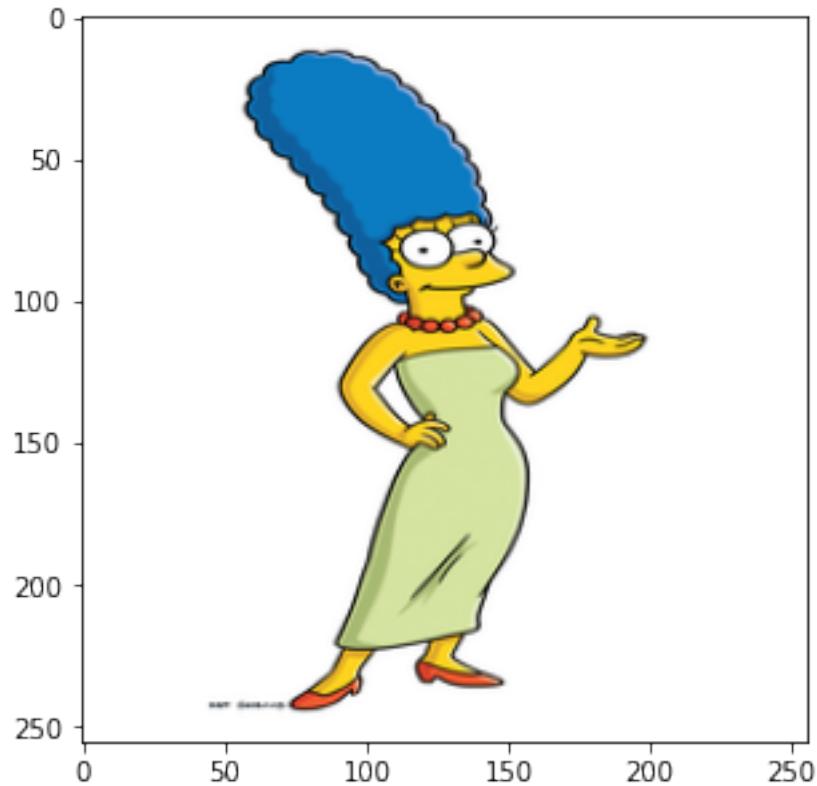
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AEB90>



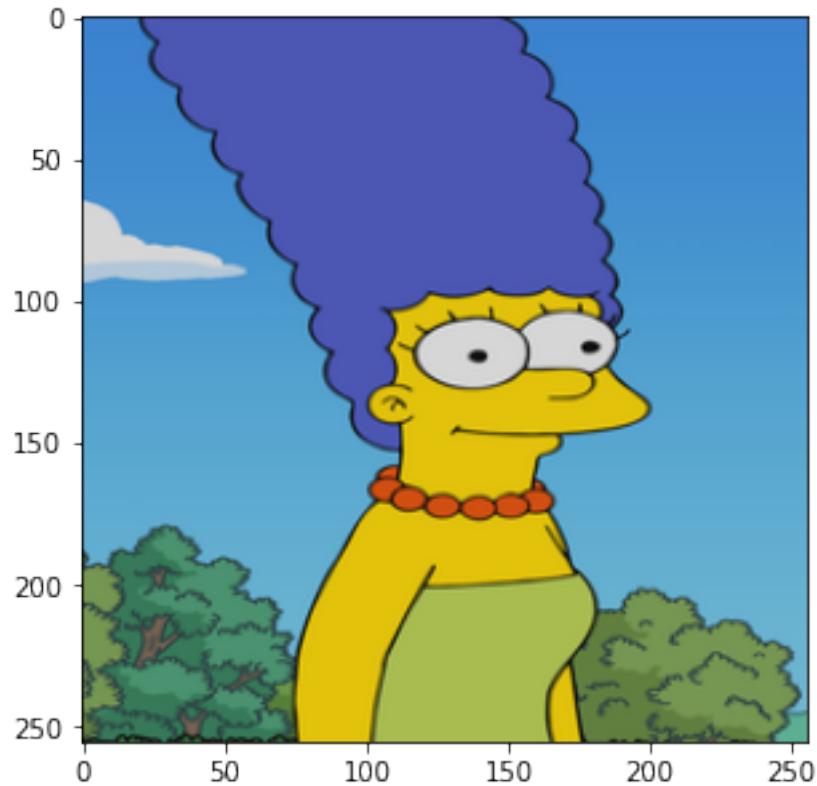
<PIL.Image.Image image mode=P size=256x256 at 0x7FB1191AEBD0>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AED10>



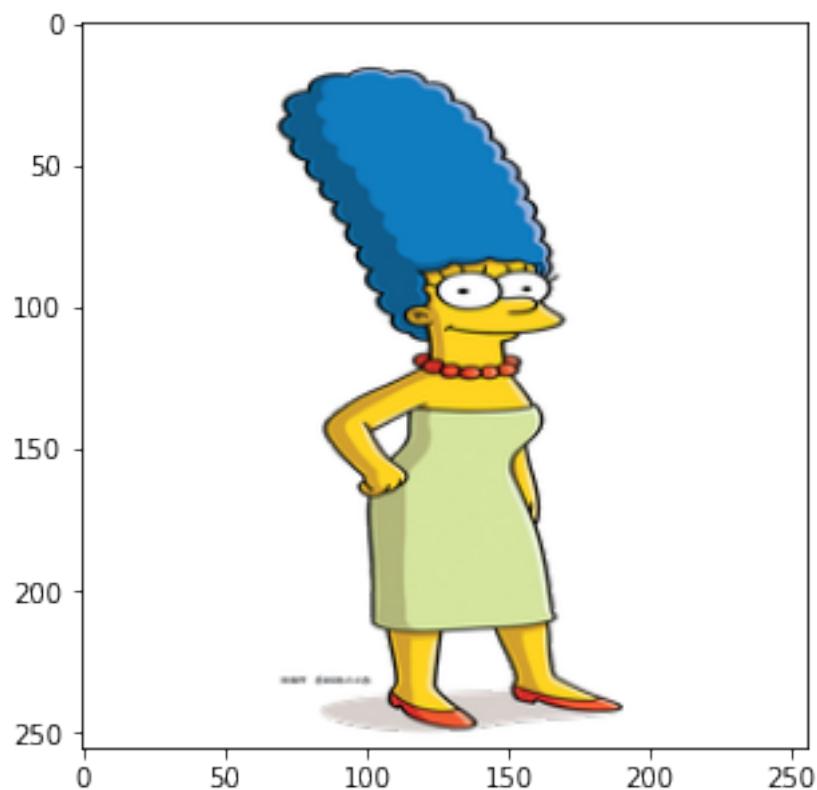
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AED50>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AEE10>



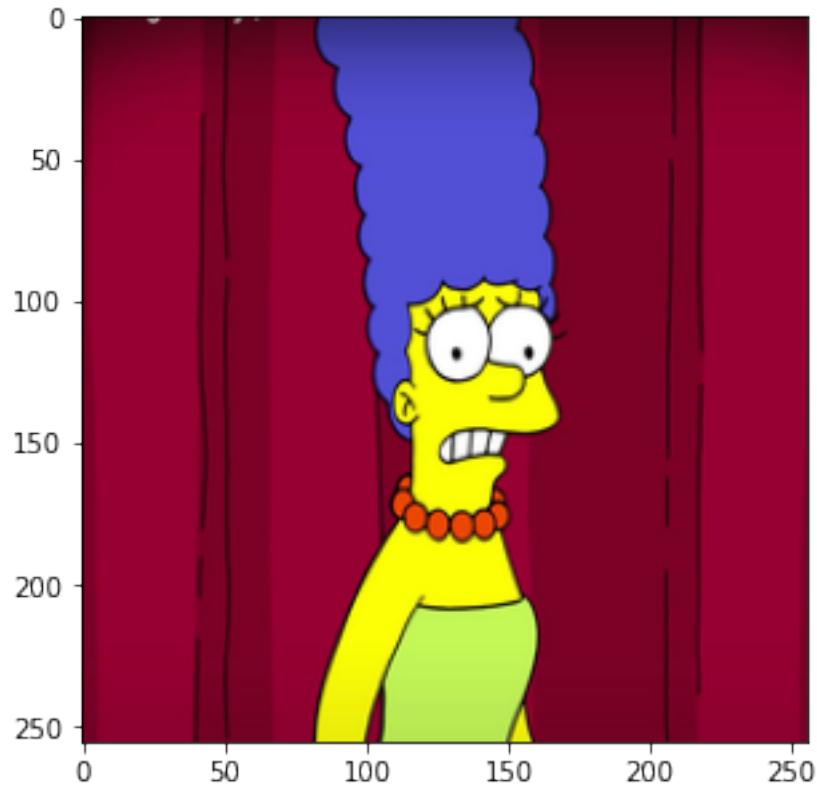
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191AEE90>



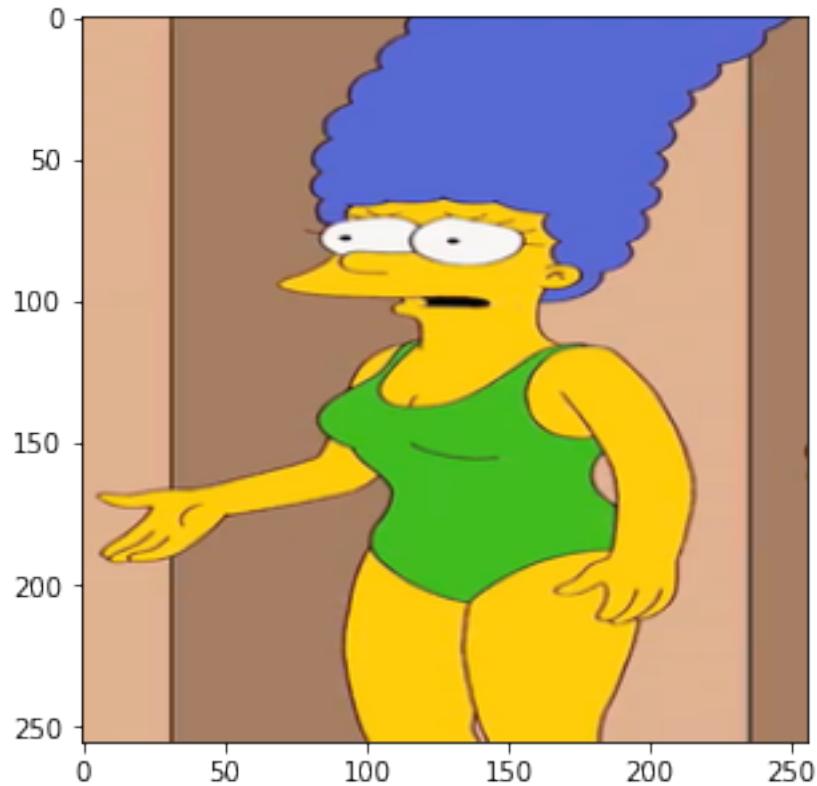
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AEE50>



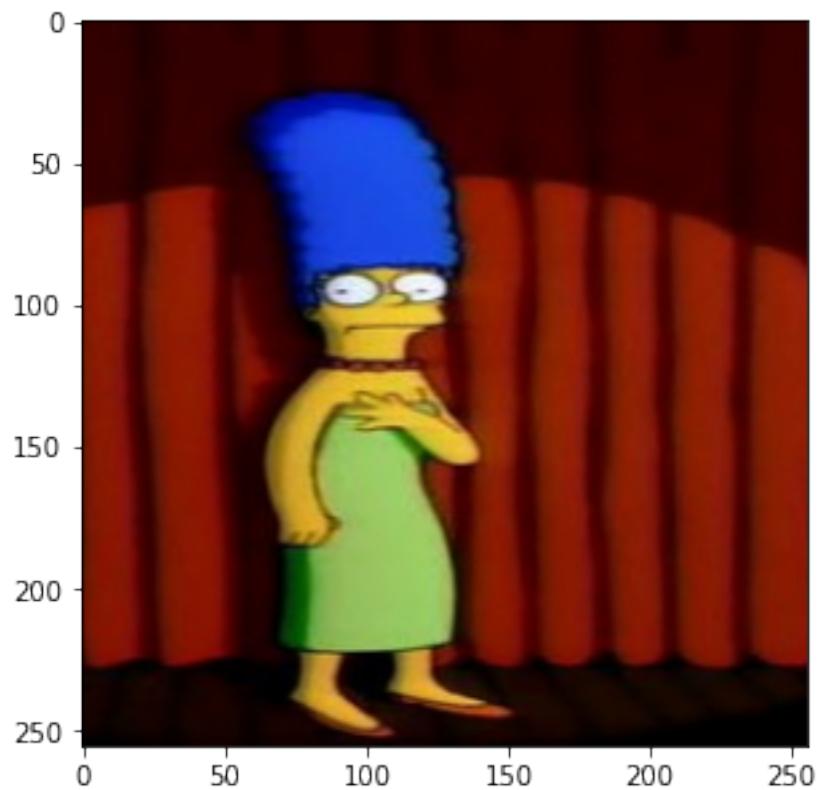
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AEED0>



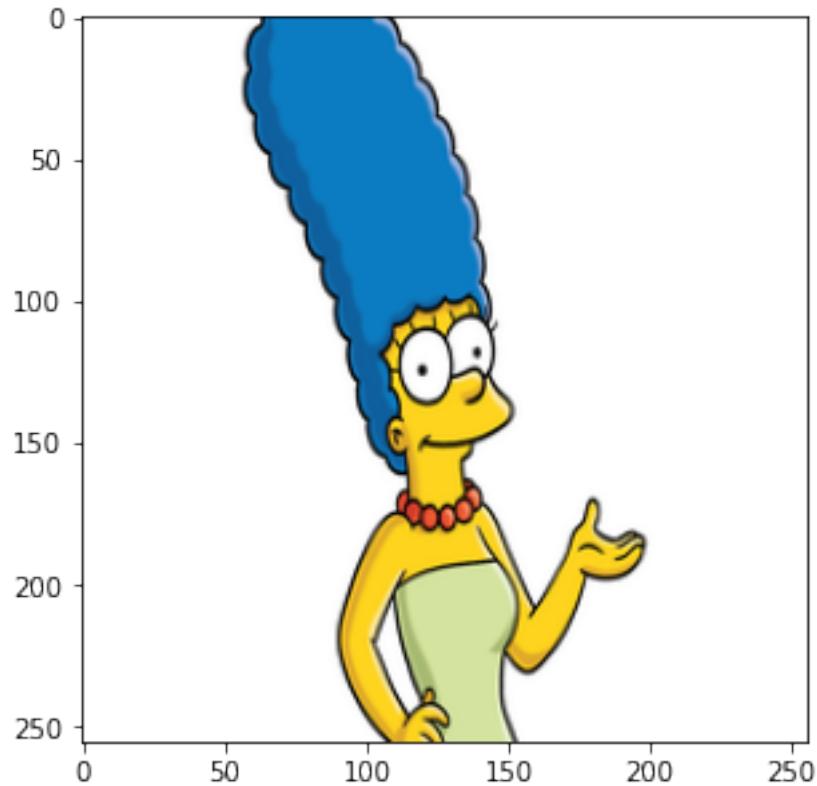
```
<PIL.Image.Image image mode=RGBA size=256x256 at 0x7FB1191BB250>
```



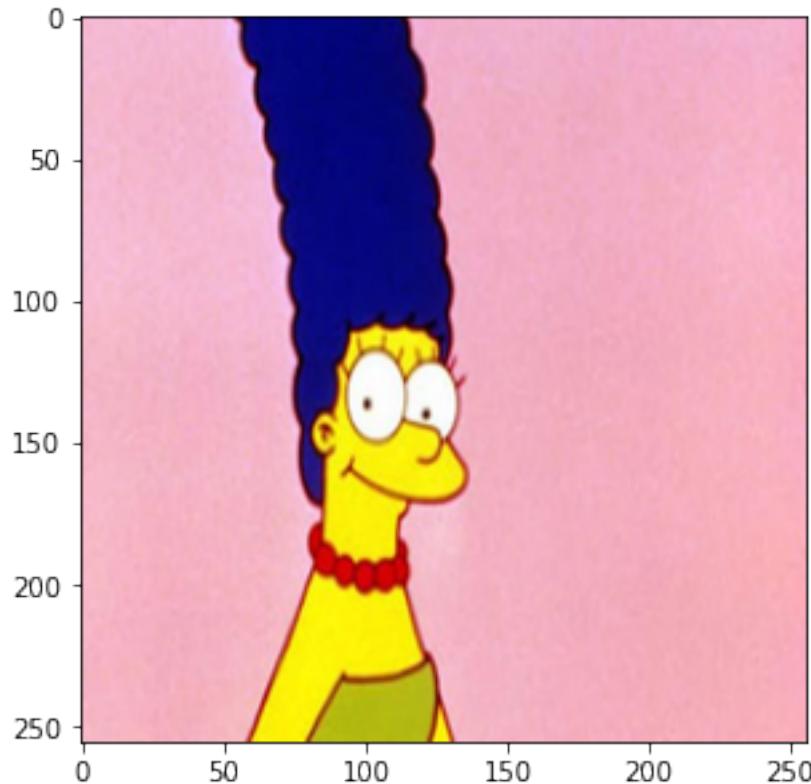
<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AEF50>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AEC50>



<PIL.Image.Image image mode=RGB size=256x256 at 0x7FB1191AE950>



The colours are not relevant in classification, resolution and detail are more important. So we will convert all the images to greyscale so TODO

```
[ ]: grayScaleHomer = []

for item in rsHomerImages:
    grayscaleim = item.convert('L')
    grayScaleHomer.append(grayscaleim)

grayScaleBart = []

for item in rsBartImages:
    grayscaleim = item.convert('L')
    grayScaleBart.append(grayscaleim)

grayScaleLisa = []

for item in rsLisaImages:
    grayscaleim = item.convert('L')
    grayScaleLisa.append(grayscaleim)
```

```

grayScaleMaggie = []

for item in rsMaggieImages:
    grayscaleim = item.convert('L')
    grayScaleMaggie.append(grayscaleim)

grayScaleMarge = []

for item in rsMargeImages:
    grayscaleim = item.convert('L')
    grayScaleMarge.append(grayscaleim)

```

/usr/local/lib/python3.7/dist-packages/PIL/Image.py:960: UserWarning: Palette images with Transparency expressed in bytes should be converted to RGBA images
 "Palette images with Transparency expressed in bytes should be "

```

[ ]: for item in grayScaleHomer:
    print(item)
    plt.imshow(item, cmap=plt.cm.gray)
    plt.show()

for item in grayScaleLisa:
    print(item)
    plt.imshow(item, cmap=plt.cm.gray)
    plt.show()

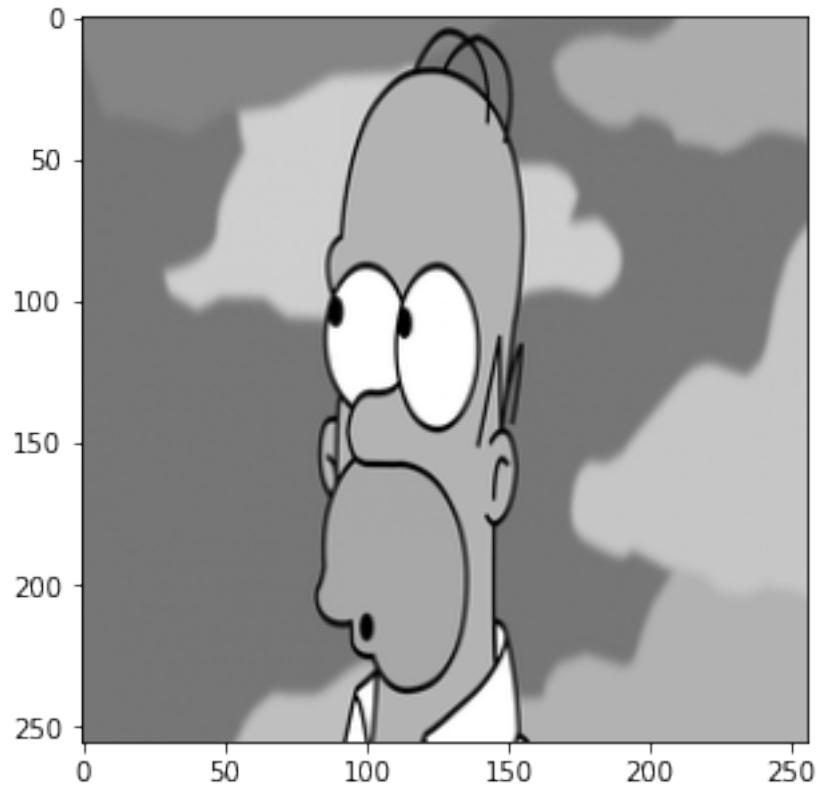
for item in grayScaleBart:
    print(item)
    plt.imshow(item, cmap=plt.cm.gray)
    plt.show()

for item in grayScaleMaggie:
    print(item)
    plt.imshow(item, cmap=plt.cm.gray)
    plt.show()

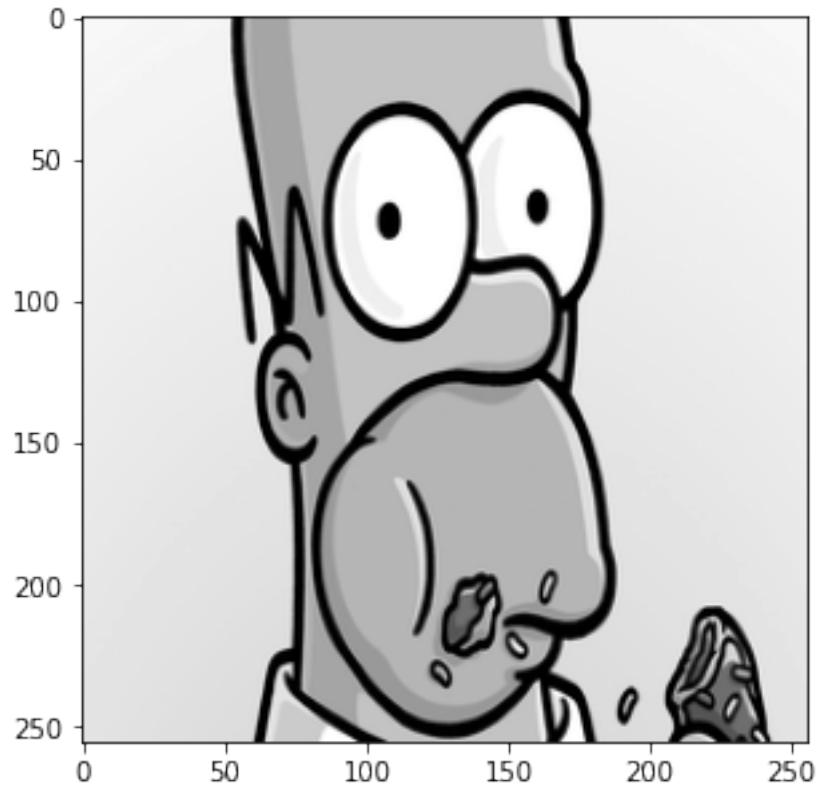
for item in grayScaleMarge:
    print(item)
    plt.imshow(item, cmap=plt.cm.gray)
    plt.show()

```

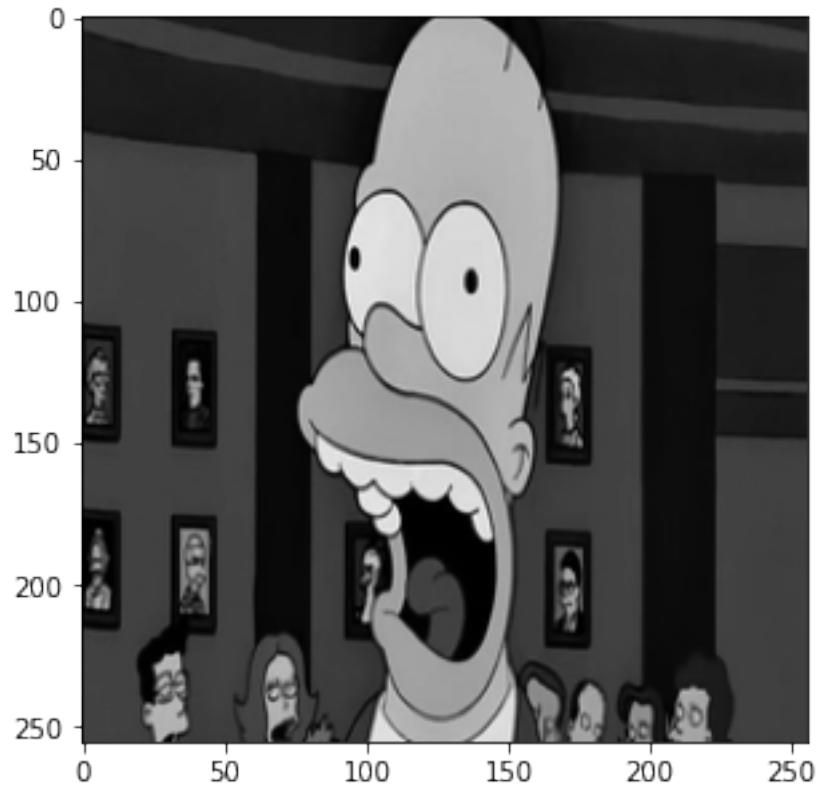
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11922D310>



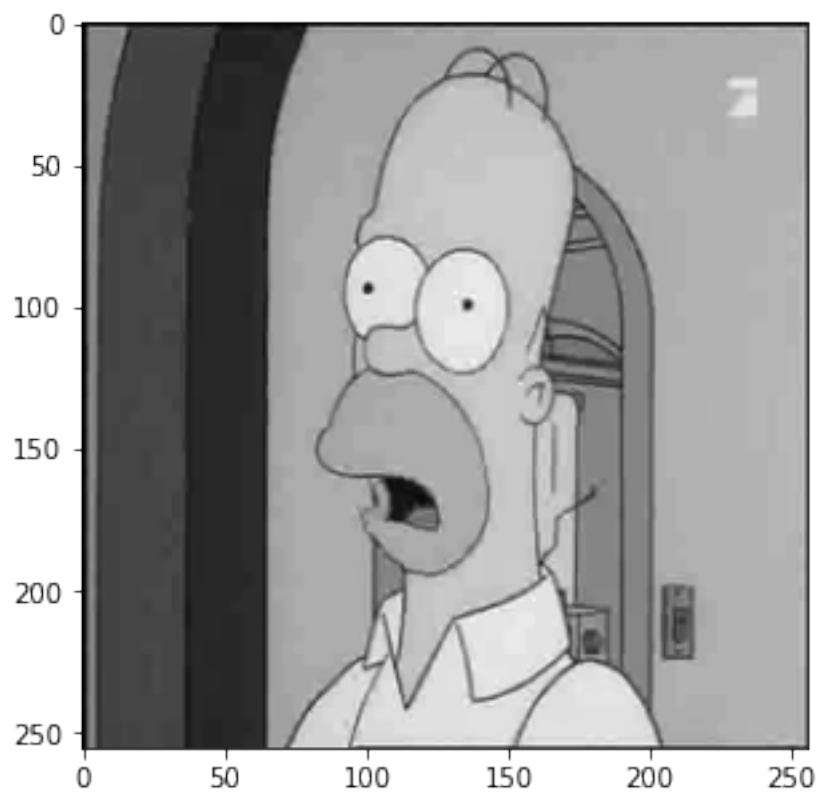
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1198401D0>



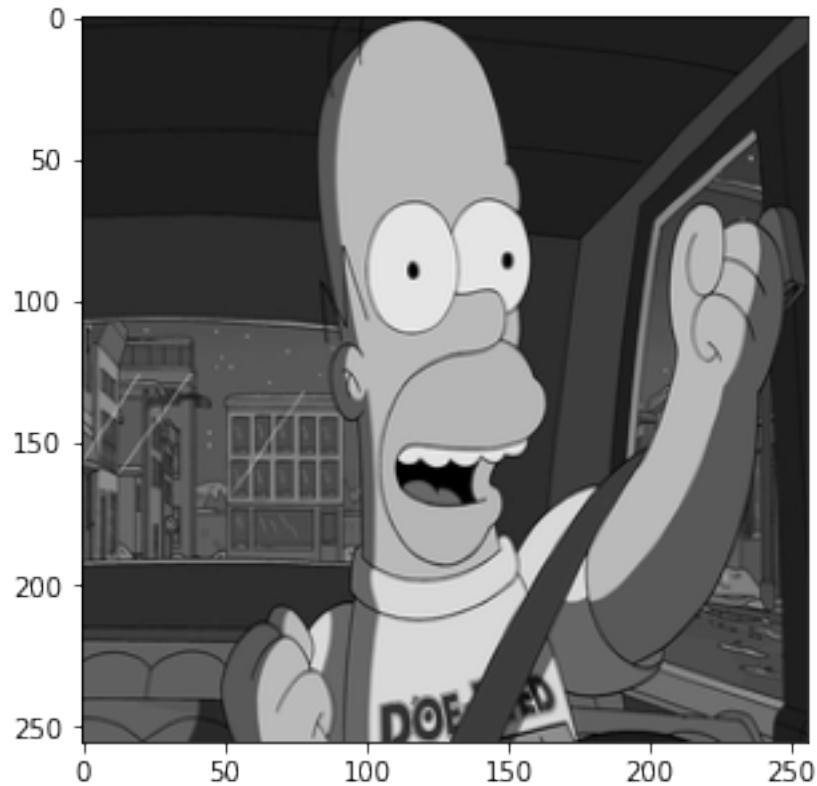
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119840850>



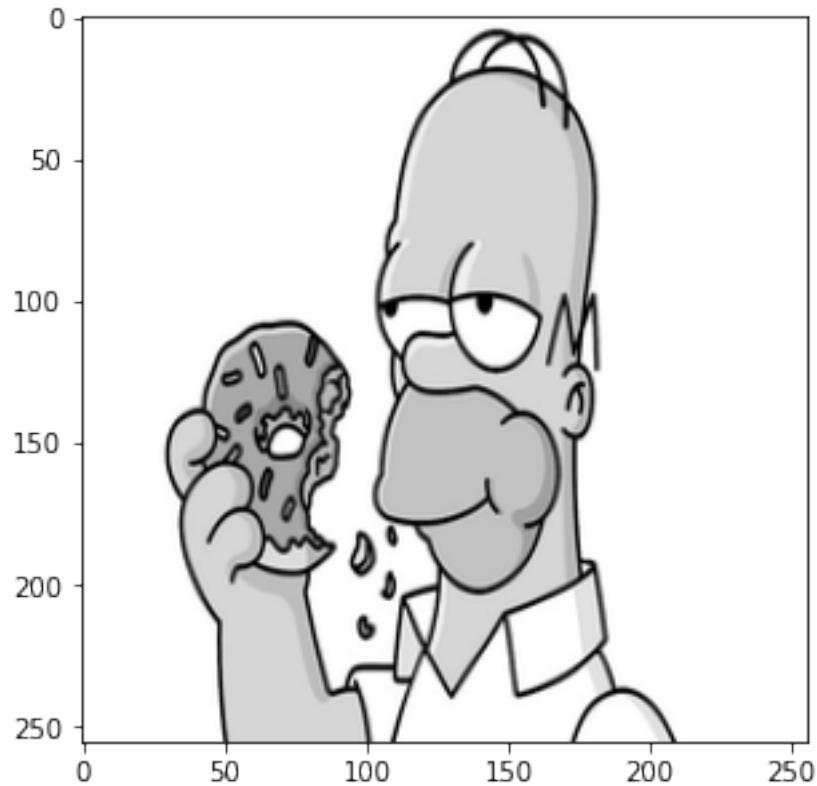
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1198402D0>



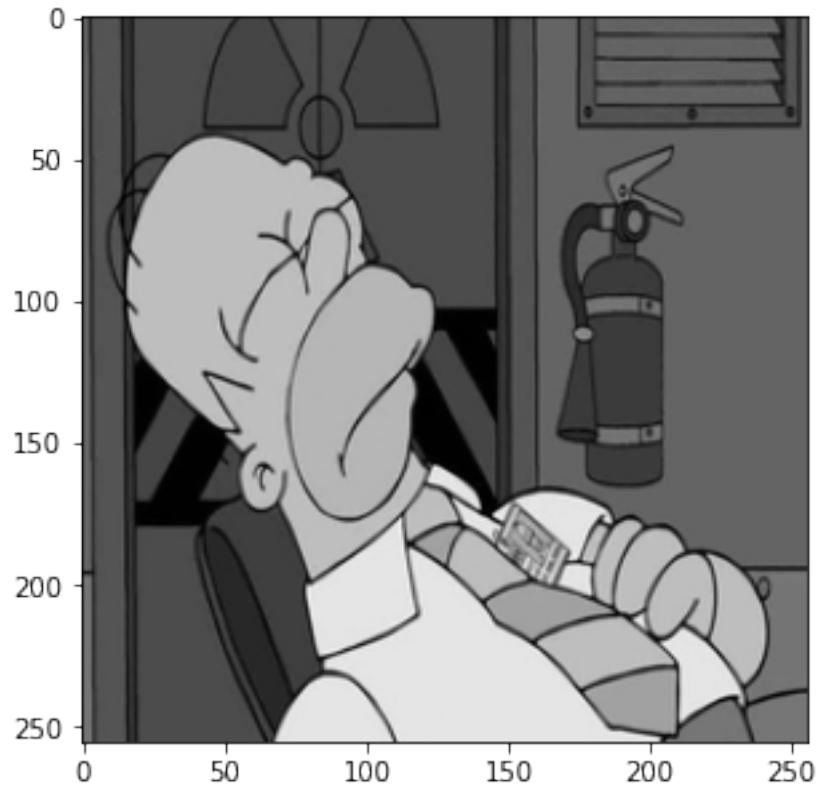
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11966E850>



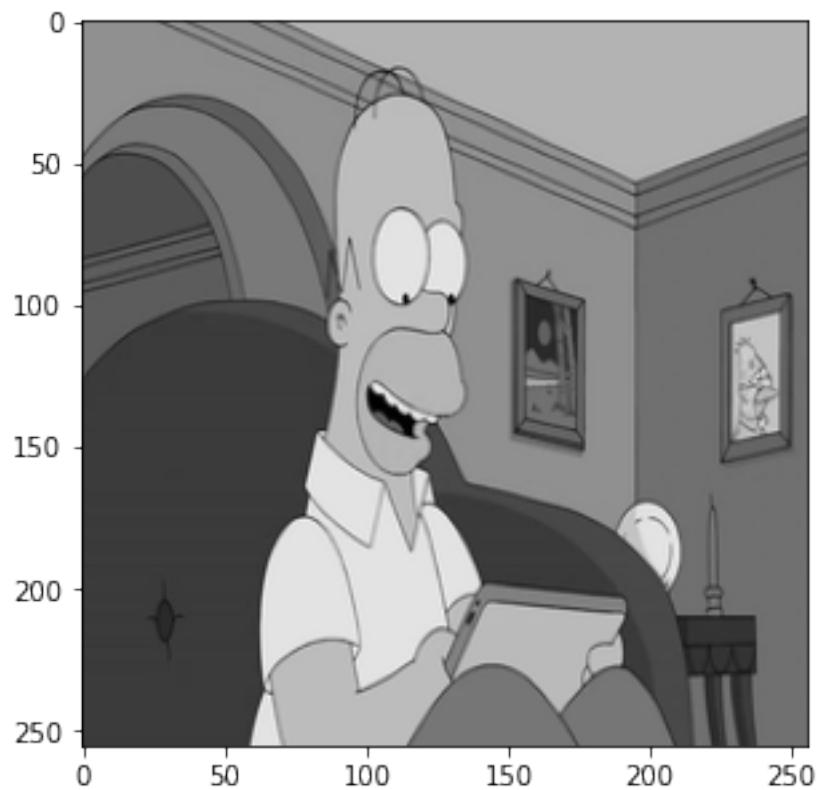
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11958C550>



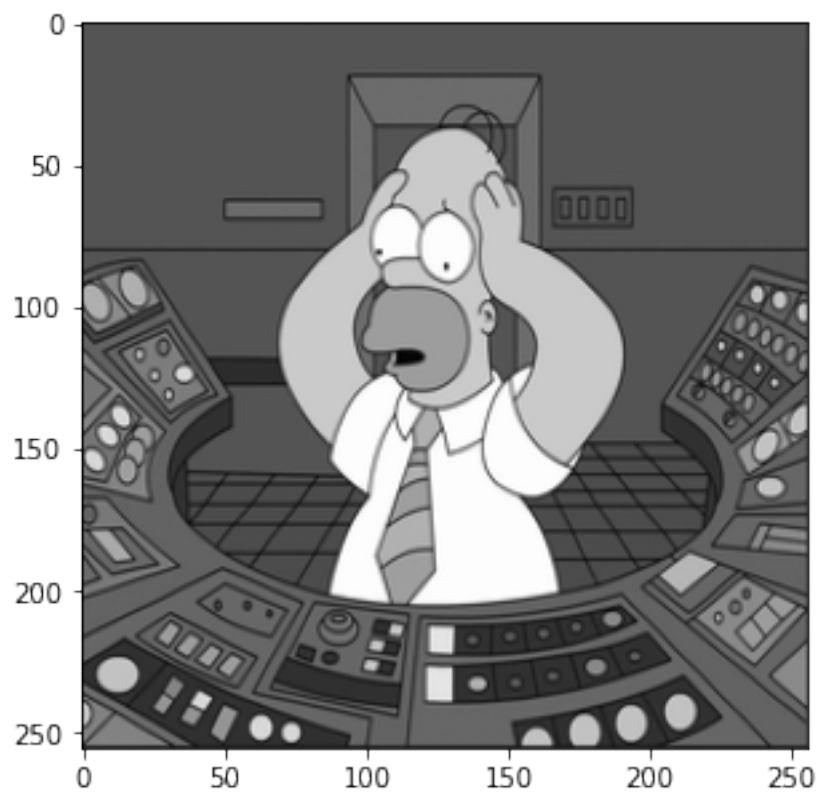
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E110>



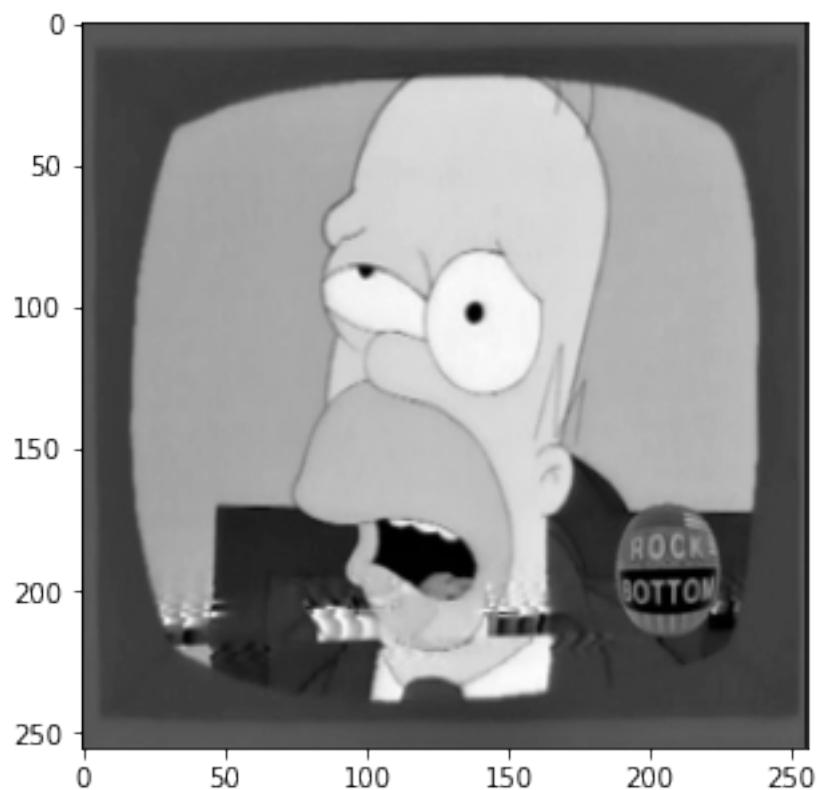
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954EED0>



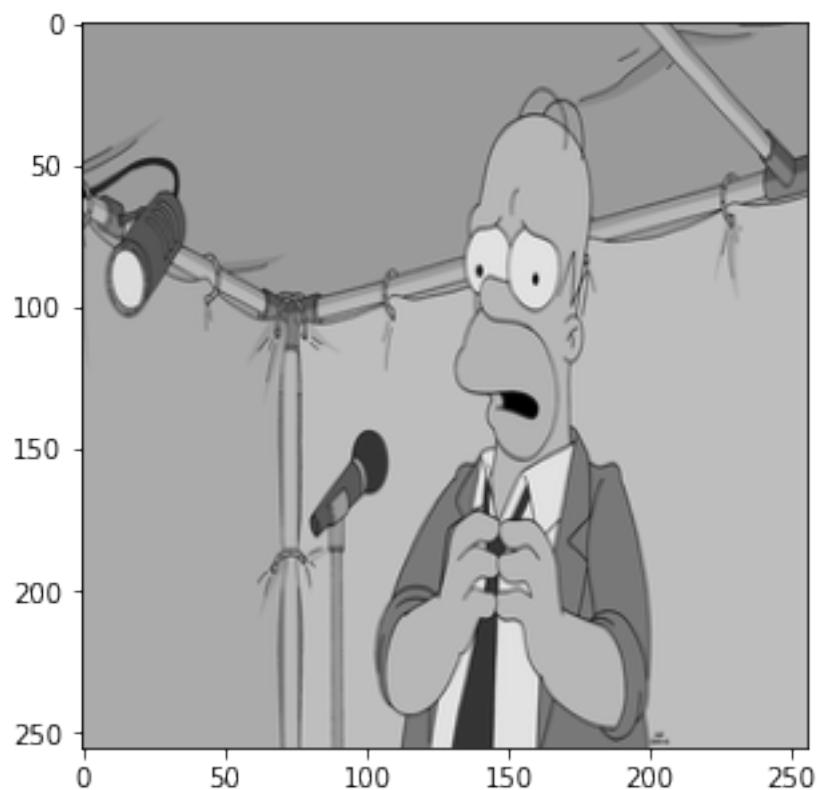
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E610>



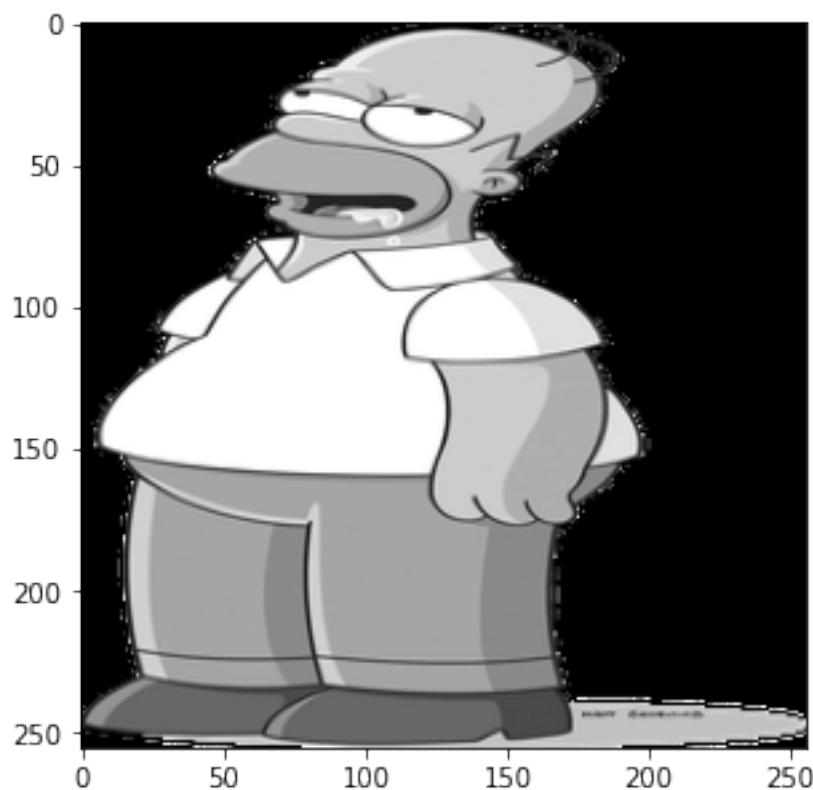
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11958CDD0>



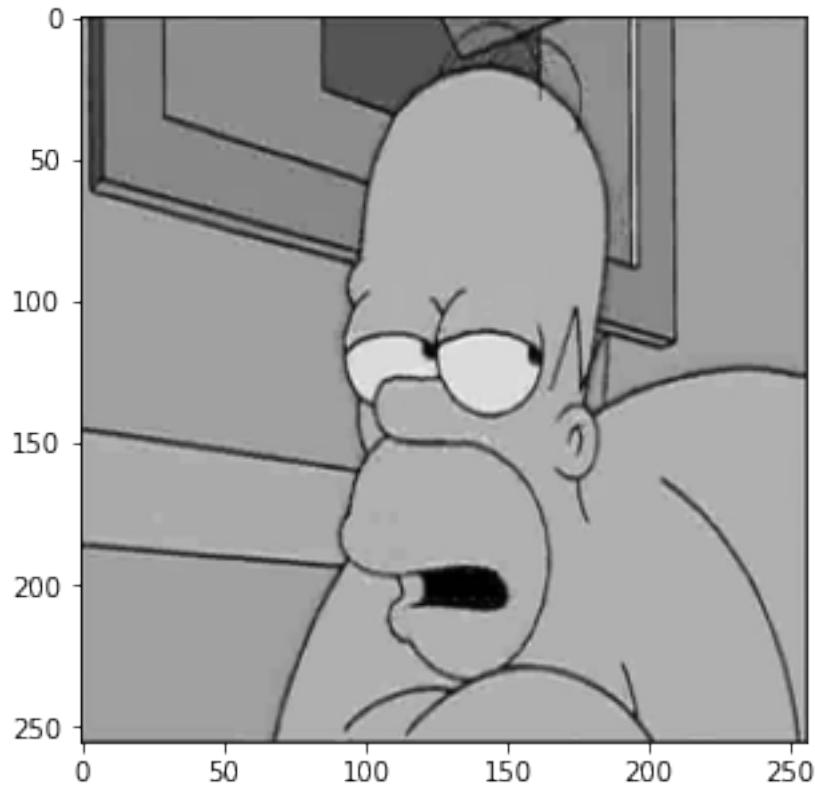
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954EA10>



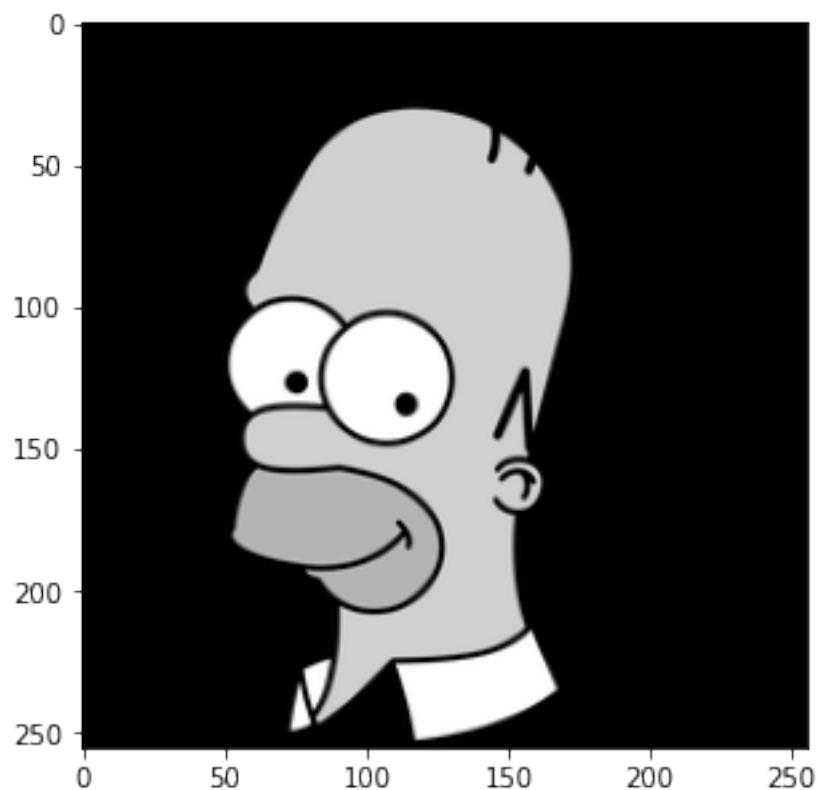
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E8D0>



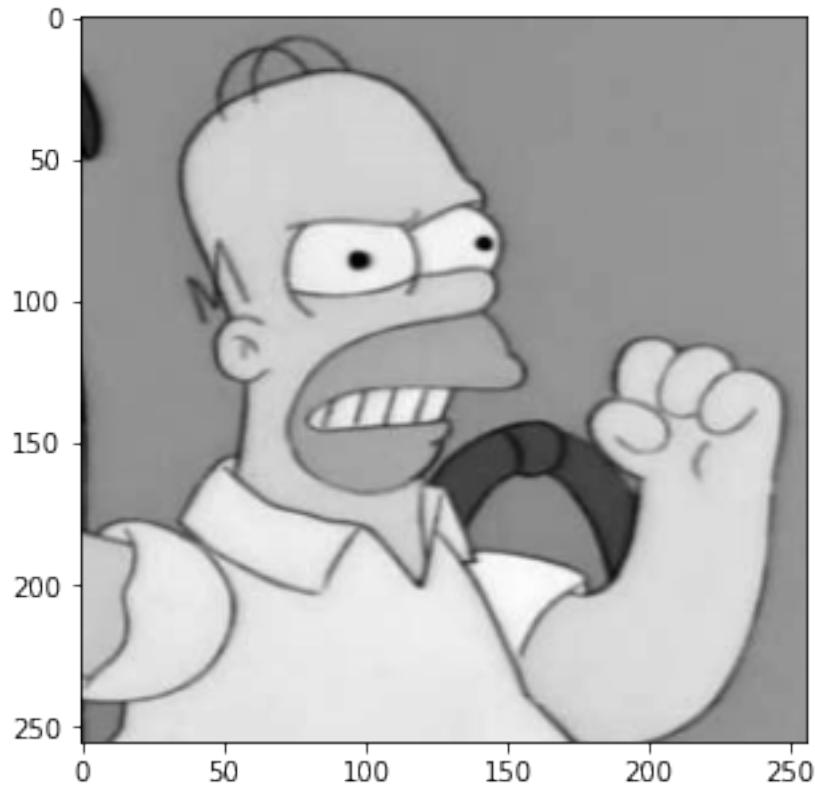
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E810>



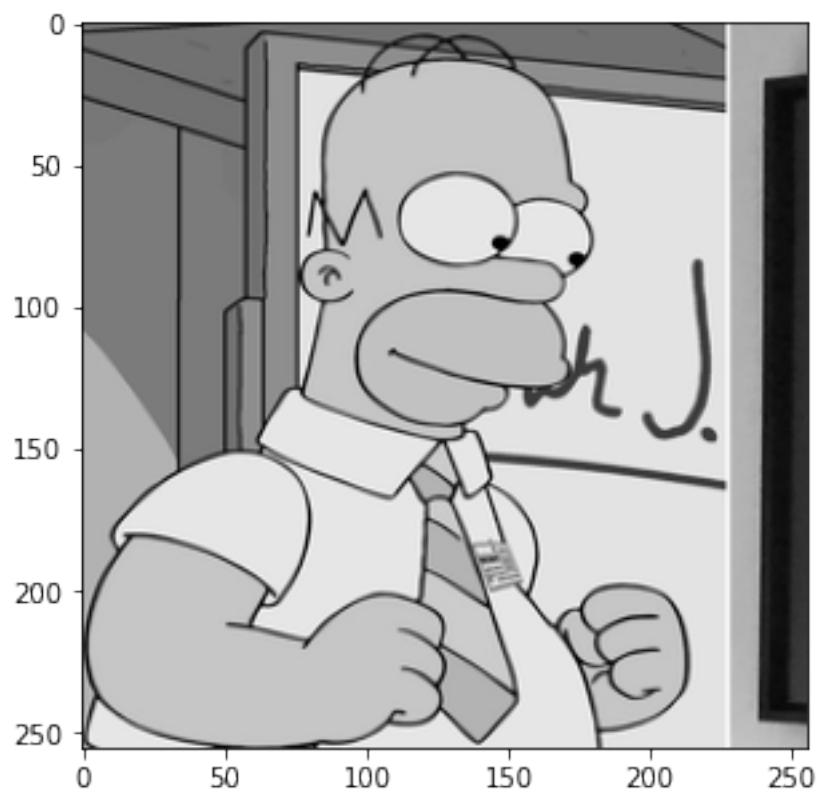
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E410>



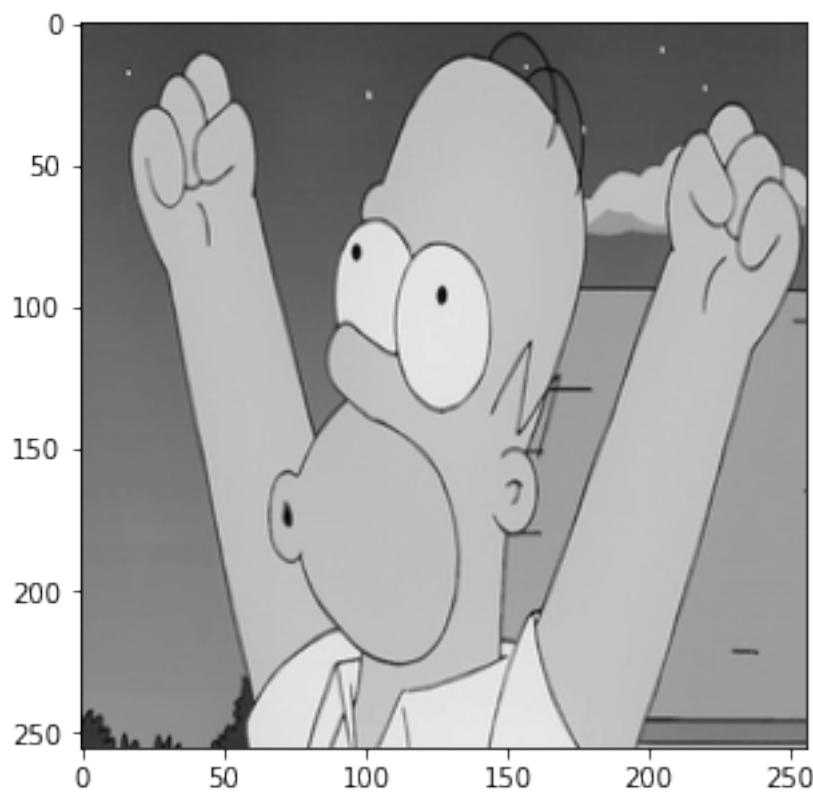
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E450>



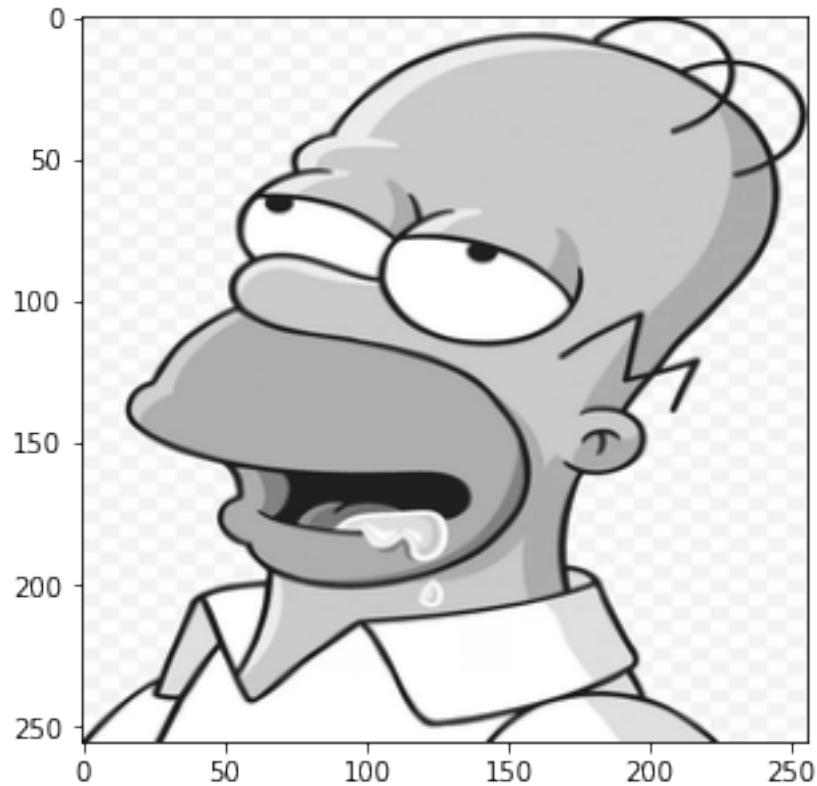
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954EB90>



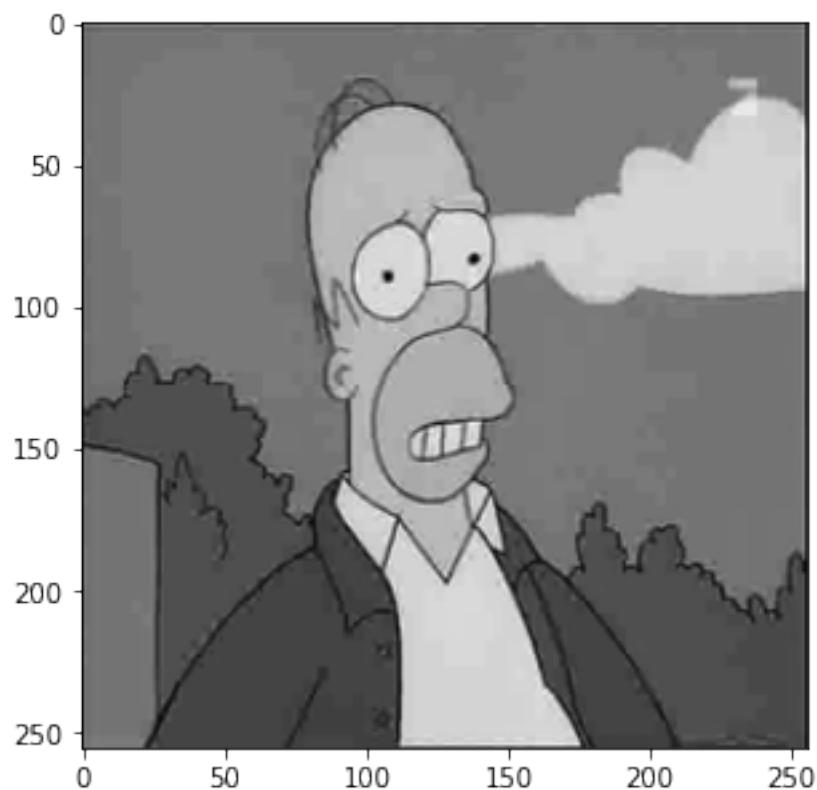
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E7D0>



<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E390>



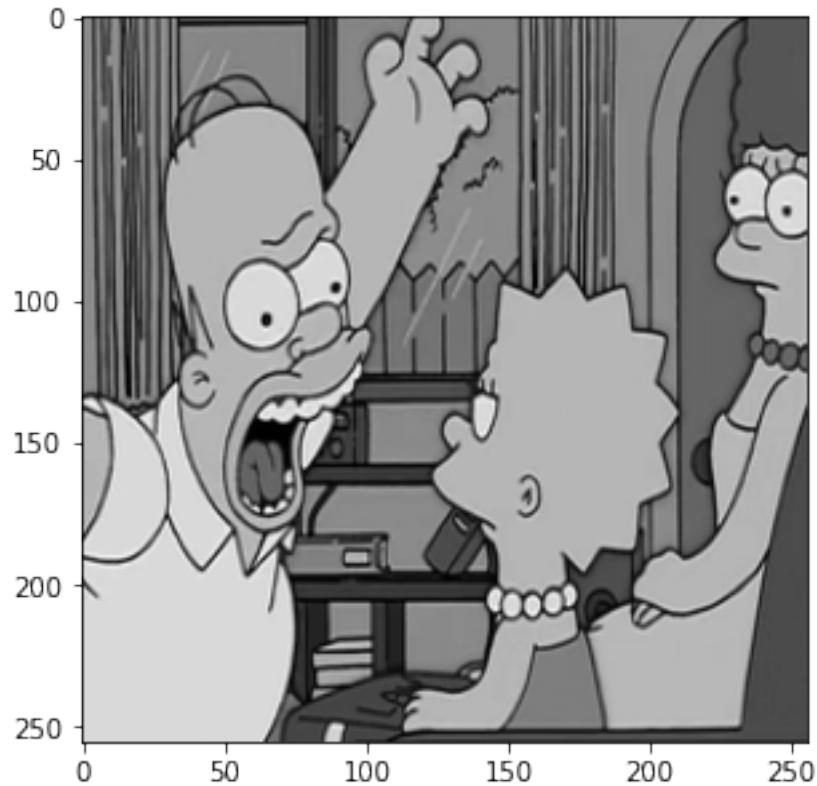
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E3D0>



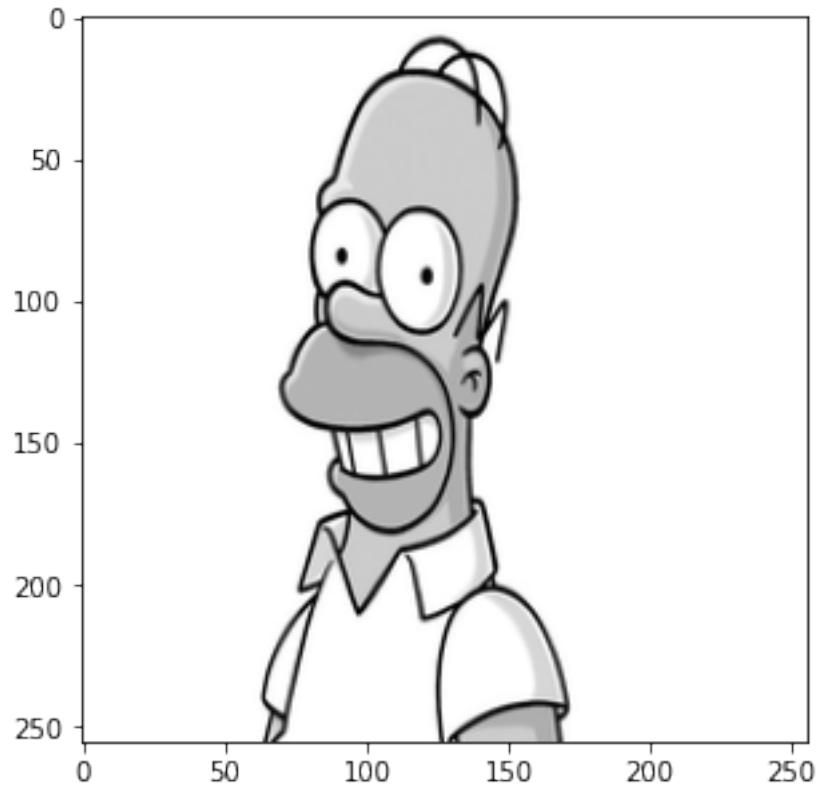
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E150>
```



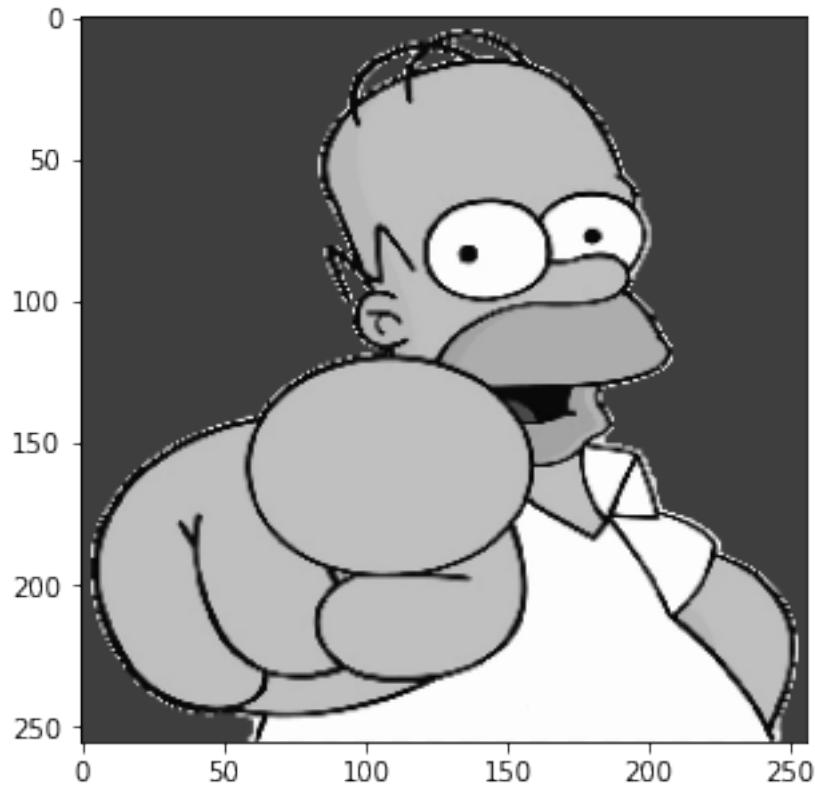
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E2D0>



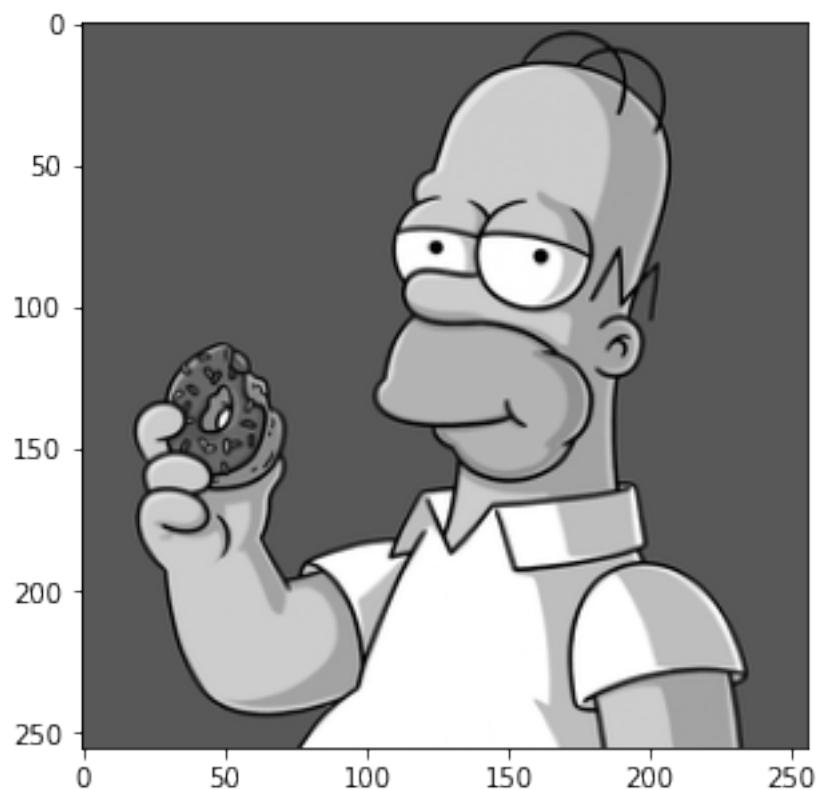
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E210>



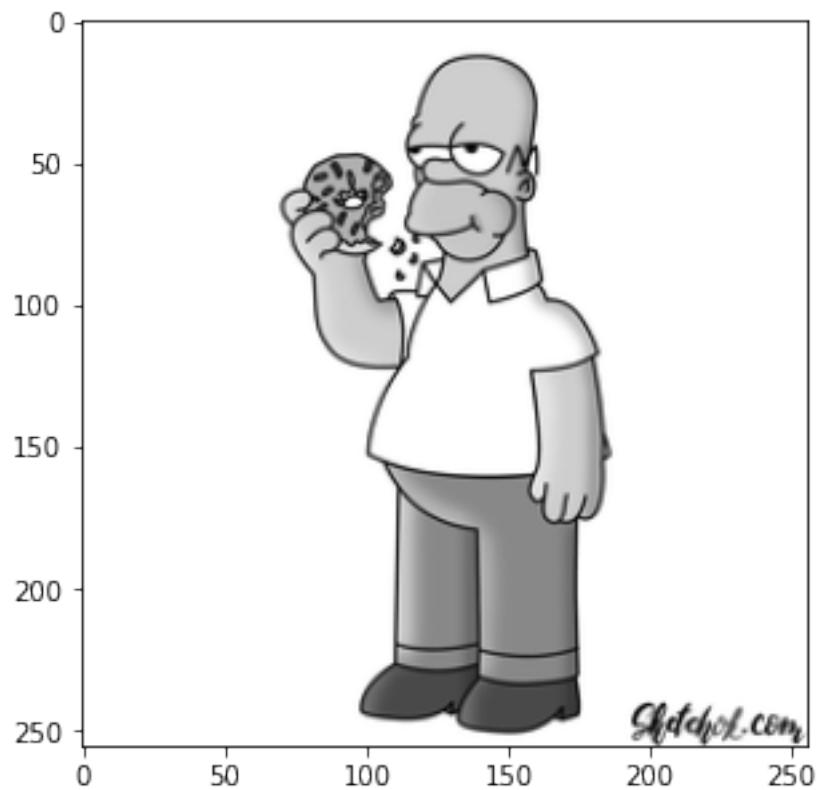
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119877850>



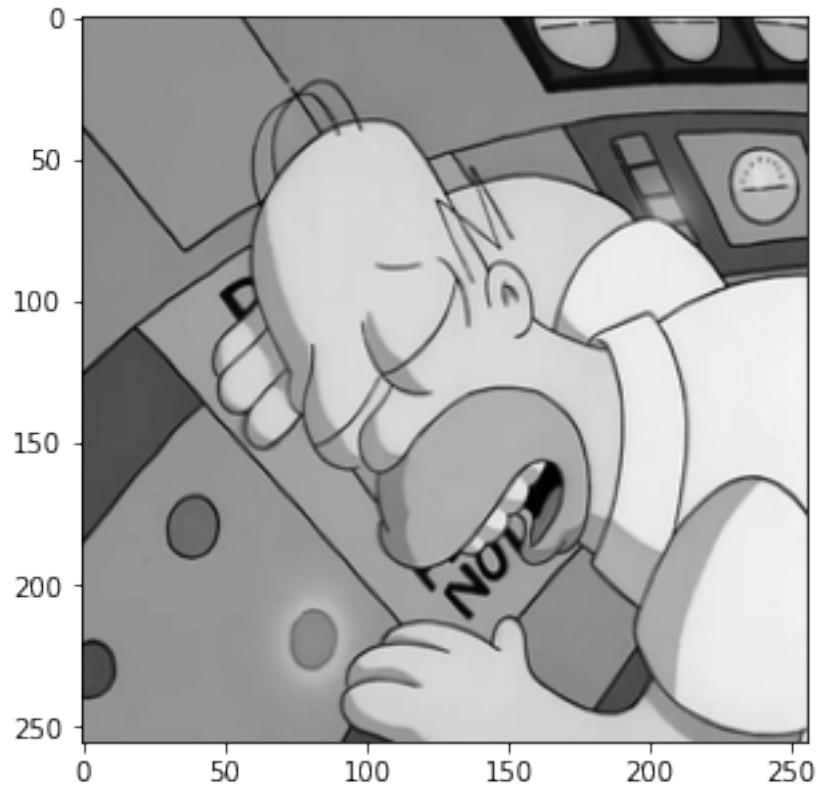
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E4D0>



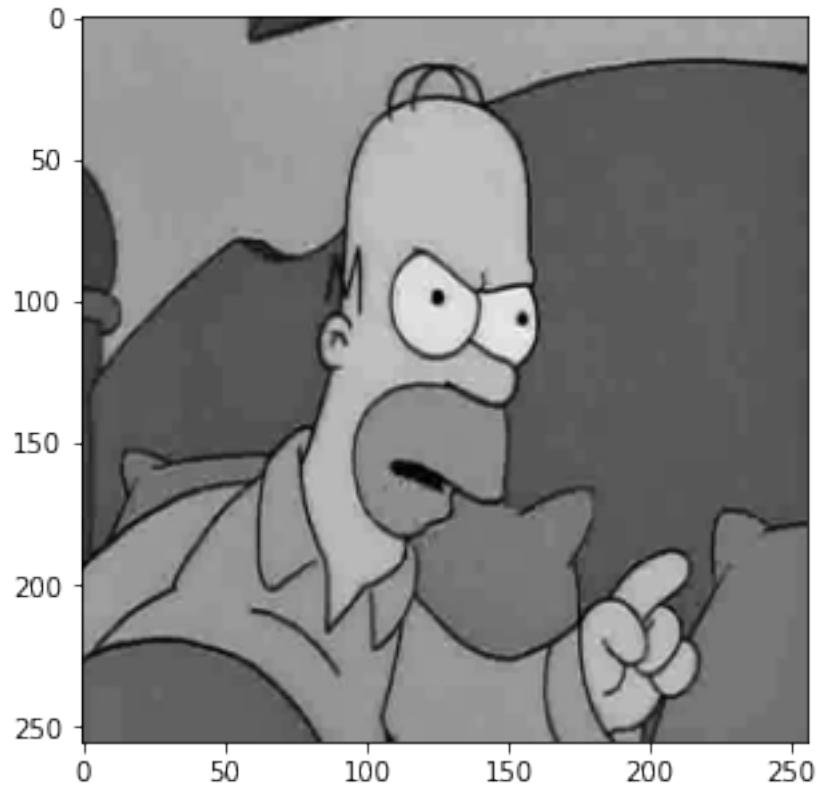
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11954E250>



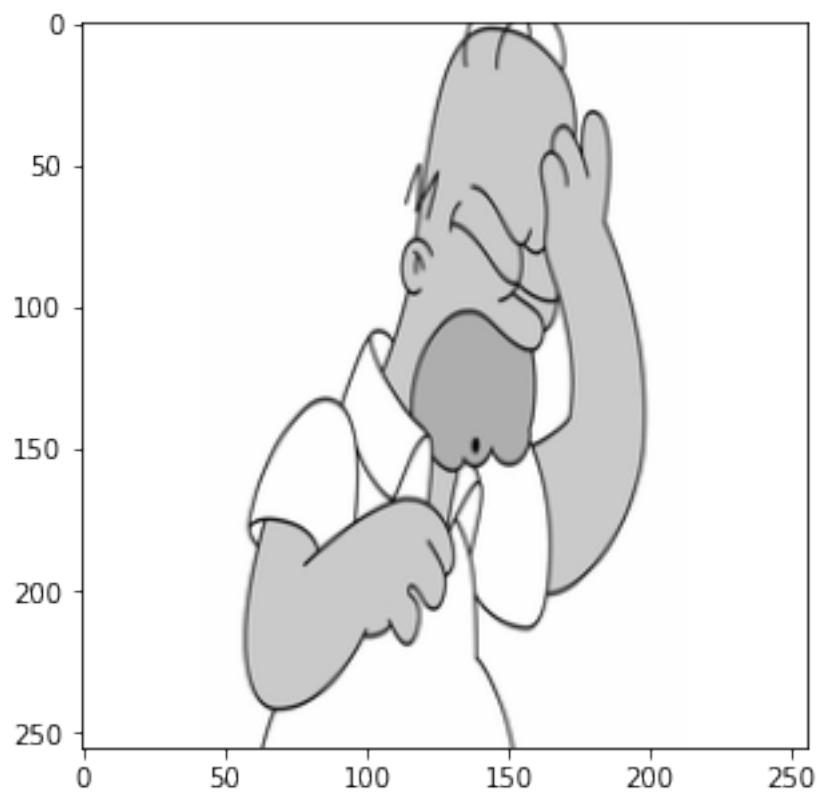
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B14E90>



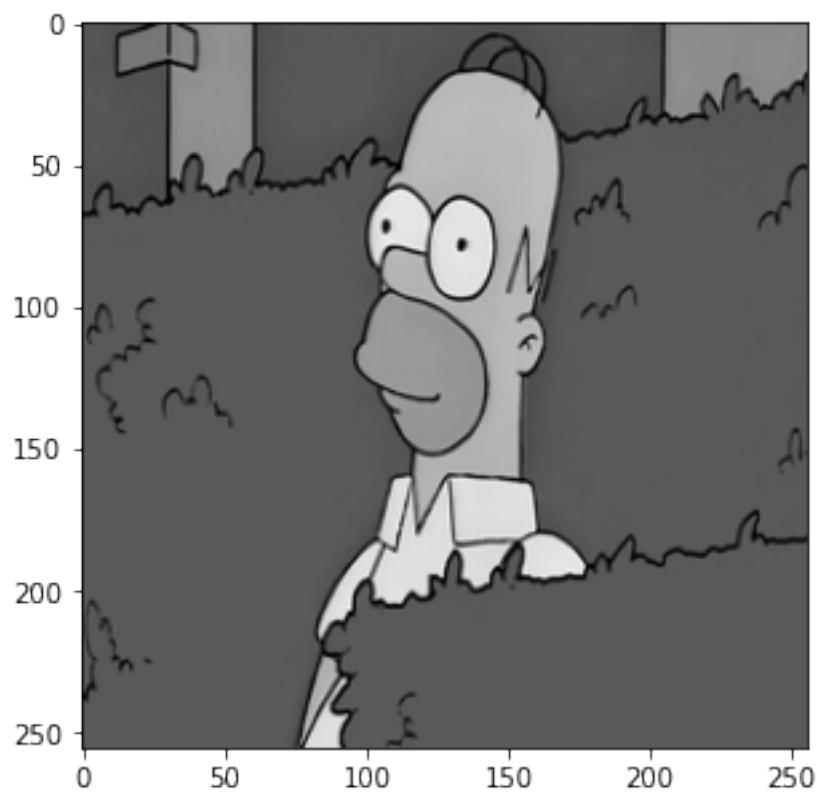
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11966CD10>



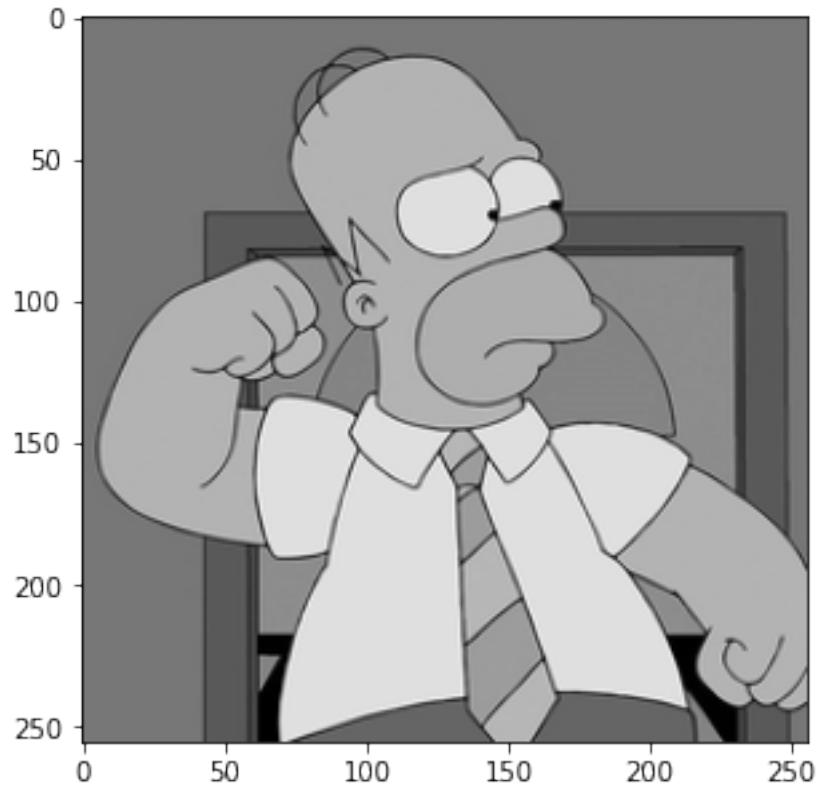
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11966C290>



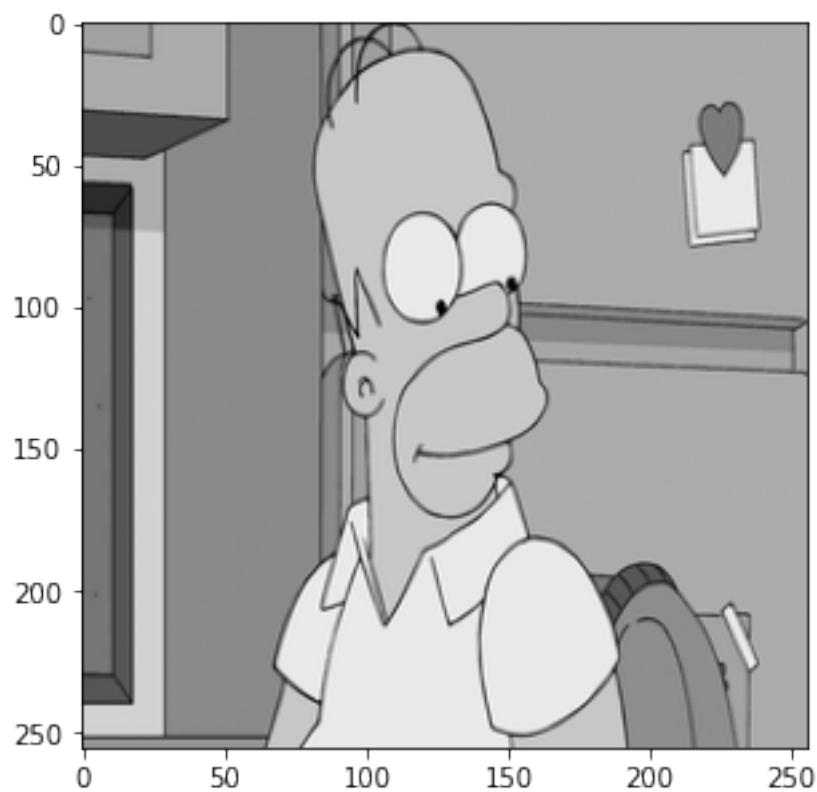
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B590>



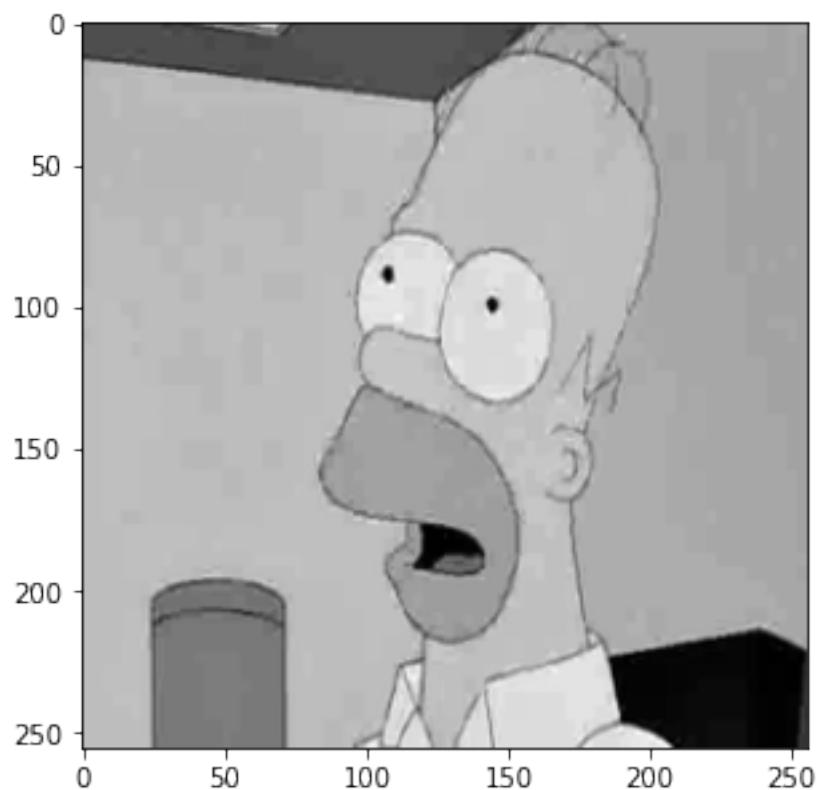
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B610>



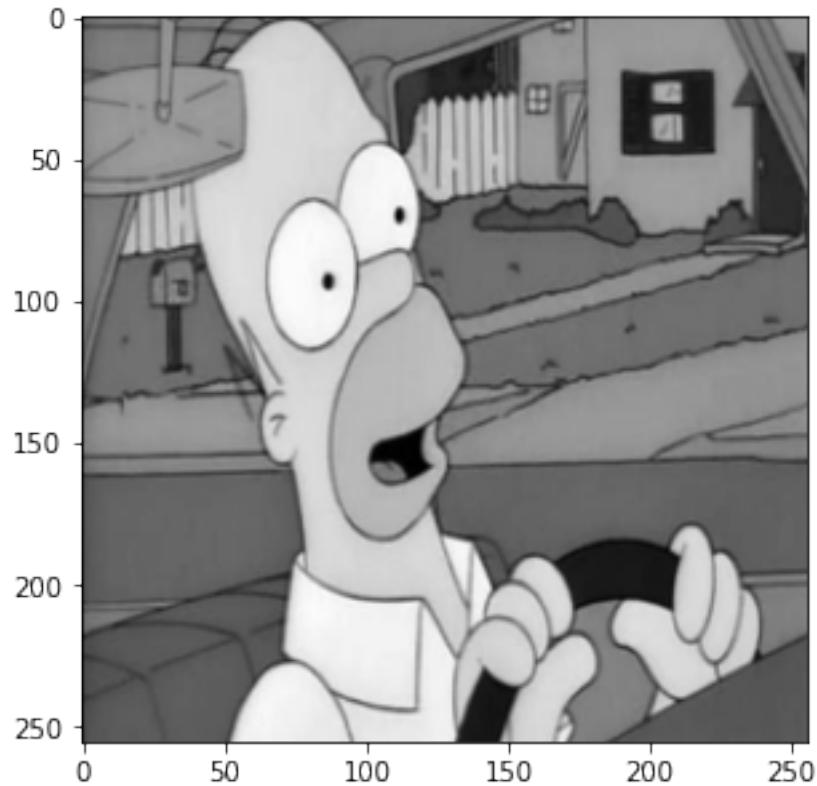
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B650>



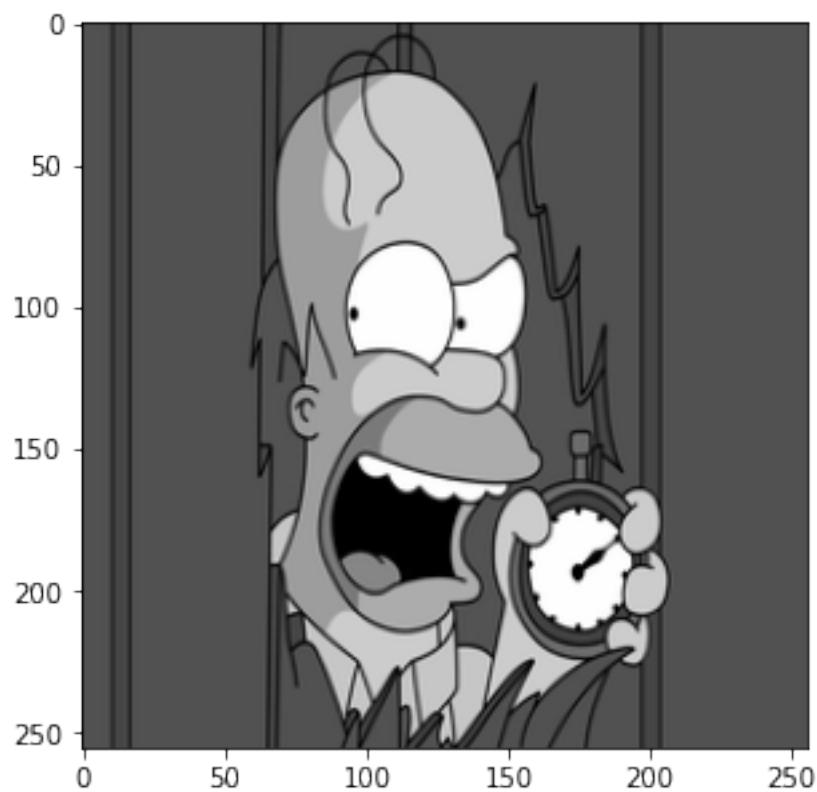
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B690>



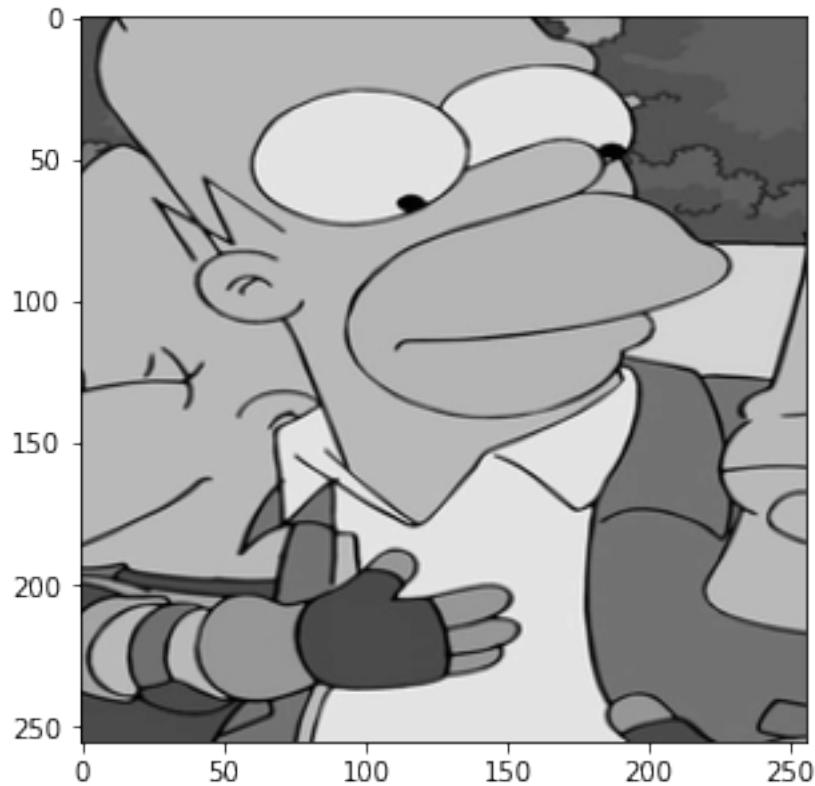
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B6D0>



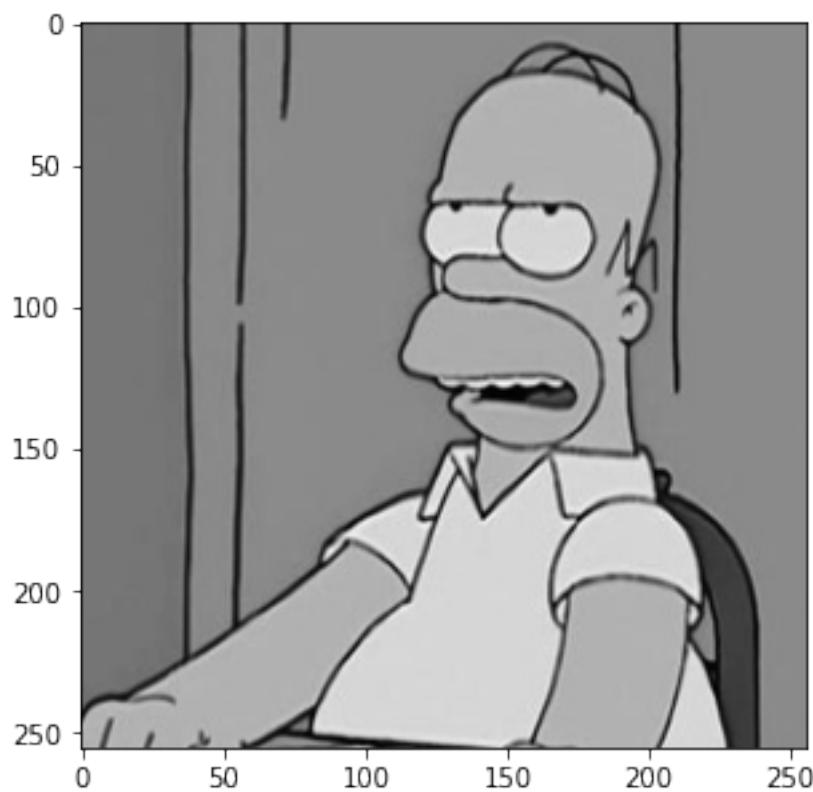
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B710>



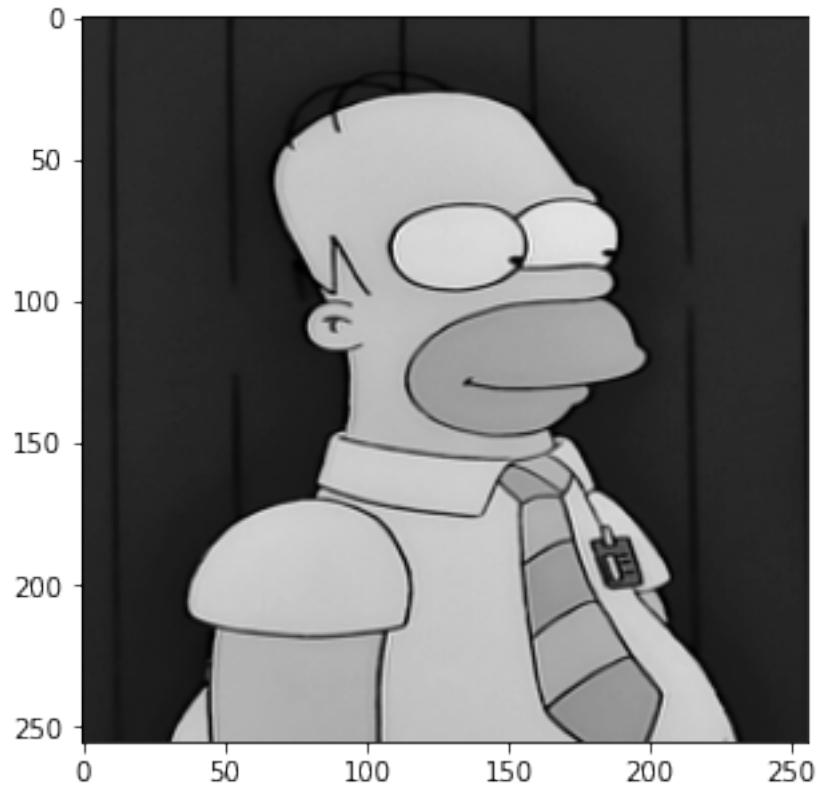
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B750>



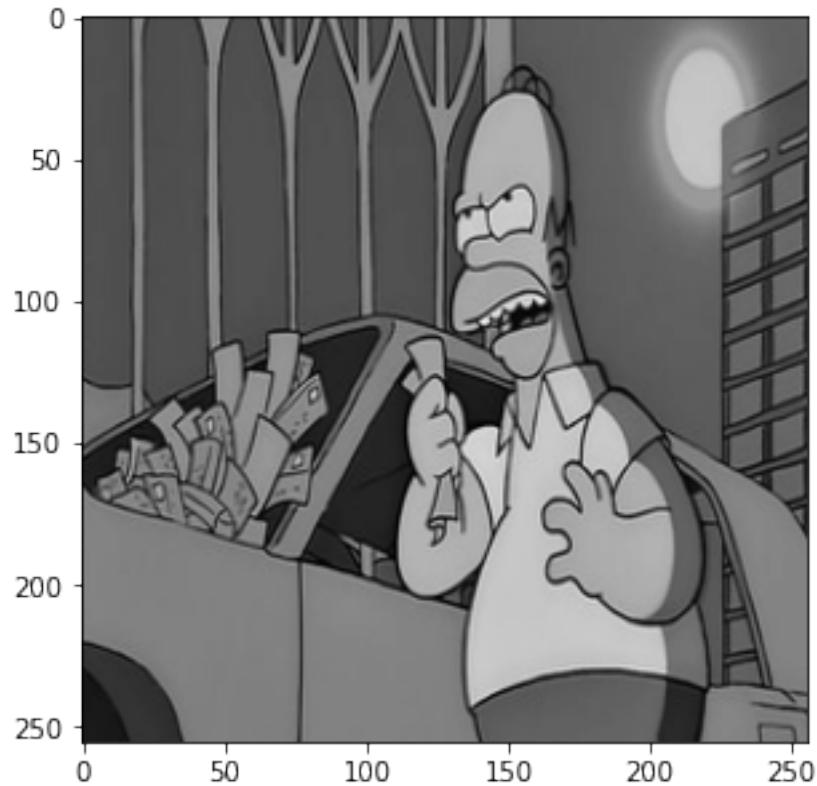
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B790>



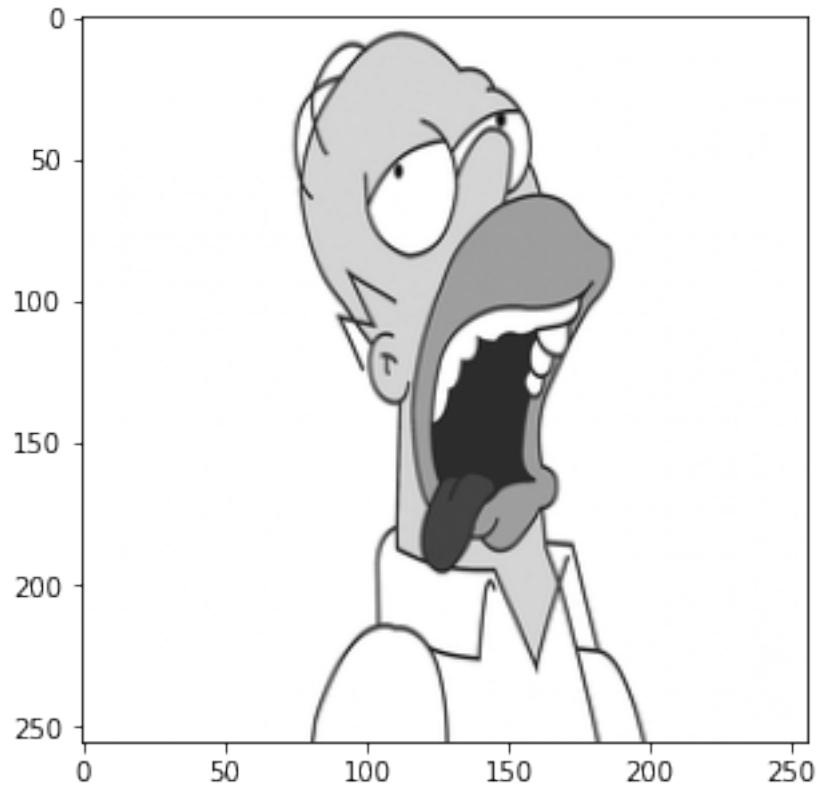
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B7D0>



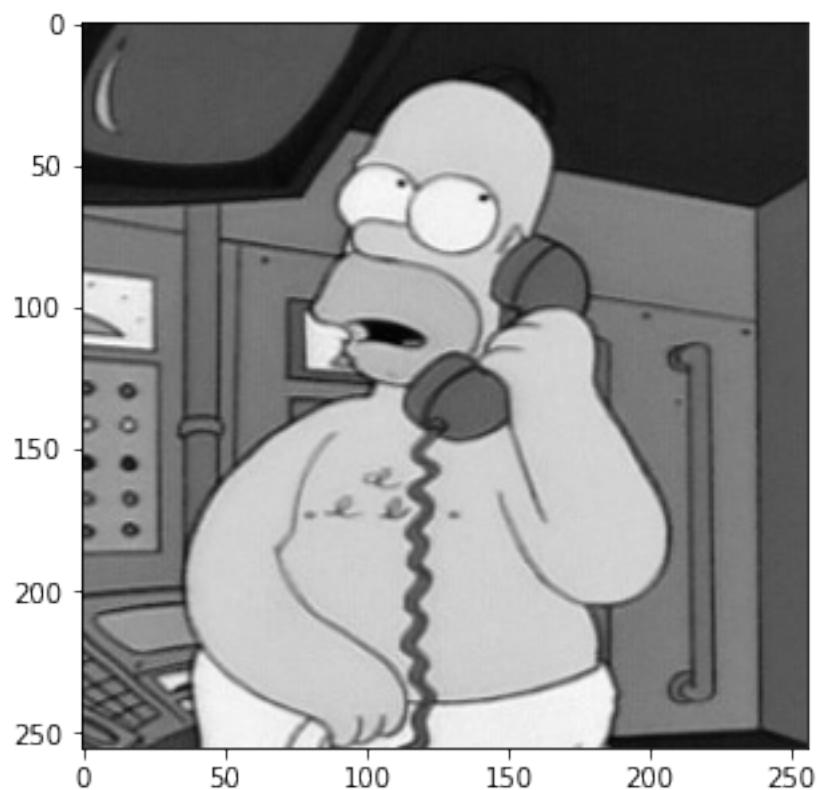
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B810>



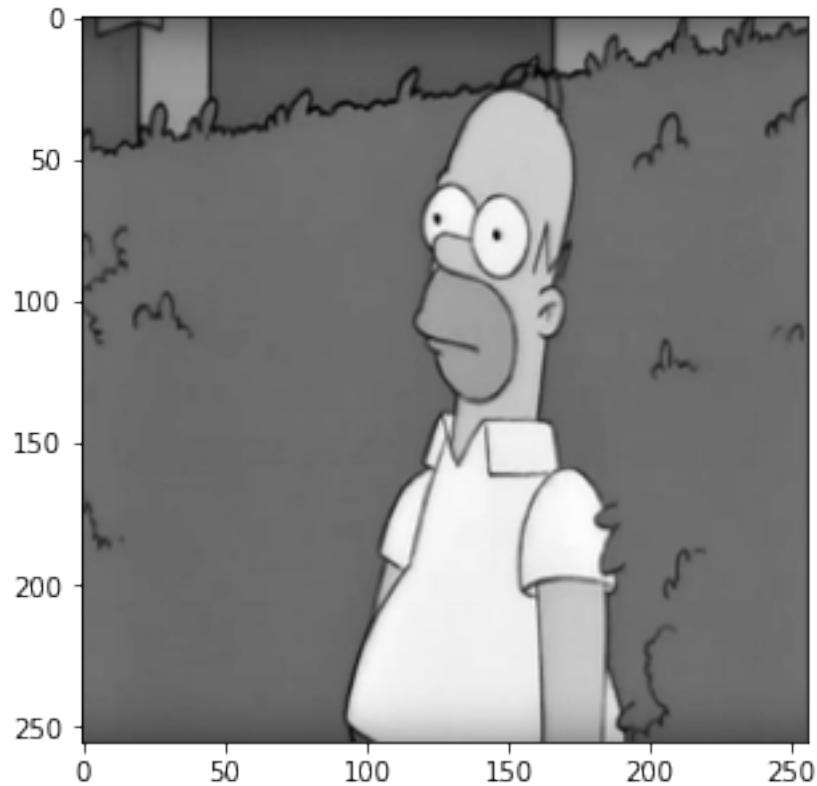
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B850>



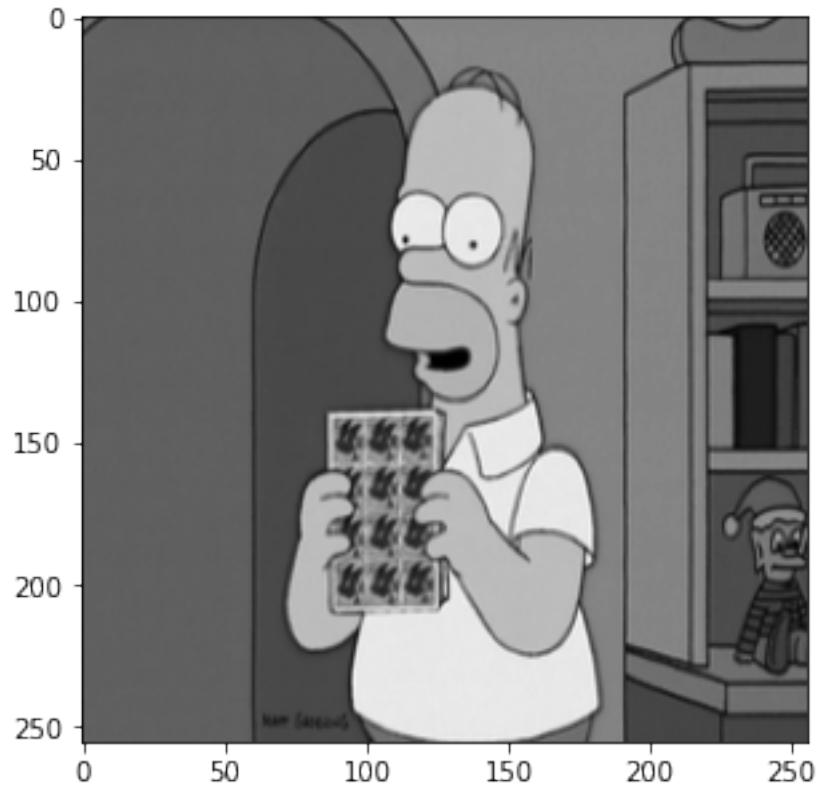
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B890>



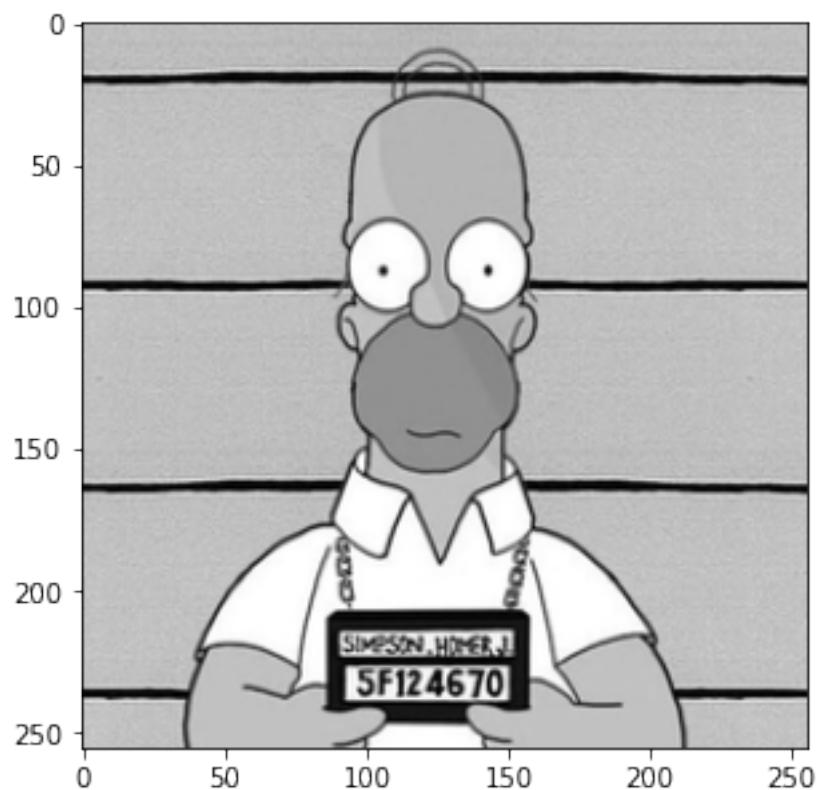
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B8D0>



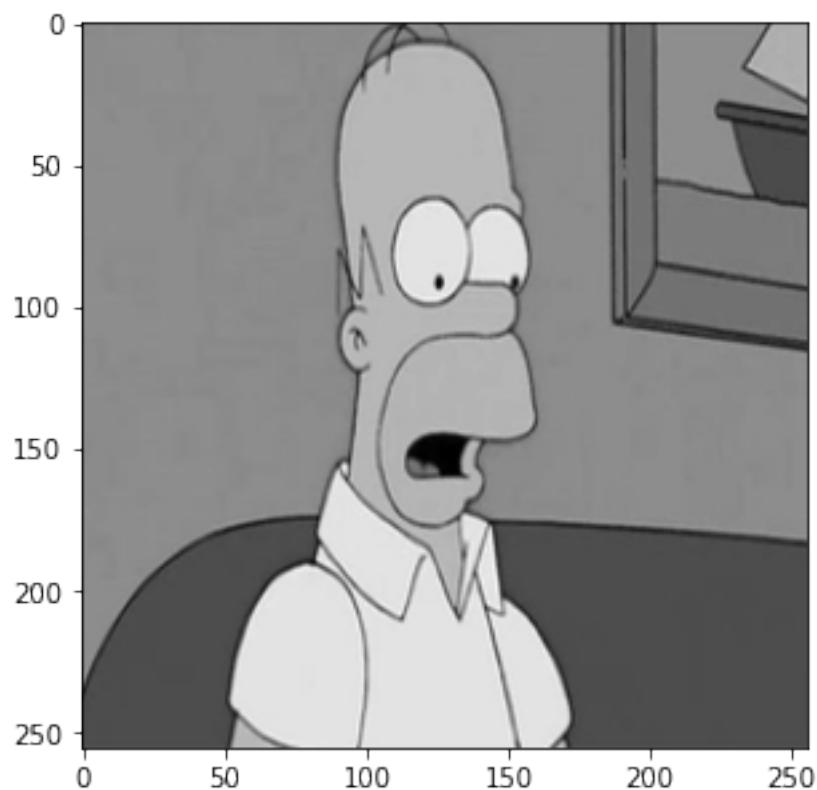
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B910>



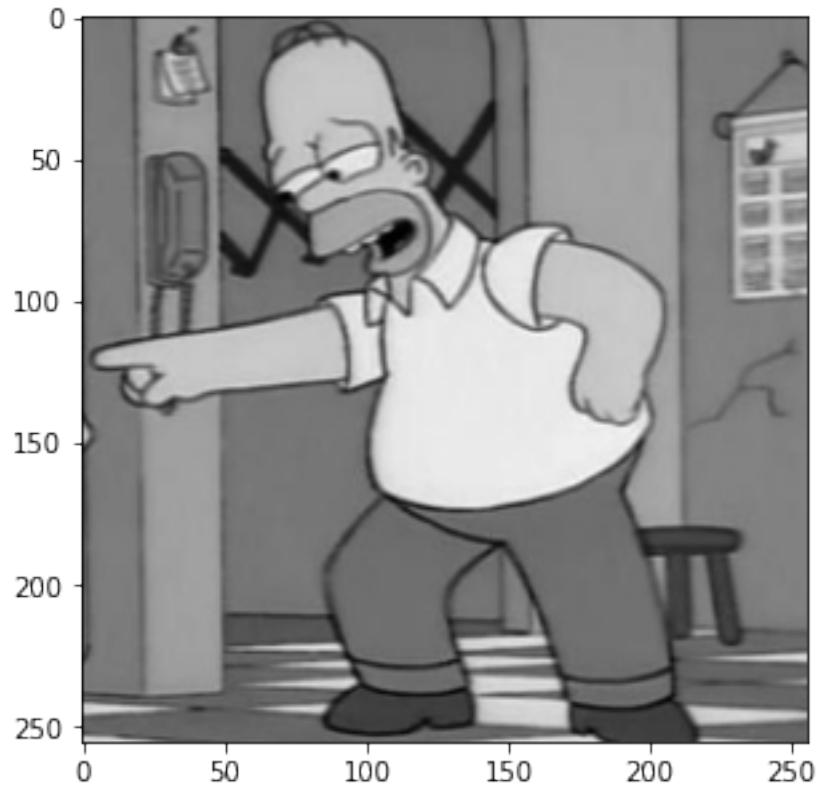
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B950>



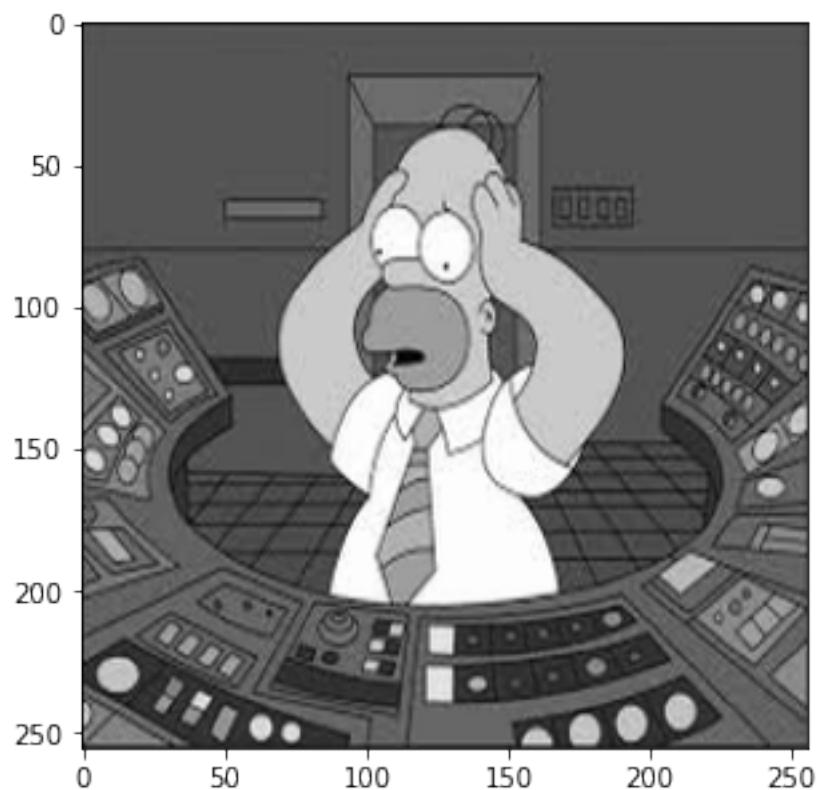
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B990>



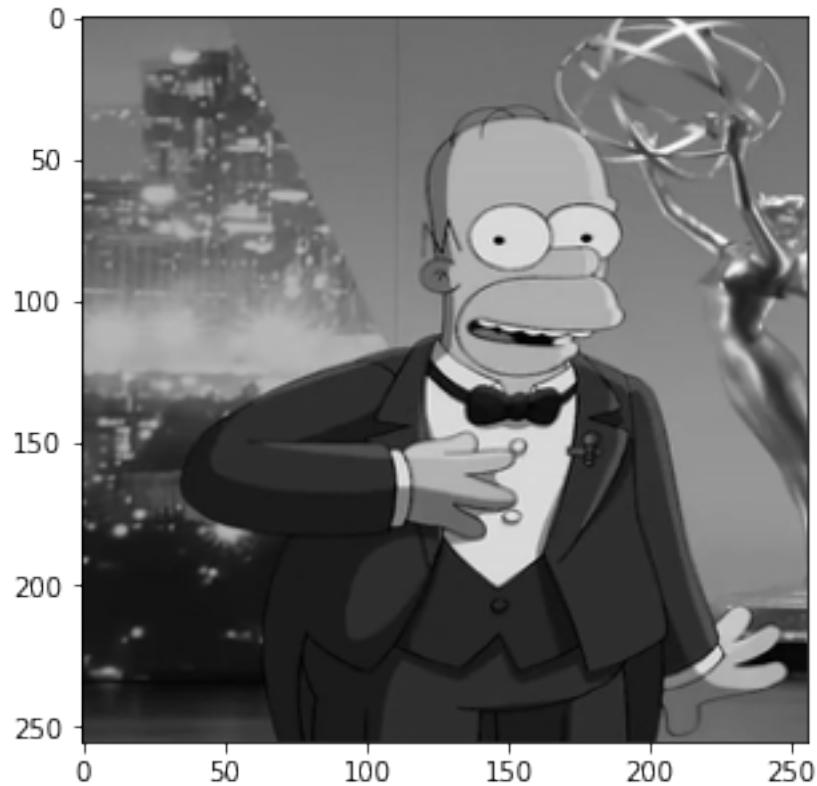
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978B9D0>



<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BA10>



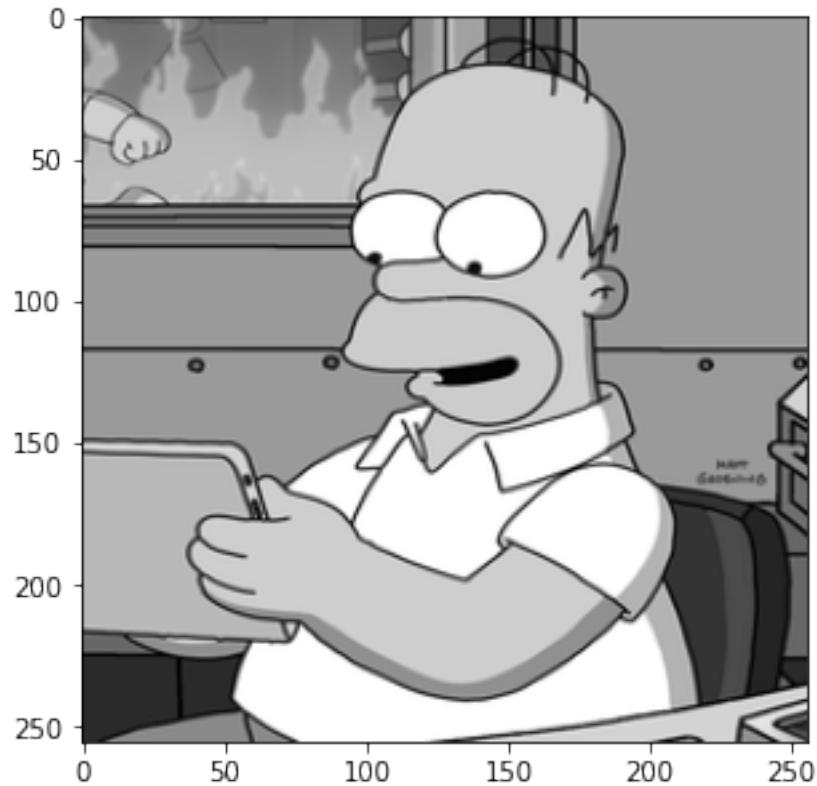
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BA50>



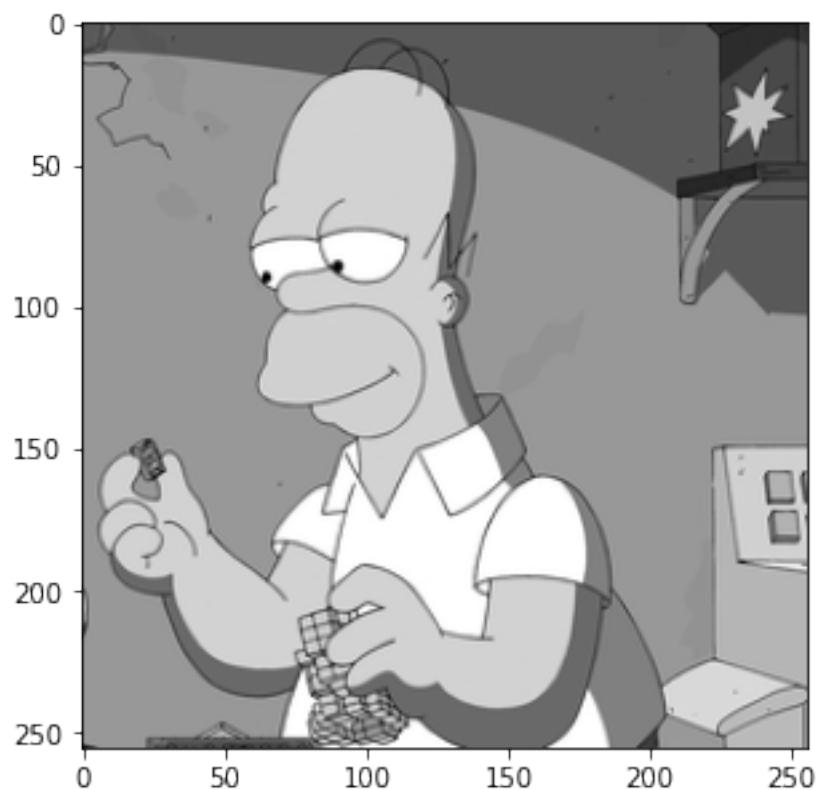
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BA90>



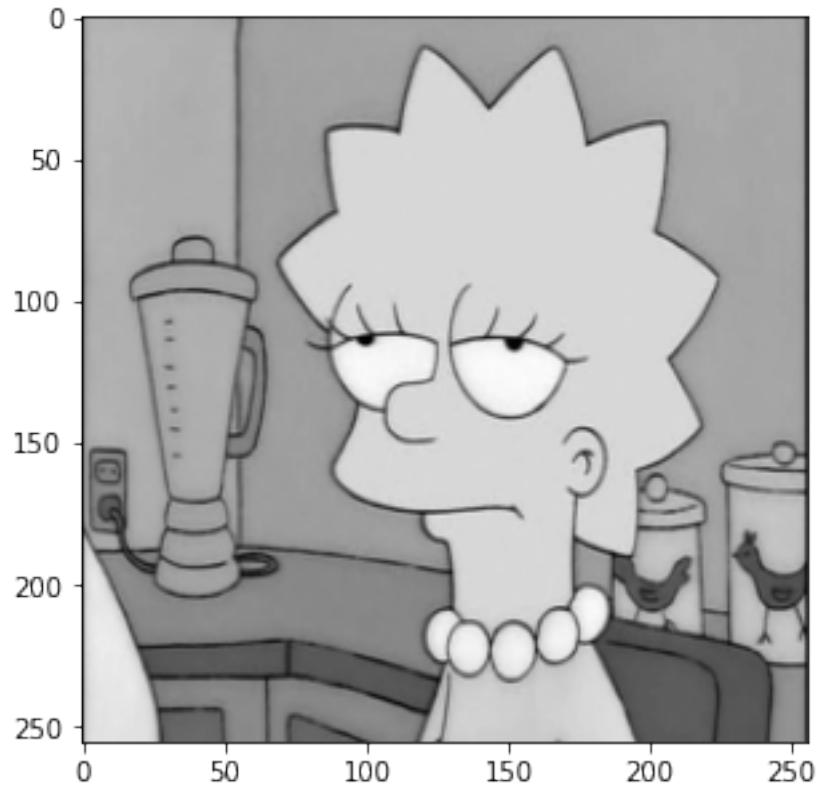
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BAD0>



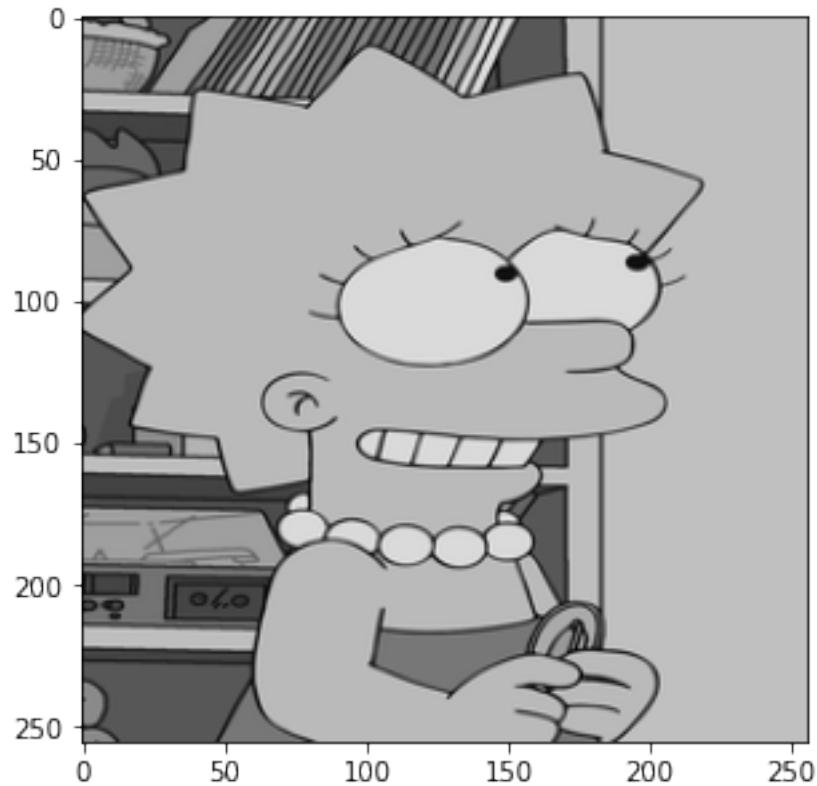
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BB10>



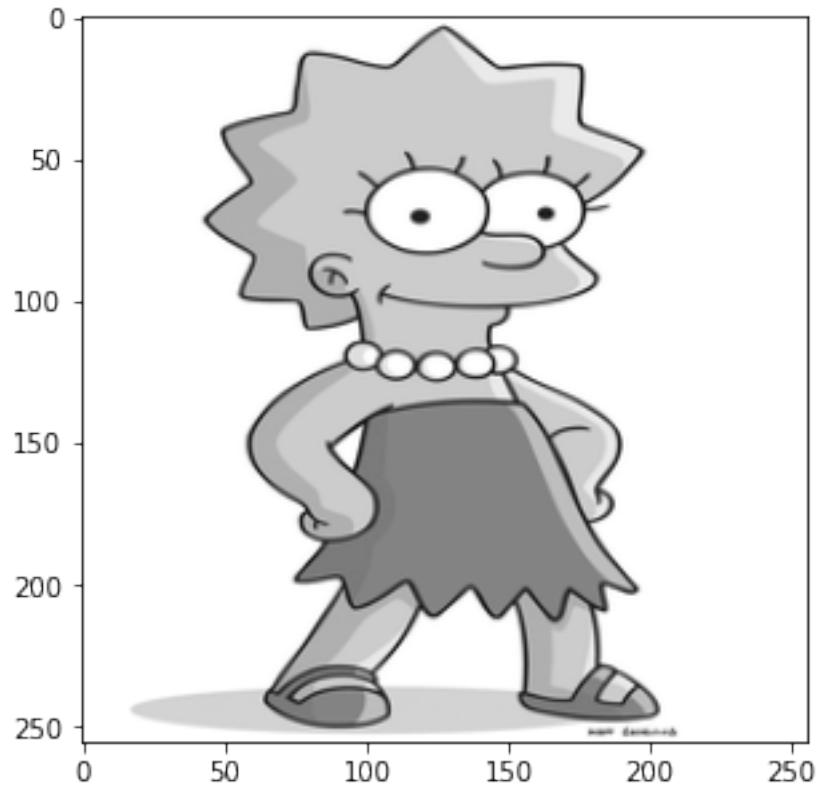
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2750>



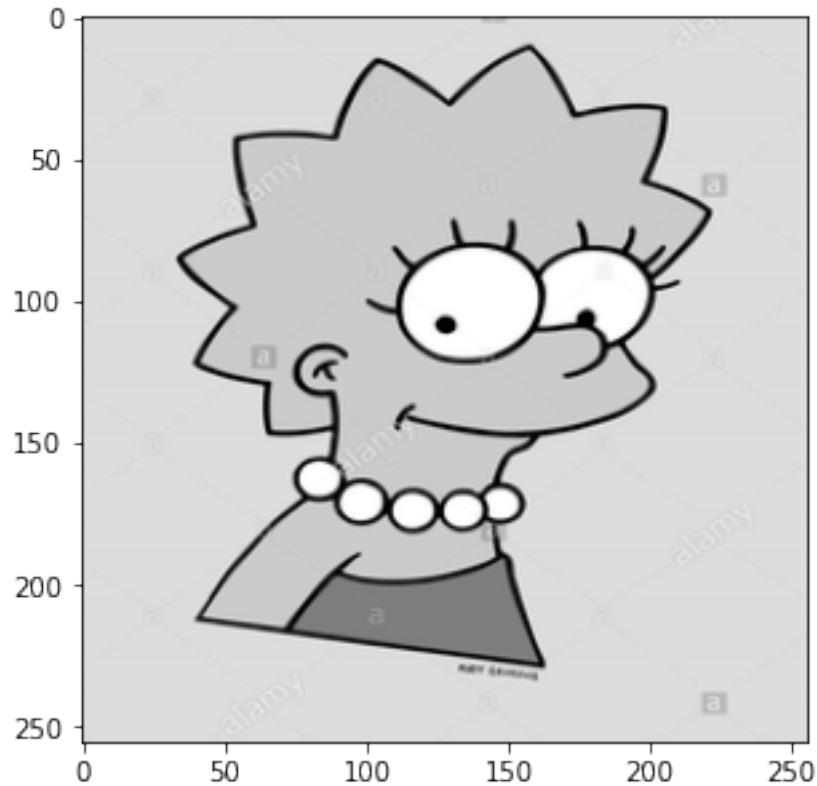
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2650>



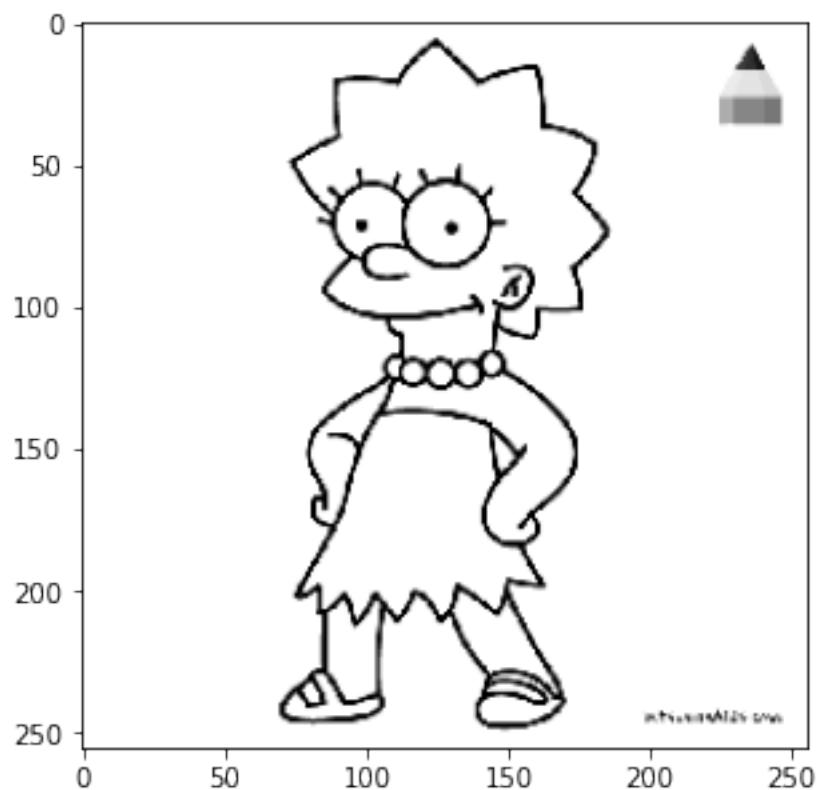
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B27D0>



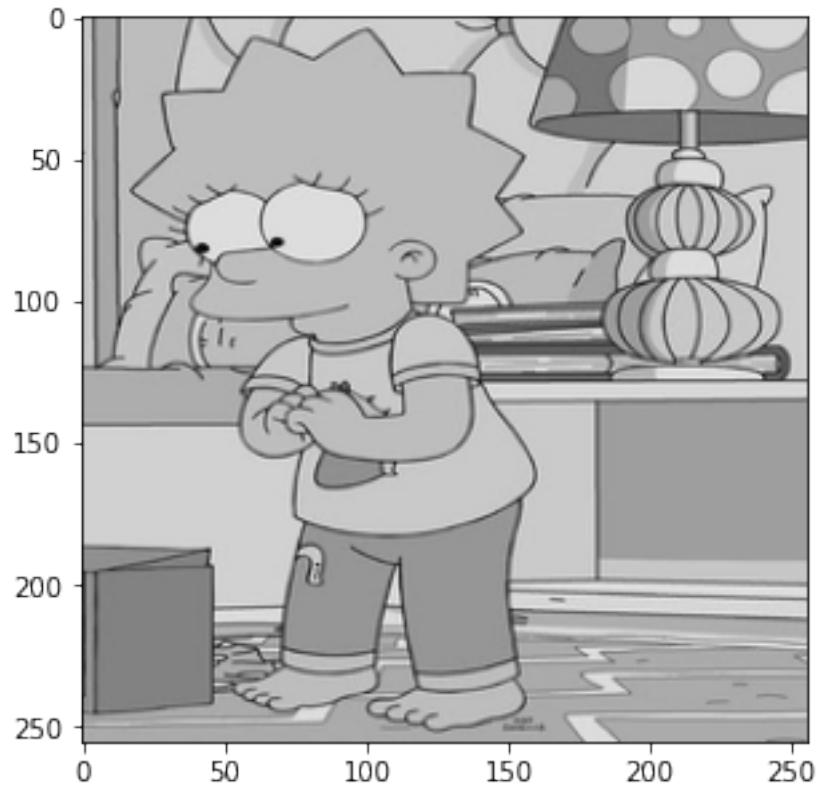
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2810>
```



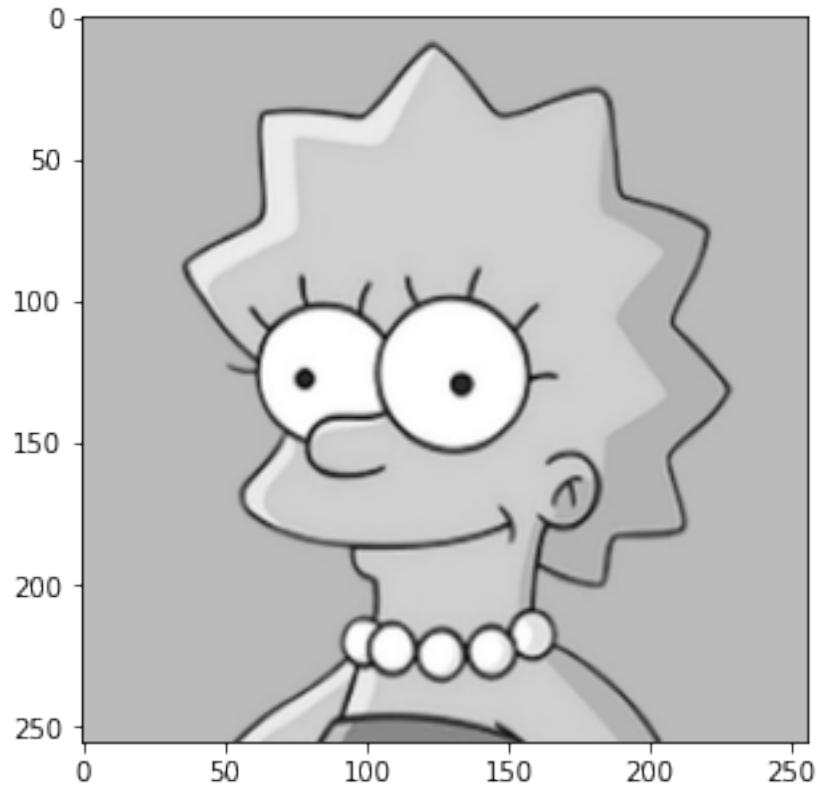
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2790>



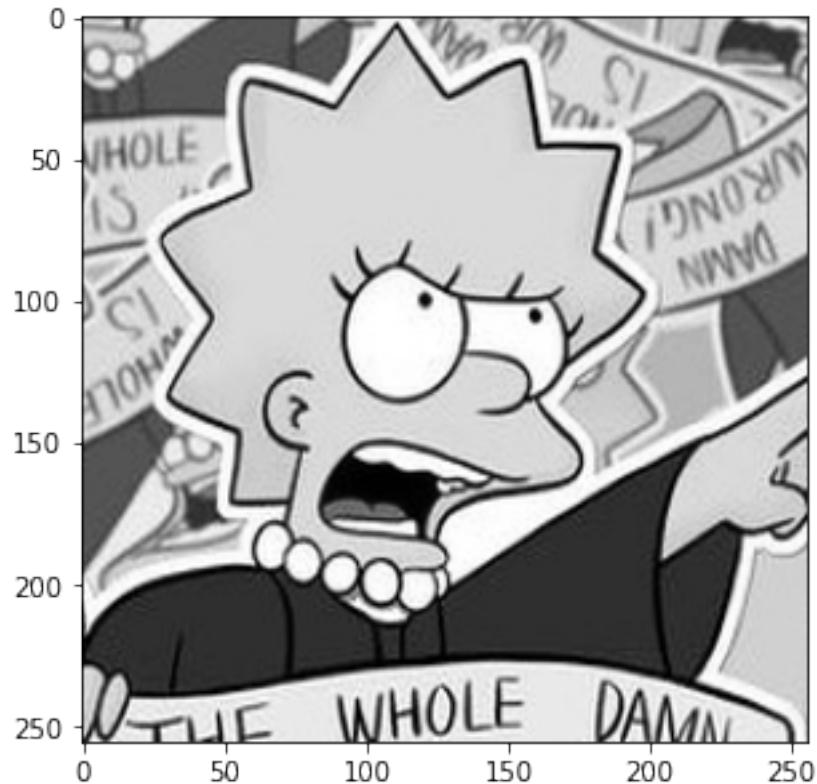
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2FD0>



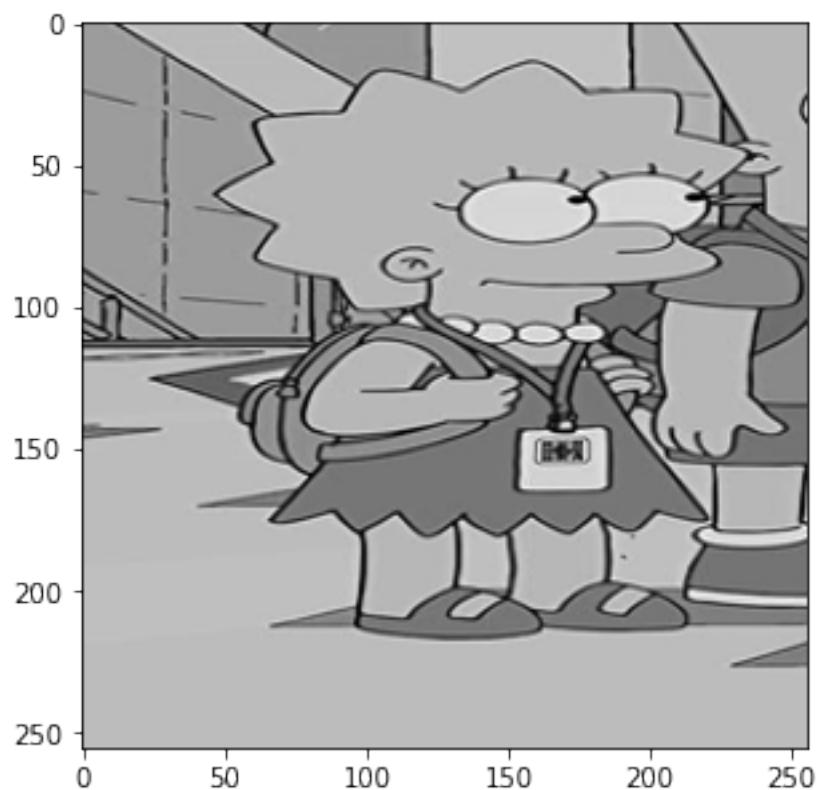
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2F50>



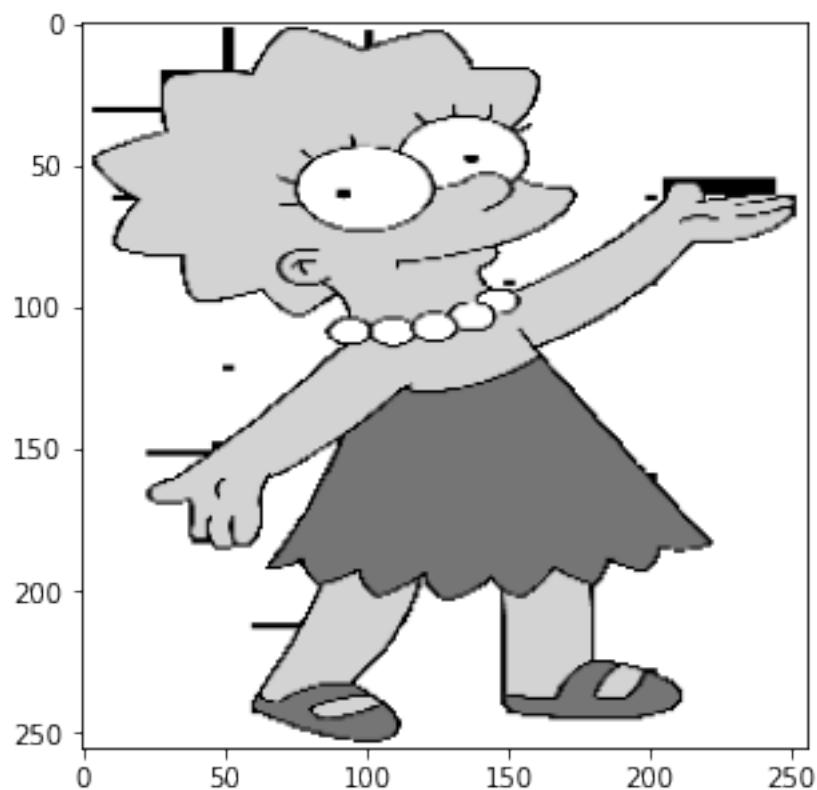
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2F10>



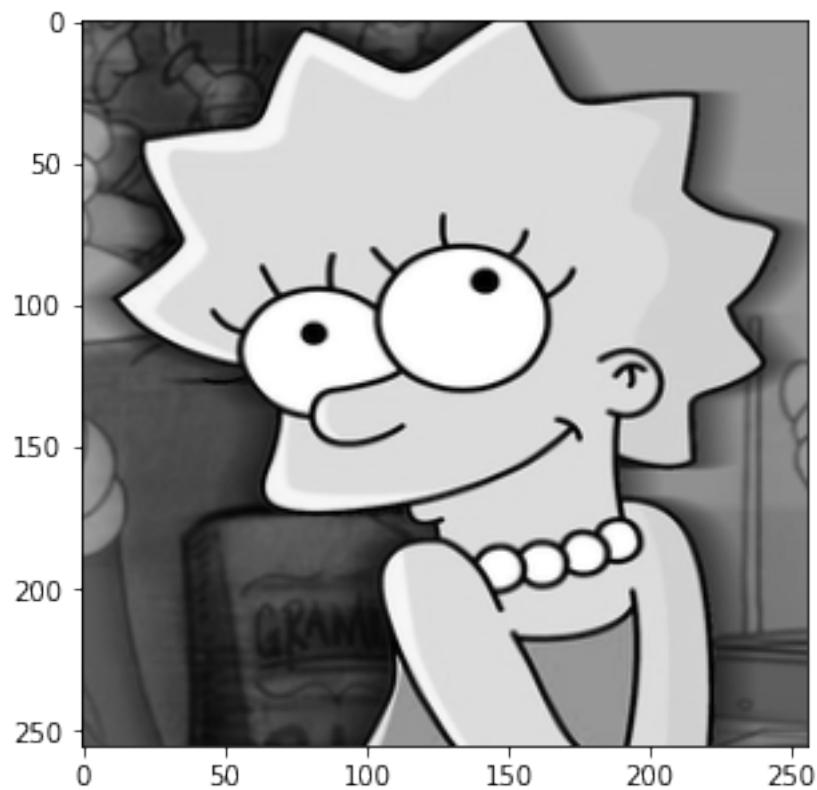
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2F90>



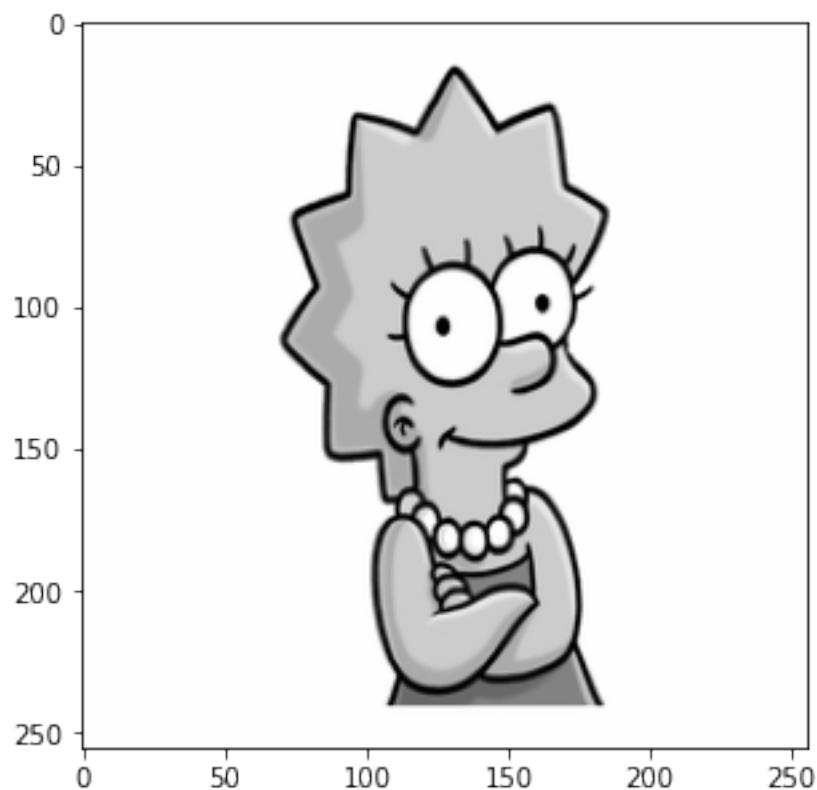
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2710>



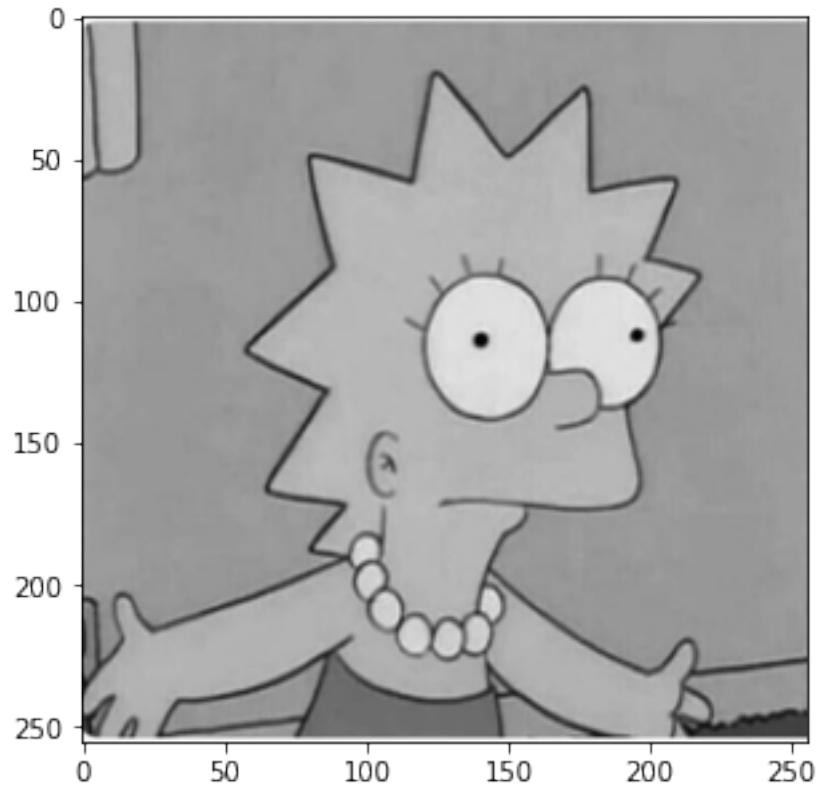
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2E50>



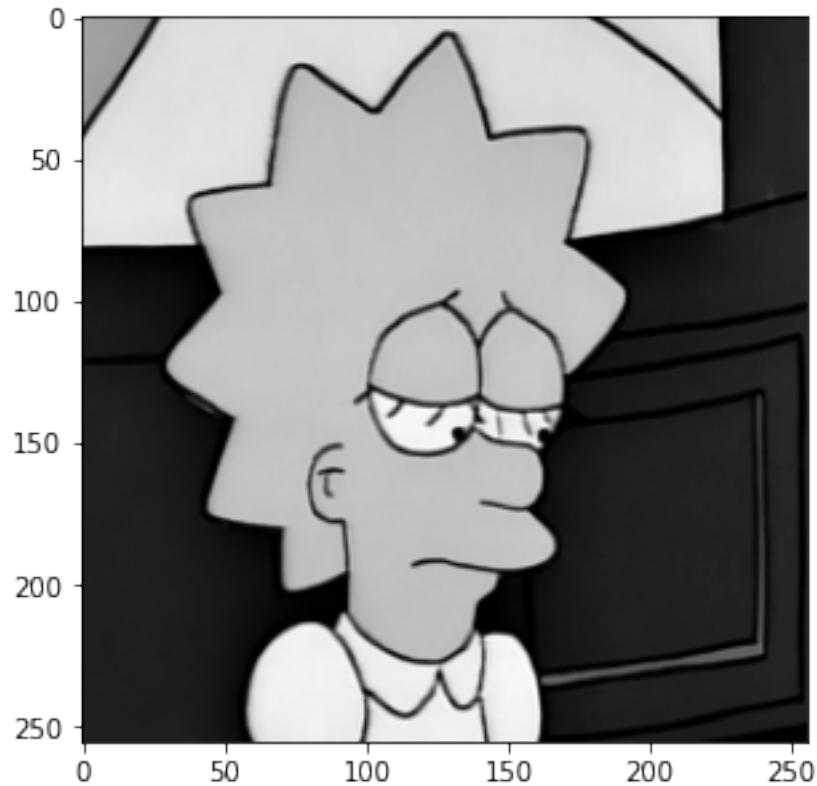
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2ED0>



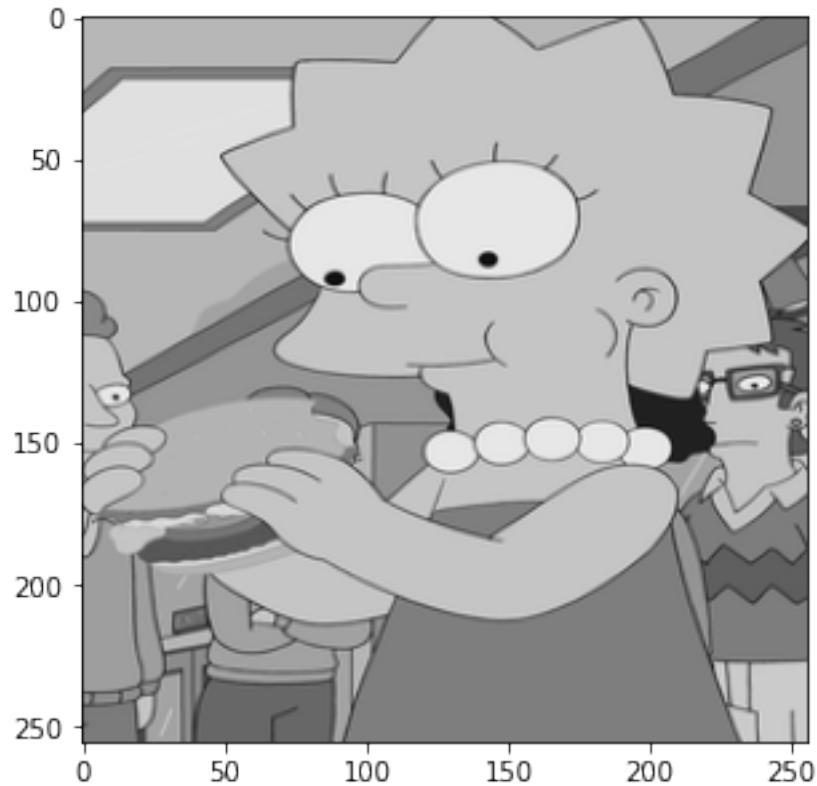
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2A10>



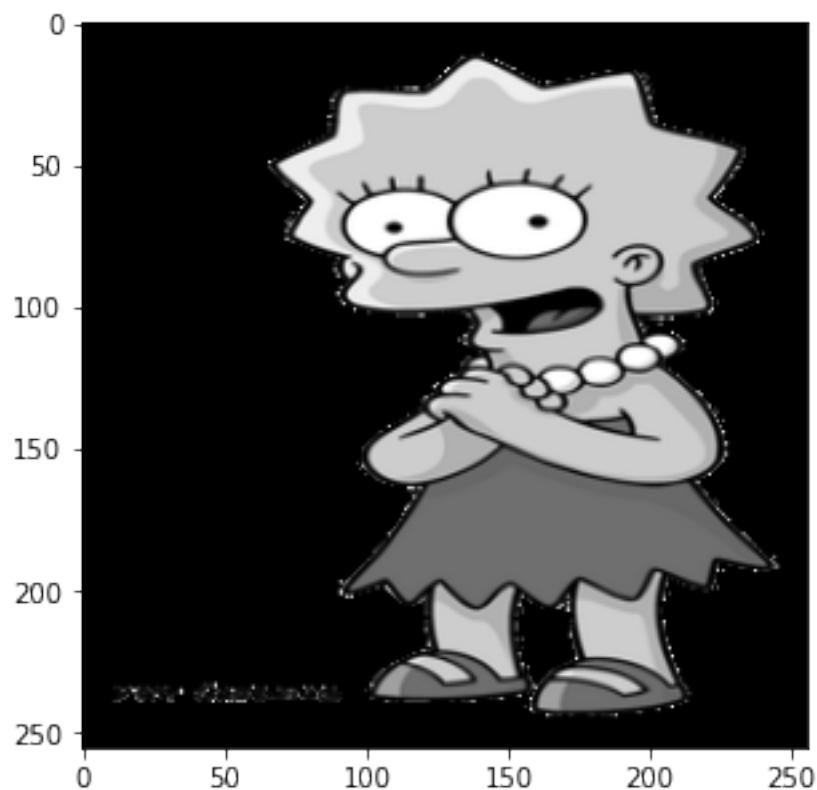
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2890>



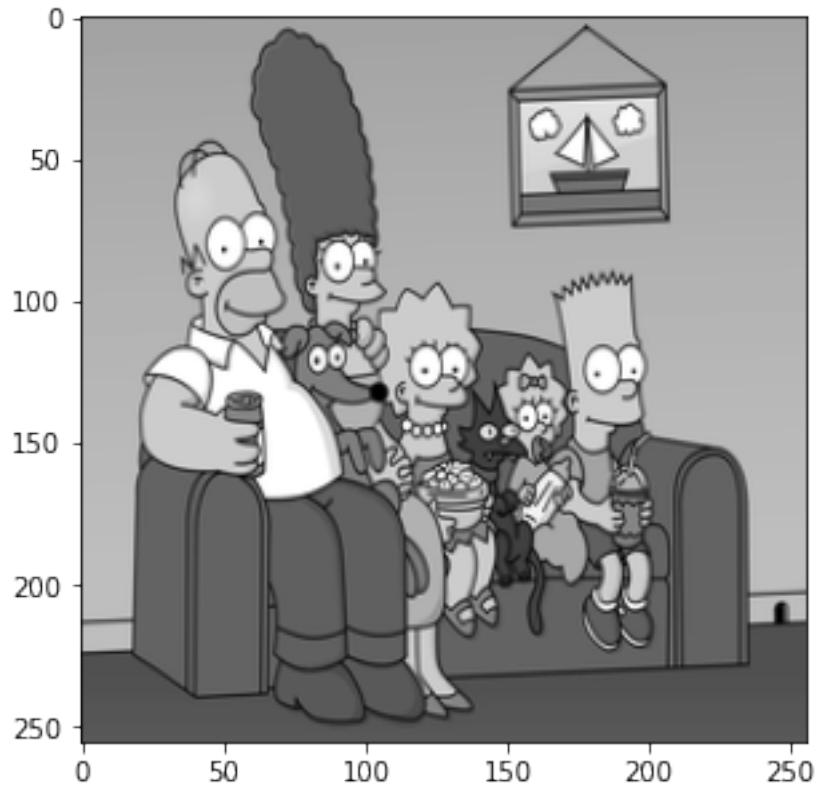
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2610>



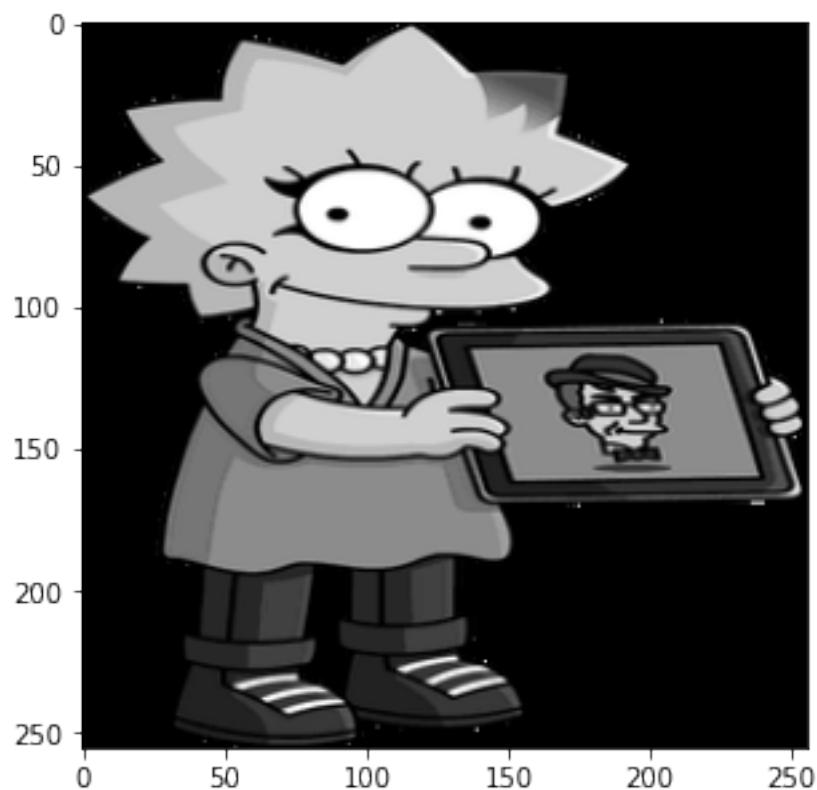
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B25D0>



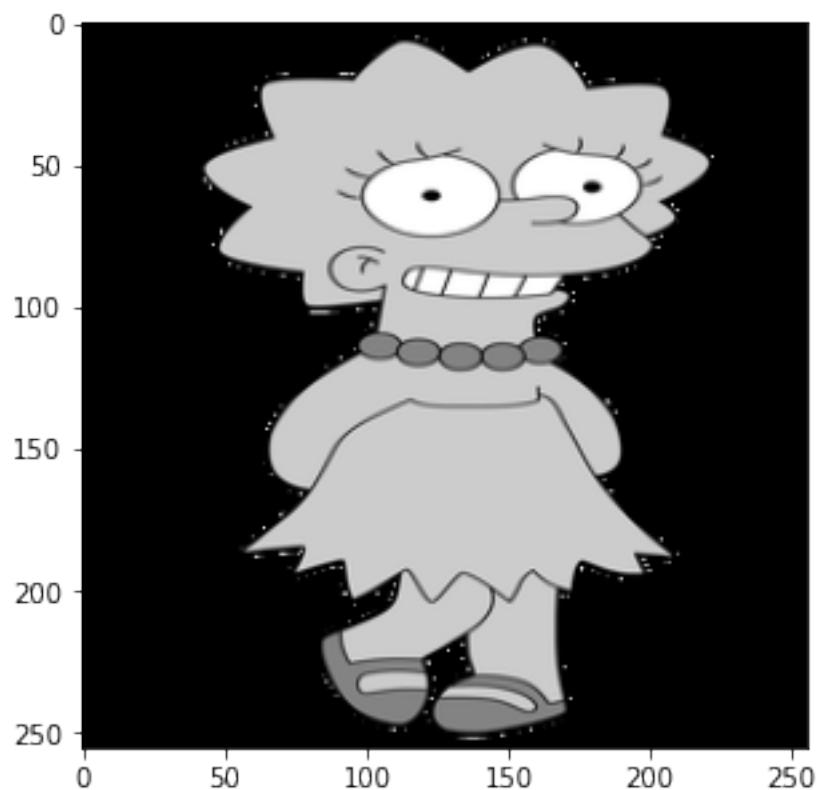
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2490>



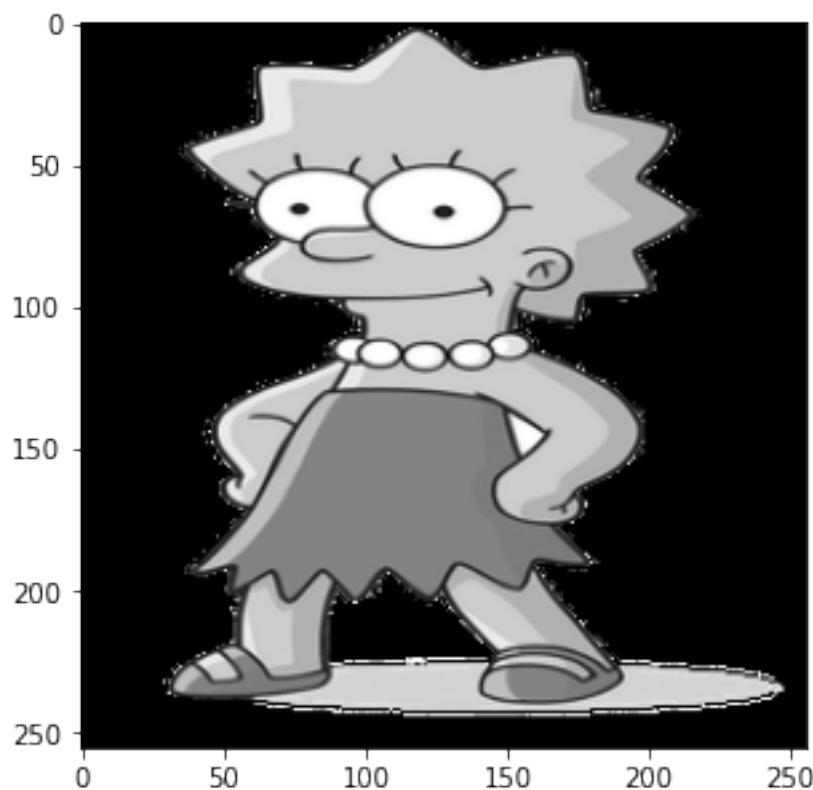
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2550>



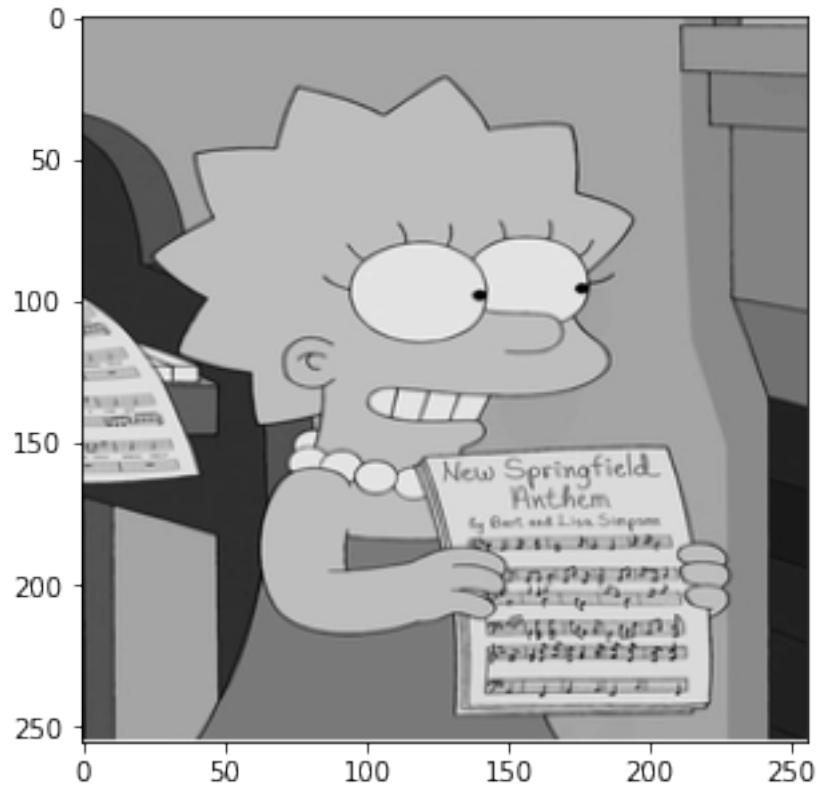
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2590>



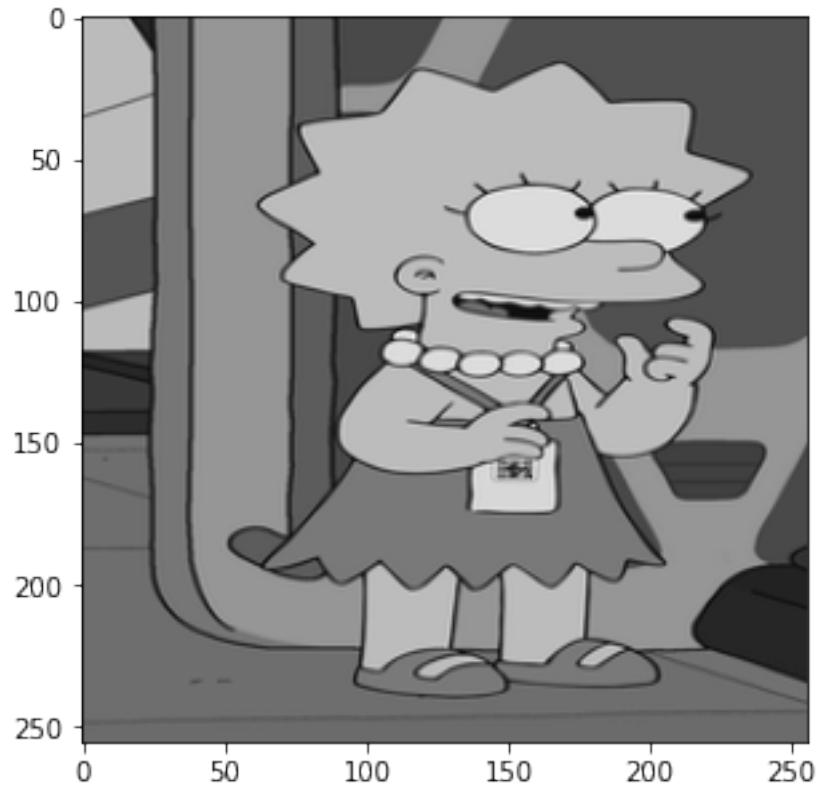
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2510>



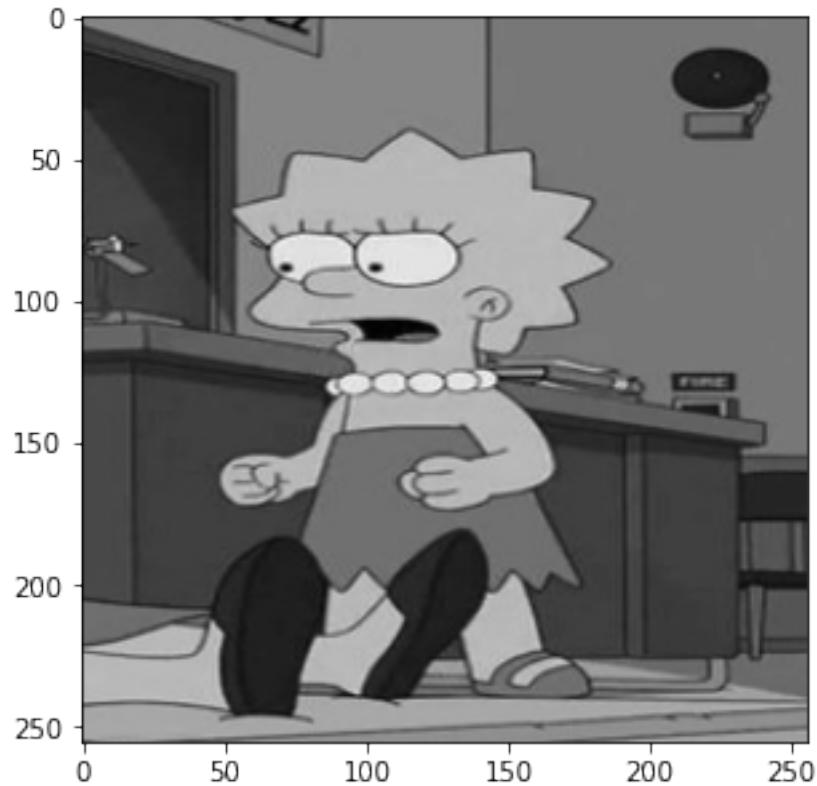
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B24D0>



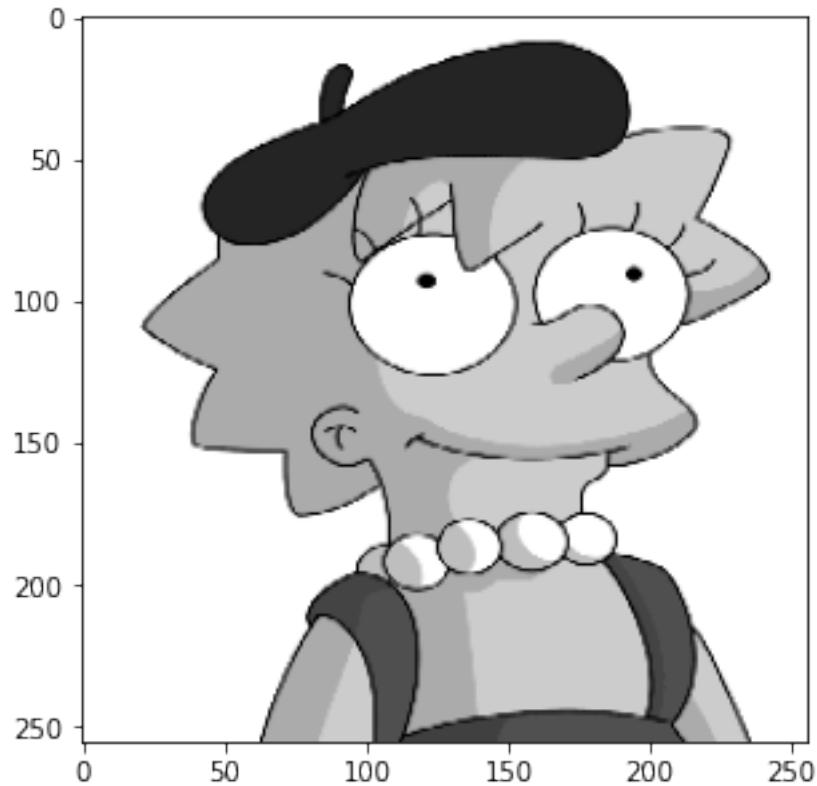
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B22D0>



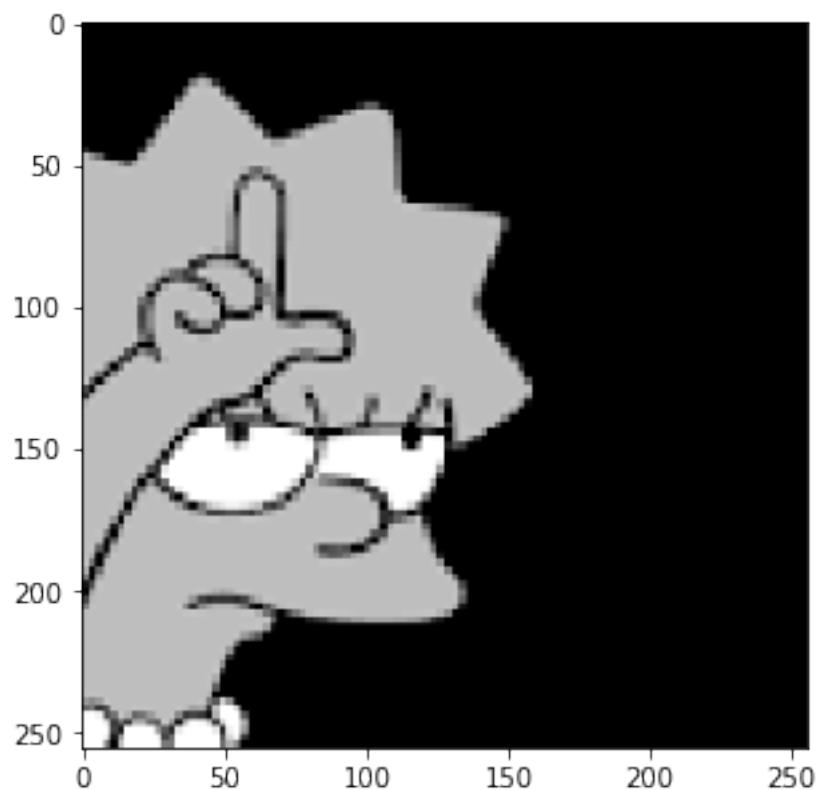
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2450>



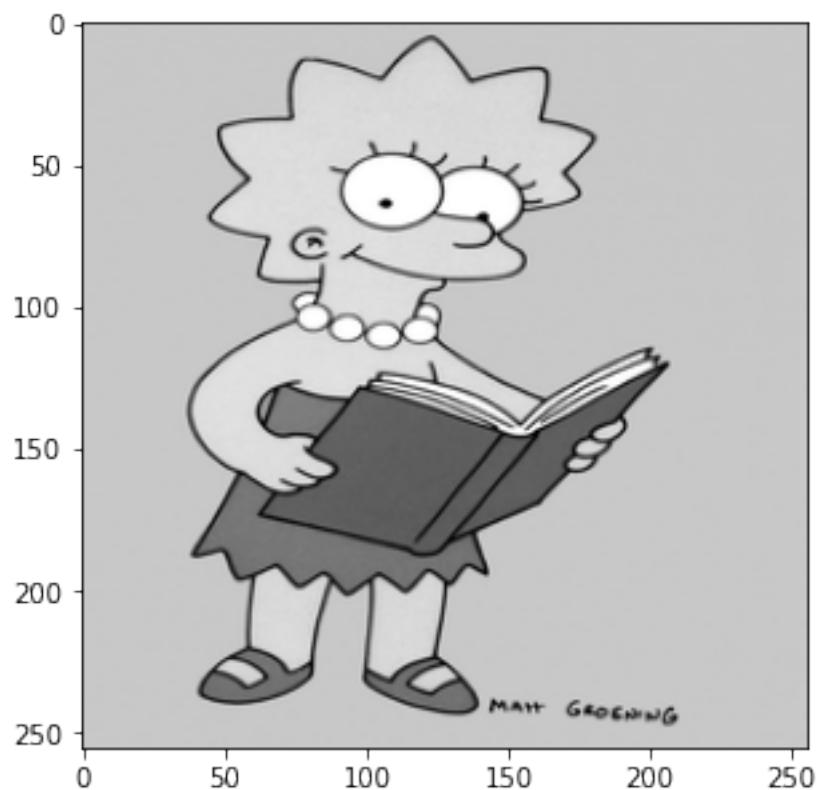
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2410>



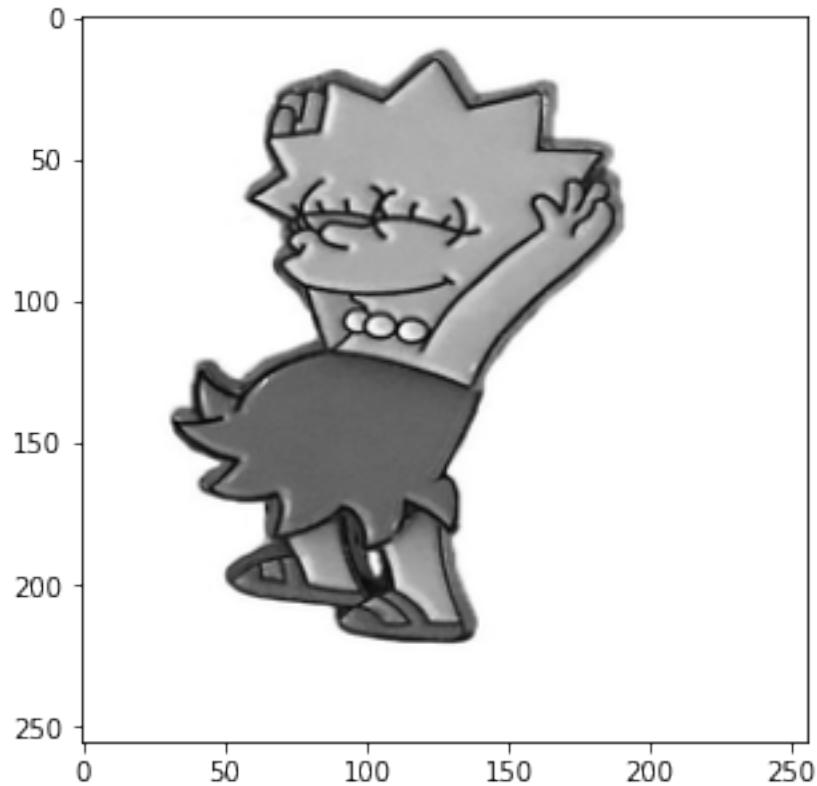
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2350>
```



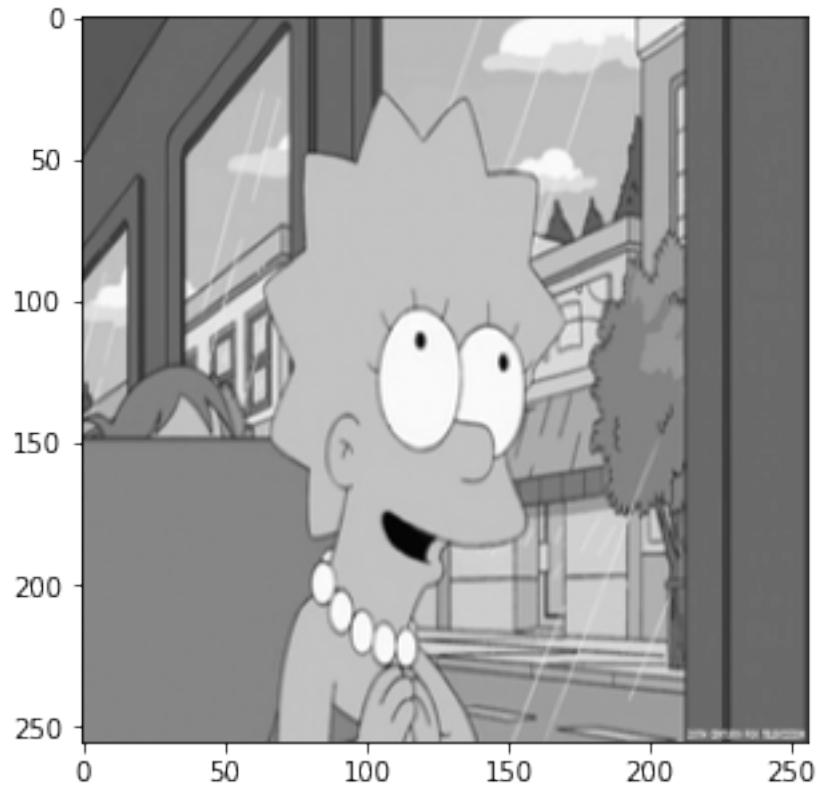
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2250>



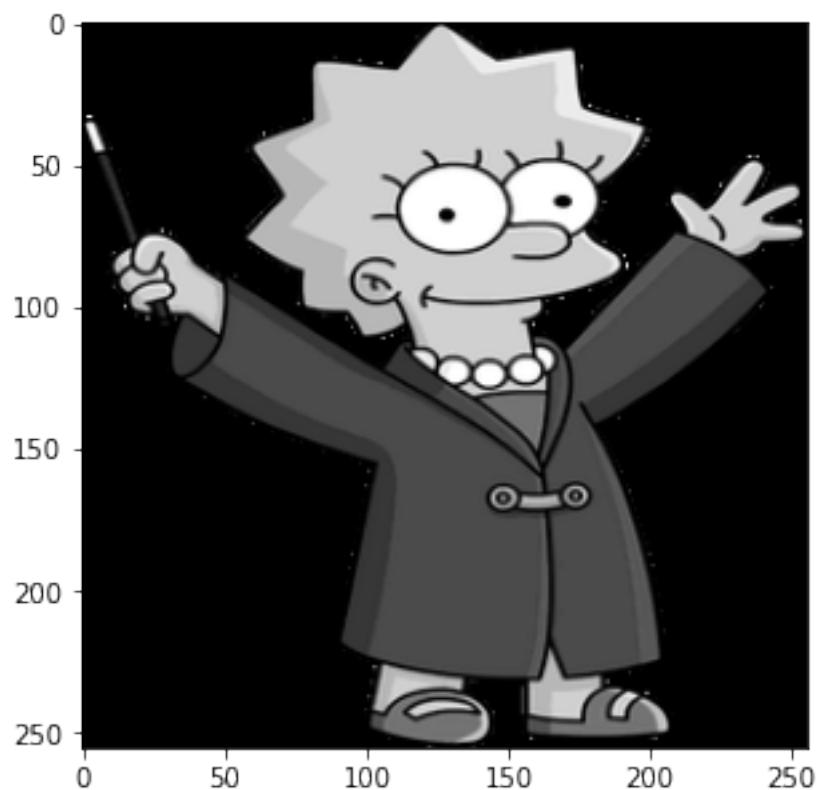
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2290>



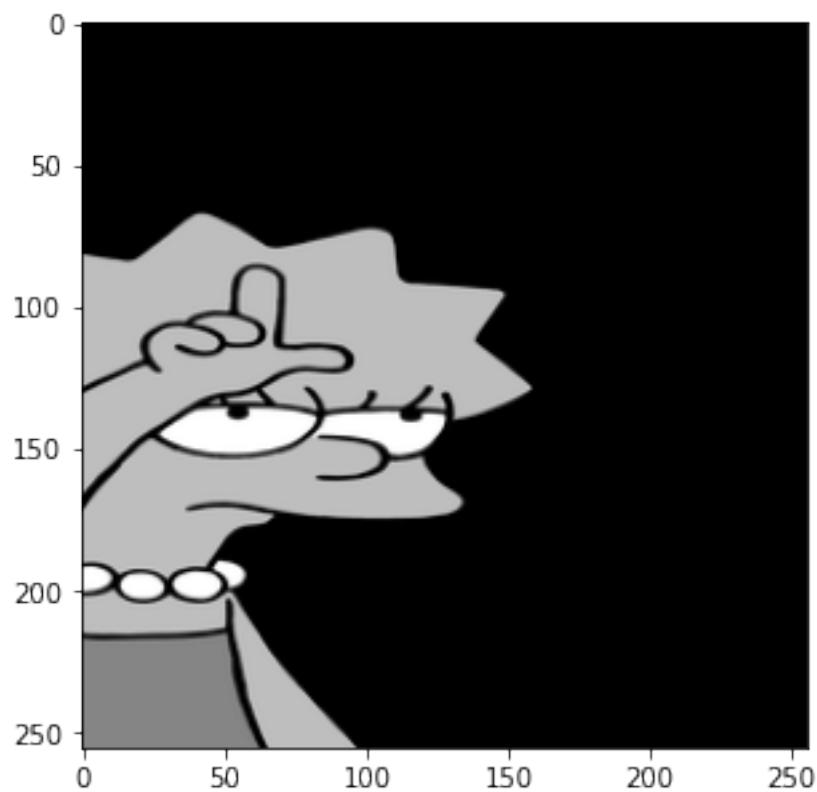
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2390>



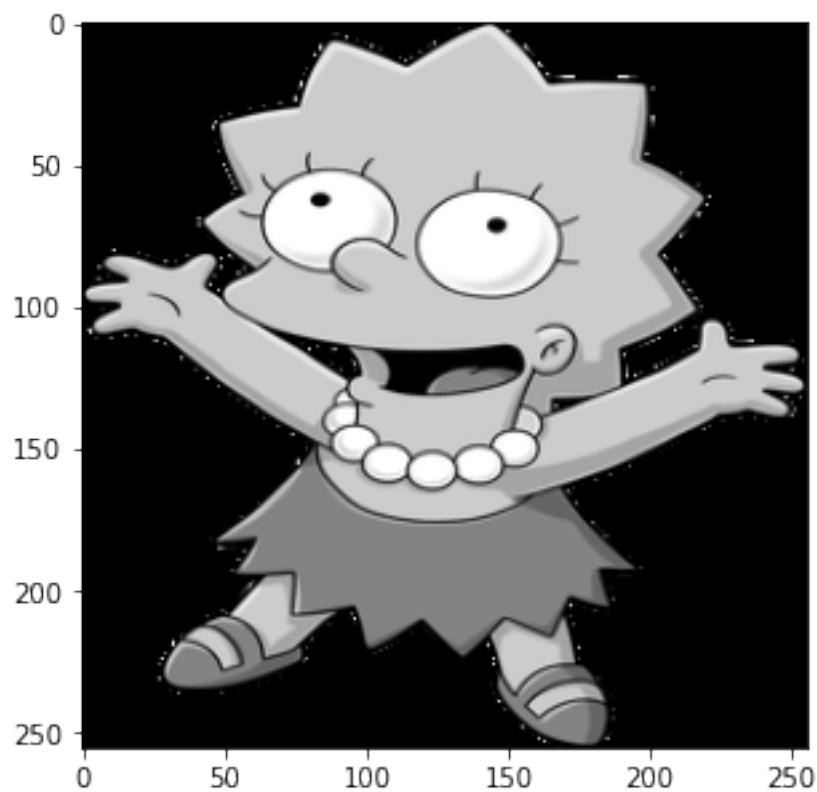
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2210>



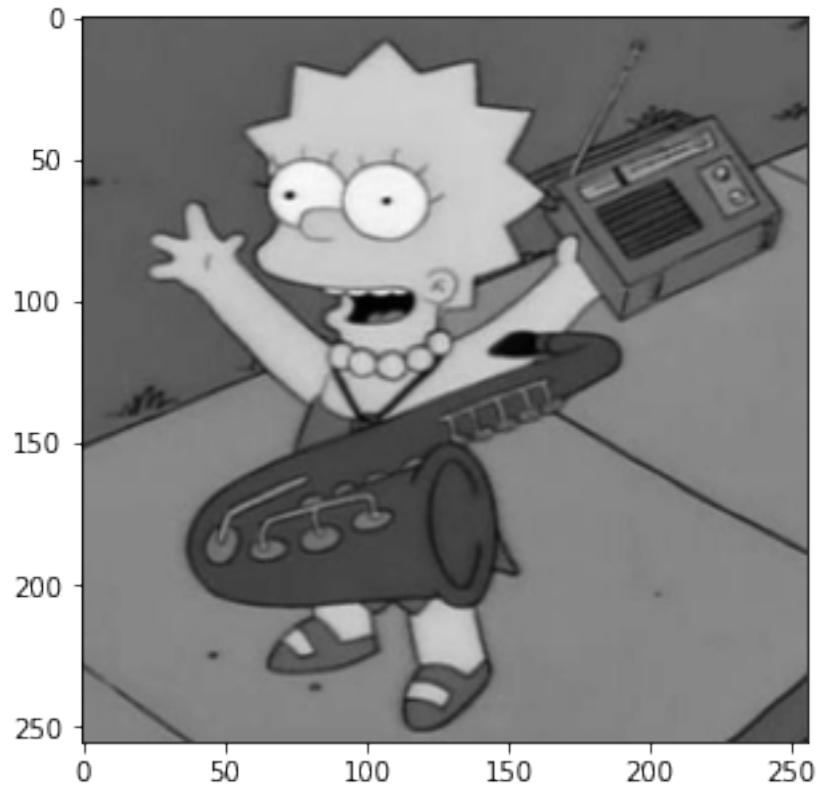
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2190>



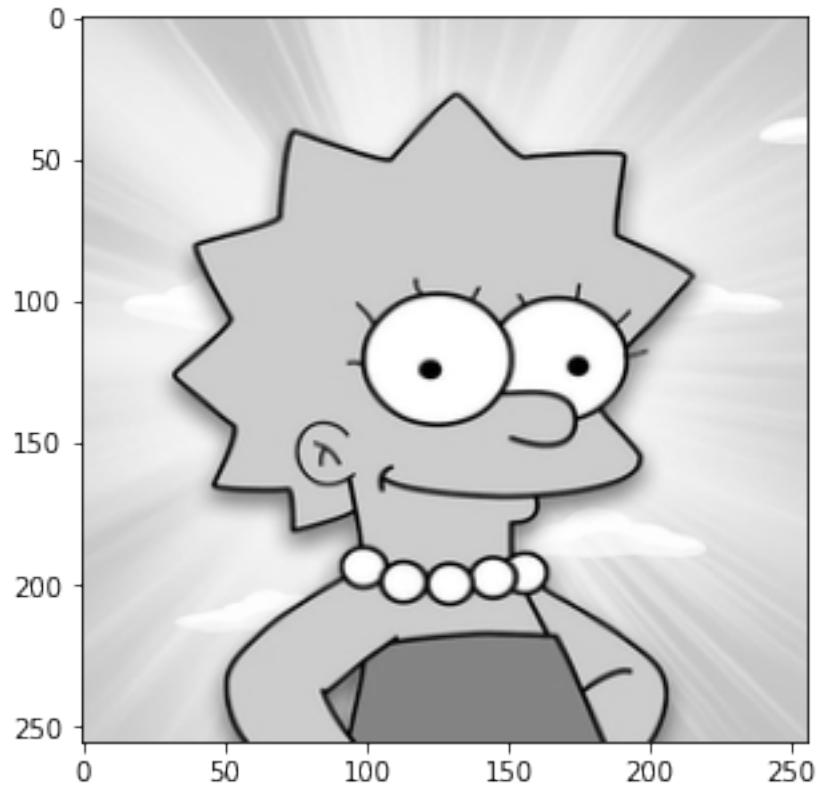
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B23D0>



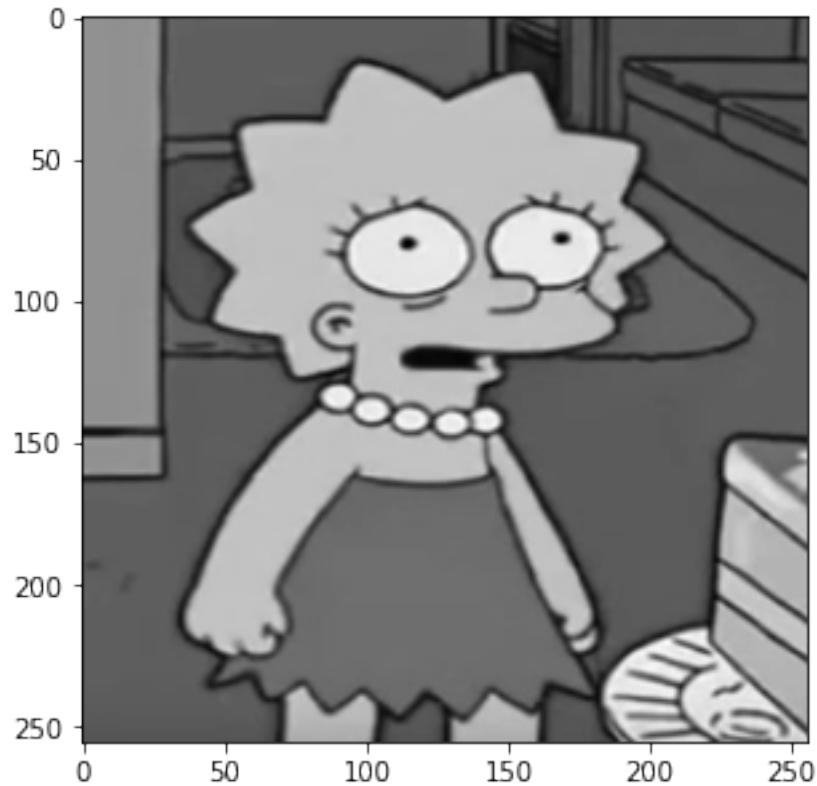
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2310>



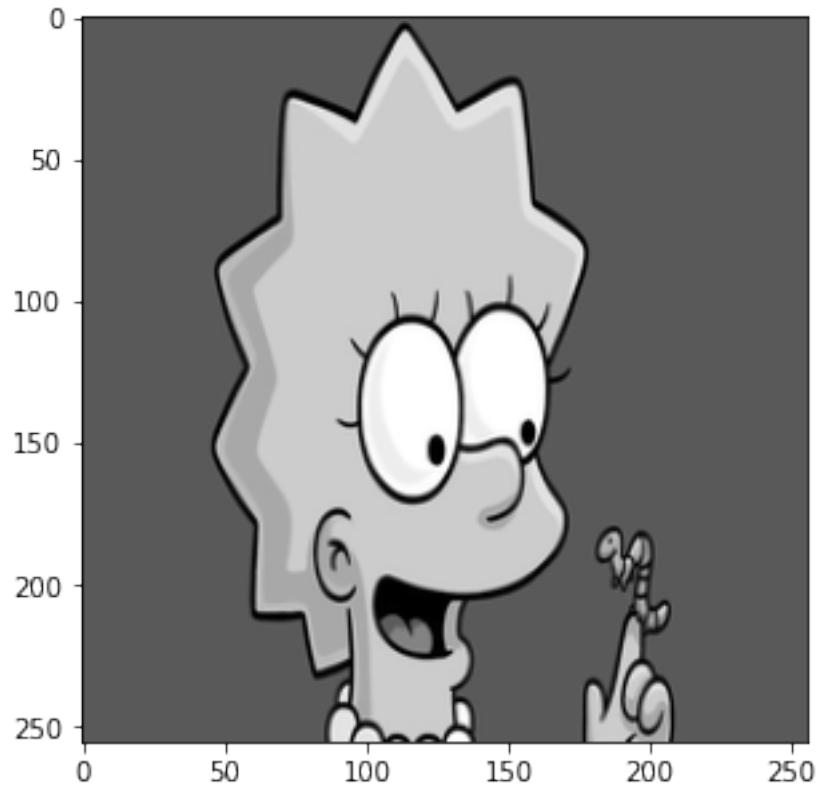
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795050>



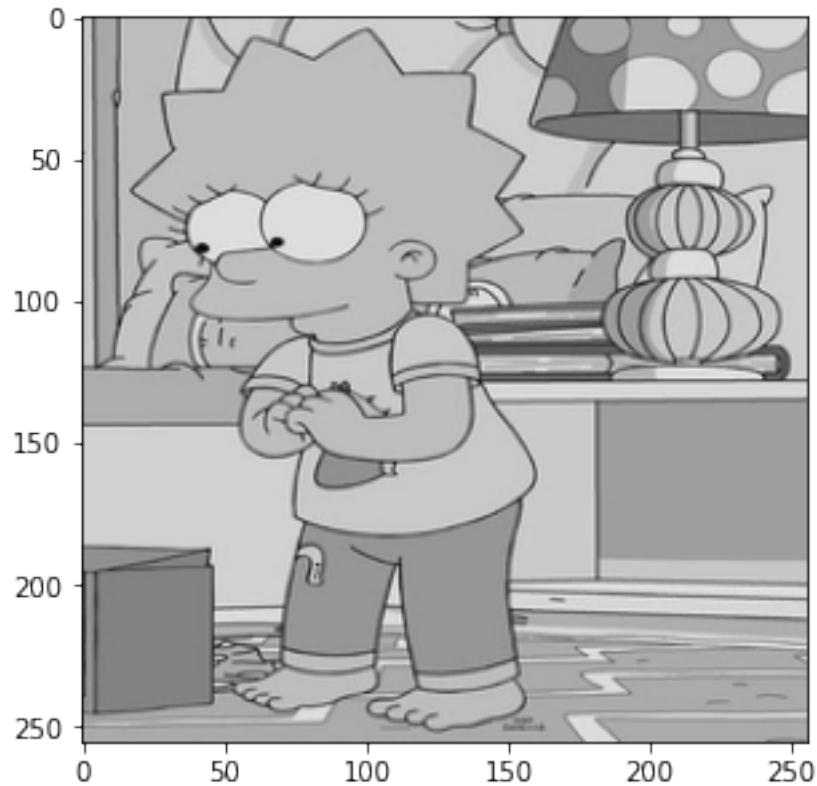
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795090>



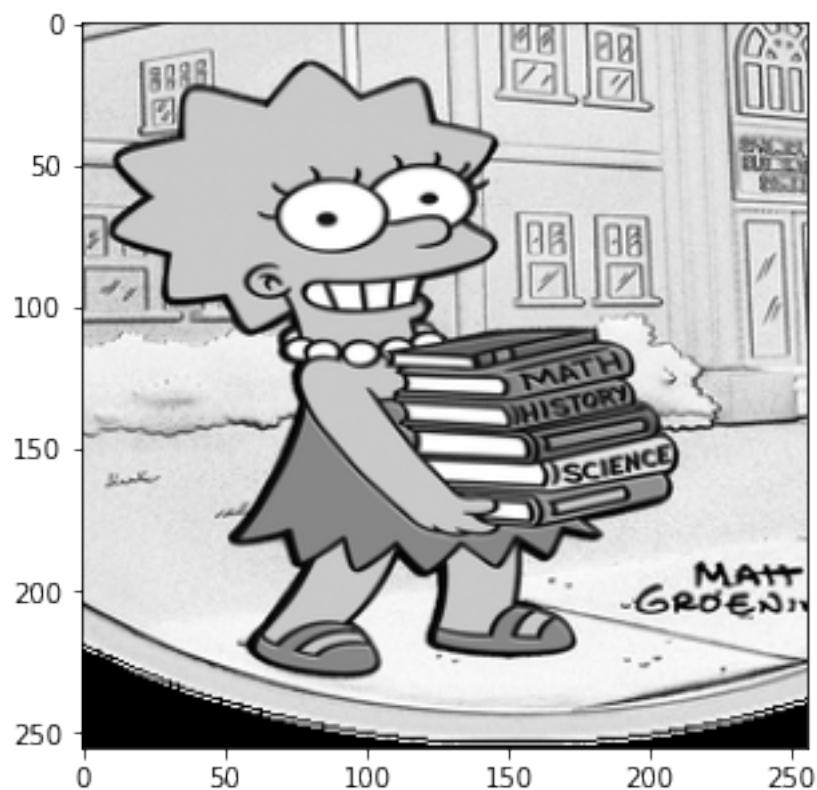
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197950D0>



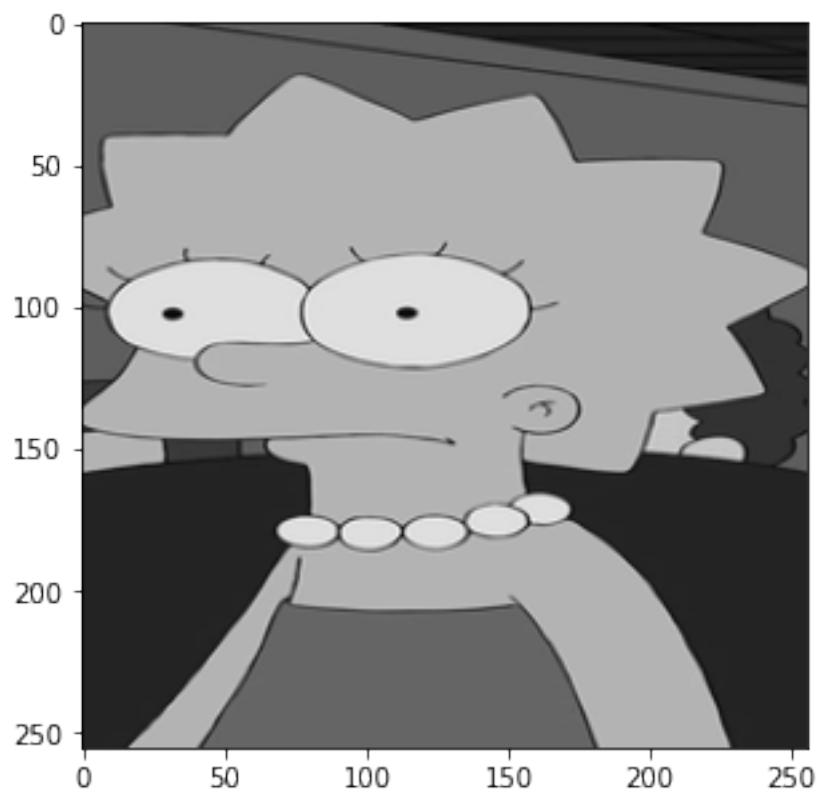
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795110>



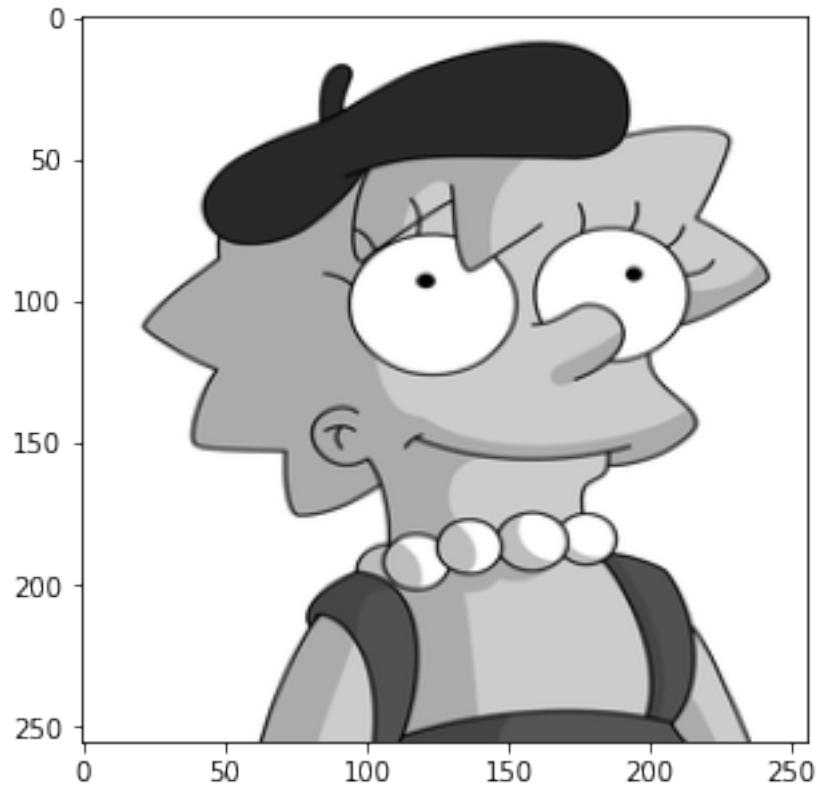
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795150>



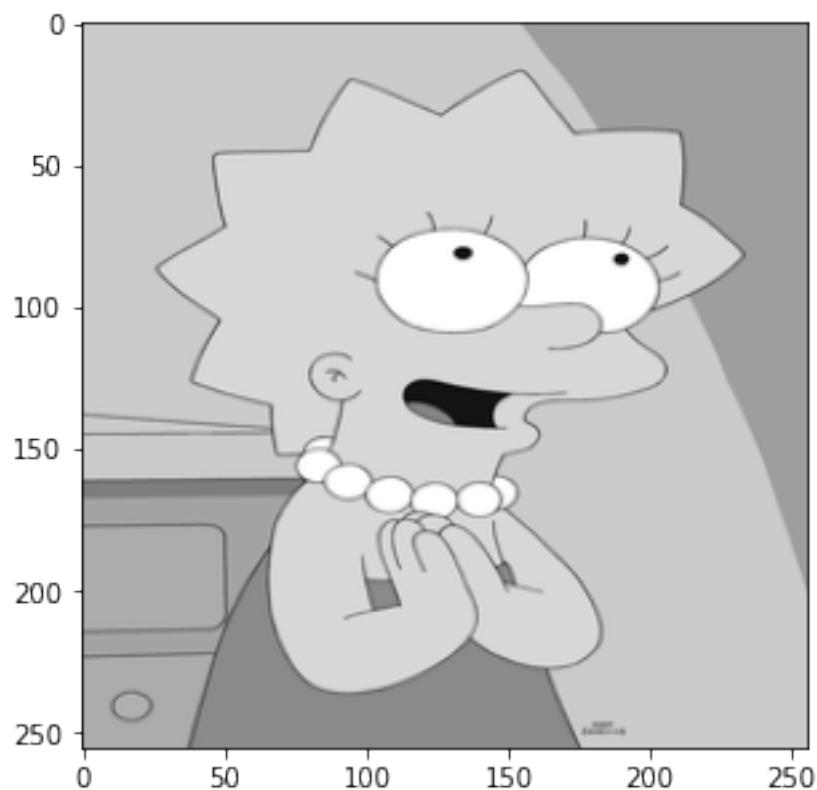
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795190>



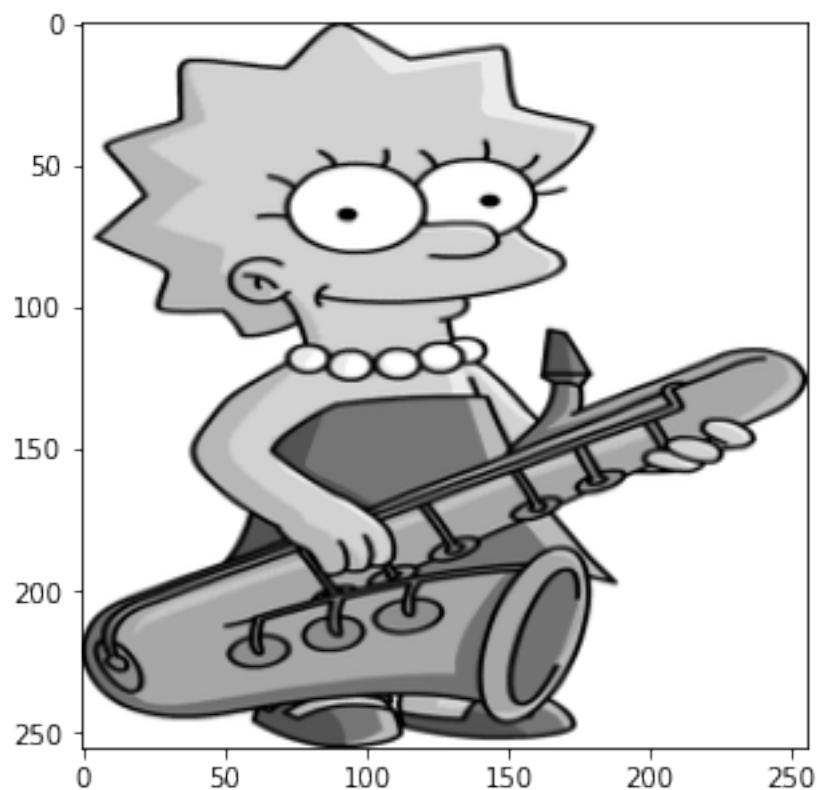
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197951D0>



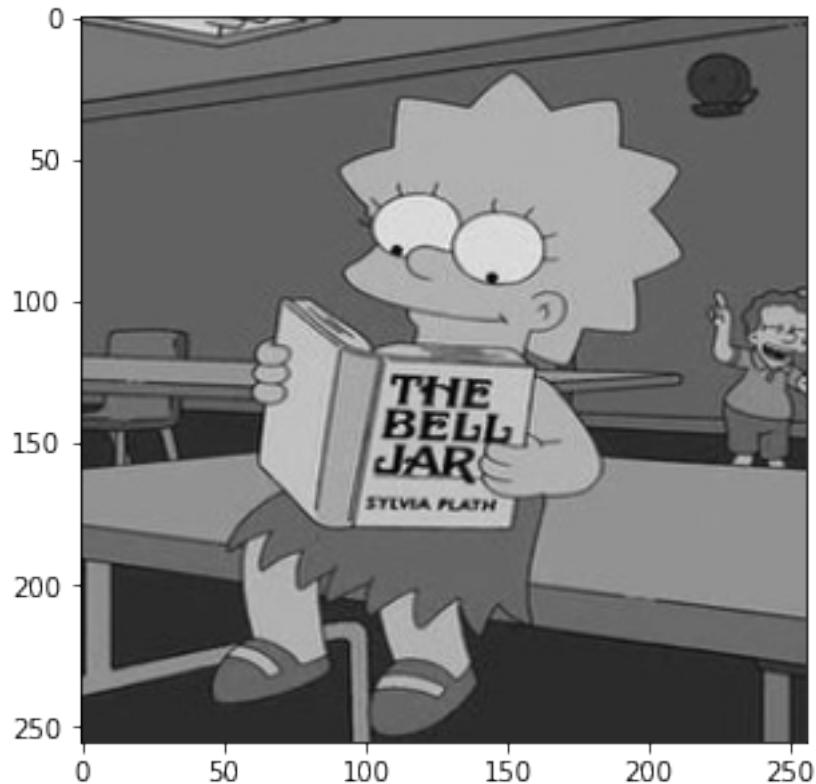
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B87690>



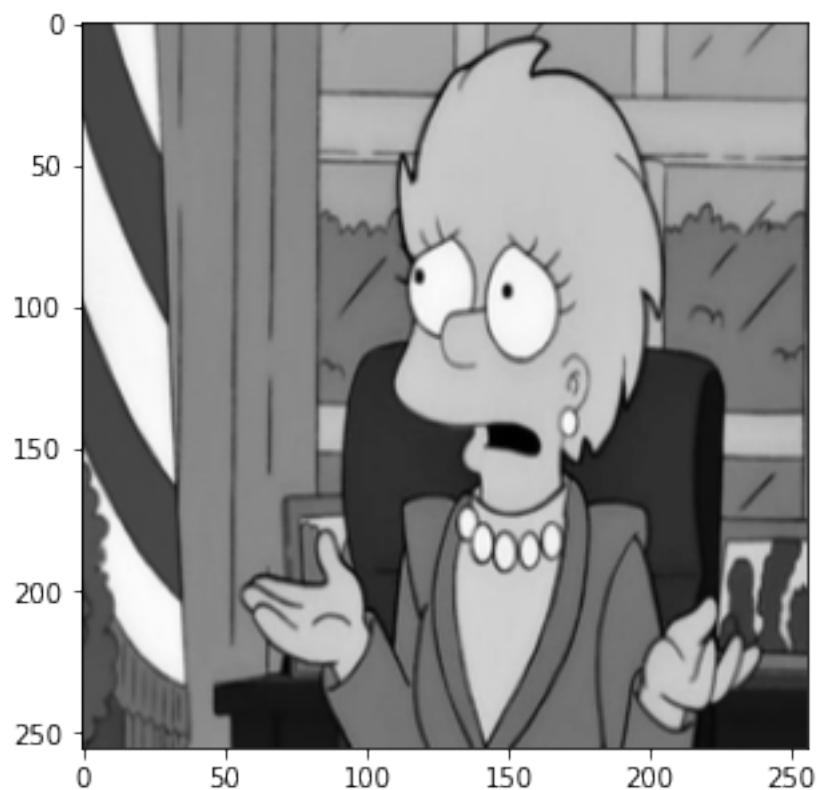
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B87190>



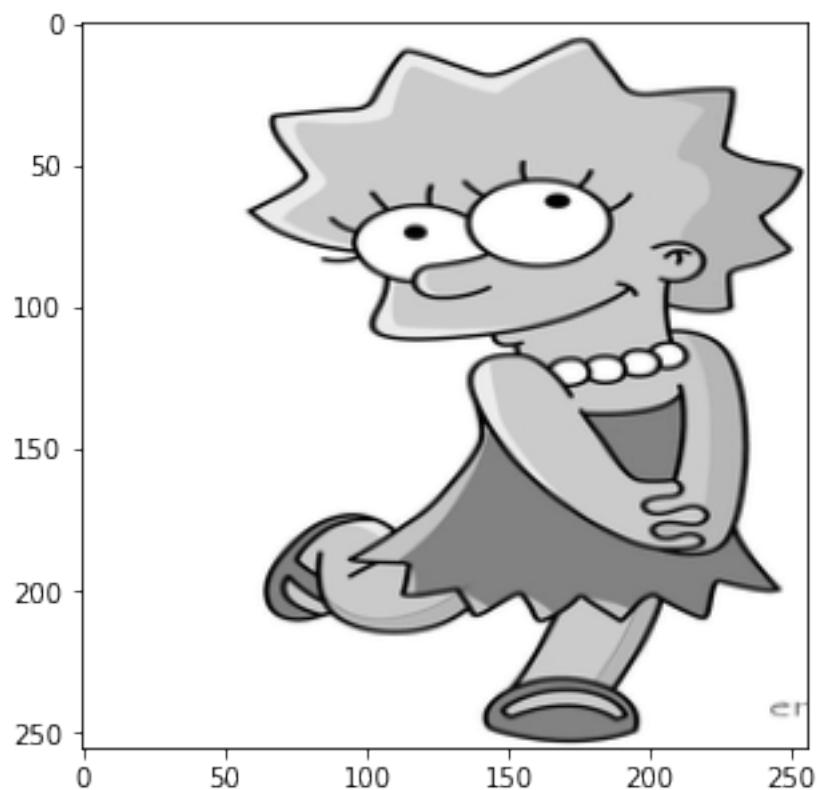
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B87590>



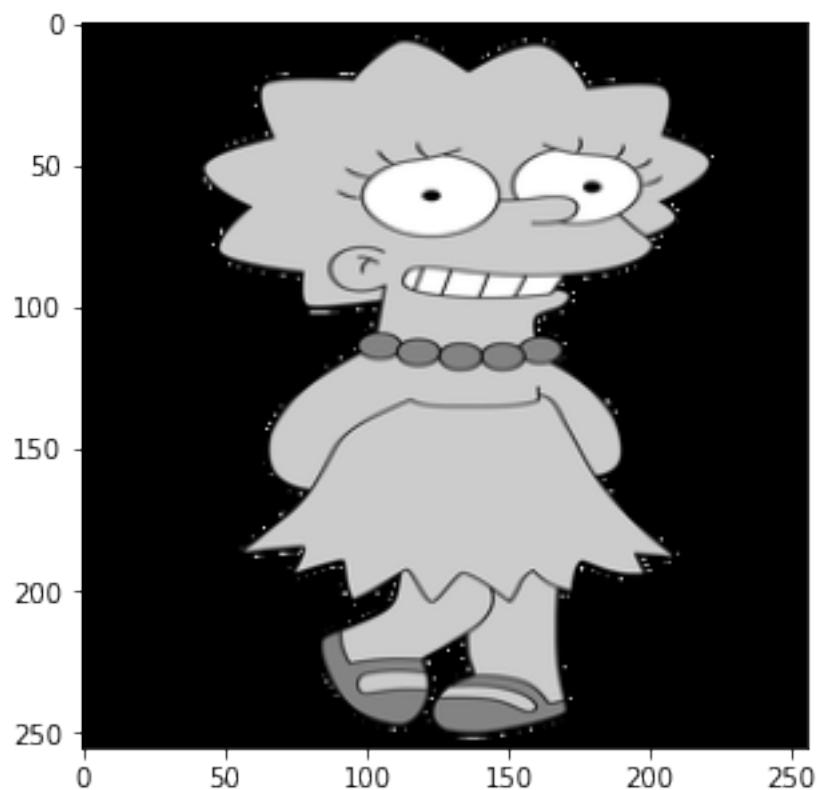
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E750>



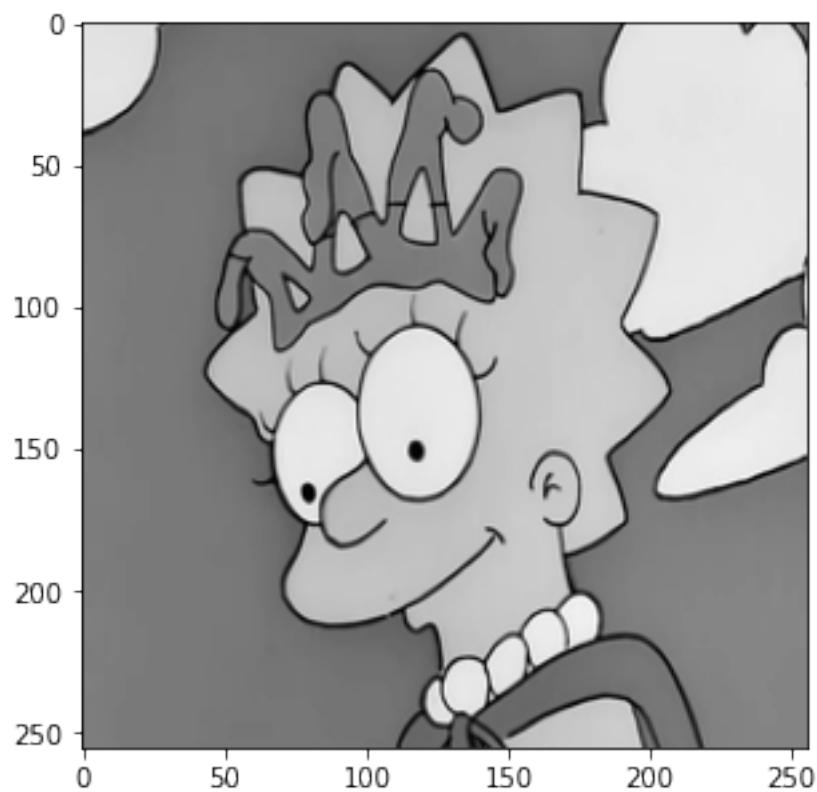
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E850>



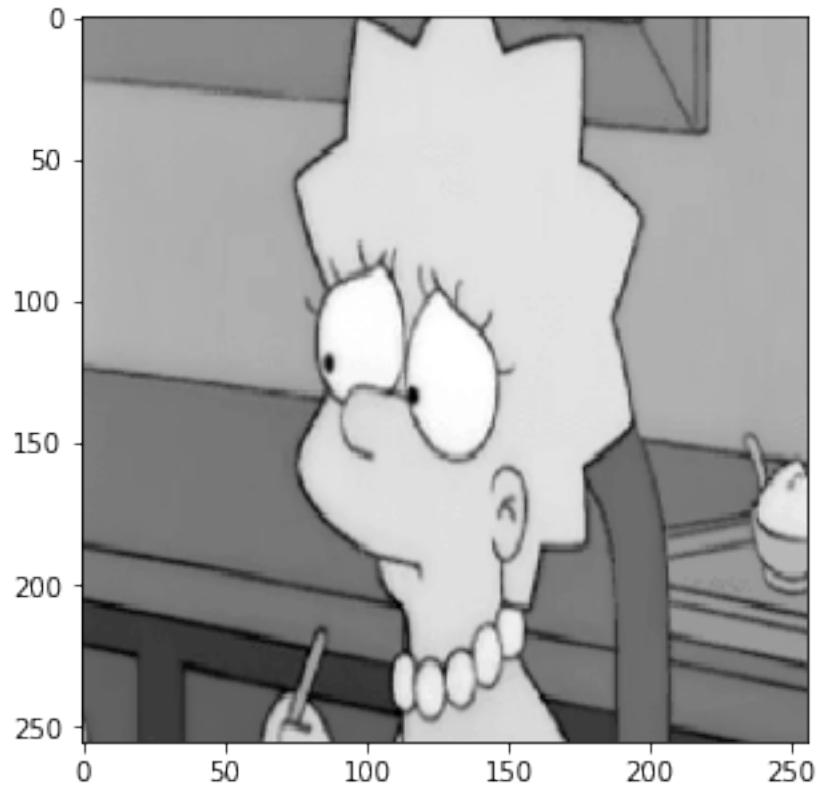
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8ED90>



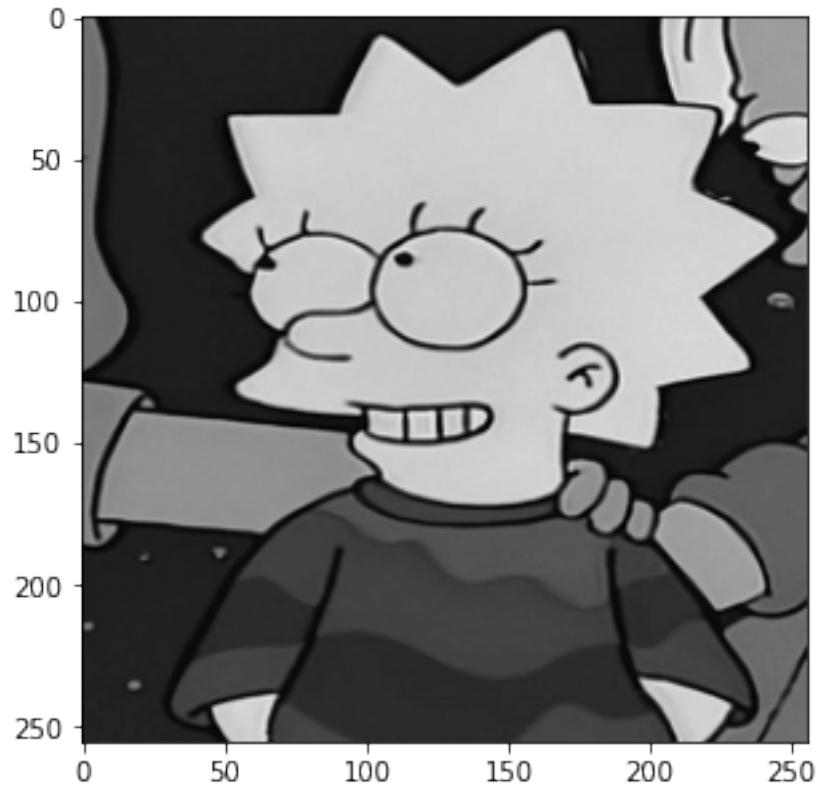
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8EB90>



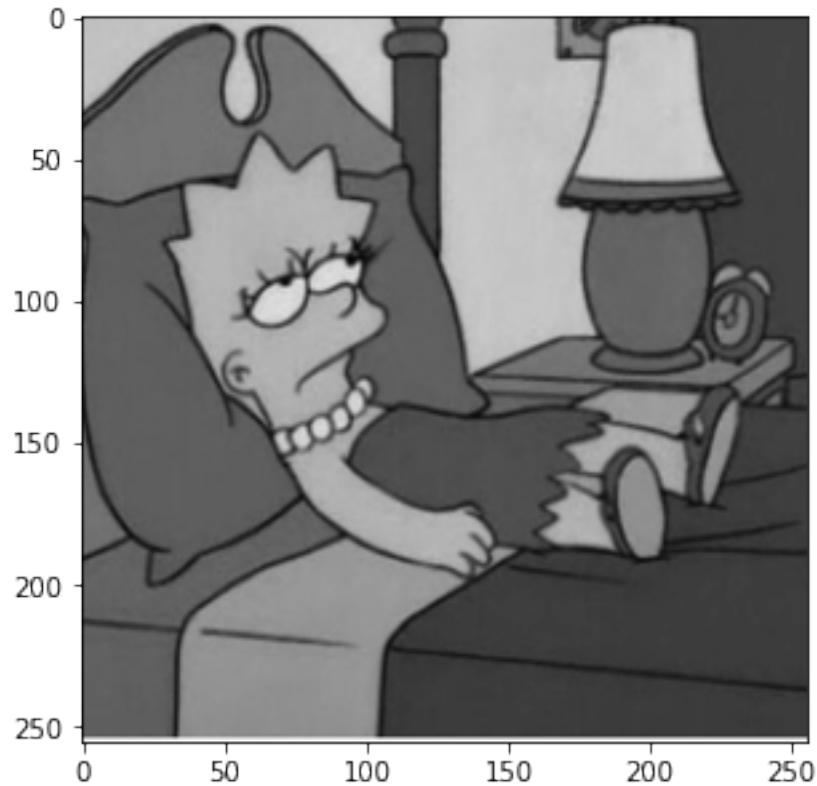
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E3D0>



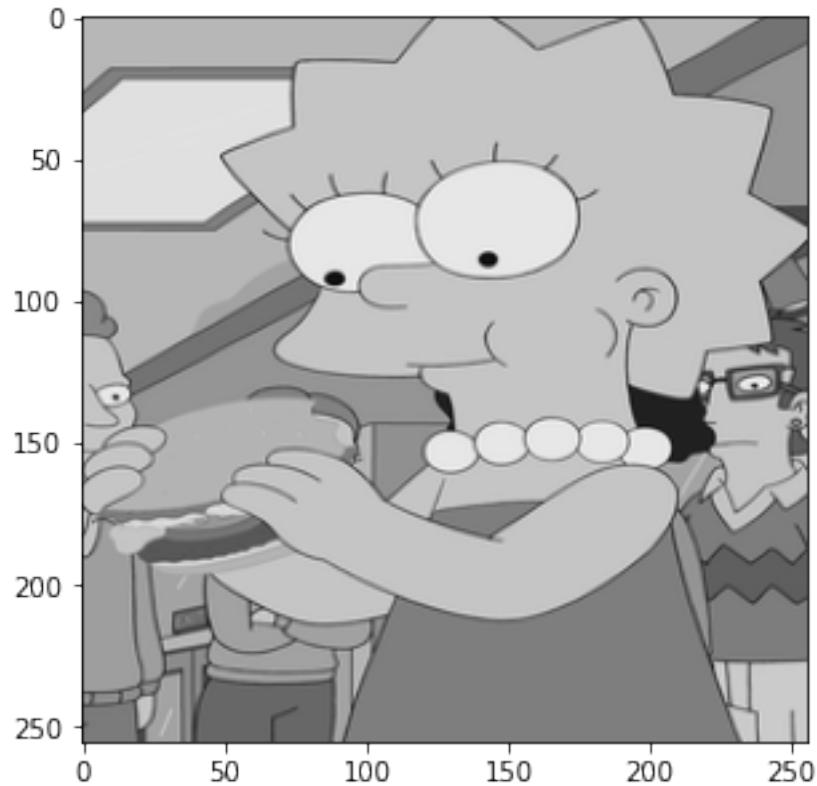
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E190>



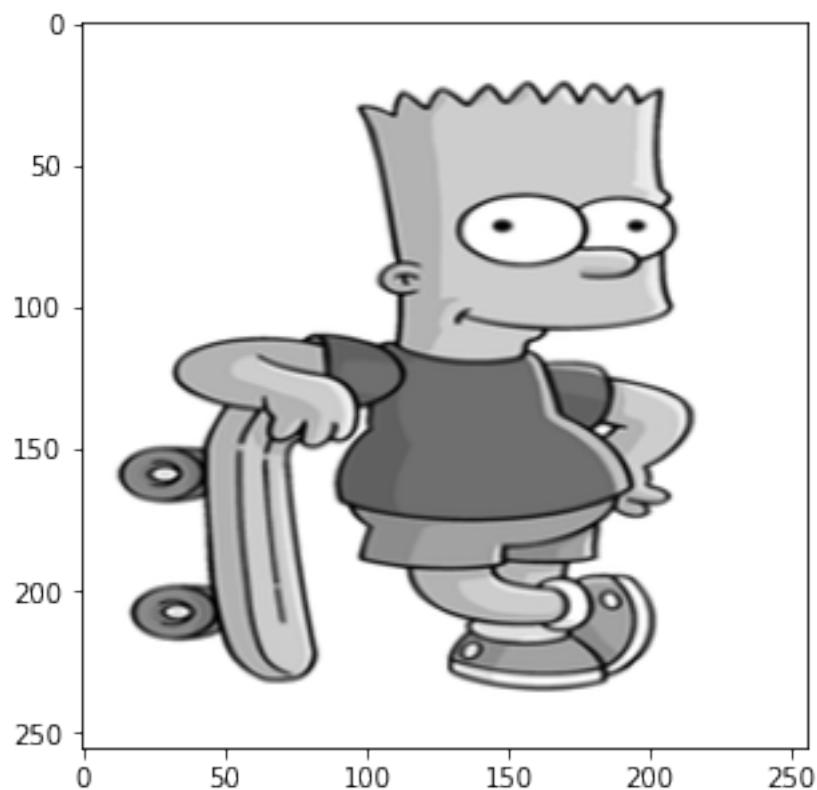
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8EC10>



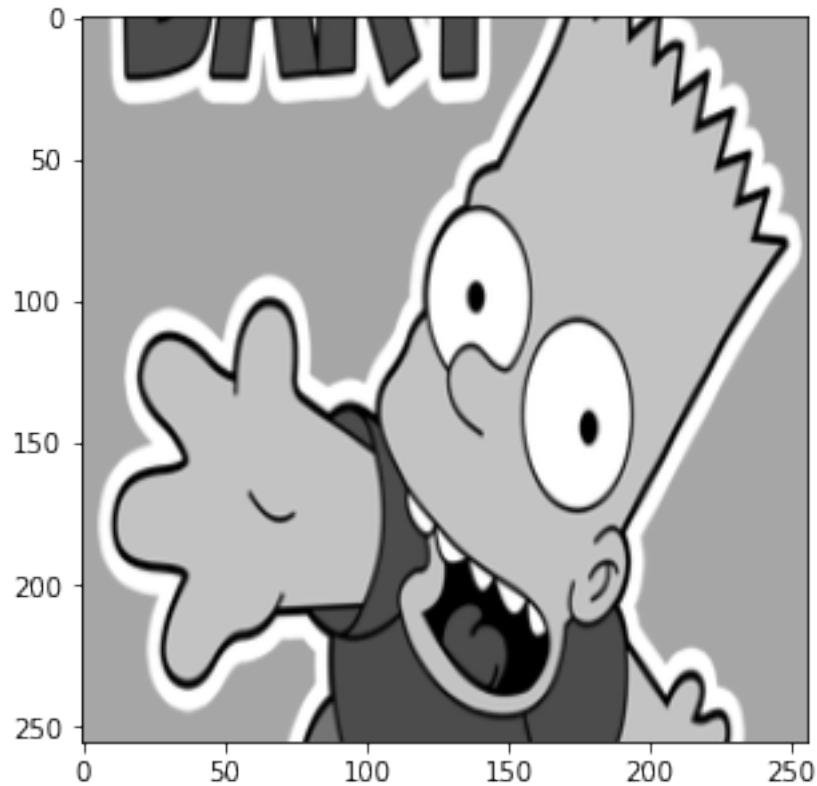
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E290>



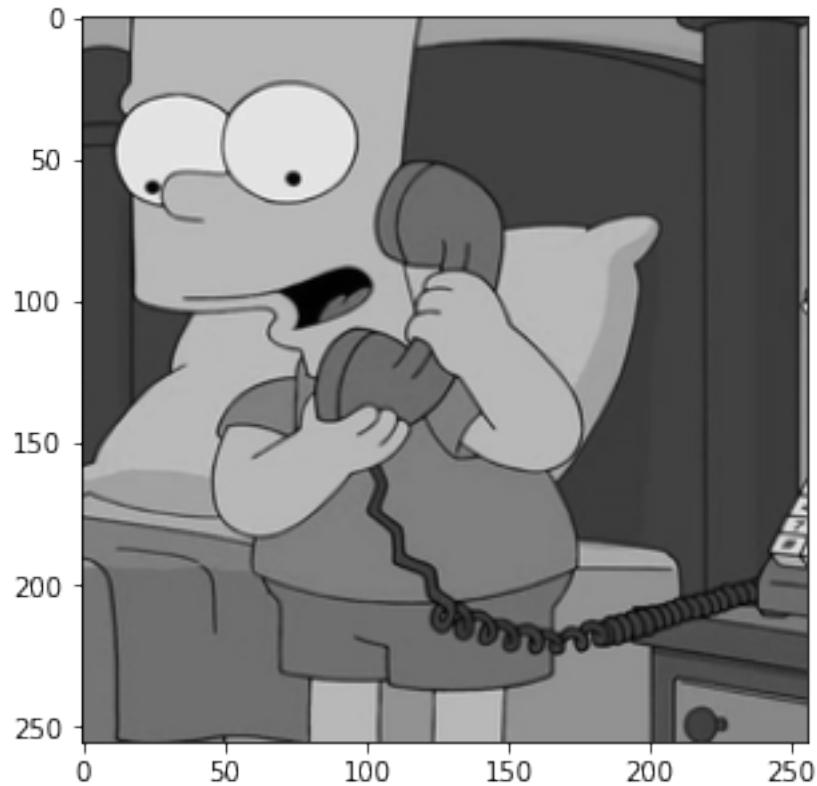
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BB50>



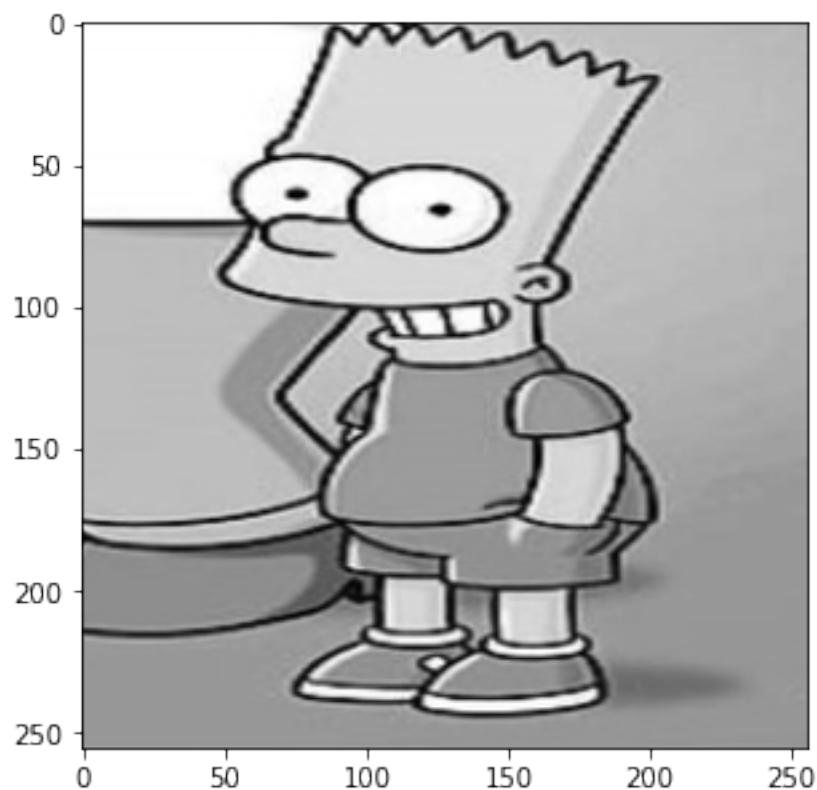
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BB90>



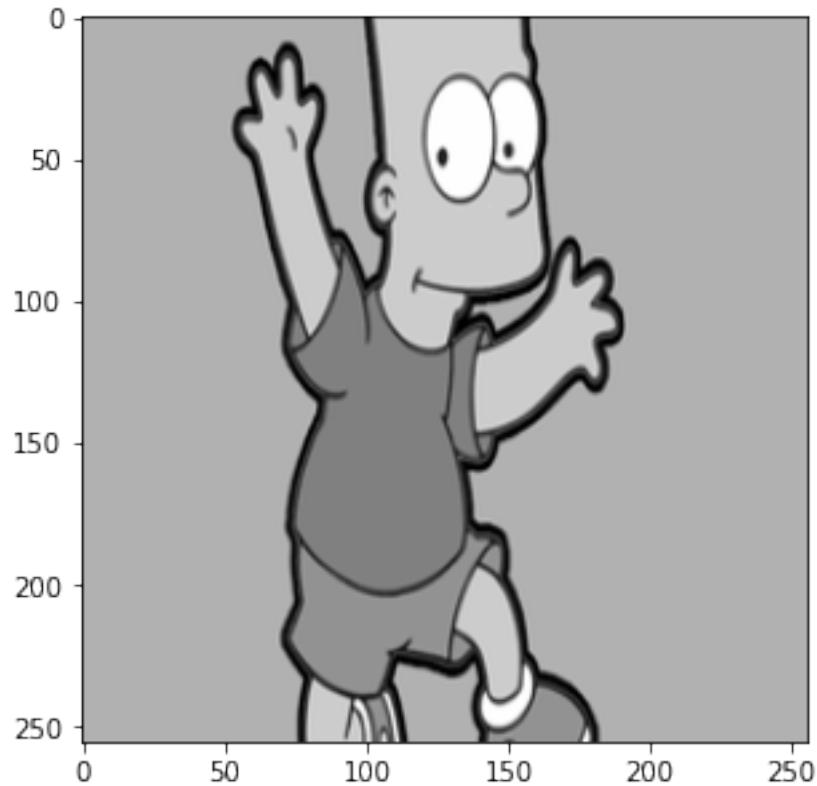
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BBD0>



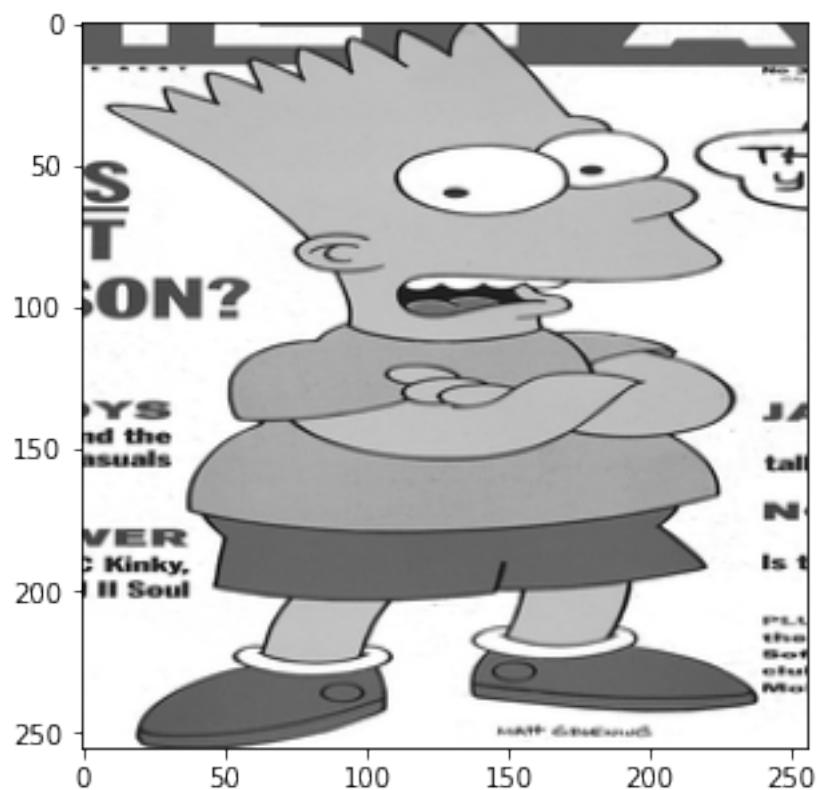
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BC10>



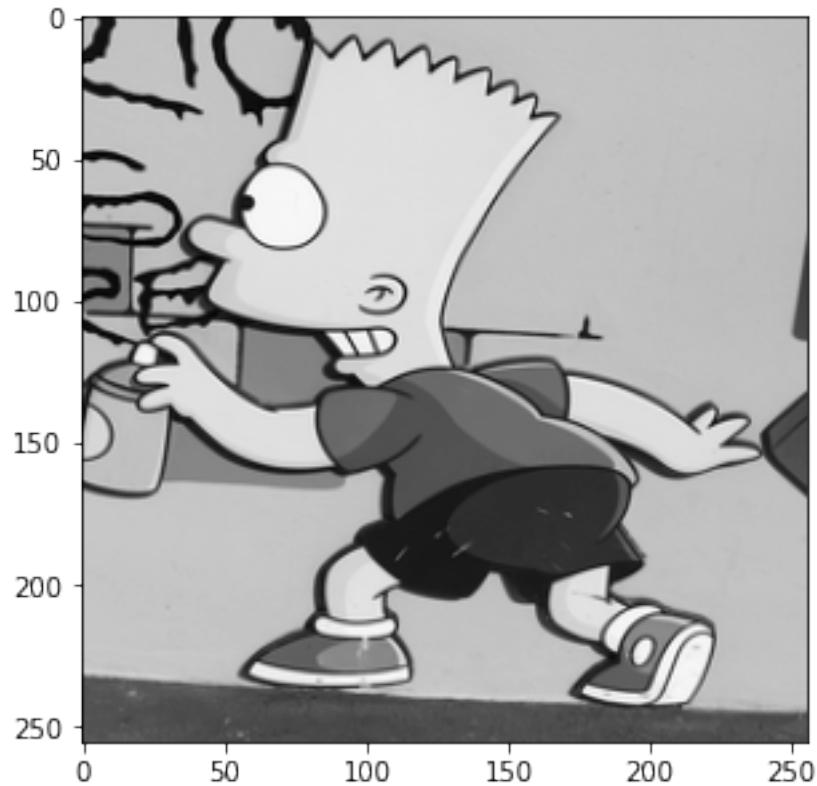
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BC50>



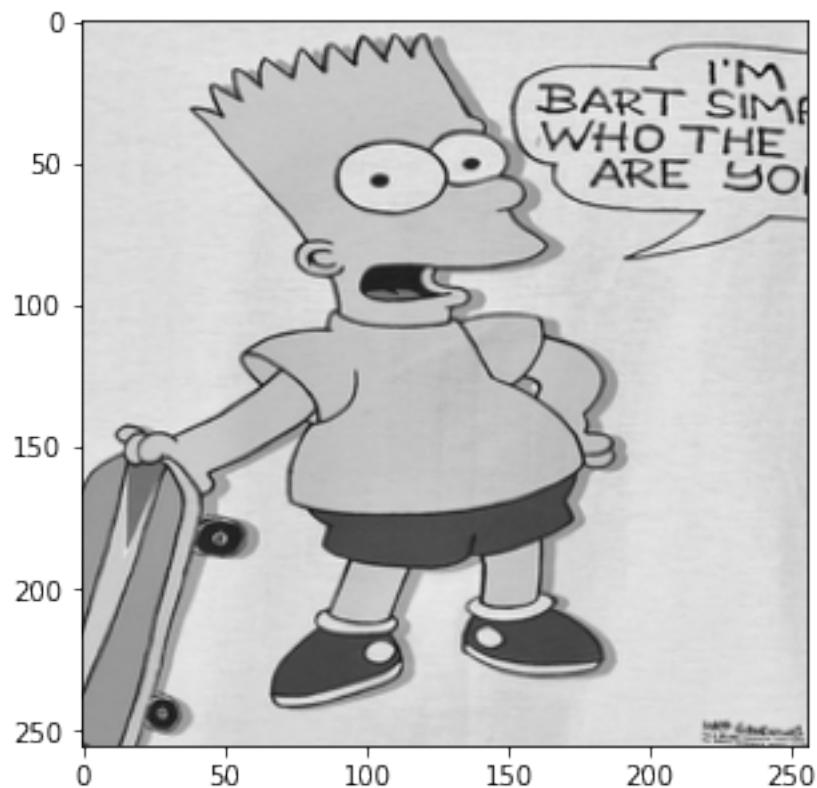
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BCD0>
```



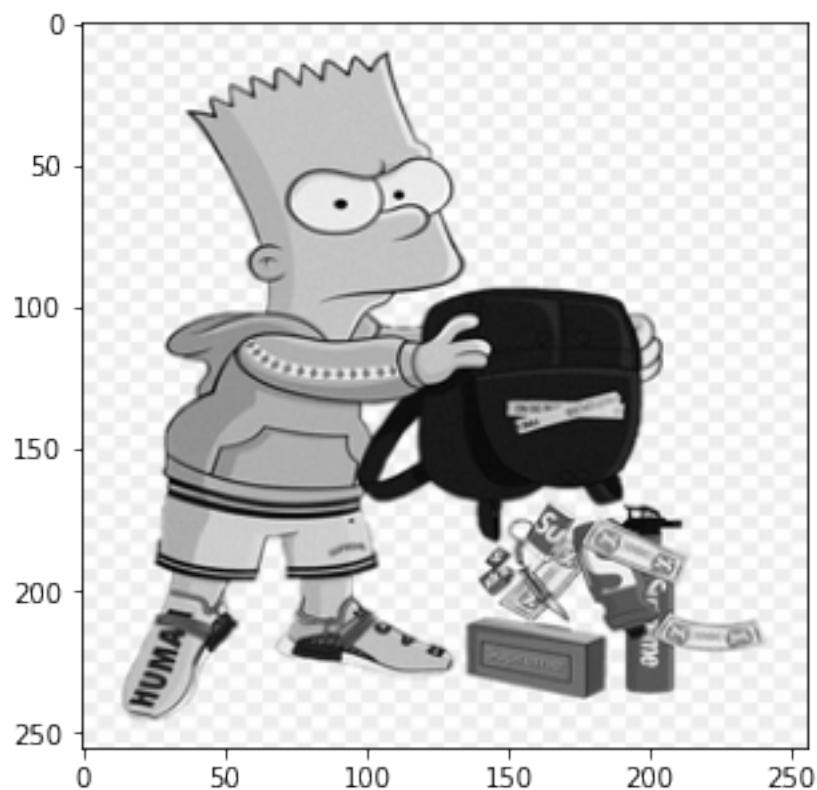
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BD10>



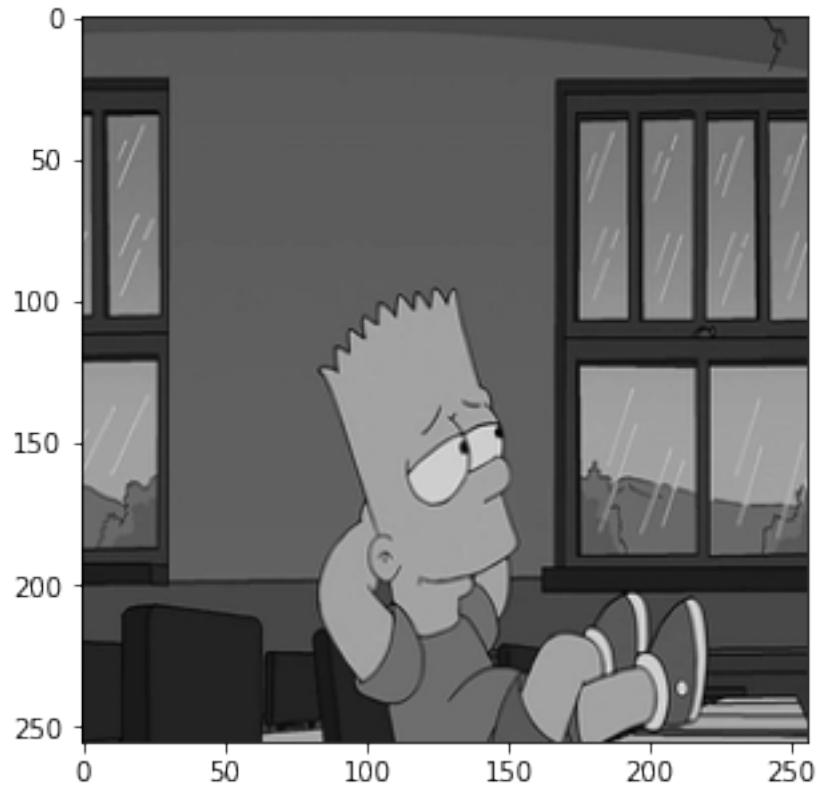
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BD50>



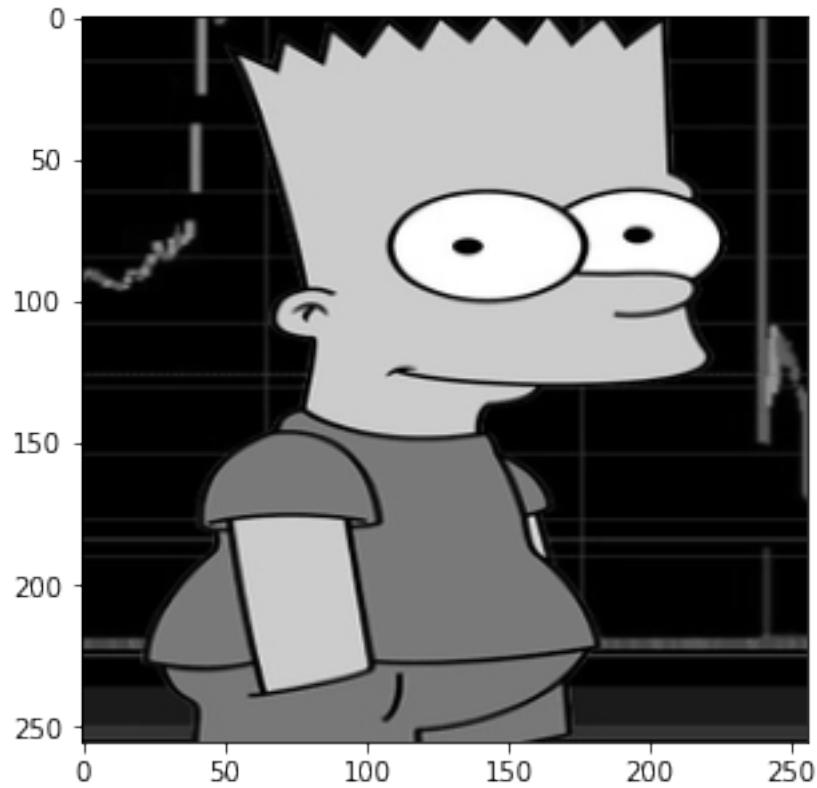
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BD90>



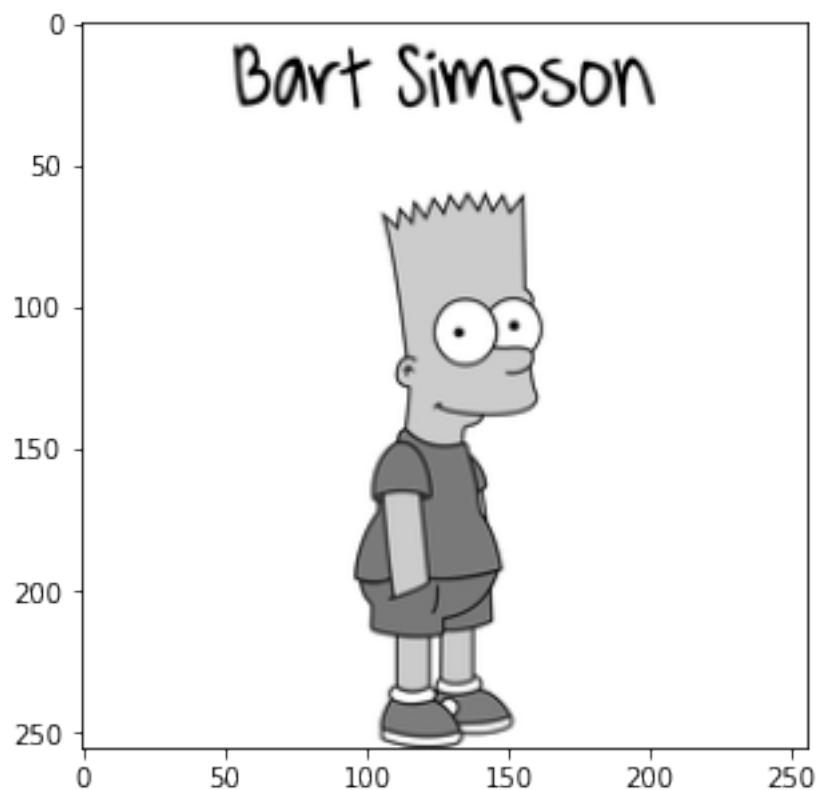
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BC90>



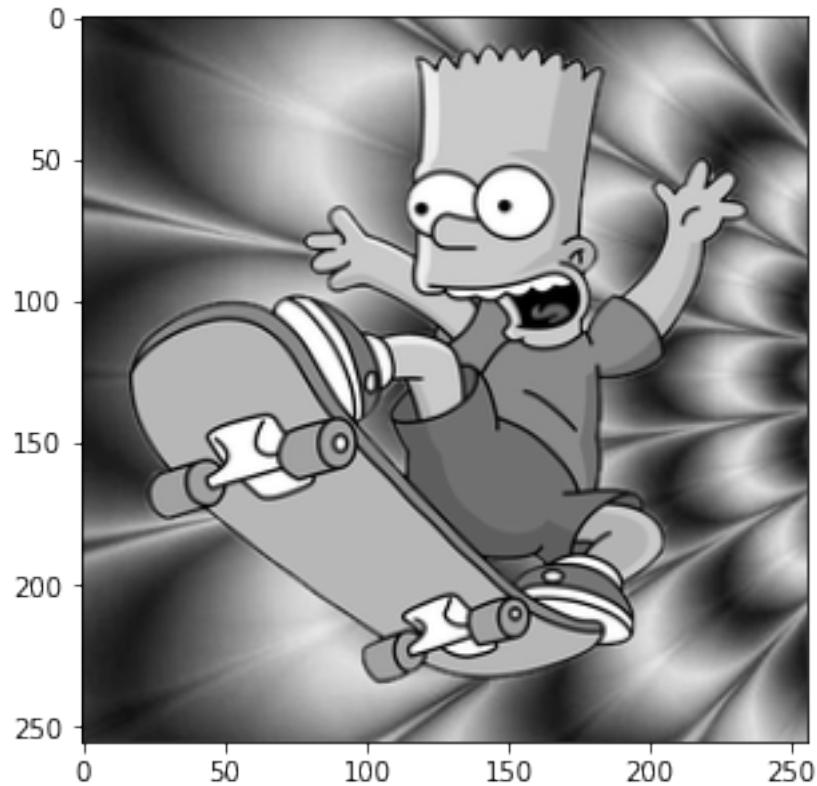
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BDD0>



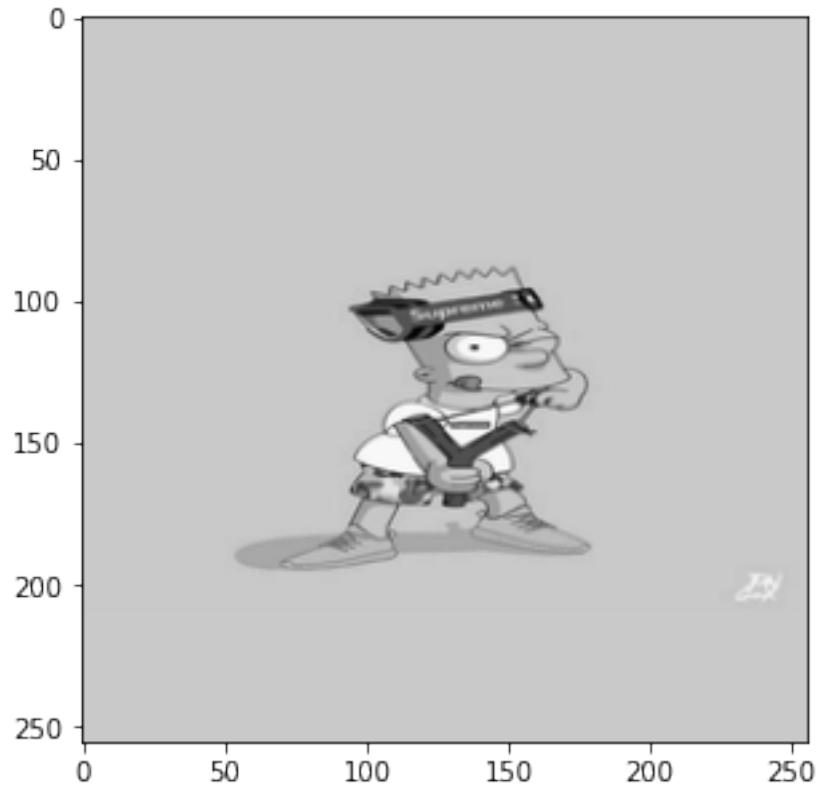
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BE10>



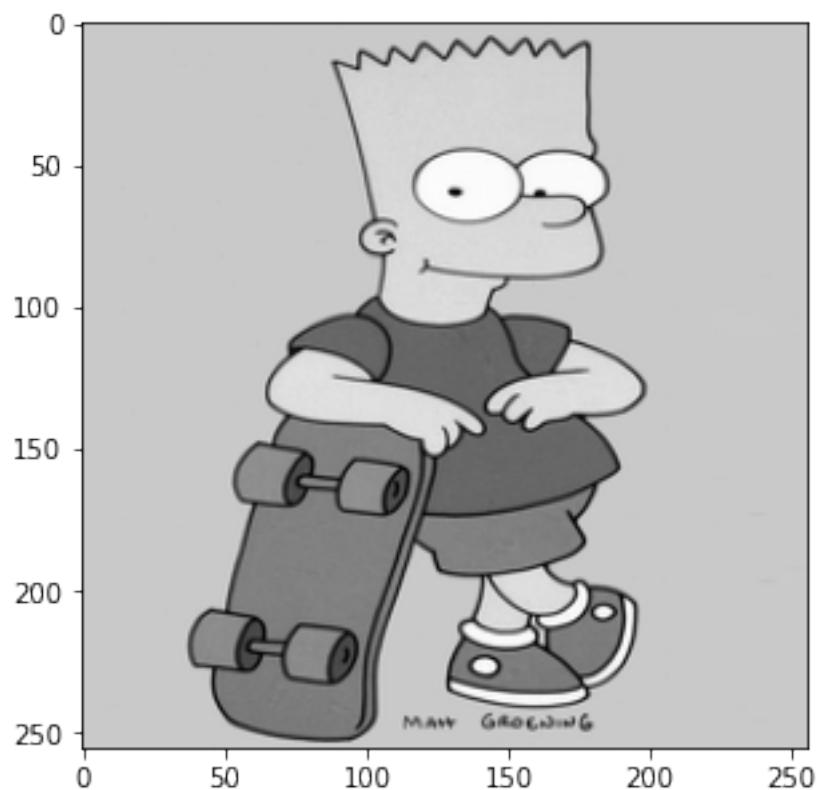
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BE50>
```



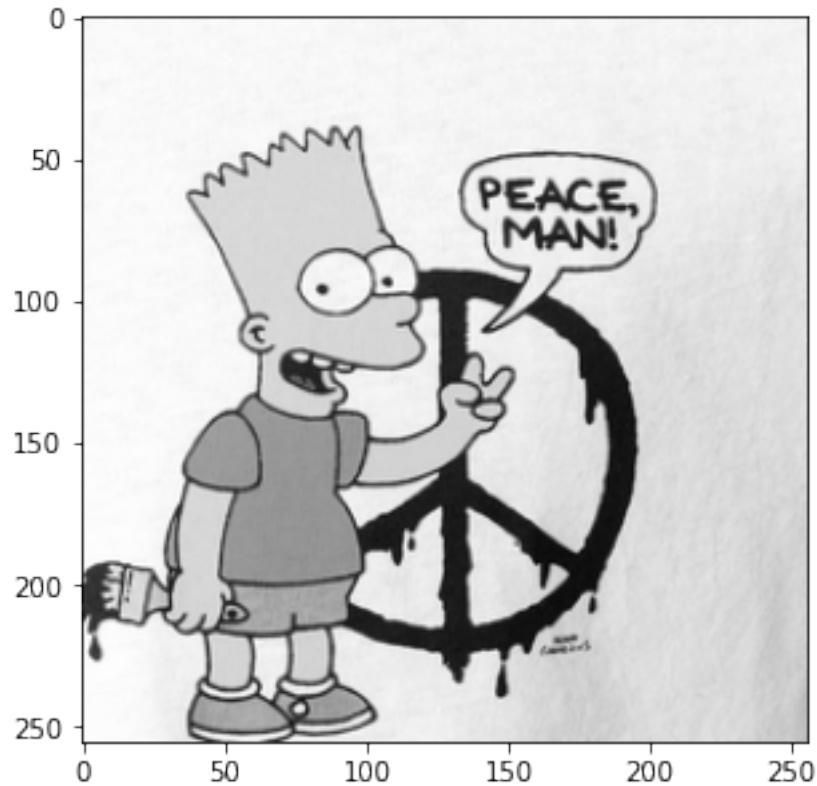
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BE90>
```



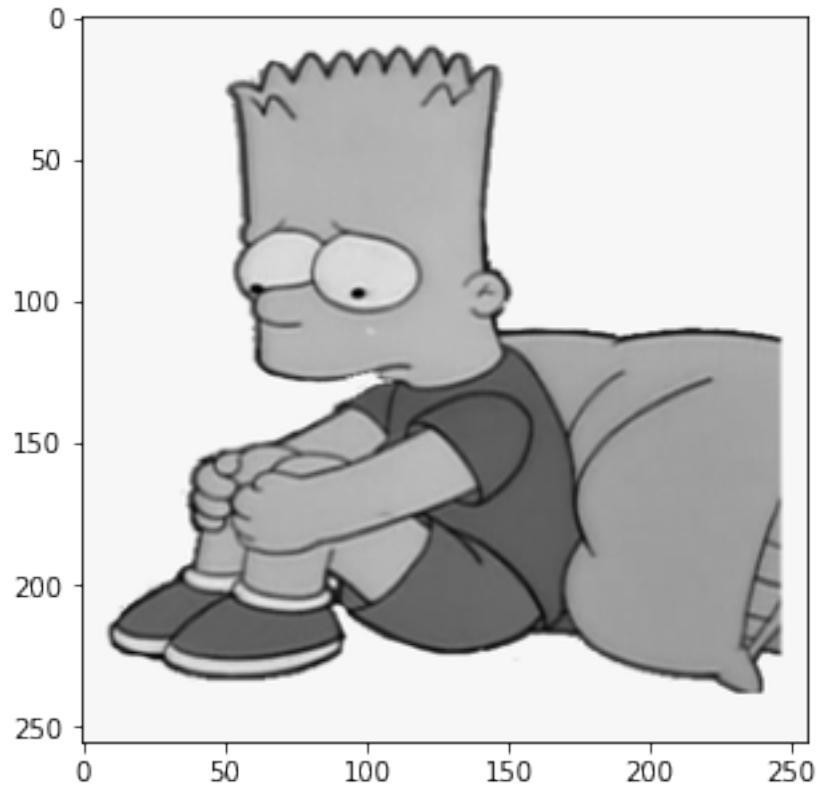
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BED0>
```



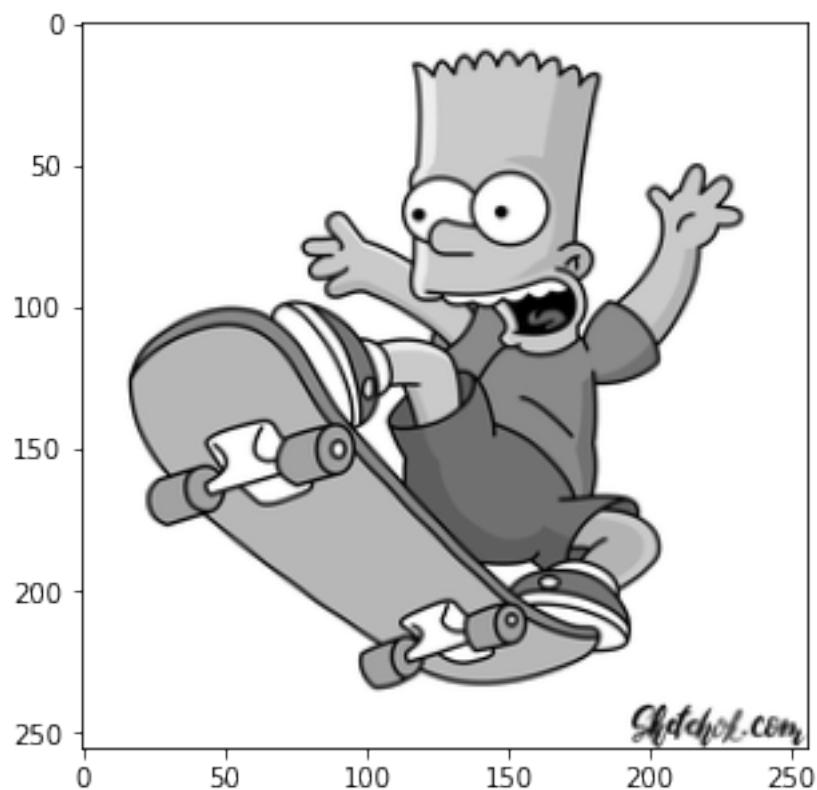
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BF10>



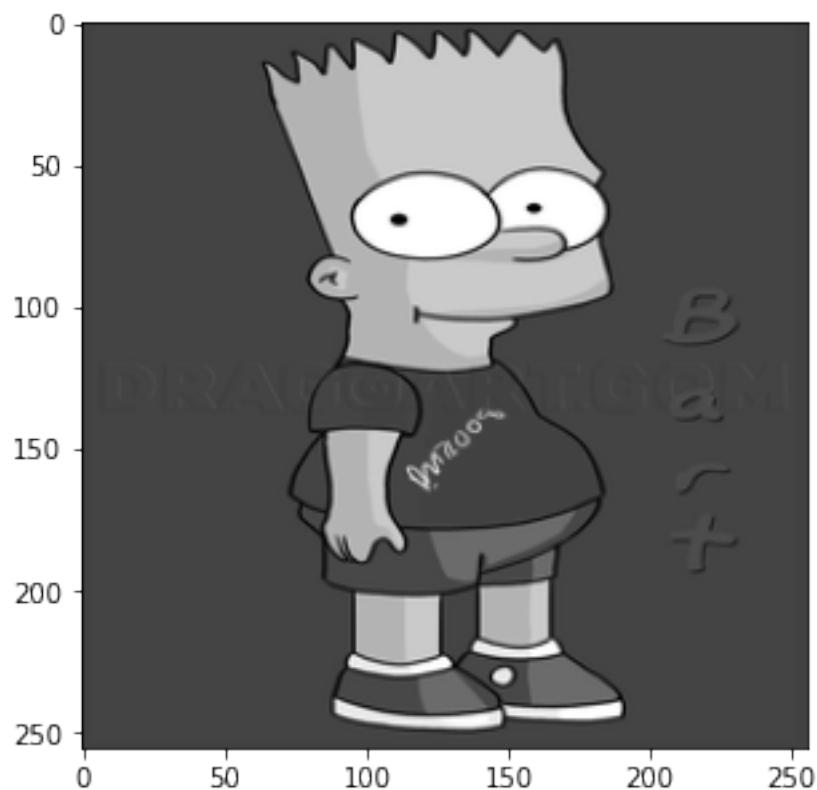
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BF50>



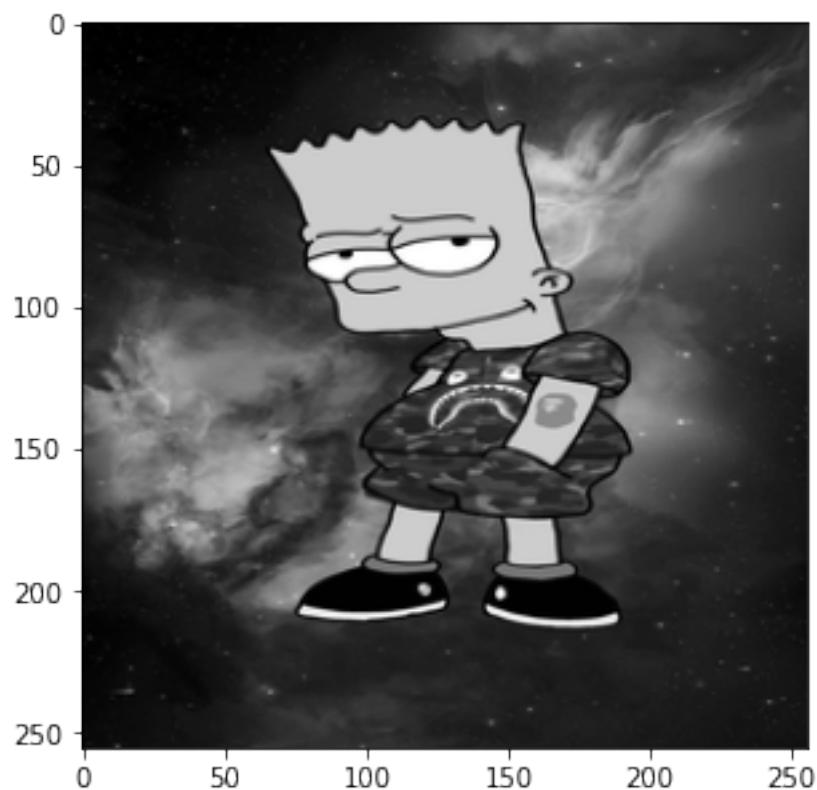
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BF90>



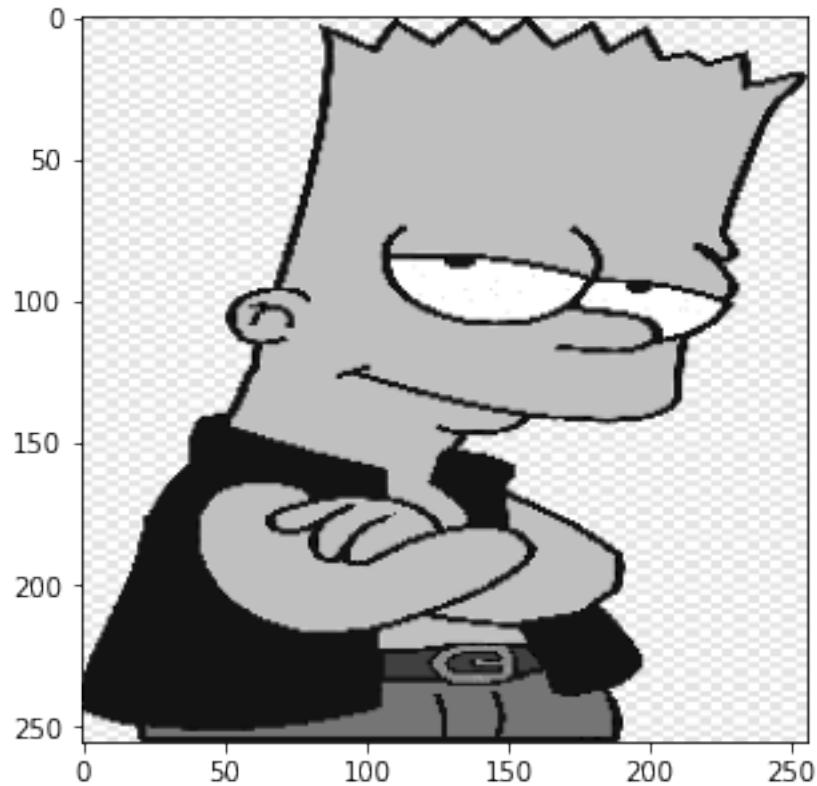
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11978BFD0>



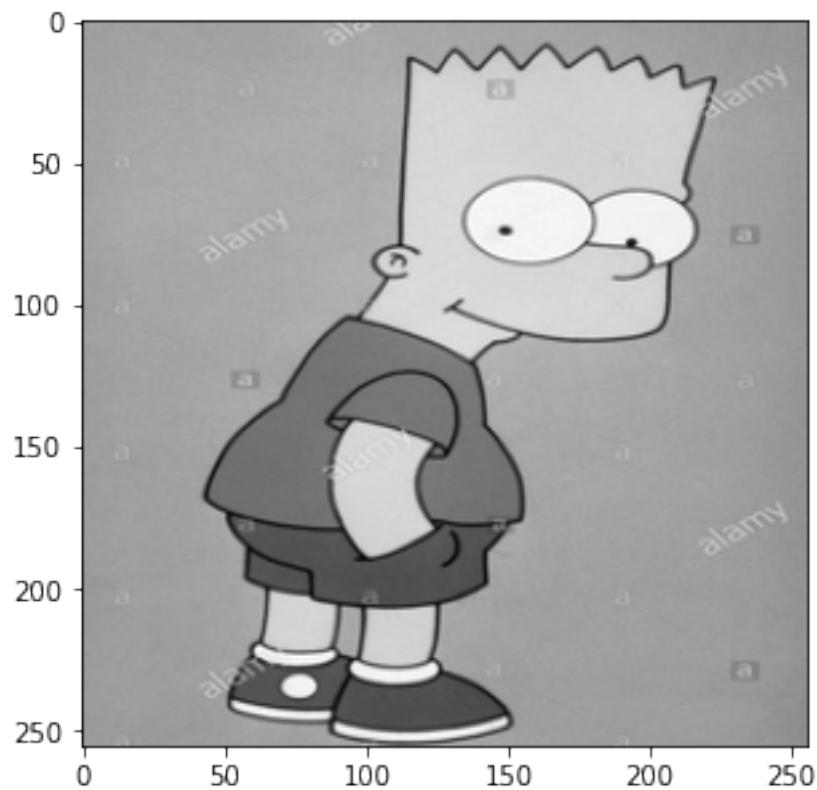
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2A50>



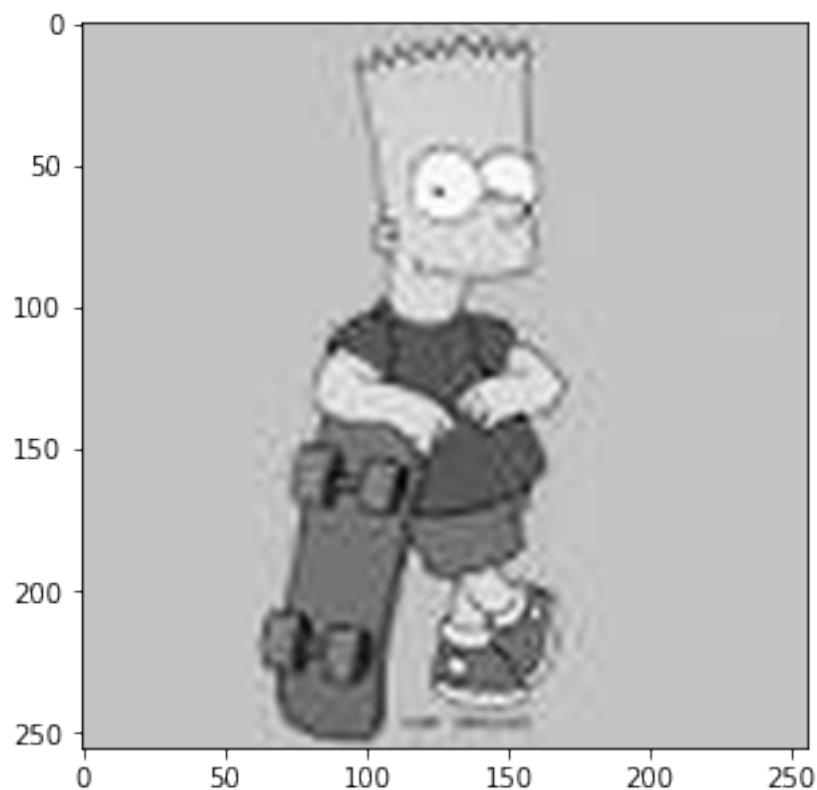
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2C10>



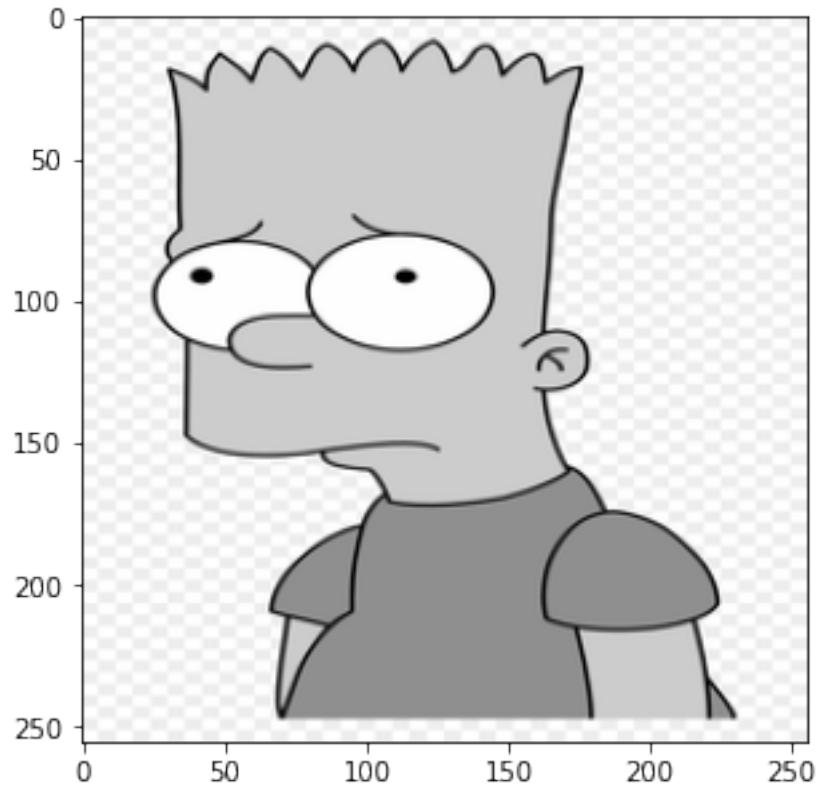
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2BD0>



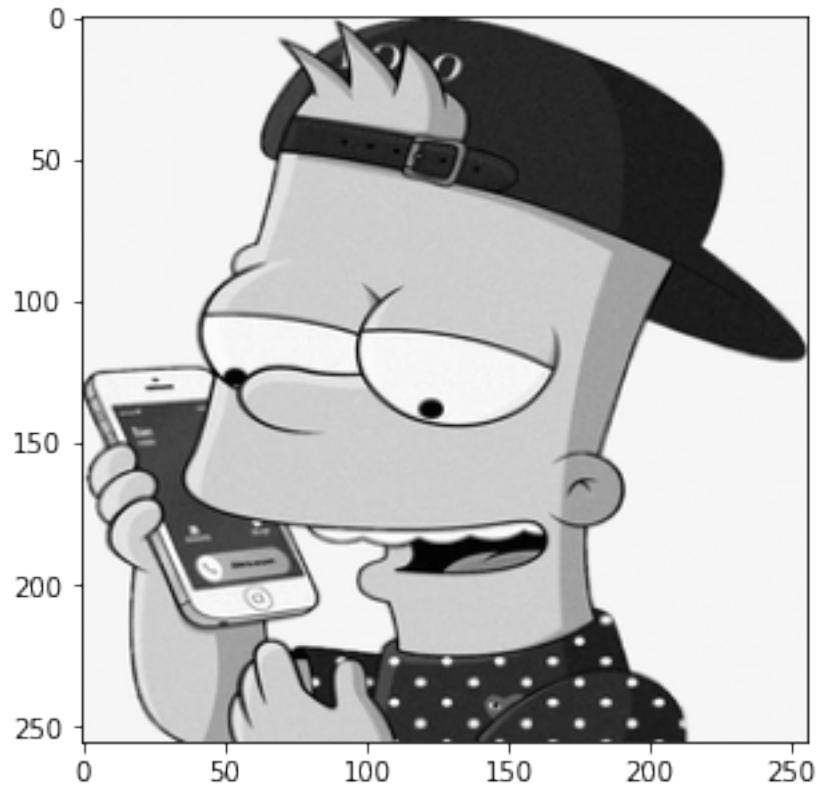
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2990>



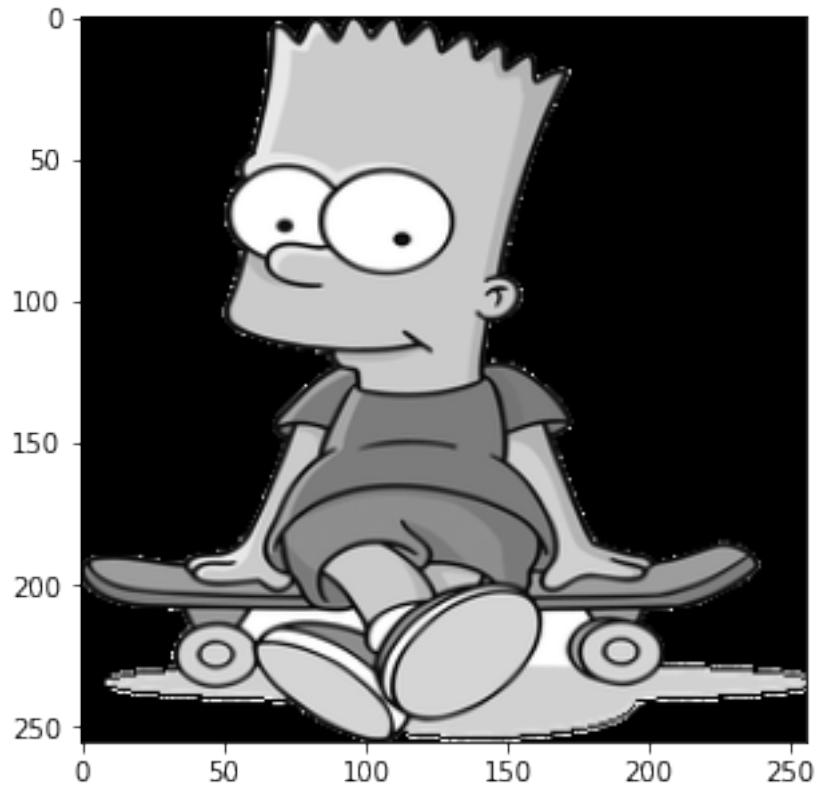
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2A90>
```



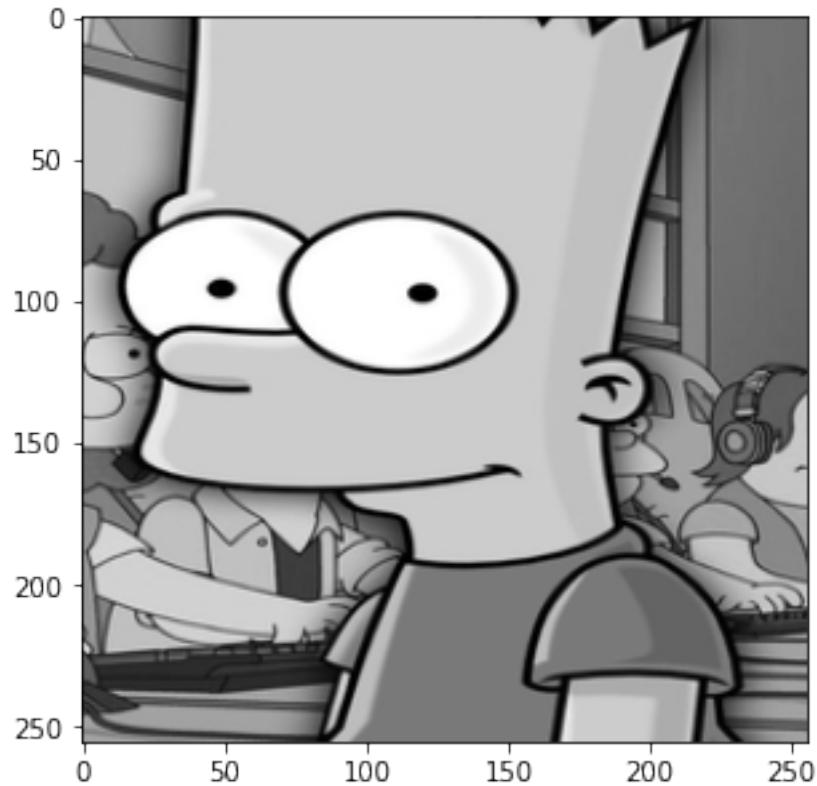
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2850>
```



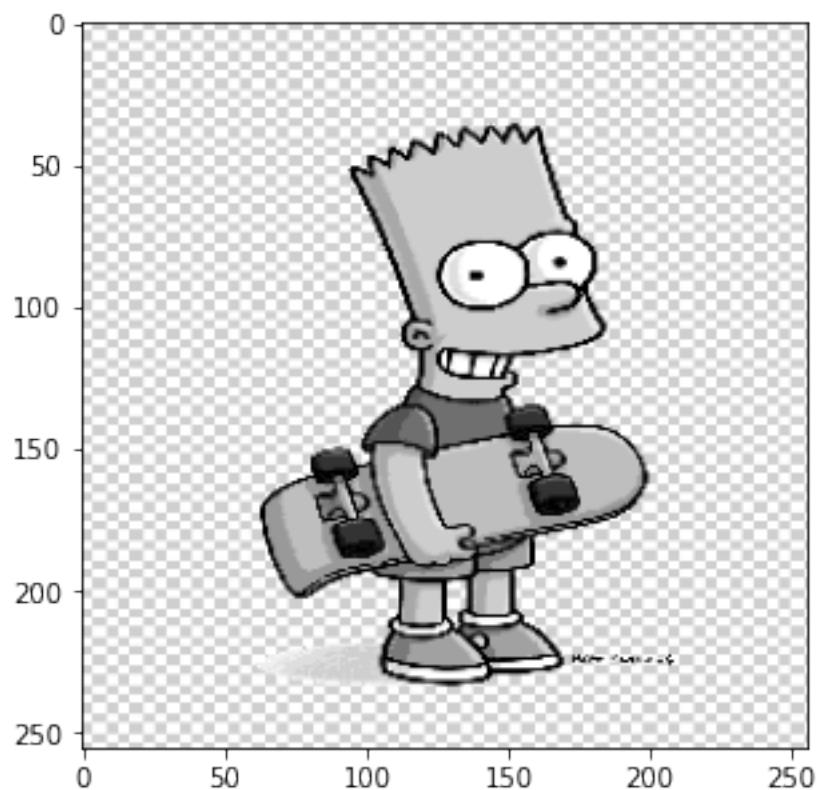
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B28D0>



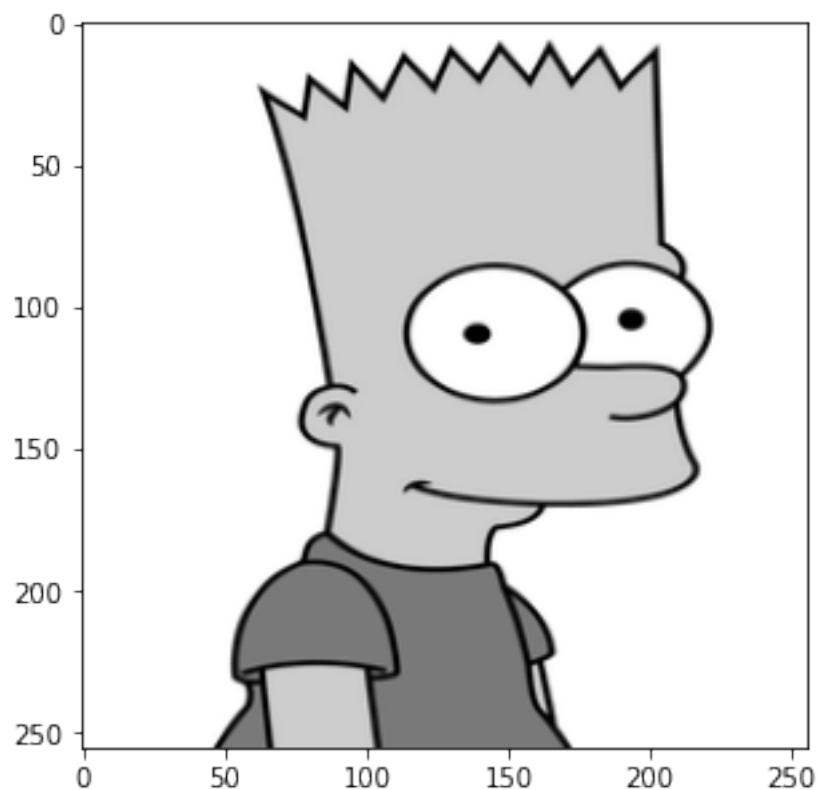
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2DD0>



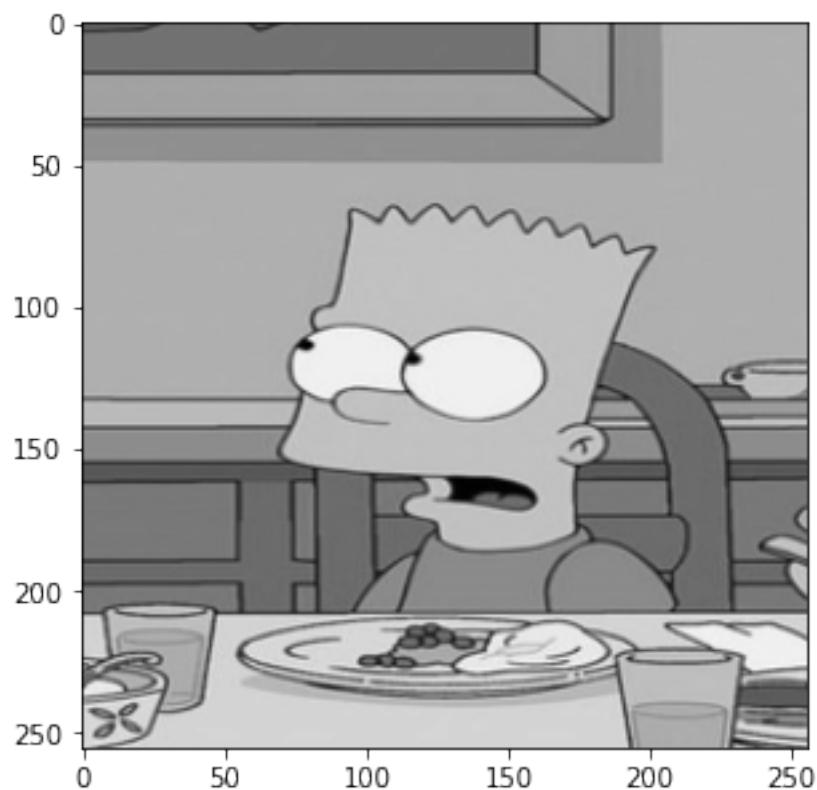
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B21D0>



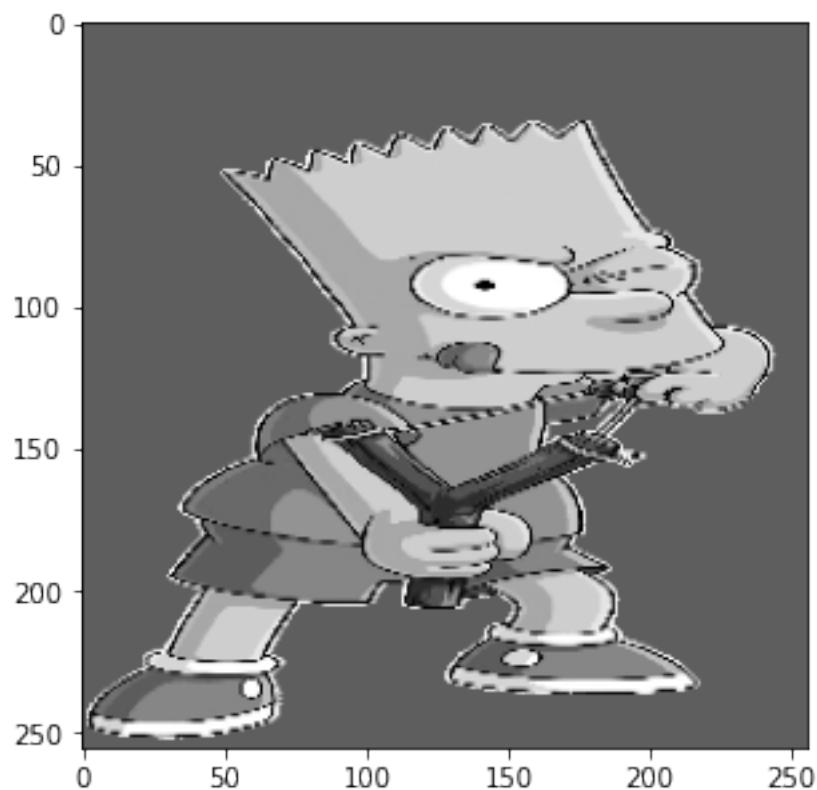
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2150>



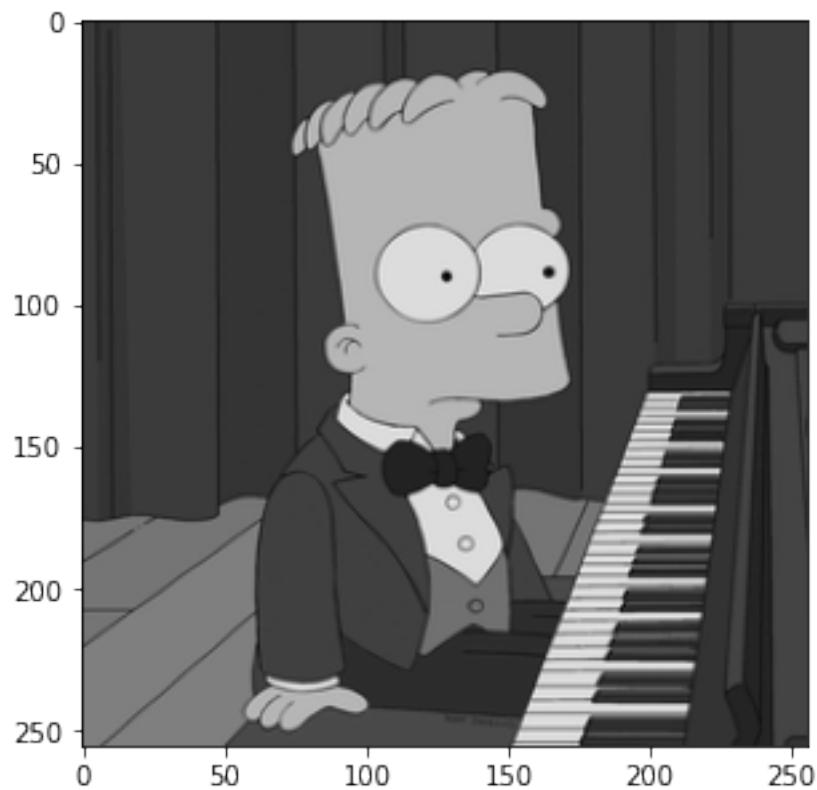
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2050>



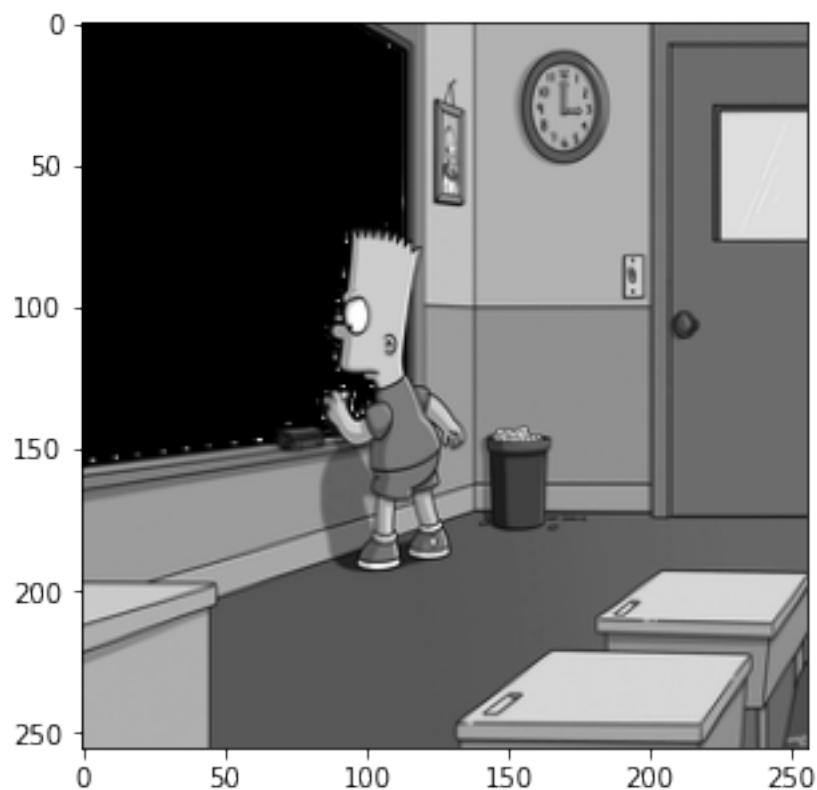
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B20D0>



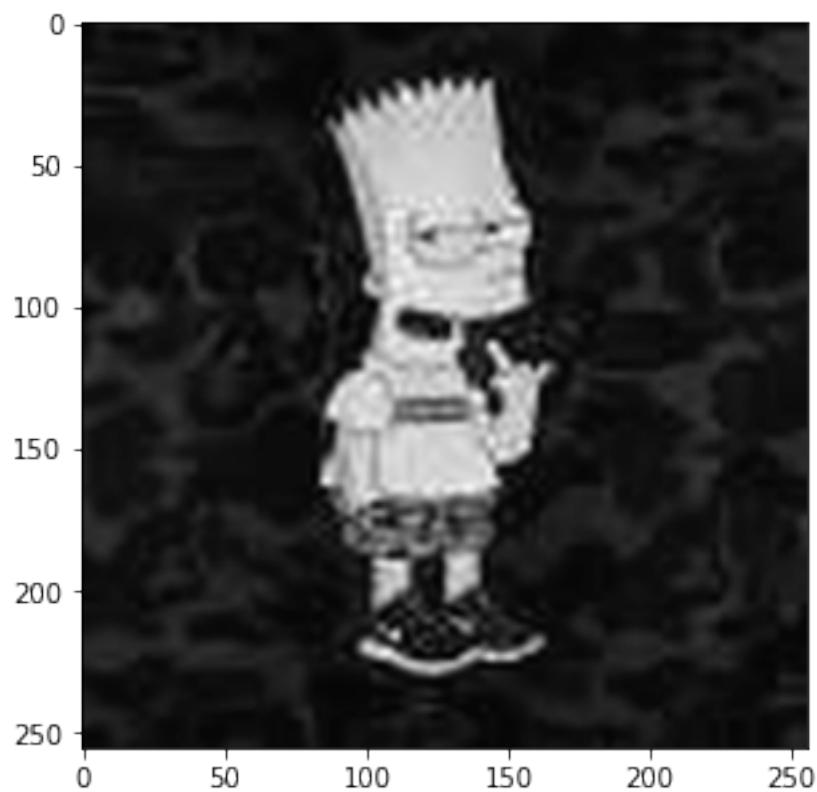
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2090>



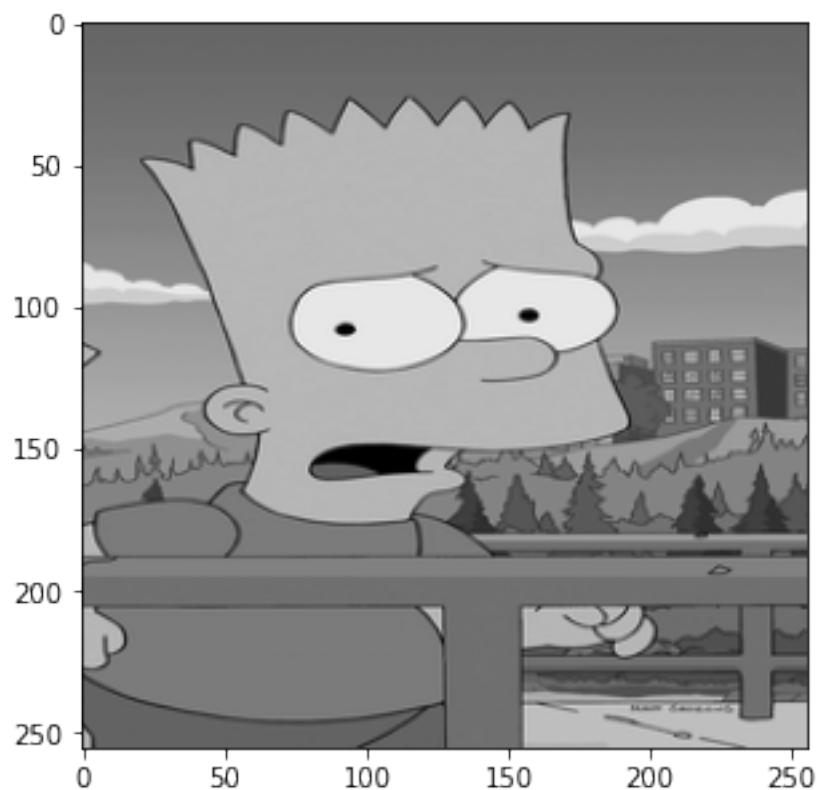
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2690>



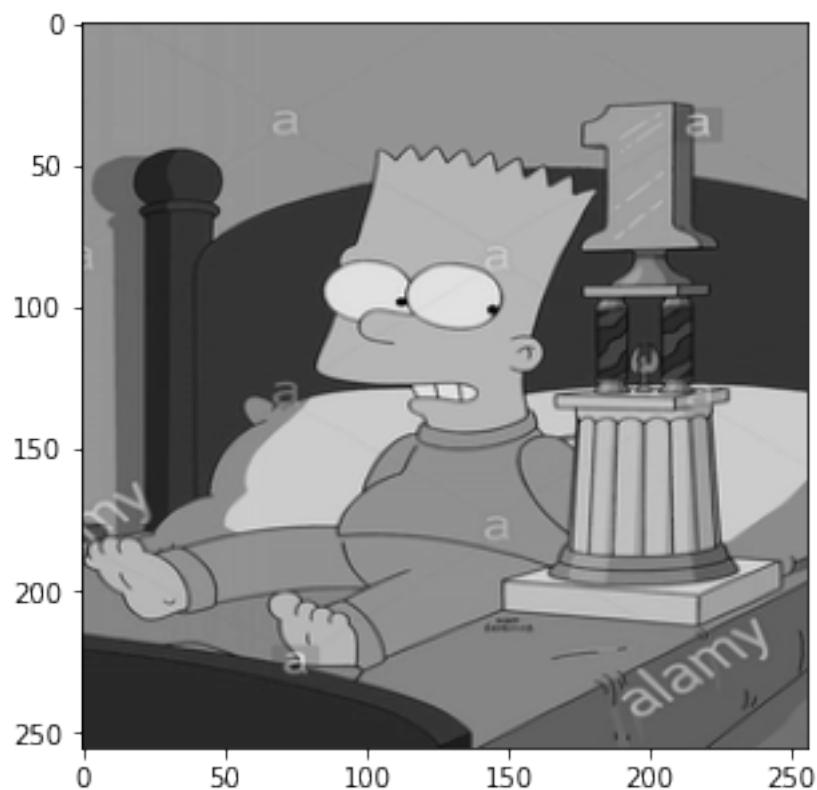
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B26D0>



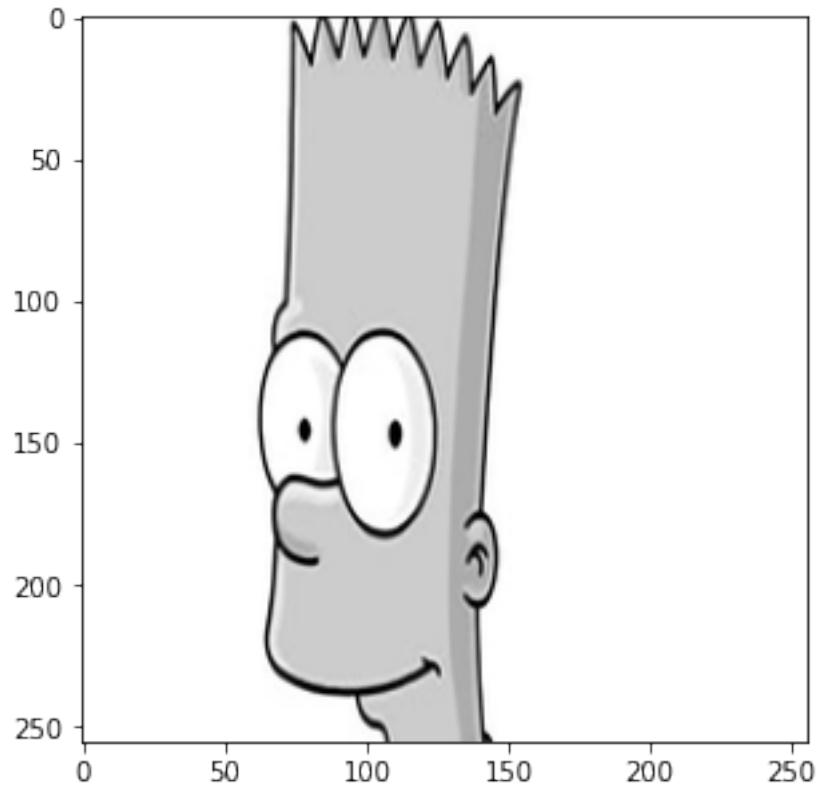
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2E90>



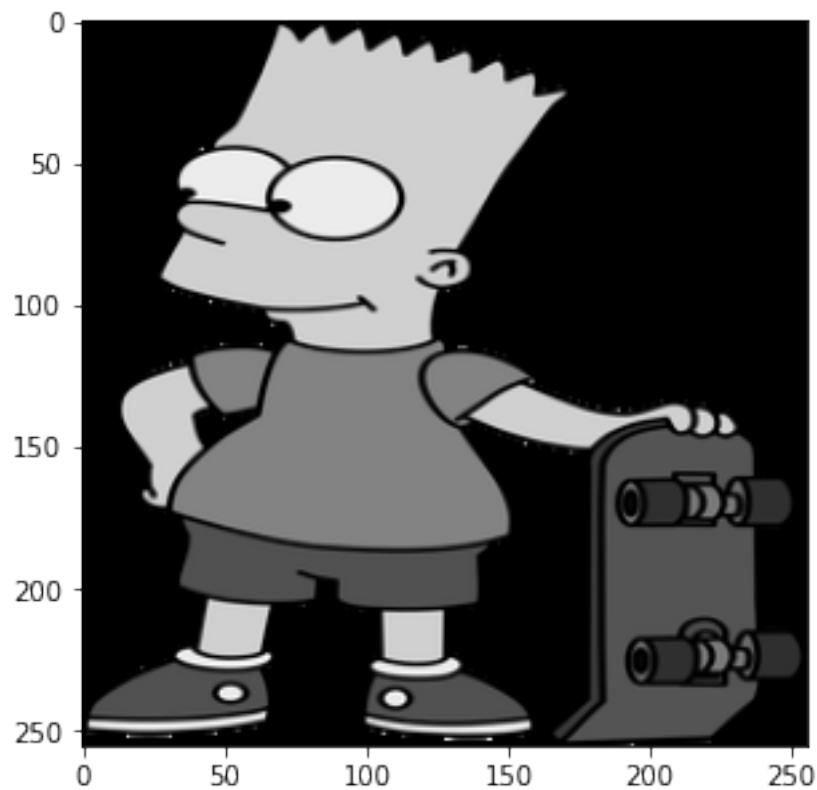
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2E10>



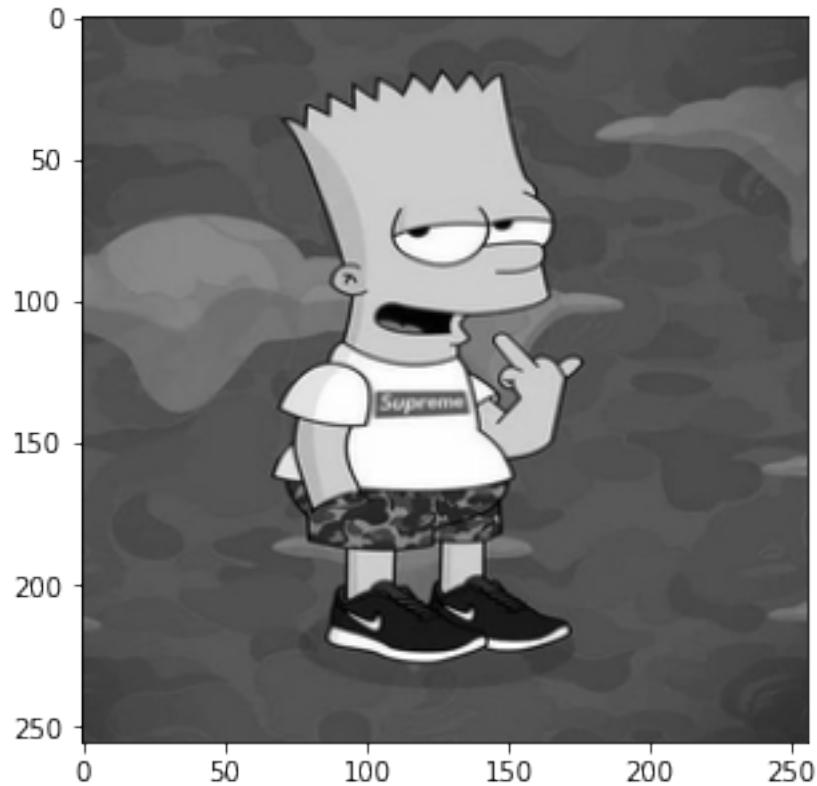
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2D90>



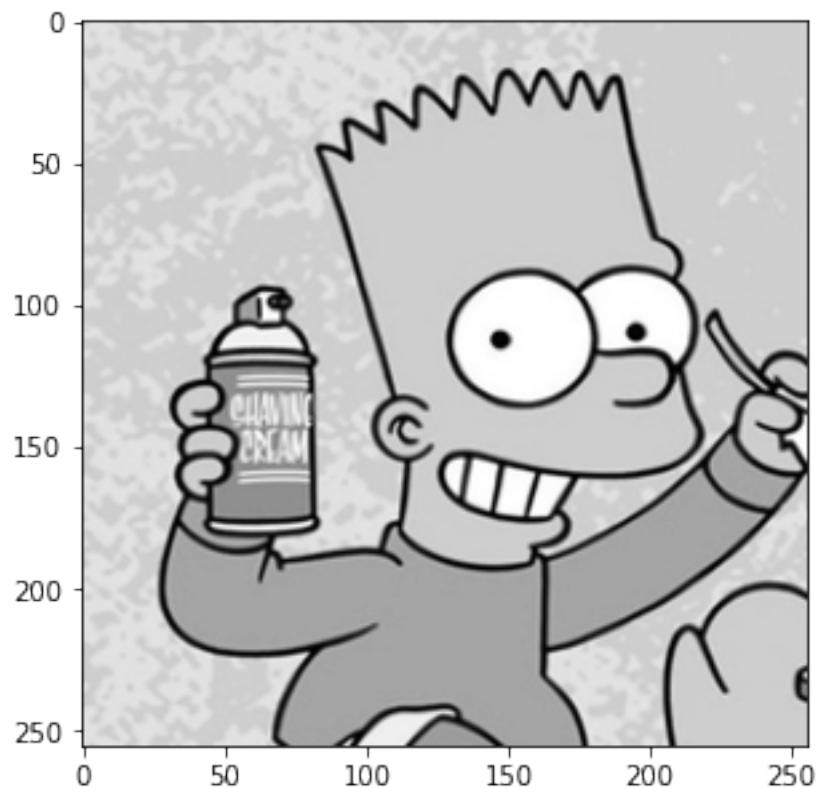
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2110>



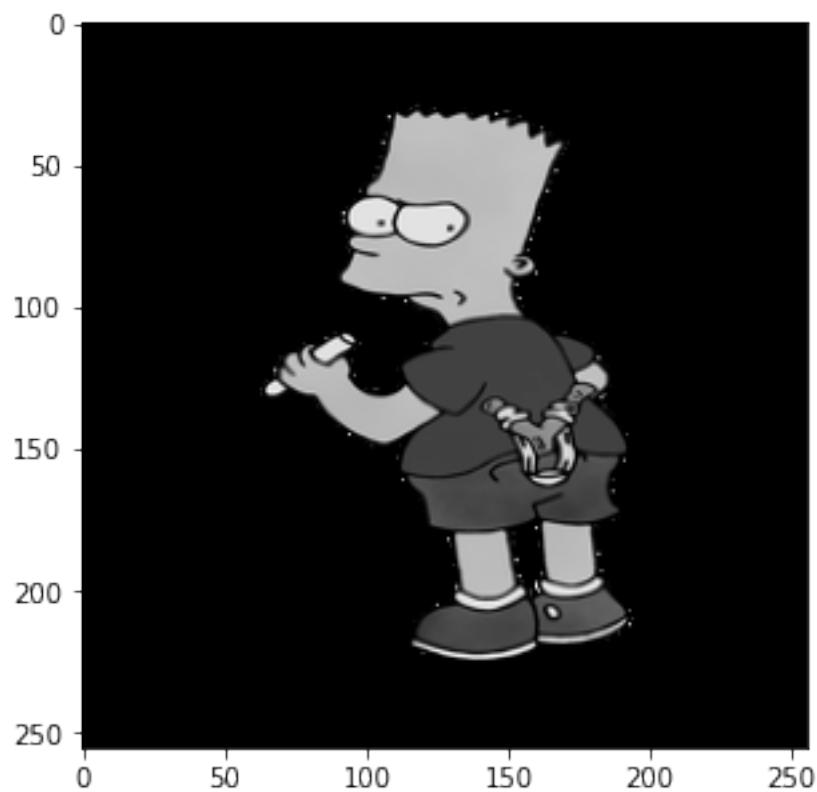
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2D10>



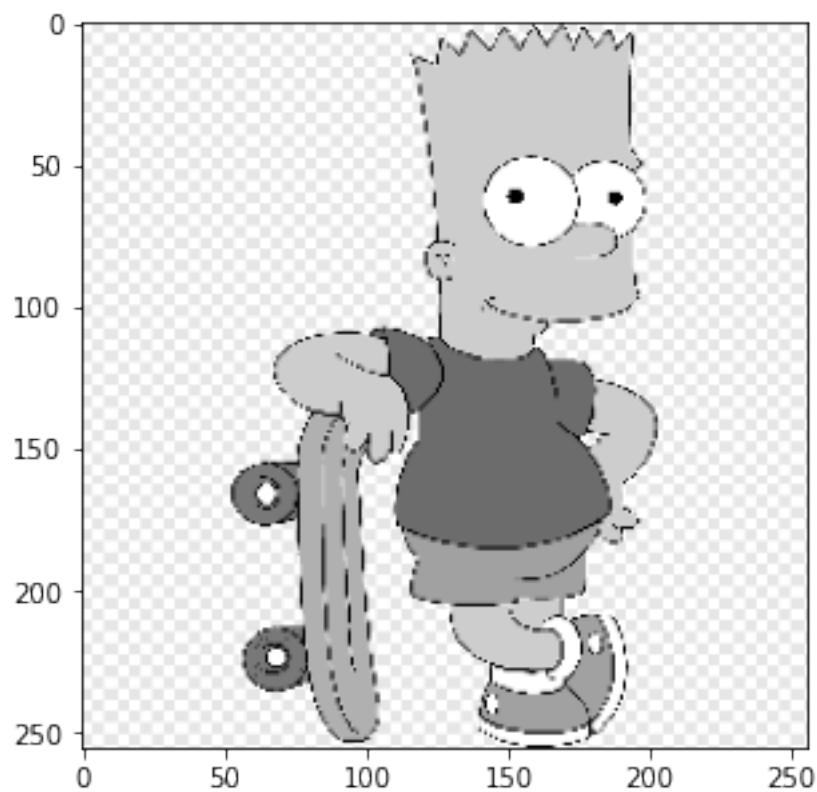
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2D50>



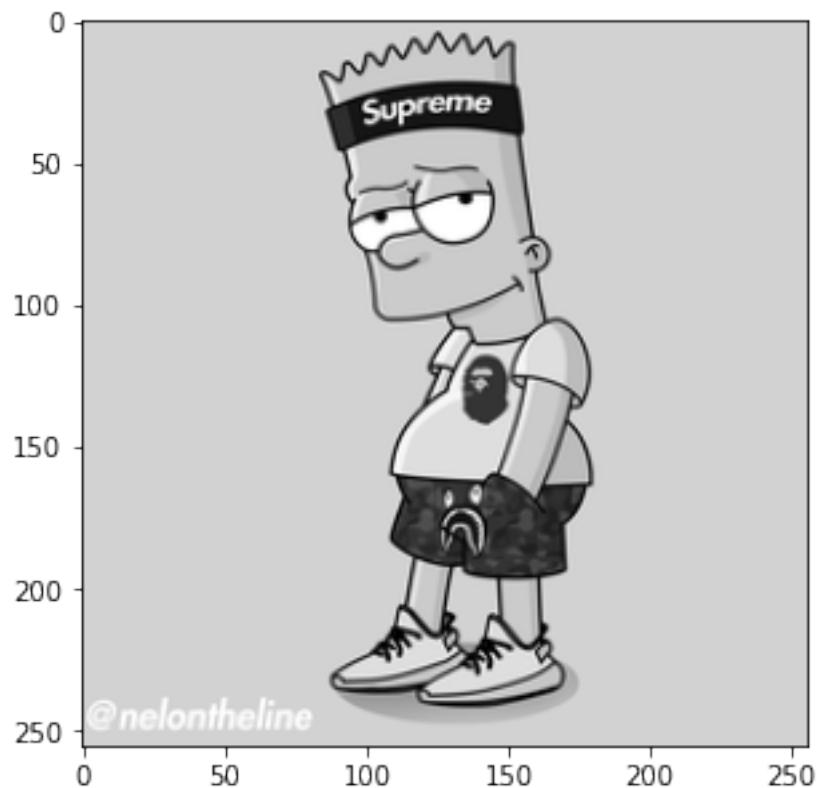
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2CD0>



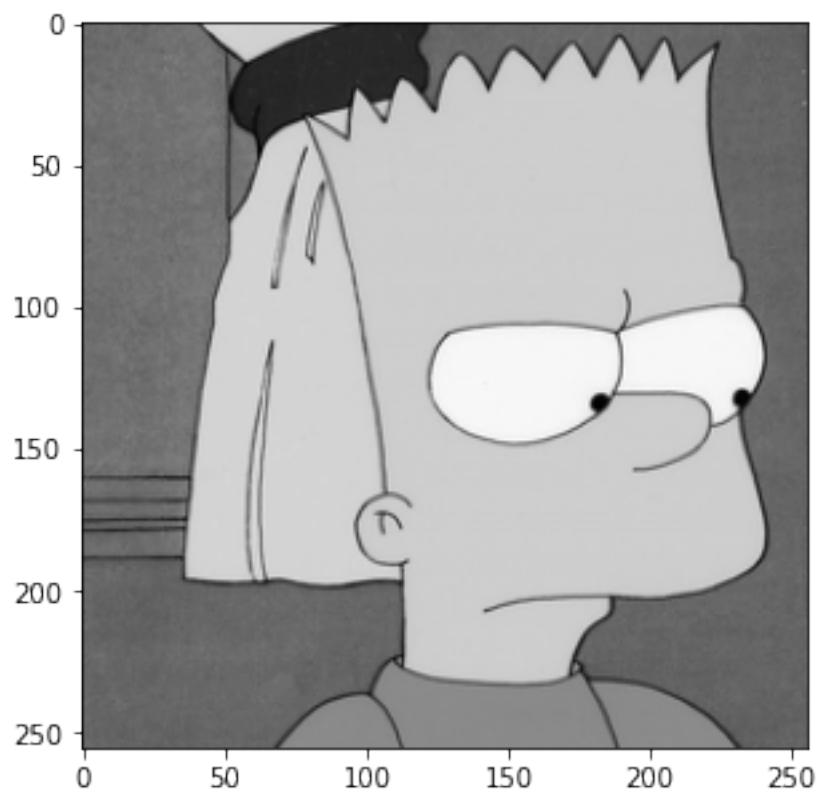
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2C90>



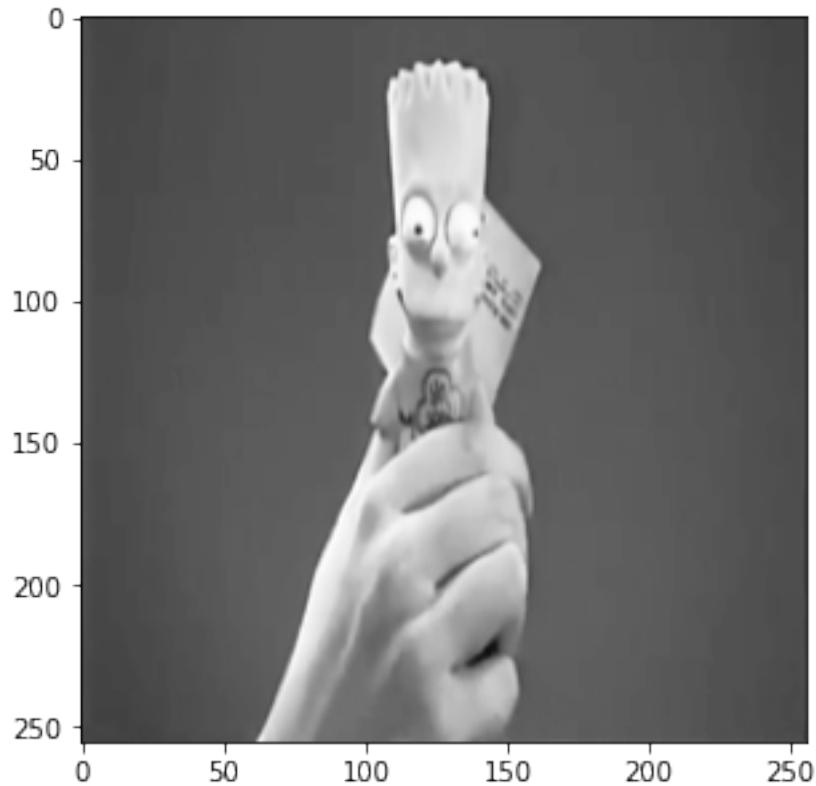
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2C50>



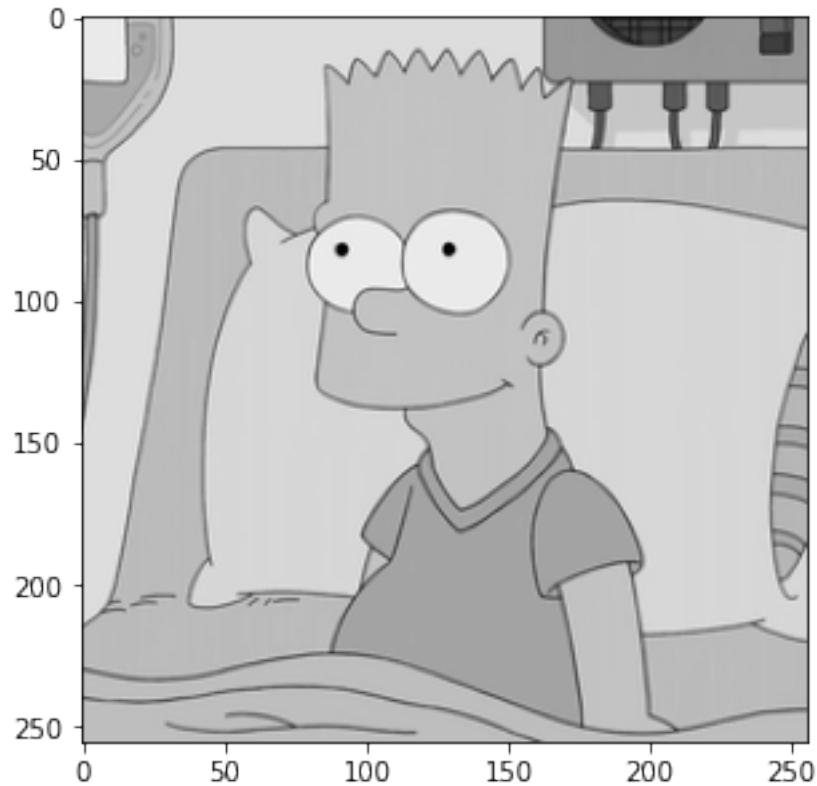
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2B90>



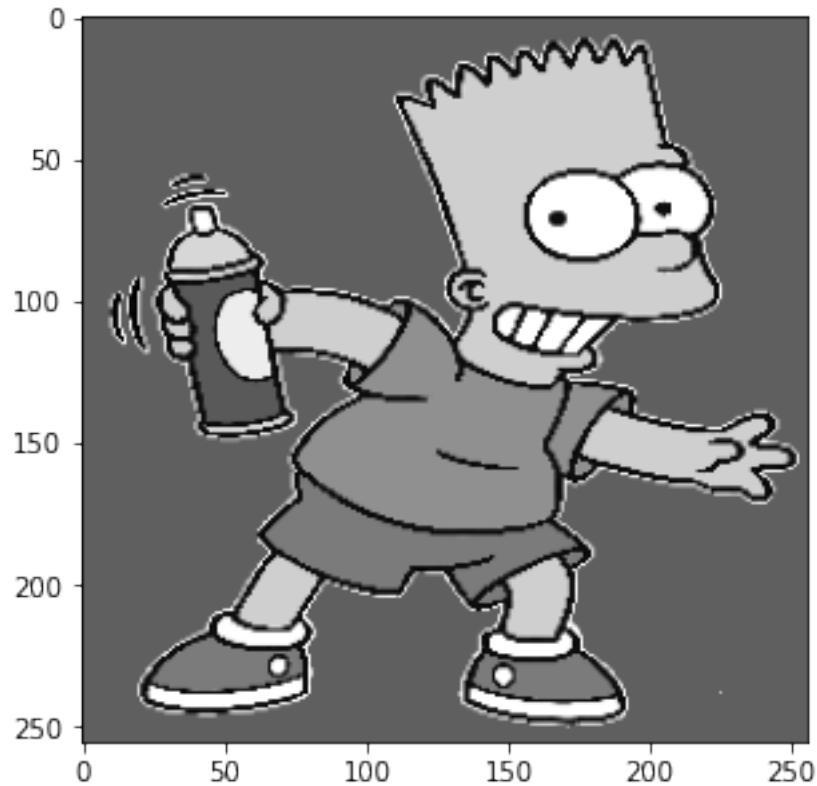
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2B50>



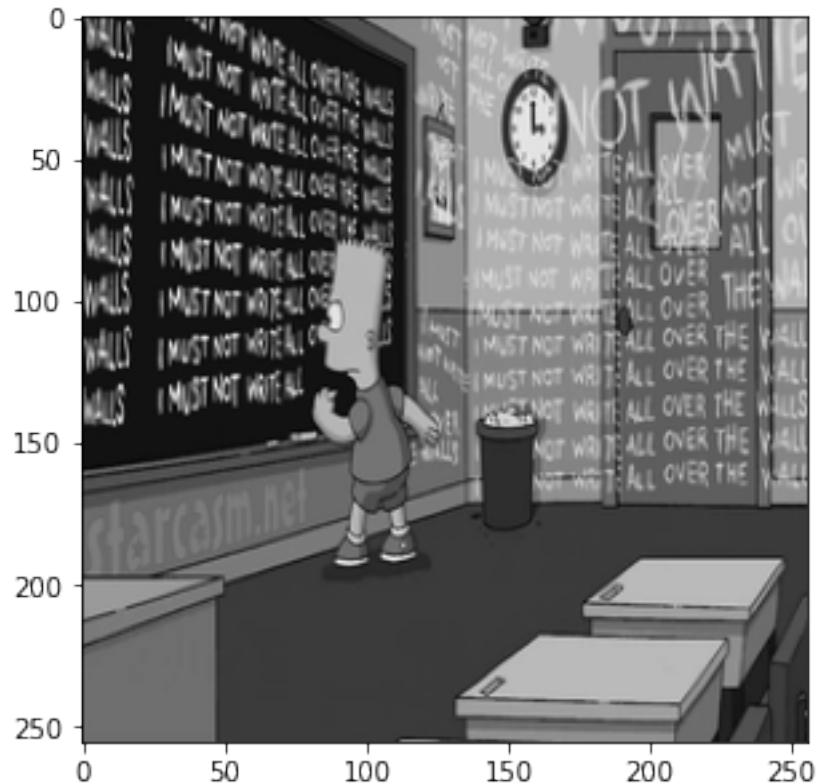
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2B10>



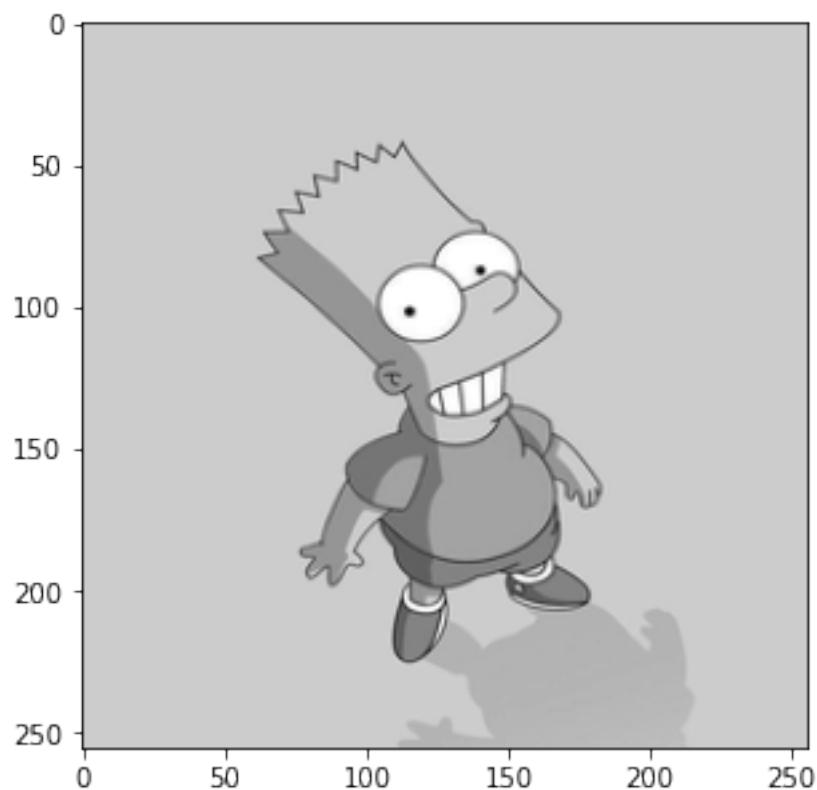
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2AD0>



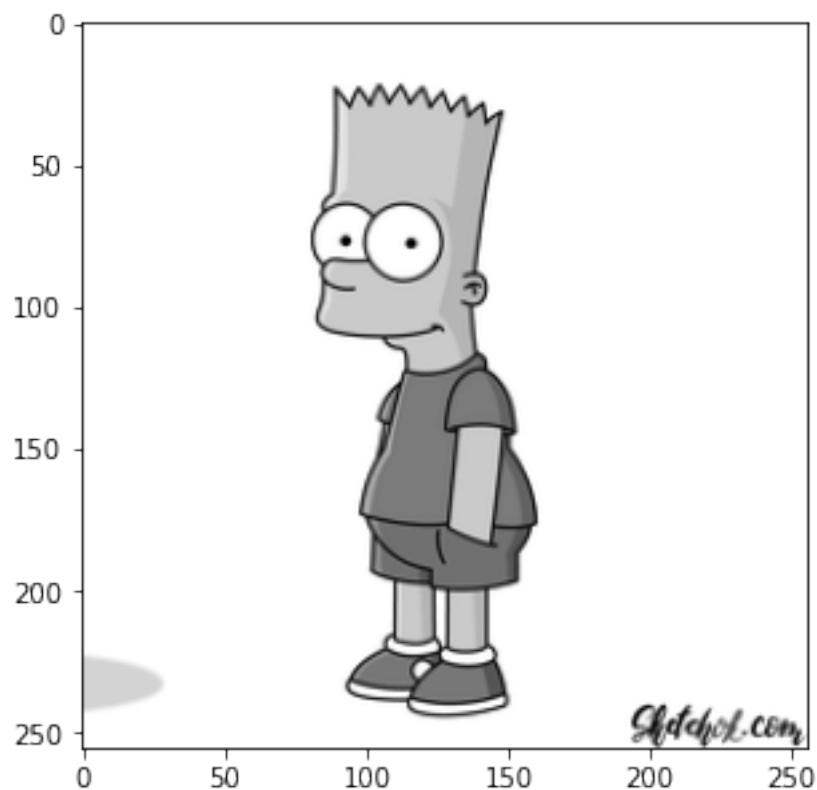
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2950>



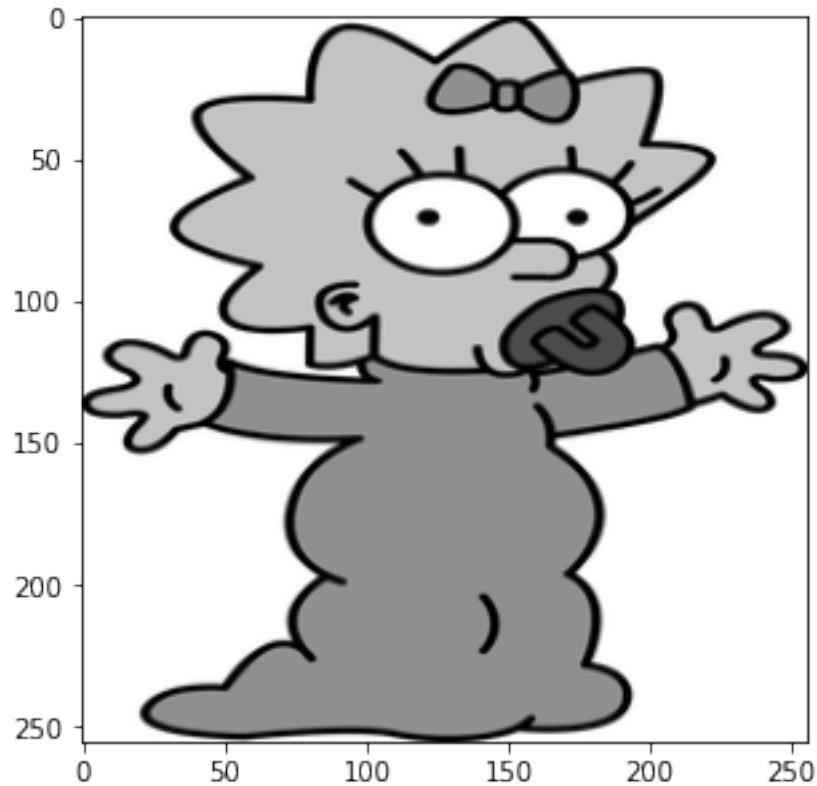
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B29D0>



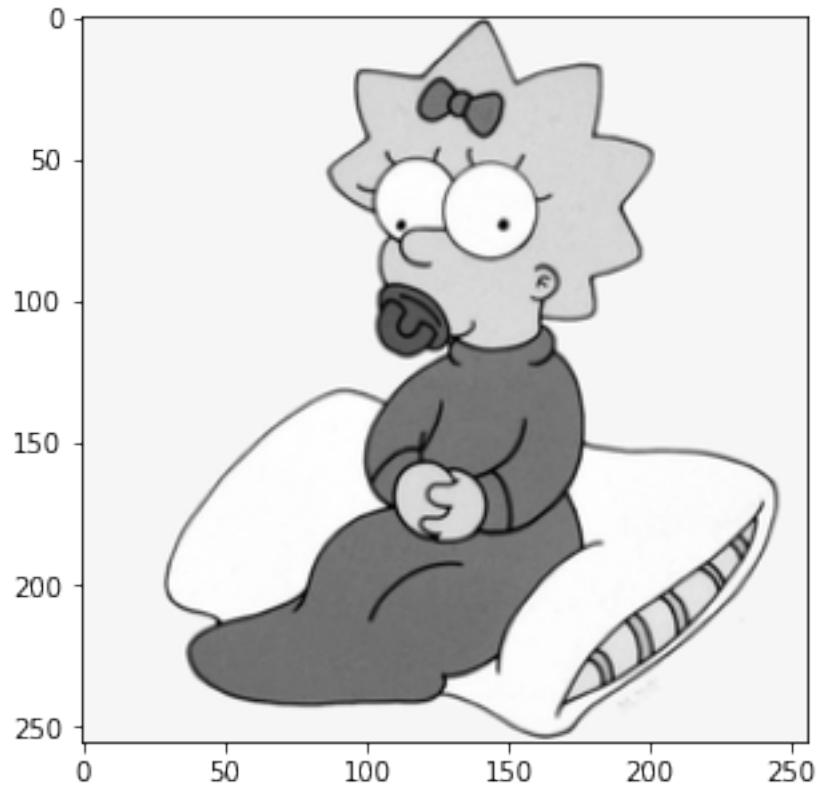
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197B2910>



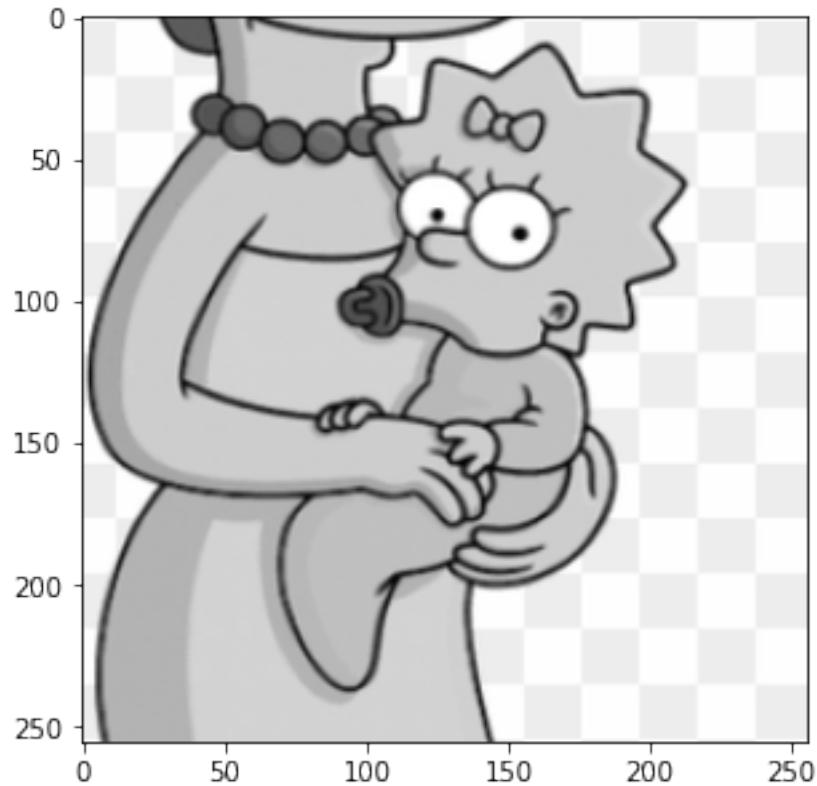
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E810>



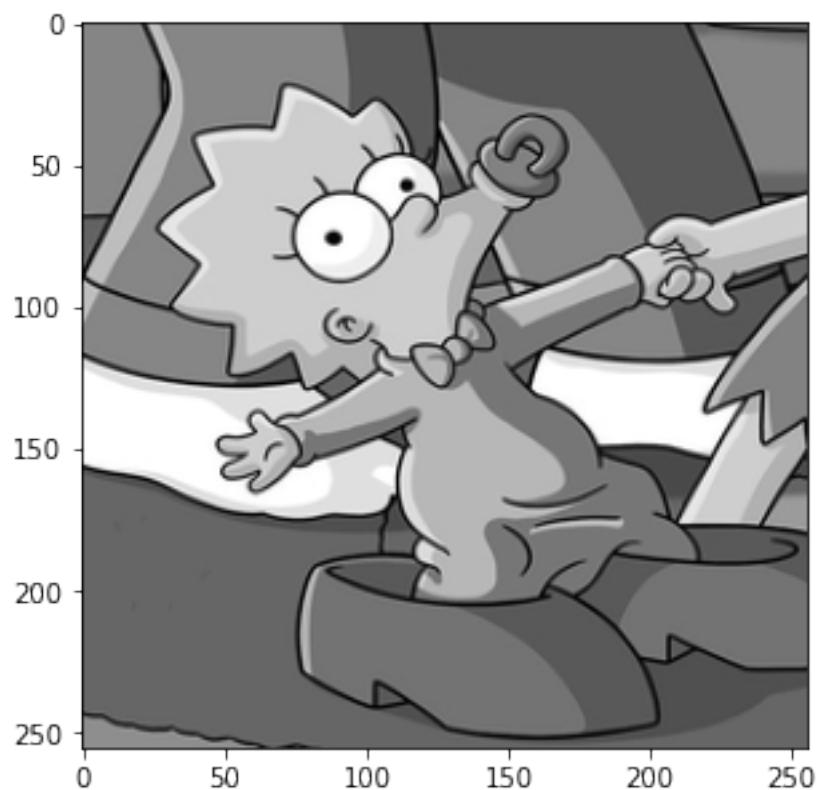
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E890>



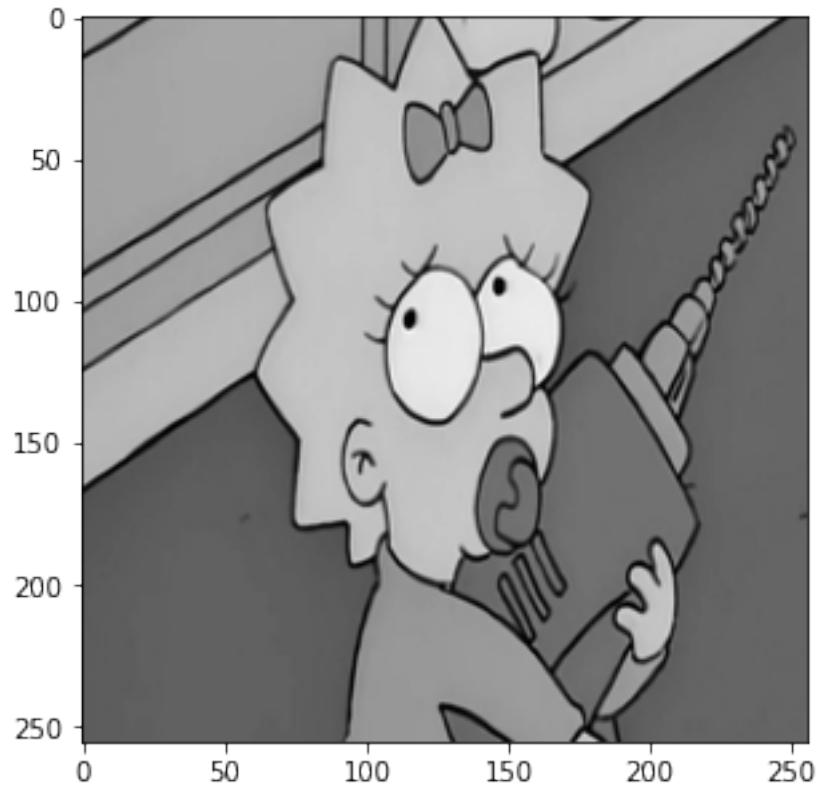
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E650>



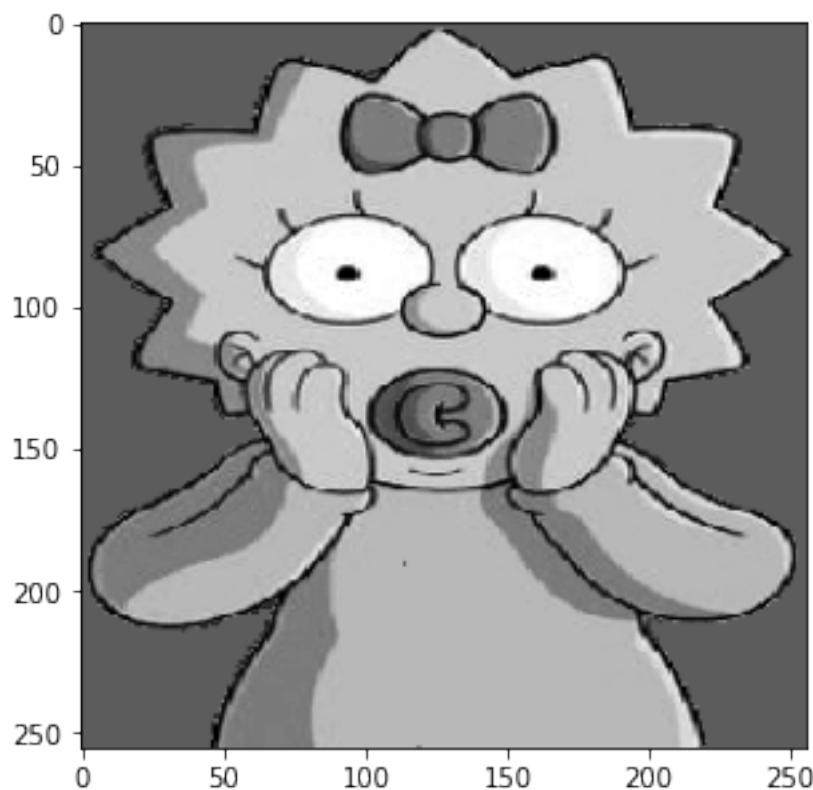
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8EA50>



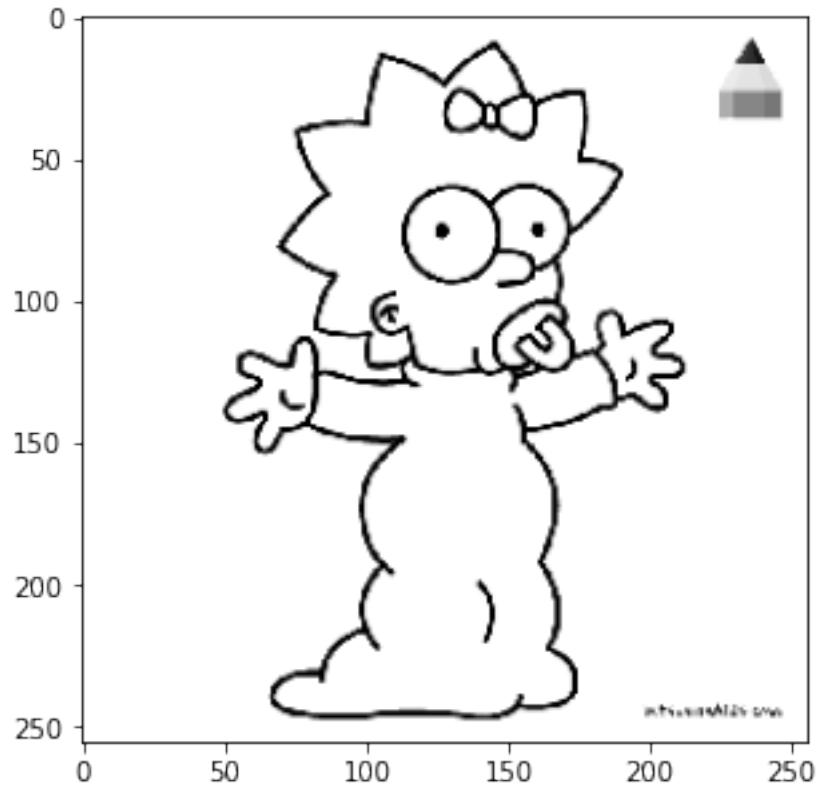
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E950>



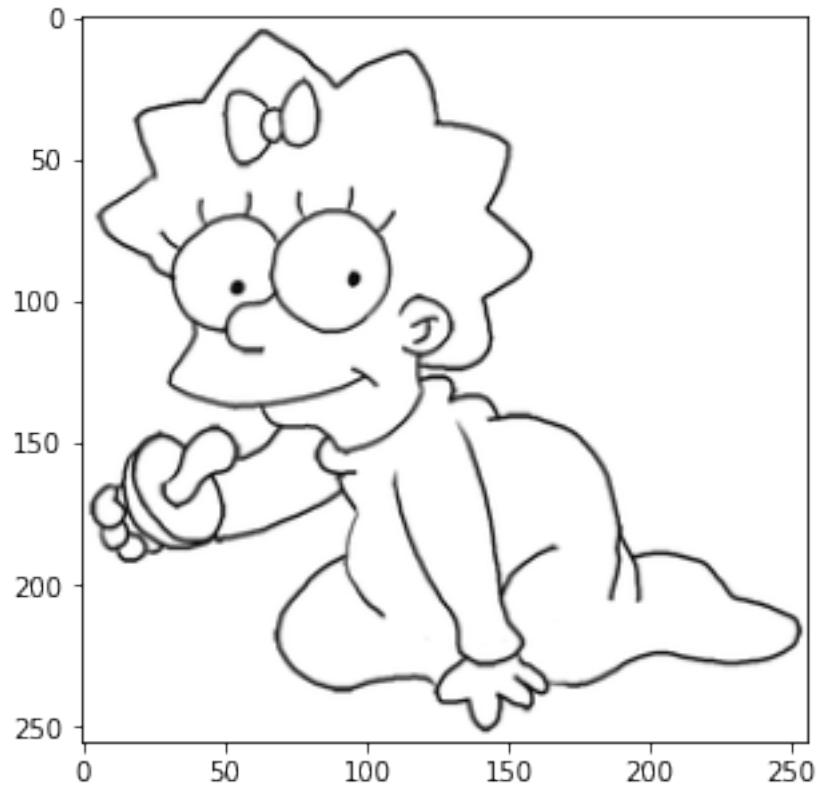
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E9D0>



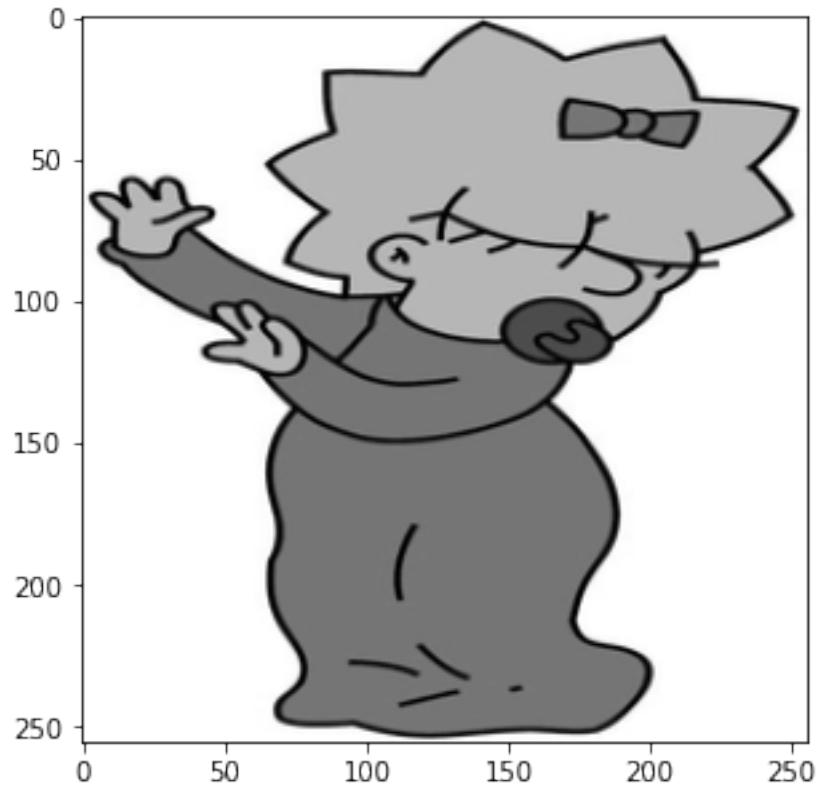
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8EAD0>



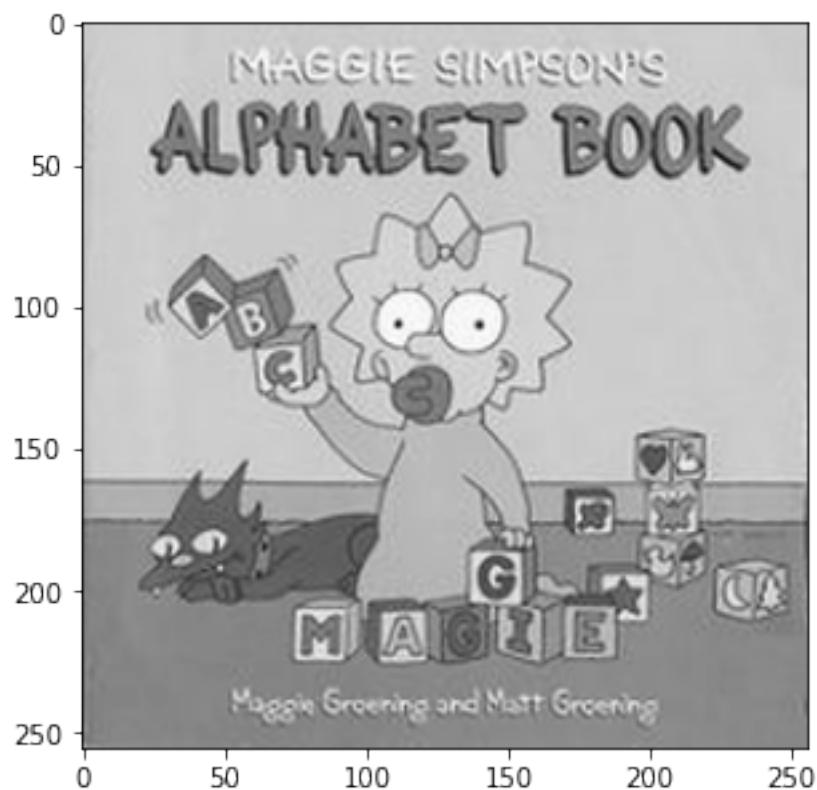
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E510>



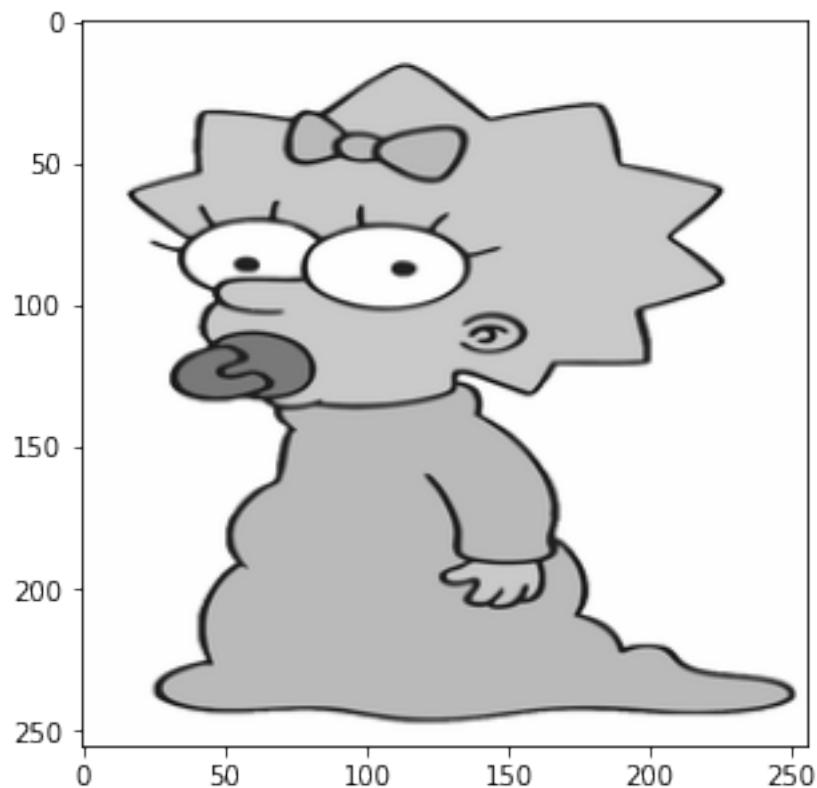
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119BF1050>



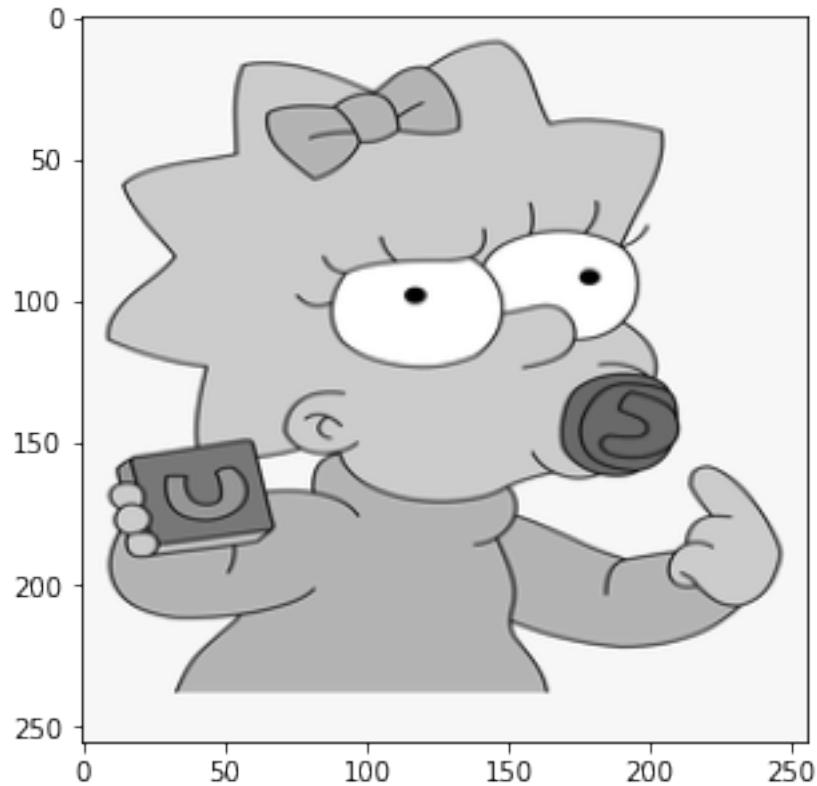
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119B8E610>



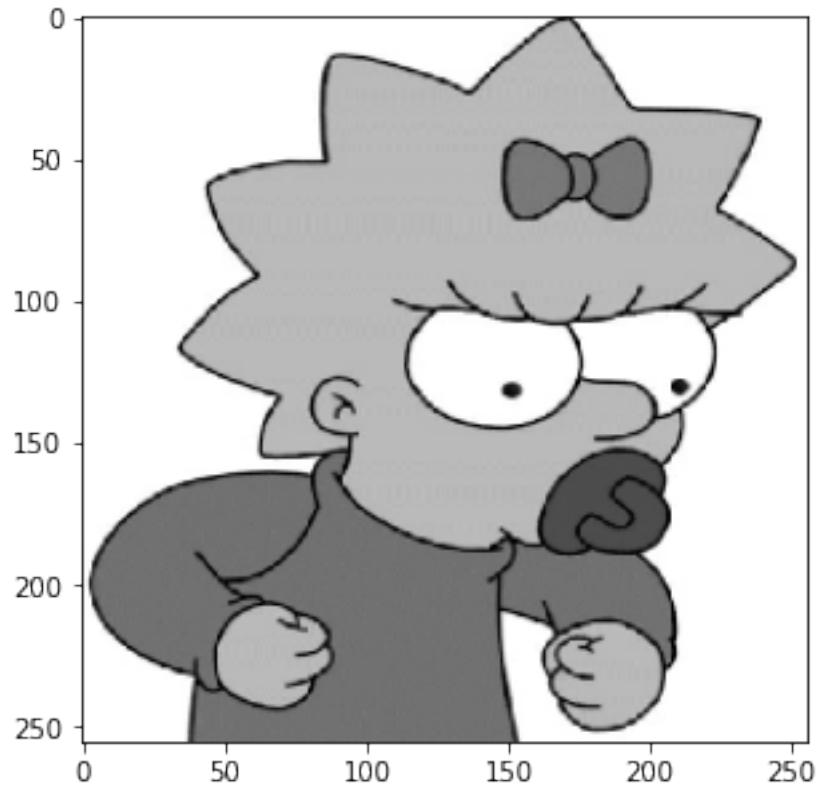
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795210>



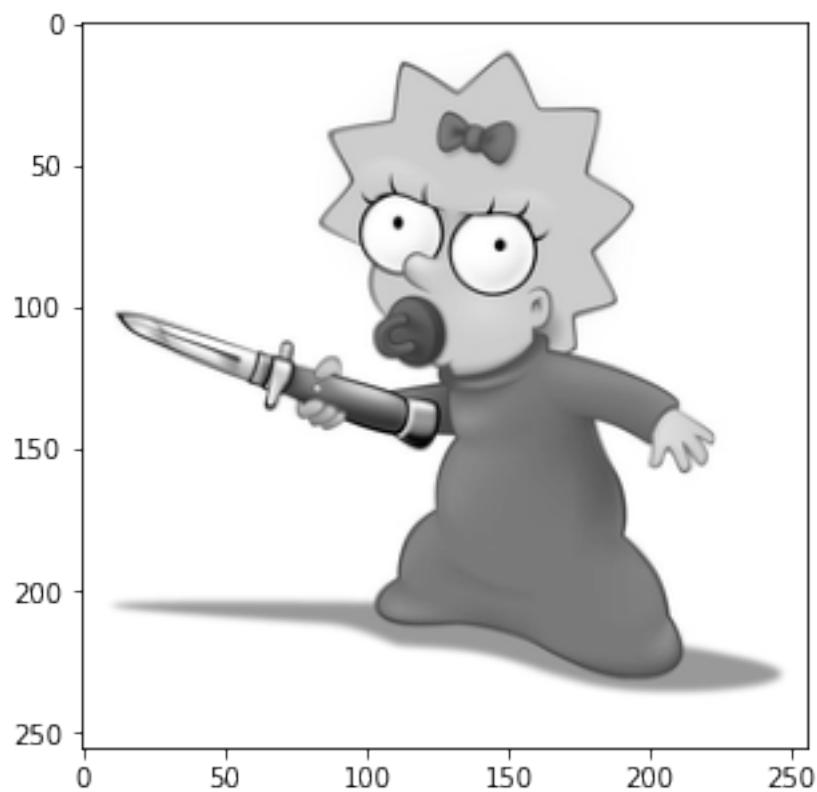
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795250>



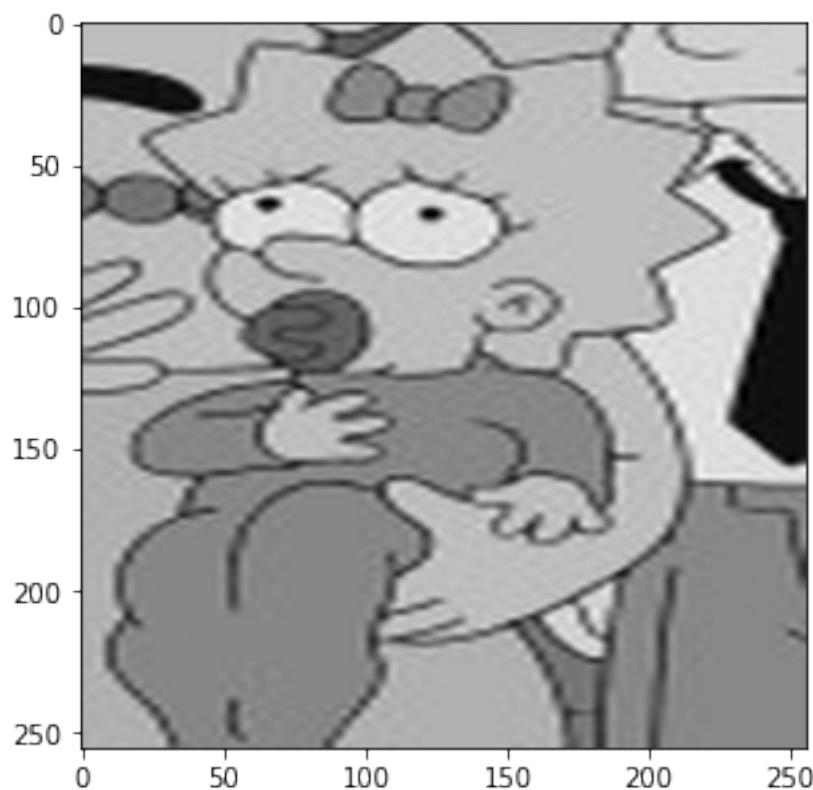
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795290>



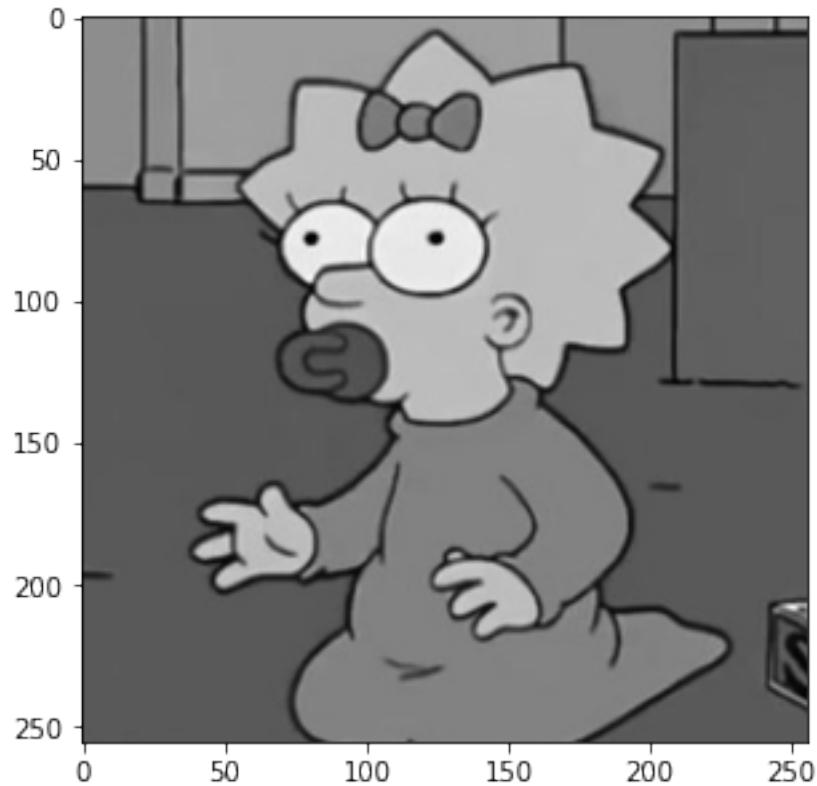
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197952D0>



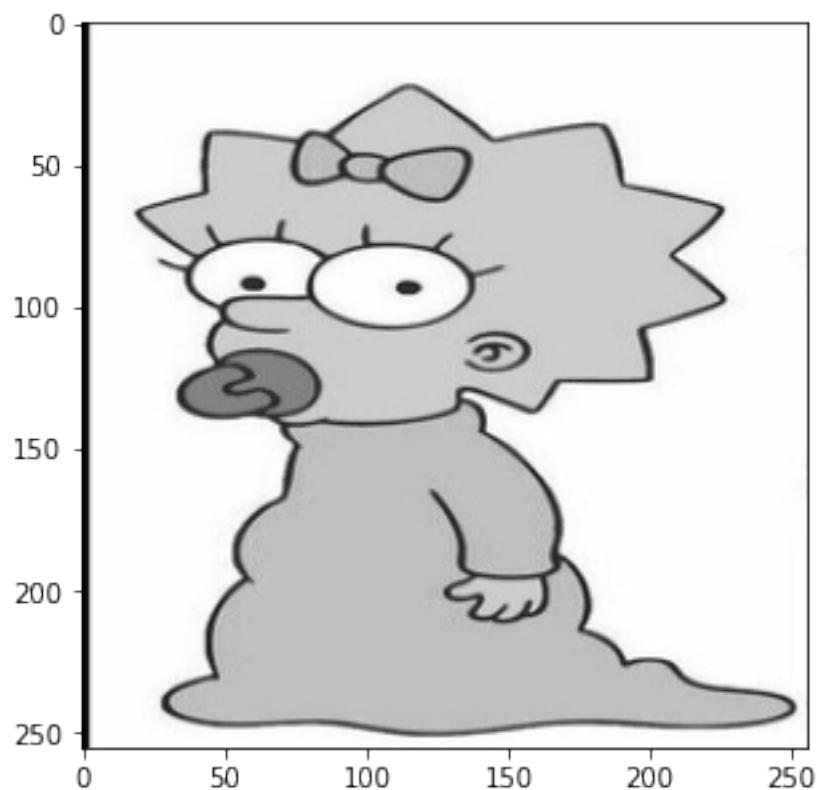
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795310>



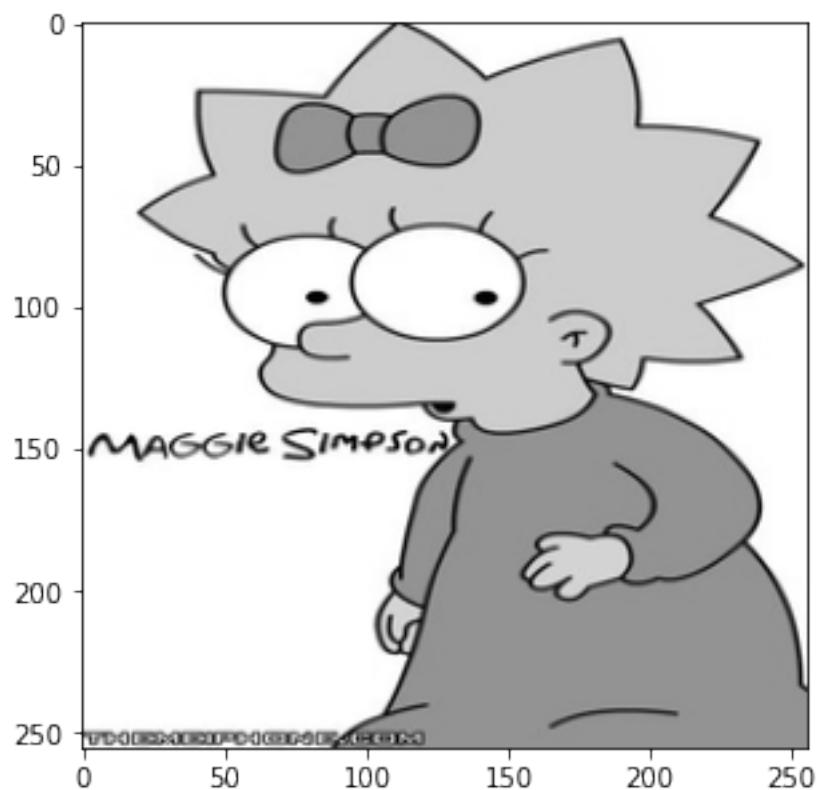
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795350>



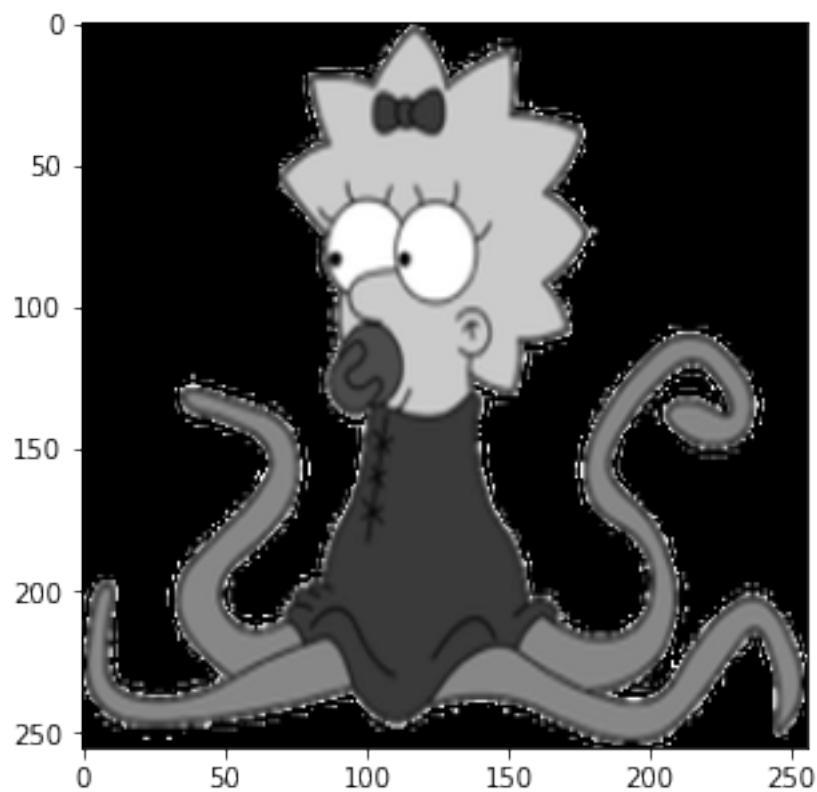
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795390>



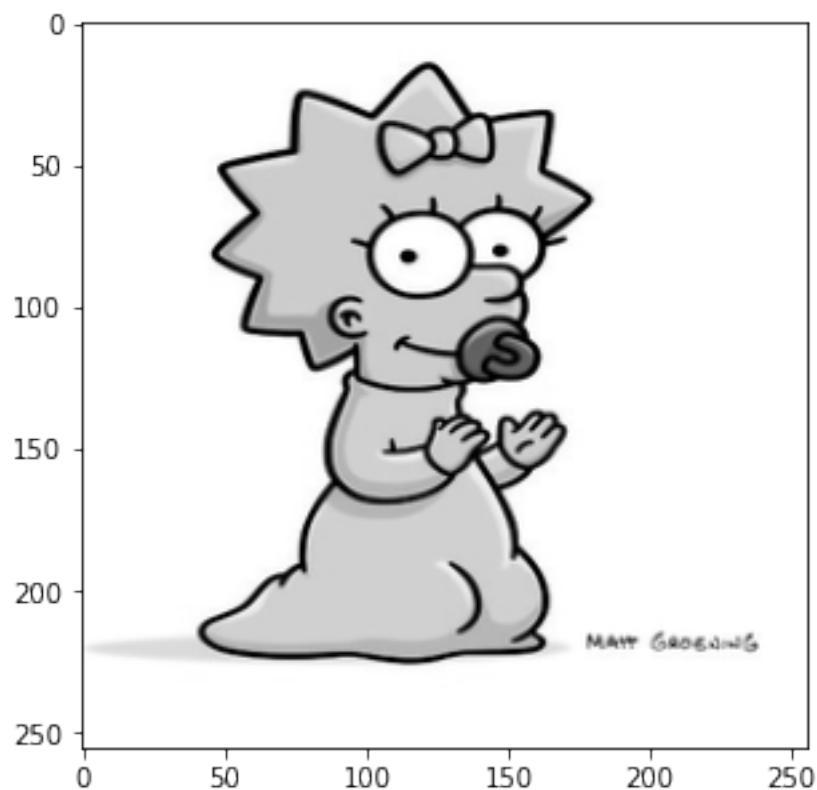
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197953D0>



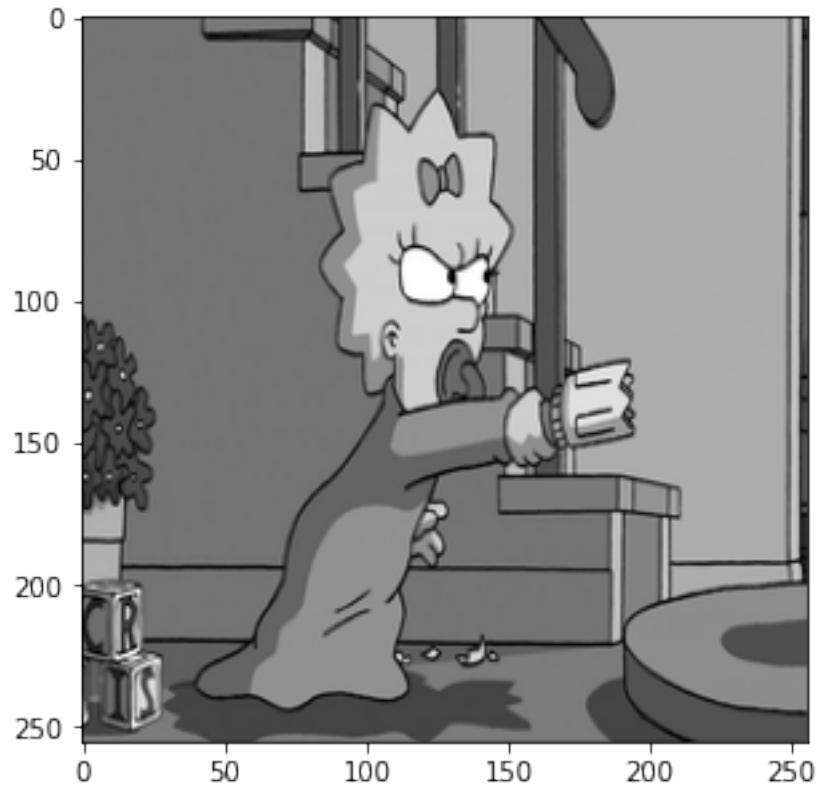
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795410>



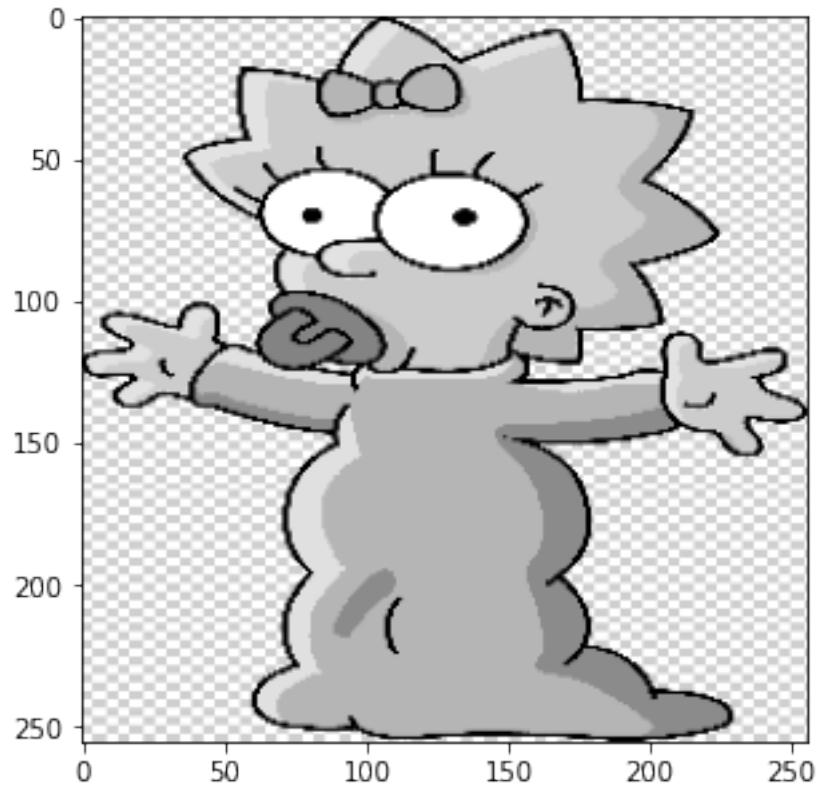
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795450>



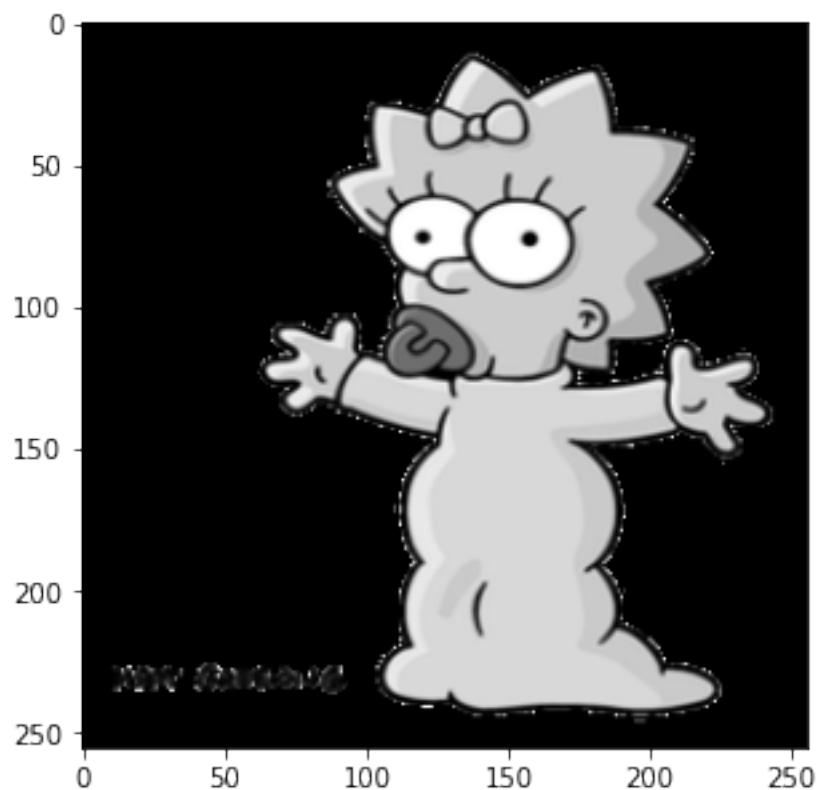
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795490>



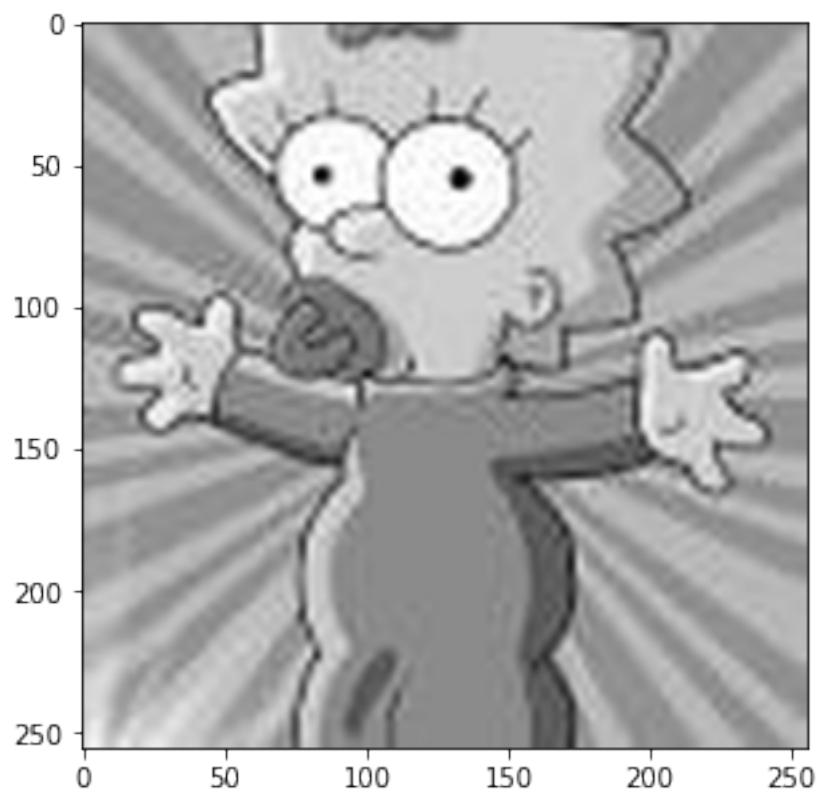
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197954D0>



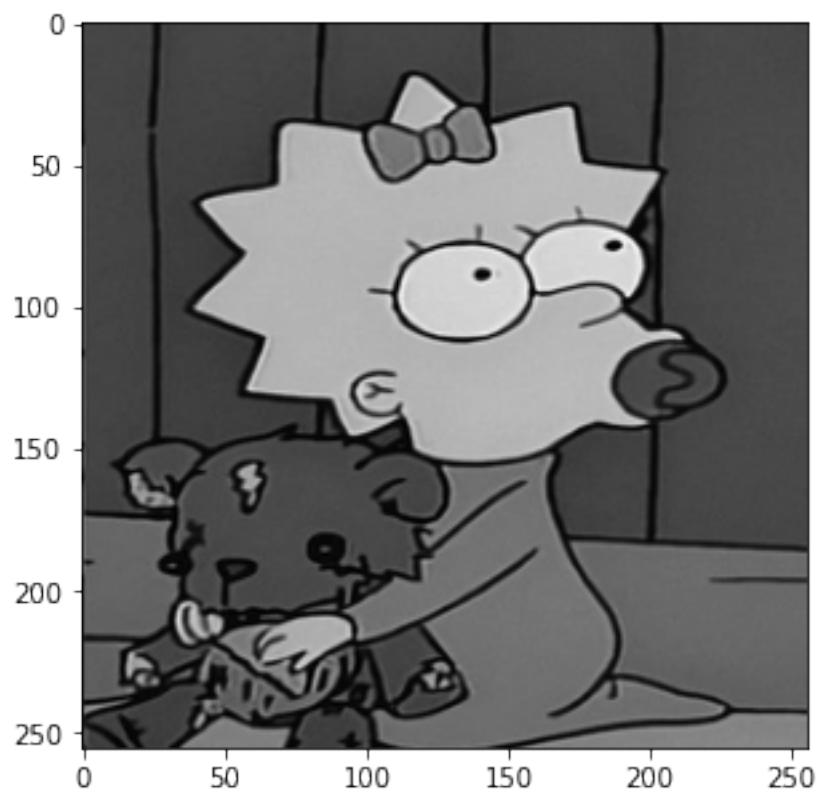
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795510>



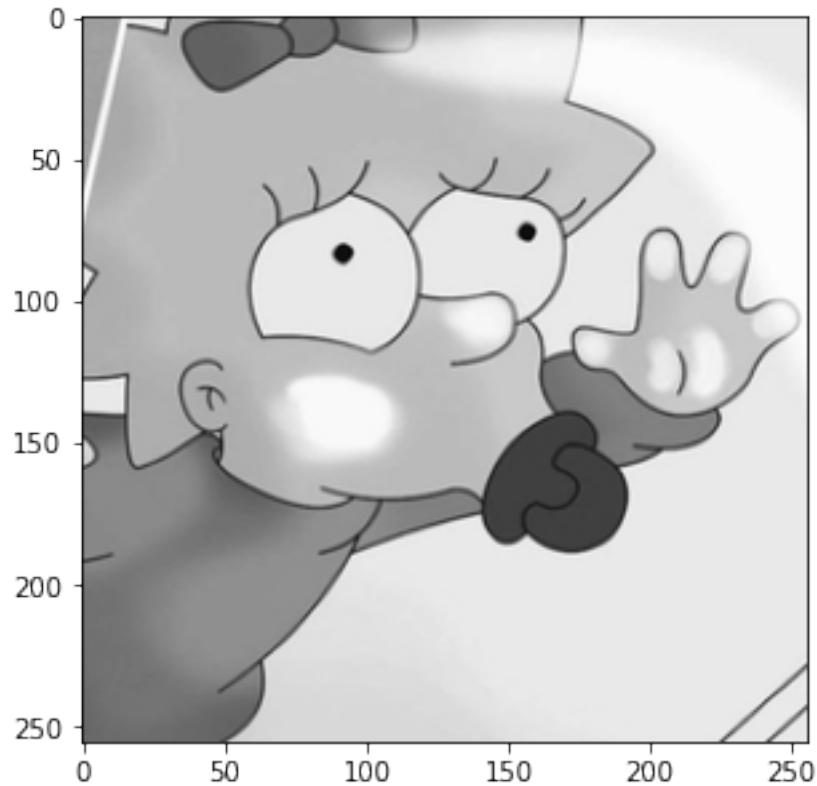
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB11979550>
```



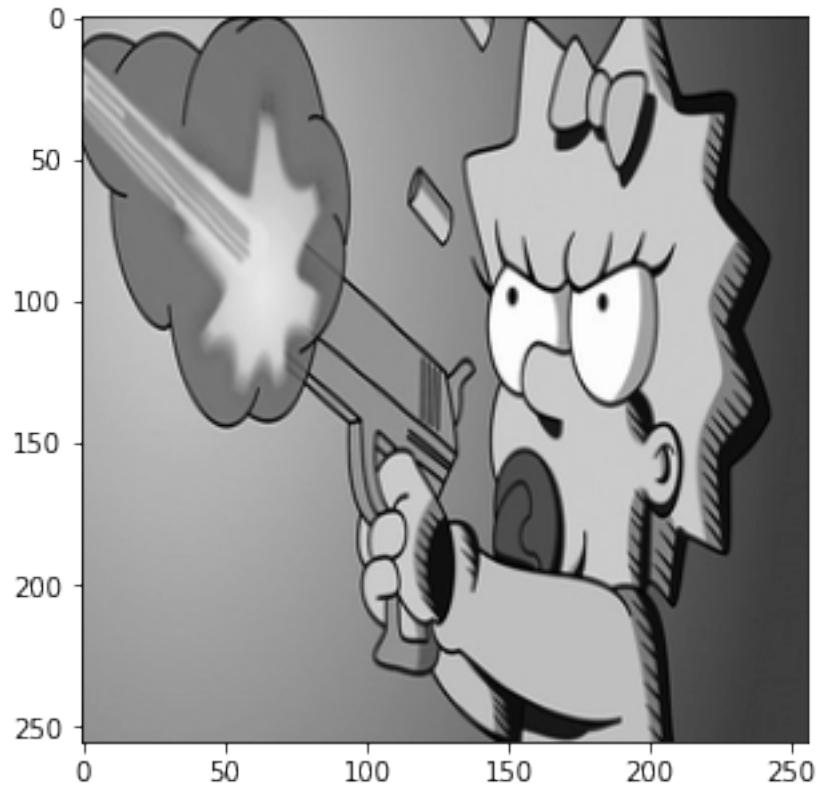
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795590>



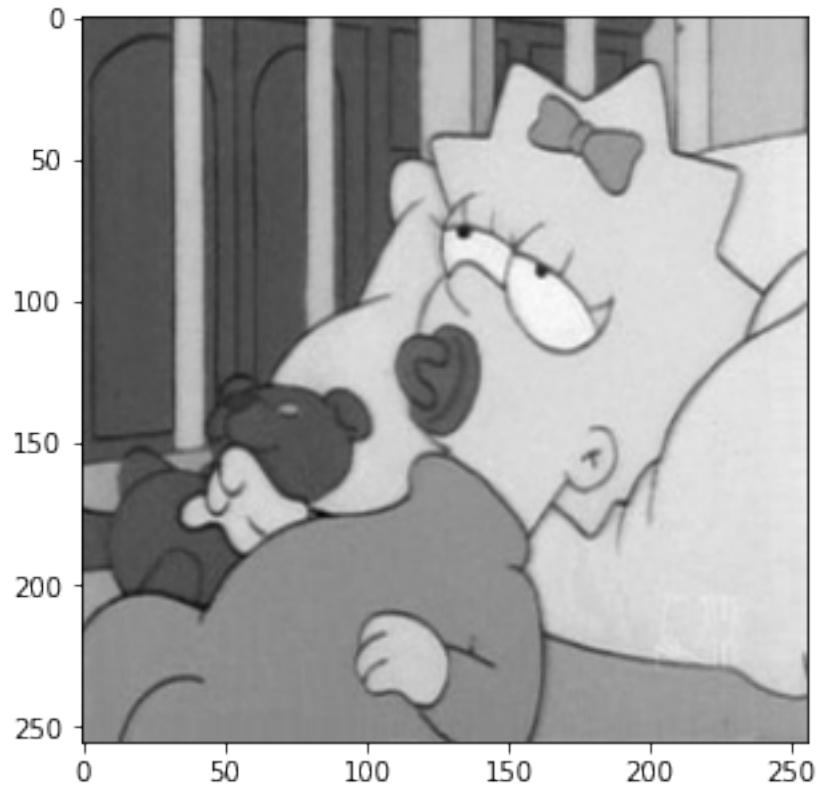
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197955D0>



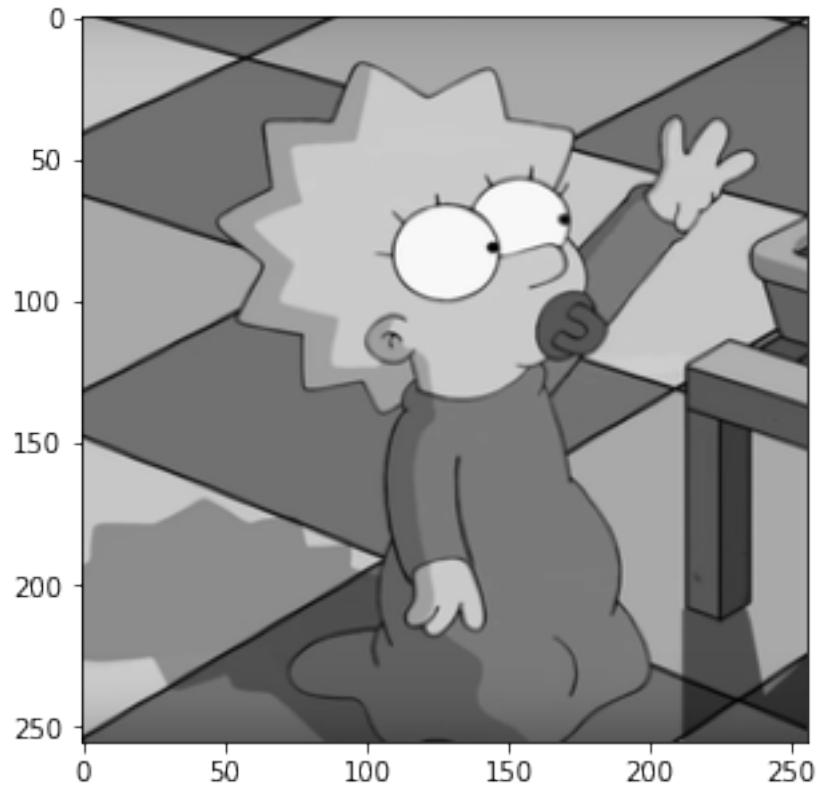
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795610>



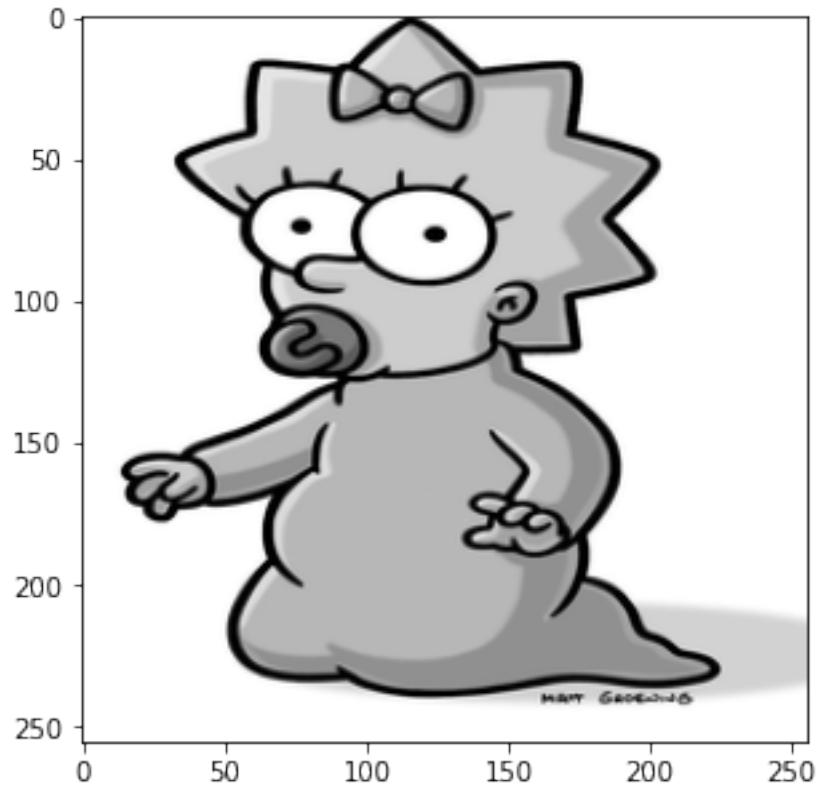
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795650>



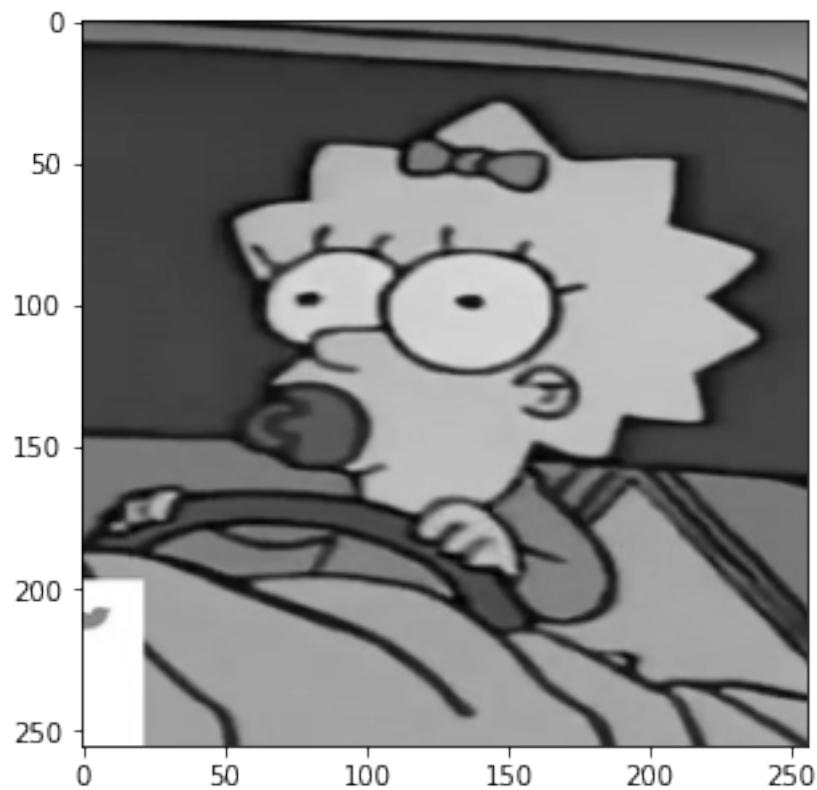
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795690>



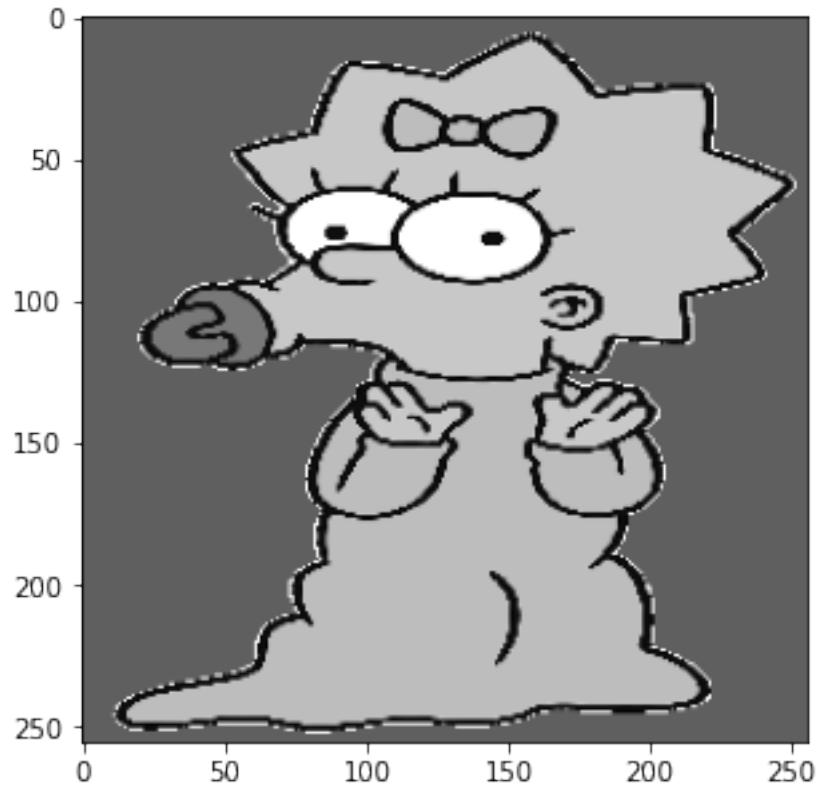
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197956D0>



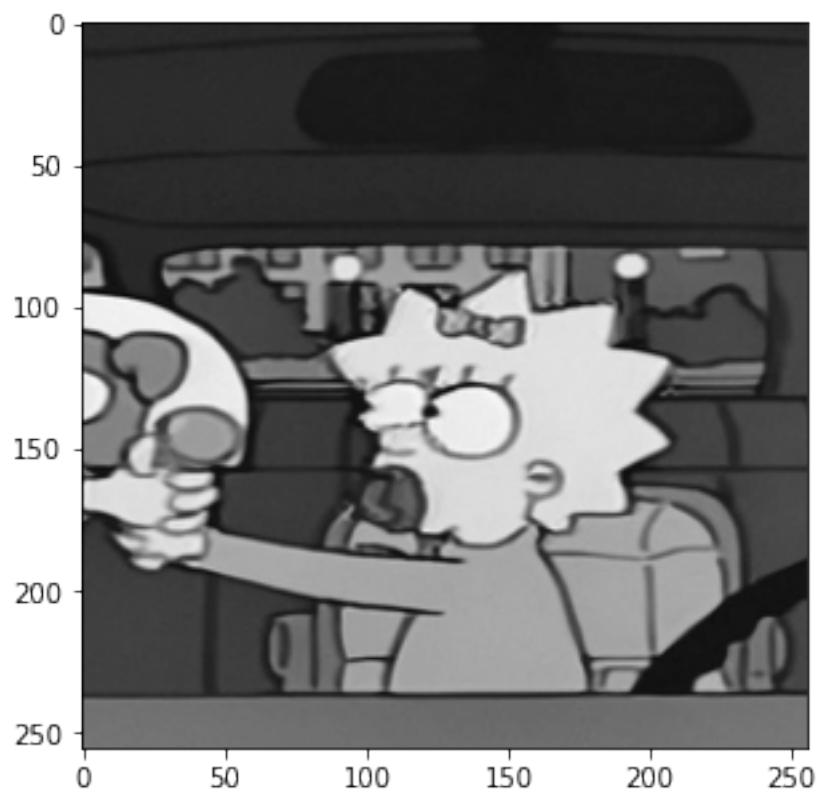
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795710>



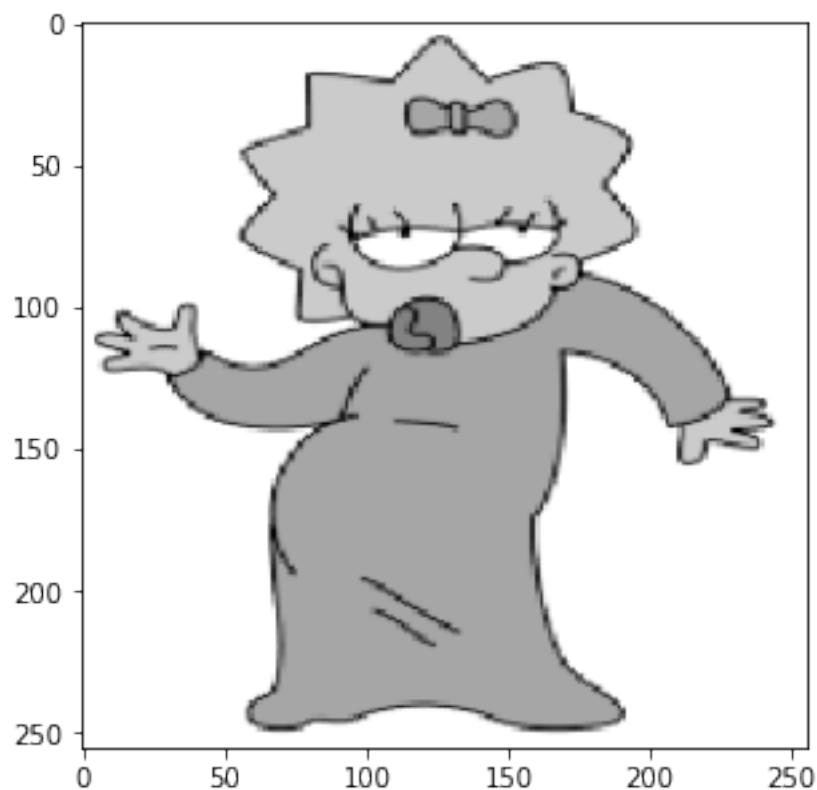
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795750>



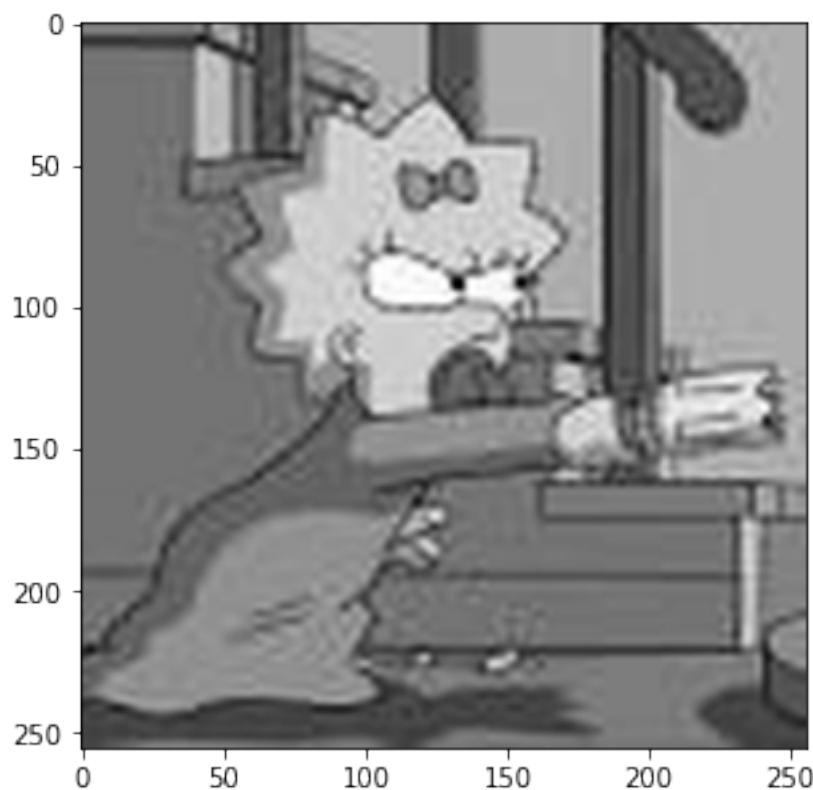
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795790>



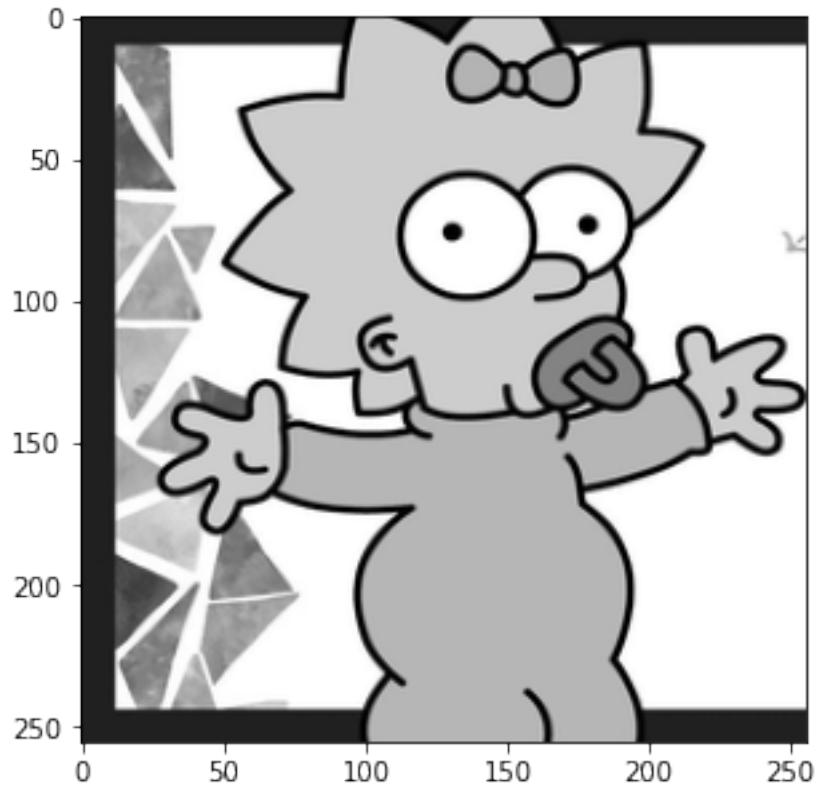
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197957D0>



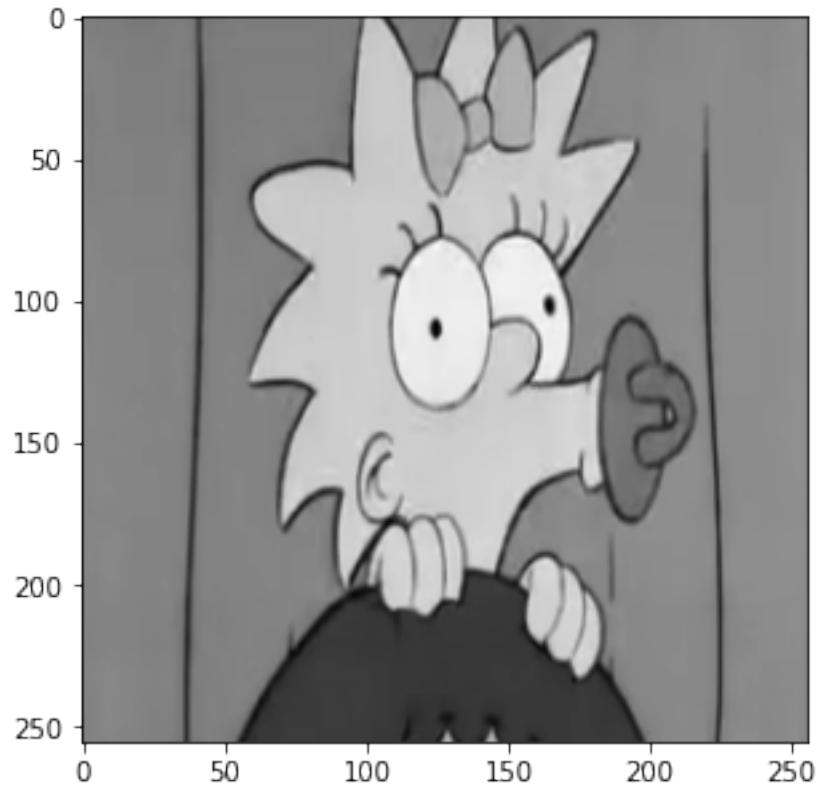
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795810>



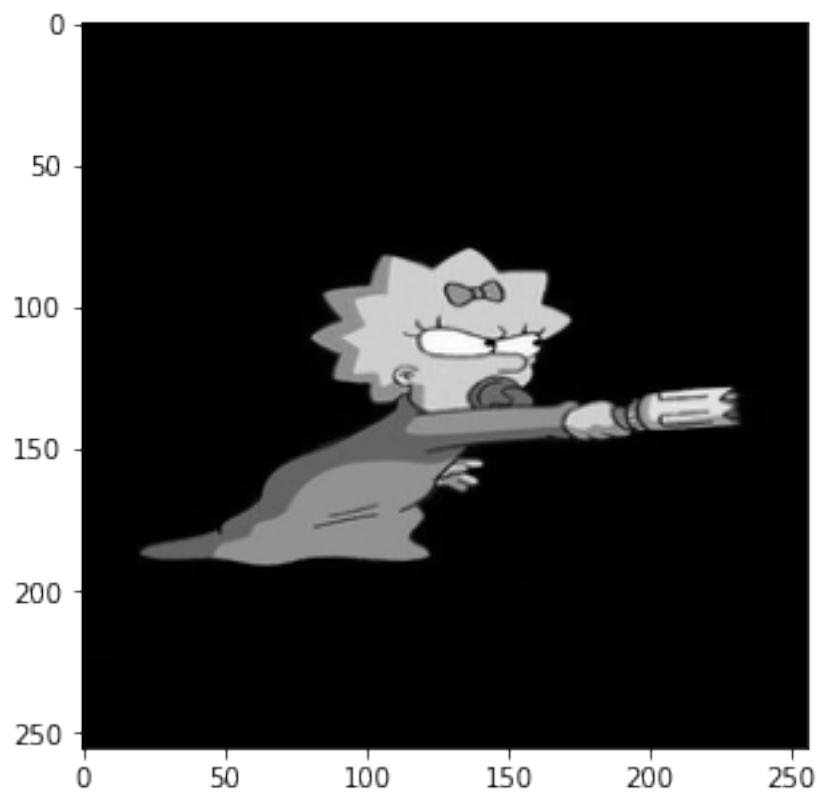
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795850>



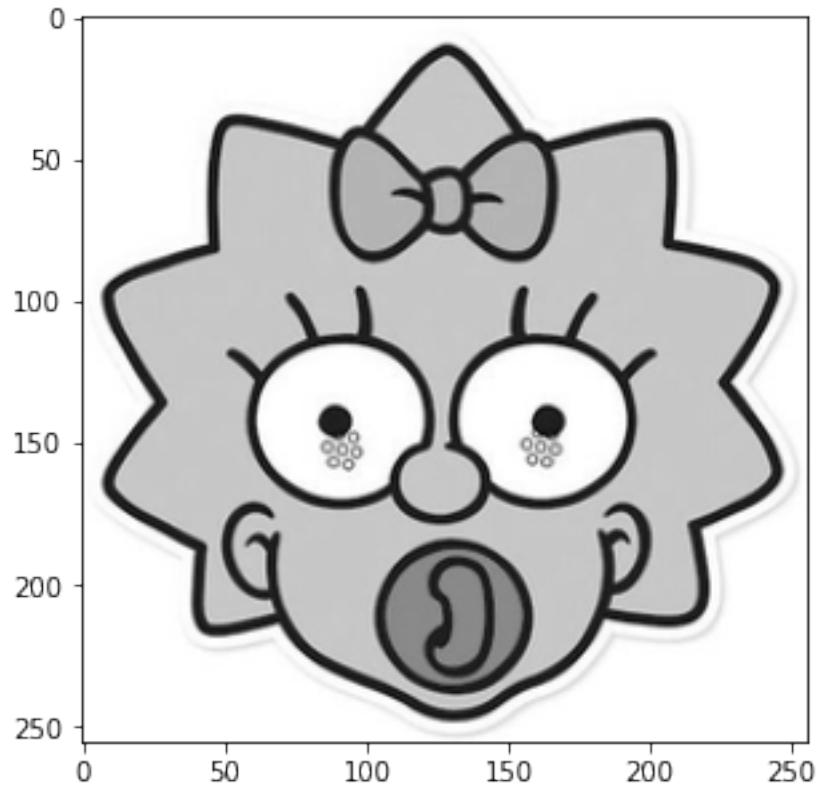
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795890>



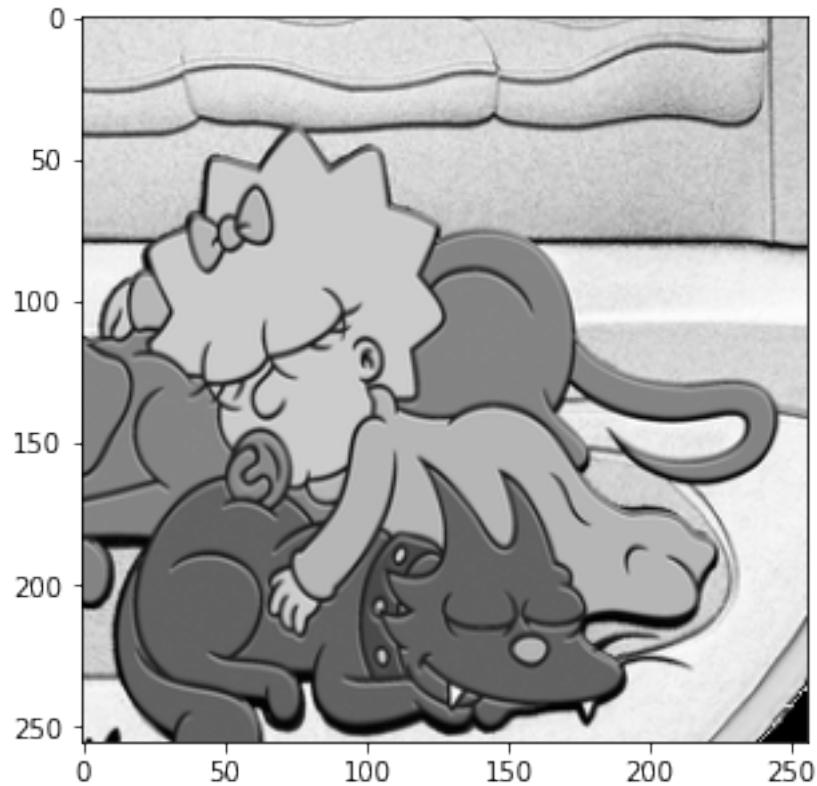
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197958D0>



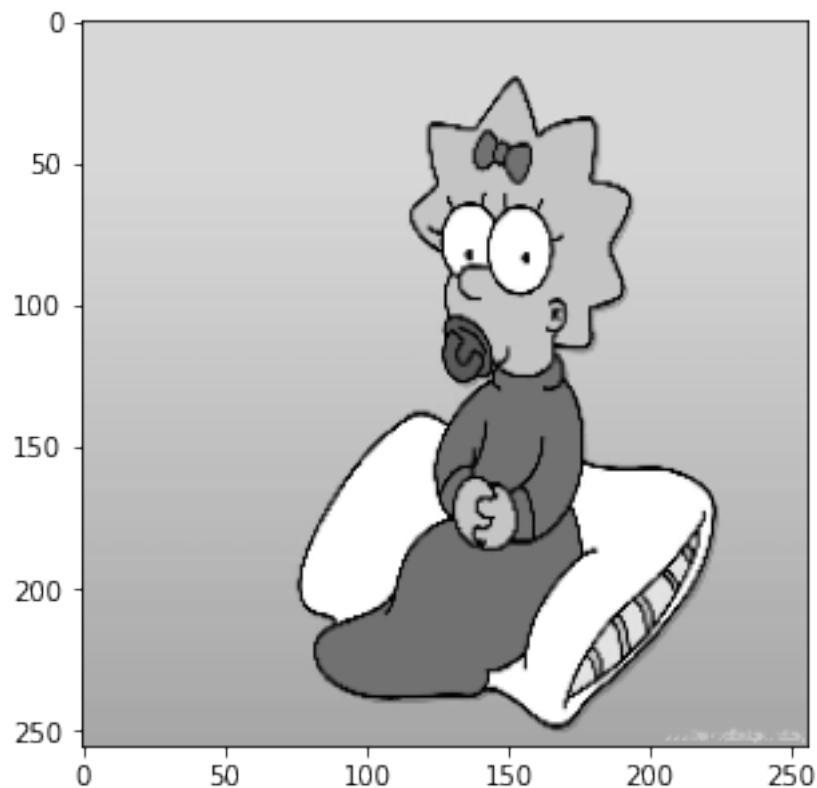
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795910>



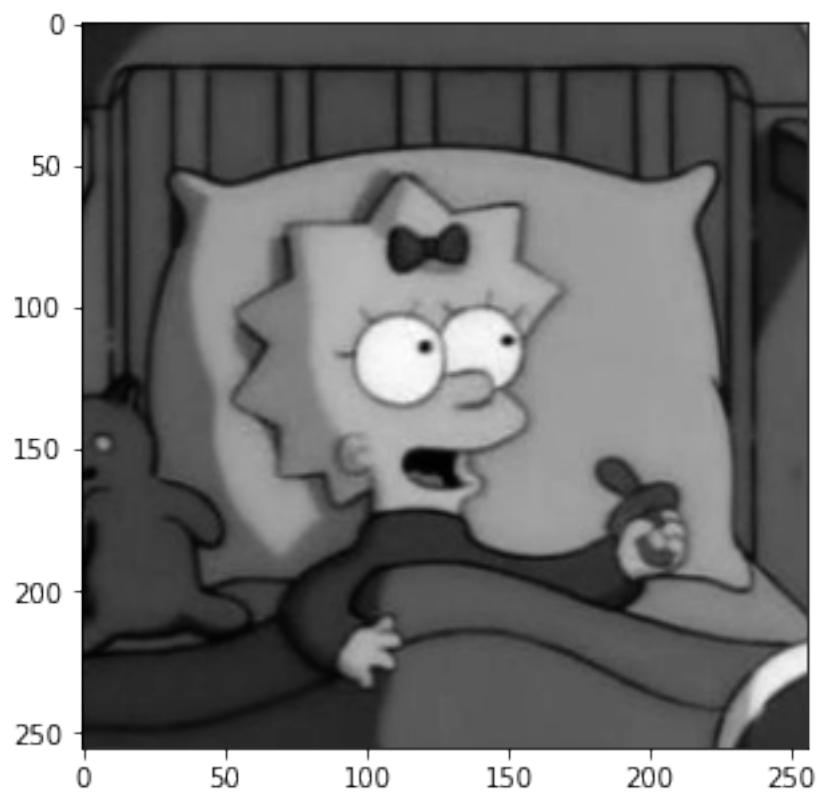
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795950>



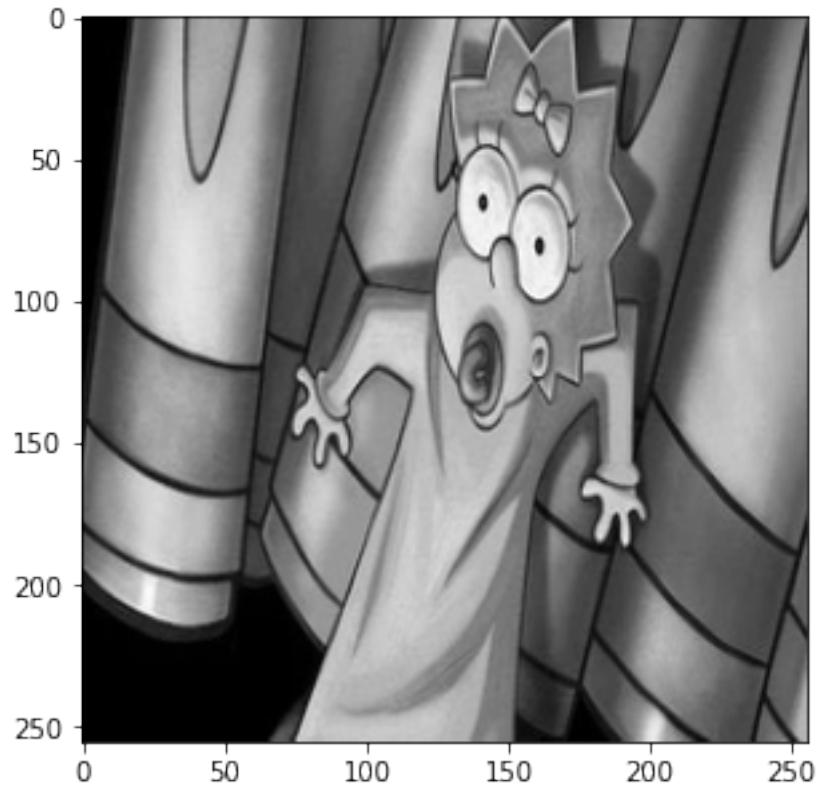
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795990>



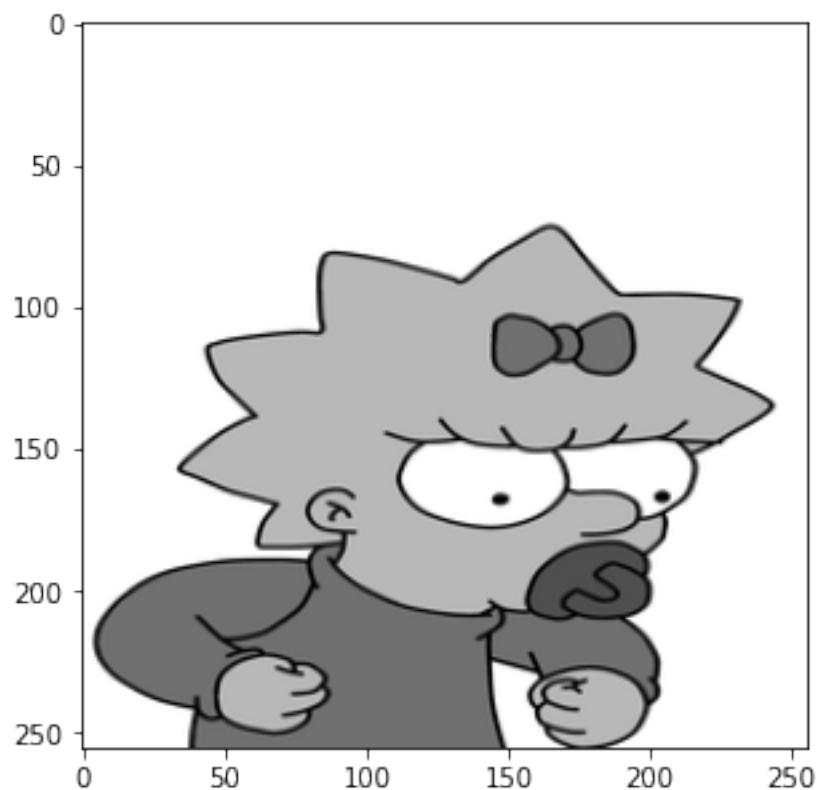
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1197959D0>



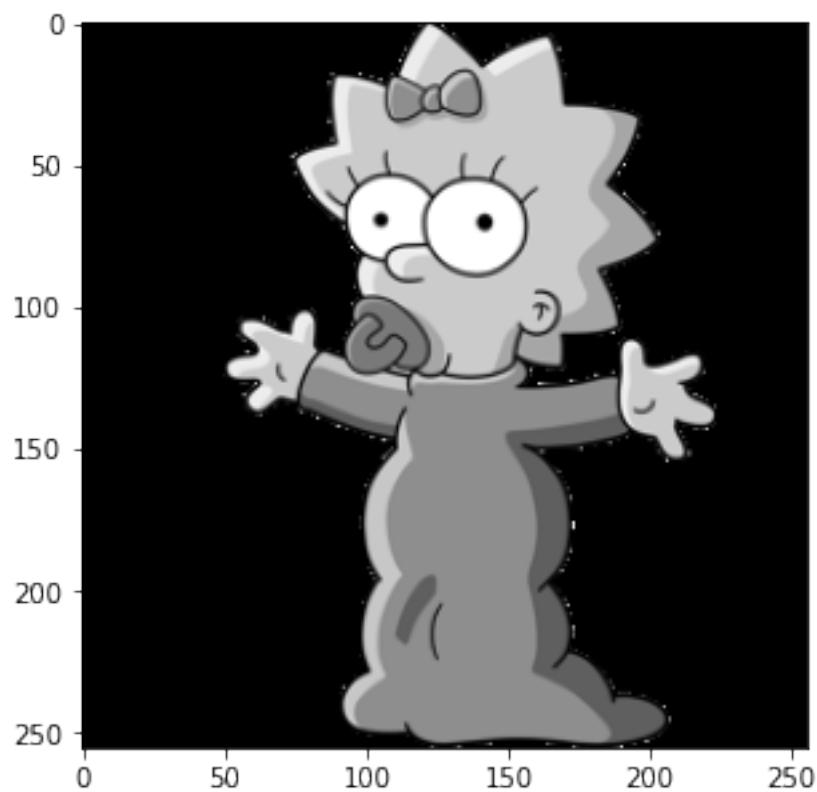
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795A10>



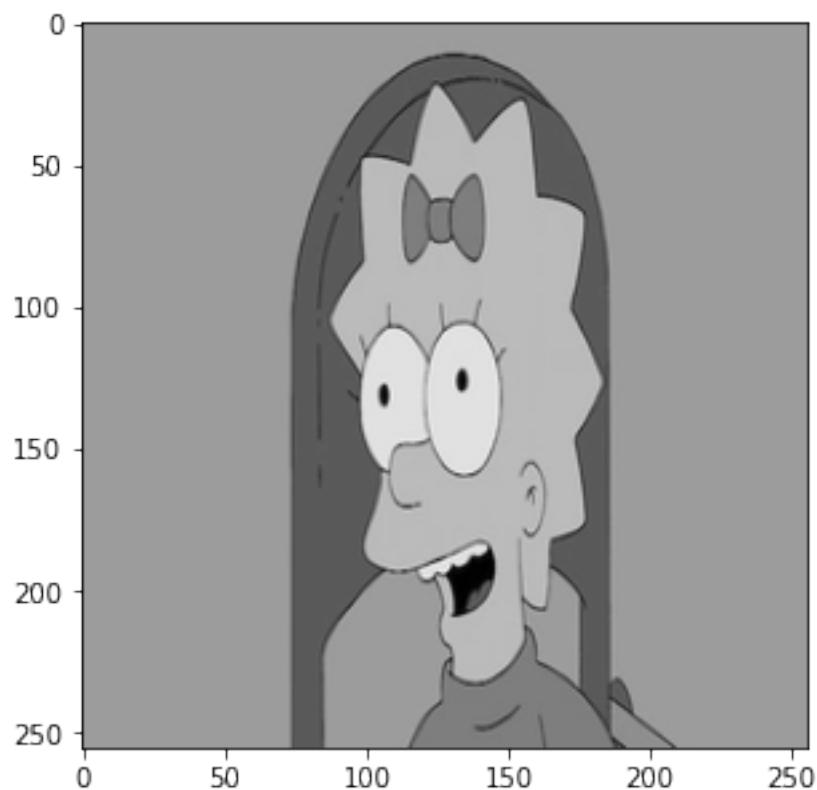
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795A50>



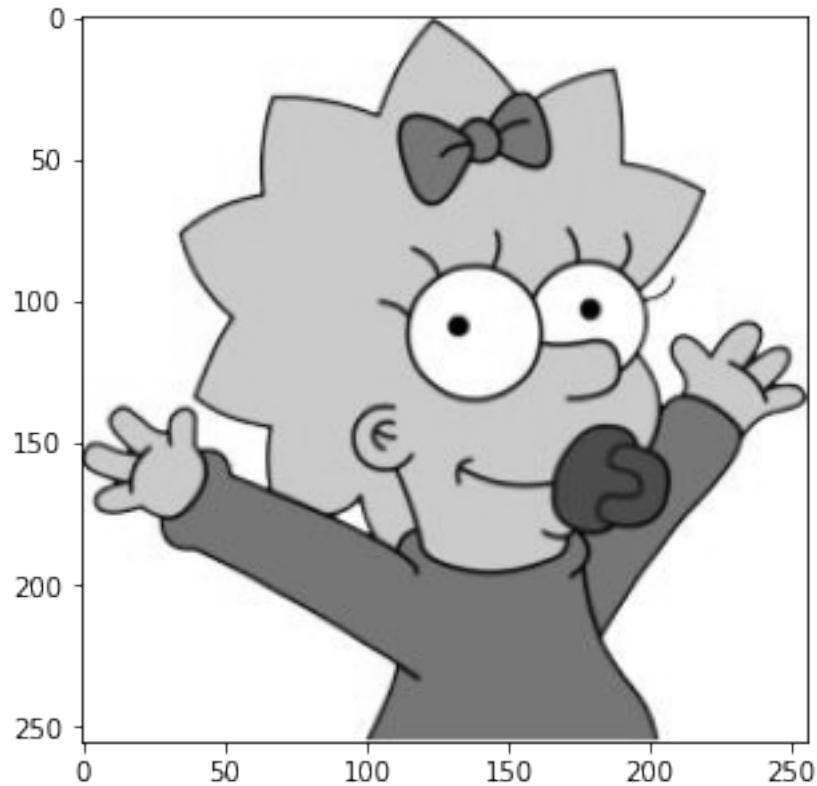
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795A90>



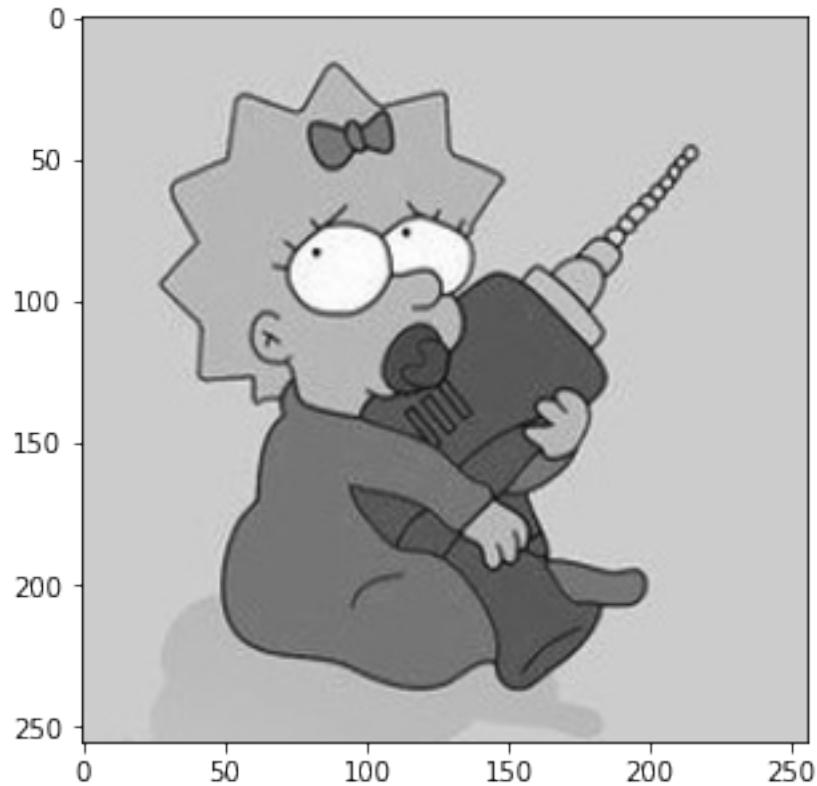
```
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795AD0>
```



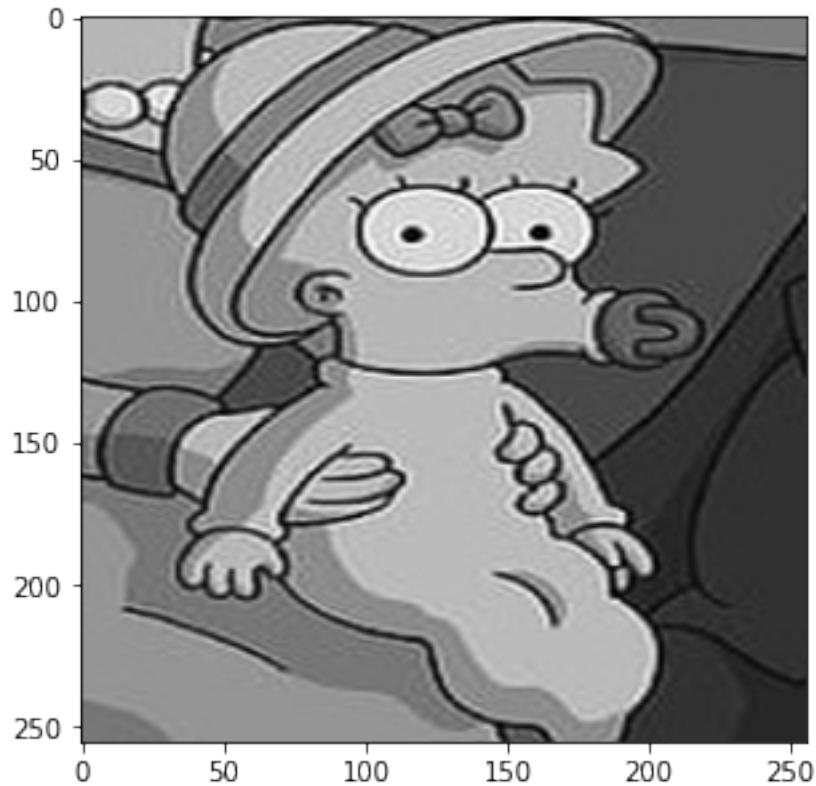
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795B10>



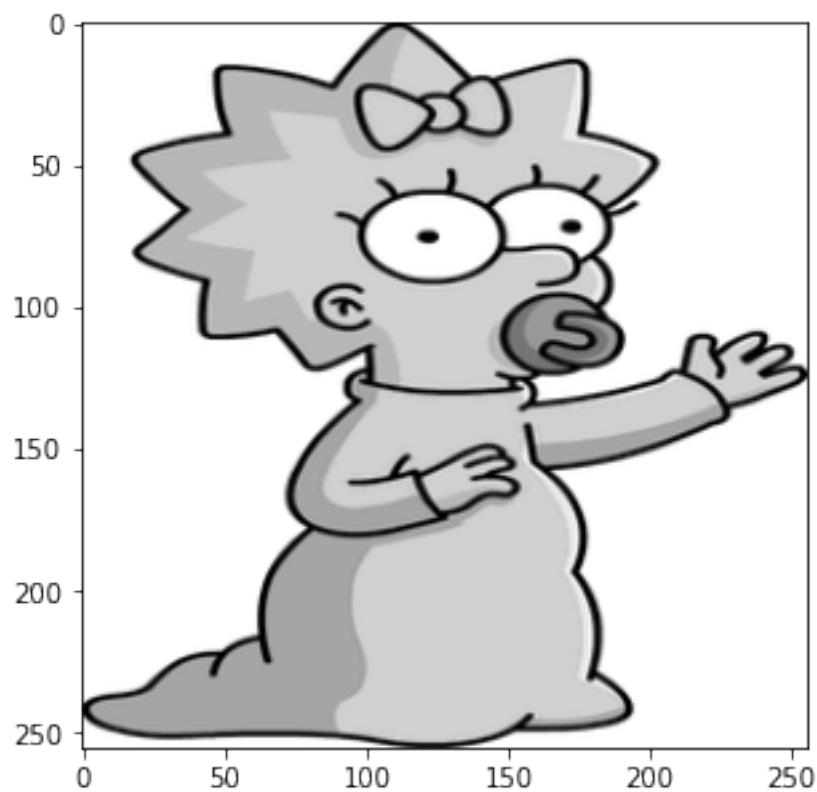
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795B50>



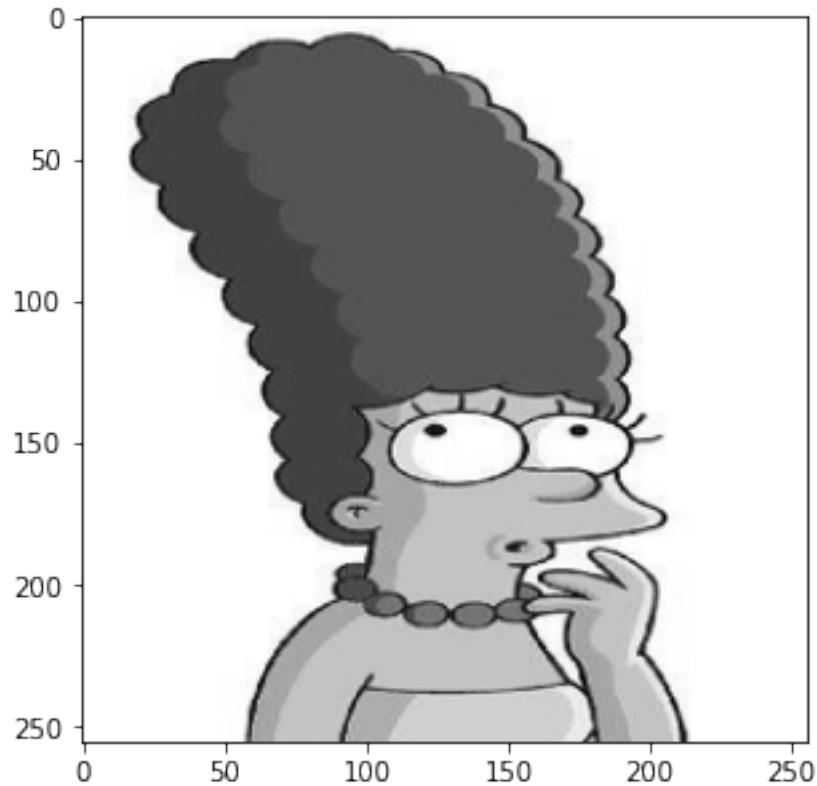
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795B90>



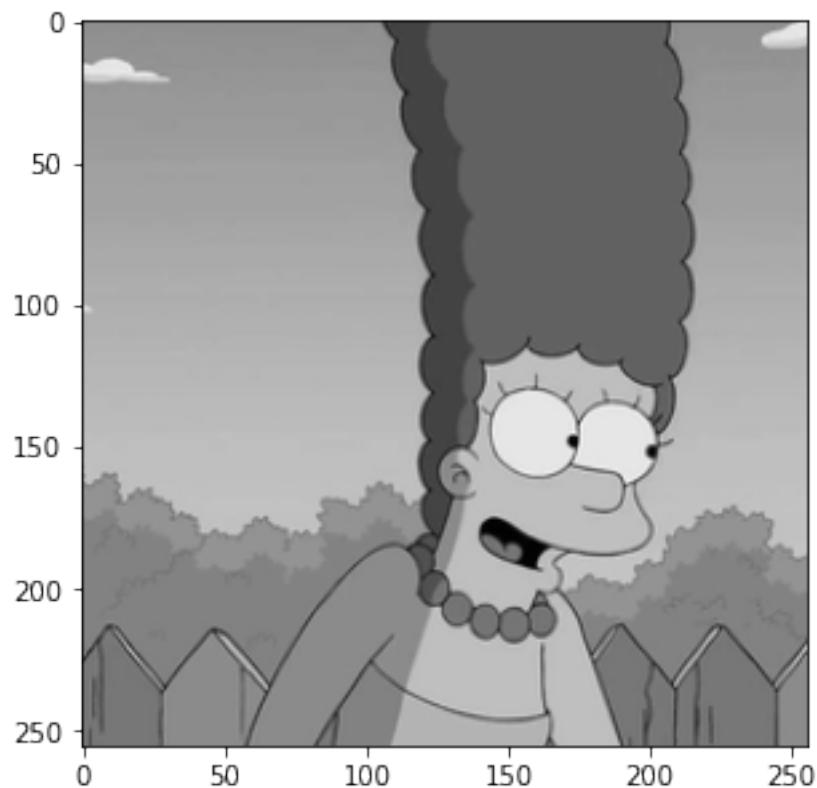
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795BD0>



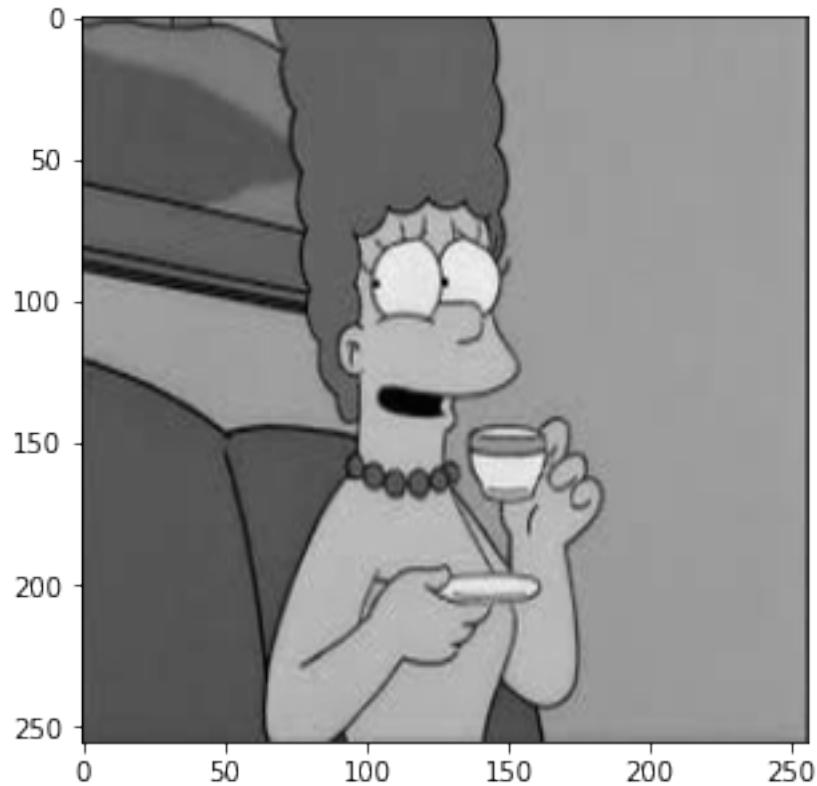
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795C10>



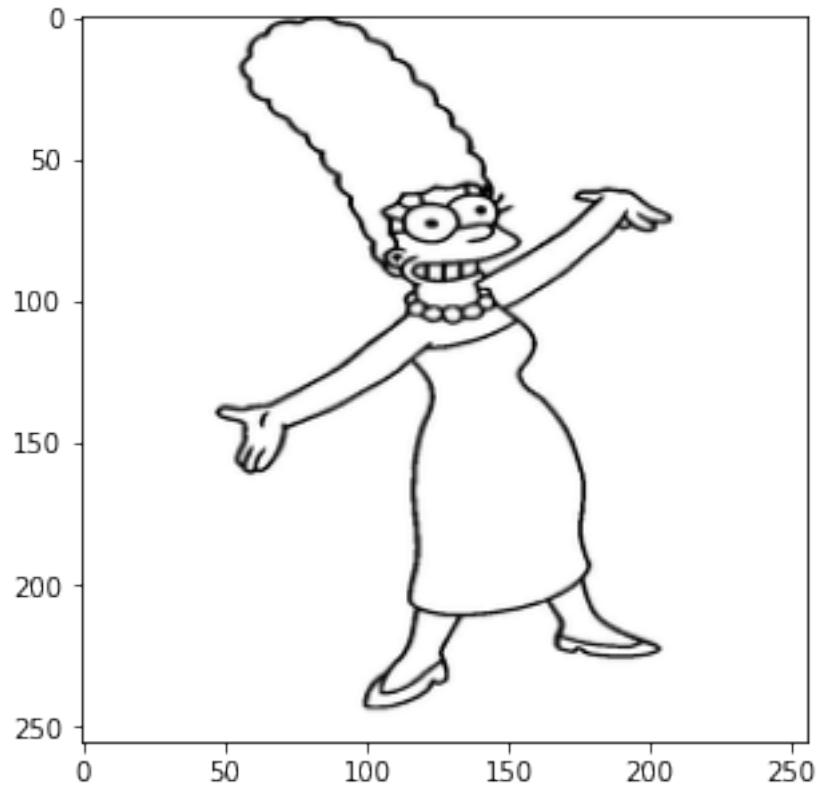
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795C50>



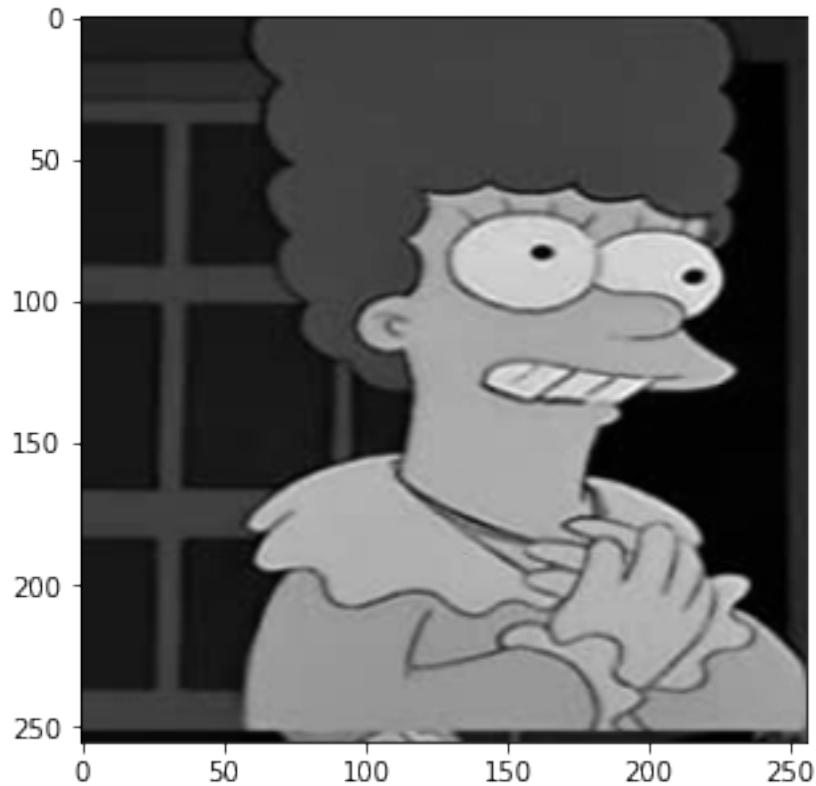
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795C90>



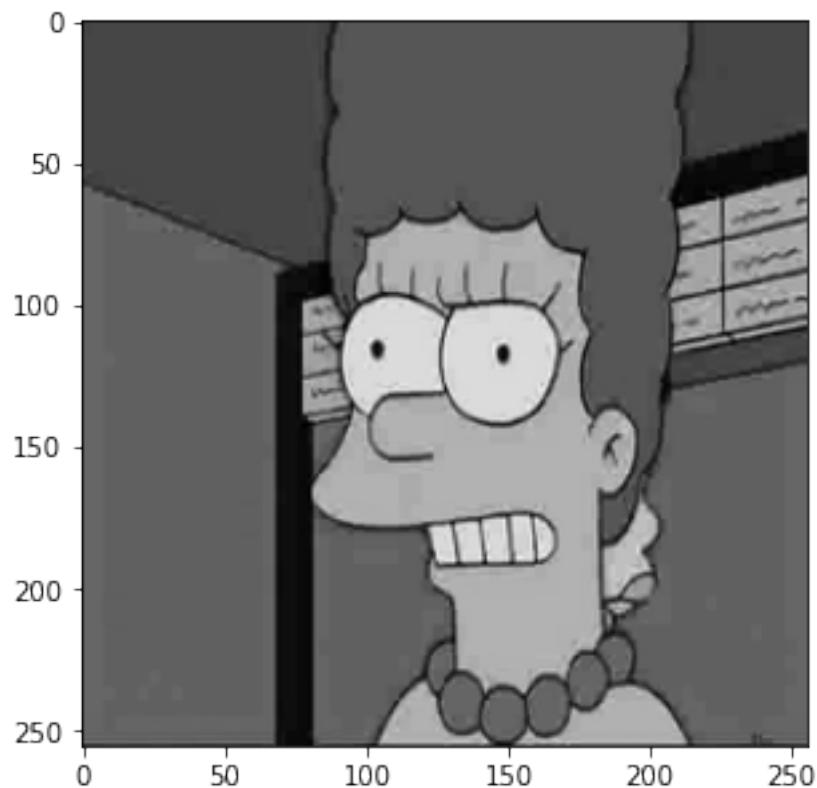
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795CD0>



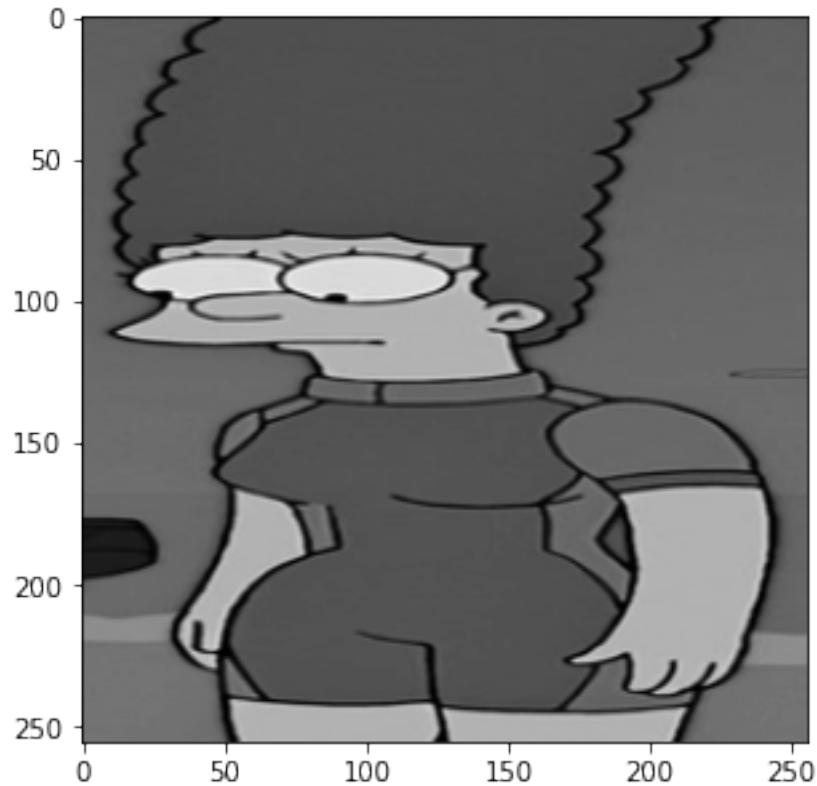
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795D10>



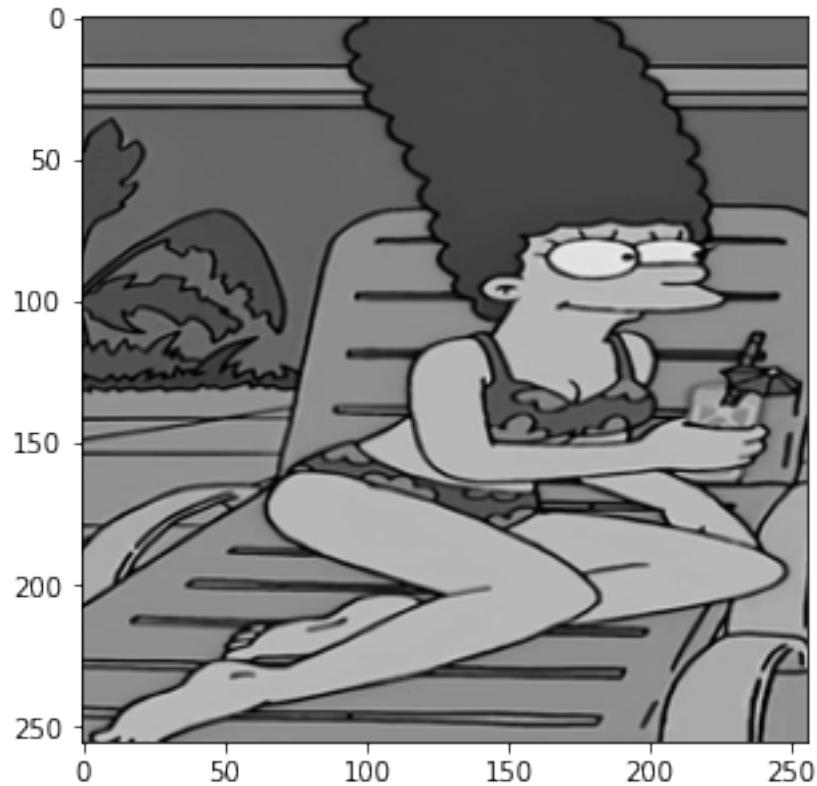
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795D90>



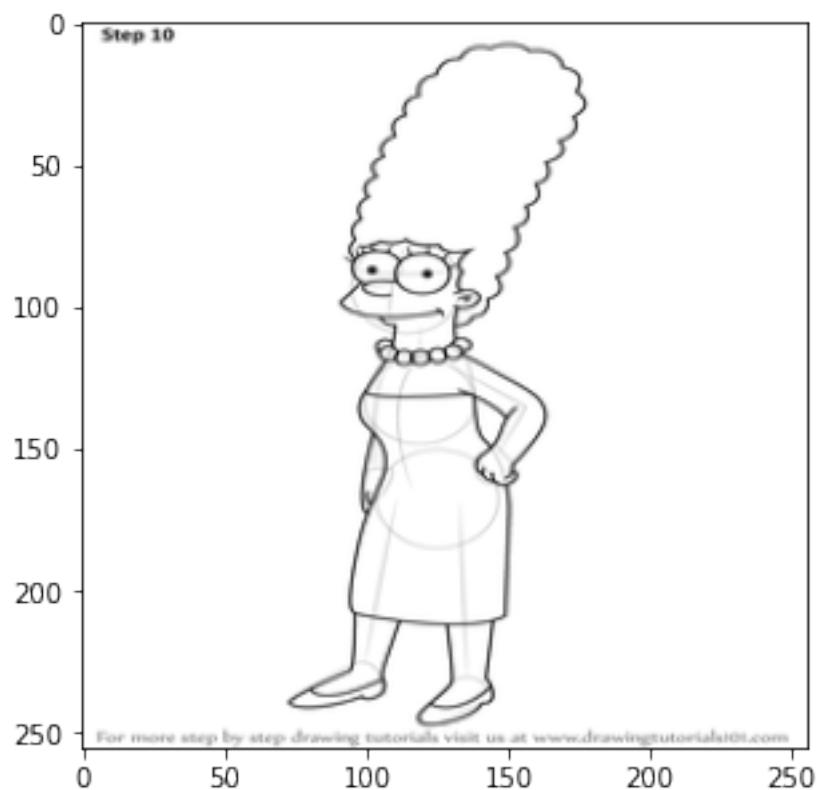
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795DD0>



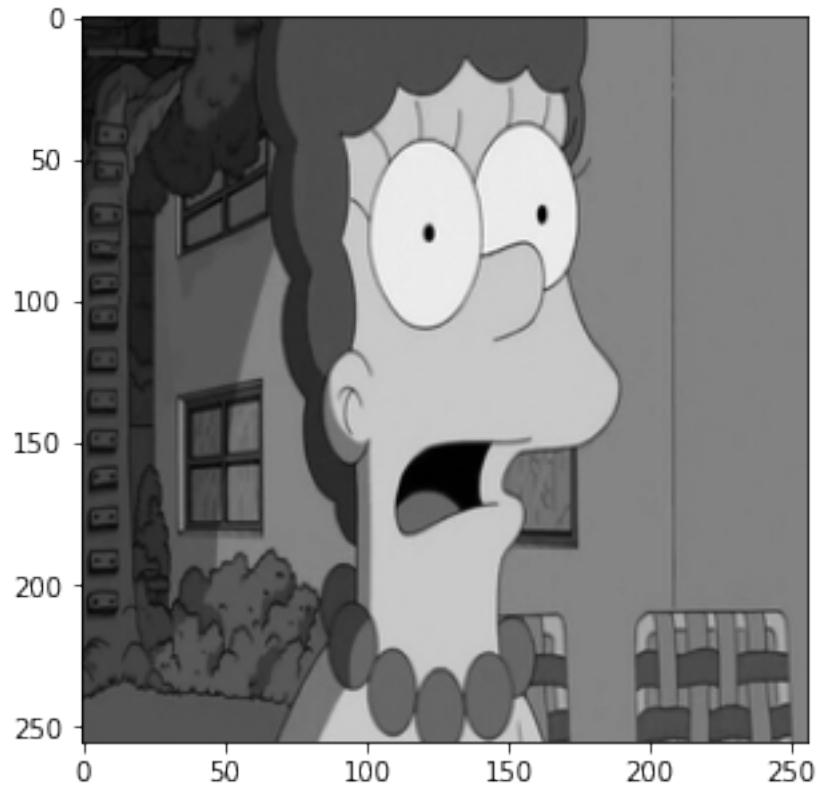
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795E10>



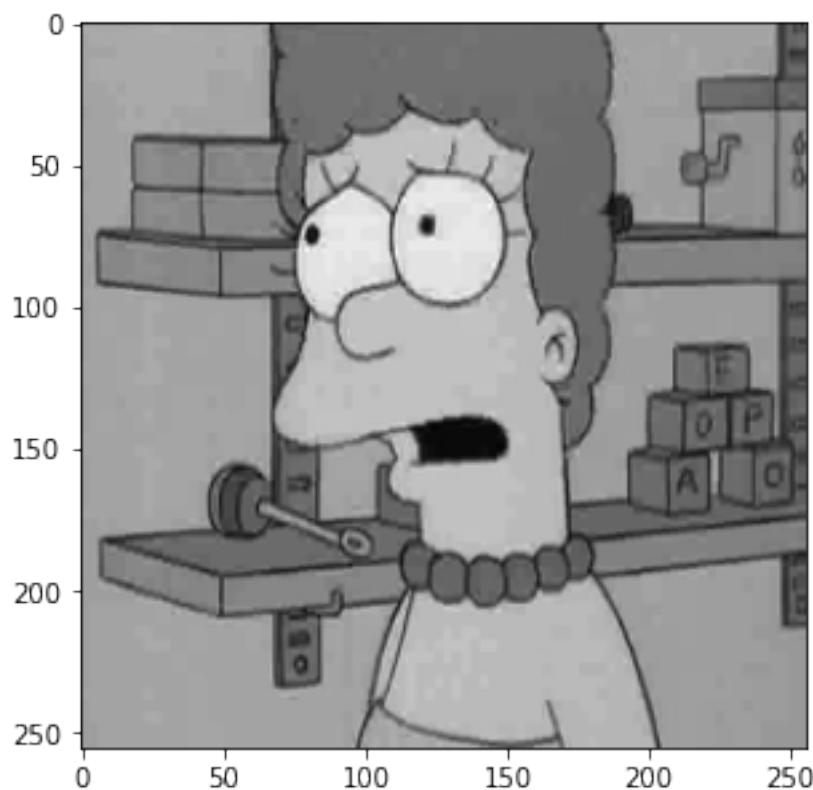
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795E50>



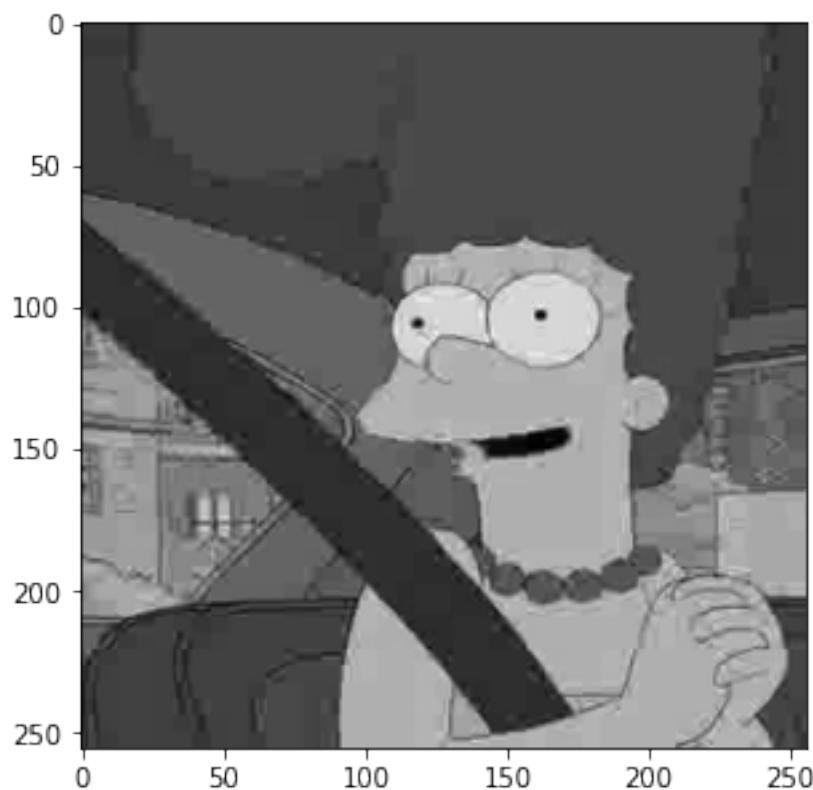
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795D50>



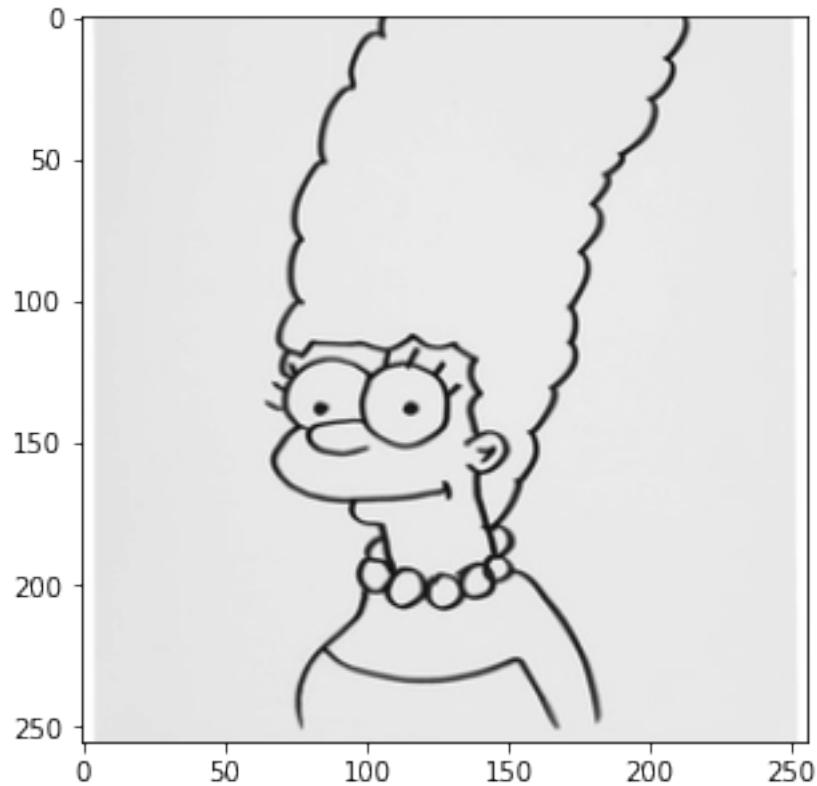
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795E90>



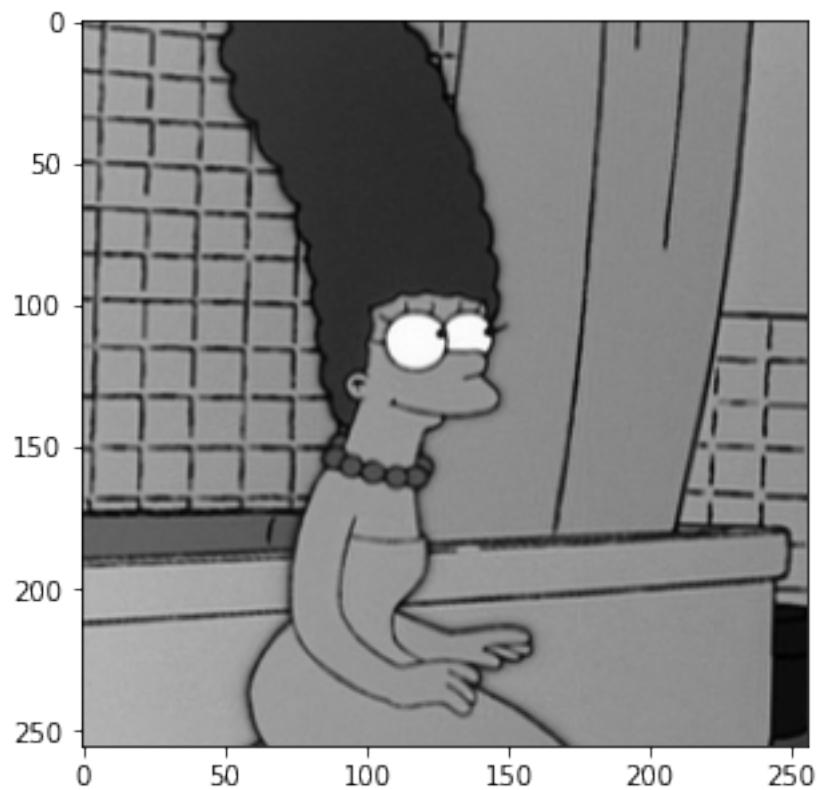
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795ED0>



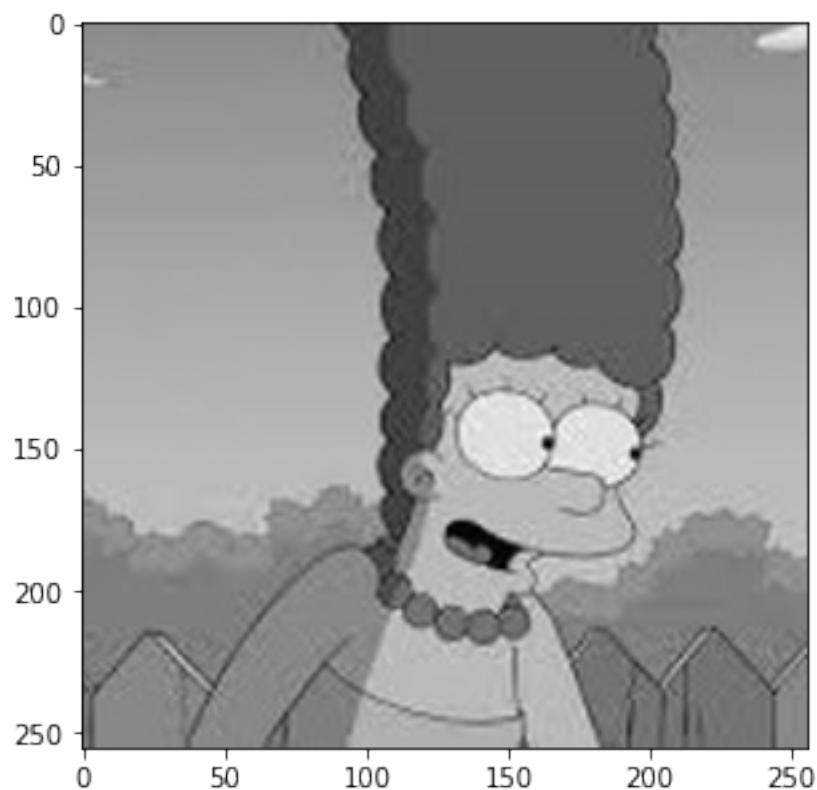
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795F10>



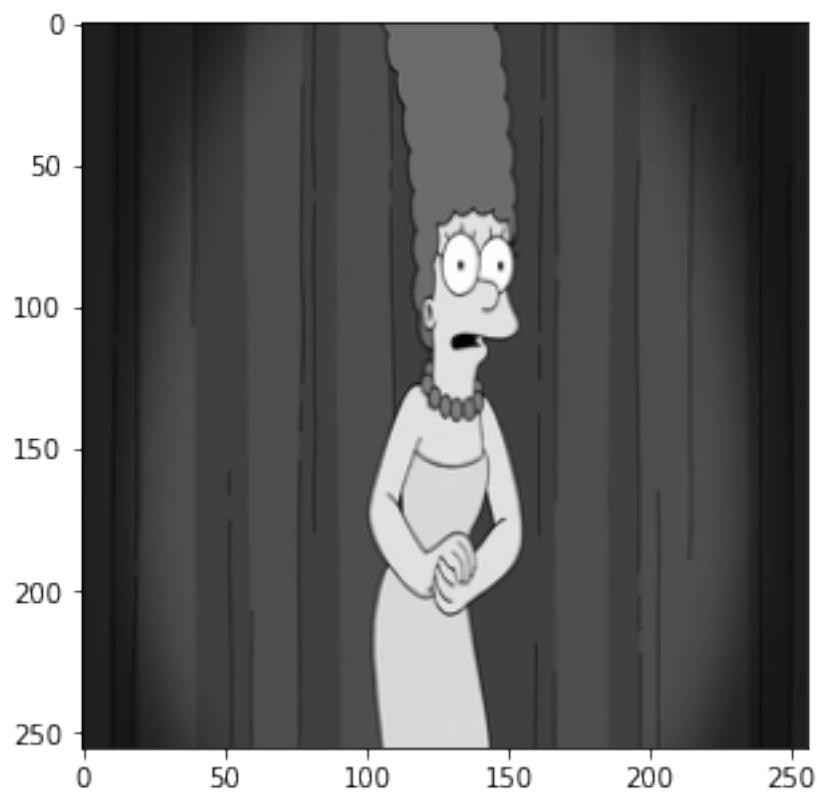
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795F50>



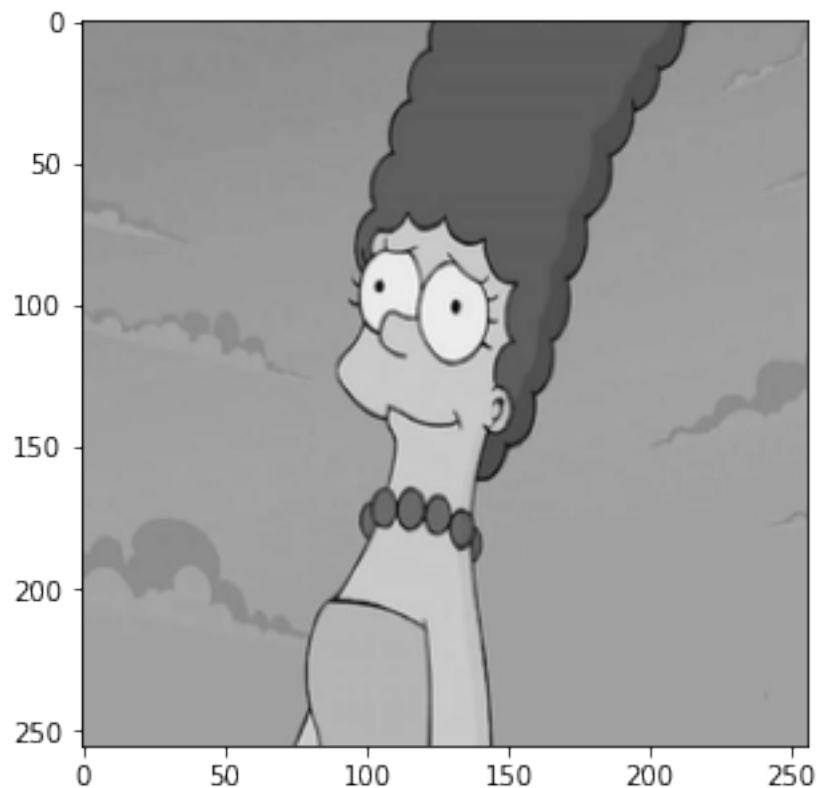
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795F90>



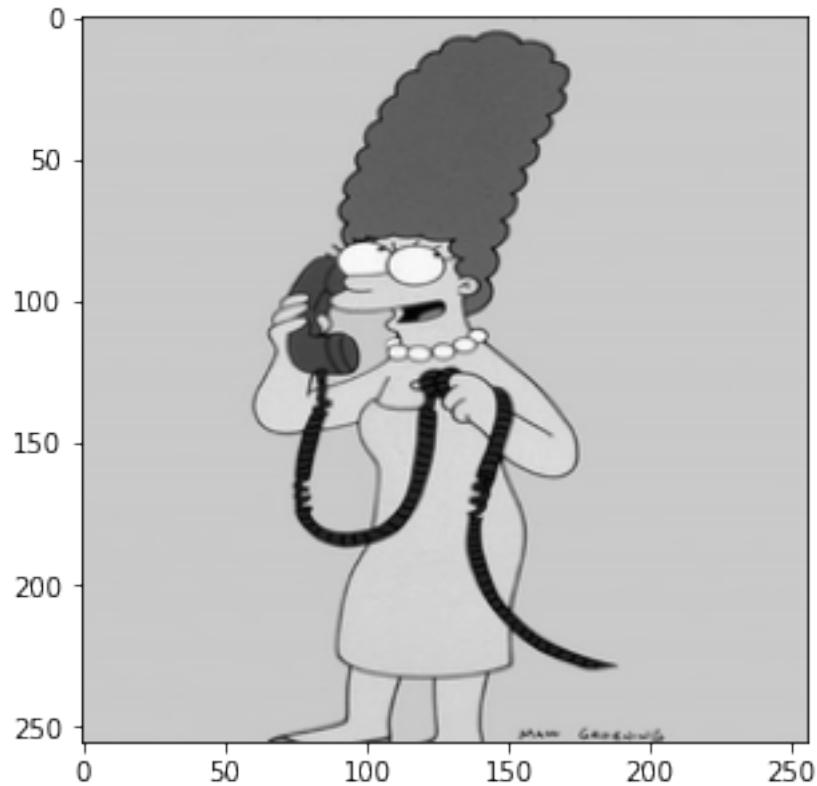
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119795FD0>



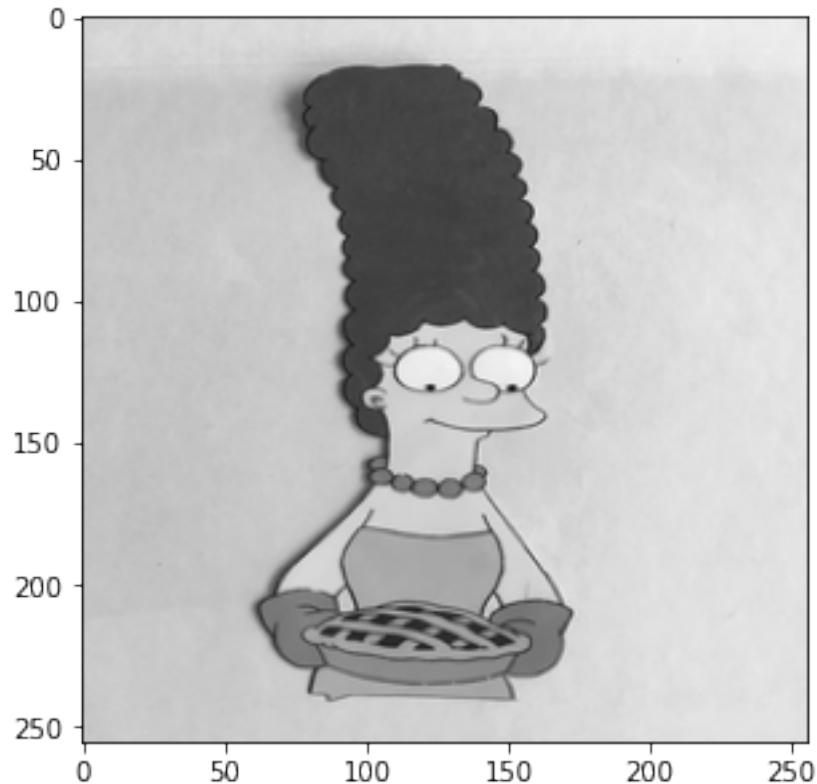
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322050>



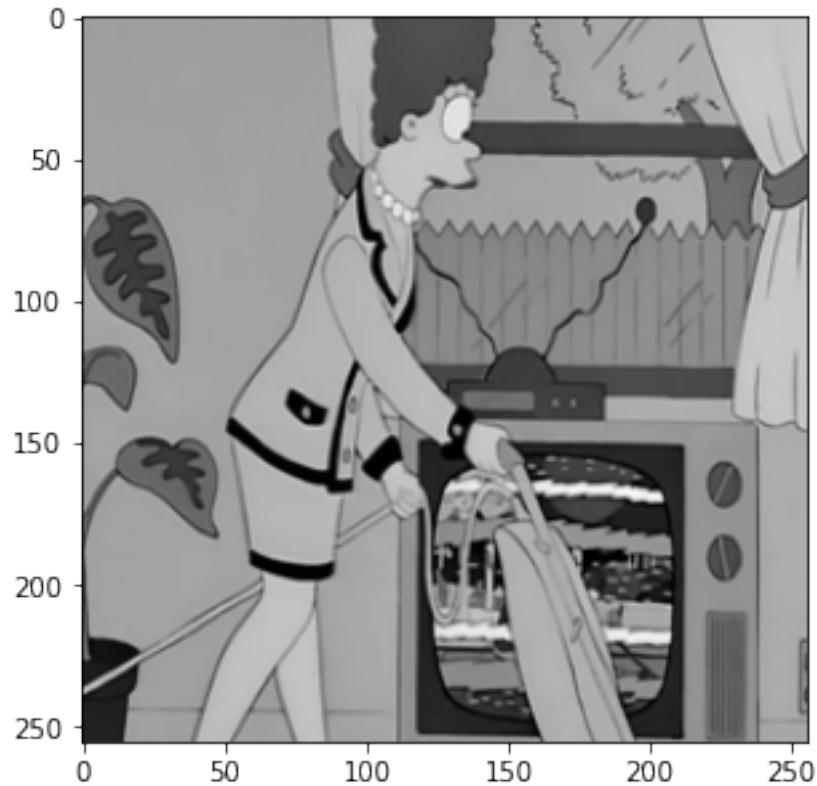
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322090>



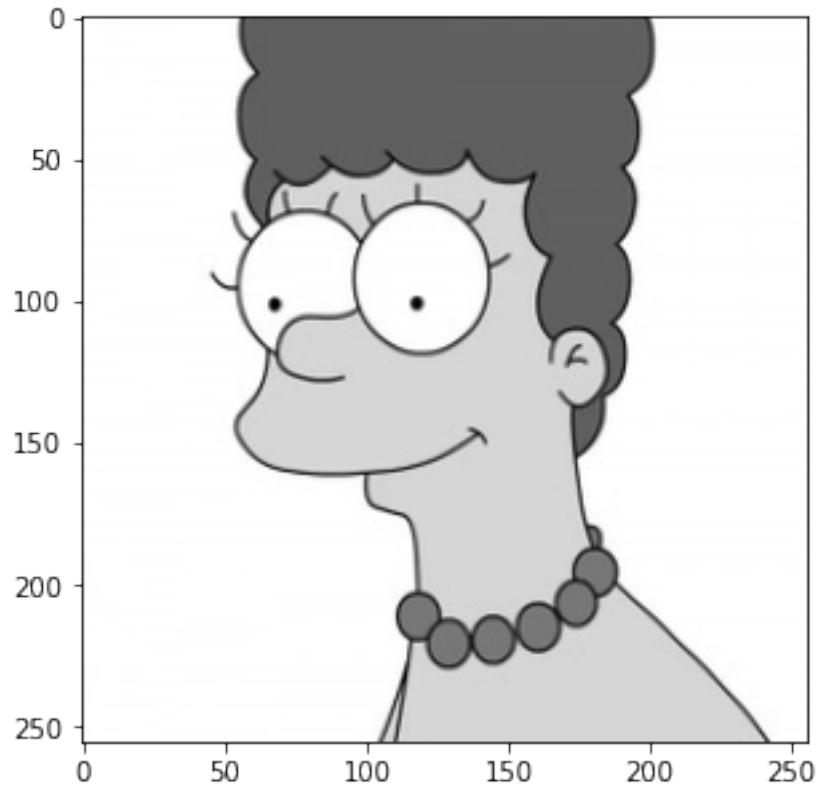
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1193220D0>



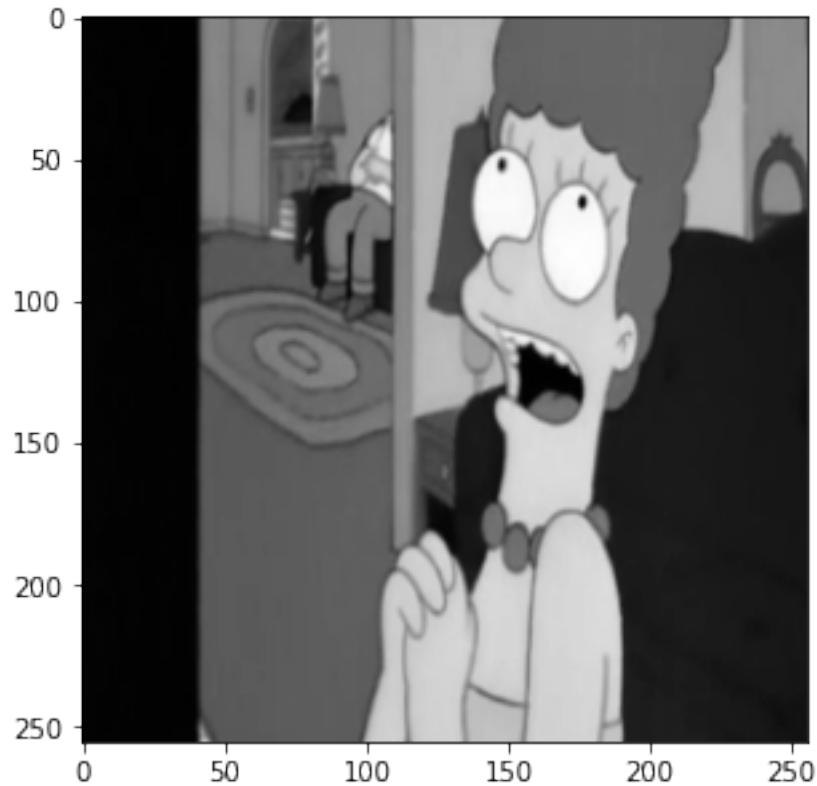
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322110>



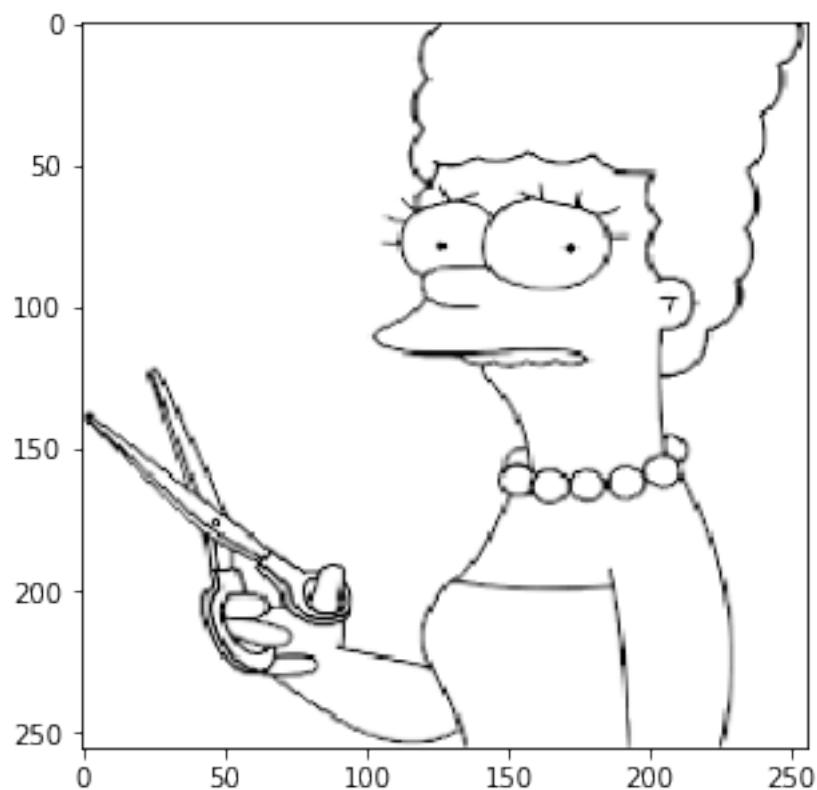
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322150>



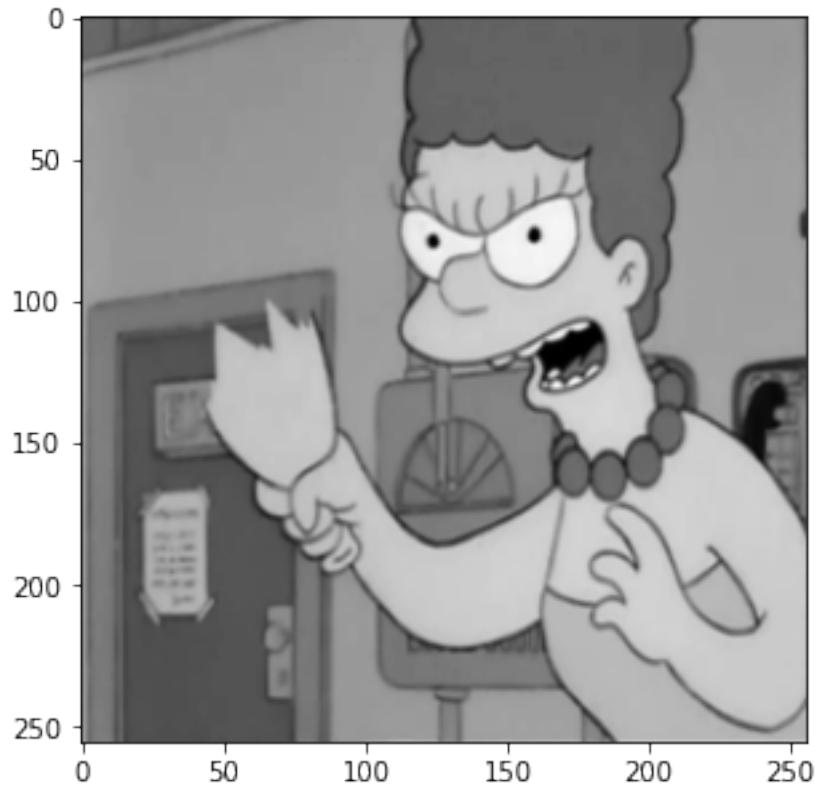
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322190>



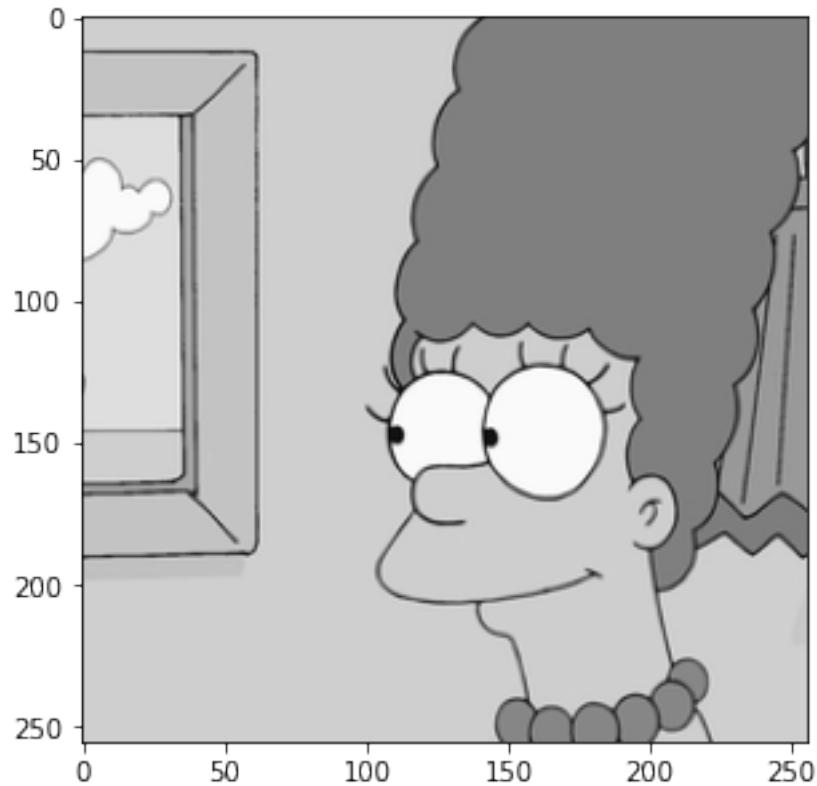
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1193221D0>



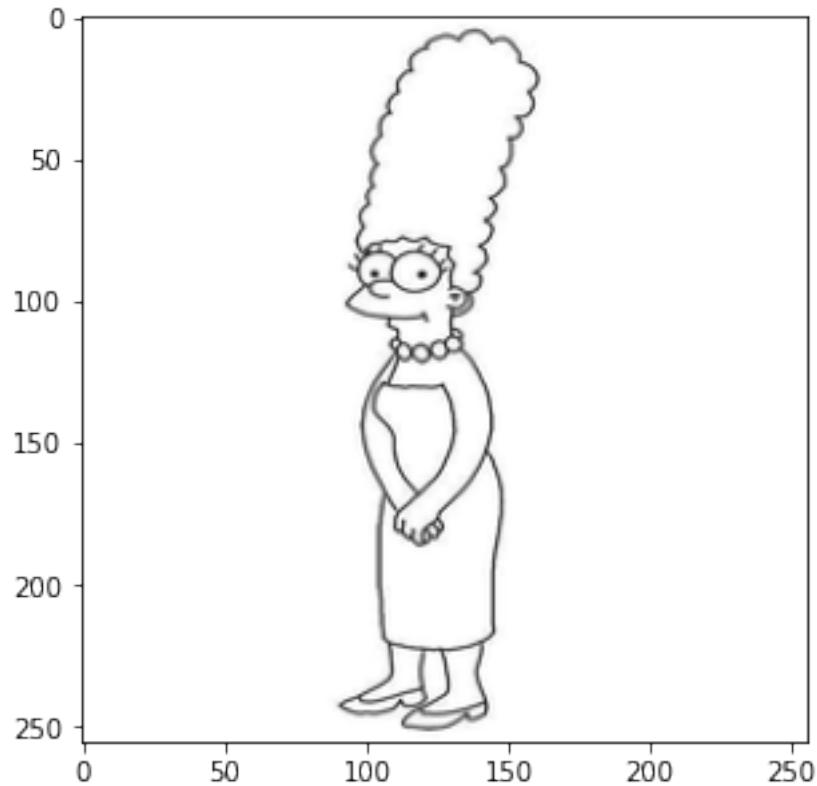
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322210>



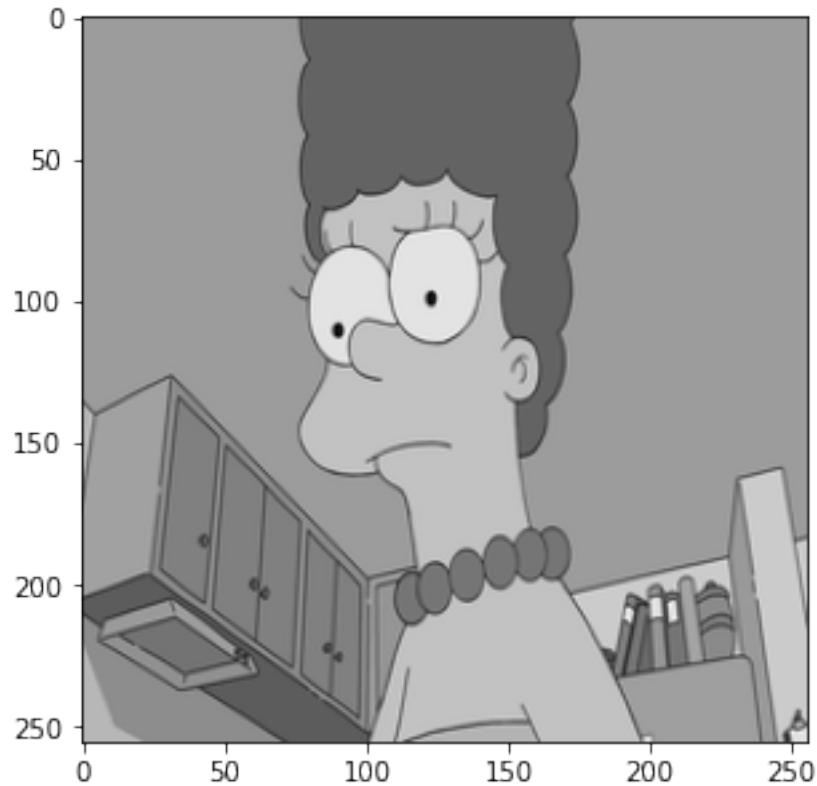
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322250>



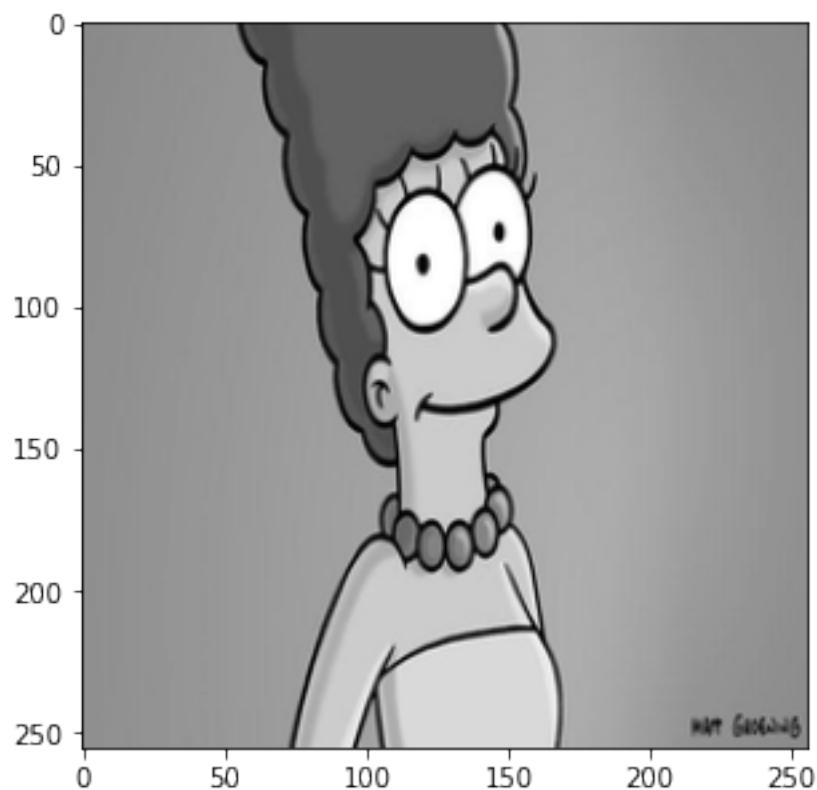
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322290>



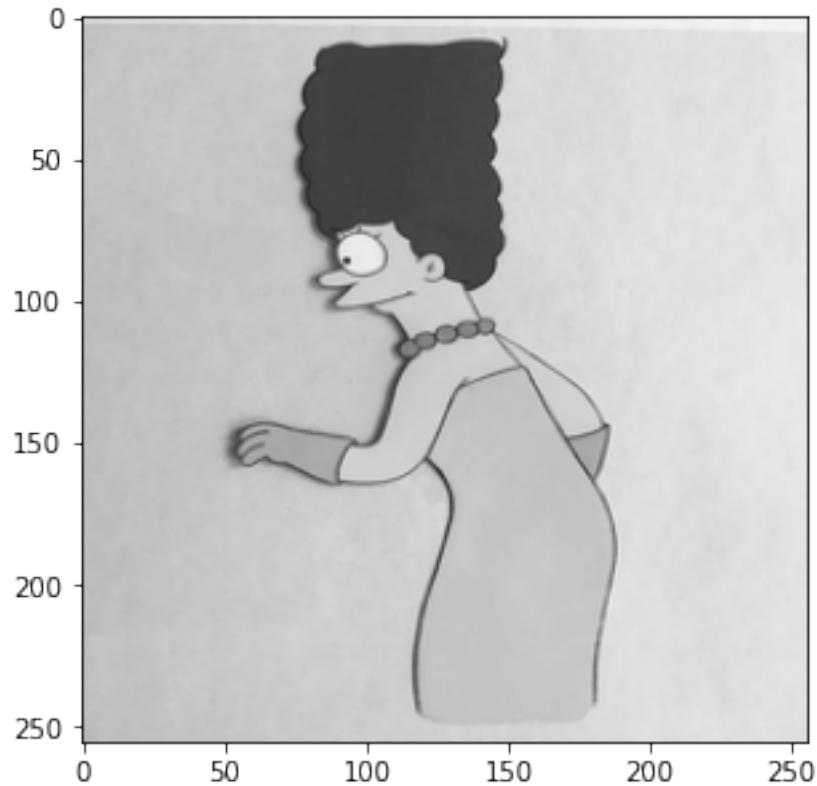
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1193222D0>



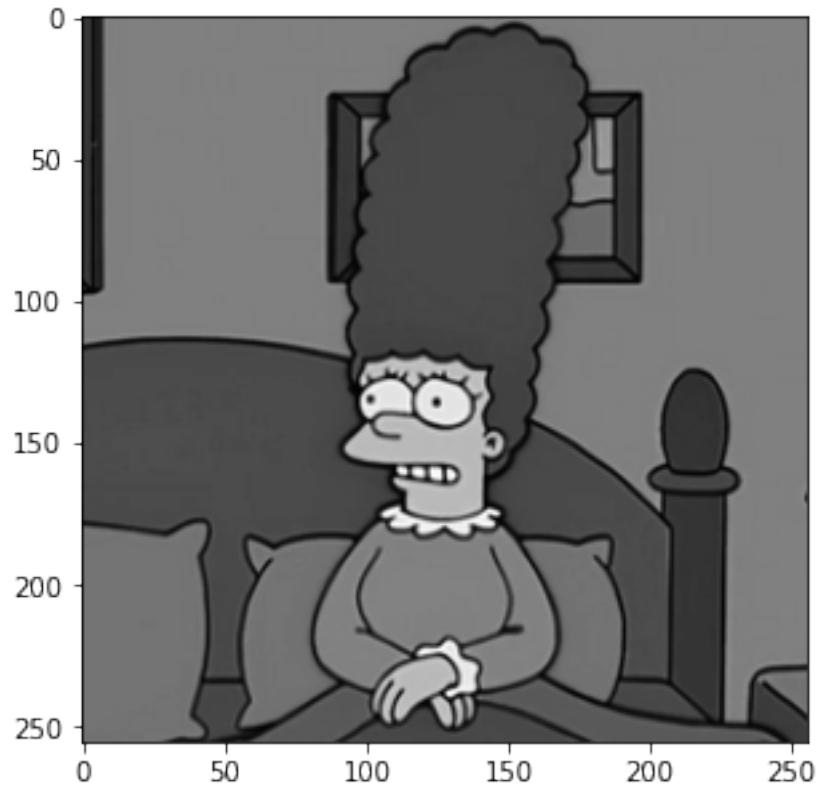
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322310>



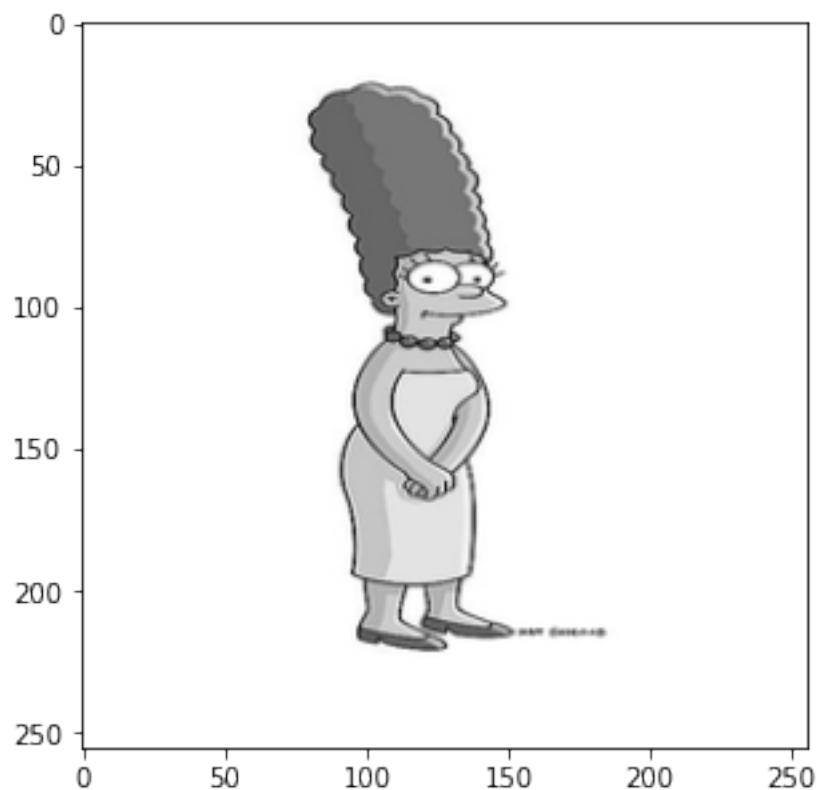
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322350>



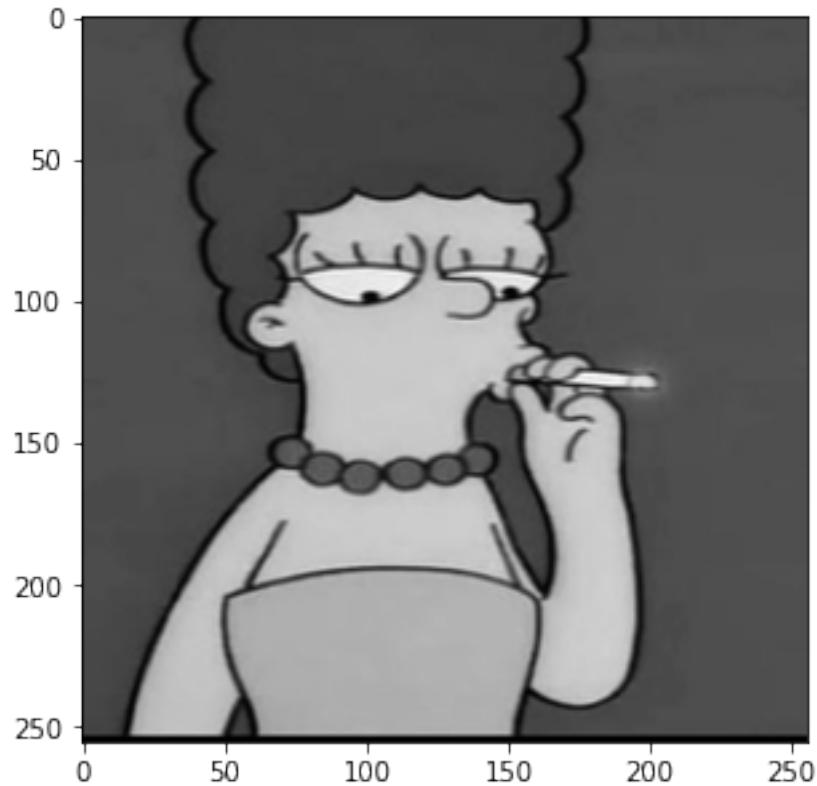
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322390>



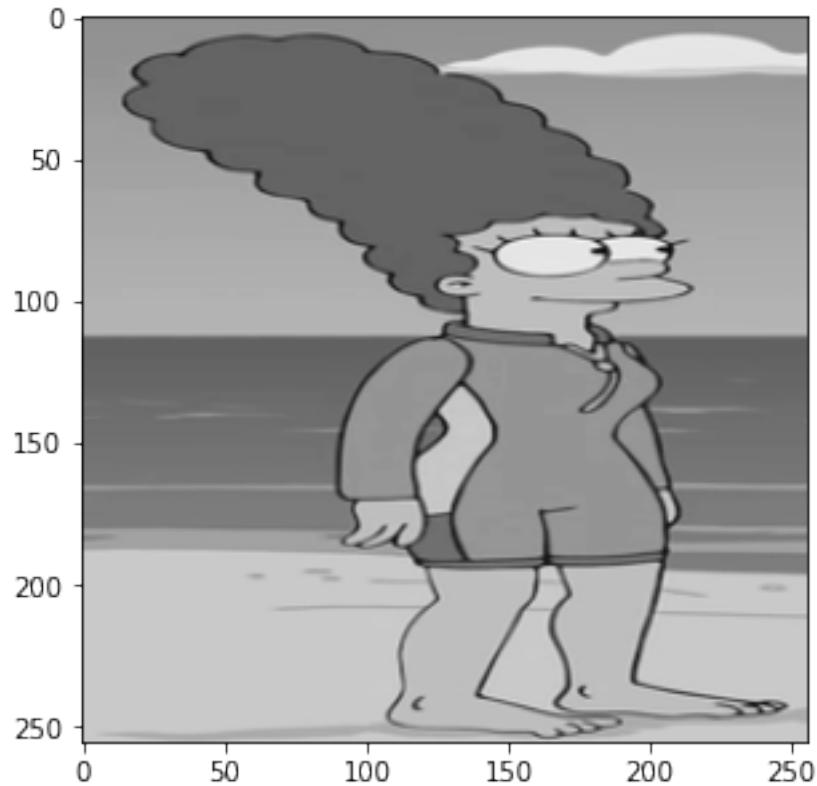
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1193223D0>



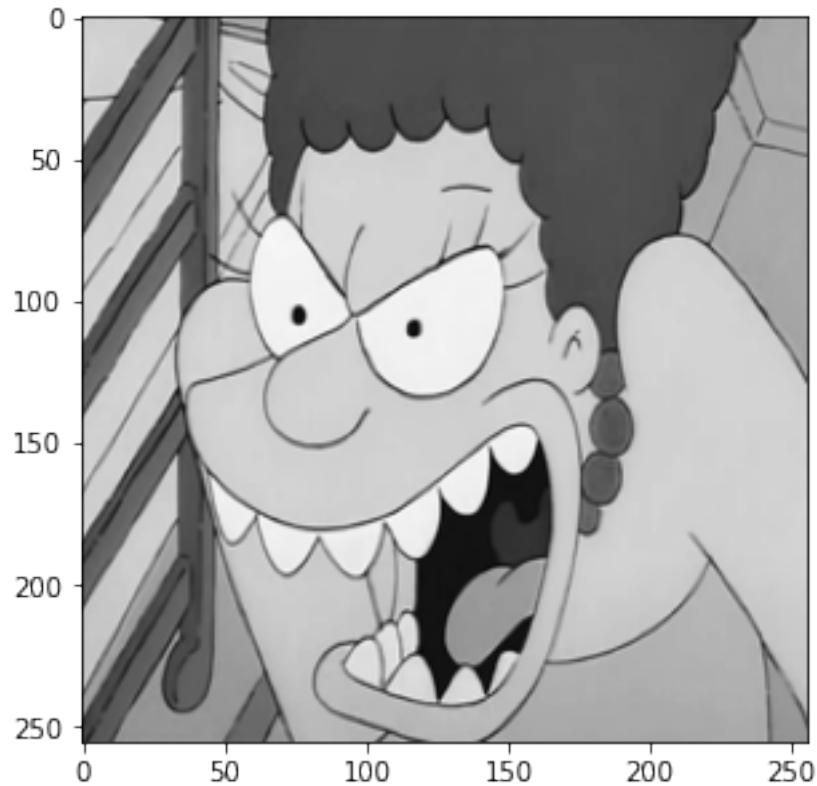
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322410>



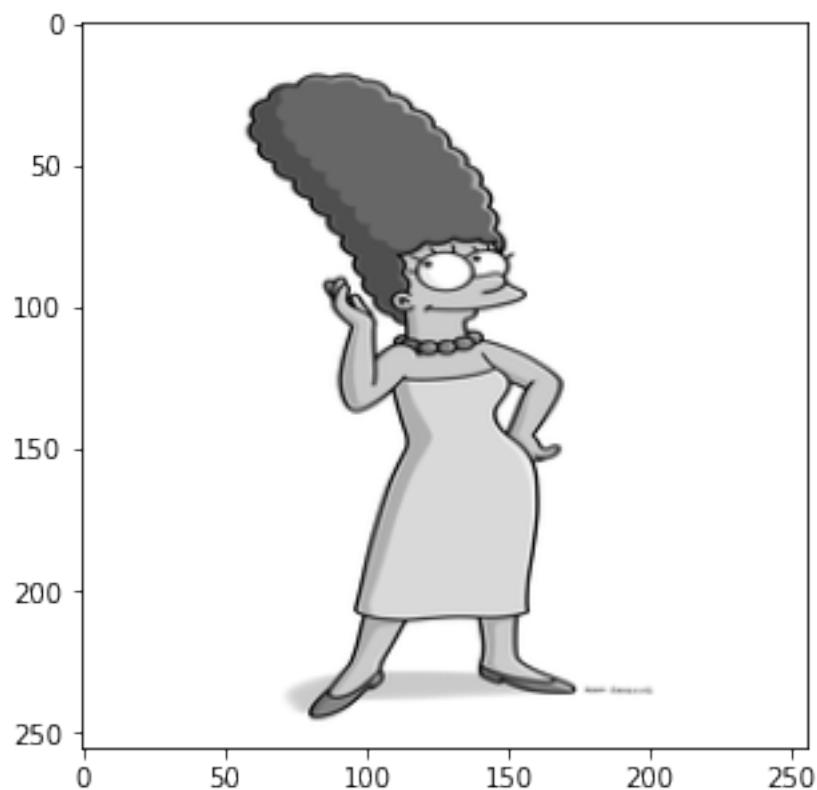
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322450>



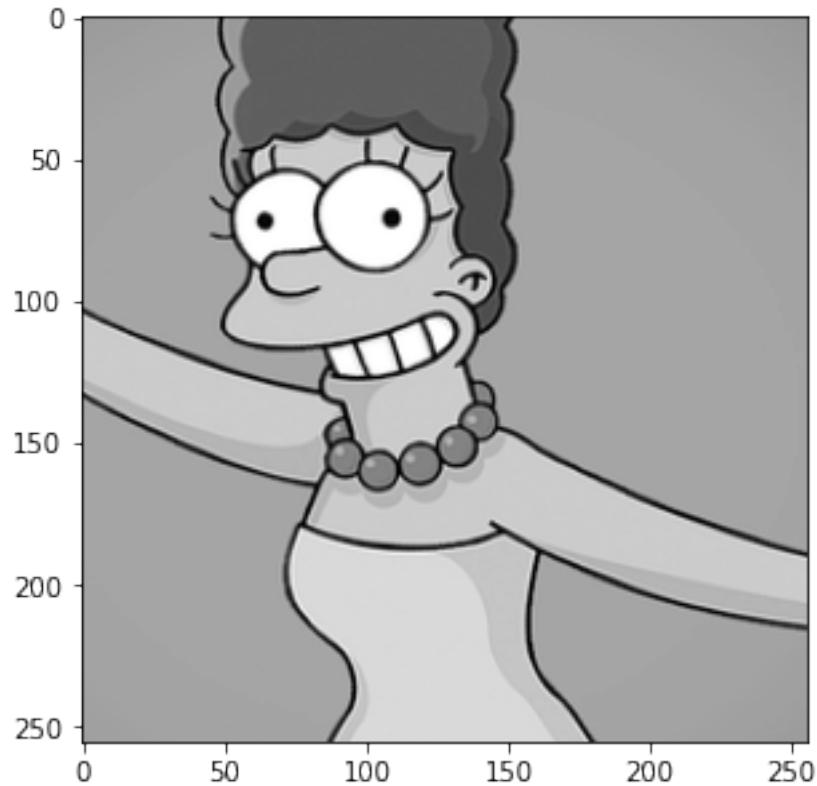
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322490>



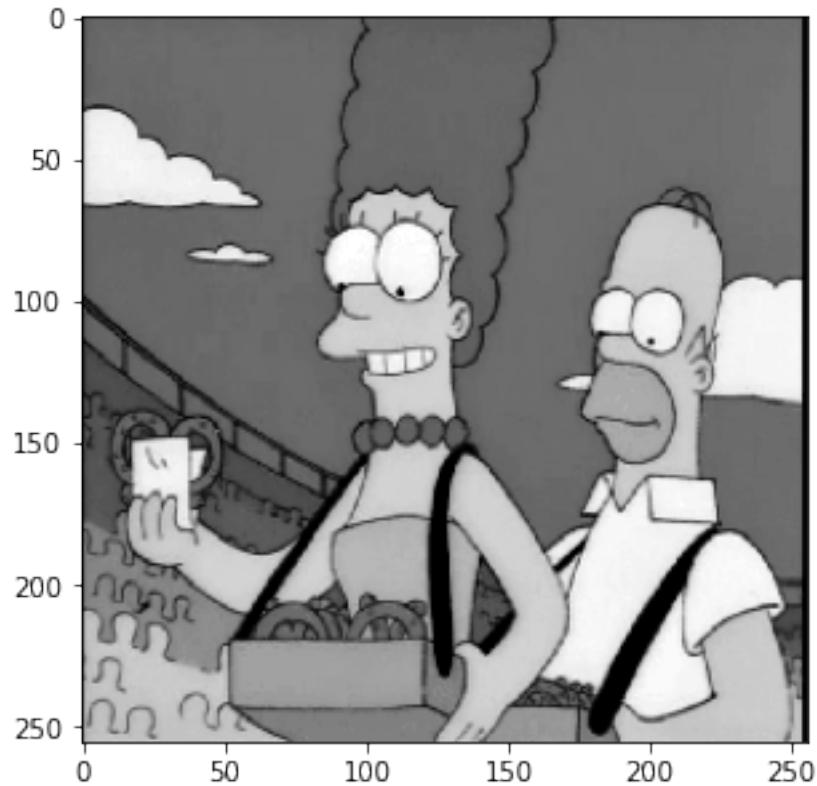
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1193224D0>



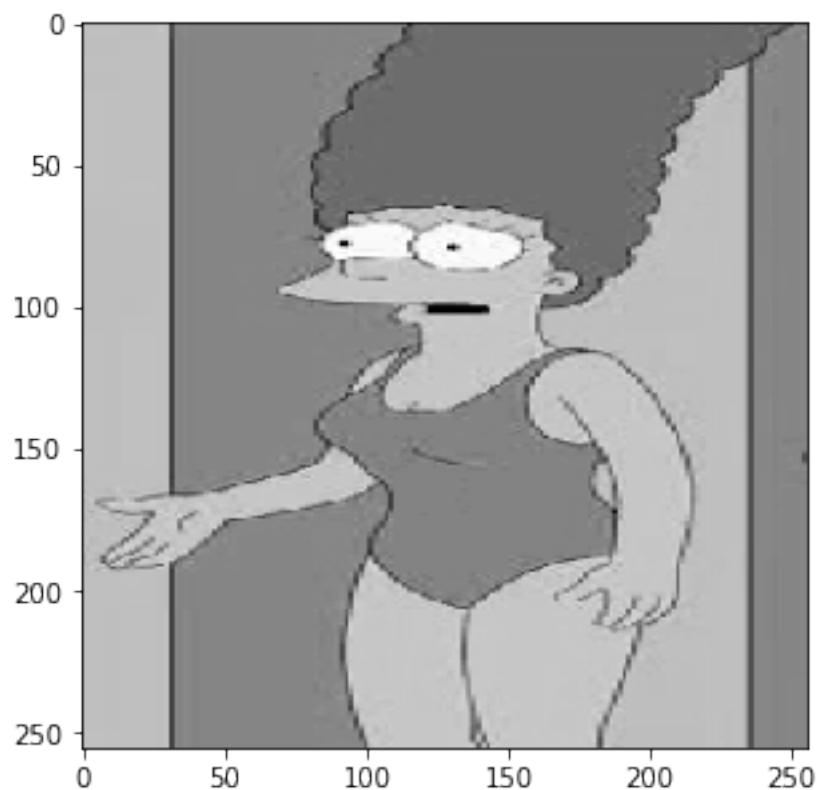
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322510>



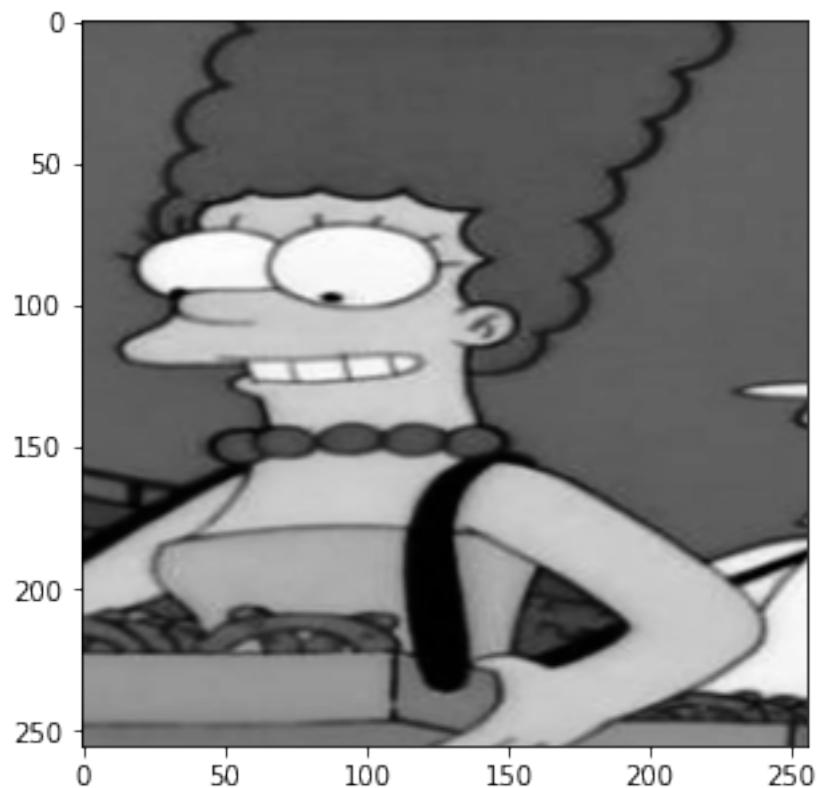
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322550>



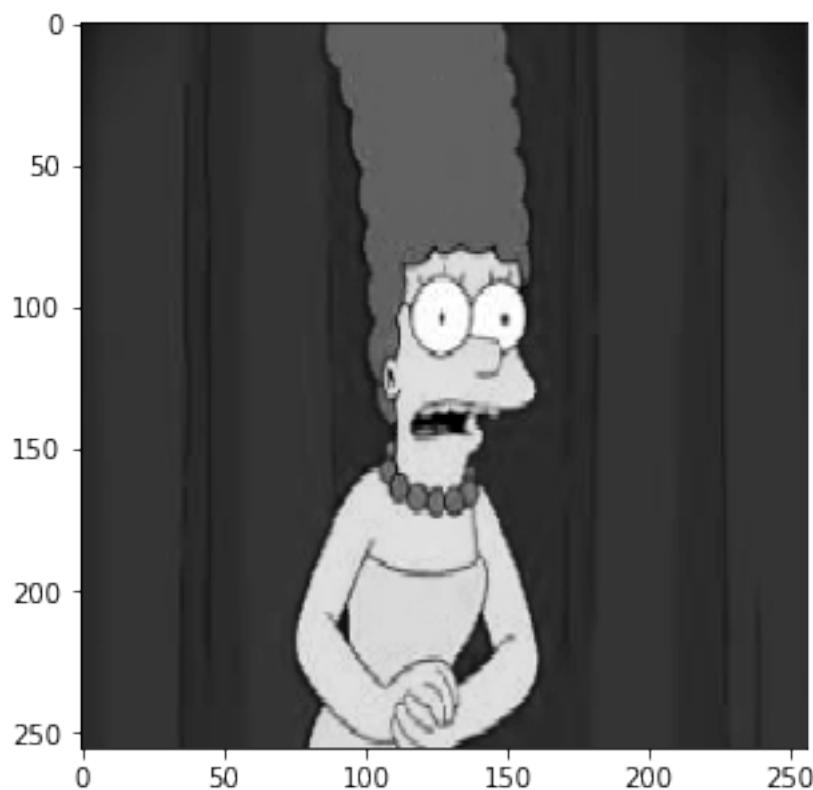
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322590>



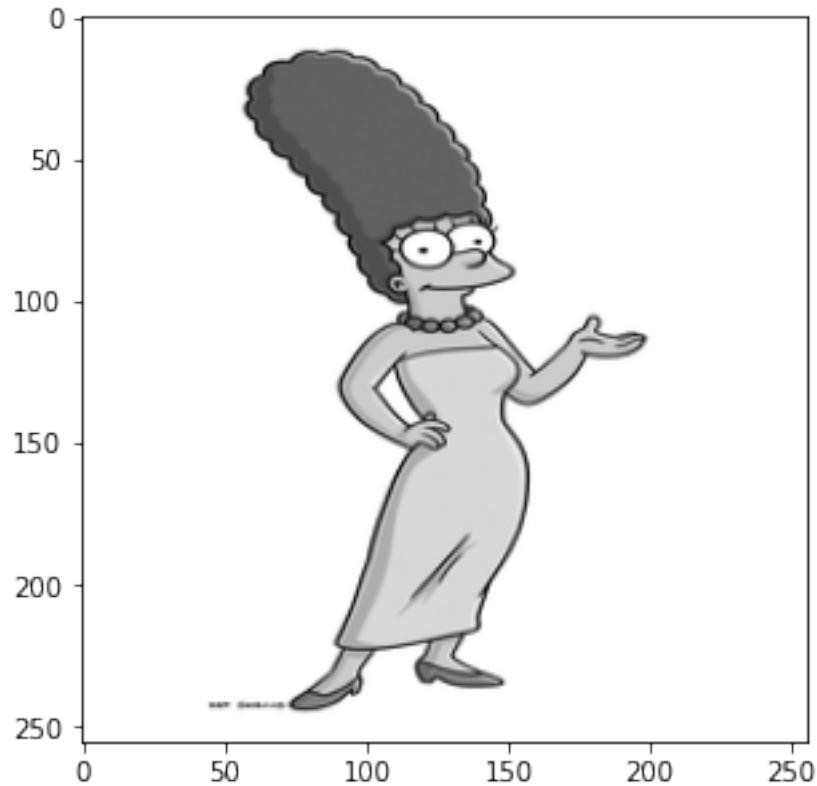
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1193225D0>



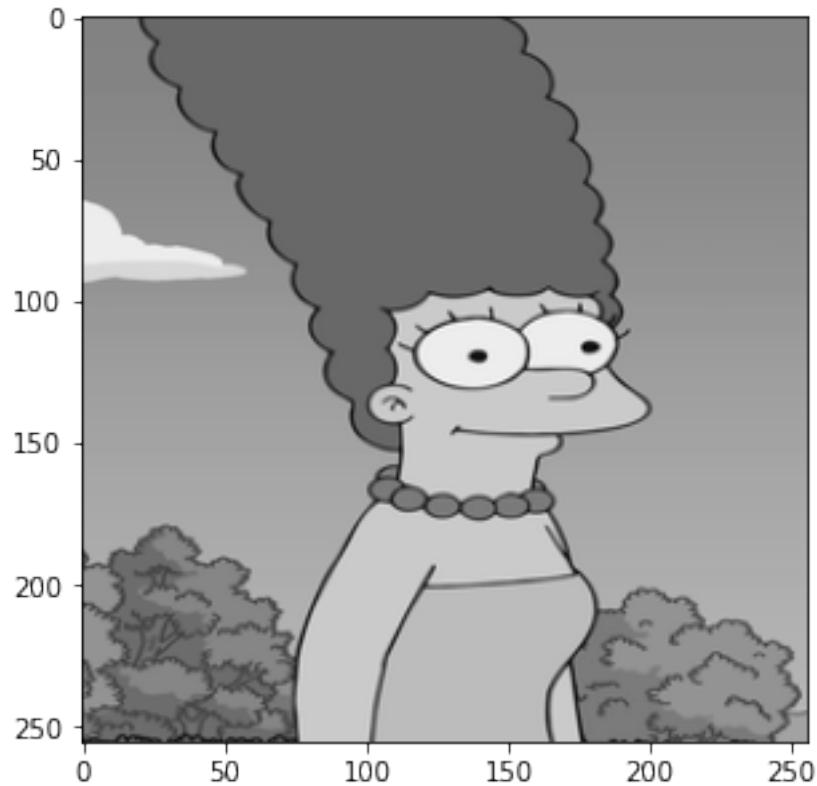
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322610>



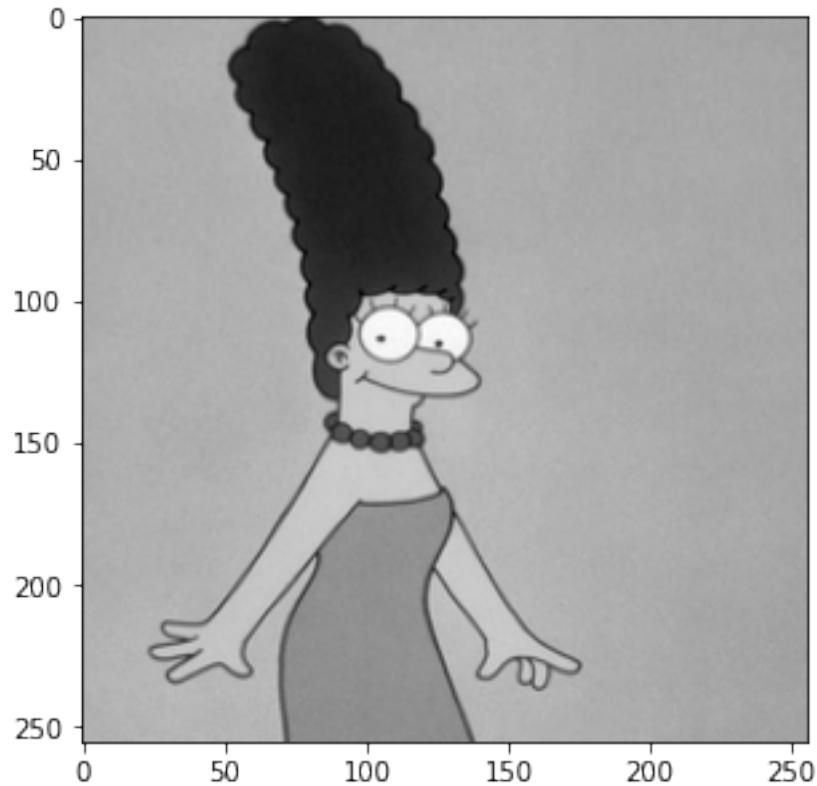
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322650>



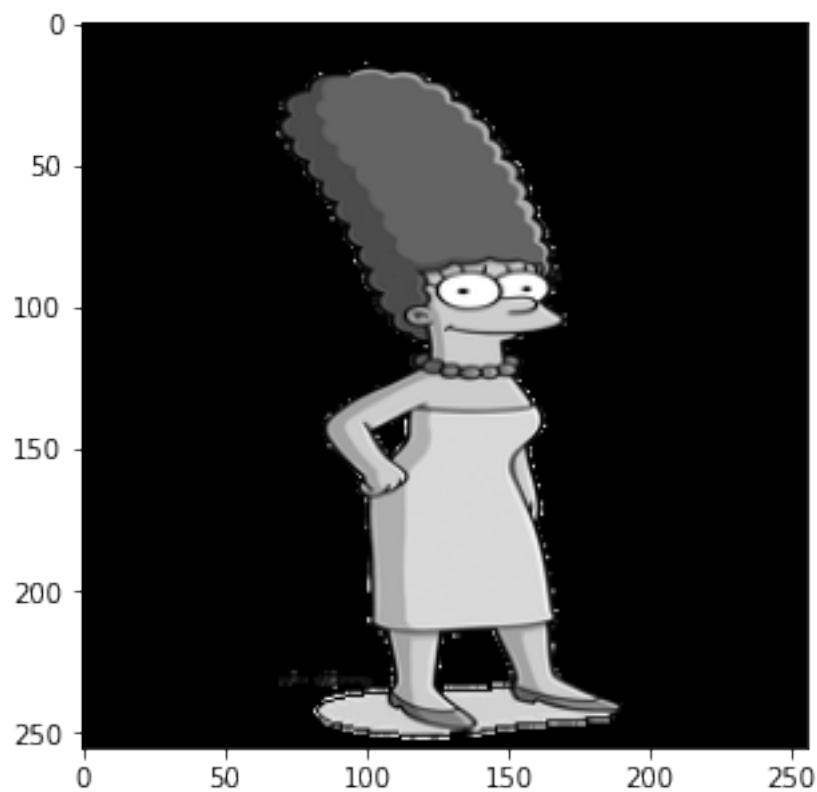
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322690>



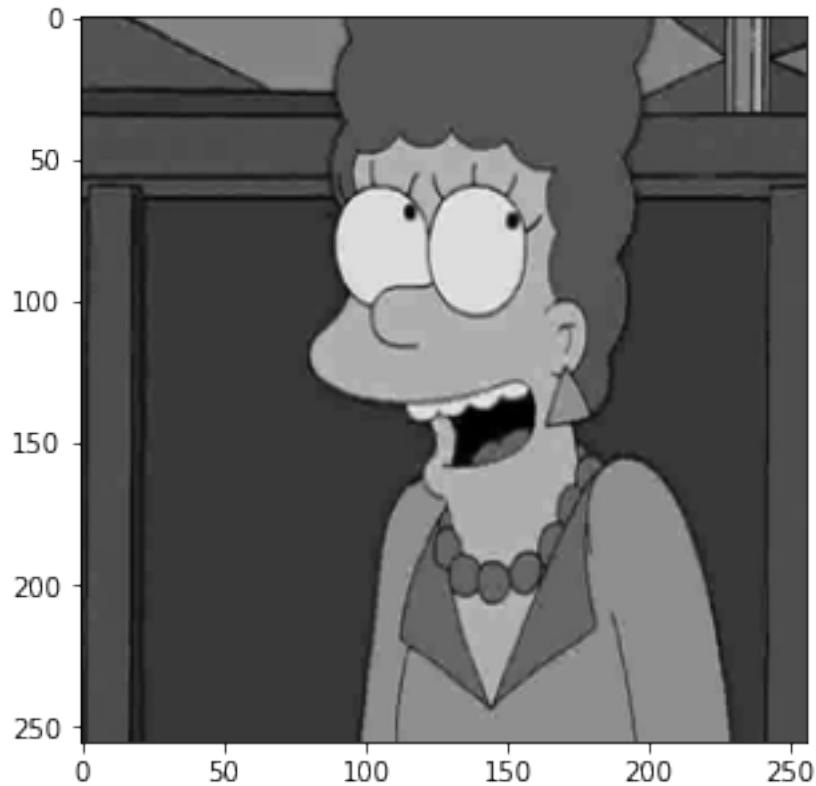
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1193226D0>



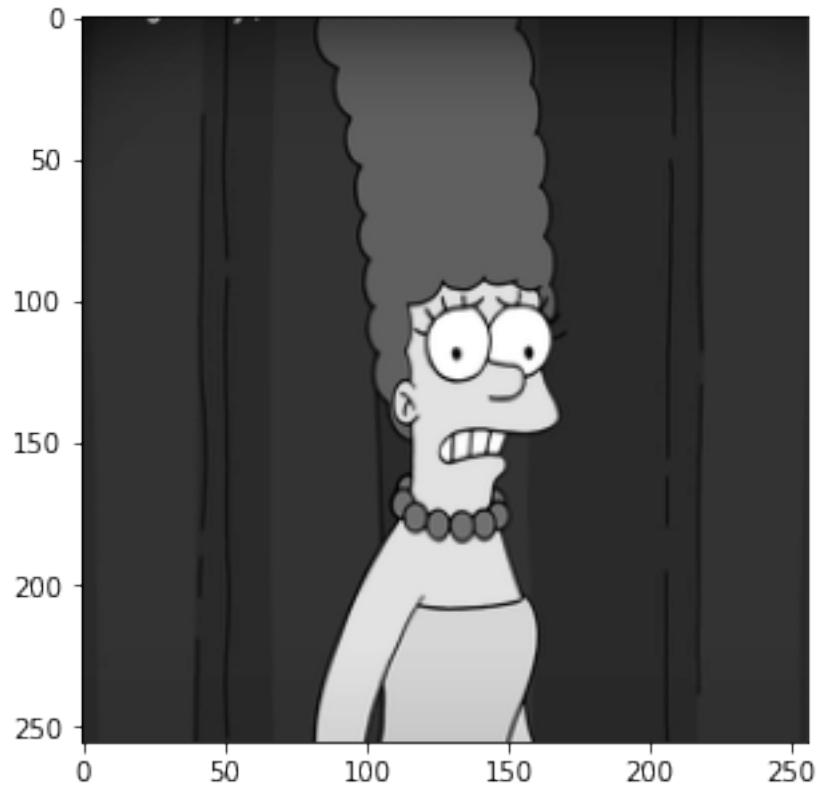
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322710>



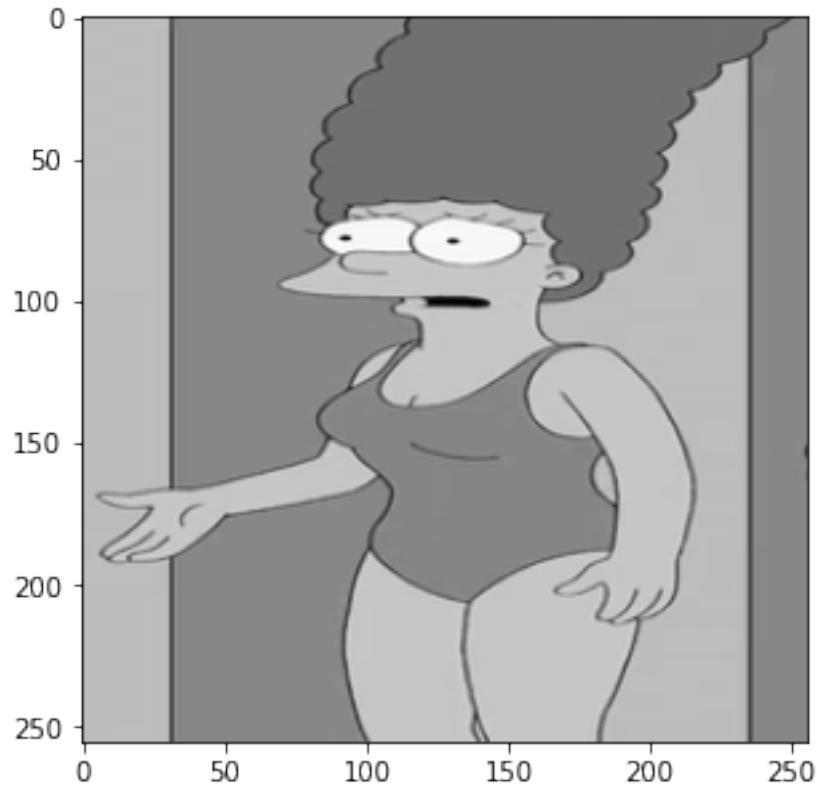
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322750>



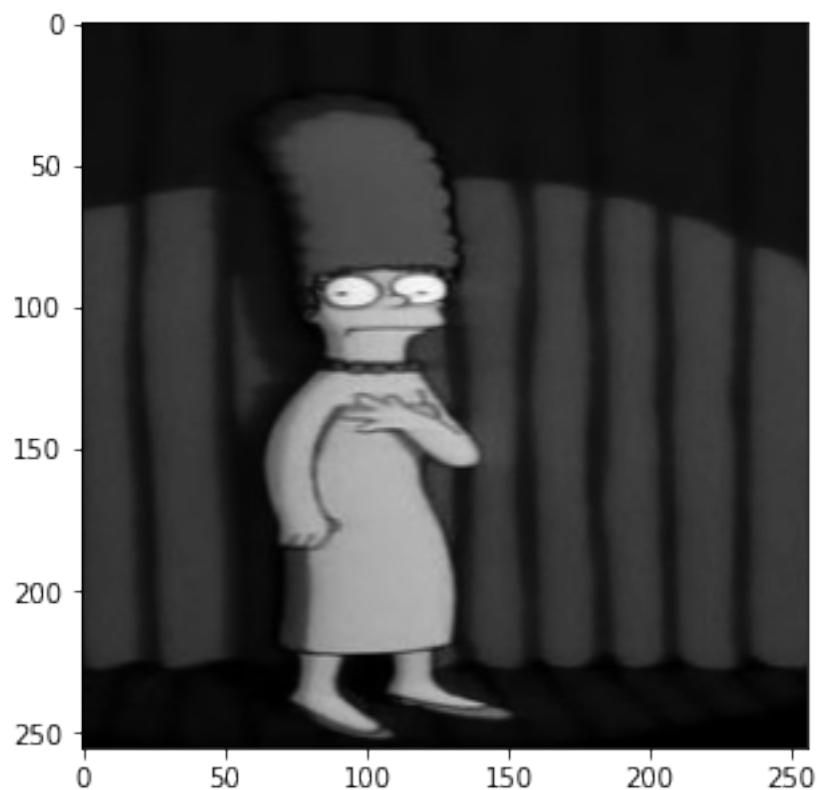
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322790>



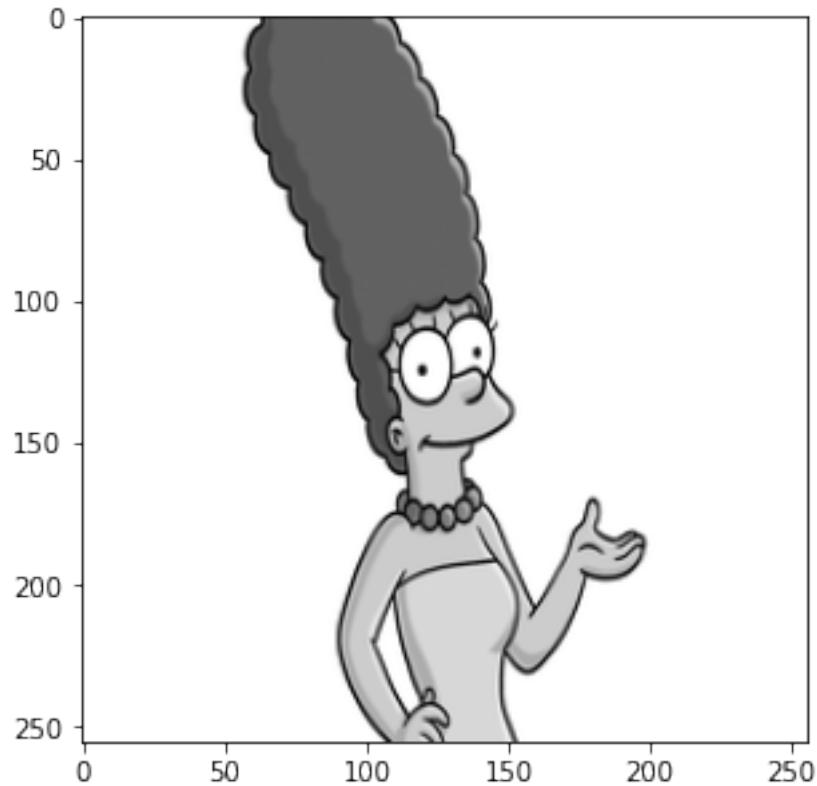
<PIL.Image.Image image mode=L size=256x256 at 0x7FB1193227D0>



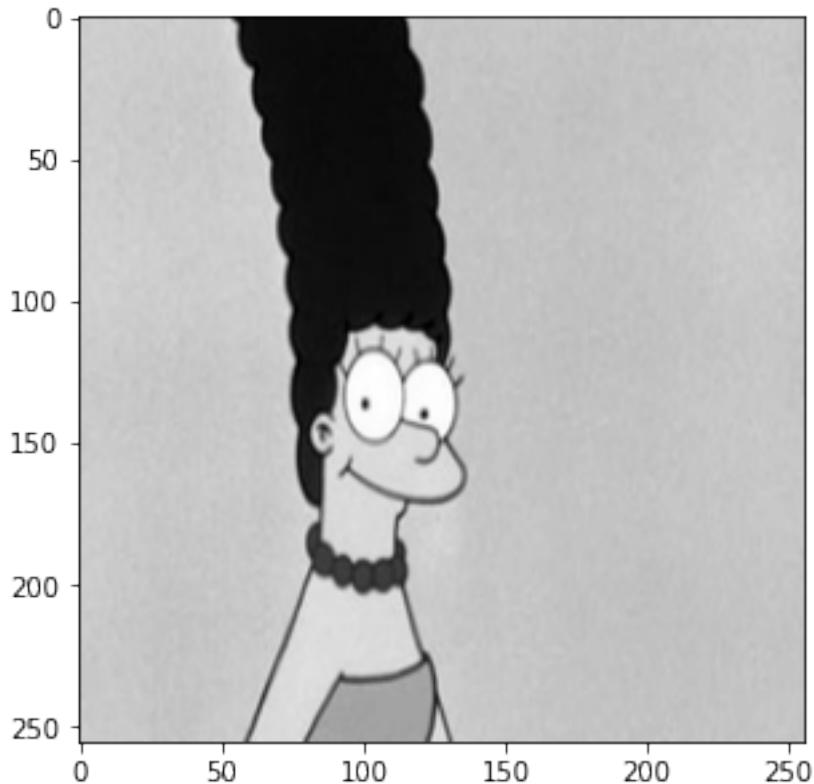
<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322810>



<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322850>



<PIL.Image.Image image mode=L size=256x256 at 0x7FB119322890>



```
[ ]:
```

```
[ ]: averageHomer = np.array(grayScaleHomer[0], 'f')

for imname in grayScaleHomer[1:]:
    averageHomer += np.array(imname)

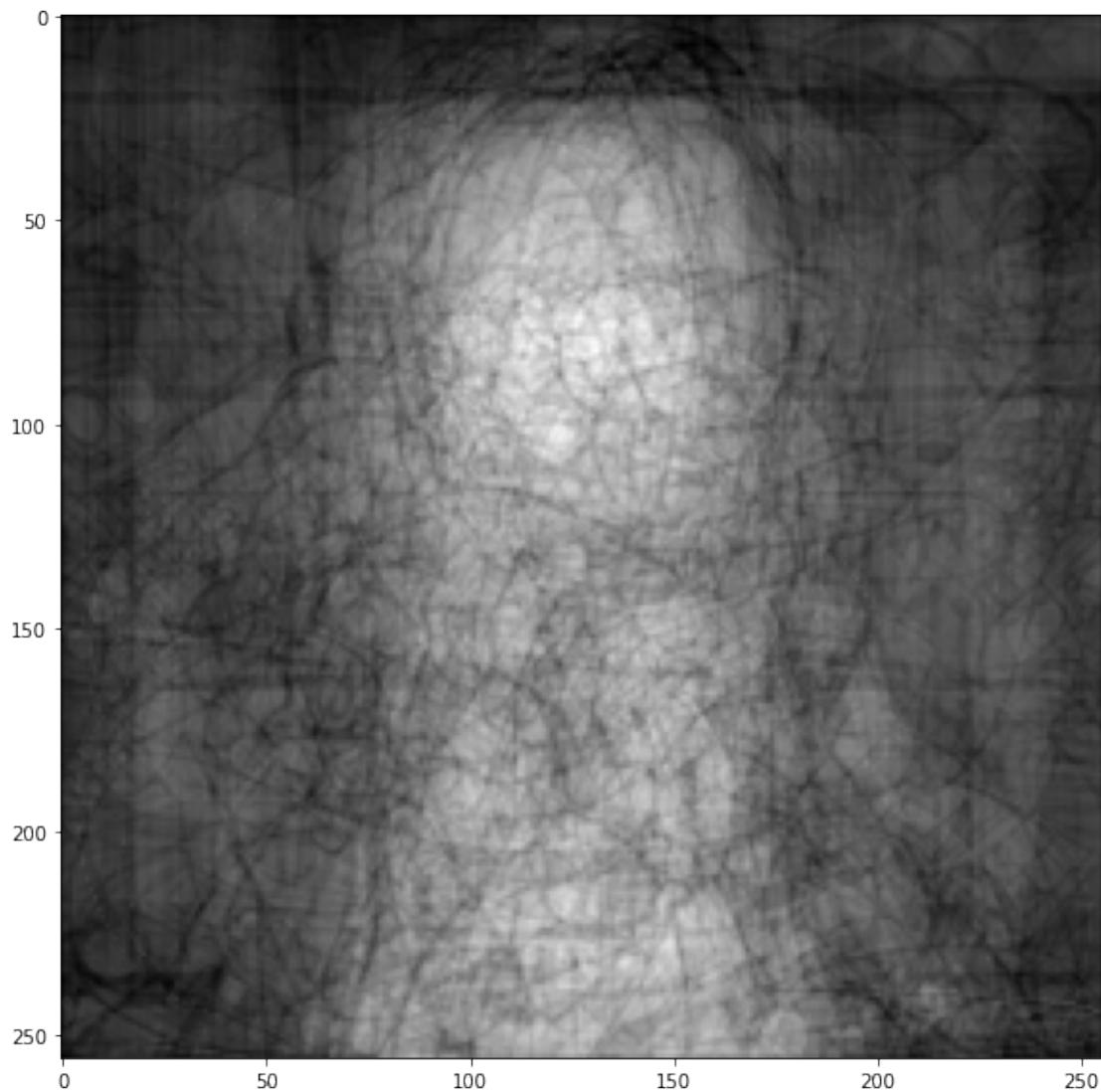
averageHomer /= len(grayScaleHomer)

# Data is in Numpy array
print(averageHomer)

plt.rcParams['figure.figsize'] = [10, 10]
plt.imshow(averageHomer, cmap=plt.cm.gray)
plt.show()
```

```
[[111.2 114.5 115. ... 123.94 125.86 120.68]
 [112.3 118.4 118.04 ... 126.74 128.76 123.5 ]
 [113.58 119.24 119.26 ... 129.52 131.14 125.5 ]
 ...
 [115.5 117. 115.74 ... 114.86 119.66 116.16]
 [113.84 114.36 114.6 ... 115.16 118.22 114.36]
```

```
[108.44 111.04 112.5 ... 117.88 119.2 113.54]]
```



```
[ ]: averageBart = np.array(grayScaleBart[0], 'f')

for imname in grayScaleBart[1:]:
    averageBart += np.array(imname)

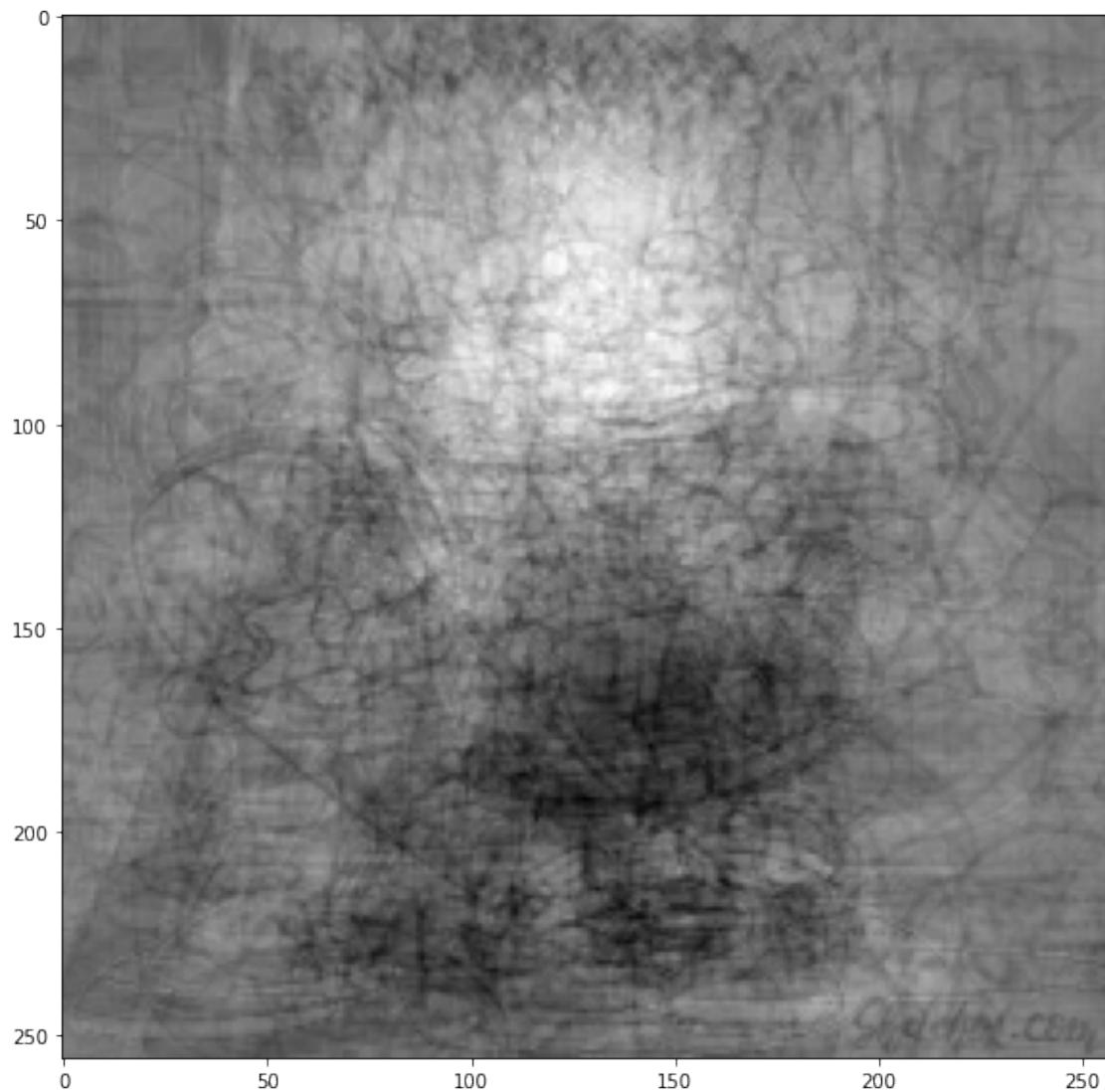
averageBart /= len(grayScaleBart)

# Data is in Numpy array
print(averageBart)

plt.rcParams['figure.figsize'] = [10, 10]
```

```
plt.imshow(averageBart, cmap=plt.cm.gray)
plt.show()

[[148.74 151.3 153.66 ... 154.5 154.78 152.58]
 [150.32 153.52 155.72 ... 154.66 154.3 152.12]
 [152.28 152.98 154.76 ... 154.34 154.36 152.24]
 ..
 [155.82 155.36 155.68 ... 148.32 151.68 151.82]
 [153.72 154.38 155.4 ... 155. 156.24 157.16]
 [153.92 152.5 152.84 ... 154.32 153.72 153.94]]
```



```
[ ]: averageLisa = np.array(grayScaleLisa[0], 'f')
```

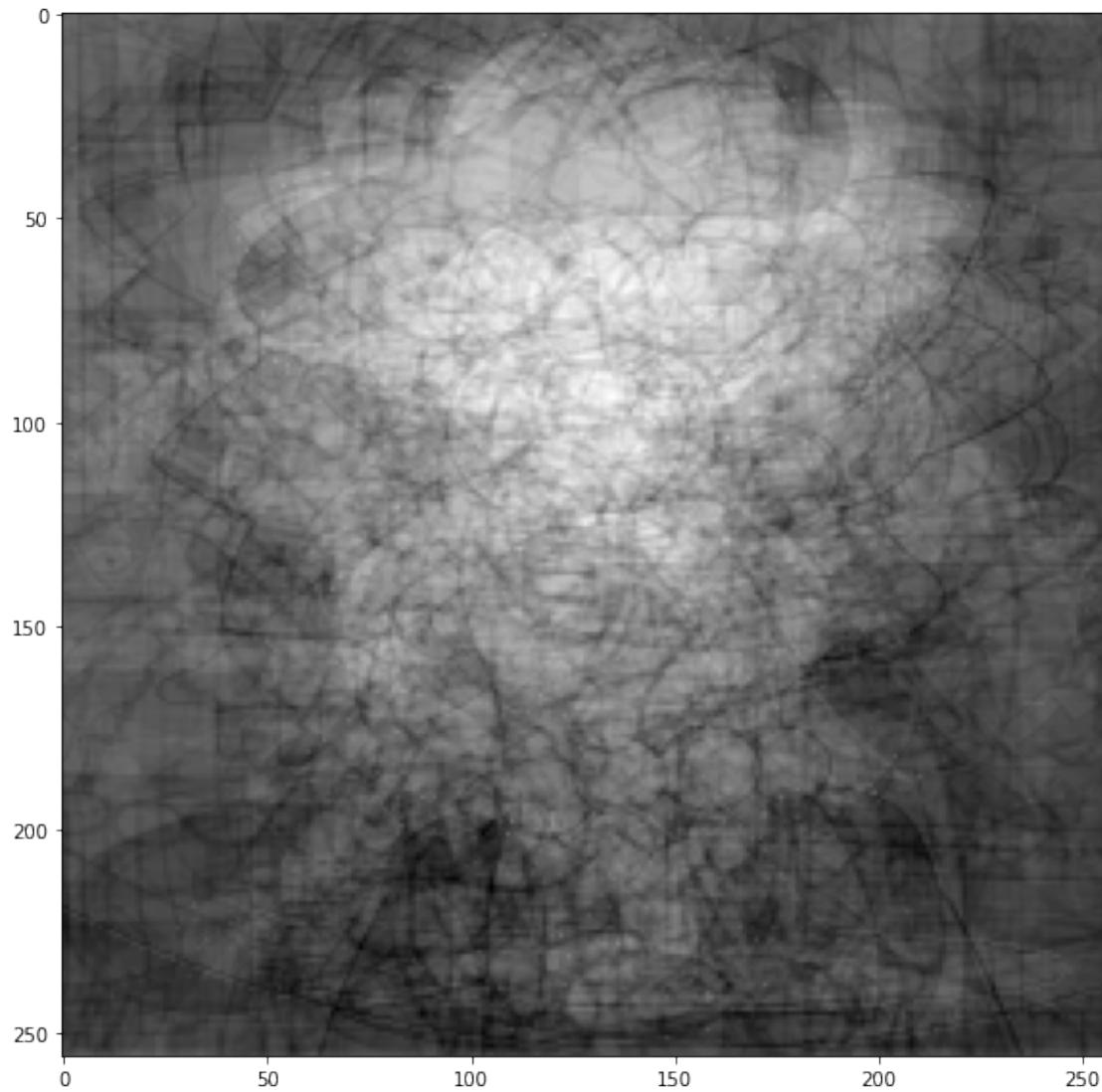
```
for imname in grayScaleLisa[1:]:
    averageLisa += np.array(imname)

averageLisa /= len(grayScaleLisa)

# Data is in Numpy array
print(averageLisa)

plt.rcParams['figure.figsize'] = [10, 10]
plt.imshow(averageLisa, cmap=plt.cm.gray)
plt.show()
```

[[140.8 143.76 144.68 ... 137.24 136.18 132.12]
 [142.94 144.14 144.44 ... 134.4 135.88 133.48]
 [144.58 146.66 145.54 ... 133.12 133.44 133.24]
 ...
 [130.58 130.78 126.26 ... 126.56 126.7 121.44]
 [131.76 130.48 129.6 ... 129.84 130. 126.88]
 [133.32 133.38 133.28 ... 133.86 132.9 131.02]]



```
[ ]: averageMaggie = np.array(grayScaleMaggie[0], 'f')

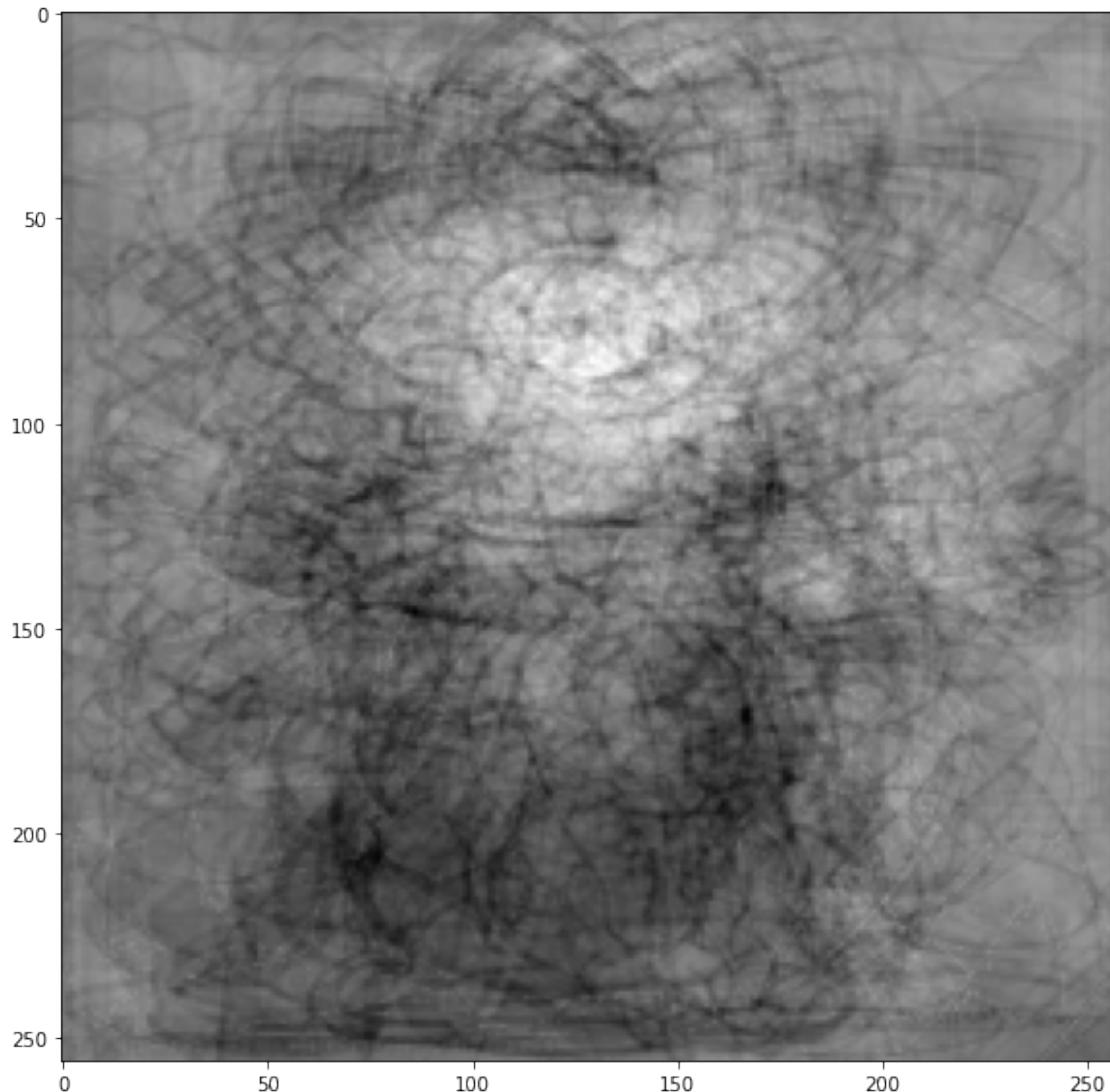
for imname in grayScaleMaggie[1:]:
    averageMaggie += np.array(imname)

averageMaggie /= len(grayScaleMaggie)

# Data is in Numpy array
print(averageMaggie)

plt.rcParams['figure.figsize'] = [10, 10]
plt.imshow(averageMaggie, cmap=plt.cm.gray)
plt.show()
```

```
[[163.06 163.44 163.34 ... 170.9 167.52 166.1 ]
 [164.24 164.46 164.76 ... 169.64 166.92 167.08]
 [164.24 164.4 164.86 ... 167.58 167.22 167.56]
 ...
 [159.92 158.8 158.2 ... 155.38 156.36 155.48]
 [159.18 159.24 158.32 ... 155.22 155.64 154.36]
 [159.6 158.06 156.12 ... 154.74 155.04 154.18]]
```



```
[ ]: averageMarge = np.array(grayScaleMarge[0], 'f')

for imname in grayScaleMarge[1:]:
    averageMarge += np.array(imname)
```

```

averageMarge /= len(grayScaleMarge)

# Data is in Numpy array
print(averageMarge)

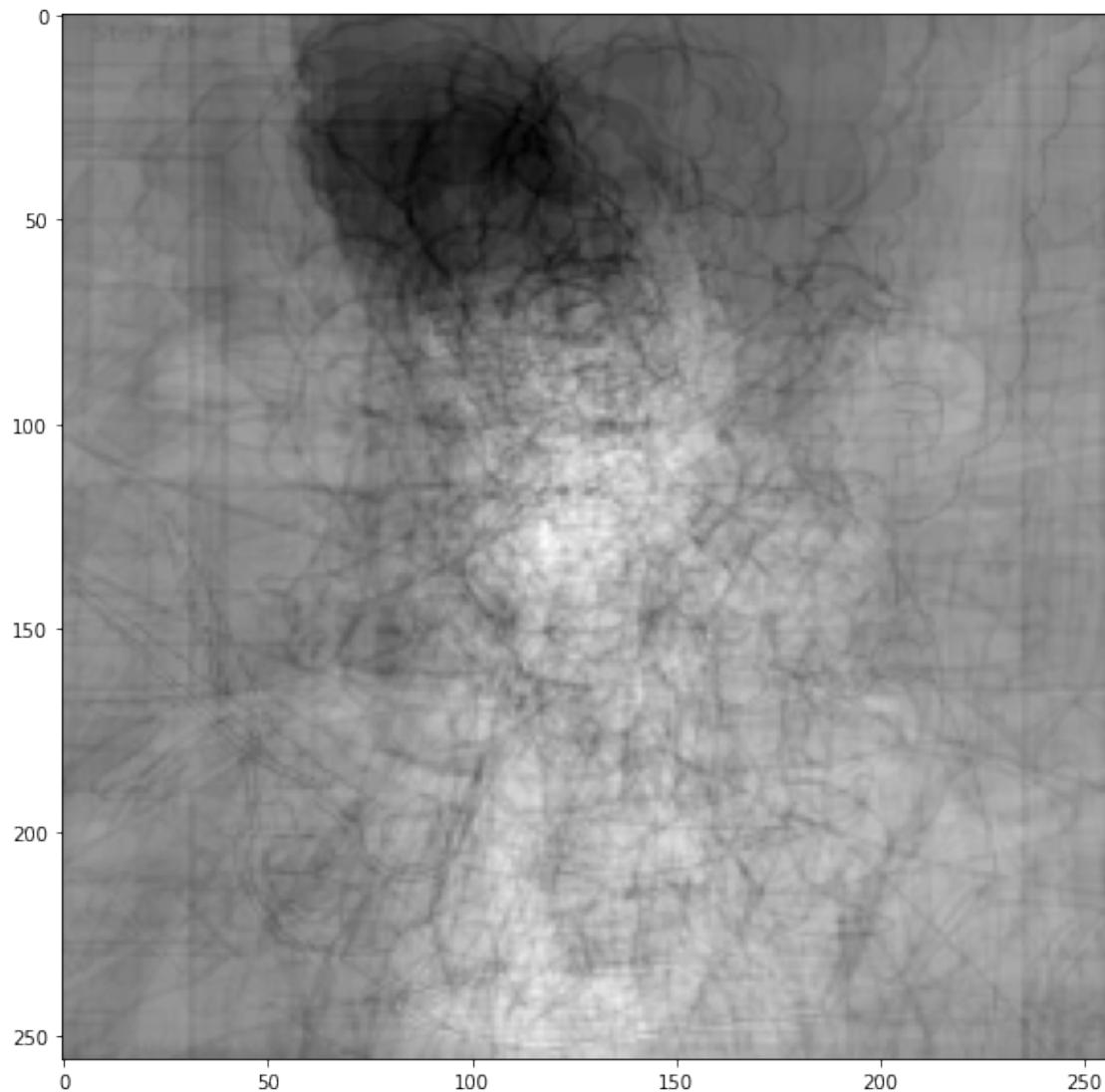
plt.rcParams['figure.figsize'] = [10, 10]
plt.imshow(averageMarge, cmap=plt.cm.gray)
plt.show()

```

```

[[136.74 140.26 141.26 ... 143.12 144.94 143.76]
 [139.34 142.96 143.42 ... 145.34 147.04 146.42]
 [140.04 142.8 142.78 ... 148.26 146.9 146.8 ]
 ...
 [142.52 143.64 141.78 ... 150.9 147.74 146.24]
 [141.6 143.06 140.9 ... 149.8 146.8 145.46]
 [138.66 140.42 139.06 ... 148.66 145.8 144.5 ]]

```



```
[ ]: for index, image in enumerate(grayScaleHomer):
    image.save("processedHomer/image-"+str(index)+".jpg")

for index, image in enumerate(grayScaleBart):
    image.save("processedBart/image-"+str(index)+".jpg")

for index, image in enumerate(grayScaleLisa):
    image.save("processedLisa/image-"+str(index)+".jpg")

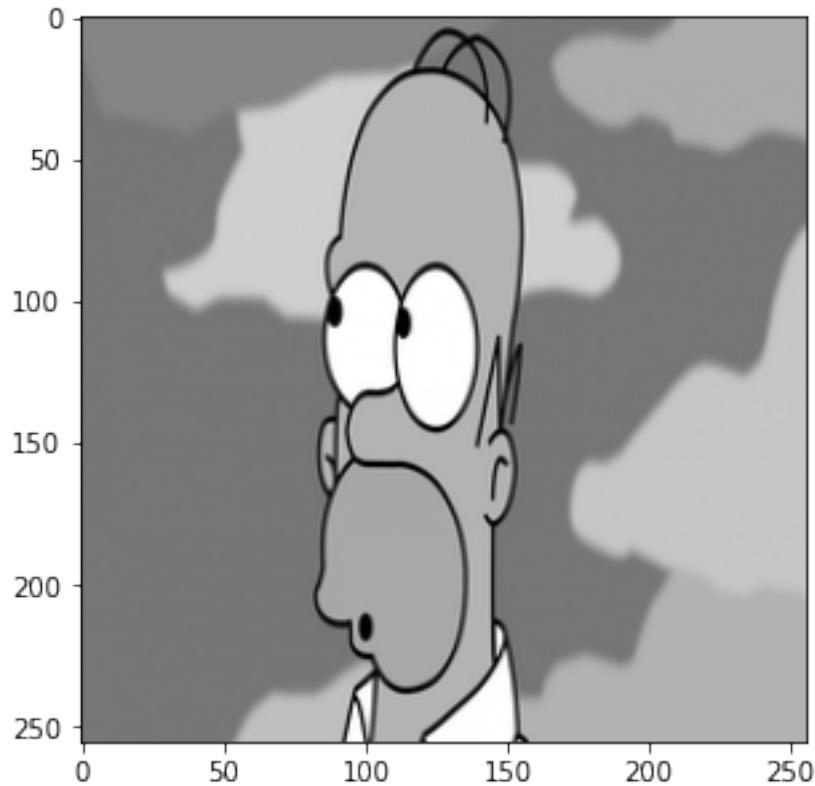
for index, image in enumerate(grayScaleMaggie):
    image.save("processedMaggie/image-"+str(index)+".jpg")

for index, image in enumerate(grayScaleMarge):
    image.save("processedMarge/image-"+str(index)+".jpg")
```

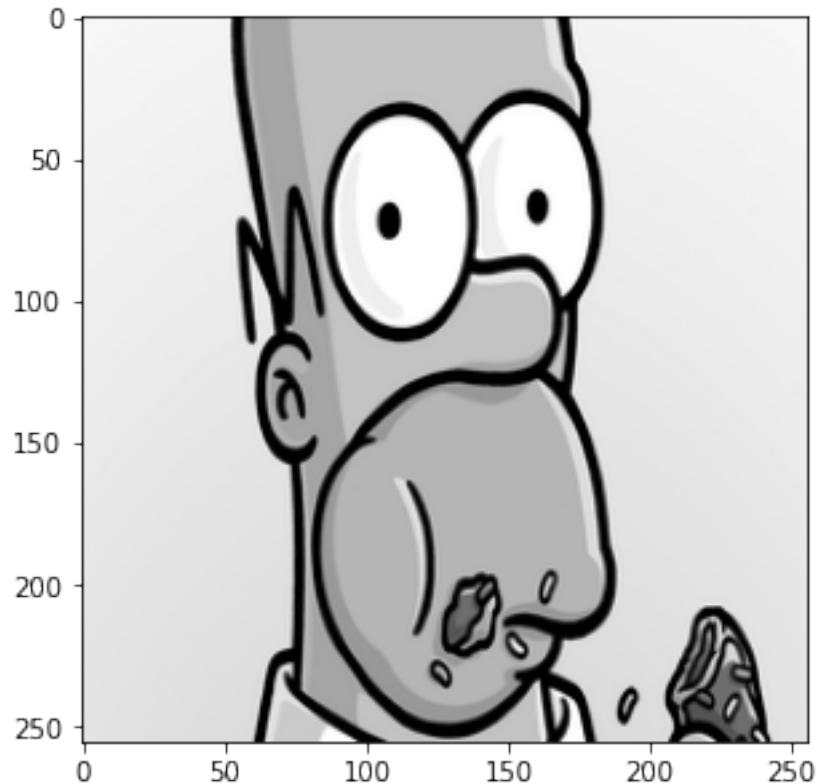
```
[ ]: plt.rcParams['figure.figsize'] = [5, 5]

for index, image in enumerate(grayScaleHomer):
    image = np.array(image)
    norm = image/(image.max()/255)
    print(norm)
    #convert back to image
    im = Image.fromarray(norm)
    im = im.convert('RGB')
    im.save("normalizedHomer/image-"+str(index)+".jpg")
    plt.imshow(im, cmap=plt.cm.gray)
    plt.show()
```

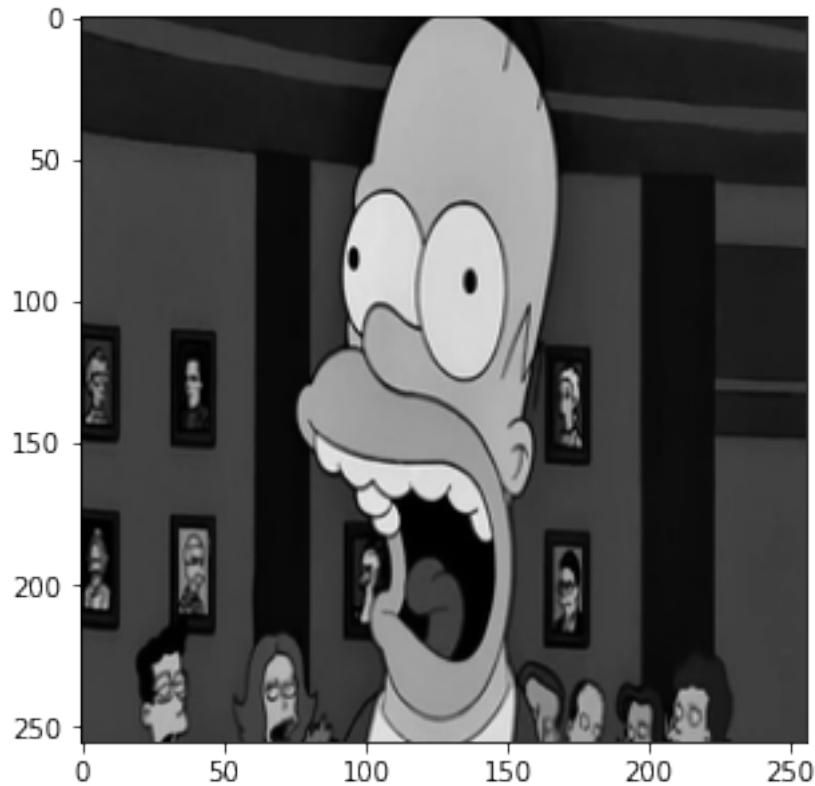
```
[[133. 133. 133. ... 172. 172. 172.]
 [133. 133. 133. ... 172. 172. 172.]
 [132. 133. 133. ... 172. 172. 172.]
 ...
 [118. 118. 118. ... 178. 178. 178.]
 [118. 118. 118. ... 178. 178. 178.]
 [118. 118. 118. ... 178. 178. 178.]]
```



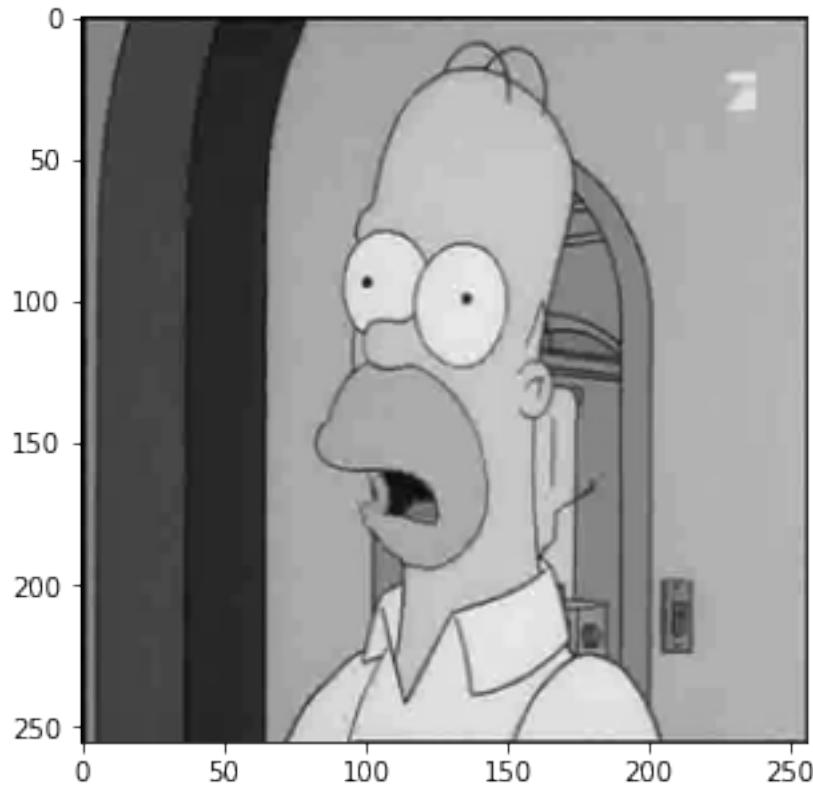
```
[[250. 249. 248. ... 248. 249. 250.]  
 [249. 249. 248. ... 248. 249. 249.]  
 [248. 248. 248. ... 248. 248. 248.]  
 ...  
 [228. 227. 227. ... 227. 227. 227.]  
 [227. 227. 227. ... 227. 227. 227.]  
 [227. 227. 227. ... 227. 227. 227.]]
```



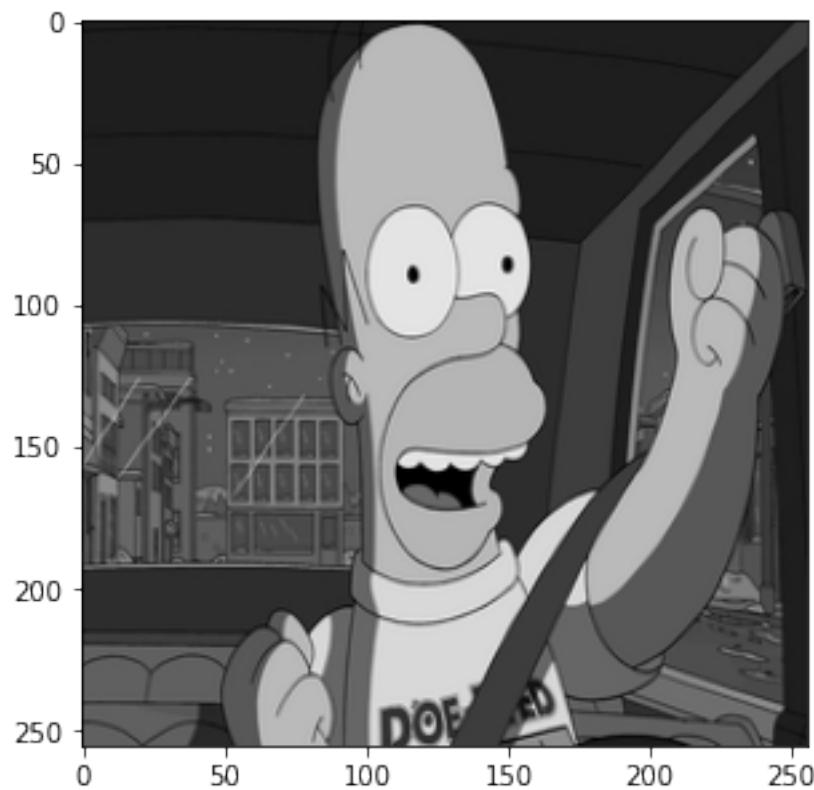
```
[[60.9561753 60.9561753 62.98804781 ... 31.4940239 31.4940239  
32.50996016]  
[64.00398406 61.97211155 62.98804781 ... 31.4940239 31.4940239  
32.50996016]  
[65.01992032 62.98804781 64.00398406 ... 31.4940239 31.4940239  
32.50996016]  
...  
[66.03585657 66.03585657 65.01992032 ... 62.98804781 62.98804781  
60.9561753 ]  
[64.00398406 64.00398406 65.01992032 ... 62.98804781 62.98804781  
60.9561753 ]  
[65.01992032 65.01992032 65.01992032 ... 62.98804781 61.97211155  
59.94023904]]
```



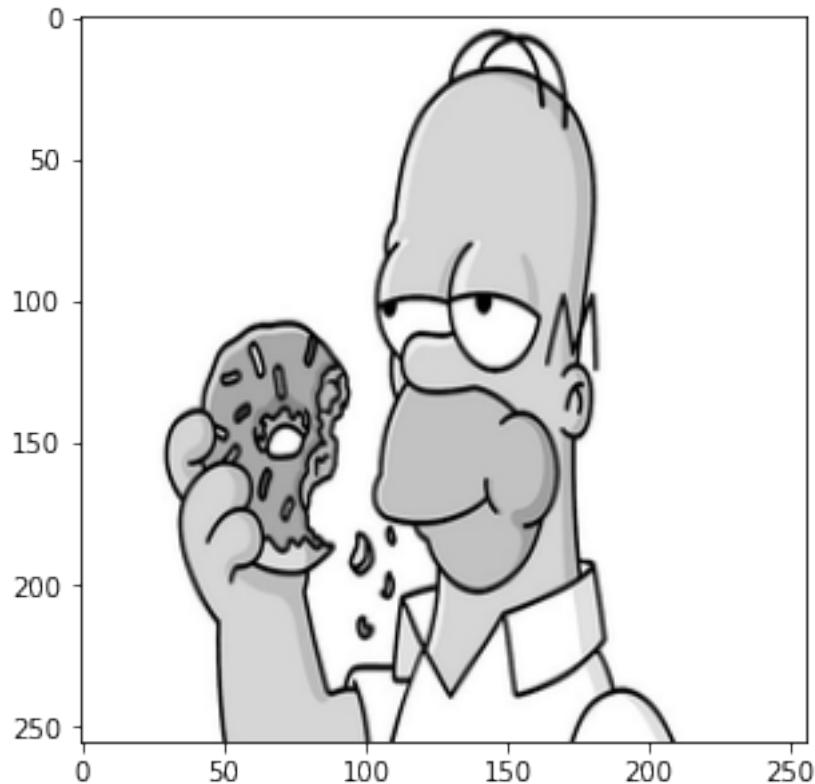
```
[[ 10.15486726   3.38495575   4.51327434 ...   3.38495575   20.30973451
  4.51327434]
 [ 6.7699115    6.7699115    44.00442478 ...   50.77433628   78.98230088
  49.6460177 ]
 [ 5.64159292   19.18141593   98.16371681 ...  183.9159292   195.19911504
 150.06637168]
...
[ 6.7699115    18.05309735   91.39380531 ...  179.40265487  183.9159292
 125.24336283]
[ 6.7699115    18.05309735   91.39380531 ...  189.55752212  192.94247788
 132.01327434]
[ 6.7699115    18.05309735   91.39380531 ...  110.57522124  110.57522124
 67.69911504]]
```



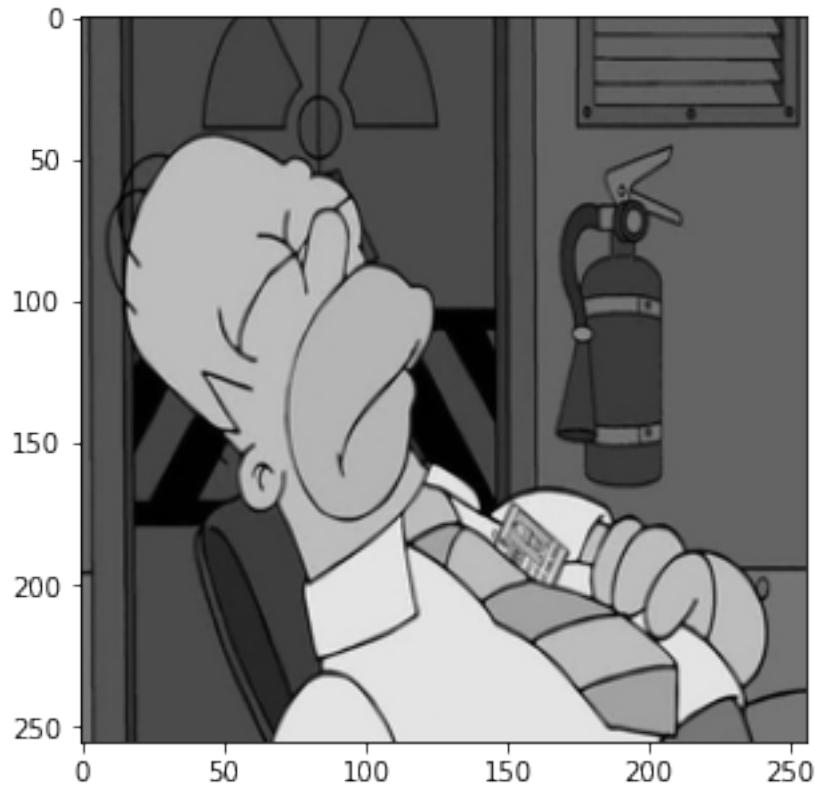
```
[[30.30737705 30.30737705 30.30737705 ... 63.75       62.70491803
 62.70491803]
[30.30737705 30.30737705 30.30737705 ... 61.65983607 62.70491803
 62.70491803]
[31.35245902 31.35245902 31.35245902 ... 62.70491803 62.70491803
 62.70491803]
...
[96.14754098 96.14754098 96.14754098 ... 25.08196721 28.21721311
 19.85655738]
[96.14754098 96.14754098 96.14754098 ... 22.99180328 30.30737705
 32.39754098]
[96.14754098 96.14754098 96.14754098 ... 22.99180328 29.26229508
 30.30737705]]
```



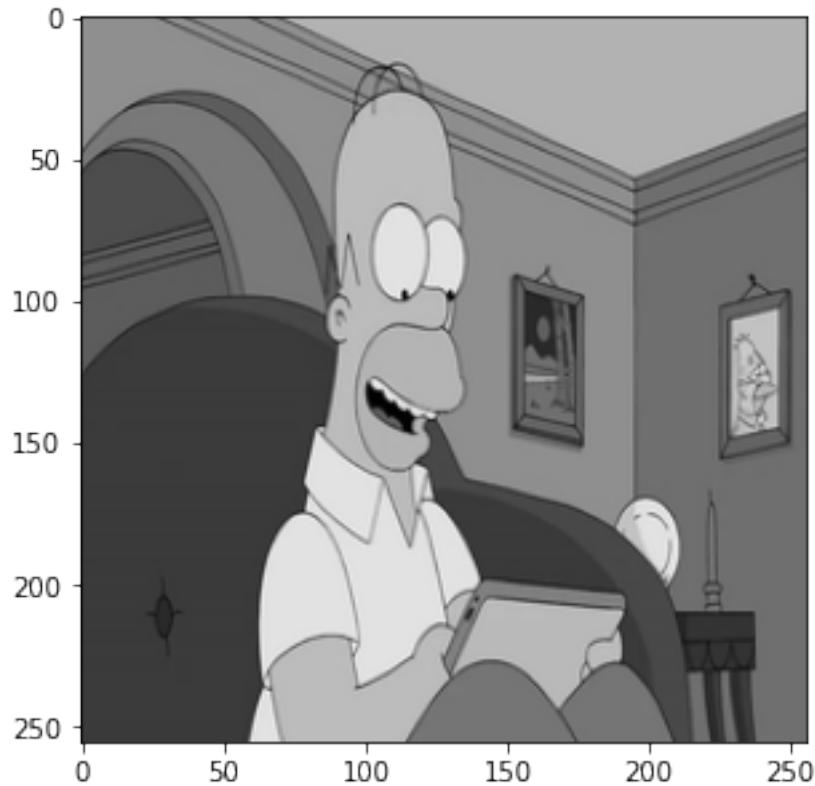
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



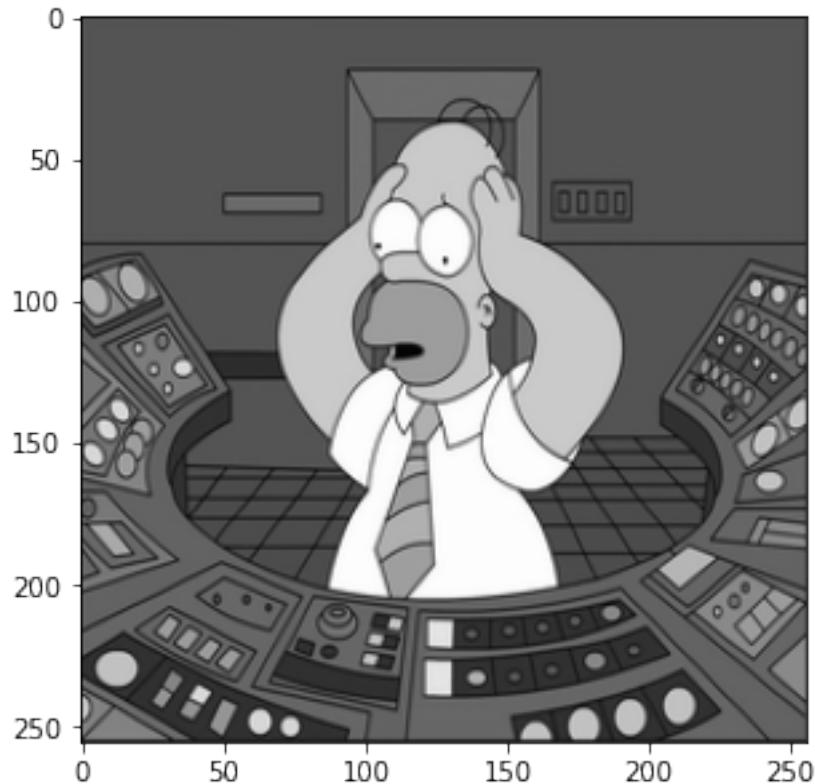
```
[[ 99.22594142 104.56066946 78.9539749 ... 53.34728033 106.69456067  
 96.0251046 ]]  
[102.42677824 105.62761506 81.08786611 ... 53.34728033 106.69456067  
 96.0251046 ]  
[101.35983264 105.62761506 80.0209205 ... 53.34728033 106.69456067  
 96.0251046 ]  
...  
[116.29707113 116.29707113 116.29707113 ... 118.43096234 118.43096234  
 118.43096234]  
[116.29707113 116.29707113 116.29707113 ... 118.43096234 118.43096234  
 118.43096234]  
[116.29707113 116.29707113 115.23012552 ... 118.43096234 118.43096234  
 118.43096234]]
```



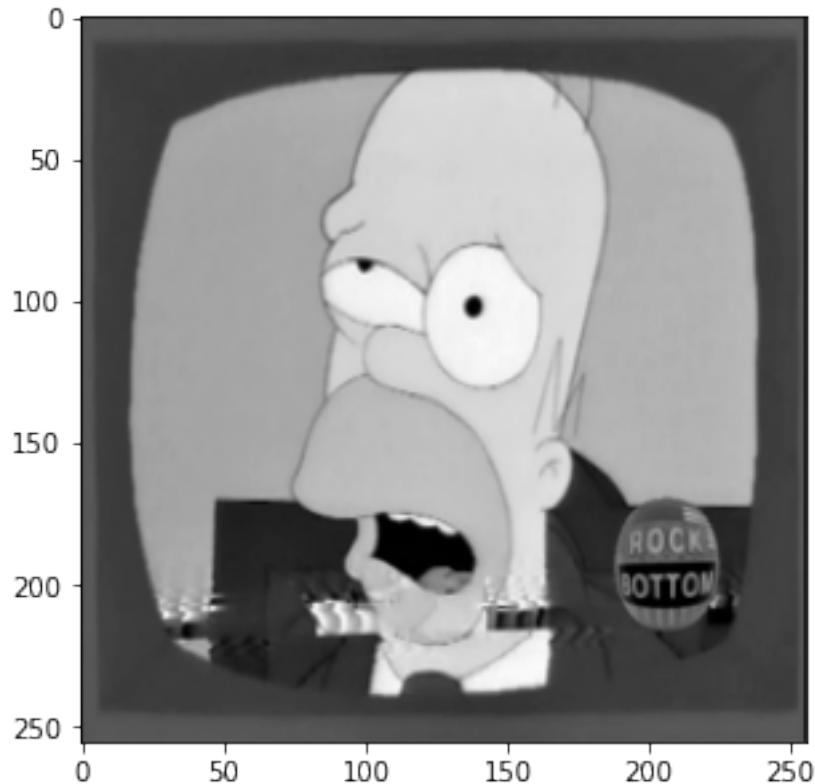
```
[[143.4375 143.4375 143.4375 ... 178.5      178.5      178.5      ]  
 [143.4375 143.4375 143.4375 ... 178.5      178.5      178.5      ]  
 [143.4375 143.4375 143.4375 ... 178.5      178.5      178.5      ]  
 ...  
 [ 58.4375   58.4375   58.4375 ... 111.5625 111.5625 111.5625]  
 [ 58.4375   58.4375   58.4375 ... 111.5625 111.5625 111.5625]  
 [ 58.4375   58.4375   58.4375 ... 111.5625 111.5625 111.5625]]
```



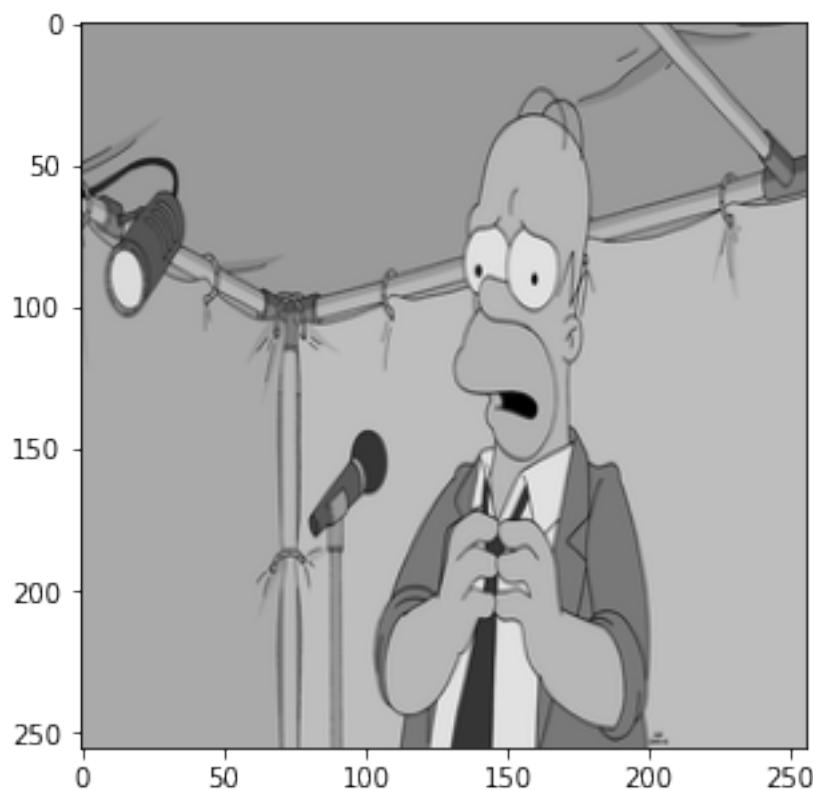
```
[[ 33.  46.  41. ... 42.  45.  36.]  
 [ 62.  93.  88. ... 87.  90.  78.]  
 [ 56.  87.  82. ... 83.  85.  74.]  
 ...  
 [ 94.  92.  58. ... 99. 102.  88.]  
 [ 36.  57. 125. ... 93.  96.  82.]  
 [ 19.  44.  50. ... 43.  43.  39.]]
```



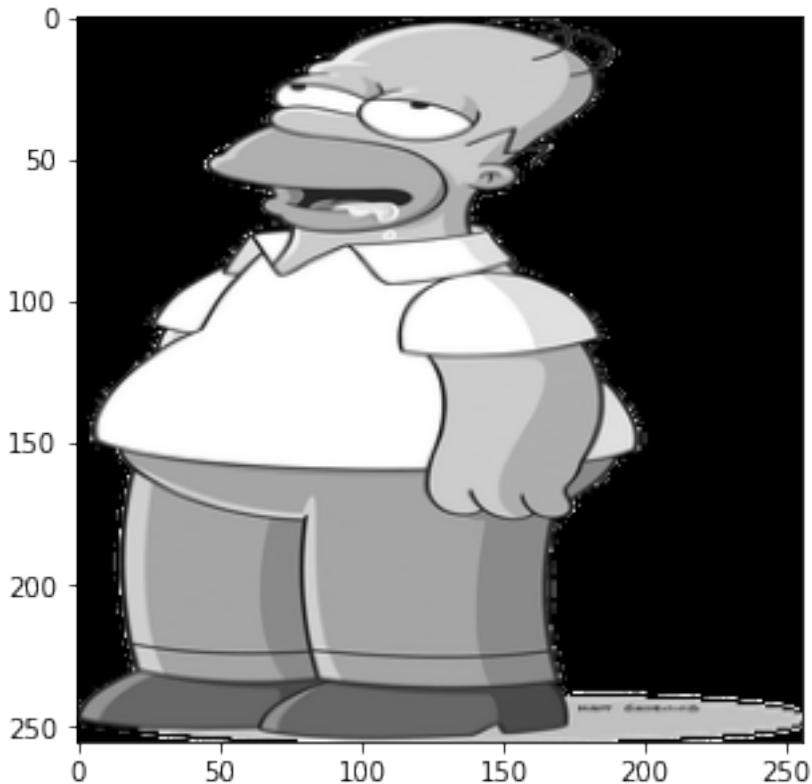
```
[[75.68493151 88.49315068 87.32876712 ... 89.65753425 95.47945205
 25.61643836]
[74.52054795 87.32876712 87.32876712 ... 93.15068493 95.47945205
 24.45205479]
[73.35616438 86.16438356 86.16438356 ... 91.98630137 94.31506849
 24.45205479]
...
[67.53424658 83.83561644 87.32876712 ... 90.82191781 90.82191781
 23.28767123]
[71.02739726 85.           85.           ... 87.32876712 90.82191781
 25.61643836]
[69.8630137 86.16438356 88.49315068 ... 91.98630137 94.31506849
 24.45205479]]
```



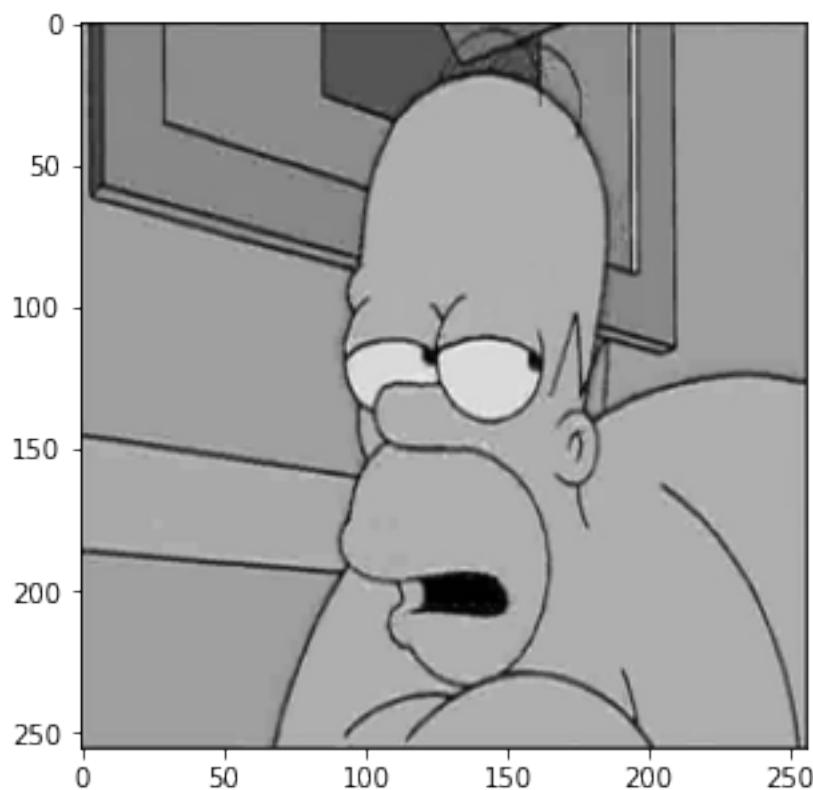
```
[[154.23387097 154.23387097 154.23387097 ... 154.23387097 154.23387097  
154.23387097]  
[154.23387097 154.23387097 154.23387097 ... 154.23387097 154.23387097  
154.23387097]  
[154.23387097 154.23387097 154.23387097 ... 154.23387097 154.23387097  
154.23387097]  
...  
[170.68548387 170.68548387 170.68548387 ... 190.22177419 190.22177419  
190.22177419]  
[170.68548387 170.68548387 170.68548387 ... 190.22177419 190.22177419  
190.22177419]  
[170.68548387 170.68548387 170.68548387 ... 190.22177419 190.22177419  
190.22177419]]
```



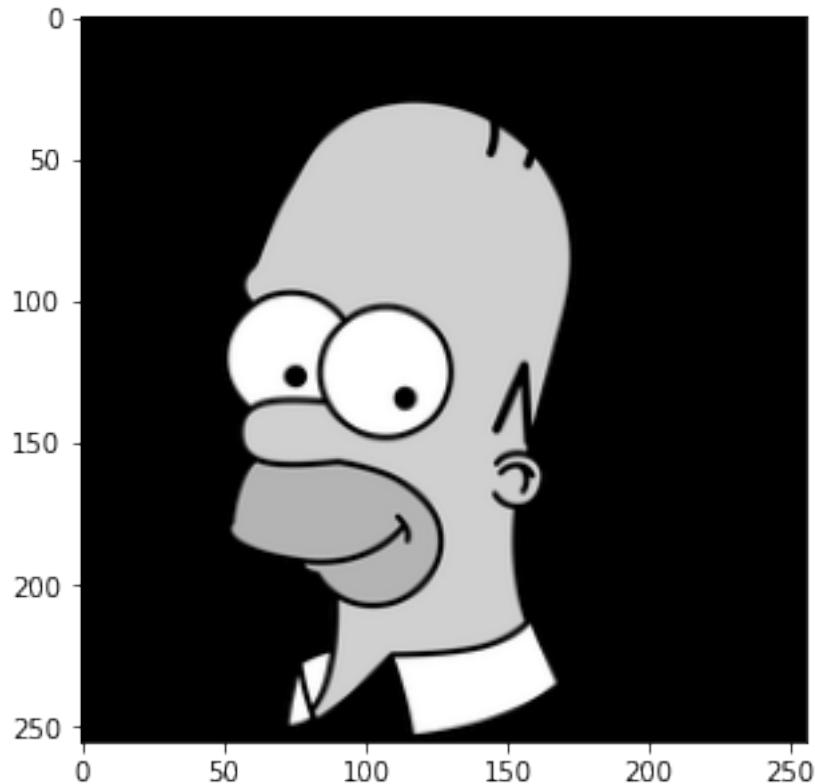
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



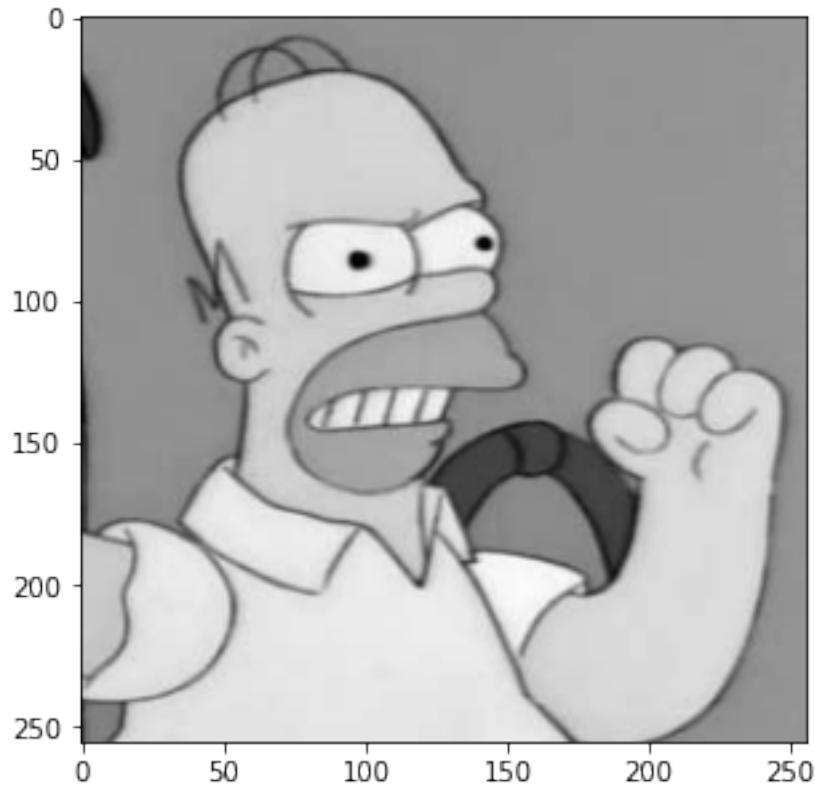
```
[[ 29.46215139  52.82868526  46.73306773 ... 60.9561753  66.03585657
 33.52589641]
 [ 92.4501992 164.58167331 128.00796813 ... 147.31075697 154.42231076
 103.62549801]
[111.75298805 192.01195219 149.34262948 ... 168.64541833 174.74103586
 125.97609562]
...
[103.62549801 161.53386454 159.50199203 ... 100.57768924 197.09163347
 124.96015936]
[103.62549801 161.53386454 159.50199203 ... 100.57768924 194.0438247
 121.9123506 ]
[ 60.9561753 110.73705179 109.72111554 ... 46.73306773 128.00796813
 62.98804781]]
```



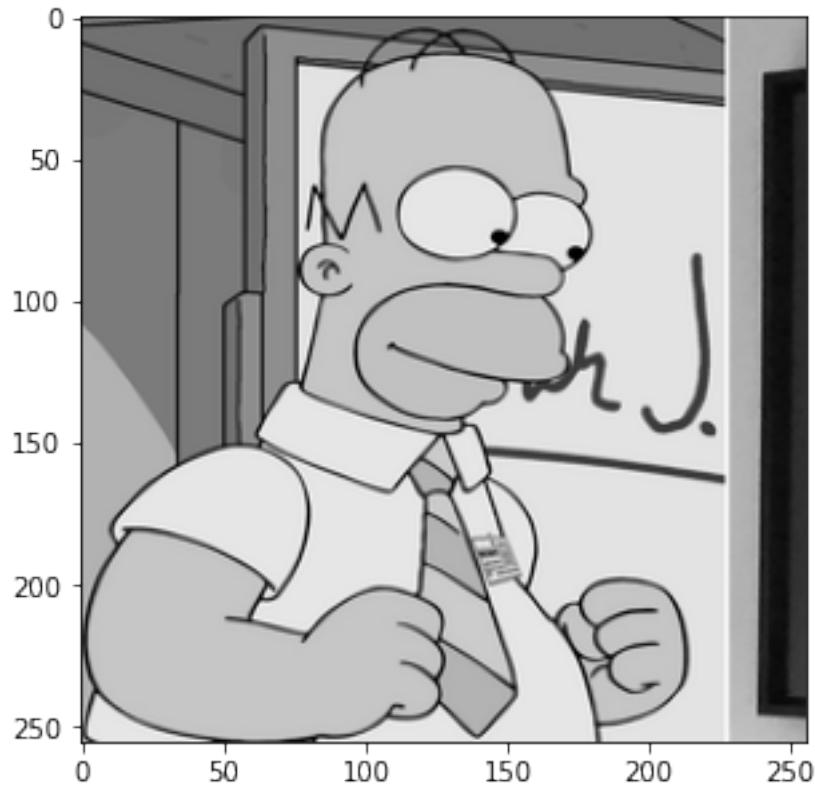
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



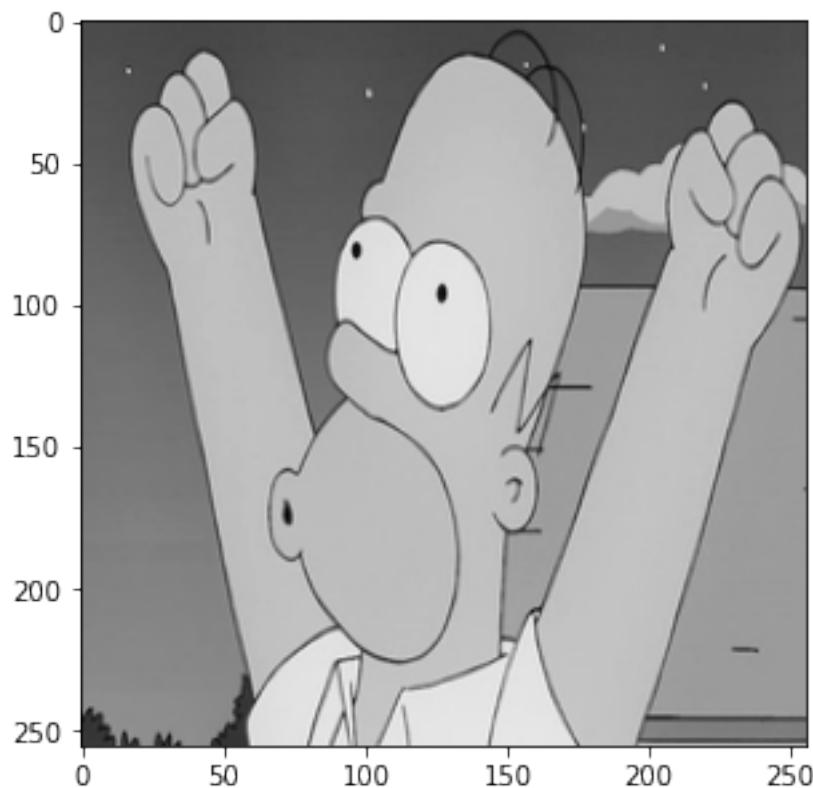
```
[[145.22421525 145.22421525 145.22421525 ... 149.79820628 149.79820628  
149.79820628]  
[145.22421525 145.22421525 145.22421525 ... 149.79820628 149.79820628  
149.79820628]  
[145.22421525 145.22421525 145.22421525 ... 149.79820628 149.79820628  
149.79820628]  
...  
[ 68.60986547 80.04484305 115.49327354 ... 148.65470852 148.65470852  
148.65470852]  
[ 67.46636771 80.04484305 116.6367713 ... 148.65470852 148.65470852  
148.65470852]  
[ 67.46636771 81.18834081 117.78026906 ... 148.65470852 148.65470852  
148.65470852]]
```



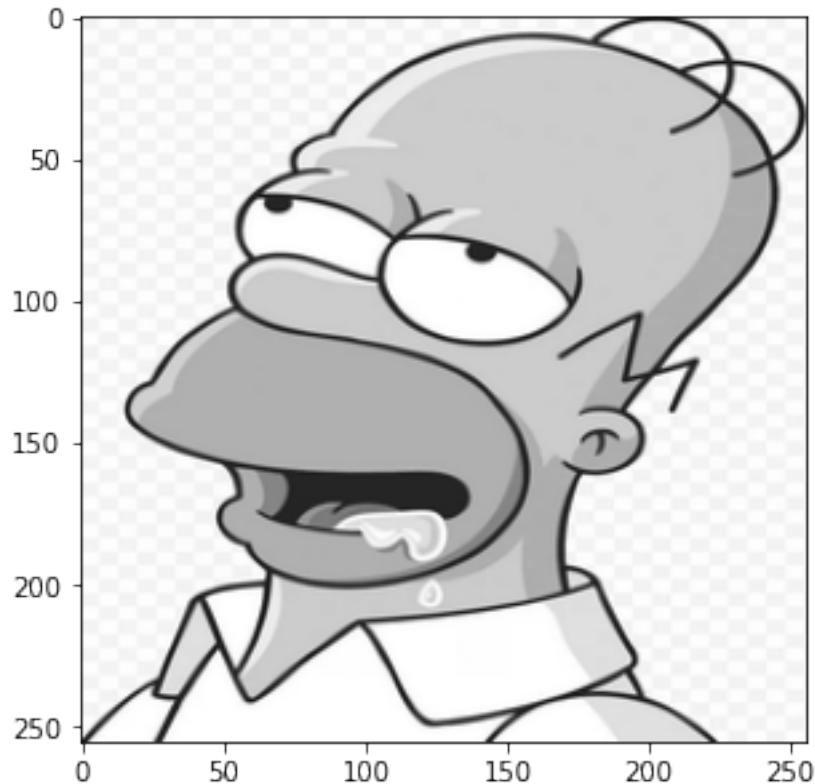
```
[[119. 119. 119. ... 170. 169. 169.]  
 [118. 119. 119. ... 169. 167. 168.]  
 [125. 119. 118. ... 168. 165. 166.]  
 ...  
 [192. 54. 106. ... 178. 173. 172.]  
 [187. 119. 13. ... 177. 178. 177.]  
 [176. 186. 132. ... 178. 178. 180.]]
```



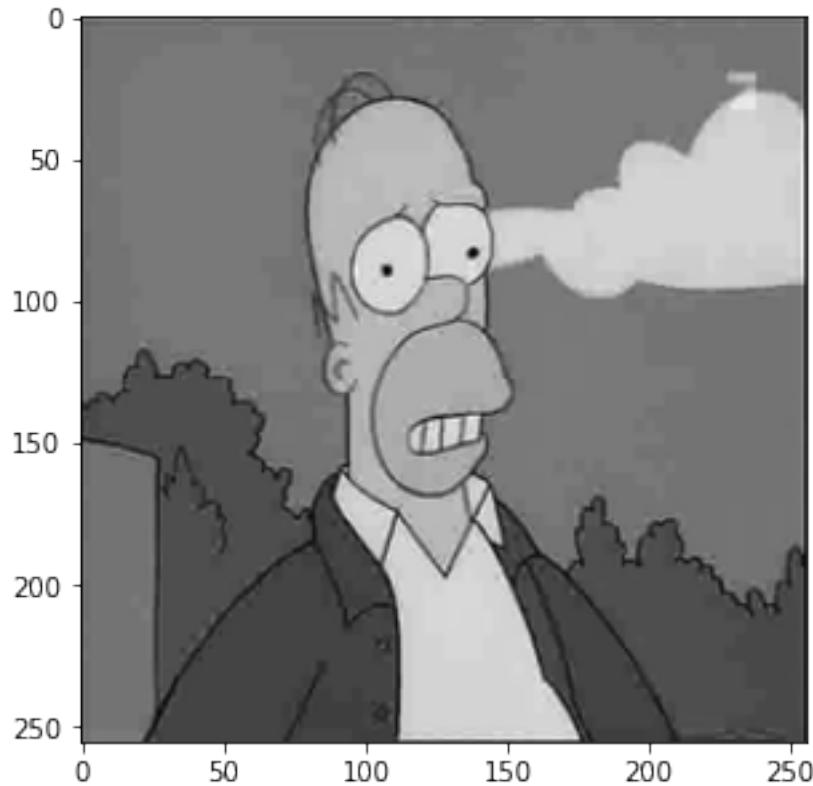
```
[[ 76.39676113  76.39676113  76.39676113 ...  75.36437247  75.36437247
  75.36437247]
 [ 76.39676113  76.39676113  77.4291498 ...  77.4291498  77.4291498
  77.4291498 ]
 [ 76.39676113  77.4291498  77.4291498 ...  75.36437247  75.36437247
  75.36437247]
 ...
 [ 57.81376518  56.78137652  55.74898785 ...  88.7854251  85.68825911
  82.59109312]
 [ 57.81376518  56.78137652  55.74898785 ...  80.52631579  80.52631579
  79.49392713]
 [ 56.78137652  55.74898785  55.74898785 ... 181.70040486 182.73279352
  182.73279352]]
```



```
[[243. 242. 243. ... 254. 254. 247.]  
 [243. 242. 243. ... 254. 254. 248.]  
 [243. 243. 242. ... 255. 254. 247.]  
 ...  
 [248. 243. 80. ... 254. 254. 247.]  
 [242. 105. 20. ... 254. 255. 248.]  
 [120. 22. 37. ... 254. 254. 248.]]
```



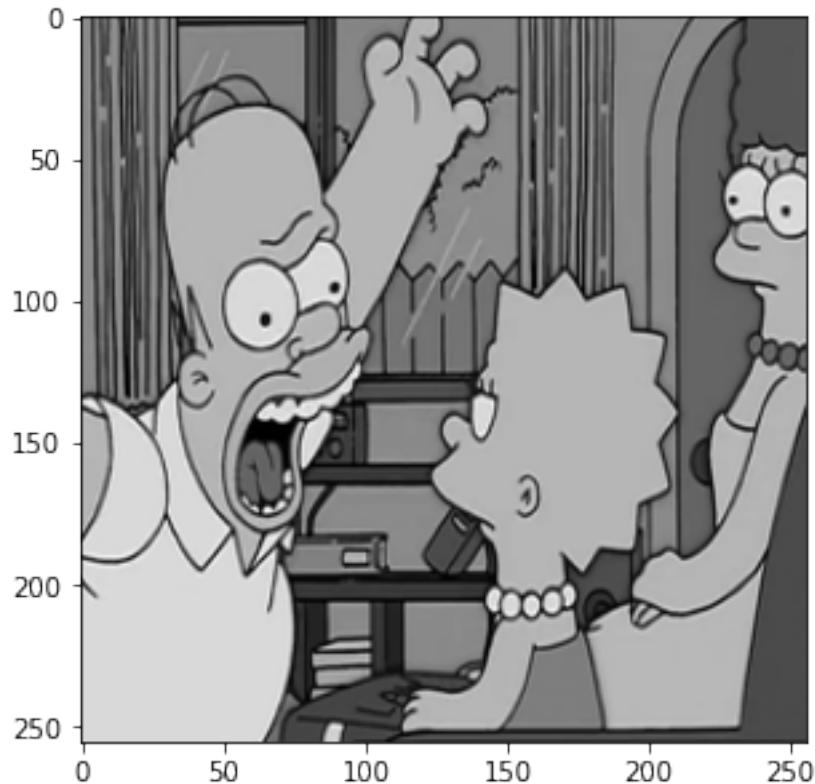
```
[[ 61.88284519 116.29707113 106.69456067 ... 119.49790795 136.56903766
  52.28033473]
 [ 73.61924686 136.56903766 125.89958159 ... 121.63179916 139.76987448
  52.28033473]
 [ 48.0125523 105.62761506 96.0251046 ... 106.69456067 124.83263598
  41.61087866]
 ...
 [ 60.81589958 124.83263598 113.09623431 ... 70.41841004 81.08786611
  22.40585774]
 [ 60.81589958 124.83263598 113.09623431 ... 70.41841004 81.08786611
  22.40585774]
 [ 60.81589958 124.83263598 113.09623431 ... 70.41841004 81.08786611
  22.40585774]]
```



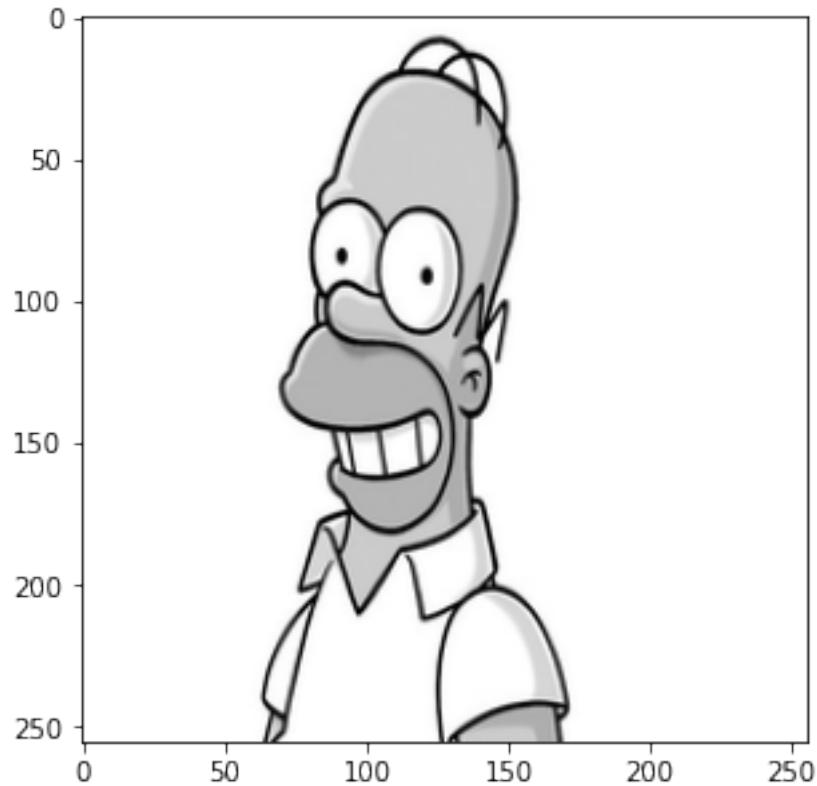
```
[[ 99.97797357  99.97797357  99.97797357 ... 102.2246696  102.2246696  
 102.2246696 ]]  
[ 99.97797357  99.97797357  99.97797357 ... 102.2246696  102.2246696  
 102.2246696 ]  
[ 99.97797357  99.97797357  99.97797357 ... 102.2246696  102.2246696  
 102.2246696 ]  
...  
[ 3.37004405   4.49339207   4.49339207 ...   4.49339207   4.49339207  
 4.49339207]  
[ 5.61674009   4.49339207   4.49339207 ...   4.49339207   5.61674009  
 4.49339207]  
[ 4.49339207   4.49339207   4.49339207 ...   5.61674009   4.49339207  
 4.49339207]]
```



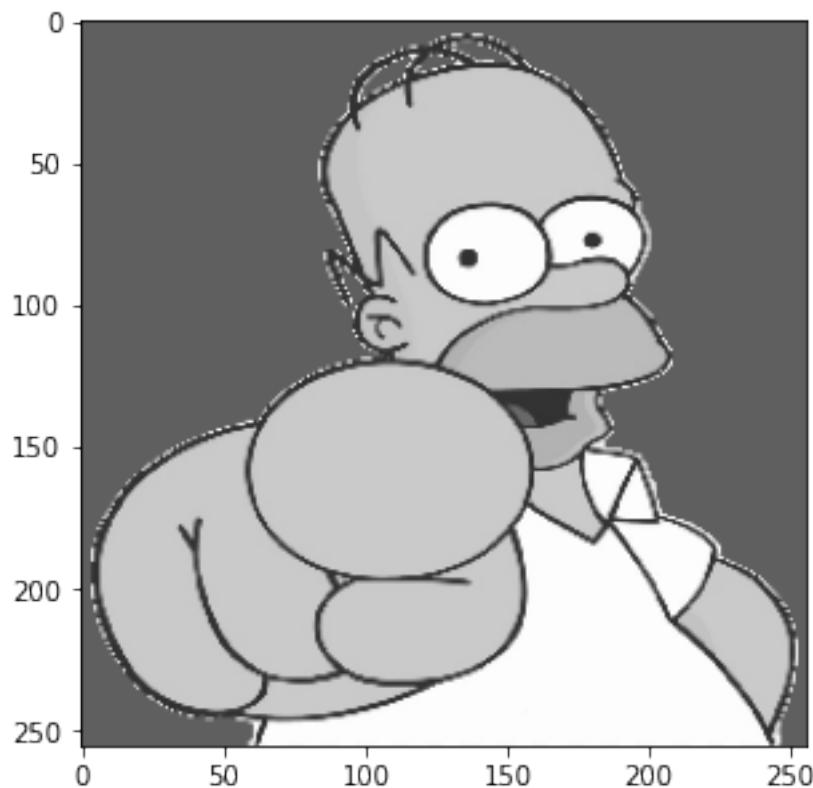
```
[[139. 139. 140. ... 85. 85. 85.]  
 [139. 139. 140. ... 85. 85. 85.]  
 [139. 139. 140. ... 85. 85. 85.]  
 ...  
 [217. 217. 217. ... 74. 74. 74.]  
 [217. 217. 217. ... 78. 78. 78.]  
 [217. 217. 217. ... 51. 51. 51.]]
```



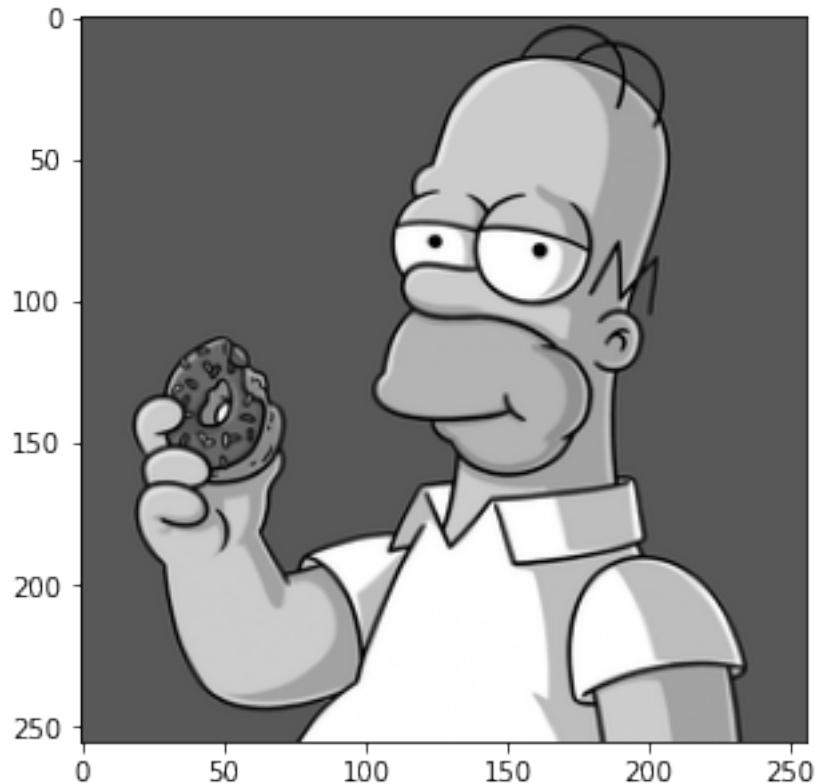
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



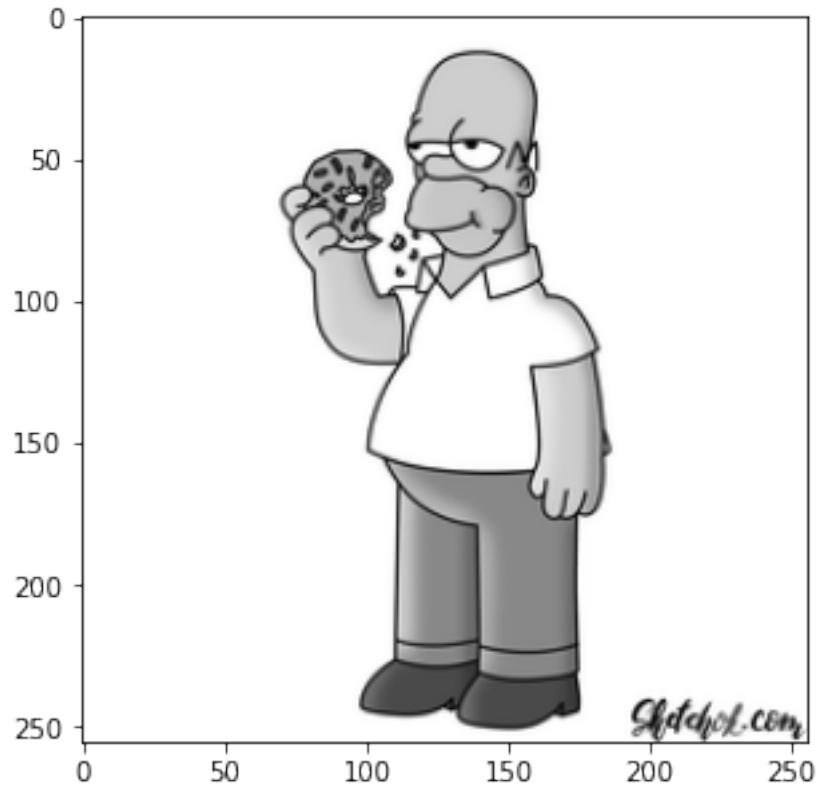
```
[[95. 95. 95. ... 95. 95. 95.]  
[95. 95. 95. ... 95. 95. 95.]  
[95. 95. 95. ... 95. 95. 95.]  
...  
[95. 95. 95. ... 95. 95. 95.]  
[95. 95. 95. ... 95. 95. 95.]  
[95. 95. 95. ... 95. 95. 95.]]
```



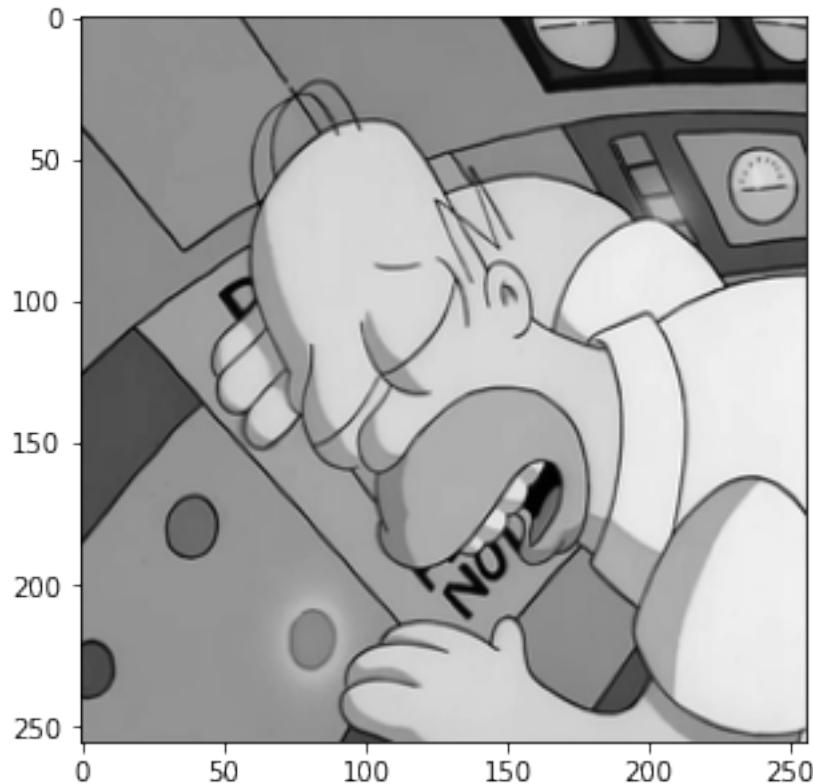
```
[[88. 88. 88. ... 88. 88. 88.]  
 [88. 88. 88. ... 88. 88. 88.]  
 [88. 88. 88. ... 88. 88. 88.]  
 ...  
 [88. 88. 88. ... 88. 88. 88.]  
 [88. 88. 88. ... 88. 88. 88.]  
 [88. 88. 88. ... 88. 88. 88.]]
```



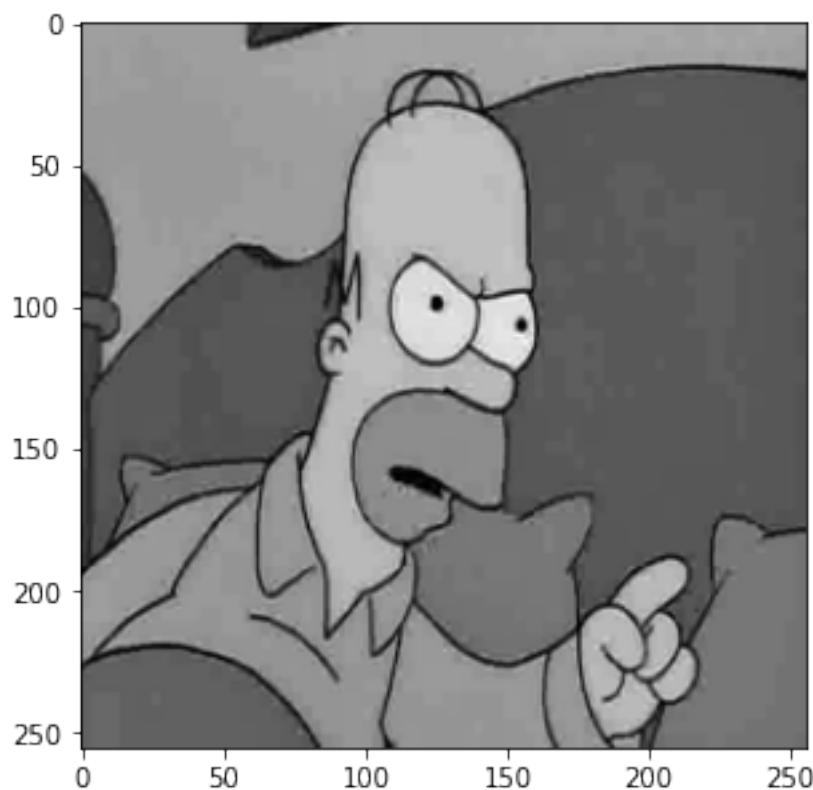
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 130. 235. 255.]  
 [255. 255. 255. ... 227. 255. 252.]  
 [255. 255. 255. ... 252. 255.]]
```



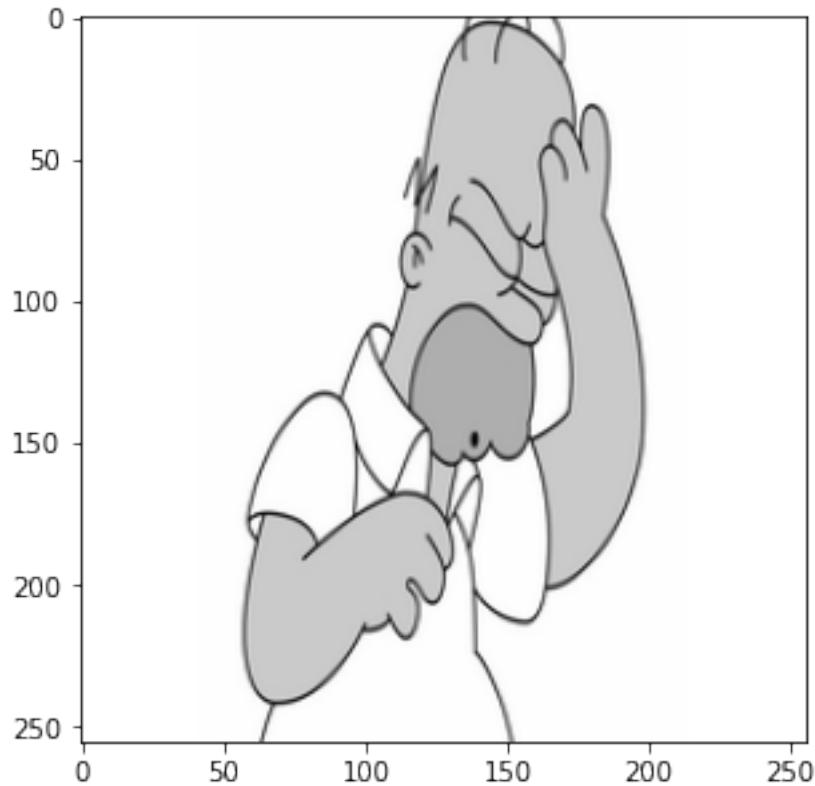
```
[[143.93333333 143.93333333 145.06666667 ... 200.6      207.4
 156.4      ]
 [143.93333333 143.93333333 145.06666667 ... 200.6      202.86666667
 189.26666667]
 [143.93333333 143.93333333 145.06666667 ... 207.4      210.8
 200.6      ]
 ...
 [147.33333333 148.46666667 147.33333333 ... 82.73333333 157.53333333
 230.06666667]
 [146.2      147.33333333 146.2      ... 80.46666667 185.86666667
 218.73333333]
 [146.2      146.2      147.33333333 ... 89.53333333 211.93333333
 215.33333333]]
```



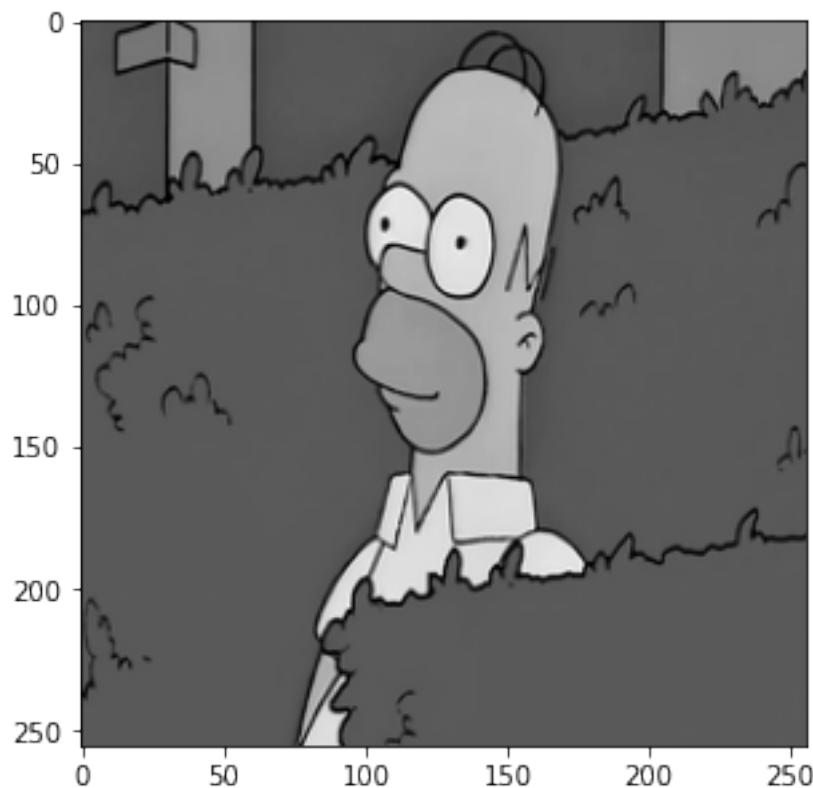
```
[[159.65217391 159.65217391 159.65217391 ... 164.08695652 164.08695652  
164.08695652]  
[159.65217391 159.65217391 159.65217391 ... 164.08695652 164.08695652  
164.08695652]  
[159.65217391 159.65217391 159.65217391 ... 164.08695652 164.08695652  
164.08695652]  
...  
[ 97.56521739 97.56521739 97.56521739 ... 109.76086957 109.76086957  
109.76086957]  
[ 97.56521739 97.56521739 97.56521739 ... 109.76086957 109.76086957  
109.76086957]  
[ 97.56521739 97.56521739 97.56521739 ... 109.76086957 109.76086957  
109.76086957]]
```



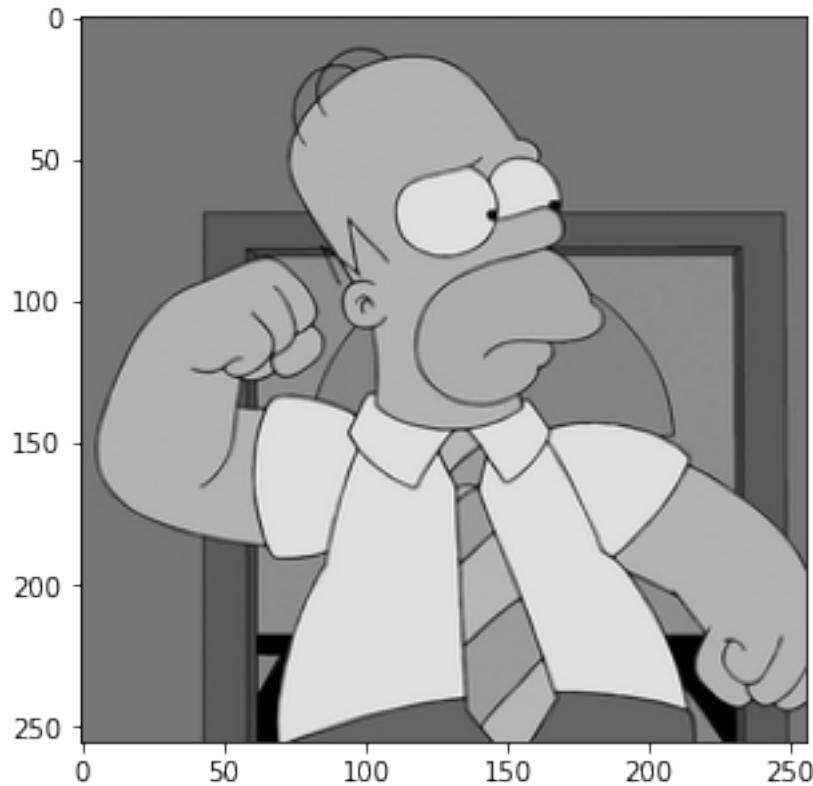
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



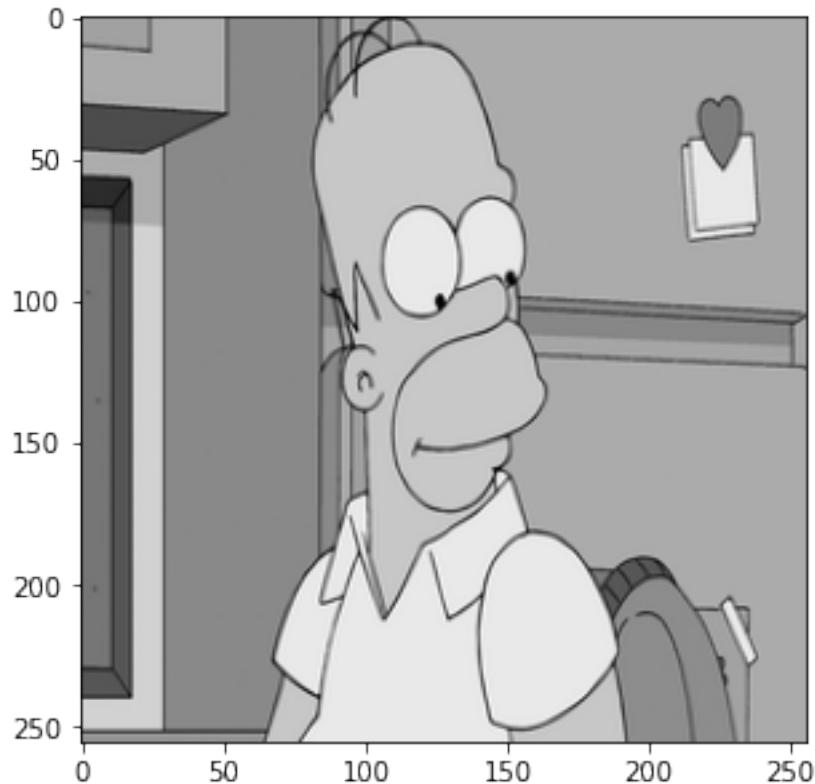
```
[[ 85.34693878  85.34693878  84.30612245 ... 138.42857143 138.42857143  
138.42857143]  
[ 85.34693878  85.34693878  84.30612245 ... 138.42857143 138.42857143  
138.42857143]  
[ 85.34693878  85.34693878  84.30612245 ... 138.42857143 138.42857143  
138.42857143]  
...  
[ 89.51020408  89.51020408  89.51020408 ... 89.51020408 89.51020408  
89.51020408]  
[ 89.51020408  89.51020408  89.51020408 ... 89.51020408 89.51020408  
89.51020408]  
[ 89.51020408  89.51020408  89.51020408 ... 89.51020408 89.51020408  
89.51020408]]
```



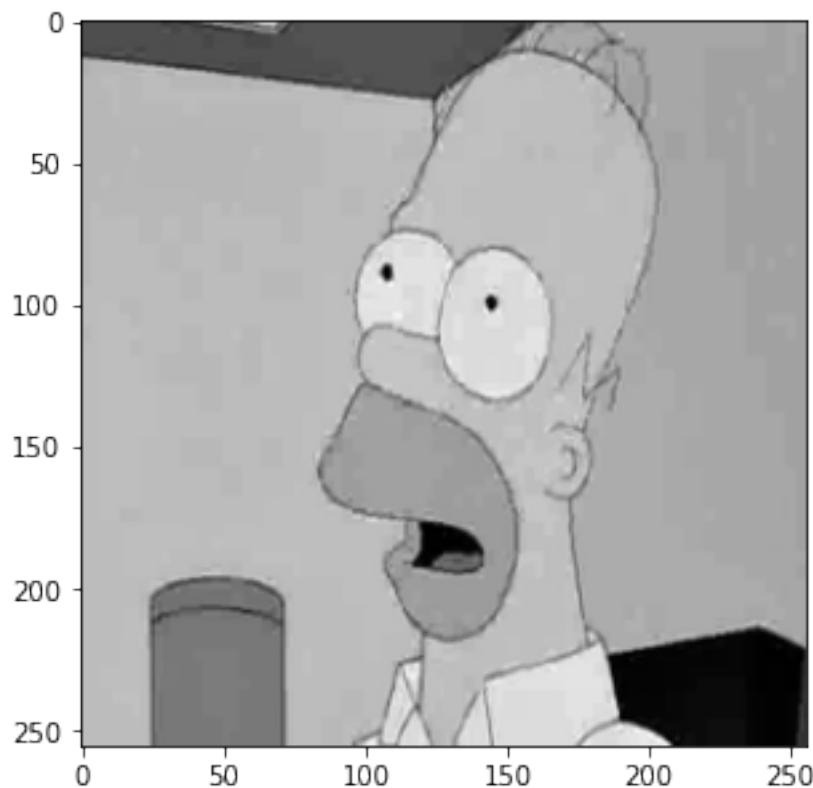
```
[[118. 117. 117. ... 118. 118. 118.]  
 [119. 119. 118. ... 119. 119. 119.]  
 [118. 119. 119. ... 118. 118. 118.]  
 ...  
 [119. 119. 119. ... 119. 119. 119.]  
 [119. 119. 119. ... 119. 119. 119.]  
 [119. 119. 119. ... 118. 120. 119.]]
```



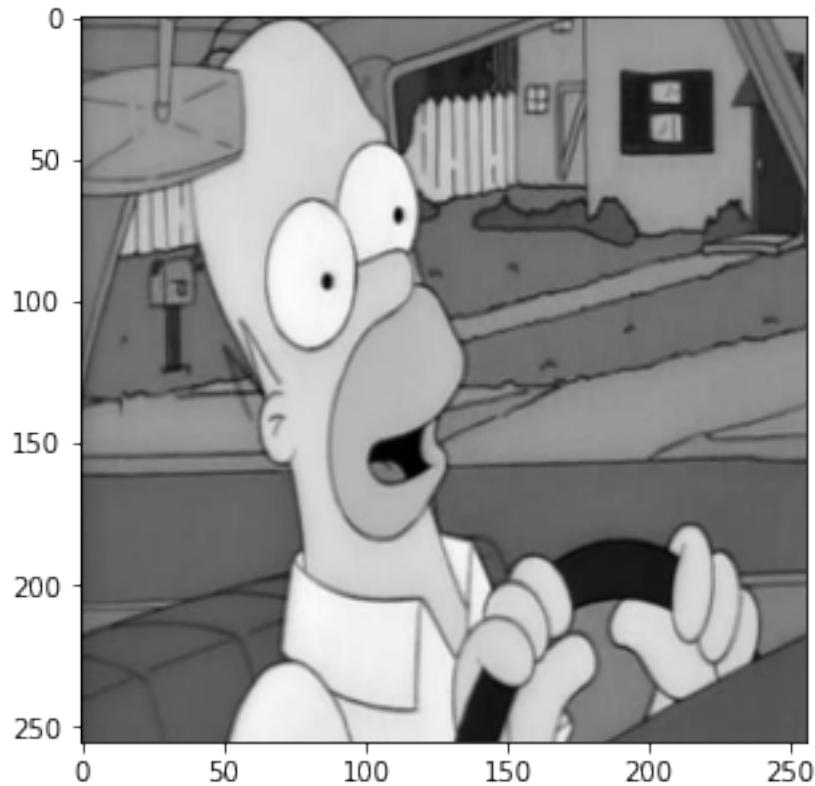
```
[[161. 161. 161. ... 172. 171. 171.]  
 [161. 161. 161. ... 172. 171. 171.]  
 [161. 161. 161. ... 171. 171. 171.]  
 ...  
 [160. 160. 159. ... 171. 171. 171.]  
 [146. 146. 146. ... 171. 171. 171.]  
 [149. 149. 149. ... 171. 172. 171.]]
```



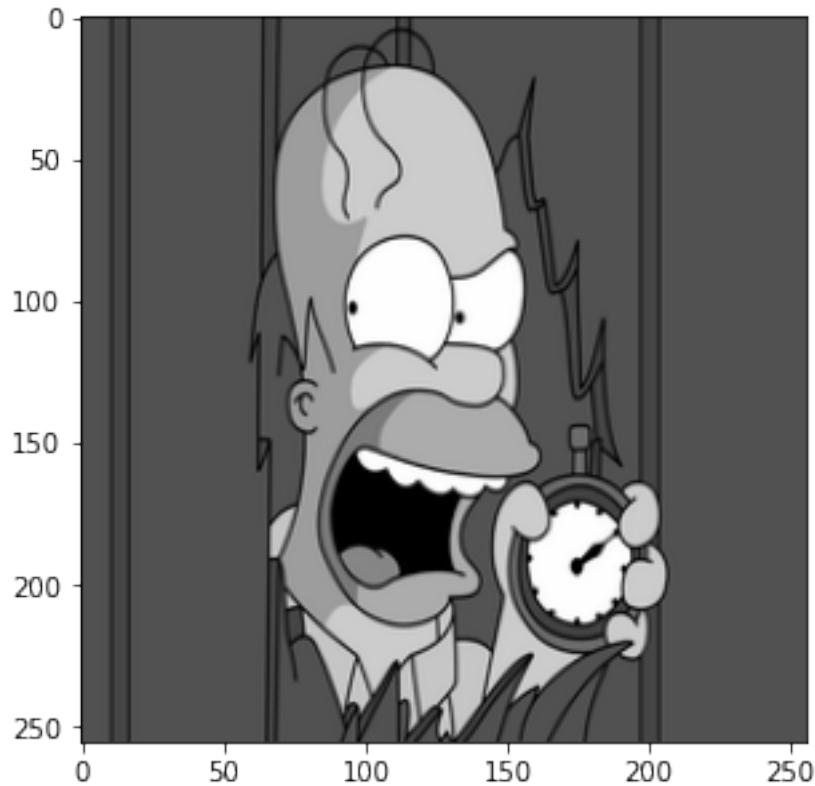
```
[[ 78.29875519  78.29875519  78.29875519 ... 167.17842324 167.17842324  
167.17842324]  
[ 78.29875519  78.29875519  78.29875519 ... 167.17842324 167.17842324  
167.17842324]  
[ 78.29875519  78.29875519  78.29875519 ... 167.17842324 167.17842324  
167.17842324]  
...  
[188.34024896 188.34024896 188.34024896 ... 55.02074689 50.78838174  
51.84647303]  
[188.34024896 188.34024896 188.34024896 ... 55.02074689 50.78838174  
51.84647303]  
[188.34024896 188.34024896 188.34024896 ... 55.02074689 50.78838174  
51.84647303]]
```



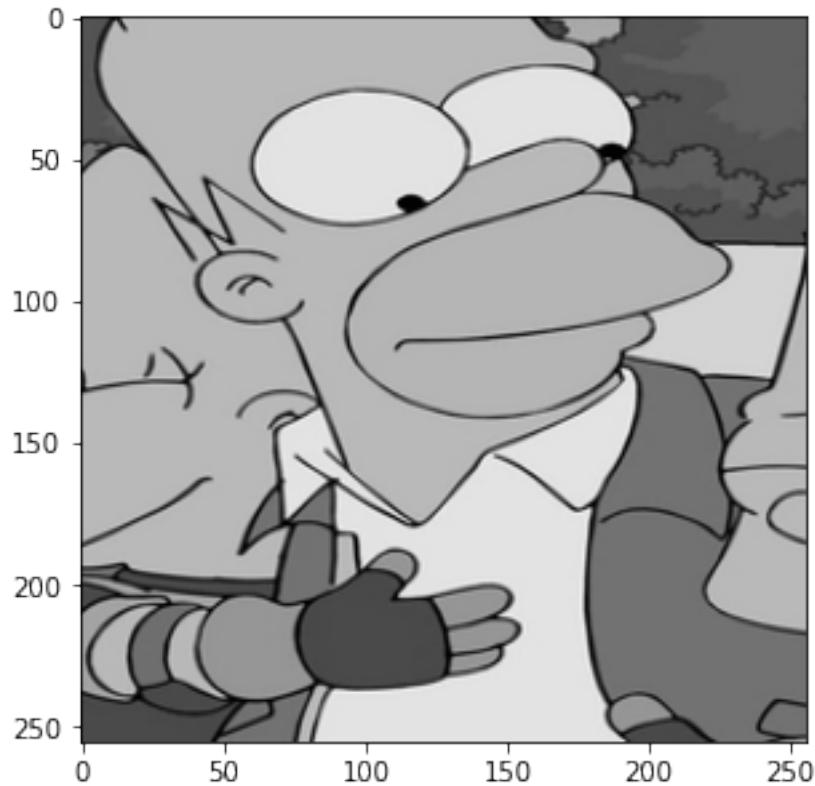
```
[[ 74.86238532 106.44495413 112.29357798 ... 152.06422018 153.23394495
154.40366972]
[ 69.01376147 100.59633028 106.44495413 ... 152.06422018 153.23394495
154.40366972]
[ 65.50458716 97.08715596 104.10550459 ... 152.06422018 153.23394495
153.23394495]
...
[ 49.12844037 71.35321101 81.88073394 ... 90.06880734 91.23853211
88.89908257]
[ 49.12844037 71.35321101 81.88073394 ... 90.06880734 91.23853211
88.89908257]
[ 49.12844037 71.35321101 81.88073394 ... 90.06880734 91.23853211
88.89908257]]
```



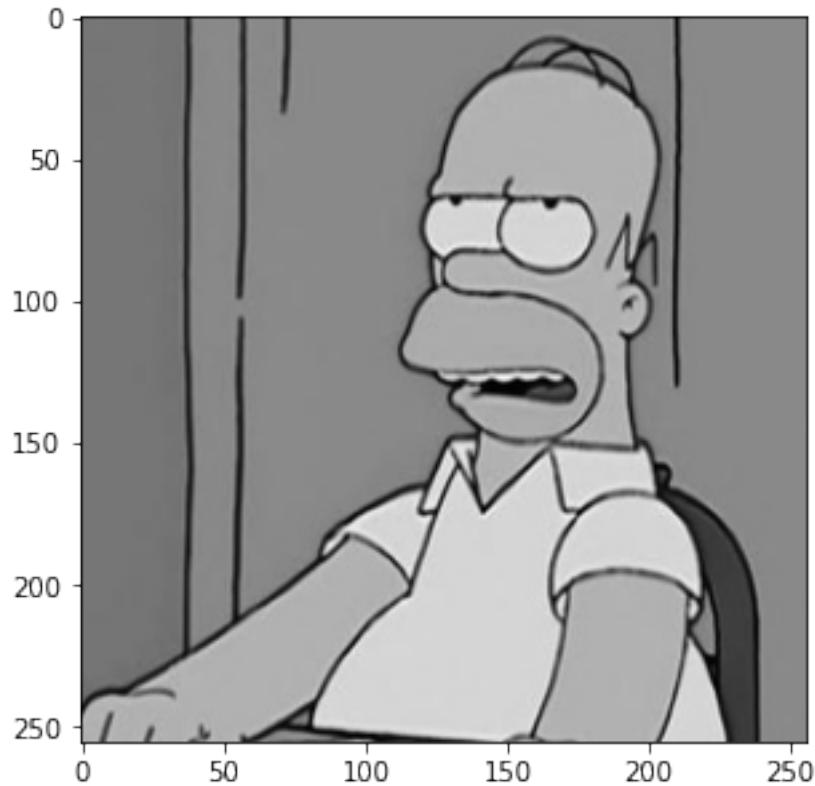
```
[[81. 81. 81. ... 81. 81. 81.]
 [81. 81. 81. ... 81. 81. 81.]
 [81. 81. 81. ... 81. 81. 81.]
 ...
 [81. 81. 81. ... 81. 81. 81.]
 [81. 81. 81. ... 81. 81. 81.]
 [81. 81. 81. ... 81. 81. 81.]]
```



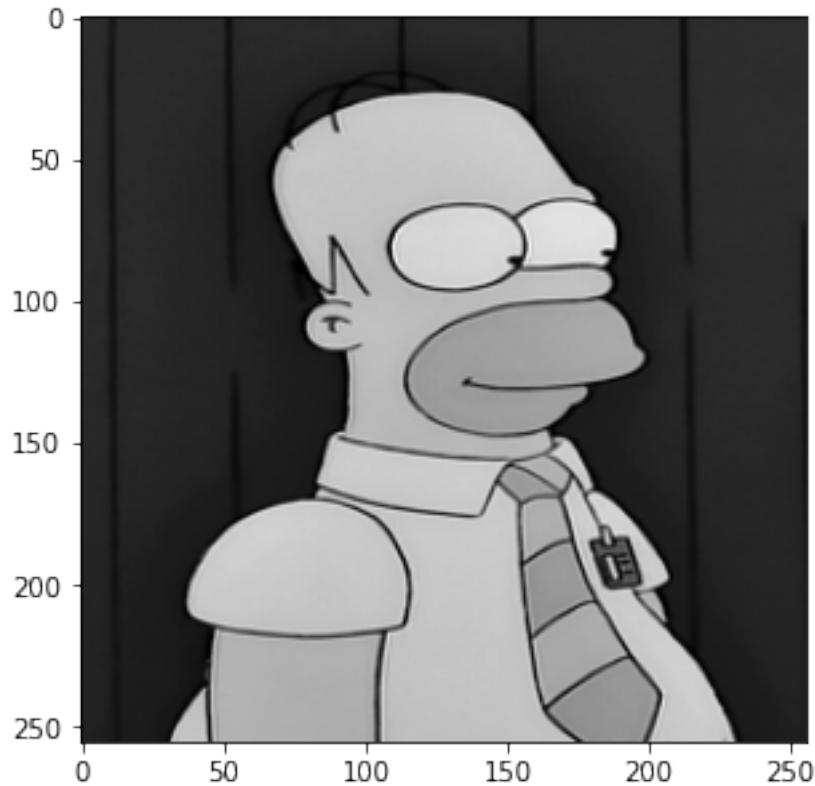
```
[[ 83.26530612  84.30612245  85.34693878 ...  95.75510204  95.75510204
  95.75510204]
 [ 83.26530612  84.30612245  85.34693878 ...  95.75510204  95.75510204
  95.75510204]
 [ 83.26530612  84.30612245  85.34693878 ...  95.75510204  95.75510204
  95.75510204]
 ...
 [ 74.93877551  74.93877551  74.93877551 ...  74.93877551  71.81632653
  66.6122449 ]
 [ 74.93877551  74.93877551  74.93877551 ...  56.20408163  67.65306122
  70.7755102 ]
 [ 74.93877551  74.93877551  74.93877551 ... 159.24489796 160.28571429
 160.28571429]]
```



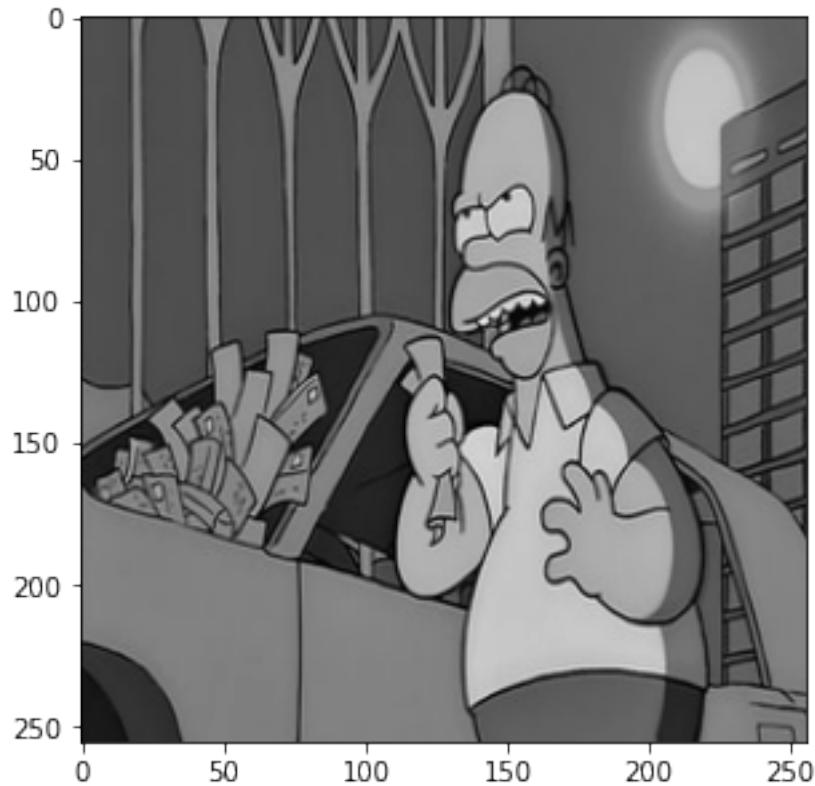
```
[[117.38095238 117.38095238 117.38095238 ... 137.61904762 137.61904762  
137.61904762]  
[117.38095238 117.38095238 117.38095238 ... 137.61904762 137.61904762  
137.61904762]  
[118.39285714 117.38095238 118.39285714 ... 137.61904762 137.61904762  
137.61904762]  
...  
[182.14285714 179.10714286 176.07142857 ... 137.61904762 137.61904762  
137.61904762]  
[179.10714286 182.14285714 173.03571429 ... 137.61904762 137.61904762  
137.61904762]  
[177.08333333 184.16666667 172.02380952 ... 137.61904762 137.61904762  
137.61904762]]
```



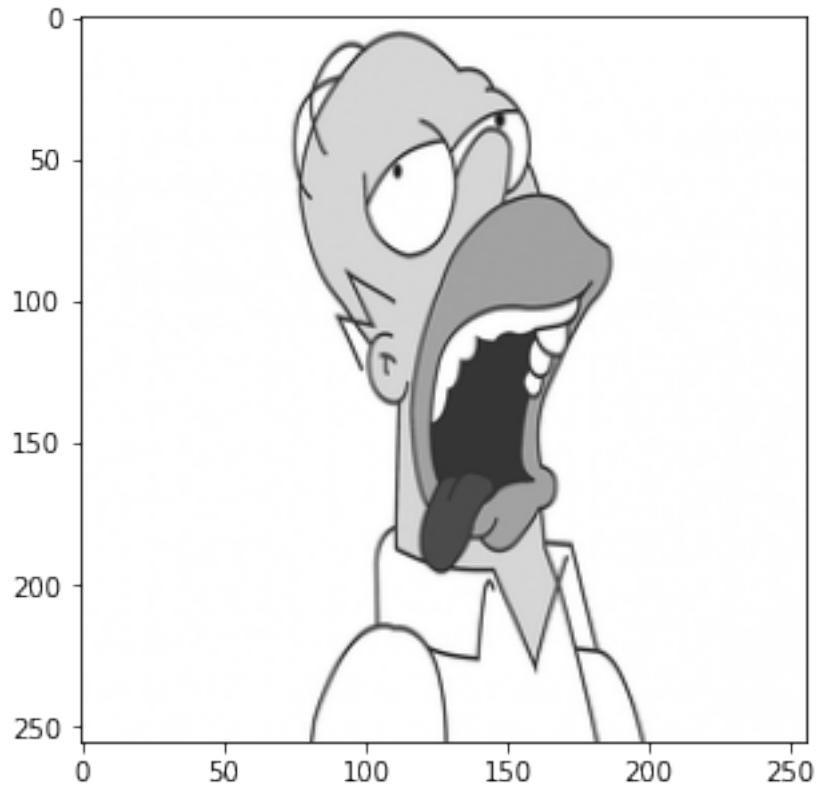
```
[[44.39732143 44.39732143 44.39732143 ... 45.53571429 45.53571429  
44.39732143]  
[45.53571429 46.67410714 45.53571429 ... 45.53571429 45.53571429  
44.39732143]  
[45.53571429 46.67410714 45.53571429 ... 46.67410714 45.53571429  
44.39732143]  
...  
[34.15178571 33.01339286 35.29017857 ... 30.73660714 33.01339286  
30.73660714]  
[35.29017857 34.15178571 34.15178571 ... 31.875 33.01339286  
30.73660714]  
[33.01339286 34.15178571 35.29017857 ... 33.01339286 33.01339286  
30.73660714]]
```



```
[[ 68.86075949  80.69620253  79.62025316 ...  96.83544304  96.83544304
  96.83544304]
 [ 68.86075949  80.69620253  79.62025316 ...  96.83544304  96.83544304
  96.83544304]
 [ 68.86075949  80.69620253  79.62025316 ...  96.83544304  96.83544304
  96.83544304]
 ...
 [ 21.51898734  27.97468354  25.82278481 ... 126.96202532 124.81012658
 124.81012658]
 [ 21.51898734  27.97468354  25.82278481 ... 128.03797468 125.88607595
 124.81012658]
 [ 21.51898734  27.97468354  25.82278481 ... 130.18987342 125.88607595
 124.81012658]]
```



```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[254. 255. 255. ... 255. 255. 255.]  
[254. 255. 255. ... 255. 255. 255.]  
[254. 255. 254. ... 255. 255. 255.]]
```



[[31. 33. 28. ... 32. 32. 32.]

[30. 29. 24. ... 34. 34. 34.]

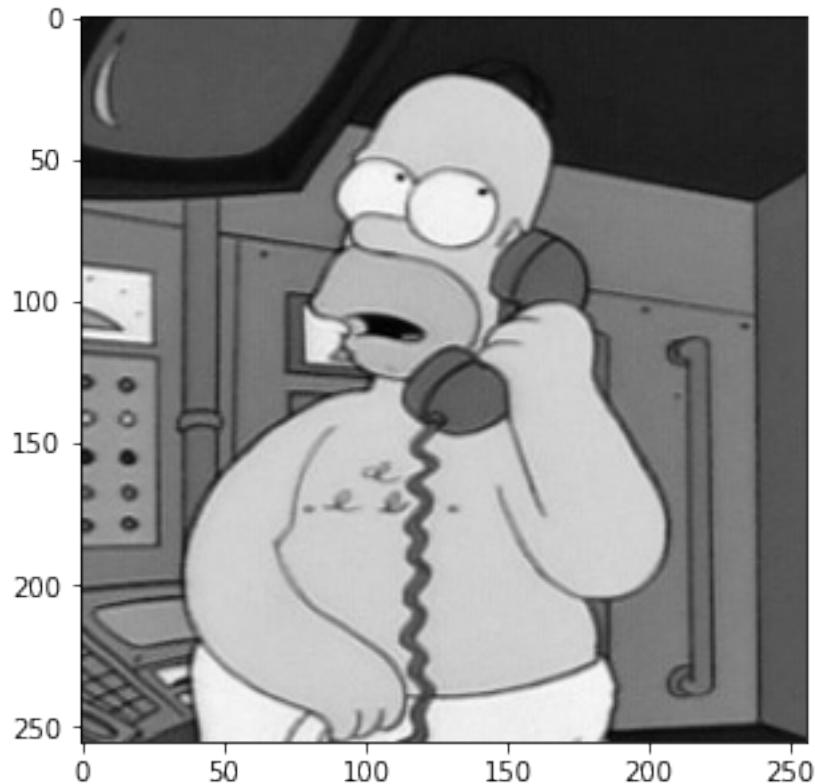
[28. 22. 17. ... 33. 33. 33.]

...

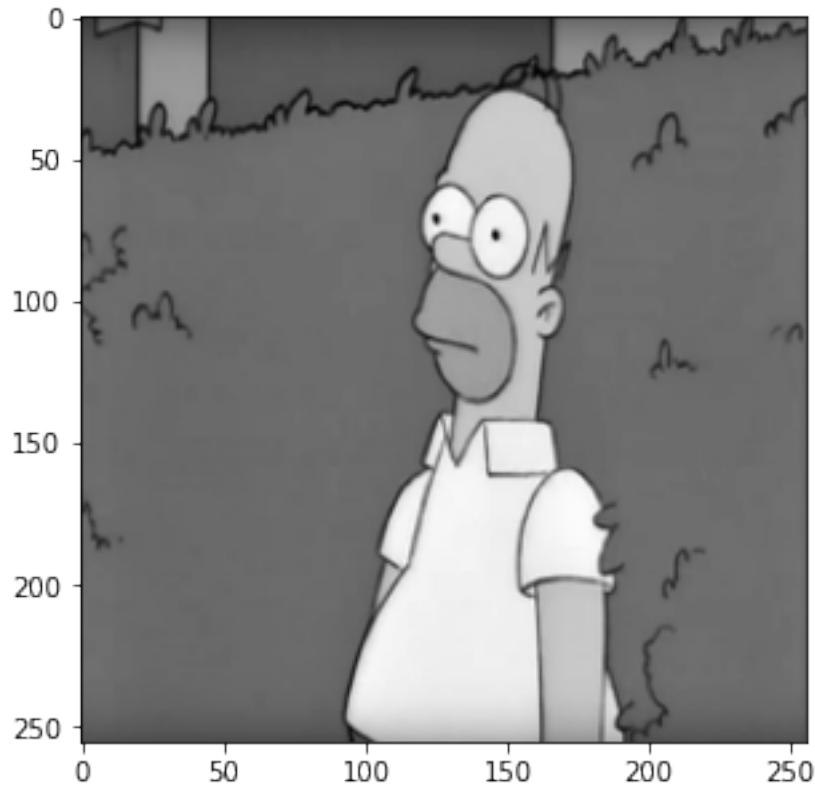
[51. 70. 86. ... 54. 67. 80.]

[69. 63. 61. ... 26. 29. 35.]

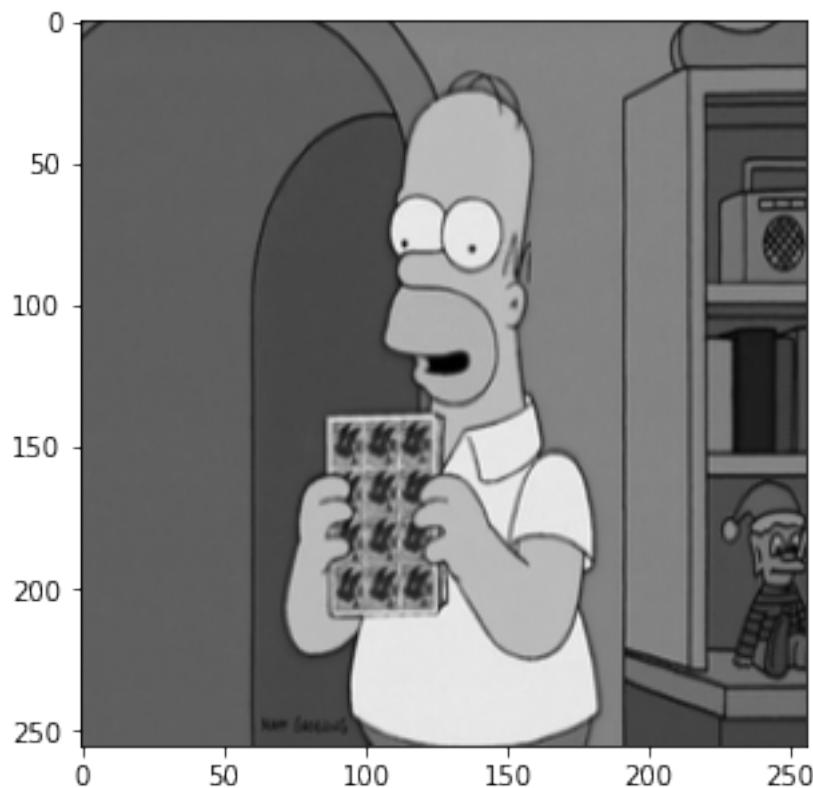
[89. 79. 73. ... 21. 22. 20.]]



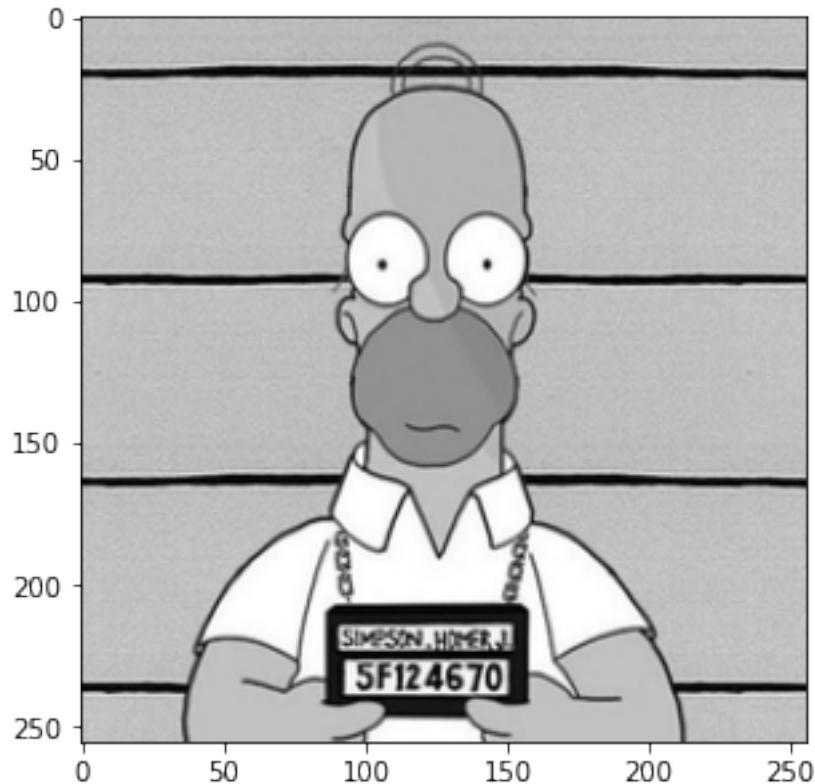
```
[[ 59.93589744  59.93589744  62.11538462 ... 101.34615385 102.43589744
  98.07692308]
 [ 62.11538462  63.20512821  64.29487179 ... 104.61538462 104.61538462
 100.25641026]
 [ 65.38461538  65.38461538  66.47435897 ... 108.97435897 112.24358974
 103.52564103]
...
[ 85.          87.17948718  87.17948718 ...  89.35897436  88.26923077
 81.73076923]
[ 82.82051282  85.          85.          ...  86.08974359  85.
 79.55128205]
[ 79.55128205  81.73076923  81.73076923 ...  85.          83.91025641
 77.37179487]]
```



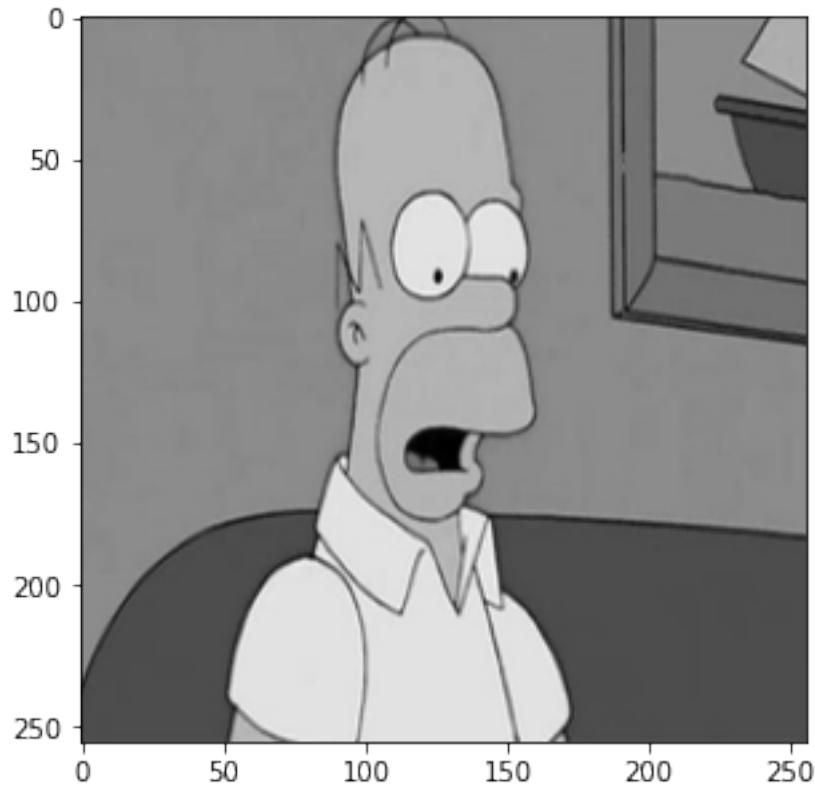
```
[[ 48.21428571  41.78571429 103.92857143 ... 109.28571429 110.35714286
  111.42857143]
 [ 50.35714286  95.35714286 113.57142857 ... 110.35714286 110.35714286
  109.28571429]
 [107.14285714 113.57142857 116.78571429 ... 111.42857143 110.35714286
  109.28571429]
 ...
 [ 96.42857143  96.42857143  96.42857143 ...   69.64285714  69.64285714
  69.64285714]
 [ 97.5          96.42857143  96.42857143 ...   69.64285714  69.64285714
  68.57142857]
 [ 95.35714286  97.5          96.42857143 ...   69.64285714  68.57142857
  67.5          ]]
```



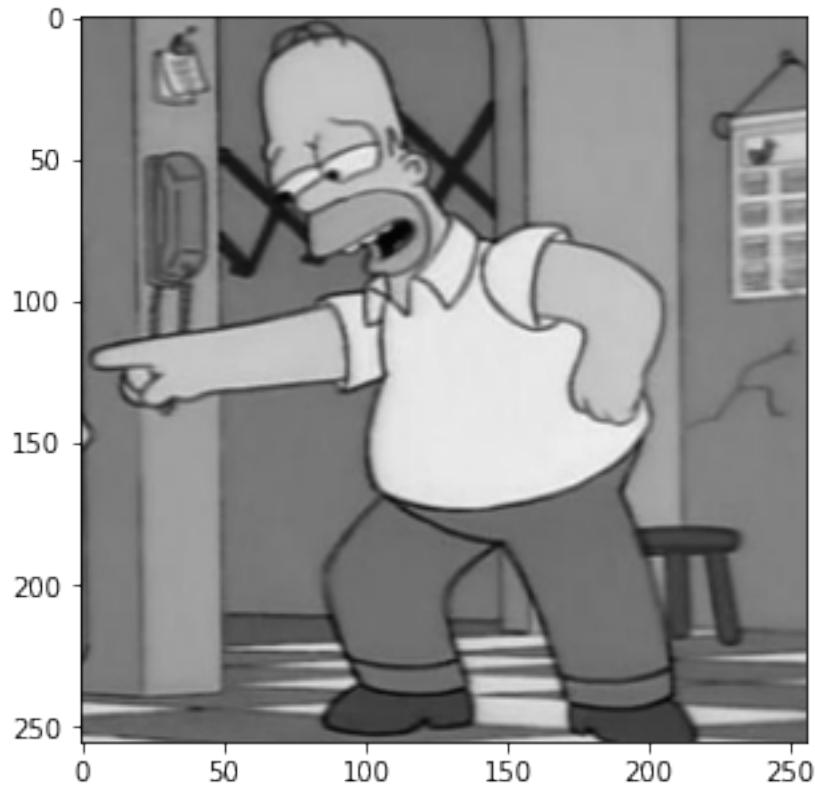
```
[[251. 251. 251. ... 249. 251. 251.]  
 [191. 195. 192. ... 192. 194. 195.]  
 [183. 185. 189. ... 188. 188. 190.]  
 ...  
 [195. 201. 198. ... 196. 202. 195.]  
 [198. 189. 194. ... 202. 191. 195.]  
 [190. 199. 192. ... 199. 189. 192.]]
```



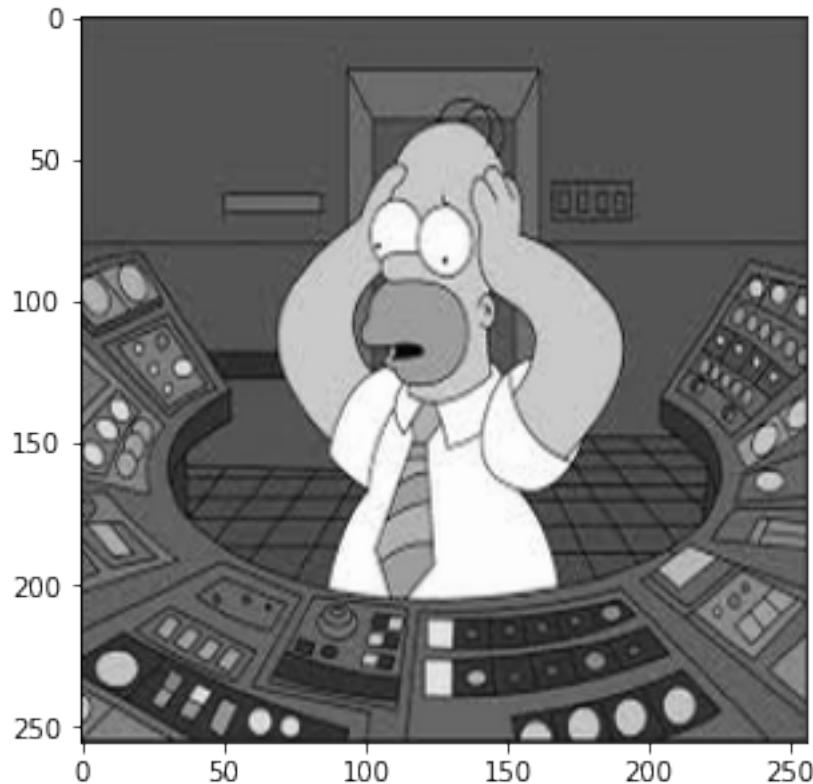
```
[[144.95850622 143.90041494 142.84232365 ... 177.7593361 176.70124481  
176.70124481]  
[138.60995851 137.55186722 136.49377593 ... 173.52697095 171.41078838  
171.41078838]  
[141.78423237 140.72614108 139.66804979 ... 174.58506224 172.46887967  
172.46887967]  
...  
[ 78.29875519 77.2406639 77.2406639 ... 77.2406639 77.2406639  
77.2406639 ]  
[ 76.18257261 76.18257261 76.18257261 ... 76.18257261 76.18257261  
76.18257261]  
[ 82.53112033 82.53112033 82.53112033 ... 82.53112033 82.53112033  
82.53112033]]
```



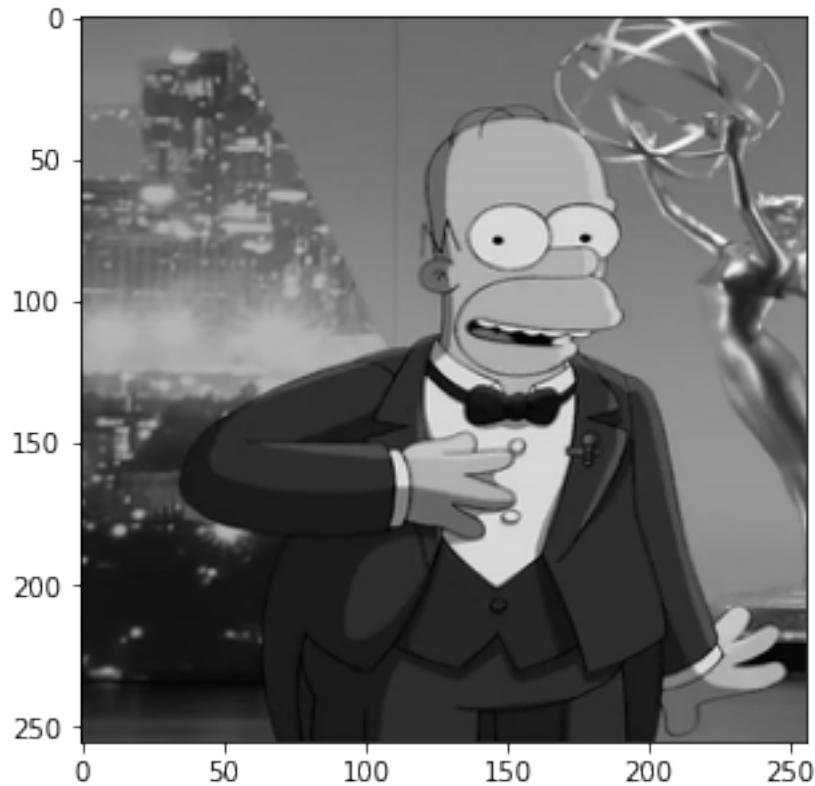
```
[[112.2 117.3 122.4 ... 125.46 125.46 126.48]
 [113.22 117.3 122.4 ... 125.46 125.46 126.48]
 [113.22 118.32 122.4 ... 125.46 125.46 126.48]
 ...
 [125.46 119.34 115.26 ... 104.04 104.04 104.04]
 [211.14 209.1 207.06 ... 100.98 102. 104.04]
 [211.14 210.12 211.14 ... 98.94 96.9 98.94]]
```



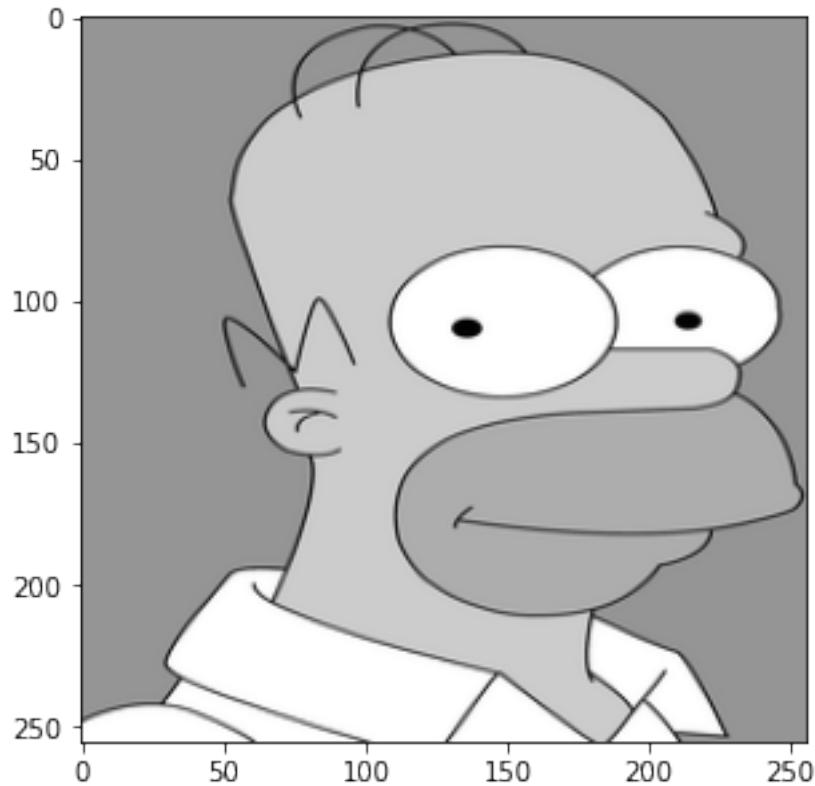
```
[[ 31.  58.  53. ... 54.  51.  48.]  
 [ 49.  76.  71. ... 73.  70.  66.]  
 [ 64.  92.  89. ... 88.  85.  81.]  
 ...  
 [ 80. 111. 103. ... 109. 105.  98.]  
 [ 43.  66.  97. ... 78.  77.  74.]  
 [ 13.  36.  54. ... 55.  53.  48.]]
```



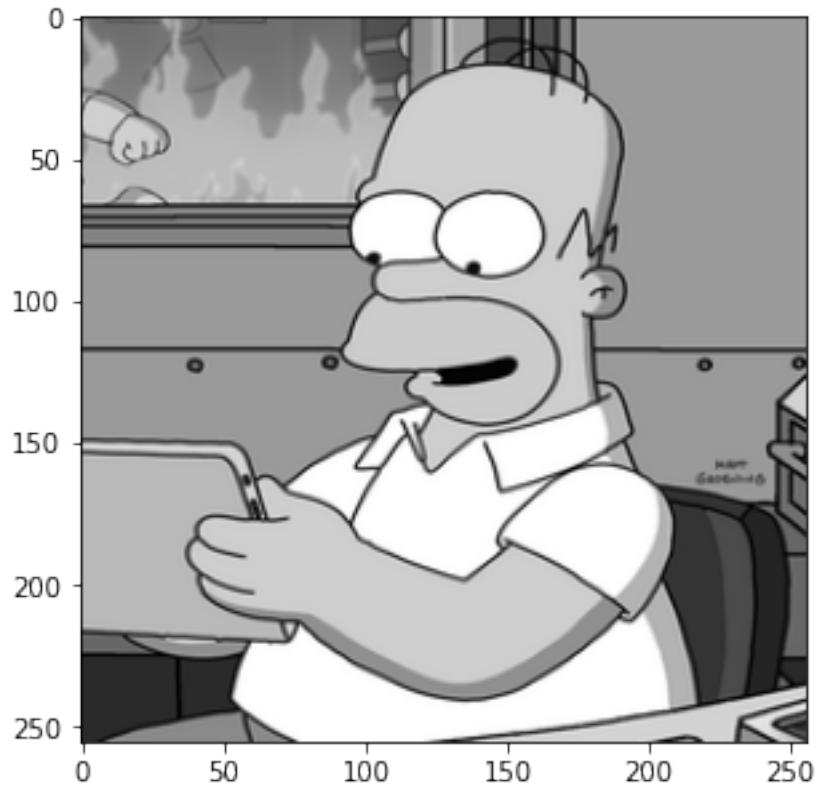
```
[[116.83266932 117.84860558 117.84860558 ... 109.72111554 109.72111554  
109.72111554]  
[114.80079681 115.81673307 115.81673307 ... 107.68924303 107.68924303  
107.68924303]  
[115.81673307 116.83266932 116.83266932 ... 107.68924303 107.68924303  
107.68924303]  
...  
[ 55.87649402 54.86055777 53.84462151 ... 101.5936255 101.5936255  
100.57768924]  
[ 57.90836653 57.90836653 57.90836653 ... 100.57768924 101.5936255  
101.5936255 ]  
[ 64.00398406 64.00398406 64.00398406 ... 102.60956175 102.60956175  
101.5936255 ]]
```



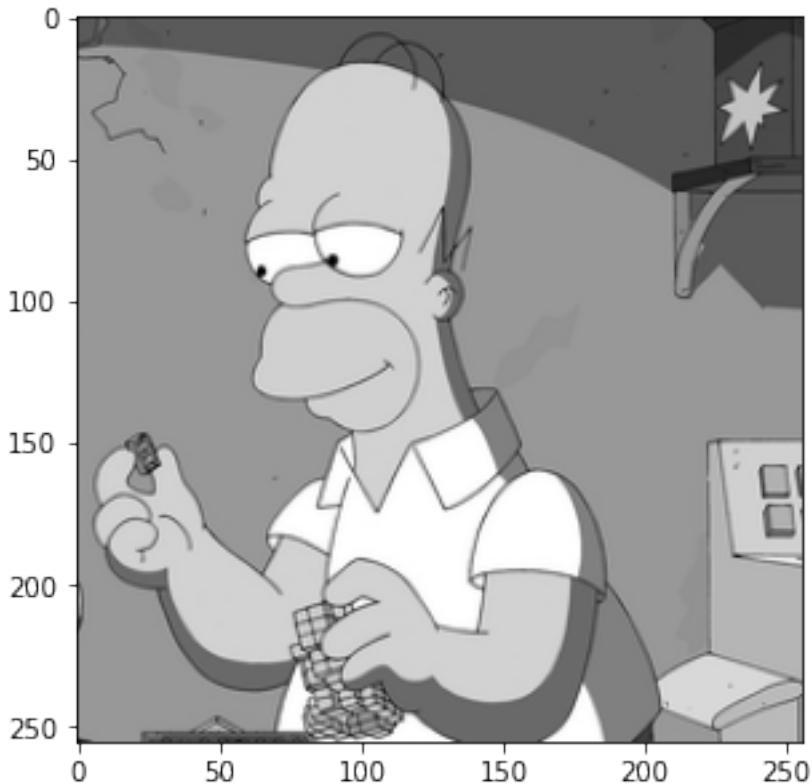
```
[[149. 149. 149. ... 149. 149. 149.]  
 [149. 149. 149. ... 149. 149. 149.]  
 [149. 149. 149. ... 149. 149. 149.]  
 ...  
 [255. 255. 255. ... 149. 149. 149.]  
 [255. 255. 255. ... 149. 149. 149.]  
 [255. 255. 255. ... 149. 149. 149.]]
```



```
[[108. 112. 114. ... 155. 155. 155.]  
 [111. 123. 128. ... 155. 155. 155.]  
 [117. 137. 138. ... 155. 155. 155.]  
 ...  
 [ 42.  42.  42. ... 52.  79.  82.]  
 [ 42.  42.  42. ... 56.  37.  54.]  
 [ 42.  42.  42. ... 177. 126.  58.]]
```



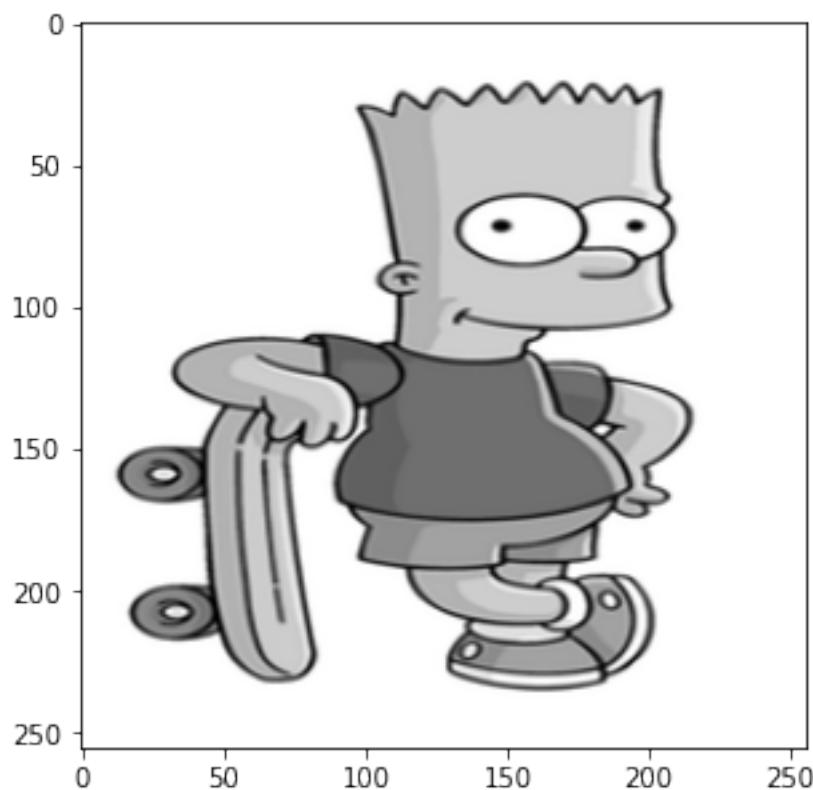
```
[[ 85.  85.  90. ... 53.  50.  50.]  
 [ 84.  87.  90. ... 54.  52.  53.]  
 [ 87.  90.  91. ... 53.  51.  41.]  
 ...  
 [163. 160. 154. ... 198. 125. 150.]  
 [ 98. 124. 146. ... 198. 129. 151.]  
 [ 34.  24.  43. ... 198. 131. 152.]]
```



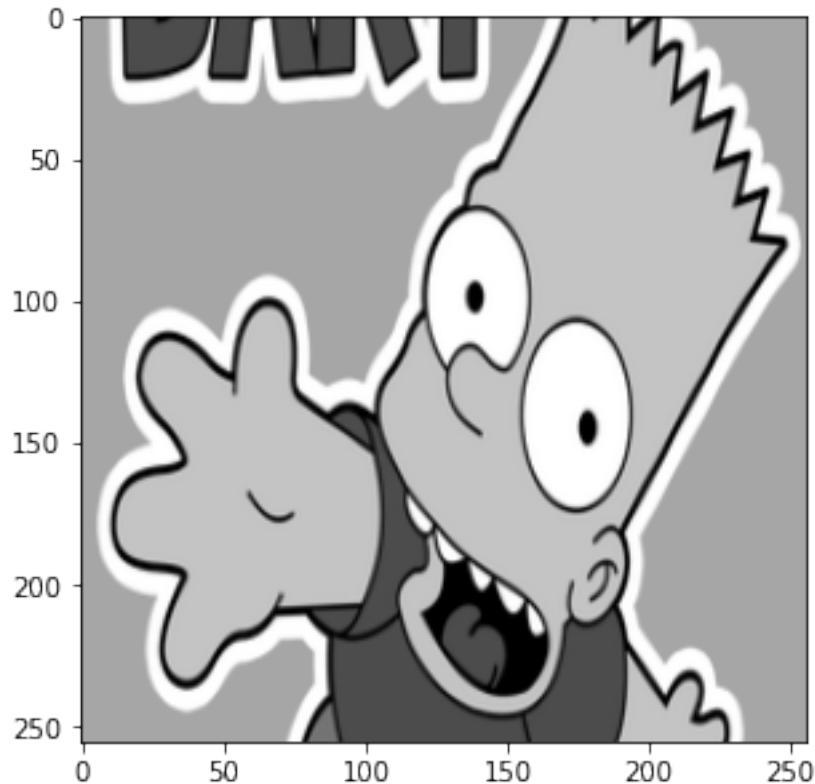
```
[ ]: plt.rcParams['figure.figsize'] = [5, 5]

for index, image in enumerate(grayScaleBart):
    image = np.array(image)
    norm = image/(image.max()/255)
    print(norm)
    #convert back to image
    im = Image.fromarray(norm)
    im = im.convert('RGB')
    im.save("normalizedBart/image-"+str(index)+".jpg")
    plt.imshow(im, cmap=plt.cm.gray)
    plt.show()
```

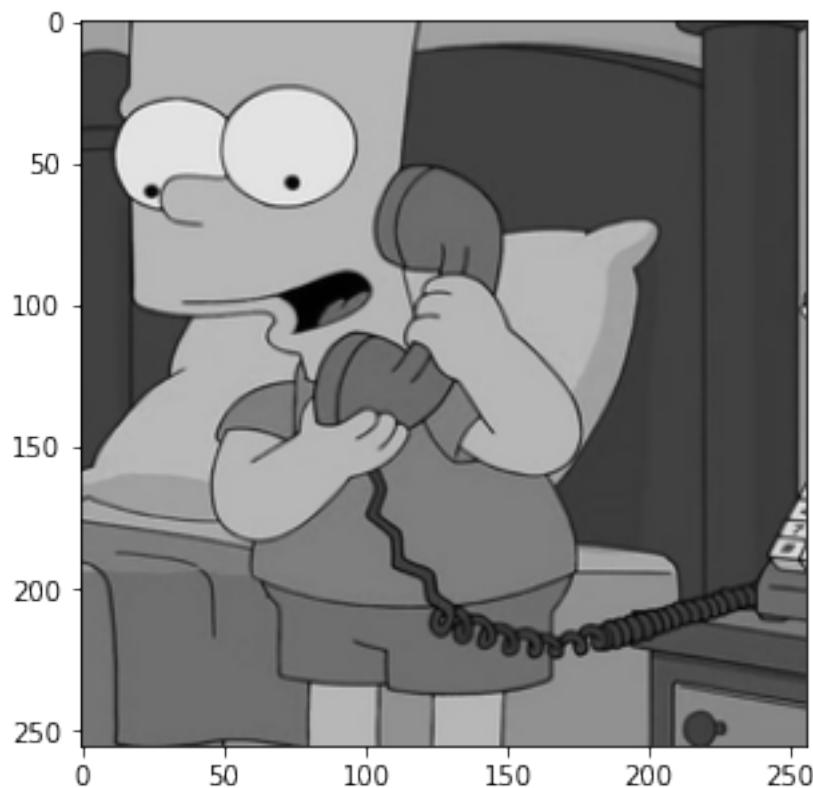
```
[[255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 ...
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]]
```



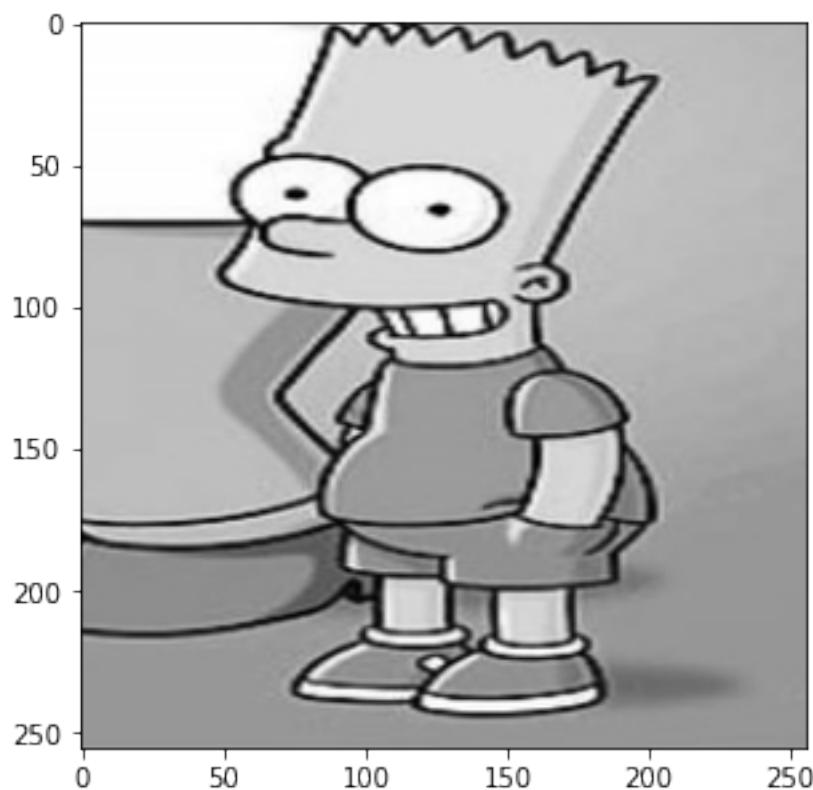
```
[[166. 166. 166. ... 166. 166. 166.]  
 [166. 166. 166. ... 166. 166. 166.]  
 [166. 166. 166. ... 166. 166. 166.]  
 ...  
 [166. 166. 166. ... 166. 166. 166.]  
 [166. 166. 166. ... 166. 166. 166.]  
 [166. 166. 166. ... 166. 166. 166.]]
```



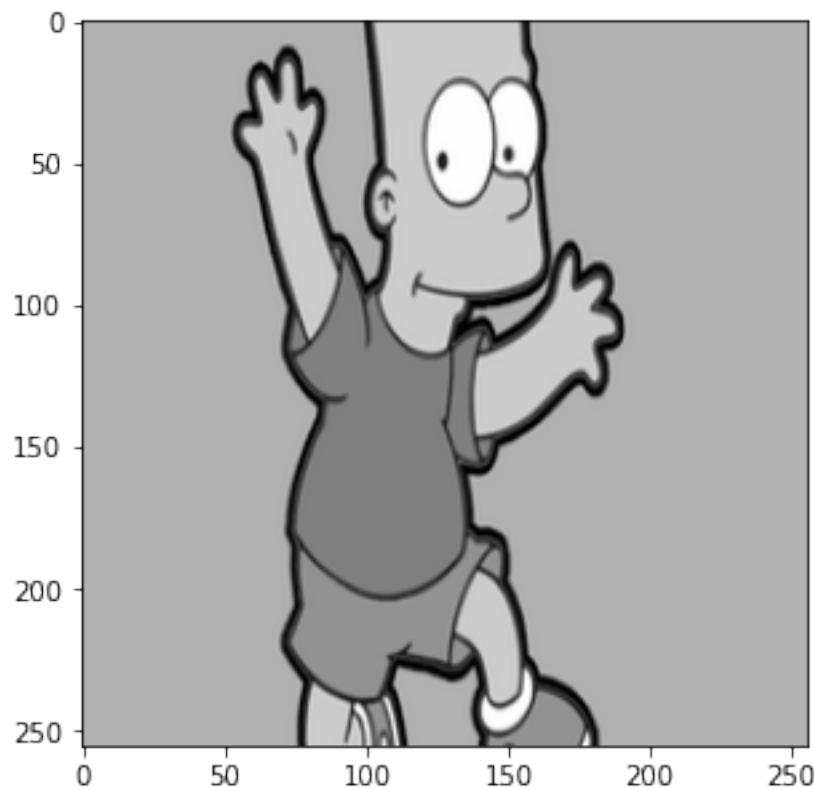
```
[[160.62992126 160.62992126 160.62992126 ... 61.24015748 60.23622047  
58.22834646]  
[160.62992126 160.62992126 160.62992126 ... 63.2480315 65.25590551  
60.23622047]  
[160.62992126 160.62992126 160.62992126 ... 53.20866142 56.22047244  
37.14566929]  
...  
[116.45669291 116.45669291 117.46062992 ... 134.52755906 134.52755906  
134.52755906]  
[116.45669291 116.45669291 117.46062992 ... 134.52755906 134.52755906  
134.52755906]  
[116.45669291 116.45669291 117.46062992 ... 134.52755906 134.52755906  
134.52755906]]
```



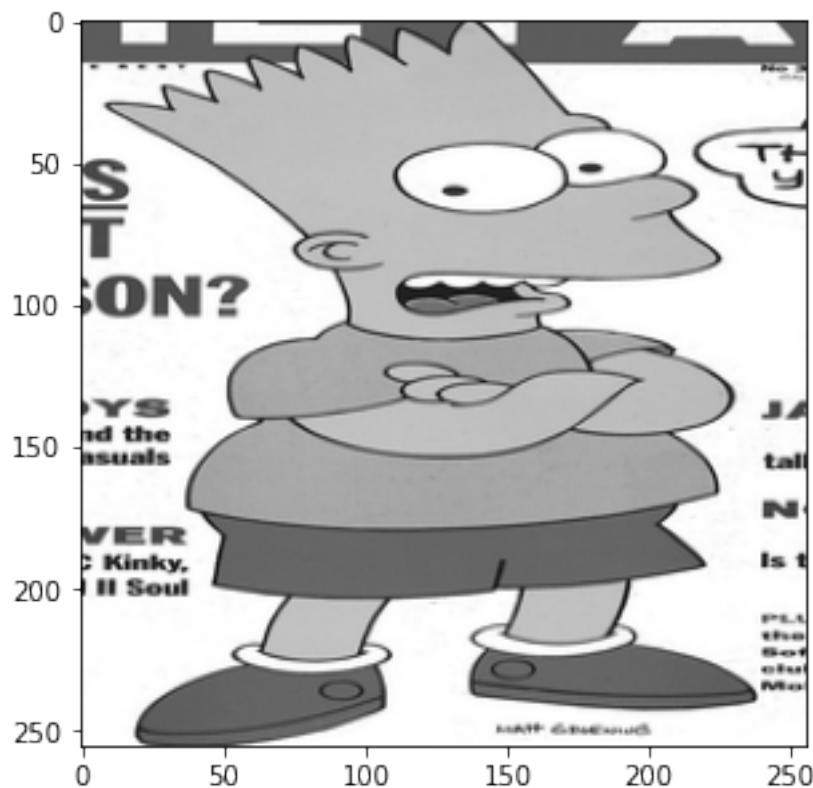
```
[[254. 254. 254. ... 180. 181. 180.]  
 [255. 255. 255. ... 180. 181. 181.]  
 [255. 255. 255. ... 181. 182. 182.]  
 ...  
 [151. 151. 151. ... 148. 148. 148.]  
 [150. 150. 150. ... 148. 148. 149.]  
 [158. 157. 158. ... 152. 152. 153.]]
```



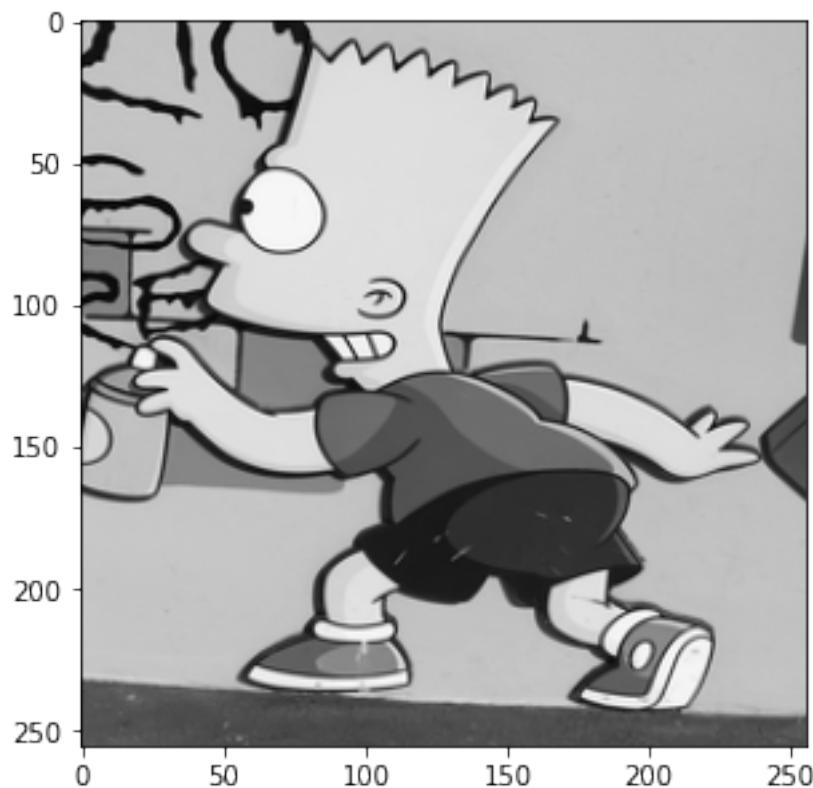
```
[[177. 177. 177. ... 177. 177. 177.]  
 [177. 177. 177. ... 177. 177. 177.]  
 [177. 177. 177. ... 177. 177. 177.]  
 ...  
 [177. 177. 177. ... 177. 177. 177.]  
 [177. 177. 177. ... 177. 177. 177.]  
 [177. 177. 177. ... 177. 177. 177.]]
```



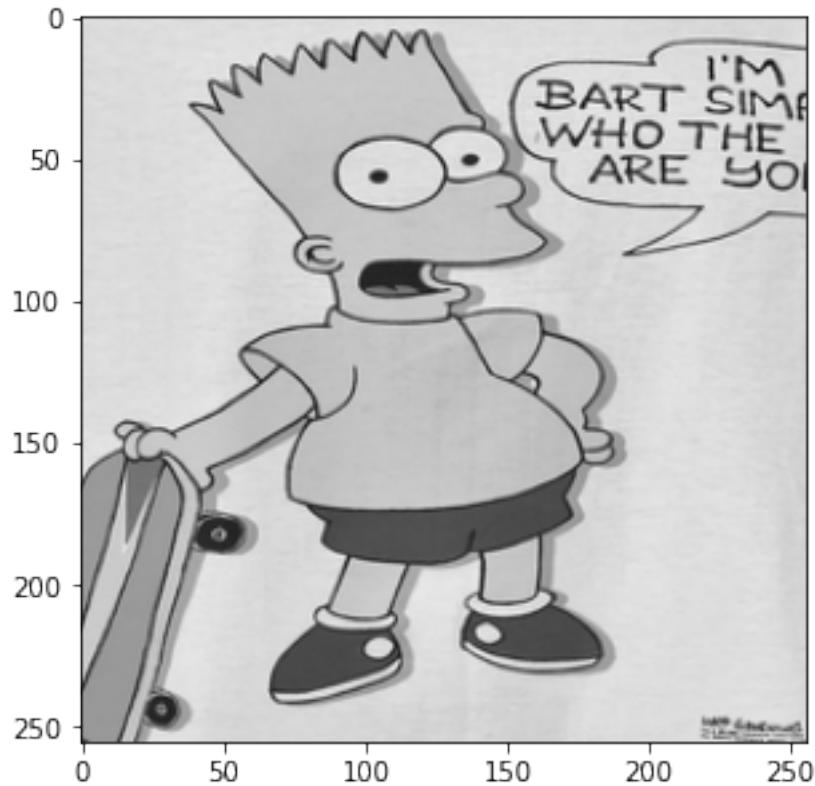
```
[[ 92.  92.  92. ... 92.  92.  91.]  
 [ 92.  92.  92. ... 91.  92.  92.]  
 [ 92.  92.  92. ... 91.  92.  92.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



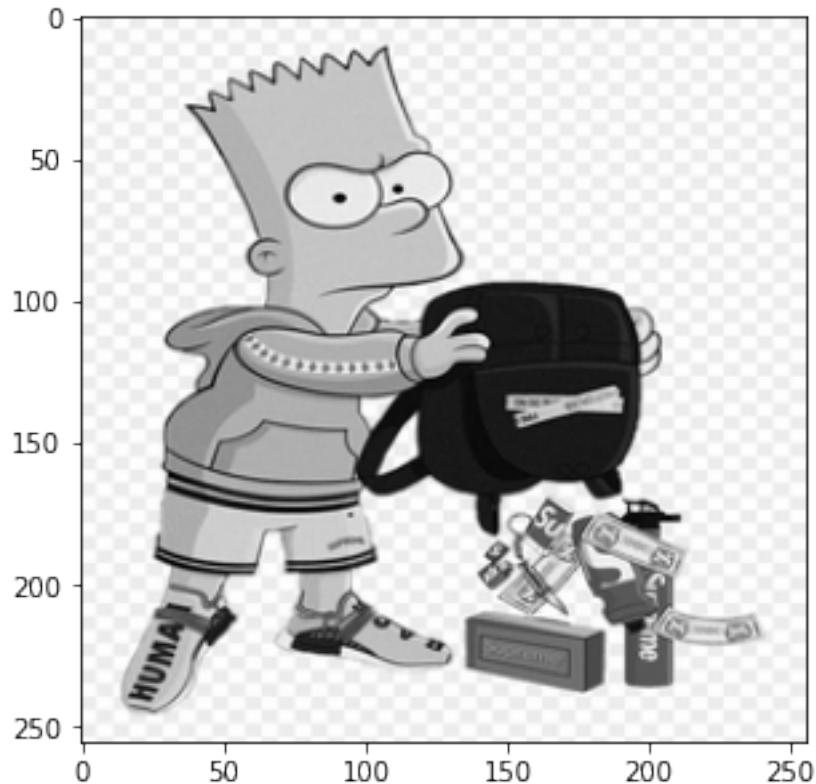
```
[[ 7.  46. 141. ... 192. 190. 189.]  
[ 95. 184. 202. ... 190. 190. 189.]  
[200. 196. 189. ... 190. 189. 189.]  
..  
[ 86.  86.  86. ...  85.  83.  86.]  
[ 94.  86.  86. ...  77.  77.  78.]  
[ 82.  84.  86. ...  78.  77.  80.]]
```



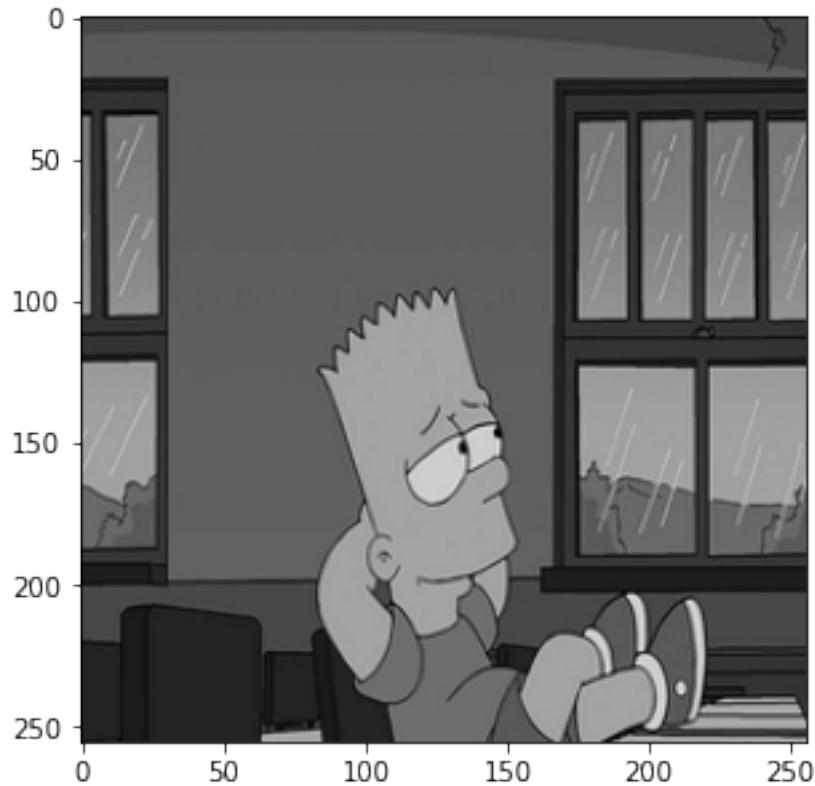
```
[[222. 221. 221. ... 216. 219. 219.]  
 [217. 221. 221. ... 218. 216. 219.]  
 [217. 221. 223. ... 219. 222. 225.]  
 ...  
 [158. 156. 152. ... 149. 152. 214.]  
 [ 88. 134. 156. ... 176. 178. 222.]  
 [136. 69. 94. ... 181. 154. 152.]]
```



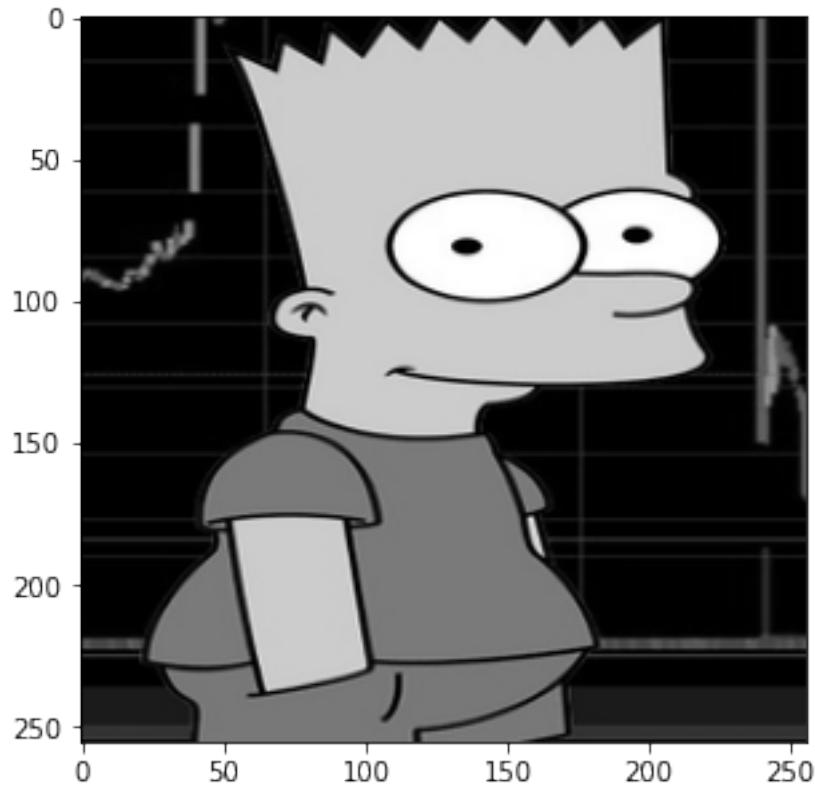
[[254. 254. 254. ... 237. 237. 237.]
[254. 254. 254. ... 237. 237. 237.]
[254. 254. 254. ... 237. 237. 237.]
...
[237. 237. 237. ... 254. 254. 254.]
[252. 252. 252. ... 240. 240. 240.]
[255. 255. 255. ... 237. 237. 237.]]



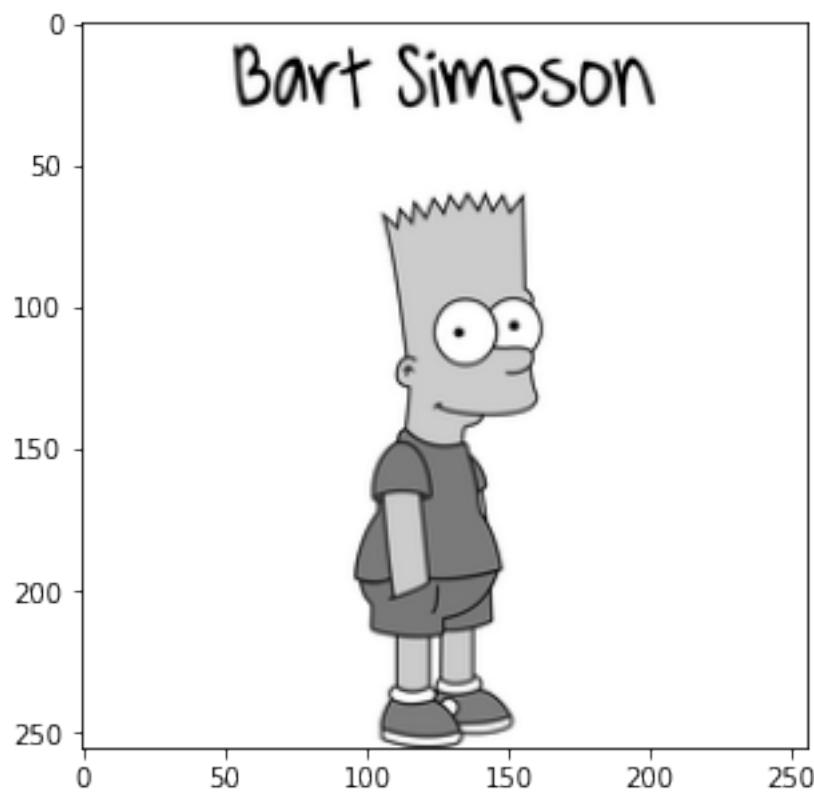
```
[[ 72.  72.  72. ... 72.  72.  72.]  
 [ 72.  72.  72. ... 72.  72.  72.]  
 [ 72.  72.  72. ... 72.  72.  72.]  
 ...  
 [ 34.  34.  35. ... 29.  29.  29.]  
 [ 35.  37.  37. ... 115. 115. 114.]  
 [ 23.  21.  20. ... 183. 184. 185.]]
```



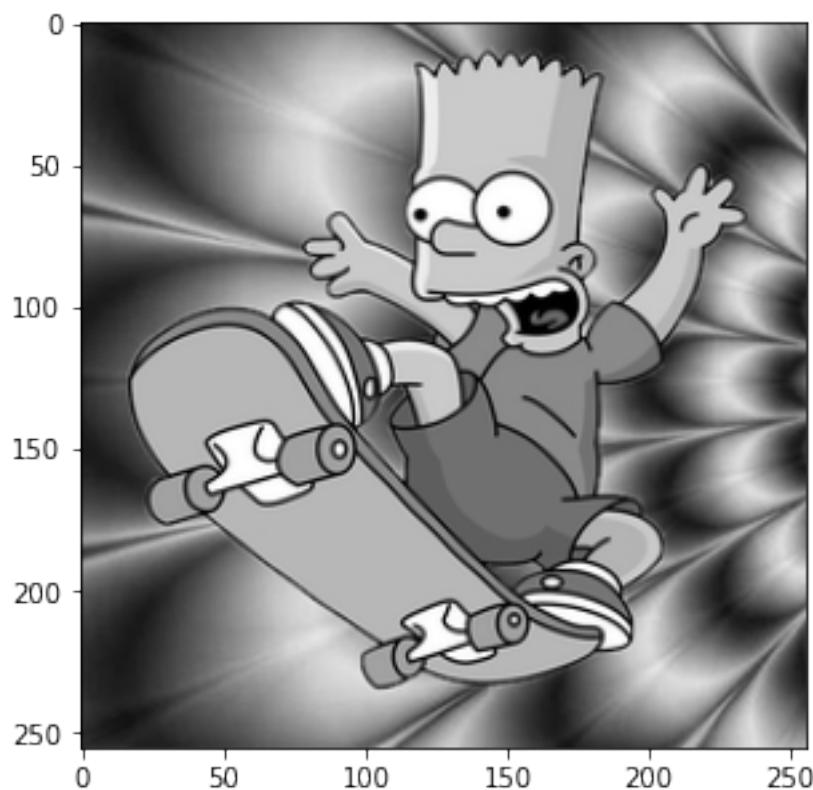
```
[[ 0.  0.  0. ... 1.  1.  0.]  
 [ 0.  0.  0. ... 1.  1.  0.]  
 [ 0.  0.  0. ... 1.  1.  0.]  
 ...  
 [50. 50. 50. ... 50. 50. 50.]  
 [61. 61. 61. ... 61. 61. 61.]  
 [12. 12. 12. ... 12. 12. 12.]]
```



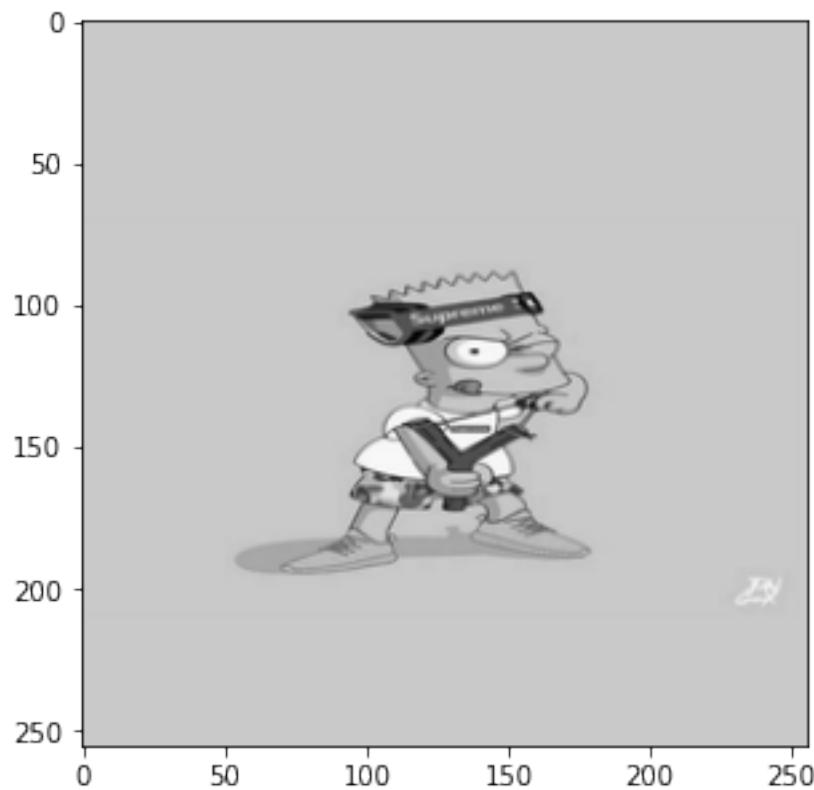
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



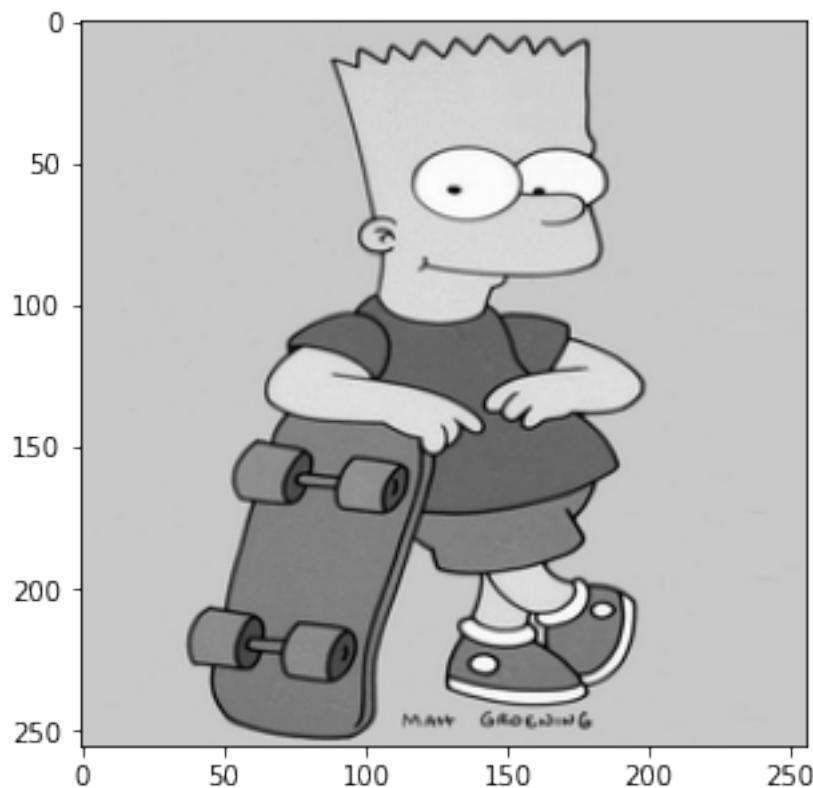
```
[[ 50.  49.  48. ... 151. 151. 152.]  
 [ 43.  38.  39. ... 147. 143. 143.]  
 [ 49.  40.  38. ... 138. 138. 136.]  
 ...  
 [ 56.  49.  44. ... 155. 158. 159.]  
 [ 51.  43.  40. ... 164. 165. 166.]  
 [ 50.  46.  47. ... 170. 169. 172.]]
```



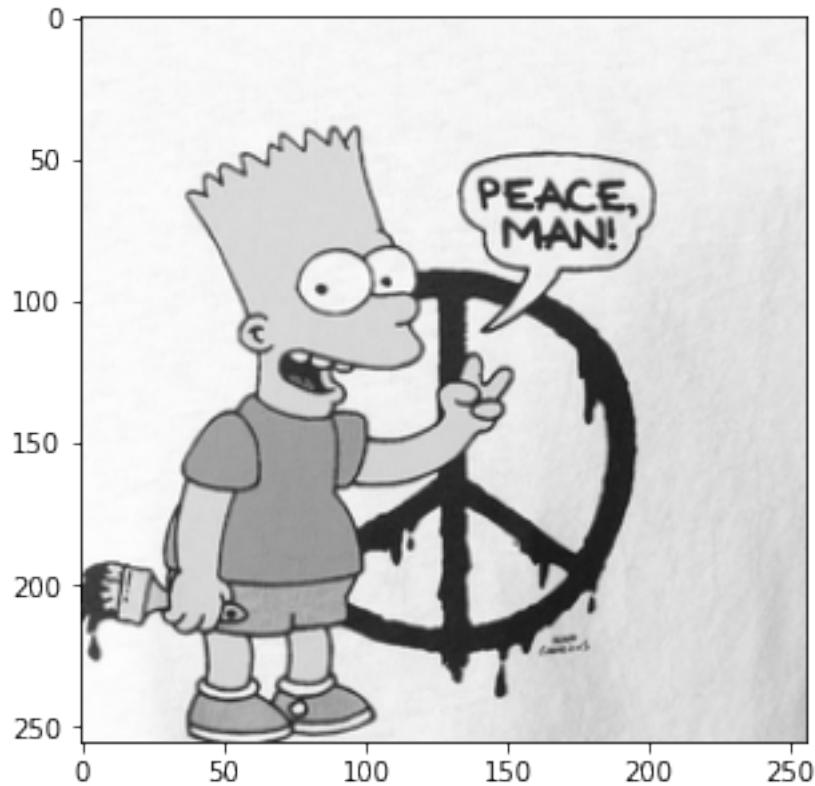
```
[[201. 201. 201. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]  
 ...  
 [201. 201. 201. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]]
```



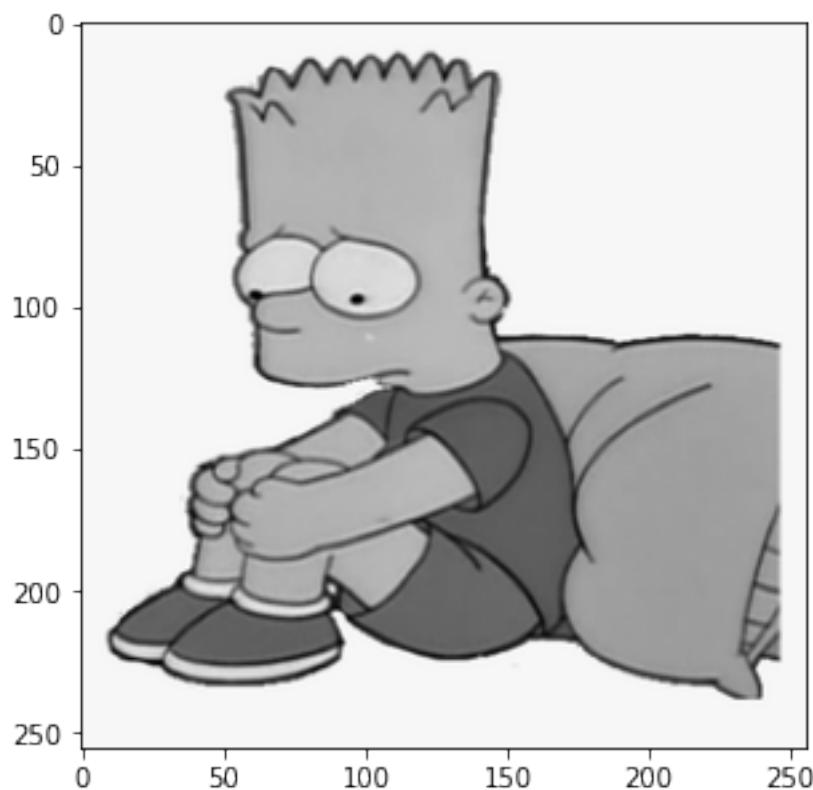
```
[[201. 201. 201. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]  
 ...  
 [201. 201. 201. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]]
```



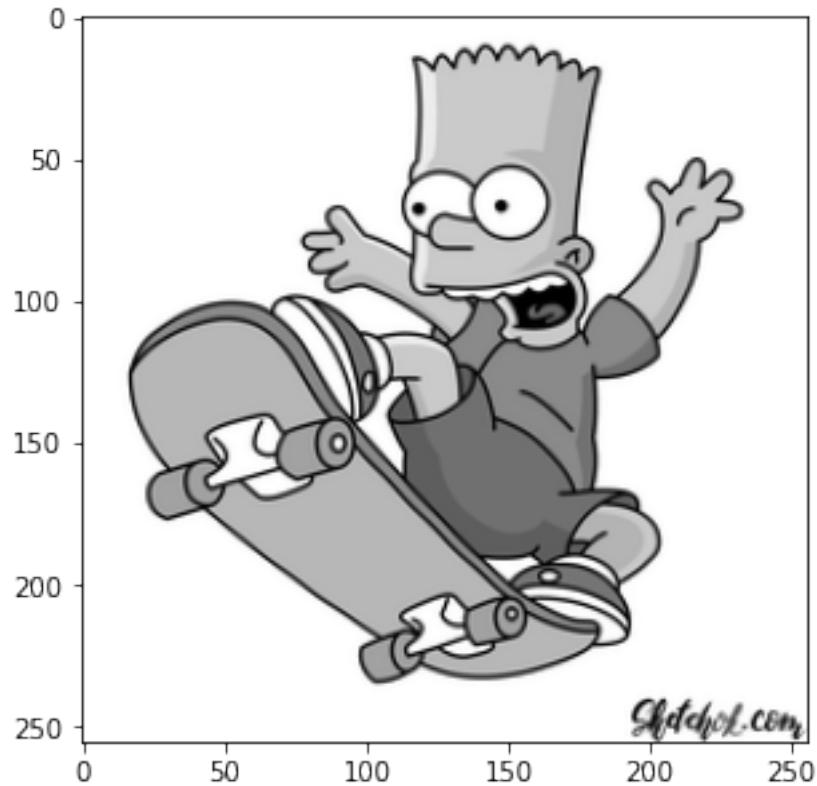
```
[[246. 246. 250. ... 246. 244. 243.]  
 [243. 245. 249. ... 244. 244. 243.]  
 [243. 246. 248. ... 245. 246. 244.]  
 ...  
 [252. 252. 252. ... 203. 199. 193.]  
 [253. 253. 253. ... 201. 194. 190.]  
 [252. 253. 251. ... 196. 193. 195.]]
```



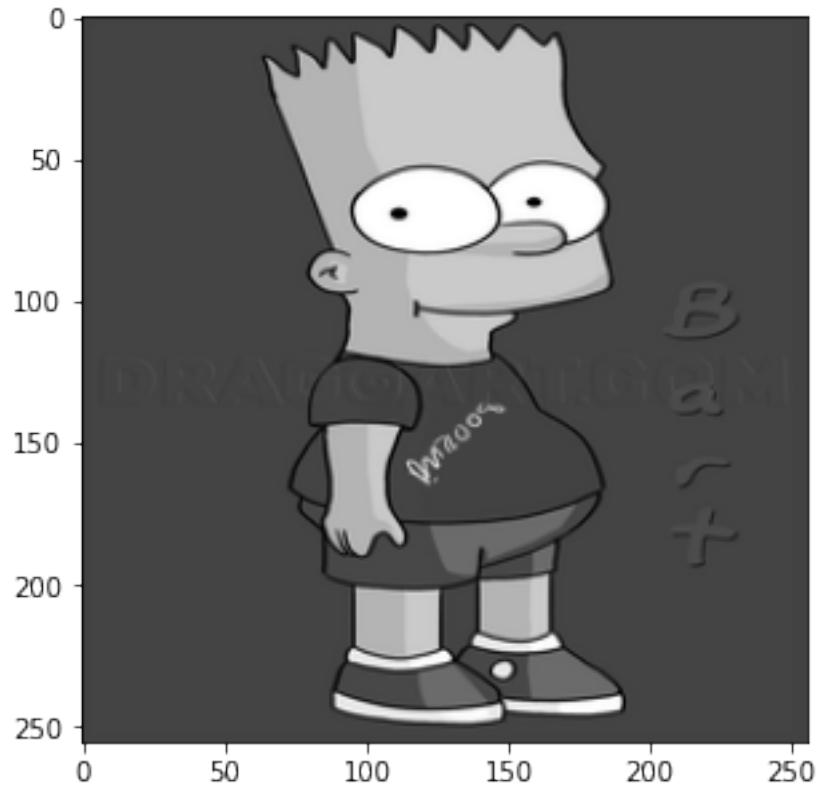
[[247. 247. 247. ... 247. 247. 247.]
[247. 247. 247. ... 247. 247. 247.]
[247. 247. 247. ... 247. 247. 247.]
...
[247. 247. 247. ... 247. 247. 247.]
[247. 247. 247. ... 247. 247. 247.]
[247. 247. 247. ... 247. 247. 247.]]



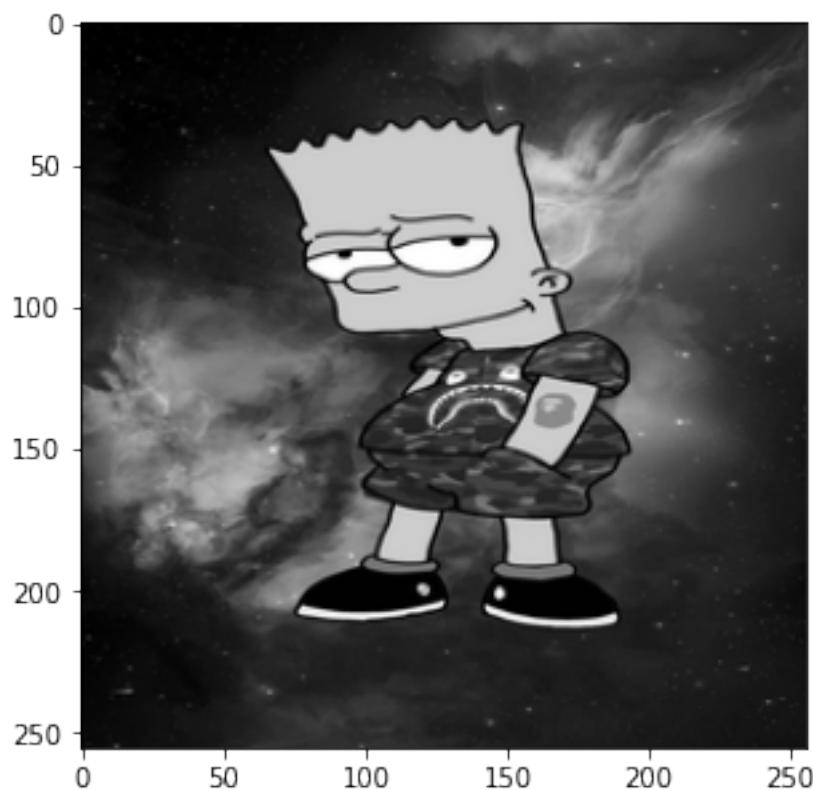
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 130. 235. 255.]  
 [255. 255. 255. ... 227. 255. 252.]  
 [255. 255. 255. ... 252. 255.]]
```



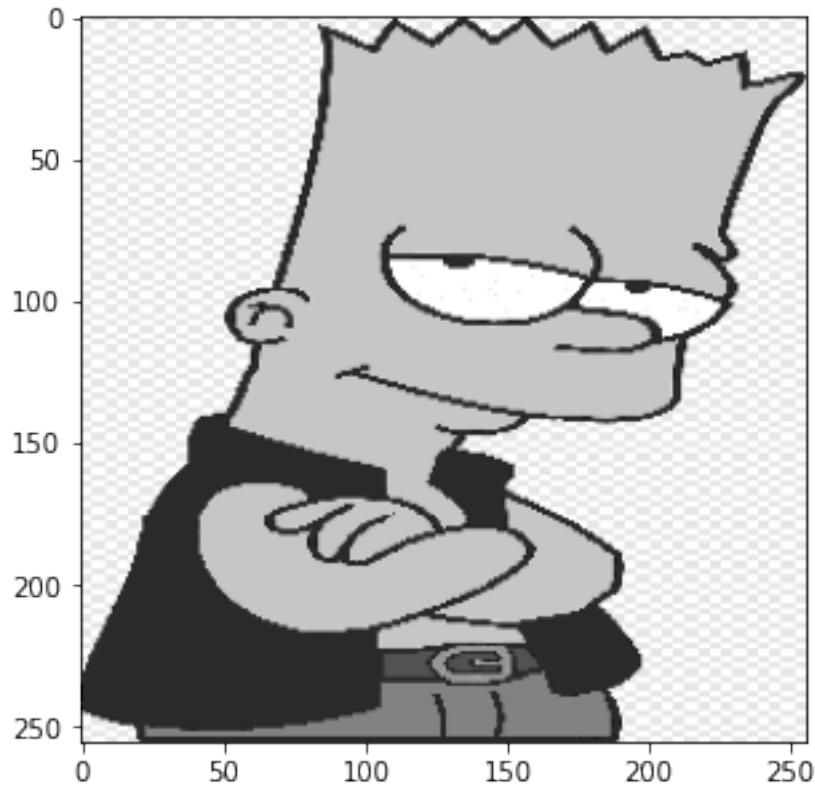
```
[[66. 66. 66. ... 66. 66. 66.]  
[66. 66. 66. ... 66. 66. 66.]  
[66. 66. 66. ... 66. 66. 66.]  
...  
[66. 66. 66. ... 66. 66. 66.]  
[66. 66. 66. ... 66. 66. 66.]  
[66. 66. 66. ... 66. 66. 66.]]
```



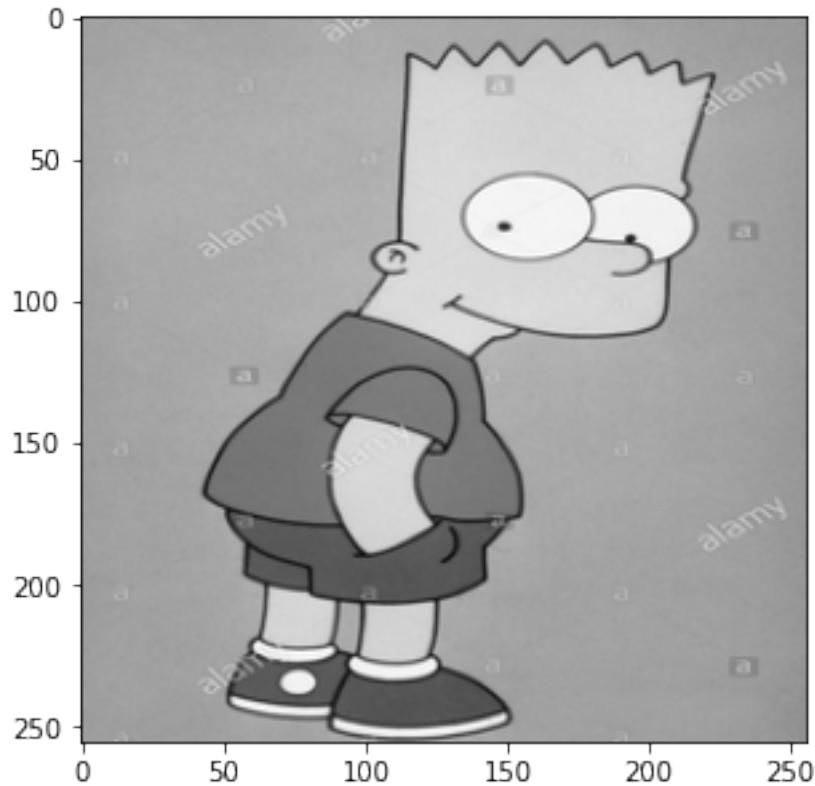
```
[[ 4.  5.  6. ... 41. 41. 39.]  
 [ 3.  2.  3. ... 40. 41. 41.]  
 [ 3.  2.  1. ... 42. 43. 41.]  
 ...  
 [ 8.  9. 13. ... 19. 18. 19.]  
 [ 9. 10. 19. ... 18. 19. 17.]  
 [12. 13. 11. ... 18. 18. 19.]]
```



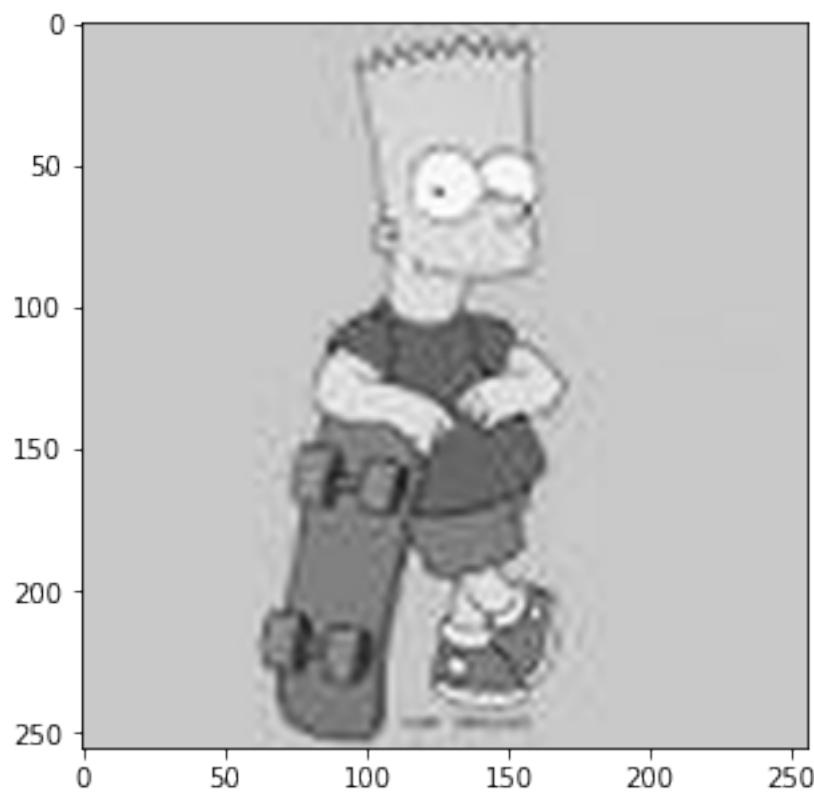
```
[[255. 255. 255. ... 230. 230. 230.]  
 [255. 255. 255. ... 230. 230. 230.]  
 [230. 230. 230. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 230. 230. 230.]  
 [230. 230. 230. ... 255. 255. 255.]  
 [230. 230. 230. ... 255. 255. 255.]]
```



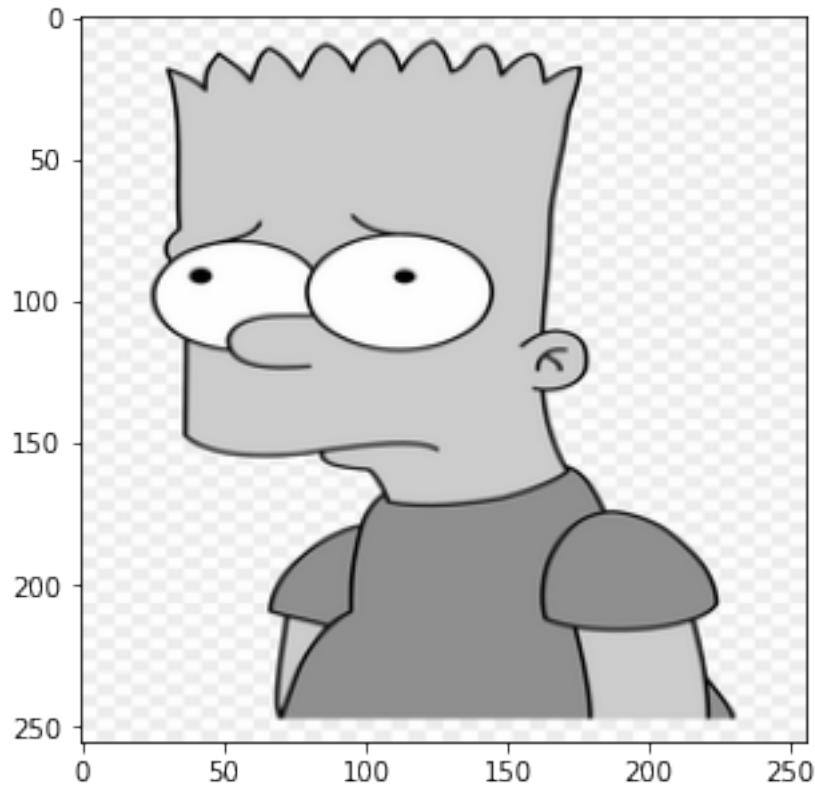
```
[[150. 151. 152. ... 141. 145. 141.]  
 [165. 163. 164. ... 157. 155. 157.]  
 [162. 160. 164. ... 152. 150. 152.]  
 ...  
 [159. 157. 157. ... 146. 144. 146.]  
 [157. 159. 160. ... 145. 143. 147.]  
 [157. 160. 160. ... 149. 148. 146.]]
```



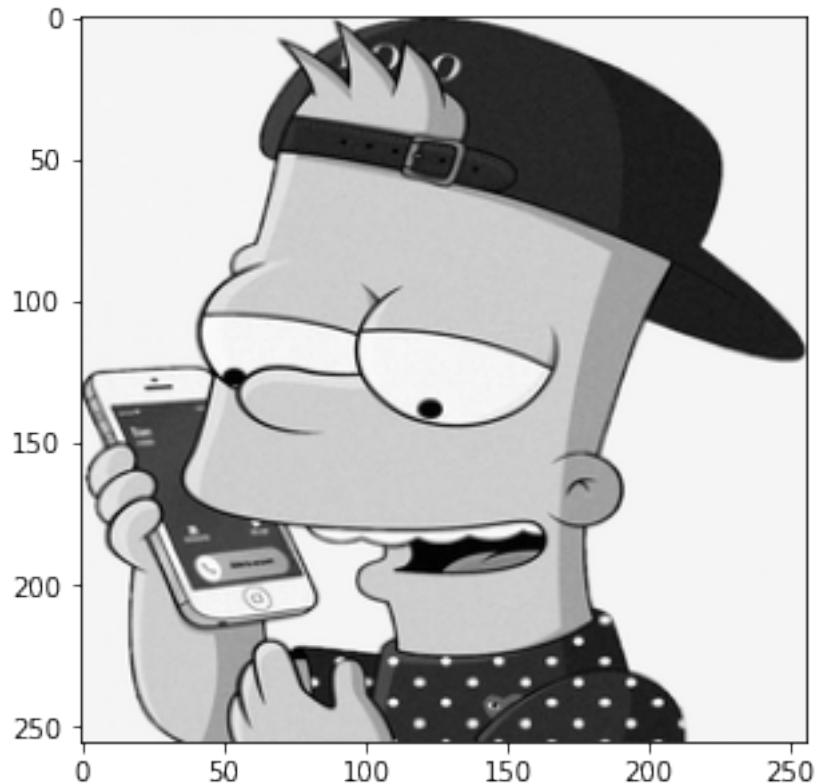
```
[[201. 201. 201. ... 201. 201. 201.]  
[201. 201. 201. ... 201. 201. 201.]  
[201. 201. 201. ... 201. 201. 201.]  
...  
[201. 201. 201. ... 201. 201. 201.]  
[201. 201. 201. ... 201. 201. 201.]  
[201. 201. 201. ... 201. 201. 201.]]
```



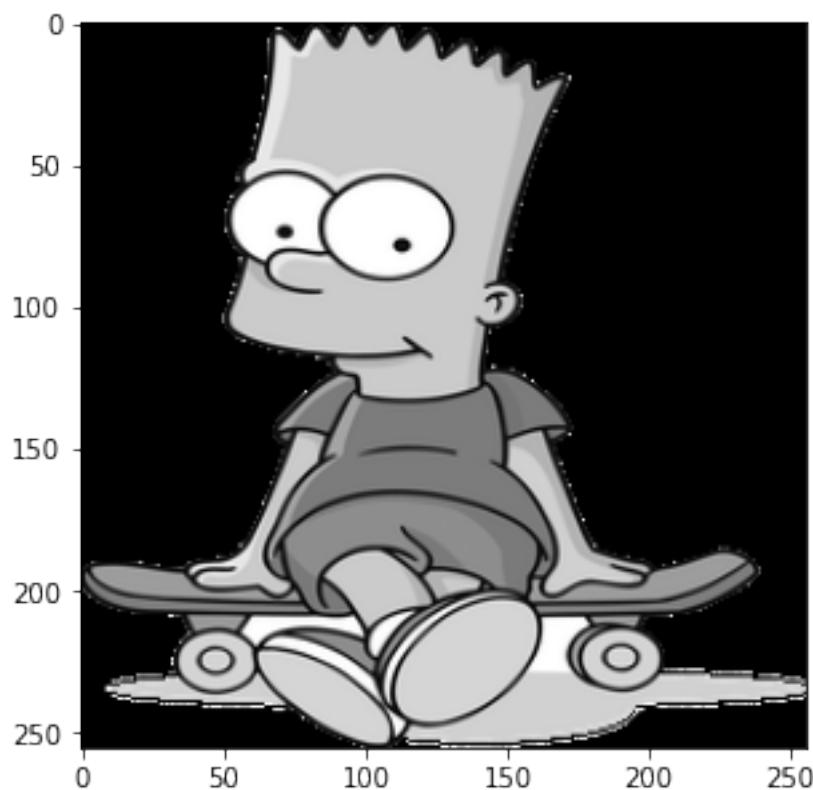
```
[[254. 254. 254. ... 237. 237. 238.]  
 [254. 254. 254. ... 237. 237. 238.]  
 [255. 254. 254. ... 237. 237. 238.]  
 ...  
 [255. 254. 254. ... 237. 237. 238.]  
 [255. 255. 255. ... 237. 236. 237.]  
 [251. 251. 251. ... 240. 240. 240.]]
```



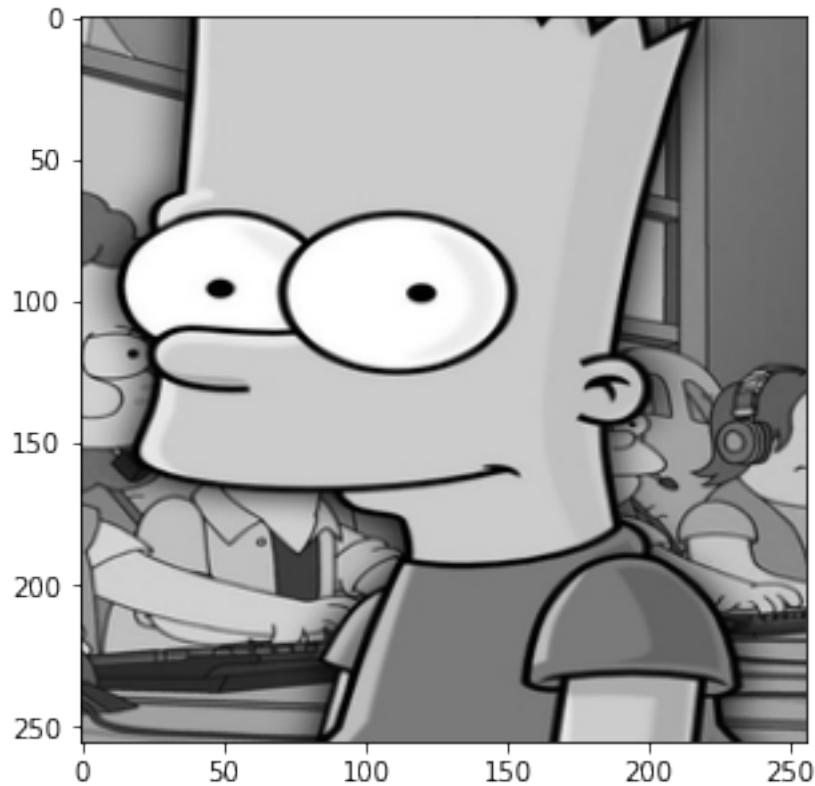
```
[[245. 246. 245. ... 246. 245. 245.]  
 [246. 245. 245. ... 246. 246. 245.]  
 [245. 245. 245. ... 246. 246. 245.]  
 ...  
 [245. 245. 245. ... 245. 245. 245.]  
 [245. 245. 245. ... 246. 246. 246.]  
 [245. 246. 246. ... 245. 246. 245.]]
```



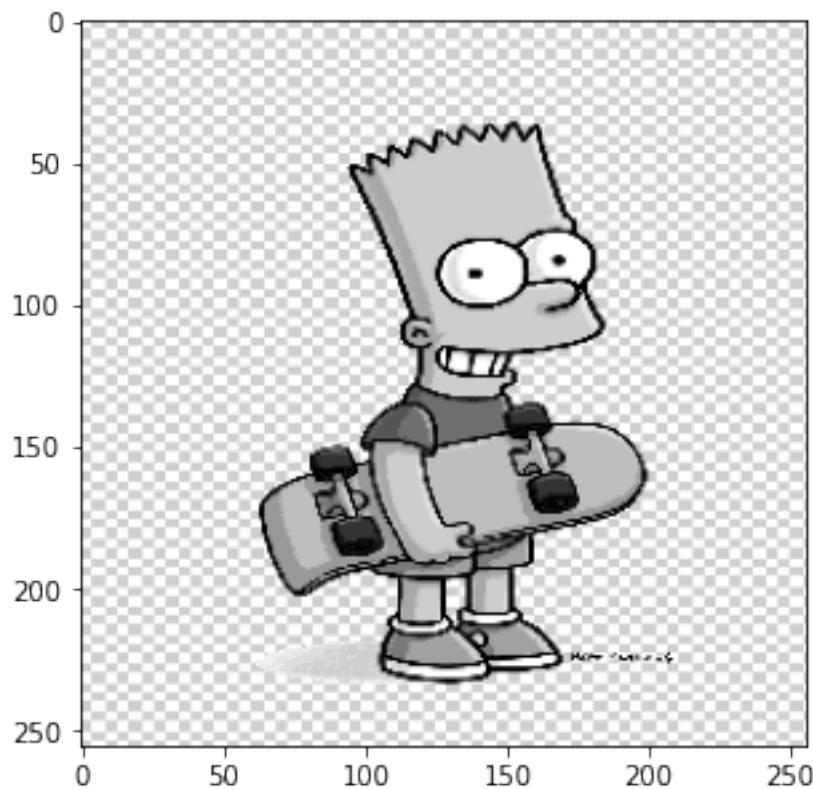
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



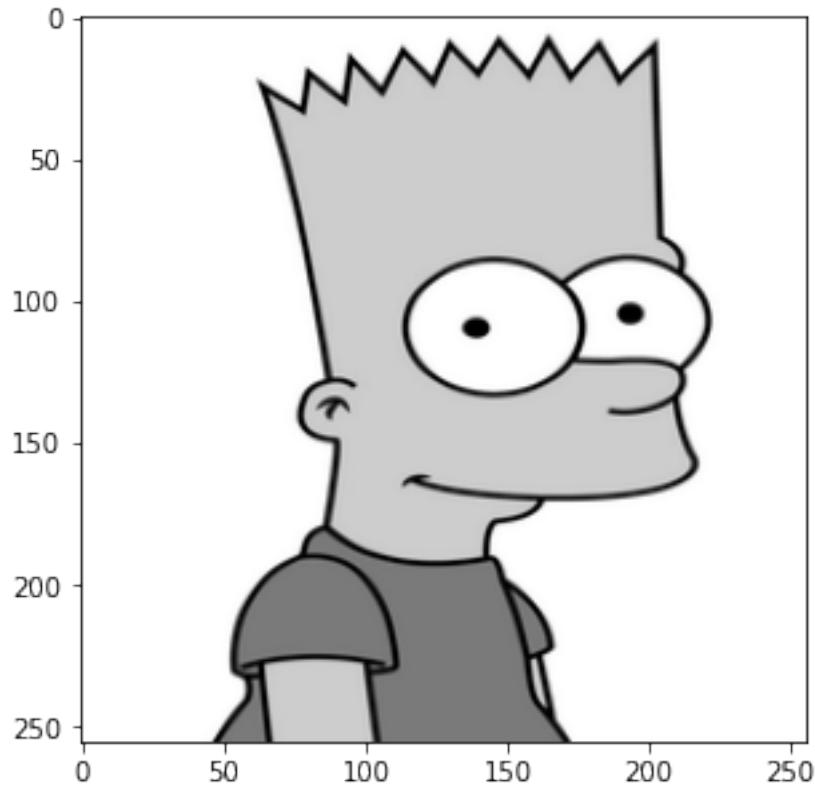
```
[[182. 212. 213. ... 111. 111. 110.]  
 [188. 212. 211. ... 111. 111. 110.]  
 [196. 215. 209. ... 111. 111. 110.]  
 ...  
 [156. 154. 153. ... 163. 164. 166.]  
 [164. 163. 165. ... 162. 163. 162.]  
 [162. 162. 161. ... 161. 162. 162.]]
```



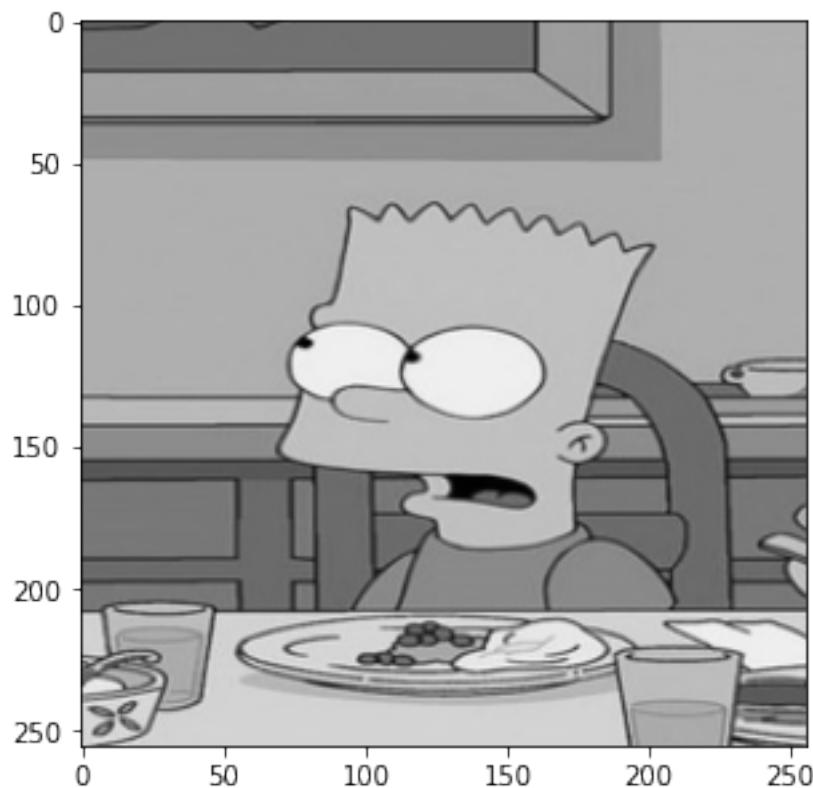
```
[[255. 255. 255. ... 255. 255. 207.]  
[255. 255. 255. ... 255. 255. 207.]  
[255. 255. 255. ... 255. 255. 207.]  
...  
[255. 255. 255. ... 255. 255. 207.]  
[207. 207. 207. ... 207. 207. 255.]  
[207. 207. 207. ... 207. 207. 255.]]
```



```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [254. 254. 254. ... 254. 254. 254.]]
```



```
[[ 97.40343348  96.30901288  95.21459227 ... 174.01287554 174.01287554
174.01287554]
[ 96.30901288  96.30901288  95.21459227 ... 176.20171674 176.20171674
176.20171674]
[ 91.93133047  91.93133047  90.83690987 ... 174.01287554 174.01287554
174.01287554]
...
[213.41201717 211.22317597 216.69527897 ... 161.97424893 149.93562232
135.70815451]
[214.50643777 216.69527897 219.97854077 ... 228.73390558 233.11158798
230.92274678]
[215.60085837 215.60085837 204.65665236 ... 105.06437768 118.19742489
131.3304721 ]]
```



[[95. 95. 95. ... 95. 95. 95.]

[95. 95. 95. ... 95. 95. 95.]

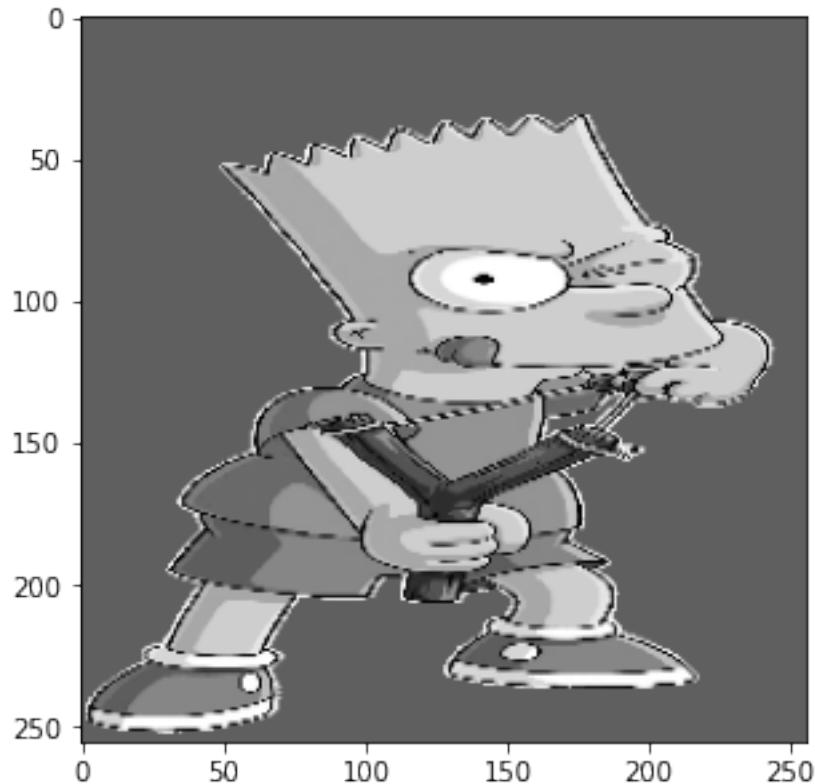
[95. 95. 95. ... 95. 95. 95.]

..

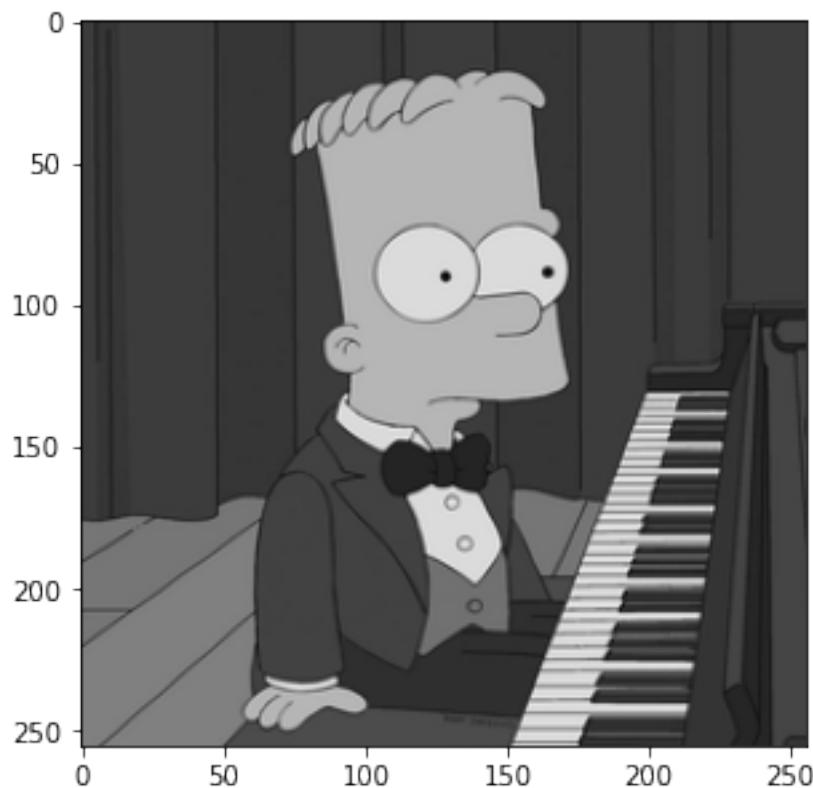
[95. 95. 95. ... 95. 95. 95.]

[95. 95. 95. ... 95. 95. 95.]

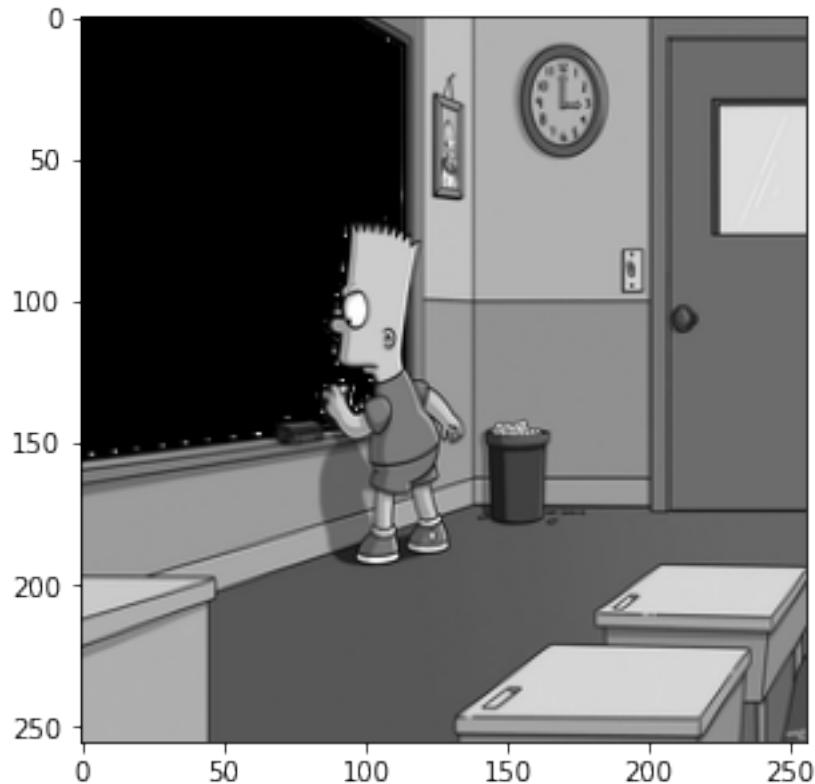
[95. 95. 95. ... 95. 95. 95.]]



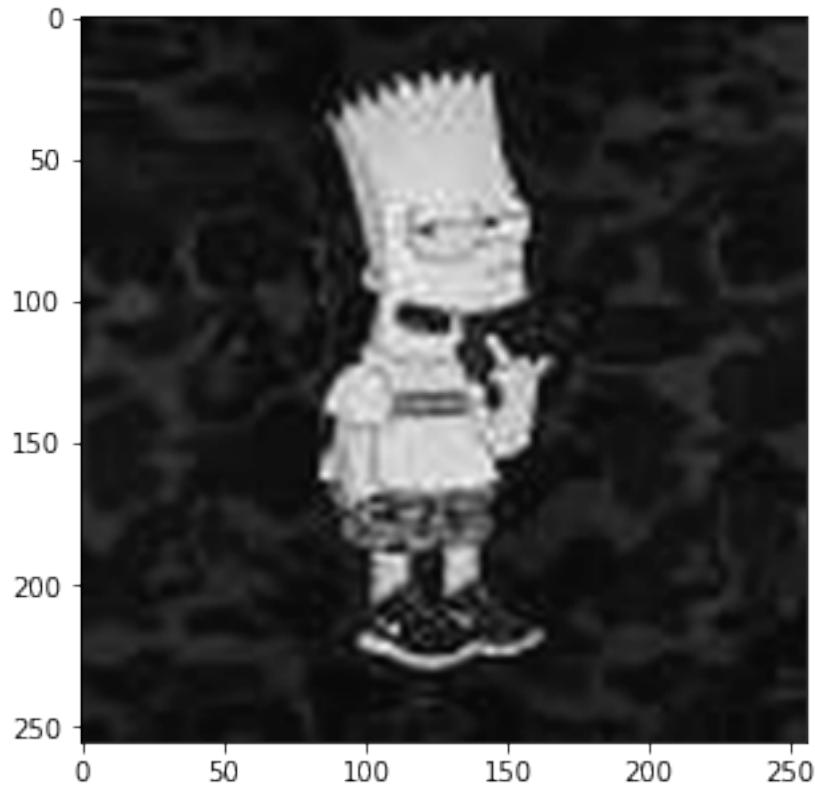
```
[[ 60.71428571  60.71428571  60.71428571 ...  60.71428571  60.71428571
  60.71428571]
 [ 60.71428571  60.71428571  60.71428571 ...  60.71428571  60.71428571
  60.71428571]
 [ 60.71428571  60.71428571  60.71428571 ...  60.71428571  60.71428571
  60.71428571]
 ...
 [128.05194805 128.05194805 128.05194805 ... 45.25974026 45.25974026
 45.25974026]
 [128.05194805 128.05194805 128.05194805 ... 55.19480519 55.19480519
 55.19480519]
 [128.05194805 128.05194805 129.15584416 ... 52.98701299 52.98701299
 52.98701299]]
```



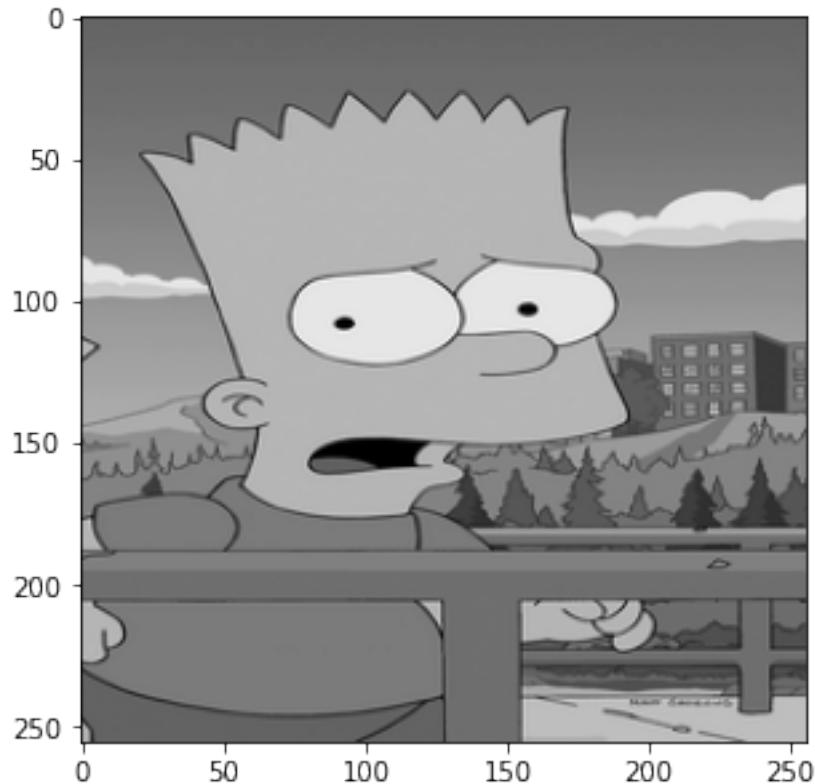
```
[[ 0.   0.   0. ... 118. 119. 119.]  
 [ 0.   0.   0. ... 118. 119. 119.]  
 [ 0.   0.   0. ... 119. 119. 119.]  
 ...  
 [175. 174. 174. ... 143. 117.  97.]  
 [175. 174. 175. ... 128. 139. 122.]  
 [175. 174. 175. ...  95. 101.  93.]]
```



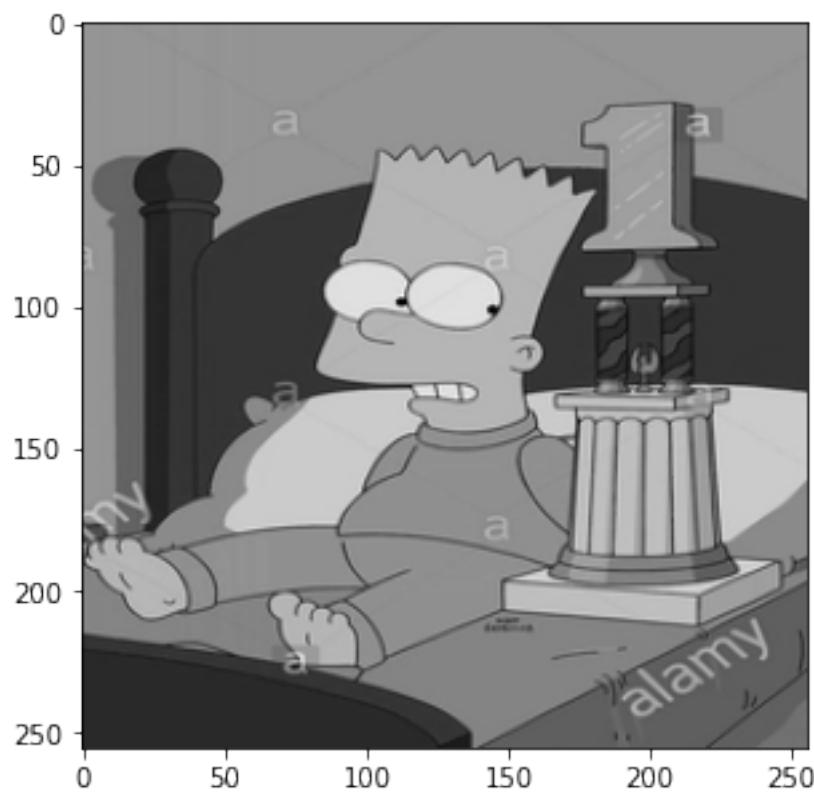
```
[[ 9.27272727  9.27272727 12.36363636 ... 13.90909091 13.90909091  
13.90909091]  
[ 9.27272727  9.27272727 12.36363636 ... 17.          17.  
17.          ]]  
[ 9.27272727  9.27272727 10.81818182 ... 24.72727273 24.72727273  
24.72727273]  
...  
[13.90909091 13.90909091 12.36363636 ... 38.63636364 37.09090909  
37.09090909]  
[15.45454545 15.45454545 13.90909091 ... 37.09090909 35.54545455  
35.54545455]  
[17.          17.          15.45454545 ... 37.09090909 35.54545455  
34.          ]]
```



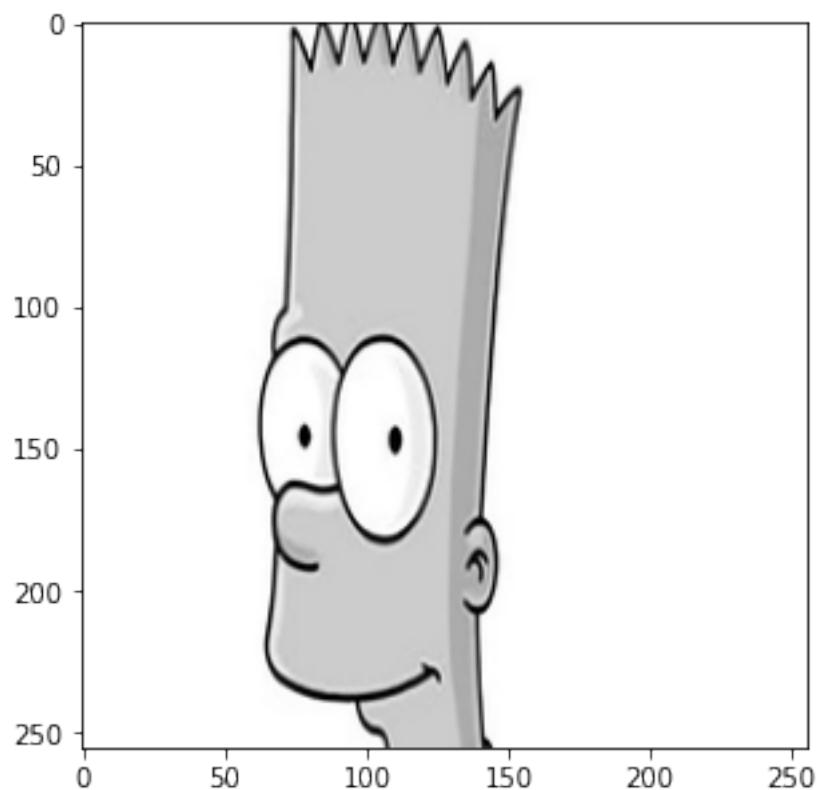
```
[[100.65789474 100.65789474 100.65789474 ... 100.65789474 99.53947368  
101.77631579]  
[100.65789474 100.65789474 100.65789474 ... 100.65789474 100.65789474  
101.77631579]  
[101.77631579 101.77631579 101.77631579 ... 101.77631579 101.77631579  
101.77631579]  
...  
[176.71052632 173.35526316 185.65789474 ... 190.13157895 190.13157895  
190.13157895]  
[176.71052632 174.47368421 186.77631579 ... 190.13157895 190.13157895  
190.13157895]  
[175.59210526 173.35526316 186.77631579 ... 190.13157895 190.13157895  
190.13157895]]
```



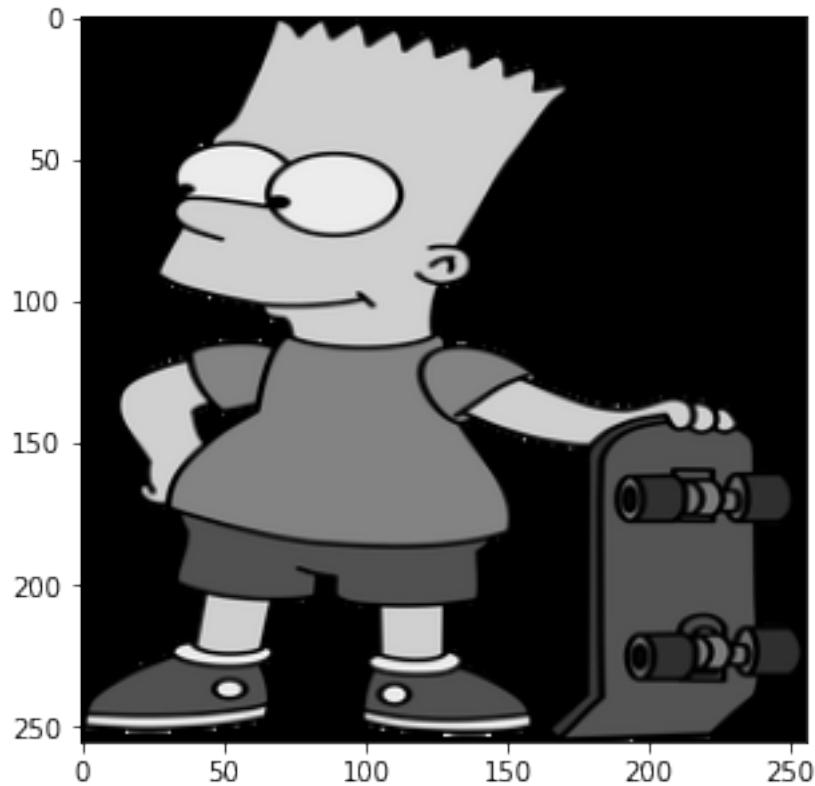
```
[[150.12096774 150.12096774 150.12096774 ... 148.06451613 148.06451613  
148.06451613]  
[149.09274194 149.09274194 149.09274194 ... 148.06451613 148.06451613  
148.06451613]  
[150.12096774 150.12096774 150.12096774 ... 148.06451613 148.06451613  
148.06451613]  
...  
[ 41.12903226 41.12903226 41.12903226 ... 114.13306452 114.13306452  
114.13306452]  
[ 41.12903226 41.12903226 41.12903226 ... 114.13306452 114.13306452  
114.13306452]  
[ 42.15725806 42.15725806 42.15725806 ... 114.13306452 114.13306452  
114.13306452]]
```



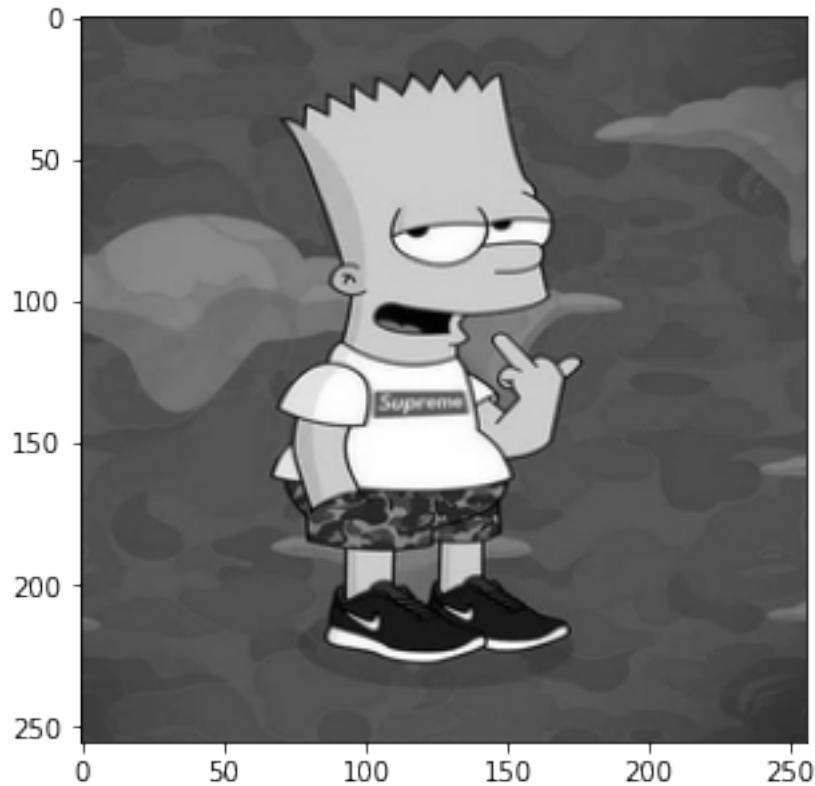
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



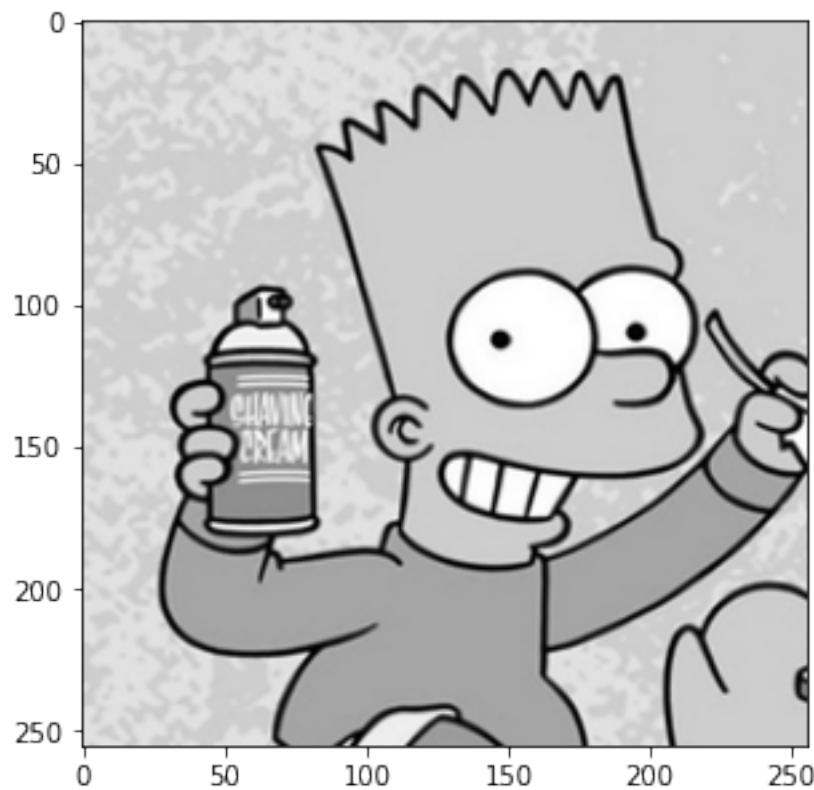
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



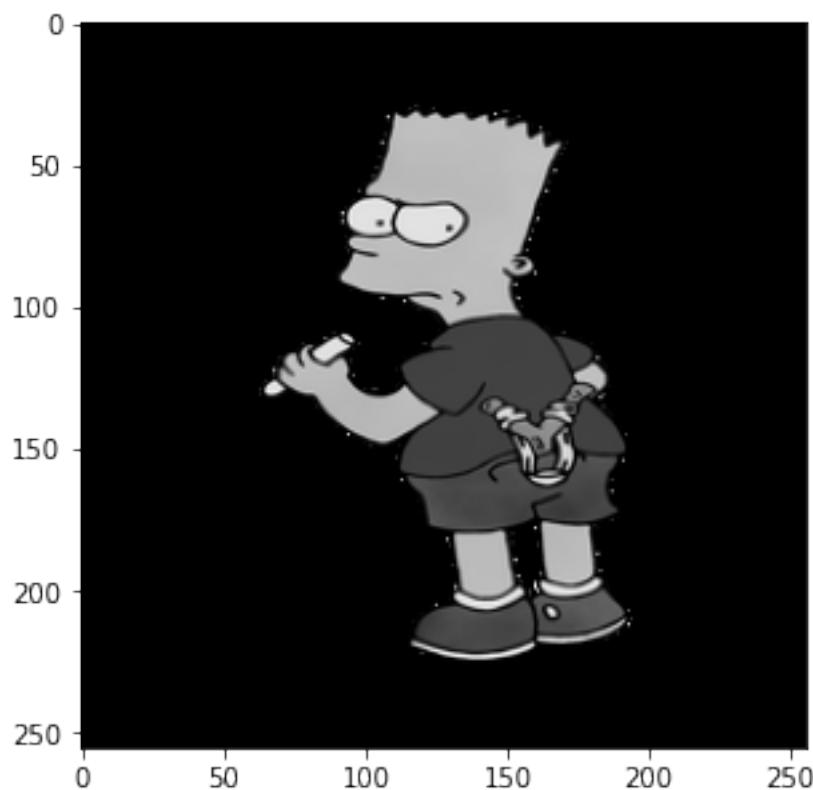
```
[[76. 76. 76. ... 68. 73. 66.]  
 [76. 76. 76. ... 67. 72. 67.]  
 [75. 75. 75. ... 65. 73. 69.]  
 ...  
 [63. 65. 65. ... 48. 49. 52.]  
 [63. 65. 68. ... 47. 48. 49.]  
 [65. 64. 64. ... 50. 49. 50.]]
```



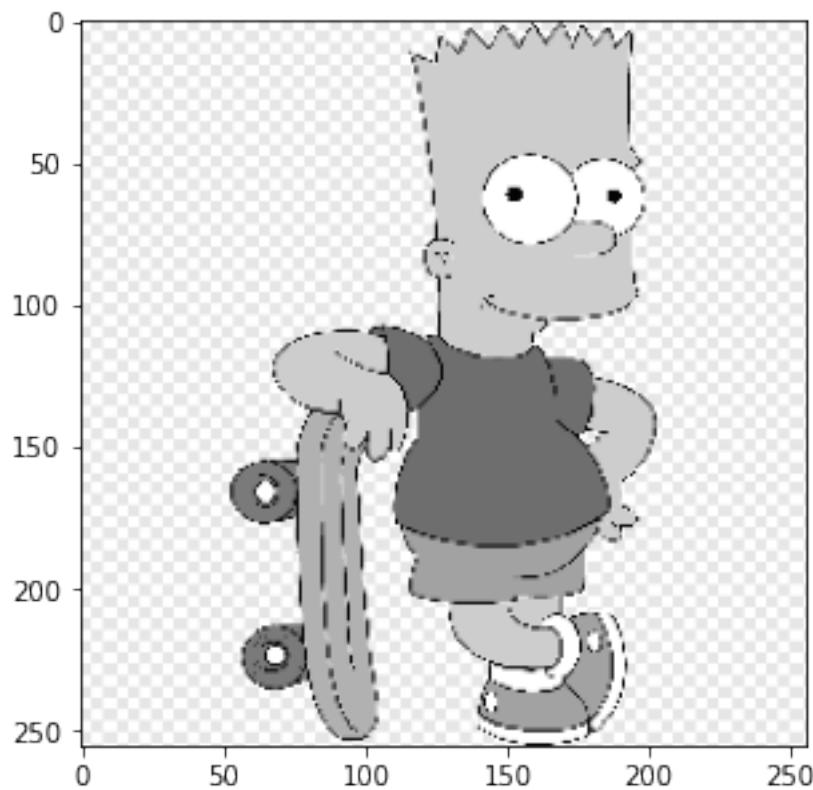
```
[[219. 211. 207. ... 207. 207. 207.]  
 [207. 206. 214. ... 207. 207. 207.]  
 [206. 214. 224. ... 207. 207. 207.]  
 ...  
 [225. 227. 226. ... 206. 206. 205.]  
 [225. 227. 225. ... 206. 206. 205.]  
 [226. 225. 218. ... 206. 206. 206.]]
```



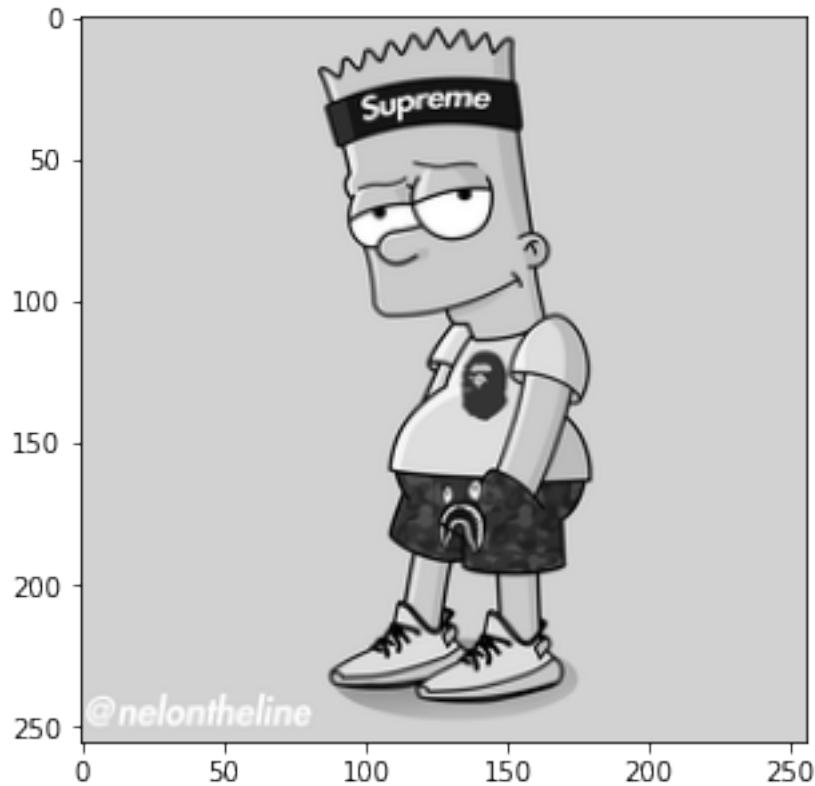
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



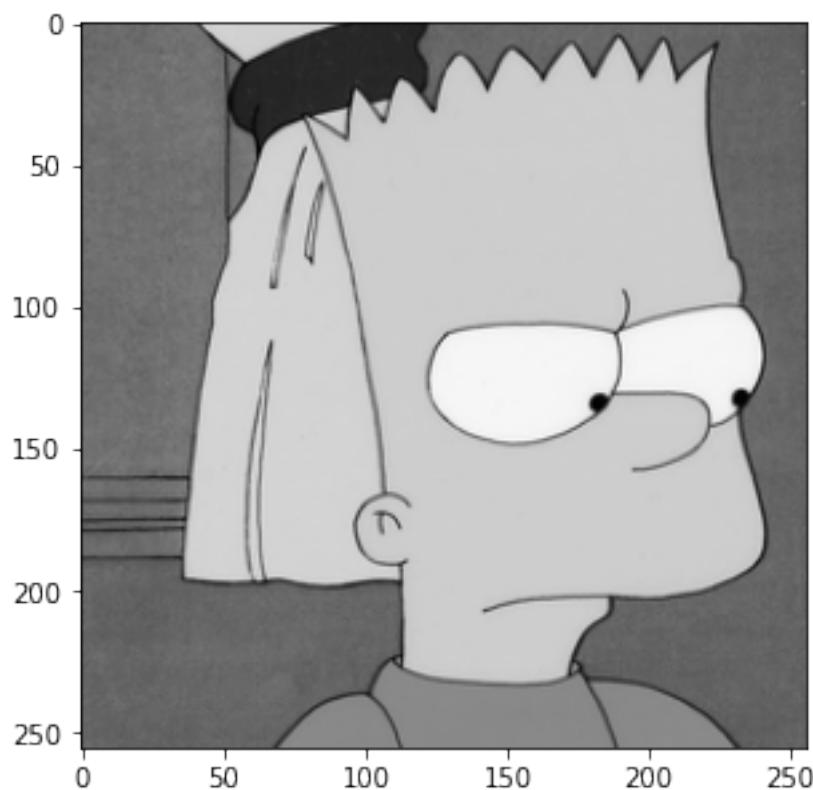
```
[[255. 255. 255. ... 255. 255. 230.]  
 [255. 255. 255. ... 255. 255. 230.]  
 [255. 255. 255. ... 255. 255. 230.]  
 ...  
 [230. 230. 230. ... 230. 230. 255.]  
 [230. 230. 230. ... 230. 230. 255.]  
 [255. 255. 255. ... 255. 255. 230.]]
```



```
[[210. 210. 210. ... 210. 210. 210.]  
 [210. 210. 210. ... 210. 210. 210.]  
 [210. 210. 210. ... 210. 210. 210.]  
 ...  
 [210. 210. 210. ... 210. 210. 210.]  
 [210. 210. 210. ... 210. 210. 210.]  
 [210. 210. 210. ... 210. 210. 210.]]
```



```
[[110.16 104.04 100.98 ... 95.88 95.88 95.88]
 [105.06 103.02 110.16 ... 99.96 95.88 89.76]
 [111.18 109.14 106.08 ... 99.96 92.82 96.9 ]
 ...
 [102.    98.94 108.12 ... 104.04 103.02 99.96]
 [100.98 103.02 109.14 ... 106.08 104.04 97.92]
 [103.02 107.1  106.08 ... 103.02 96.9  100.98]]
```



[[80. 81. 78. ... 70. 70. 59.]

[80. 81. 78. ... 70. 70. 59.]

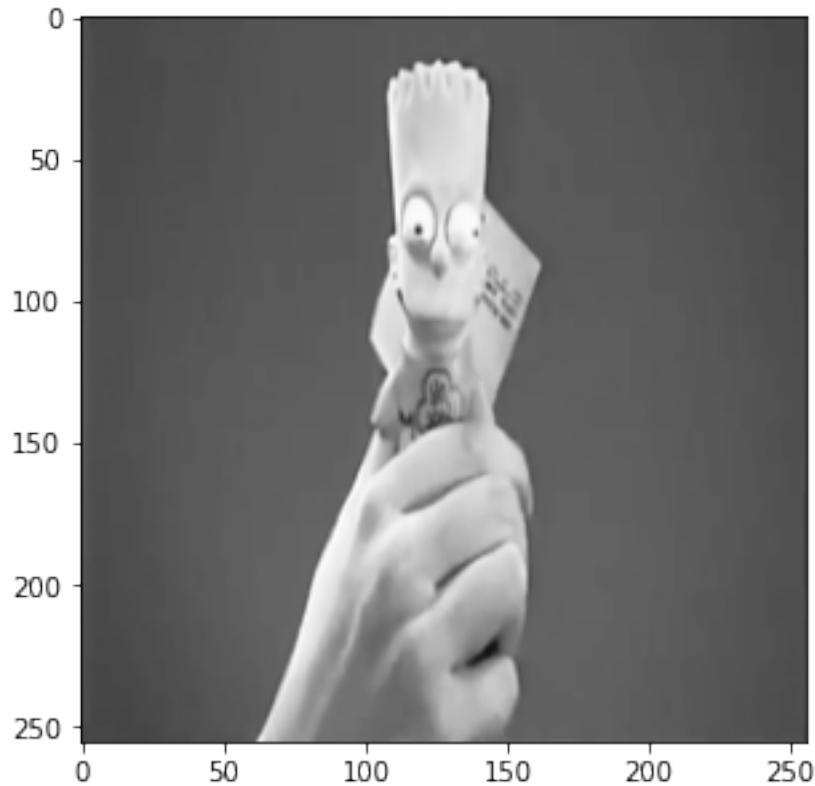
[80. 81. 79. ... 70. 71. 59.]

...

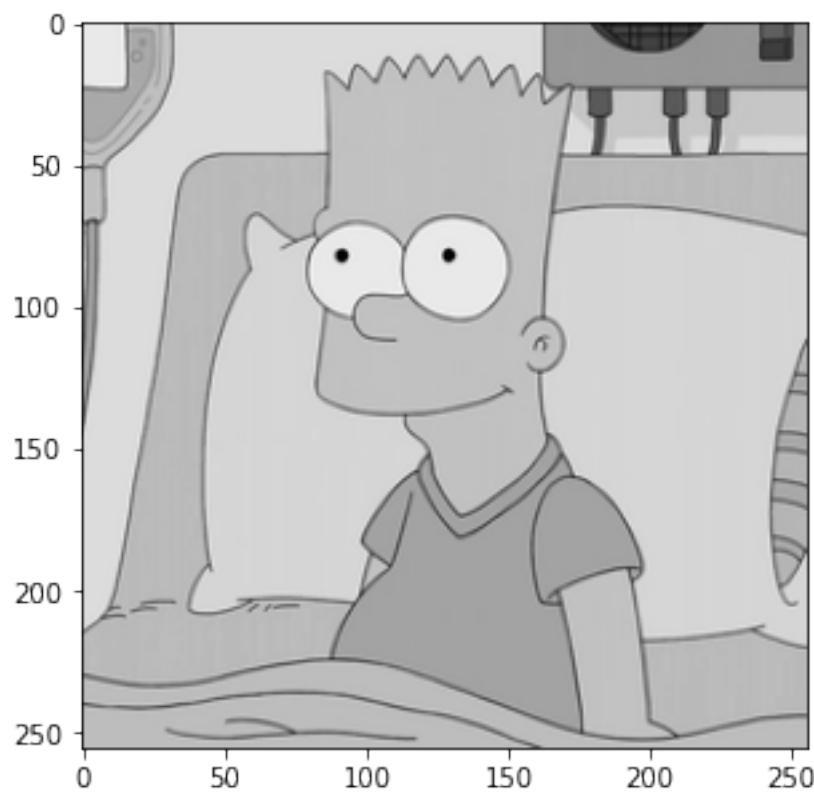
[80. 75. 72. ... 79. 79. 63.]

[80. 74. 72. ... 80. 81. 63.]

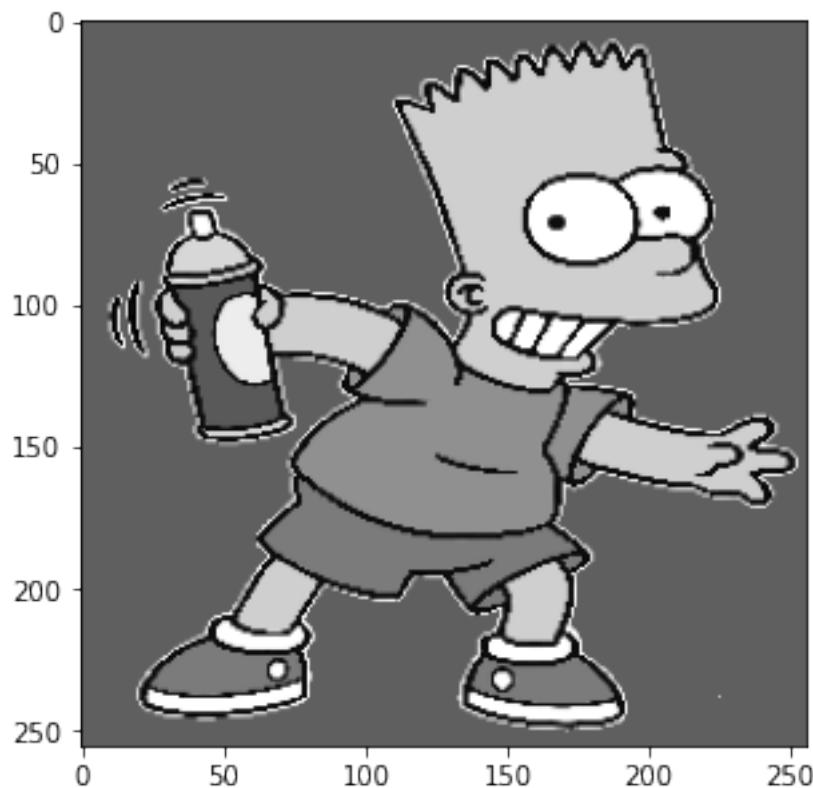
[78. 74. 72. ... 82. 83. 69.]]



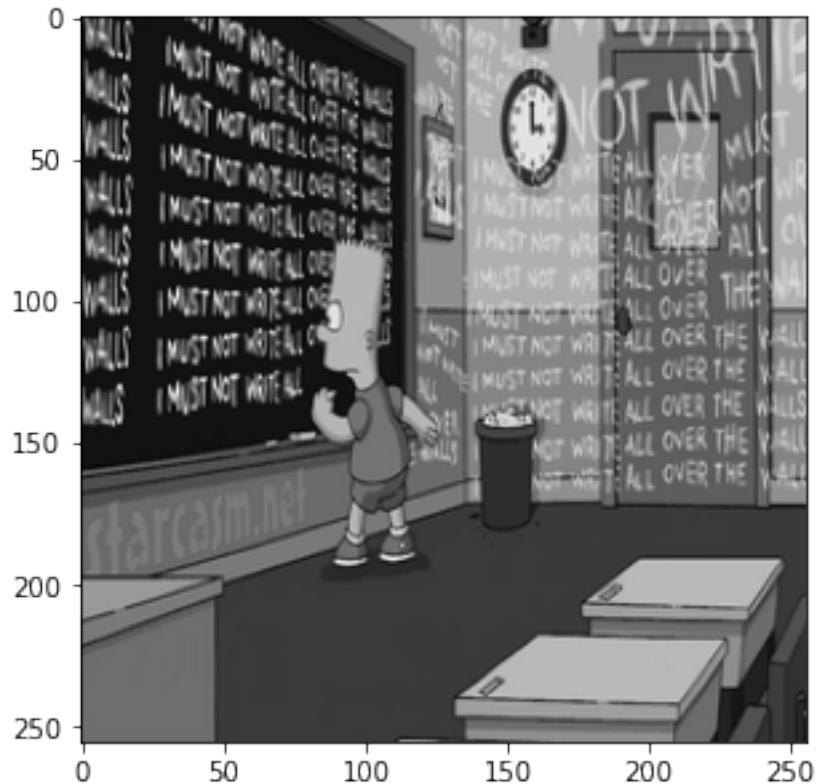
```
[[233.84279476 233.84279476 233.84279476 ... 150.32751092 152.55458515  
151.44104803]  
[233.84279476 233.84279476 233.84279476 ... 150.32751092 152.55458515  
151.44104803]  
[233.84279476 233.84279476 233.84279476 ... 150.32751092 152.55458515  
151.44104803]  
...  
[189.30131004 189.30131004 189.30131004 ... 189.30131004 189.30131004  
189.30131004]  
[189.30131004 189.30131004 189.30131004 ... 189.30131004 189.30131004  
189.30131004]  
[189.30131004 189.30131004 189.30131004 ... 189.30131004 189.30131004  
189.30131004]]
```



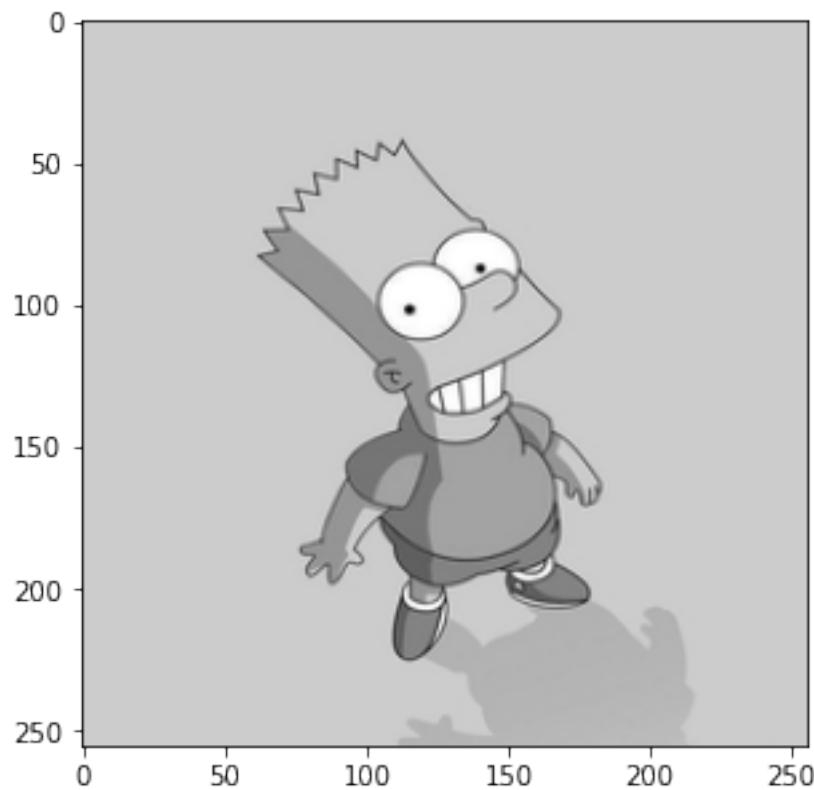
```
[[95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]  
 ...  
 [95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]]
```



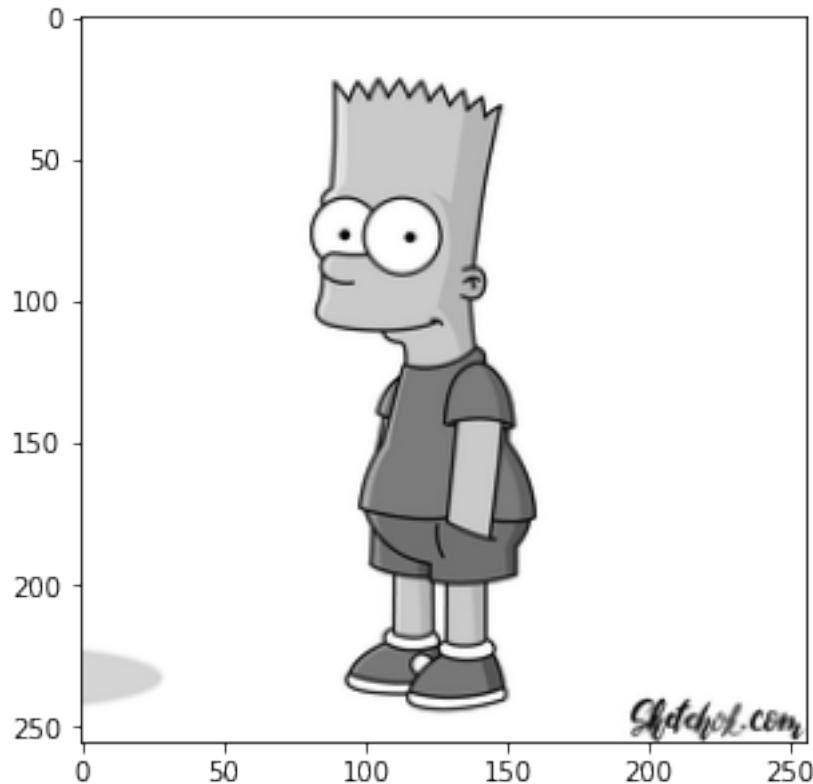
```
[[ 8.  80. 107. ... 219. 224. 216.]  
 [ 11.  63. 136. ... 210. 191. 181.]  
 [ 16.  35. 127. ... 185. 175. 177.]  
 ...  
 [109. 109. 109. ... 43. 41. 15.]  
 [109. 109. 109. ... 43. 40. 18.]  
 [109. 109. 109. ... 43. 38. 20.]]
```



```
[[204. 204. 204. ... 204. 204. 204.]  
[204. 204. 204. ... 204. 204. 204.]  
[204. 204. 204. ... 204. 204. 204.]  
...  
[204. 204. 204. ... 204. 204. 204.]  
[204. 204. 204. ... 204. 204. 204.]  
[204. 204. 204. ... 204. 204. 204.]]
```

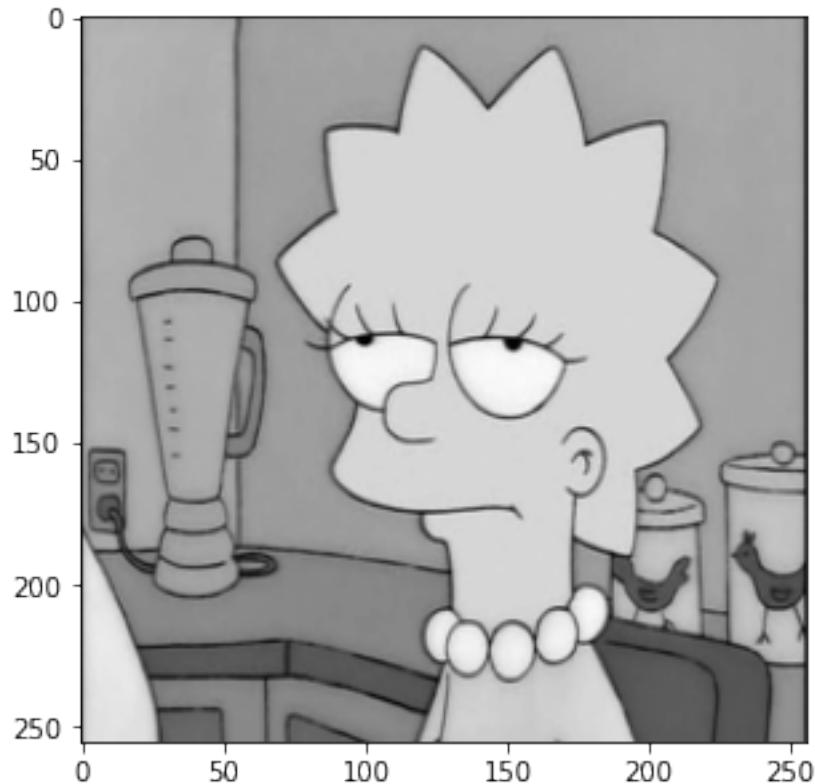


```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 130. 235. 255.]  
 [255. 255. 255. ... 227. 255. 252.]  
 [255. 255. 255. ... 252. 252. 255.]]
```

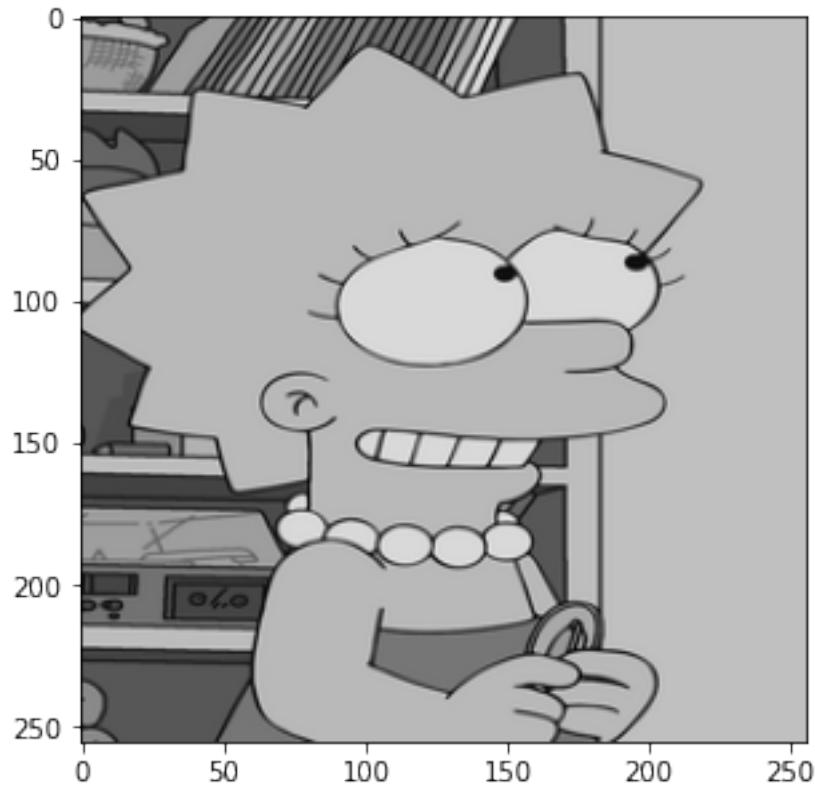


```
[ ]: for index, image in enumerate(grayScaleLisa):
    image = np.array(image)
    norm = image/(image.max()/255)
    print(norm)
    #convert back to image
    im = Image.fromarray(norm)
    im = im.convert('RGB')
    im.save("normalizedLisa/image-"+str(index)+".jpg")
    plt.imshow(im, cmap=plt.cm.gray)
    plt.show()
```

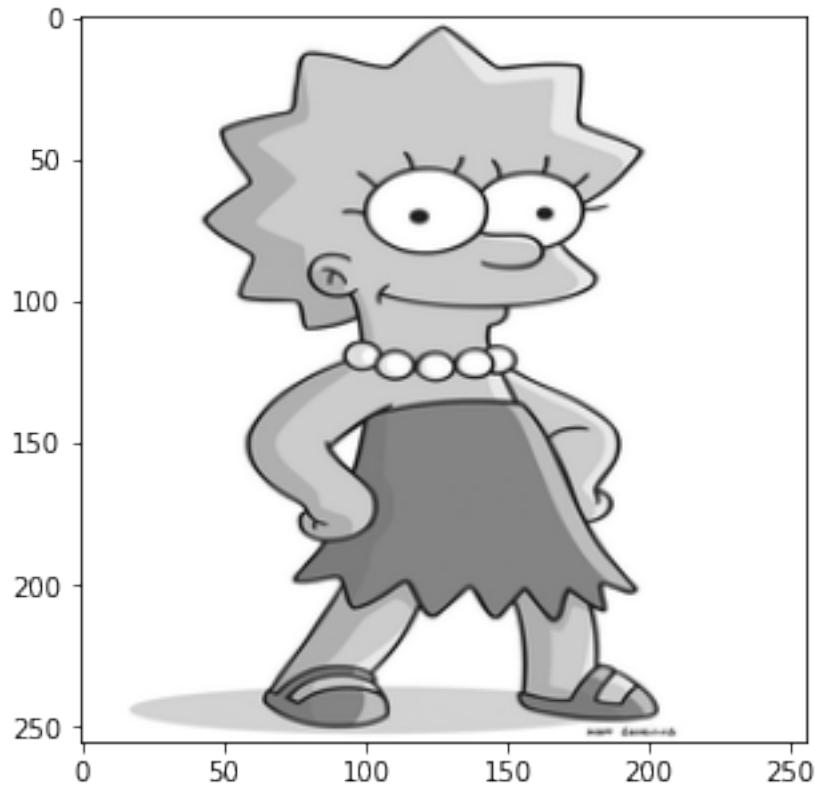
```
[[ 54.825 154.275 172.125 ... 161.925 153.      63.75 ]
 [ 53.55   151.725 172.125 ... 161.925 153.      63.75 ]
 [ 53.55   151.725 172.125 ... 161.925 153.      63.75 ]
 ...
 [ 70.125 189.975 216.75 ... 141.525 133.875  54.825]
 [ 70.125 189.975 216.75 ... 141.525 133.875  54.825]
 [ 70.125 189.975 216.75 ... 141.525 133.875  54.825]]
```



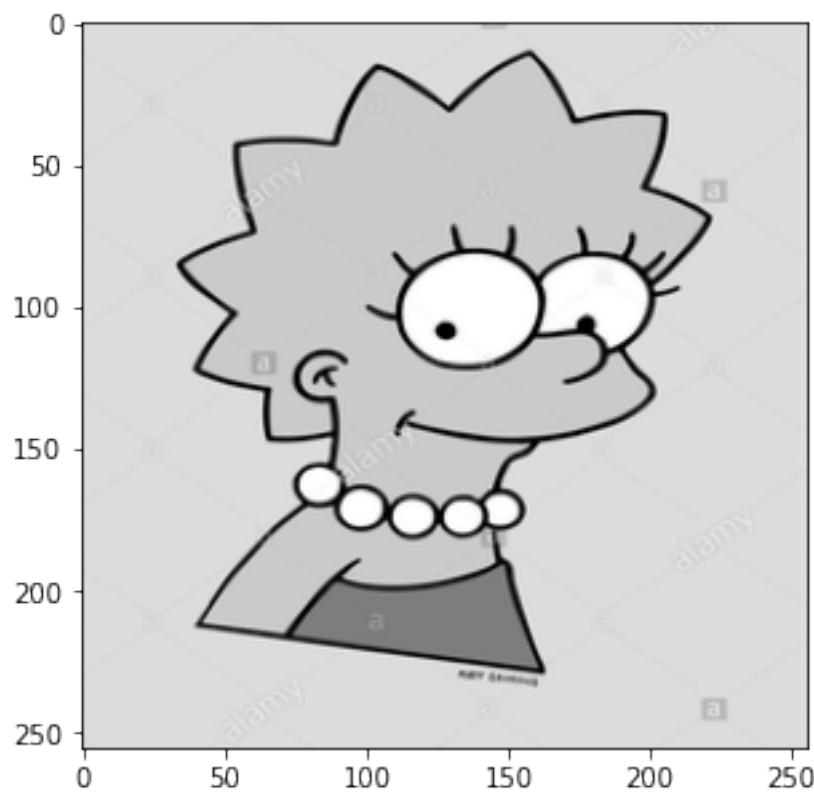
```
[[ 68.06772908  81.2749004   93.46613546 ... 192.01195219 192.01195219  
192.01195219]  
[ 50.79681275  37.58964143  29.46215139 ... 192.01195219 192.01195219  
192.01195219]  
[185.91633466 171.69322709 118.86454183 ... 192.01195219 192.01195219  
192.01195219]  
...  
[139.18326693 136.13545817 137.15139442 ... 192.01195219 192.01195219  
192.01195219]  
[138.16733068 138.16733068 138.16733068 ... 192.01195219 192.01195219  
192.01195219]  
[138.16733068 138.16733068 138.16733068 ... 192.01195219 192.01195219  
192.01195219]]
```



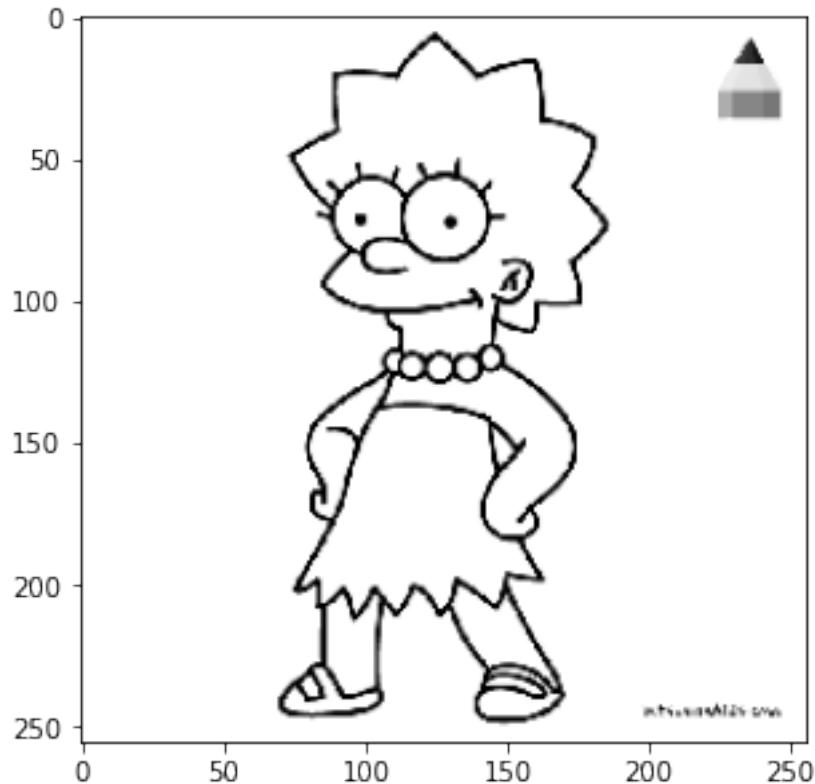
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



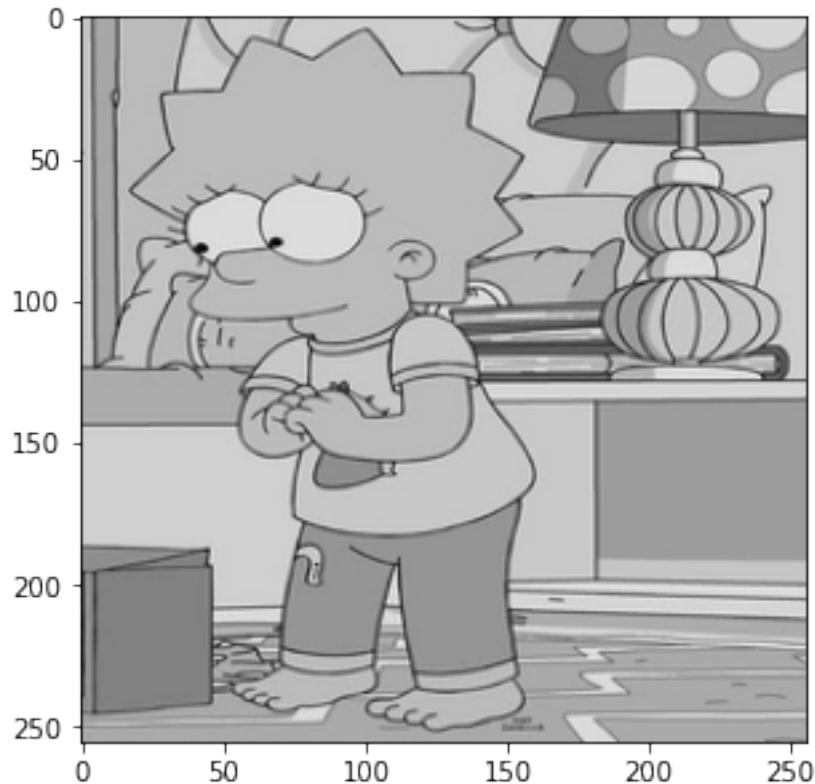
```
[[220, 220, 220, ... 220, 220, 220.]  
[221, 221, 221, ... 221, 221, 221.]  
[220, 220, 220, ... 220, 220, 220.]  
...  
[219, 216, 219, ... 221, 221, 221.]  
[221, 220, 217, ... 220, 220, 220.]  
[220, 220, 220, ... 221, 221, 221.]]
```



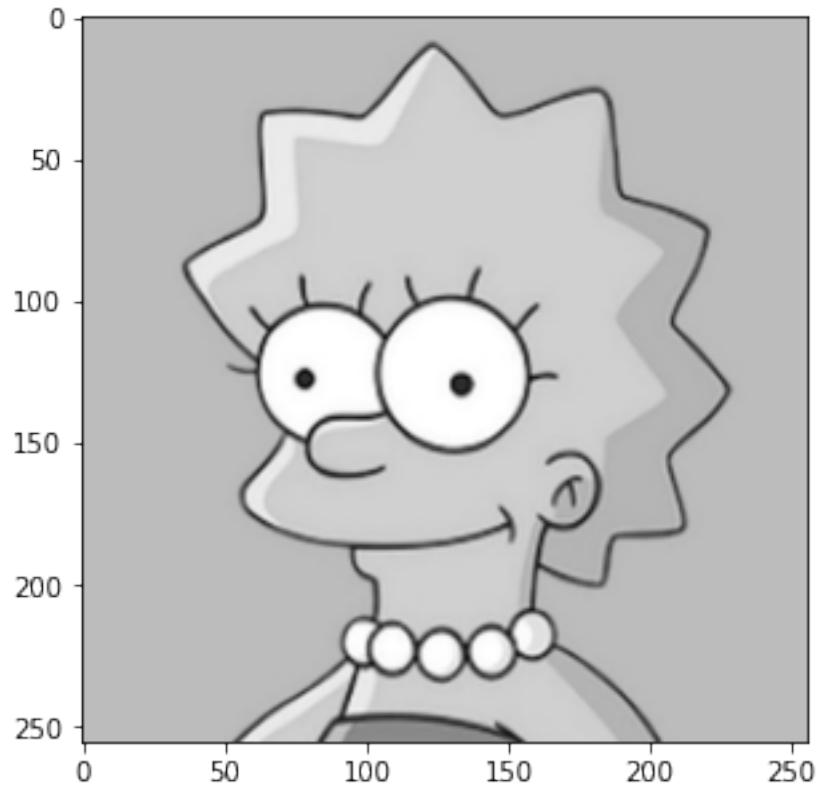
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



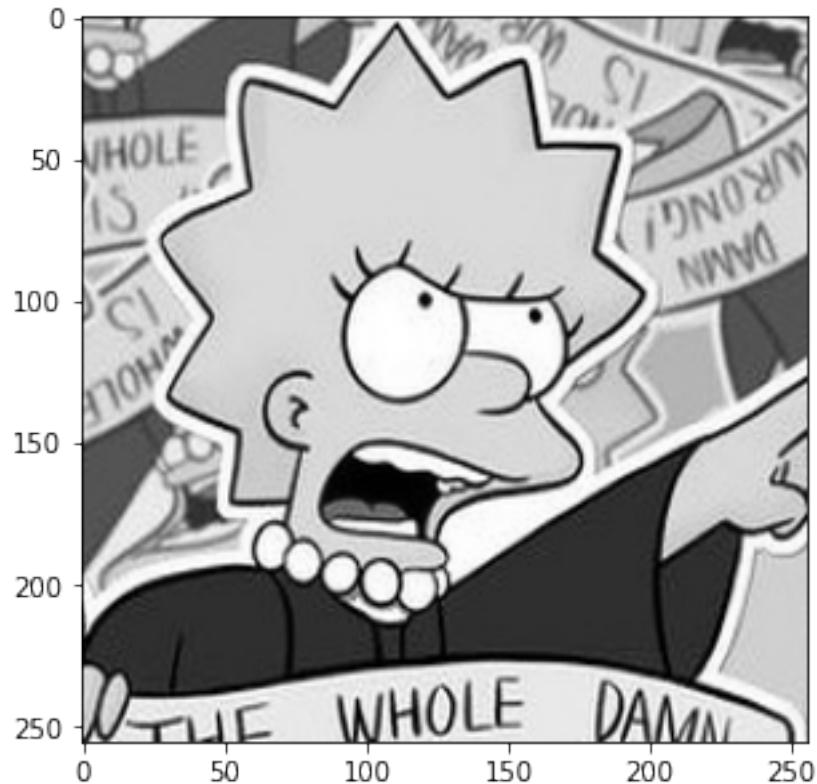
```
[[199.25101215 197.18623482 198.21862348 ... 214.73684211 216.80161943  
212.67206478]  
[199.25101215 197.18623482 198.21862348 ... 214.73684211 213.70445344  
221.96356275]  
[199.25101215 197.18623482 198.21862348 ... 217.8340081 212.67206478  
222.99595142]  
...  
[169.31174089 172.40890688 171.37651822 ... 172.40890688 170.34412955  
170.34412955]  
[170.34412955 171.37651822 170.34412955 ... 163.11740891 170.34412955  
173.4412955]  
[170.34412955 171.37651822 169.31174089 ... 129.048583 122.85425101  
140.4048583 ]]
```



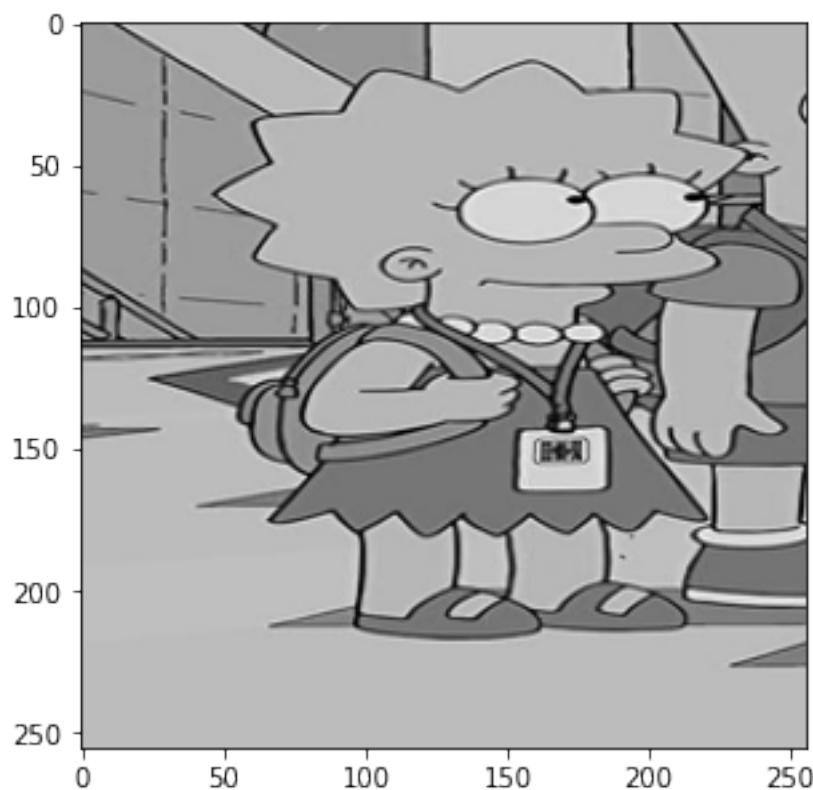
```
[[188. 188. 188. ... 188. 188. 188.]  
 [188. 188. 188. ... 188. 188. 188.]  
 [188. 188. 188. ... 188. 188. 188.]  
 ...  
 [188. 188. 188. ... 188. 188. 188.]  
 [188. 188. 188. ... 188. 188. 188.]  
 [188. 188. 188. ... 188. 188. 188.]]
```



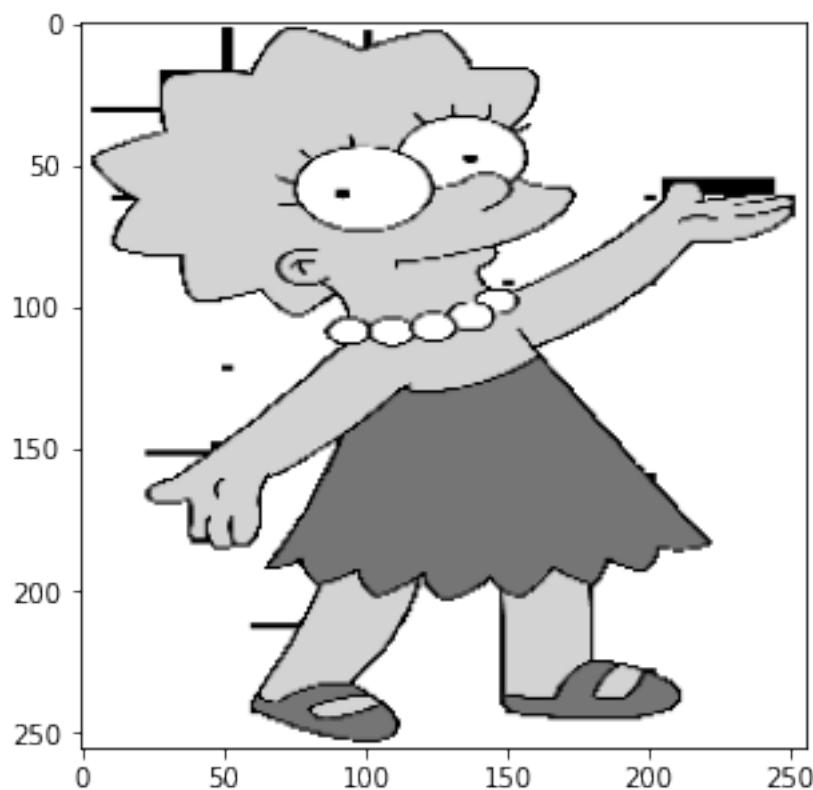
```
[[149. 147. 142. ... 129. 130. 135.]  
 [199. 198. 196. ... 167. 151. 145.]  
 [196. 198. 200. ... 220. 199. 191.]  
 ...  
 [233. 180. 13. ... 248. 246. 195.]  
 [195. 66. 22. ... 246. 247. 211.]  
 [101. 21. 59. ... 250. 246. 212.]]
```



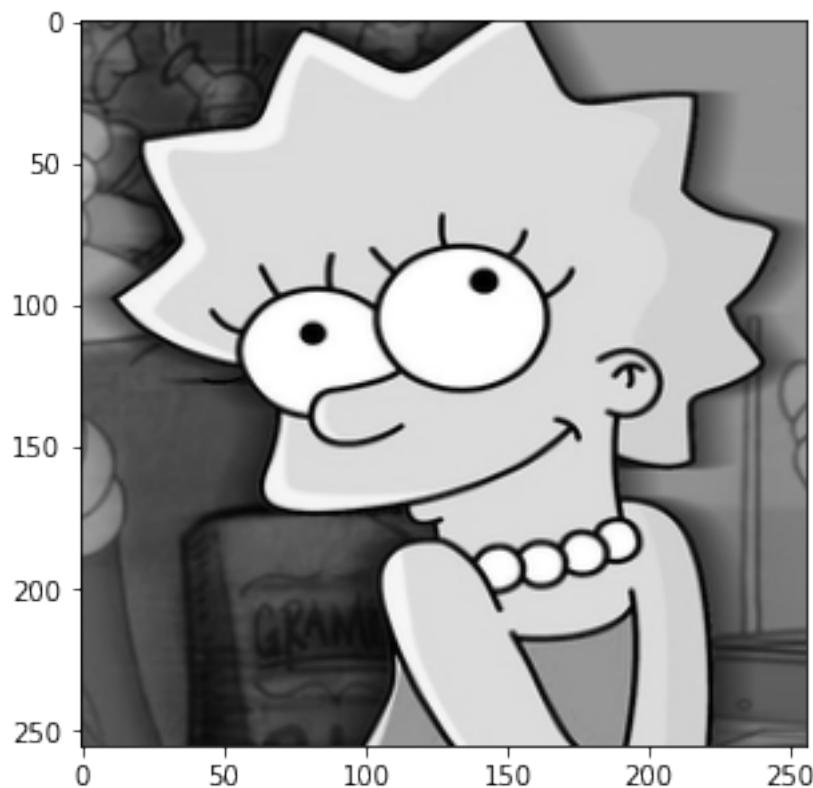
```
[[156. 156. 156. ... 183. 183. 183.]  
[156. 156. 156. ... 183. 183. 183.]  
[156. 156. 156. ... 183. 183. 183.]  
...  
[188. 188. 188. ... 188. 188. 188.]  
[188. 188. 188. ... 188. 188. 188.]  
[188. 188. 188. ... 188. 188. 188.]]
```



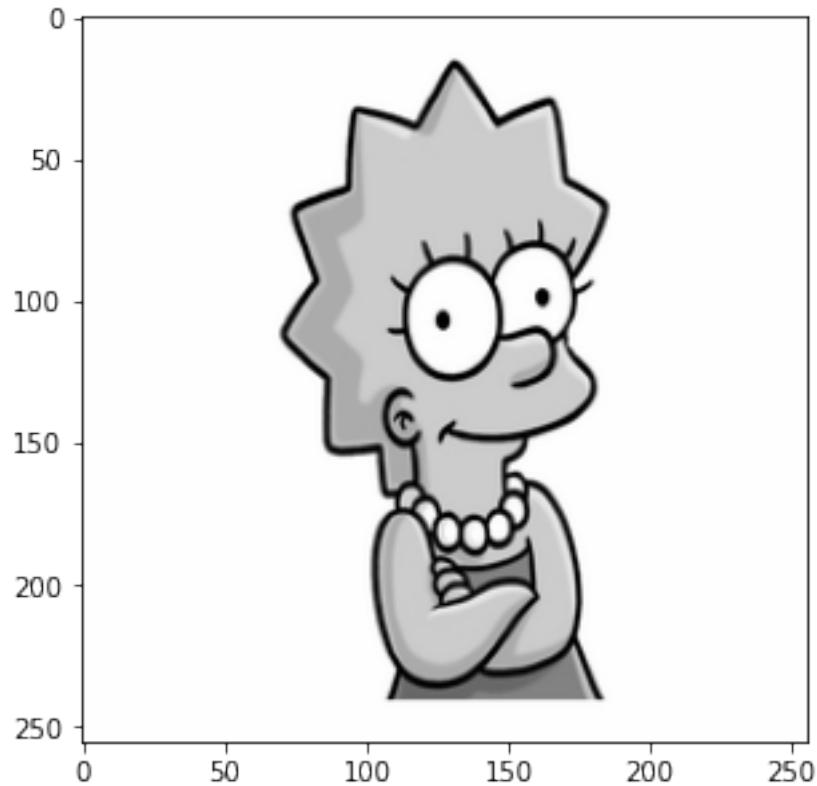
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



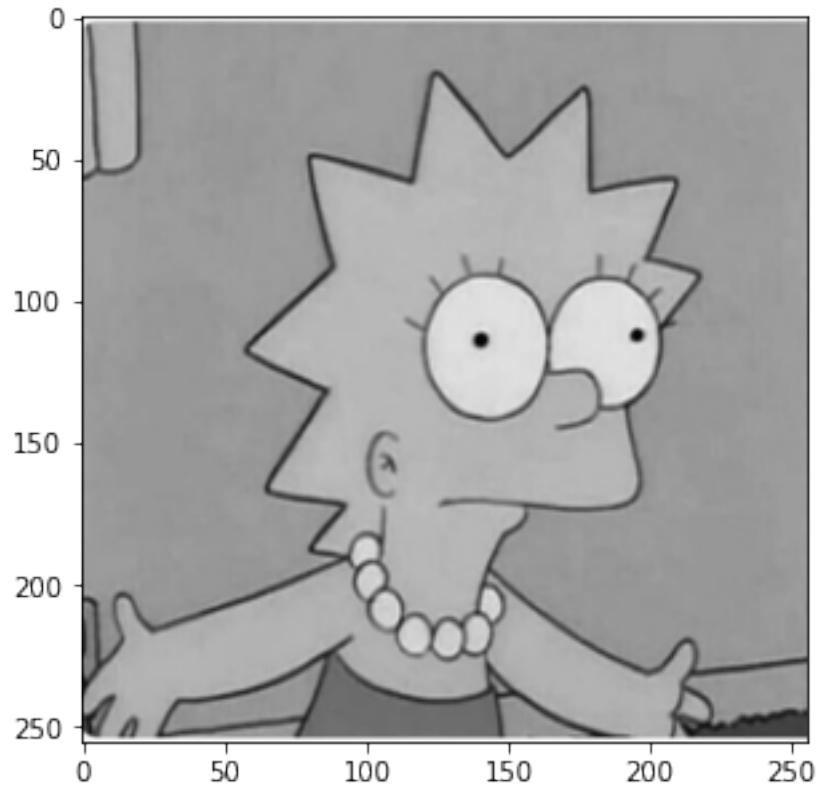
```
[[ 76.  83.  76. ... 153. 153. 153.]  
 [ 83. 100.  90. ... 153. 153. 153.]  
 [ 92.  99.  94. ... 153. 153. 153.]  
 ..  
 [ 81.  81.  84. ... 107. 105. 104.]  
 [ 79.  82.  82. ... 114. 112. 111.]  
 [ 73.  82.  83. ...  84.  83.  82.]]
```



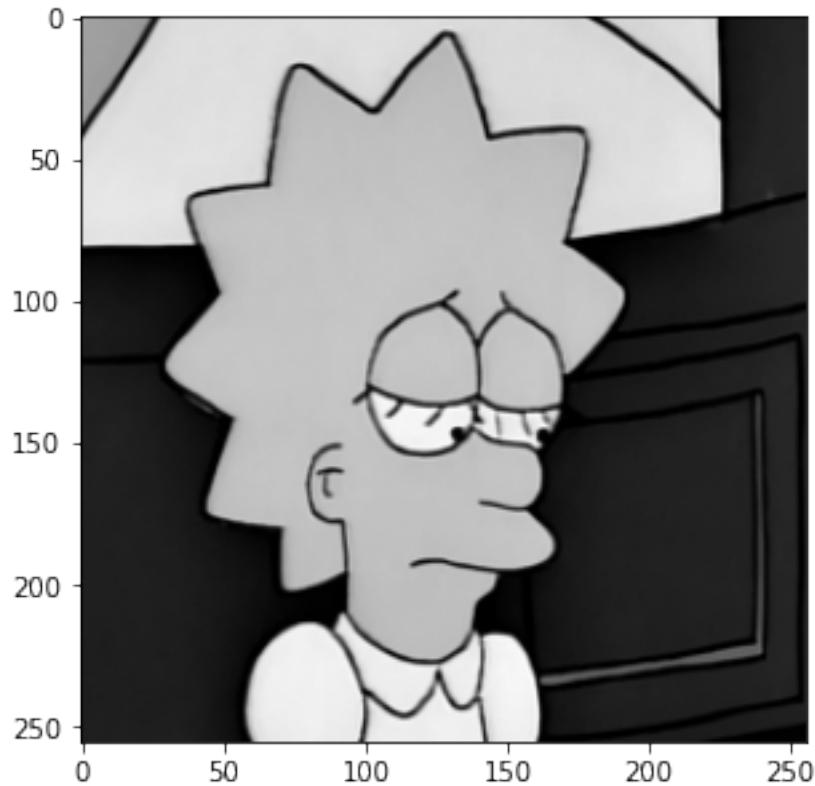
```
[[254. 254. 254. ... 254. 254. 254.]  
 [254. 254. 254. ... 254. 254. 254.]  
 [254. 254. 254. ... 254. 254. 254.]  
 ...  
 [254. 254. 254. ... 254. 254. 254.]  
 [254. 254. 254. ... 254. 254. 254.]  
 [254. 254. 254. ... 254. 254. 254.]]
```



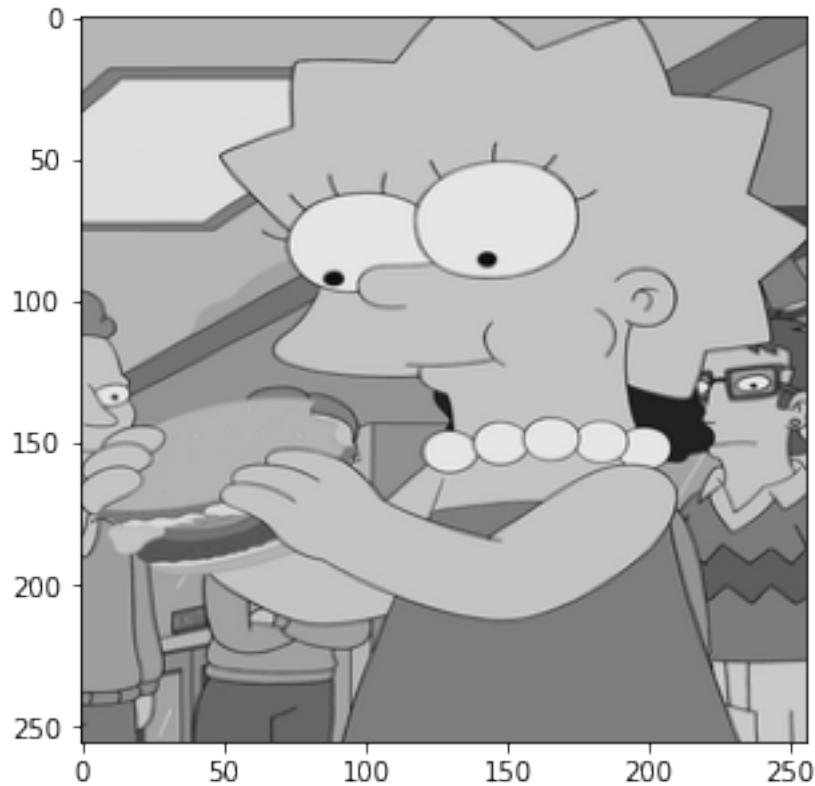
```
[[252. 252. 248. ... 252. 251. 251.]  
 [253. 250. 252. ... 241. 241. 242.]  
 [248. 251. 231. ... 175. 174. 173.]  
 ...  
 [252. 239. 177. ... 60. 59. 59.]  
 [248. 250. 251. ... 170. 170. 170.]  
 [251. 251. 248. ... 254. 253. 253.]]
```



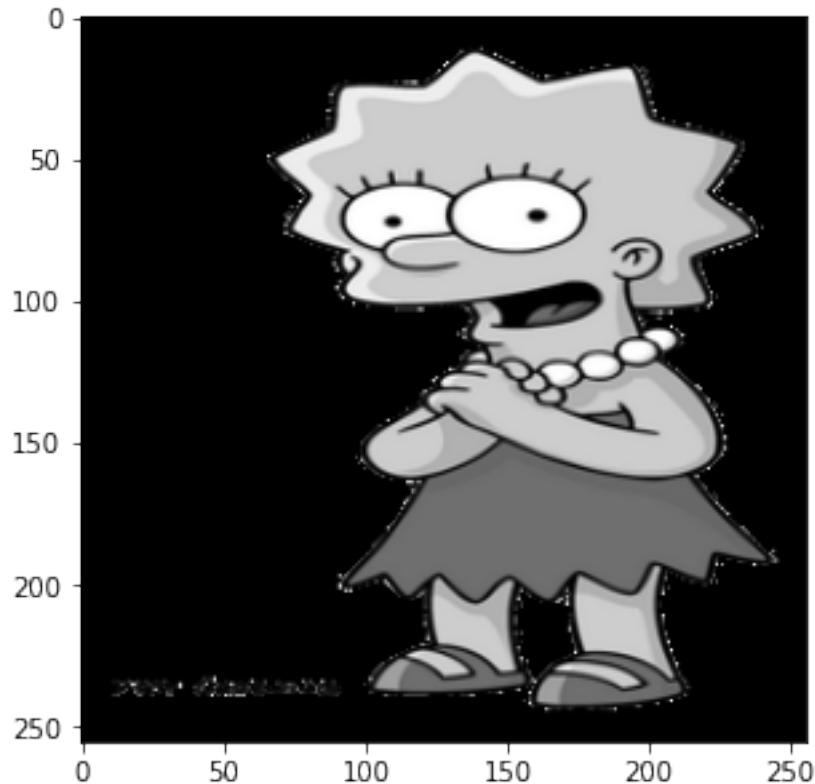
```
[[162. 163. 162. ... 37. 37. 37.]  
 [163. 163. 162. ... 37. 37. 37.]  
 [161. 162. 162. ... 37. 37. 37.]  
 ...  
 [ 32. 31. 31. ... 3. 2. 3.]  
 [ 31. 31. 31. ... 3. 3. 3.]  
 [ 32. 31. 30. ... 3. 3. 3.]]
```



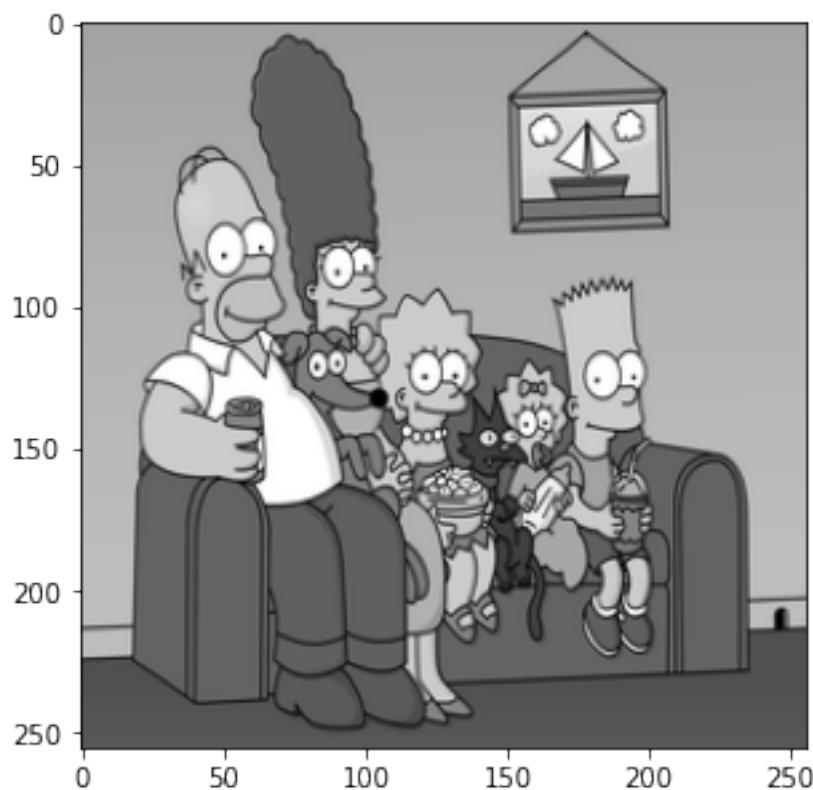
```
[[182.91139241 182.91139241 182.91139241 ... 115.12658228 115.12658228  
115.12658228]  
[182.91139241 182.91139241 182.91139241 ... 115.12658228 115.12658228  
115.12658228]  
[182.91139241 182.91139241 182.91139241 ... 115.12658228 115.12658228  
115.12658228]  
...  
[122.65822785 122.65822785 122.65822785 ... 200.12658228 202.27848101  
202.27848101]  
[122.65822785 122.65822785 122.65822785 ... 200.12658228 202.27848101  
202.27848101]  
[122.65822785 122.65822785 122.65822785 ... 200.12658228 202.27848101  
202.27848101]]
```



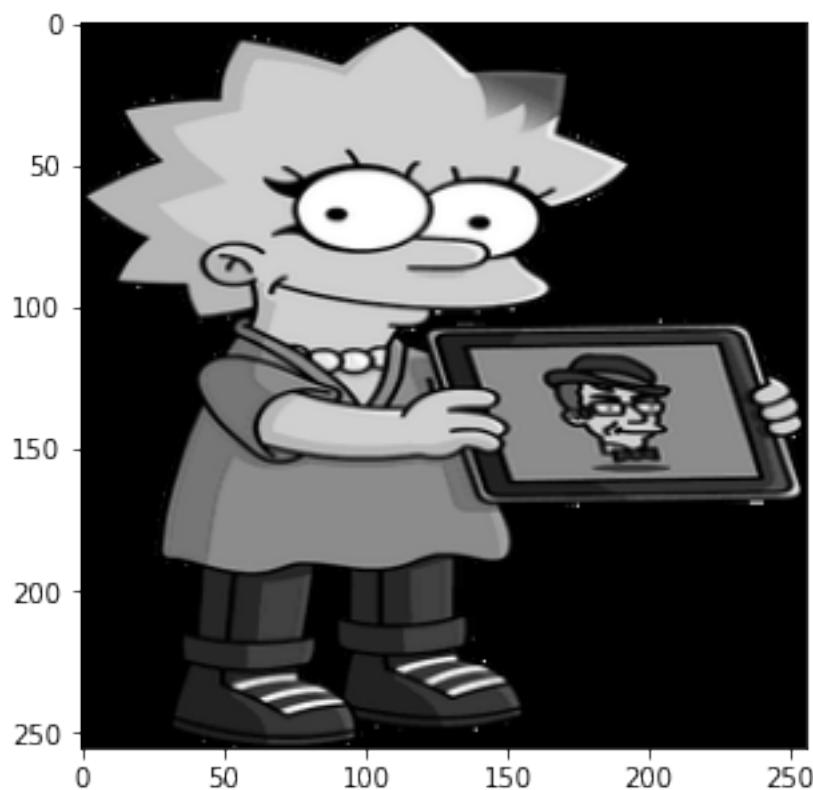
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



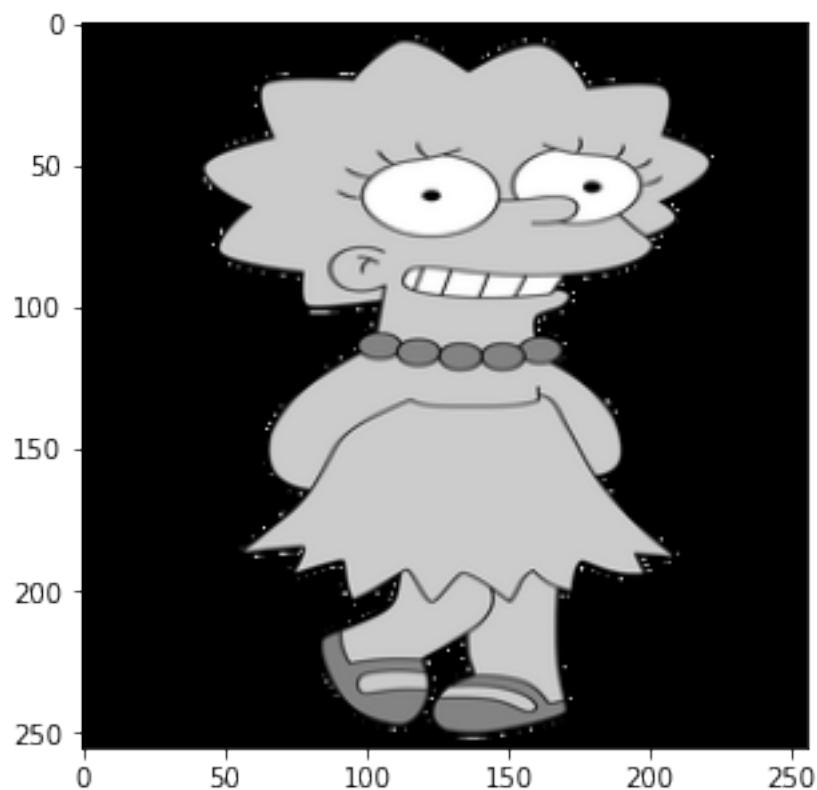
```
[[162. 162. 162. ... 164. 164. 164.]  
 [162. 162. 162. ... 164. 164. 164.]  
 [162. 162. 162. ... 164. 164. 164.]  
 ...  
 [ 92.  87.  88. ...  88.  88.  88.]  
 [ 93.  88.  88. ...  88.  88.  88.]  
 [ 93.  88.  88. ...  88.  88.  88.]]
```



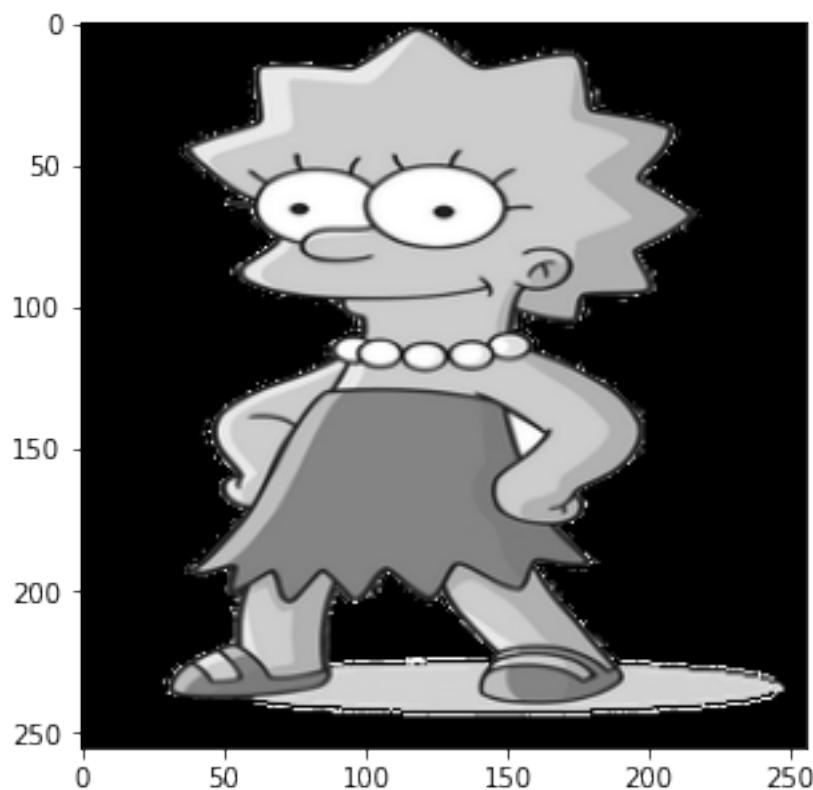
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



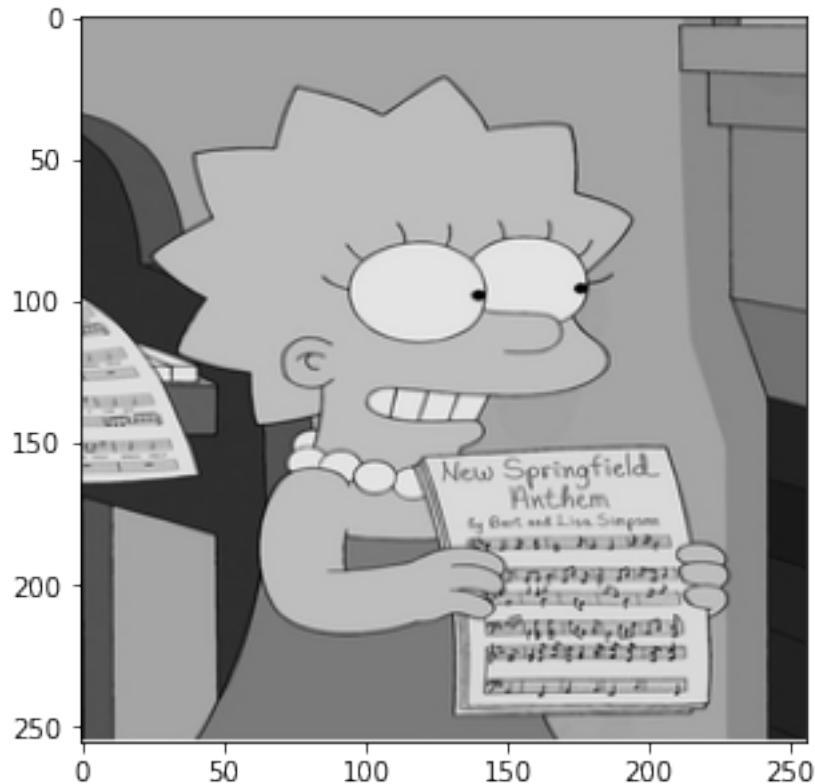
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



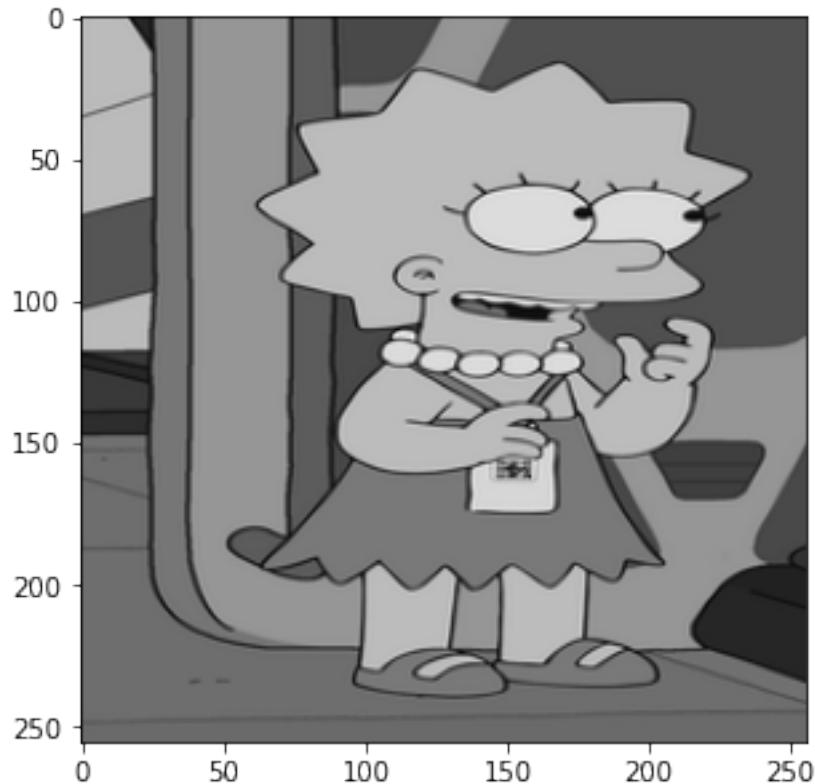
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



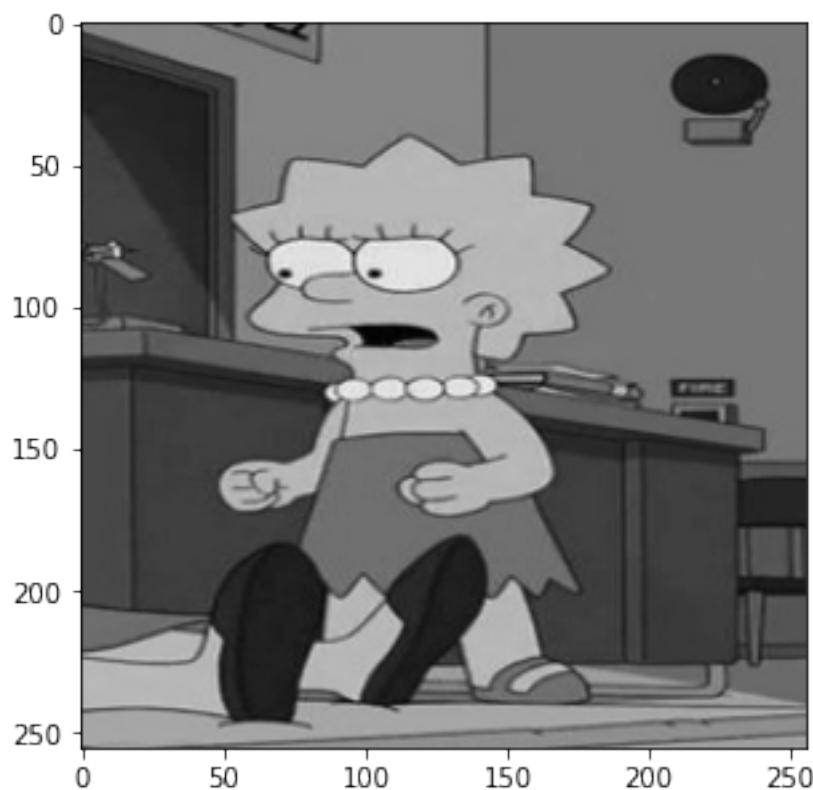
```
[[193. 193. 193. ... 199. 199. 199.]  
 [161. 161. 161. ... 167. 167. 167.]  
 [165. 165. 165. ... 182. 182. 182.]  
 ...  
 [ 38.  45.  91. ... 34.  37.  37.]  
 [ 37.  37.  84. ... 20.  25.  29.]  
 [209. 209. 216. ... 203. 200. 201.]]
```



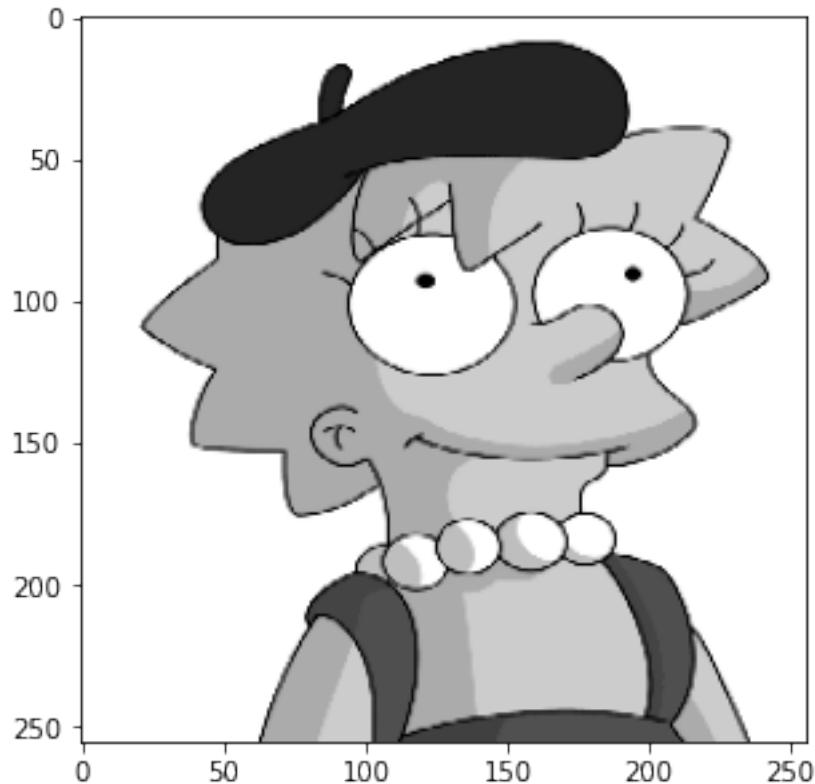
```
[[172.74193548 190.22177419 186.10887097 ... 80.2016129 80.2016129  
80.2016129 ]  
[173.77016129 190.22177419 186.10887097 ... 80.2016129 80.2016129  
80.2016129 ]  
[173.77016129 190.22177419 186.10887097 ... 80.2016129 80.2016129  
80.2016129 ]  
...  
[106.93548387 106.93548387 106.93548387 ... 106.93548387 106.93548387  
106.93548387]  
[106.93548387 106.93548387 106.93548387 ... 106.93548387 106.93548387  
106.93548387]  
[106.93548387 106.93548387 106.93548387 ... 106.93548387 106.93548387  
106.93548387]]
```



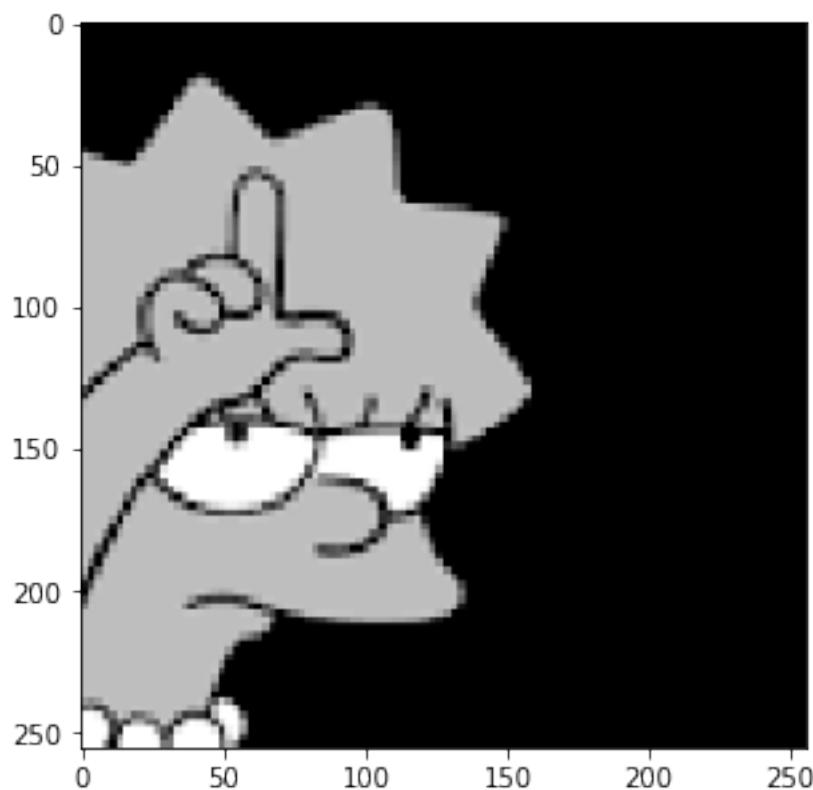
```
[[131.23430962 131.23430962 133.36820084 ... 119.49790795 119.49790795  
119.49790795]  
[135.50209205 130.16736402 132.30125523 ... 119.49790795 119.49790795  
119.49790795]  
[140.83682008 134.43514644 132.30125523 ... 119.49790795 119.49790795  
119.49790795]  
...  
[167.51046025 164.30962343 167.51046025 ... 157.90794979 157.90794979  
156.84100418]  
[167.51046025 167.51046025 169.64435146 ... 158.9748954 158.9748954  
158.9748954 ]  
[101.35983264 106.69456067 108.82845188 ... 158.9748954 158.9748954  
158.9748954 ]]
```



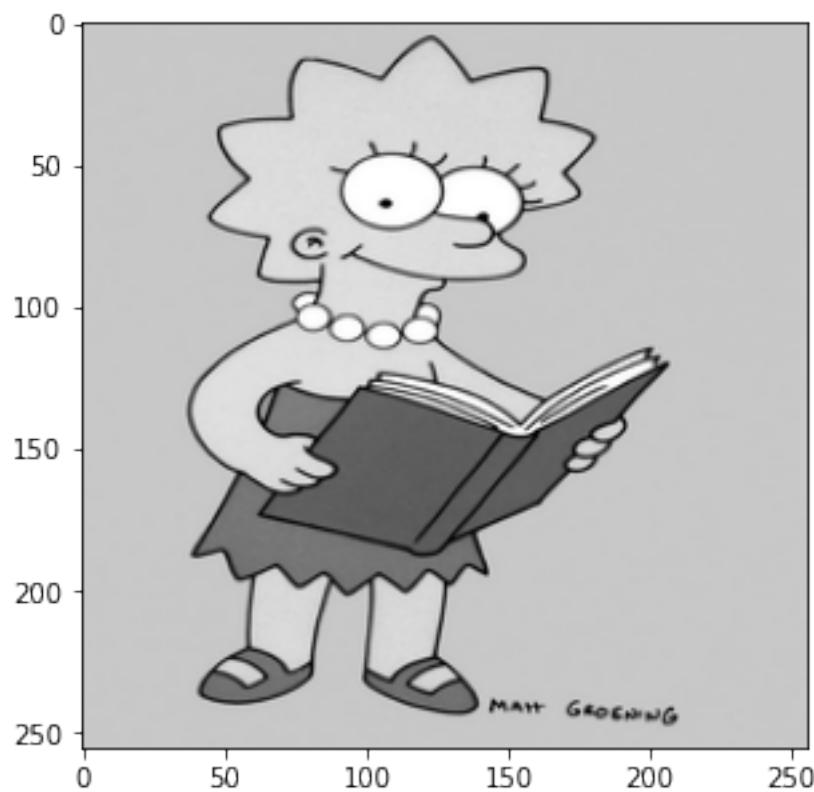
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



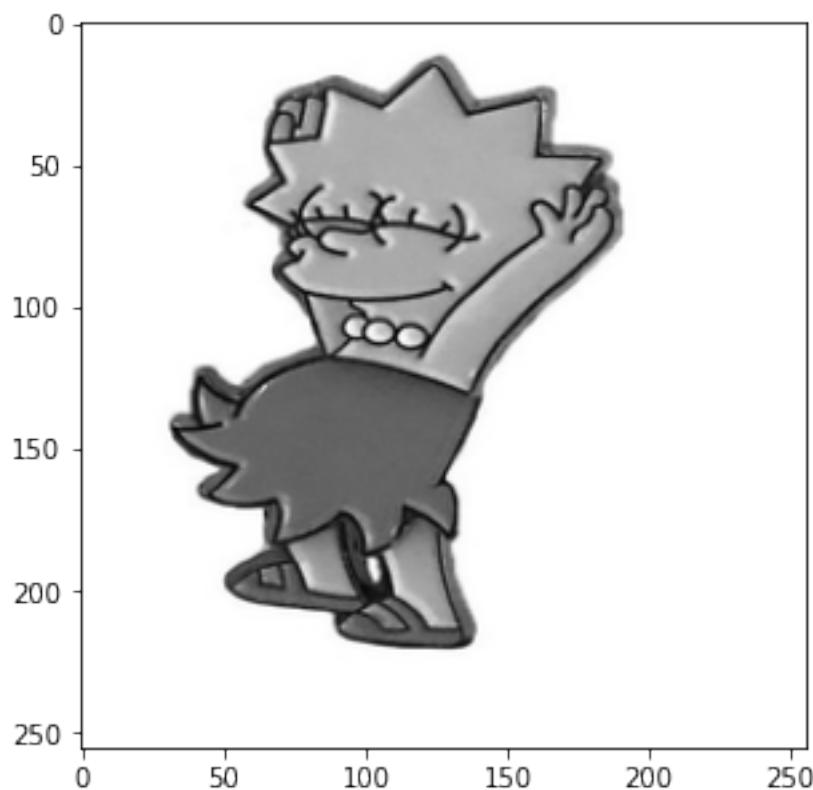
```
[[ 0.  0.  0. ... 0.  0.  0.]  
 [ 0.  0.  0. ... 0.  0.  0.]  
 [ 0.  0.  0. ... 0.  0.  0.]  
 ...  
 [255. 255. 255. ... 0.  0.  0.]  
 [255. 255. 255. ... 0.  0.  0.]  
 [255. 255. 255. ... 0.  0.  0.]]
```



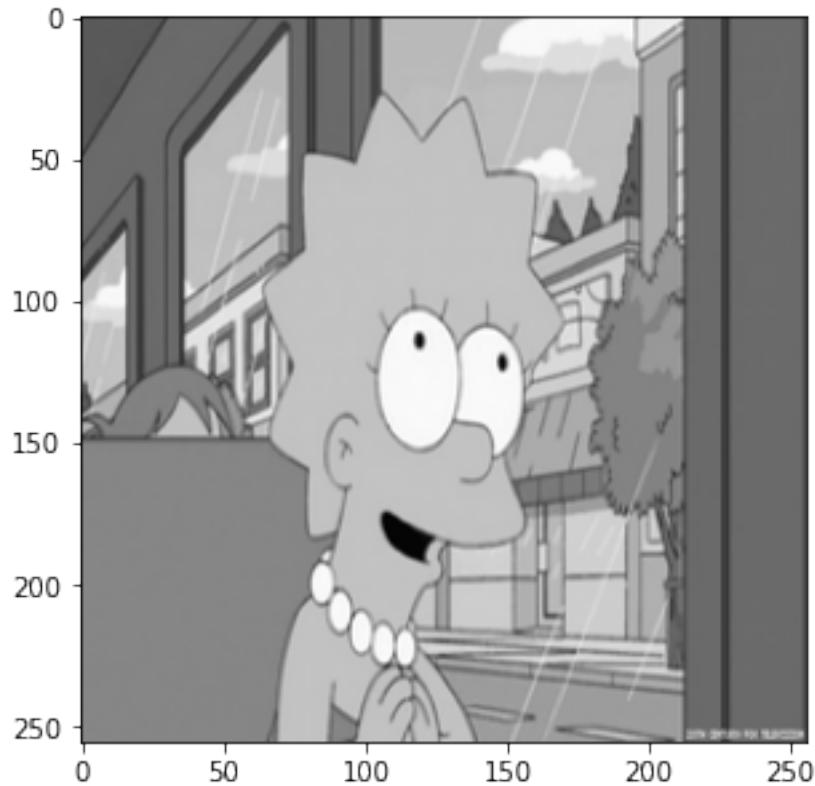
```
[[199. 199. 199. ... 199. 199. 199.]  
 [199. 199. 199. ... 199. 199. 199.]  
 [199. 199. 199. ... 199. 199. 200.]  
 ...  
 [199. 199. 199. ... 200. 200. 200.]  
 [200. 200. 200. ... 200. 200. 200.]  
 [199. 200. 200. ... 200. 200. 200.]]
```



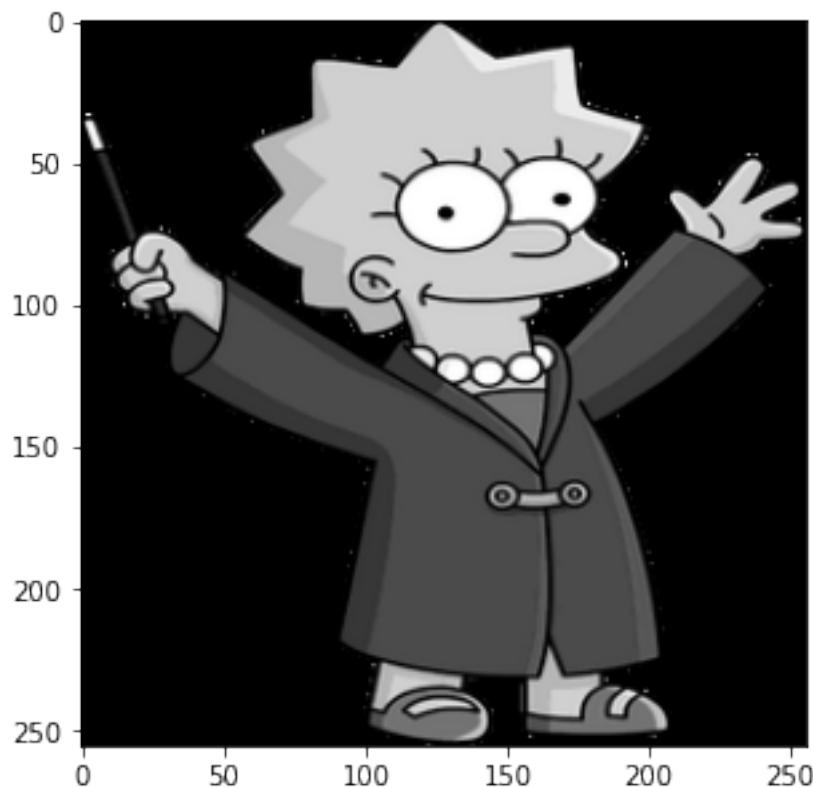
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



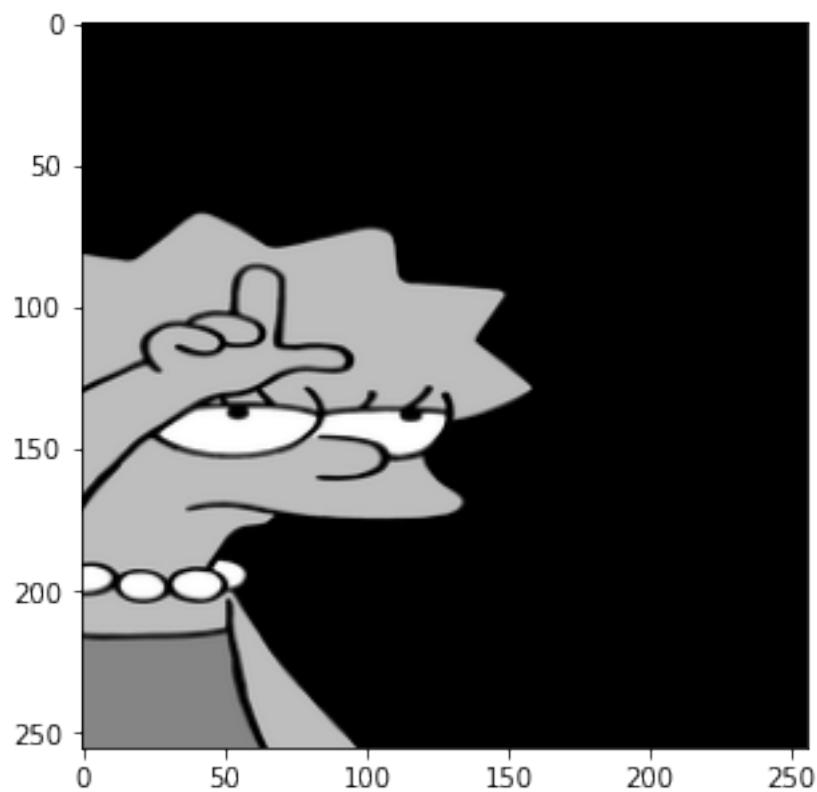
```
[[ 89.  90.  90. ... 108.  95.  72.]  
 [ 90.  90.  90. ... 108.  95.  72.]  
 [ 90.  90.  90. ... 108.  94.  72.]  
 ...  
 [133. 134. 134. ... 183. 185.  90.]  
 [134. 134. 134. ... 149. 137.  86.]  
 [134. 134. 134. ... 104.  93.  72.]]
```



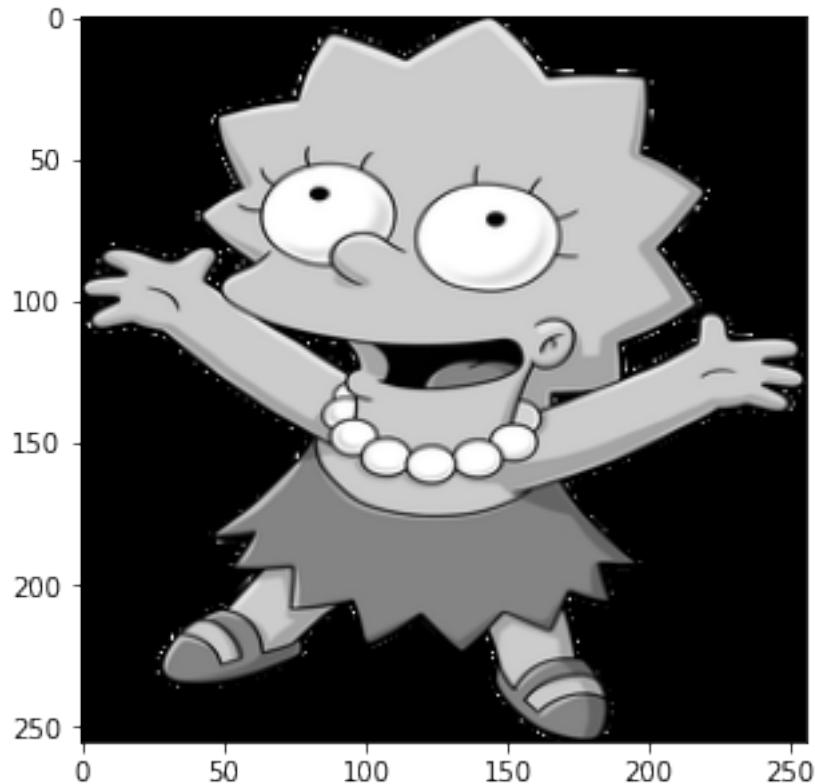
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



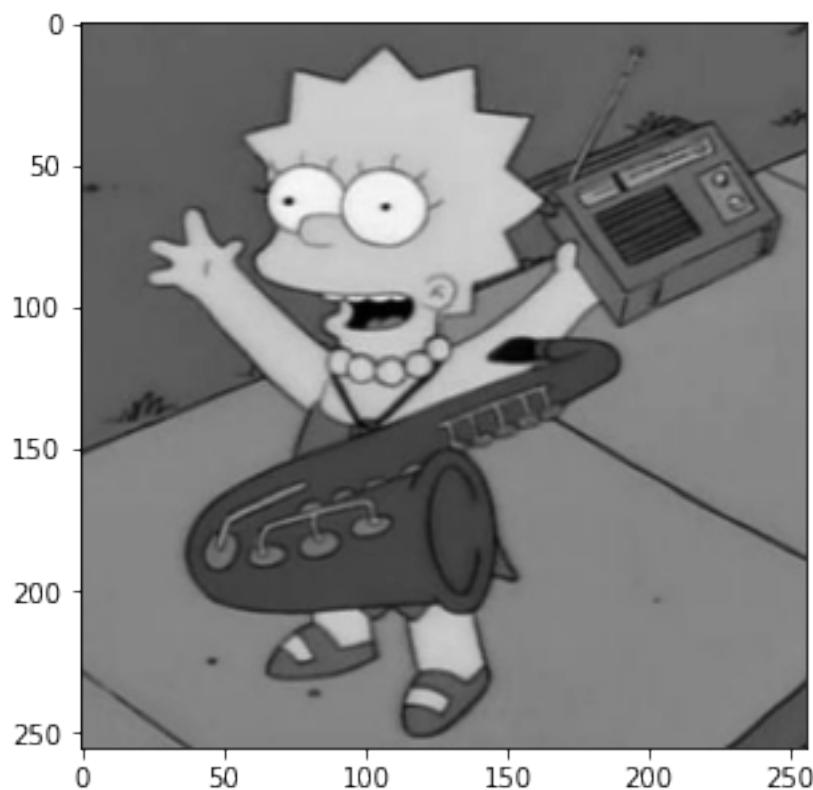
```
[[ 0.  0.  0. ... 0.  0.  0.]  
 [ 0.  0.  0. ... 0.  0.  0.]  
 [ 0.  0.  0. ... 0.  0.  0.]  
 ...  
 [133. 133. 133. ... 0.  0.  0.]  
 [133. 133. 133. ... 0.  0.  0.]  
 [133. 133. 133. ... 0.  0.  0.]]
```



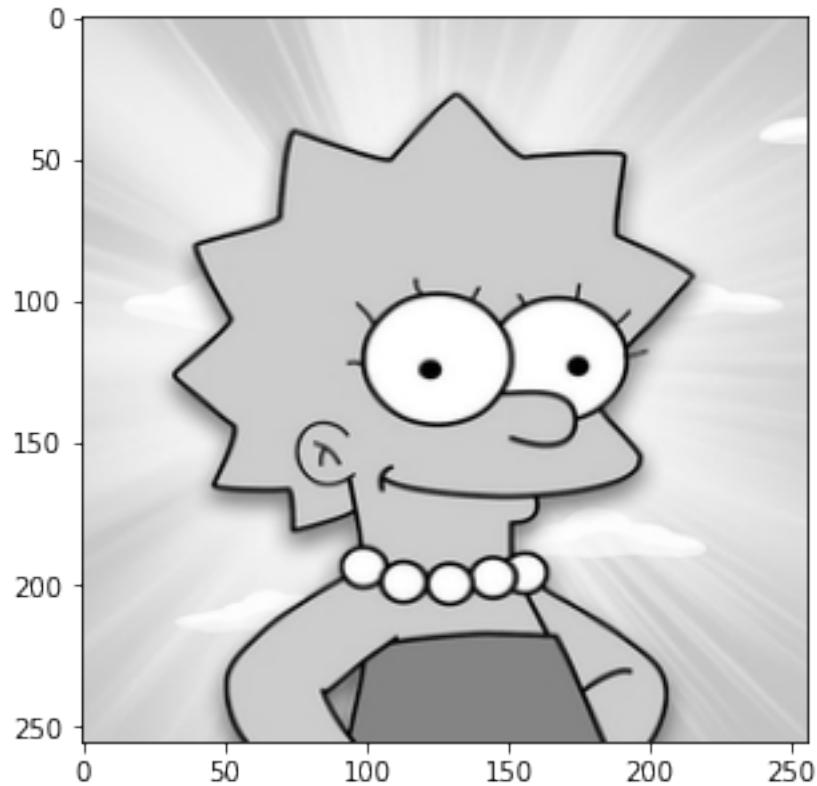
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



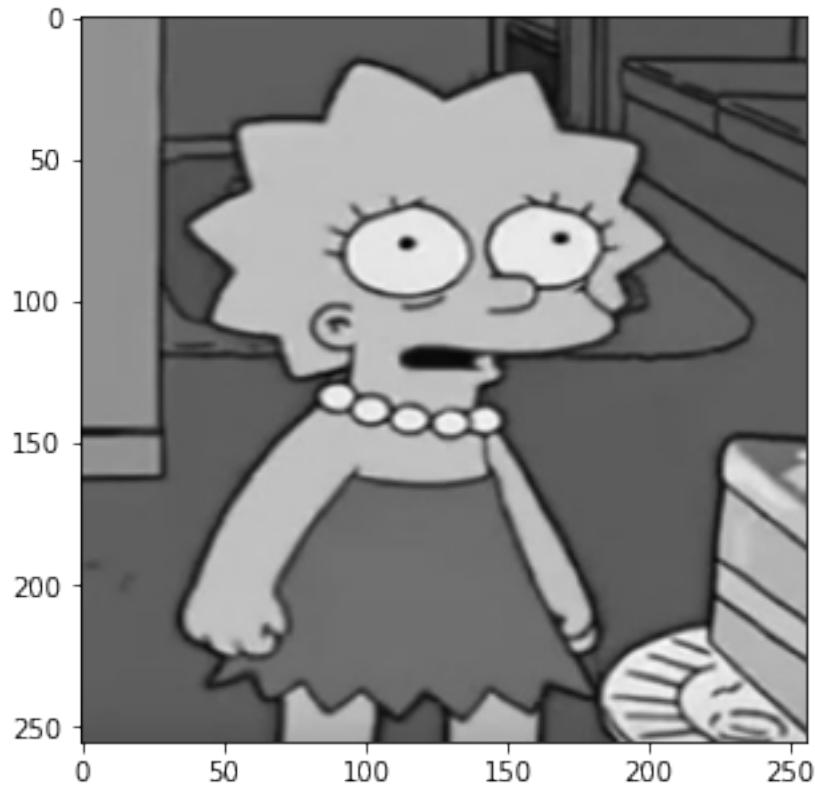
```
[[ 99.16666667  99.16666667  96.80555556 ...  97.98611111  94.44444444
  79.09722222]
 [ 99.16666667  99.16666667  96.80555556 ...  82.63888889  63.75
  43.68055556]
 [ 99.16666667  99.16666667  96.80555556 ...  50.76388889  43.68055556
  46.04166667]
...
[131.04166667 131.04166667 131.04166667 ...  80.27777778  86.18055556
 82.63888889]
[131.04166667 131.04166667 131.04166667 ...  82.63888889  75.55555556
 54.30555556]
[131.04166667 131.04166667 131.04166667 ...  70.83333333  50.76388889
 25.97222222]]
```



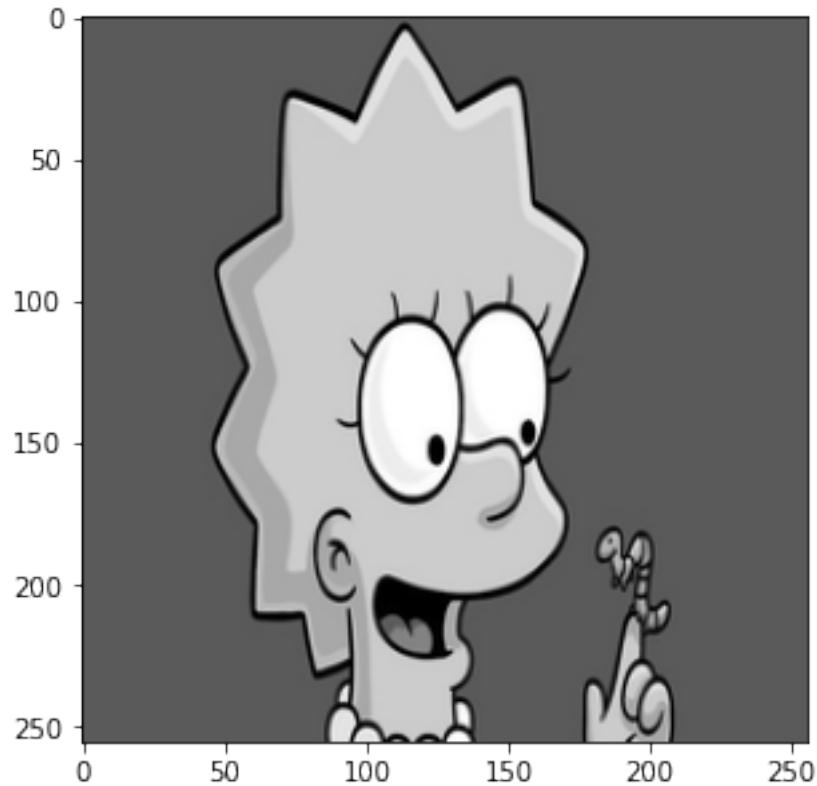
```
[[226. 226. 226. ... 200. 199. 199.]  
 [226. 226. 226. ... 199. 199. 199.]  
 [227. 227. 227. ... 199. 199. 199.]  
 ...  
 [204. 205. 207. ... 201. 205. 207.]  
 [205. 206. 207. ... 198. 200. 205.]  
 [205. 207. 206. ... 197. 198. 200.]]
```



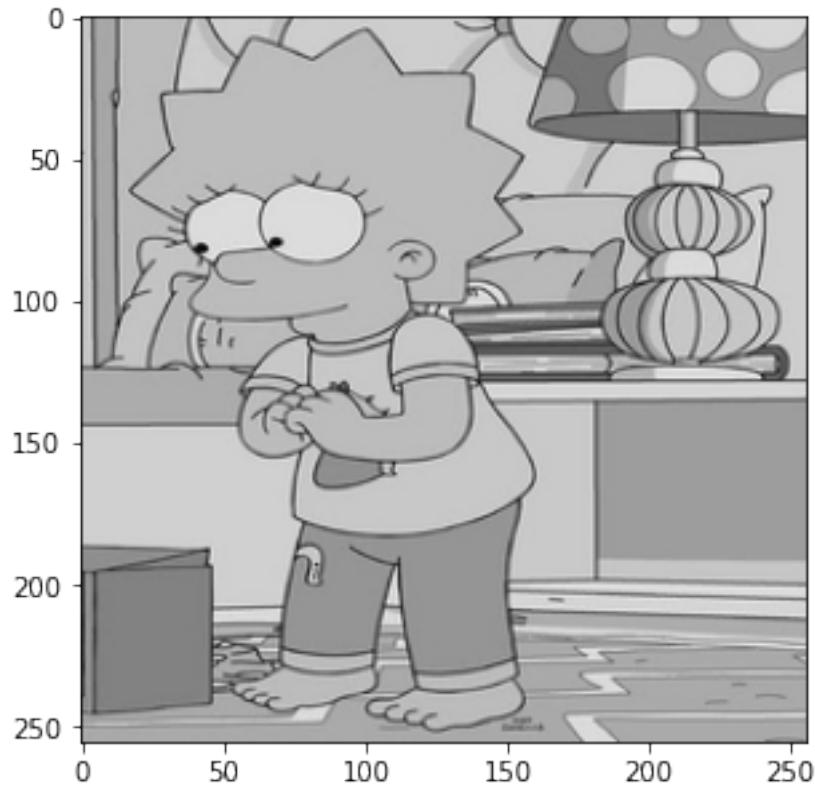
```
[[145.86 145.86 145.86 ... 70.38 87.72 90.78]
 [145.86 145.86 145.86 ... 71.4   89.76 89.76]
 [145.86 145.86 145.86 ... 73.44 91.8   88.74]
 ...
 [ 86.7   86.7   86.7   ... 219.3  220.32 165.24]
 [ 85.68  85.68  85.68 ... 213.18 216.24 213.18]
 [ 85.68  85.68  85.68 ... 212.16 209.1  218.28]]
```



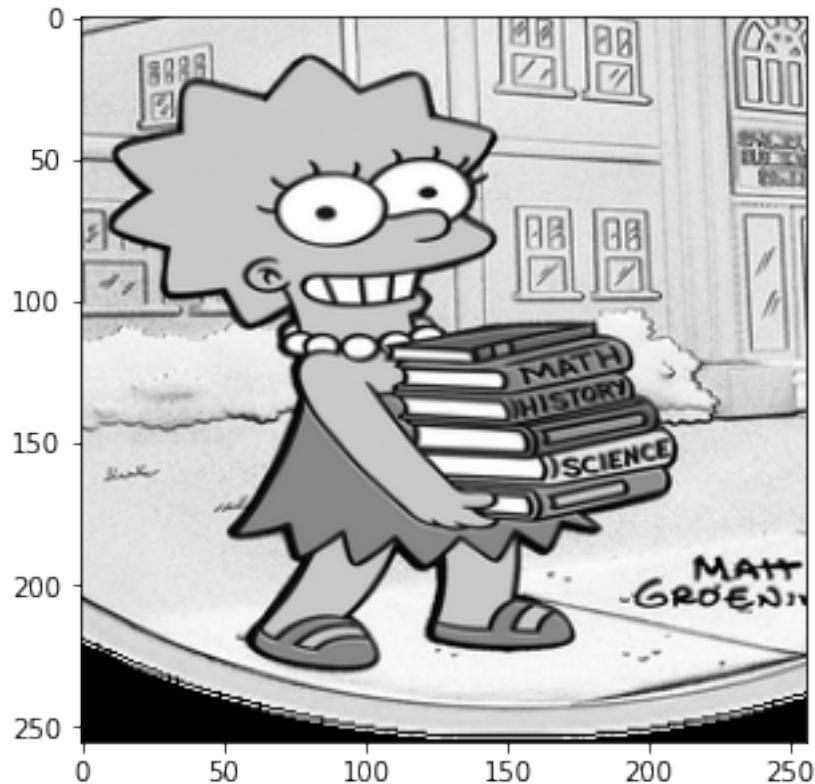
```
[[90. 90. 90. ... 90. 90. 90.]  
 [90. 90. 90. ... 90. 90. 90.]  
 [90. 90. 90. ... 90. 90. 90.]  
 ...  
 [90. 90. 90. ... 90. 90. 90.]  
 [90. 90. 90. ... 90. 90. 90.]  
 [90. 90. 90. ... 90. 90. 90.]]
```



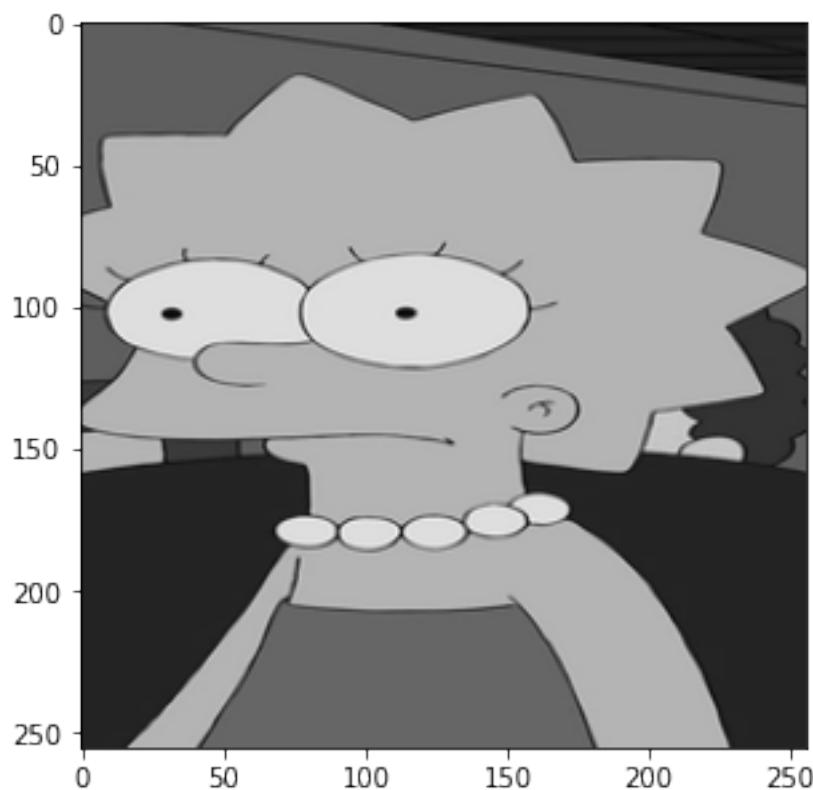
```
[[199.25101215 197.18623482 198.21862348 ... 214.73684211 216.80161943  
212.67206478]  
[199.25101215 197.18623482 198.21862348 ... 214.73684211 213.70445344  
221.96356275]  
[199.25101215 197.18623482 198.21862348 ... 217.8340081 212.67206478  
222.99595142]  
...  
[169.31174089 172.40890688 171.37651822 ... 172.40890688 170.34412955  
170.34412955]  
[170.34412955 171.37651822 170.34412955 ... 163.11740891 170.34412955  
173.4412955]  
[170.34412955 171.37651822 169.31174089 ... 129.048583 122.85425101  
140.4048583 ]]
```



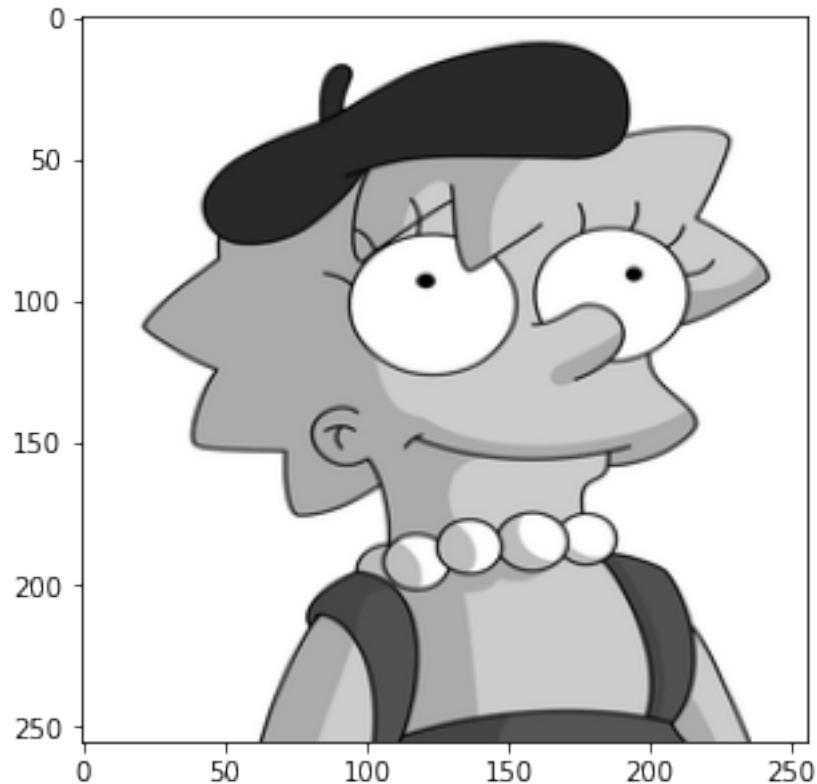
```
[[ 90. 130. 171. ... 255. 212. 120.]  
 [208. 170. 182. ... 129. 241. 218.]  
 [224. 233. 235. ... 54. 155. 249.]  
 ...  
 [ 0.  0.  0. ...  0.  0.  0.]  
 [ 0.  0.  0. ...  0.  0.  0.]  
 [ 0.  0.  0. ...  0.  0.  0.]]
```



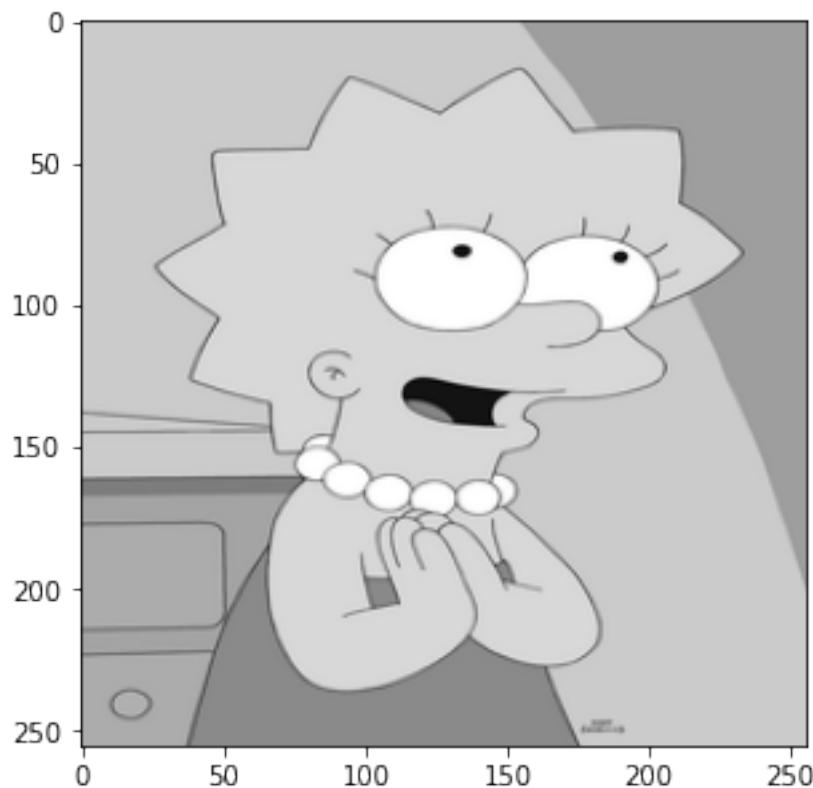
```
[[94.83471074 94.83471074 94.83471074 ... 20.02066116 20.02066116  
20.02066116]  
[94.83471074 94.83471074 94.83471074 ... 25.2892562 25.2892562  
25.2892562 ]  
[94.83471074 94.83471074 94.83471074 ... 32.66528926 32.66528926  
32.66528926]  
...  
[35.82644628 35.82644628 35.82644628 ... 35.82644628 35.82644628  
35.82644628]  
[35.82644628 35.82644628 35.82644628 ... 35.82644628 35.82644628  
35.82644628]  
[35.82644628 35.82644628 35.82644628 ... 35.82644628 35.82644628  
35.82644628]]
```



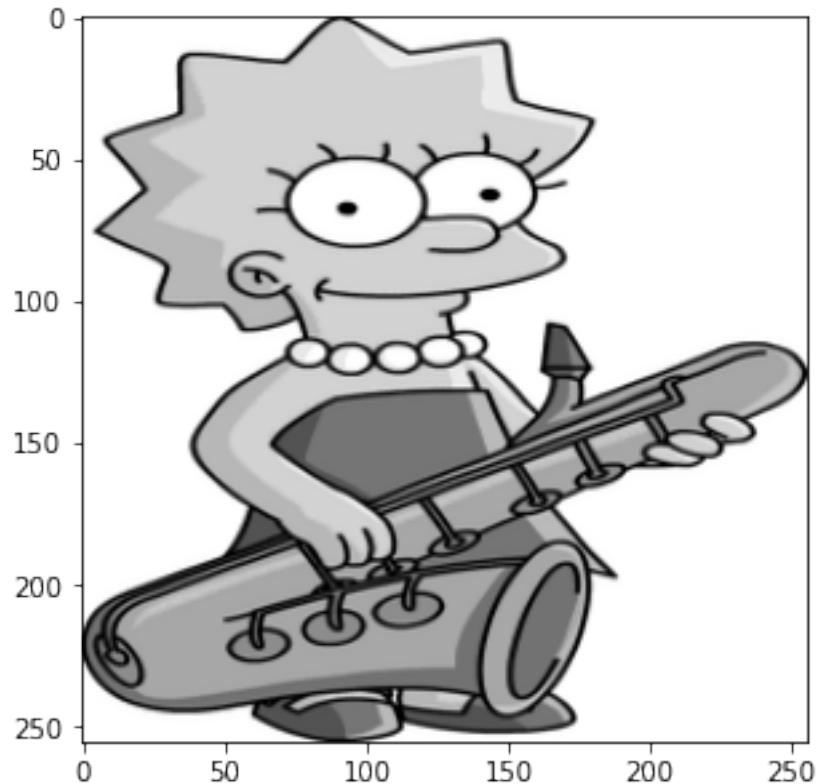
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



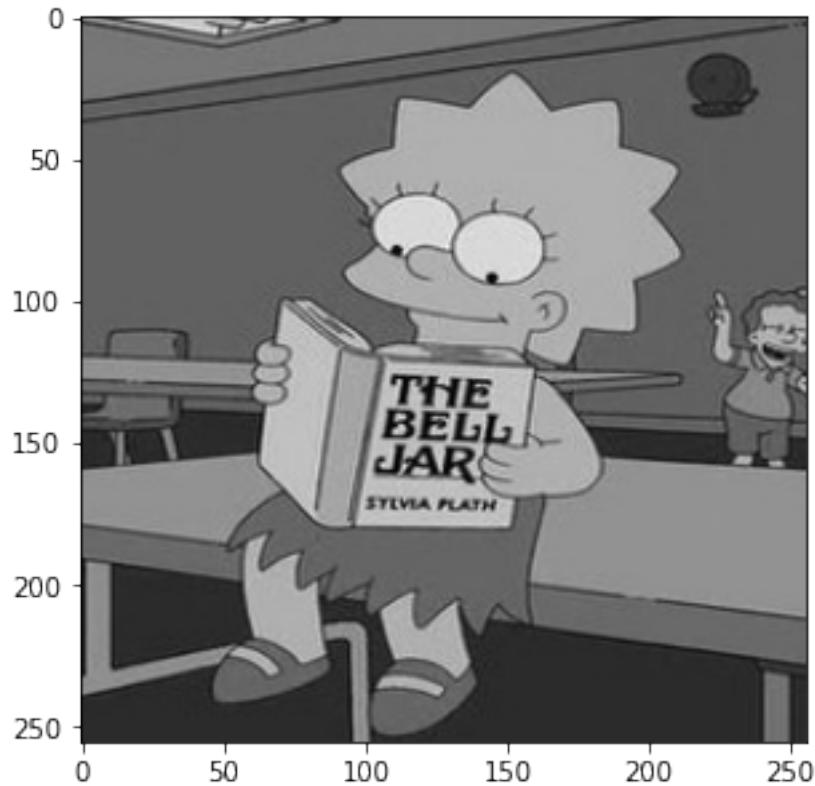
```
[[203. 203. 203. ... 158. 158. 158.]  
 [203. 203. 203. ... 158. 158. 158.]  
 [203. 203. 203. ... 158. 158. 158.]  
 ...  
 [172. 172. 172. ... 203. 203. 203.]  
 [172. 172. 172. ... 203. 203. 203.]  
 [172. 172. 172. ... 203. 203. 203.]]
```



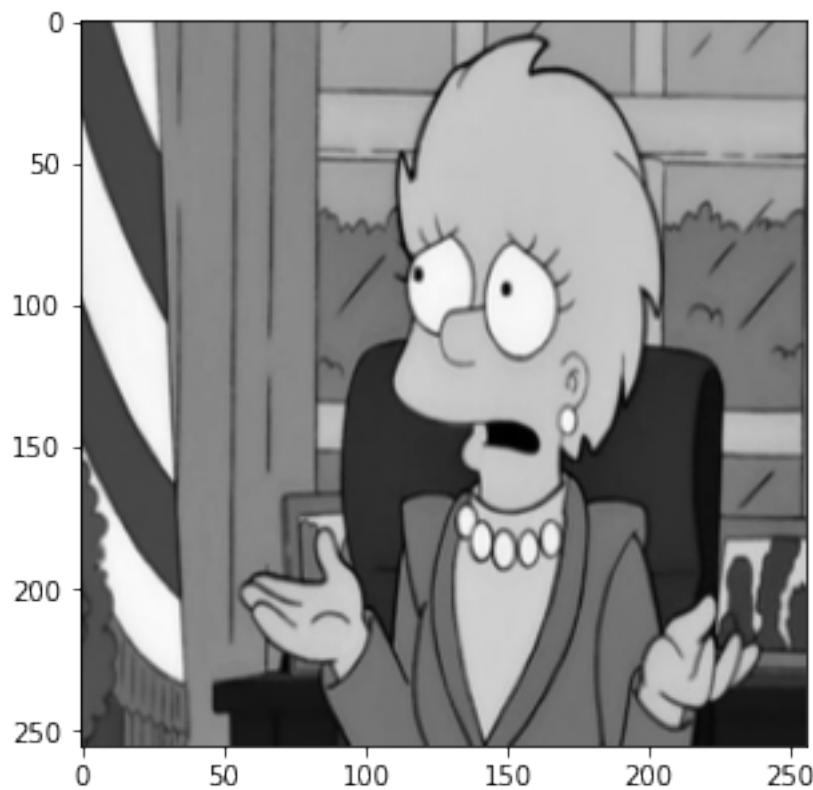
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



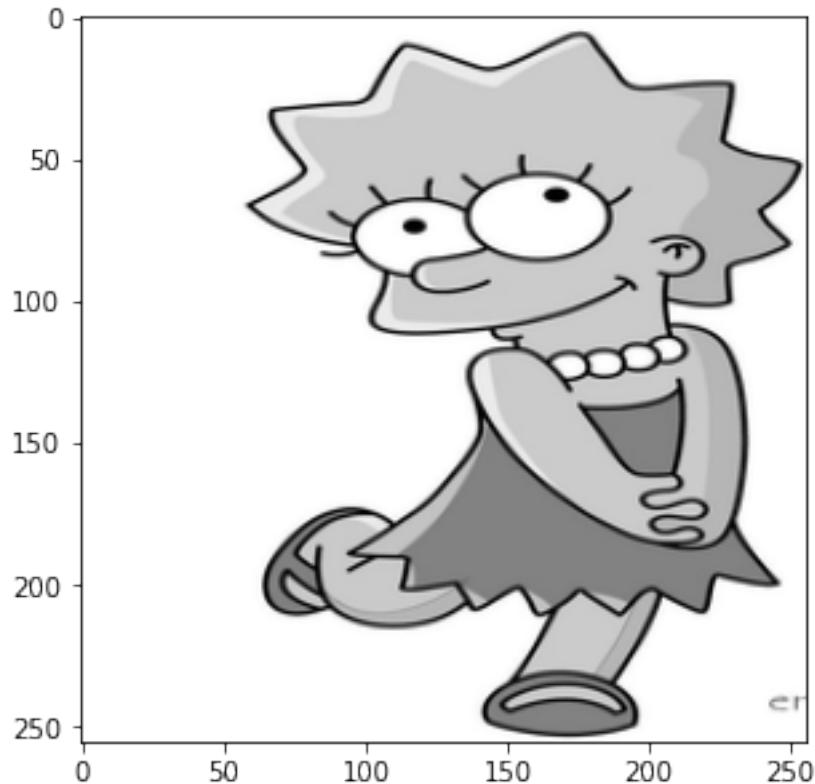
```
[[148.32669323 144.26294821 140.19920319 ... 101.5936255 95.49800797
 94.48207171]
[119.88047809 133.0876494 142.2310757 ... 89.40239044 97.52988048
 102.60956175]
[ 43.68525896 54.86055777 66.03585657 ... 94.48207171 74.16334661
 56.89243028]
...
[ 43.68525896 43.68525896 43.68525896 ... 41.65338645 43.68525896
 44.70119522]
[ 43.68525896 43.68525896 43.68525896 ... 41.65338645 43.68525896
 44.70119522]
[ 43.68525896 43.68525896 43.68525896 ... 41.65338645 43.68525896
 44.70119522]]
```



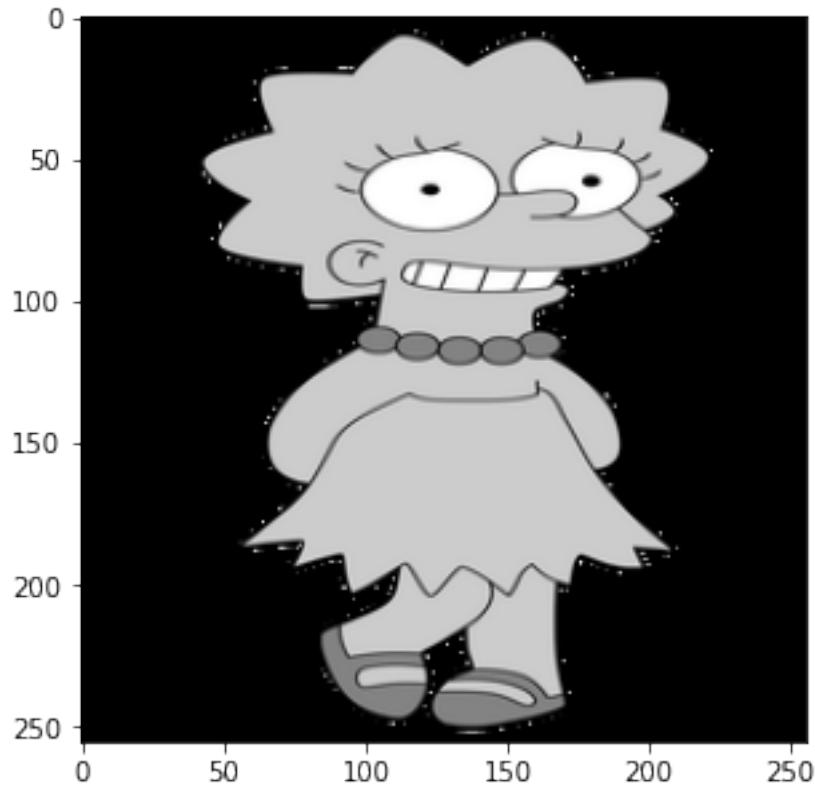
```
[[ 69.0625  70.125   69.0625 ... 161.5     160.4375 154.0625]
 [ 69.0625  70.125   69.0625 ... 162.5625 160.4375 154.0625]
 [ 69.0625  70.125   69.0625 ... 162.5625 161.5     155.125 ]
 ...
 [ 80.75     59.5     39.3125 ... 22.3125  22.3125  22.3125]
 [ 81.8125  47.8125  35.0625 ... 22.3125  22.3125  22.3125]
 [ 80.75     53.125   43.5625 ... 23.375    23.375    23.375 ]]
```



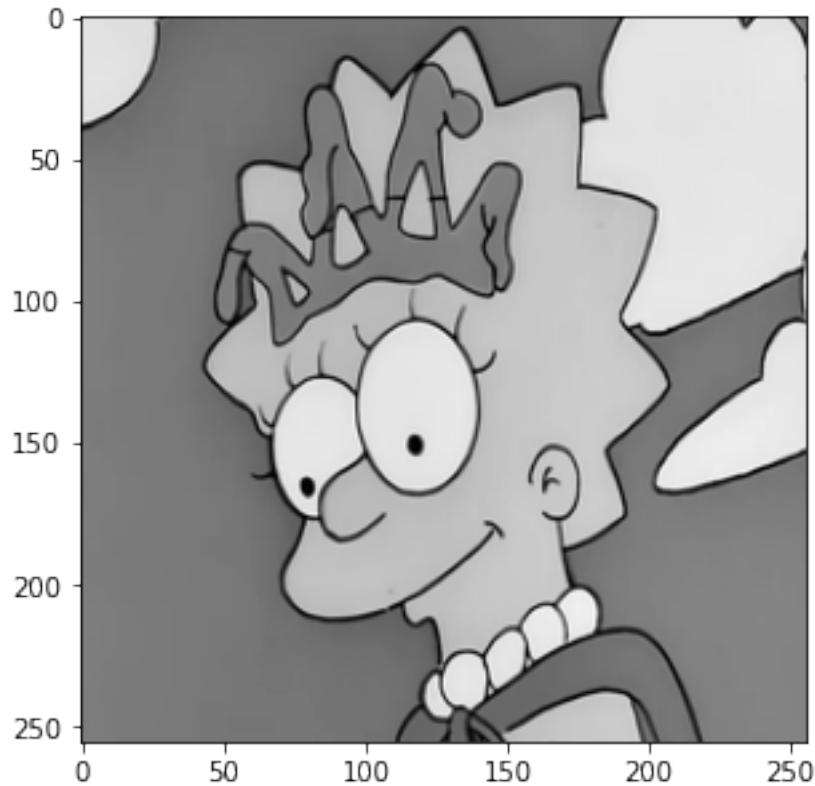
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



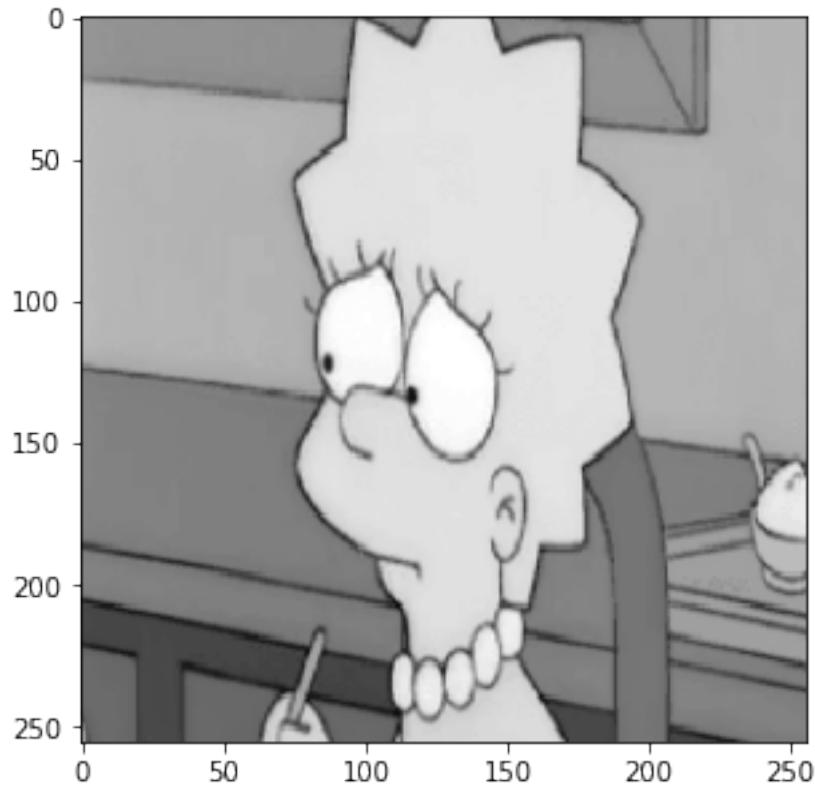
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



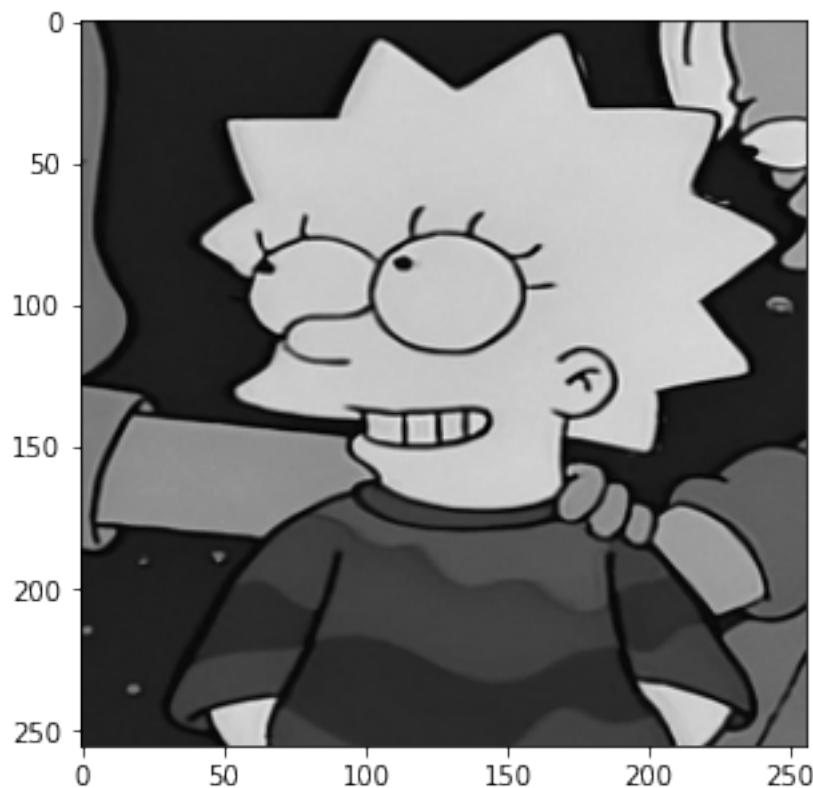
```
[[227.60330579 227.60330579 226.54958678 ... 115.90909091 118.01652893  
119.07024793]  
[227.60330579 227.60330579 226.54958678 ... 121.17768595 116.96280992  
119.07024793]  
[227.60330579 227.60330579 226.54958678 ... 141.19834711 116.96280992  
118.01652893]  
...  
[125.39256198 125.39256198 125.39256198 ... 131.71487603 131.71487603  
131.71487603]  
[125.39256198 125.39256198 125.39256198 ... 131.71487603 131.71487603  
131.71487603]  
[125.39256198 125.39256198 125.39256198 ... 131.71487603 131.71487603  
131.71487603]]
```



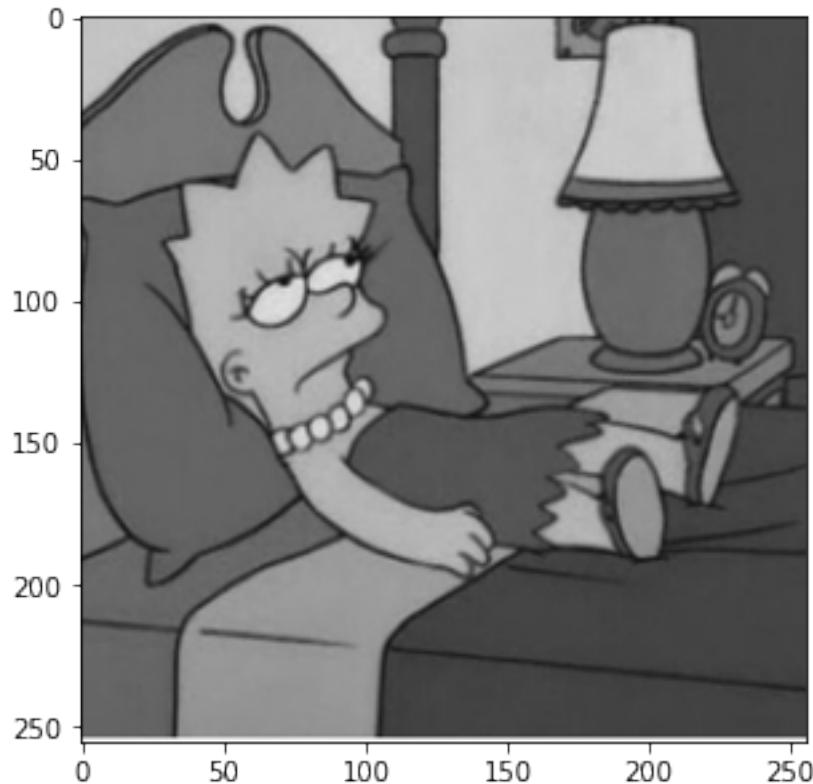
```
[[149.175 149.175 149.175 ... 181.05 178.5 178.5 ]  
 [149.175 149.175 149.175 ... 181.05 179.775 178.5 ]  
 [149.175 149.175 149.175 ... 179.775 179.775 178.5 ]  
 ...  
 [160.65 163.2 91.8 ... 127.5 127.5 127.5 ]  
 [160.65 163.2 91.8 ... 124.95 127.5 127.5 ]  
 [160.65 163.2 91.8 ... 124.95 127.5 124.95 ]]
```



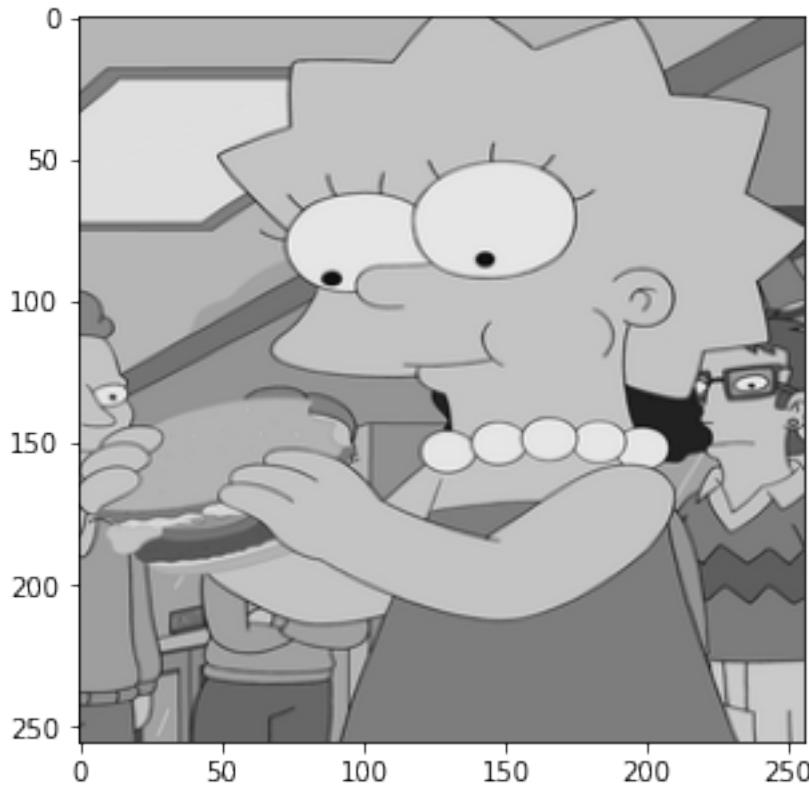
```
[[112.0766129 112.0766129 115.16129032 ... 154.23387097 154.23387097  
155.26209677]  
[112.0766129 112.0766129 113.10483871 ... 156.29032258 153.20564516  
155.26209677]  
[112.0766129 113.10483871 112.0766129 ... 155.26209677 155.26209677  
155.26209677]  
...  
[ 34.95967742 34.95967742 33.93145161 ... 108.99193548 111.0483871  
111.0483871 ]  
[ 34.95967742 33.93145161 34.95967742 ... 110.02016129 111.0483871  
111.0483871 ]  
[ 33.93145161 33.93145161 33.93145161 ... 111.0483871 112.0766129  
112.0766129 ]]
```



```
[[194. 194. 195. ... 72. 72. 72.]  
 [195. 195. 195. ... 72. 72. 72.]  
 [195. 195. 194. ... 72. 72. 72.]  
 ...  
 [ 89.  89.  90. ... 52. 52. 52.]  
 [187. 187. 187. ... 175. 175. 174.]  
 [255. 255. 255. ... 255. 255. 255.]]
```

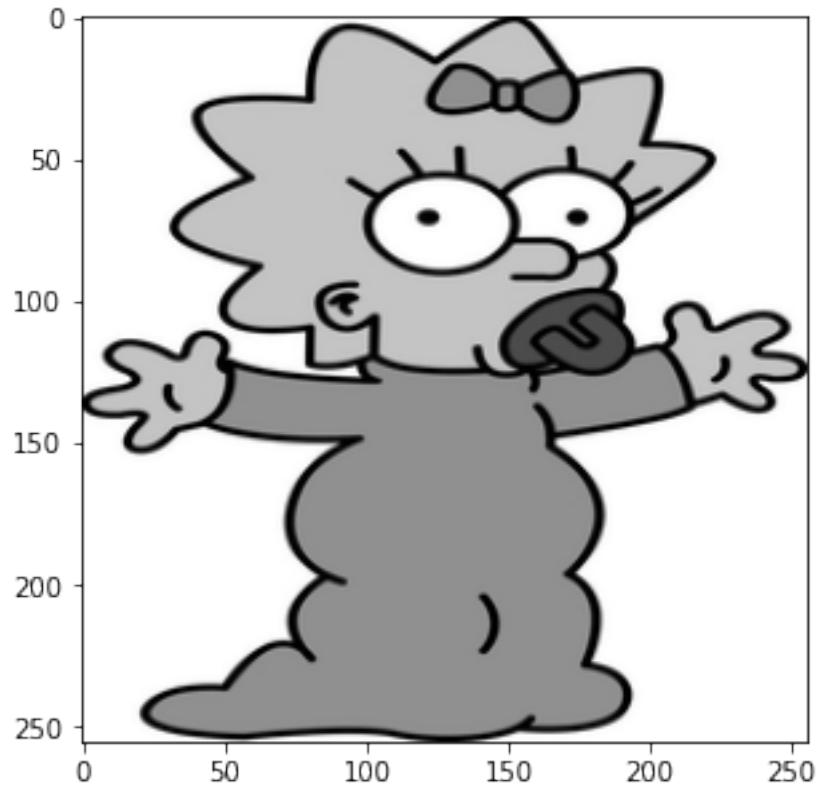


```
[[182.91139241 182.91139241 182.91139241 ... 115.12658228 115.12658228  
115.12658228]  
[182.91139241 182.91139241 182.91139241 ... 115.12658228 115.12658228  
115.12658228]  
[182.91139241 182.91139241 182.91139241 ... 115.12658228 115.12658228  
115.12658228]  
...  
[122.65822785 122.65822785 122.65822785 ... 200.12658228 202.27848101  
202.27848101]  
[122.65822785 122.65822785 122.65822785 ... 200.12658228 202.27848101  
202.27848101]  
[122.65822785 122.65822785 122.65822785 ... 200.12658228 202.27848101  
202.27848101]]
```

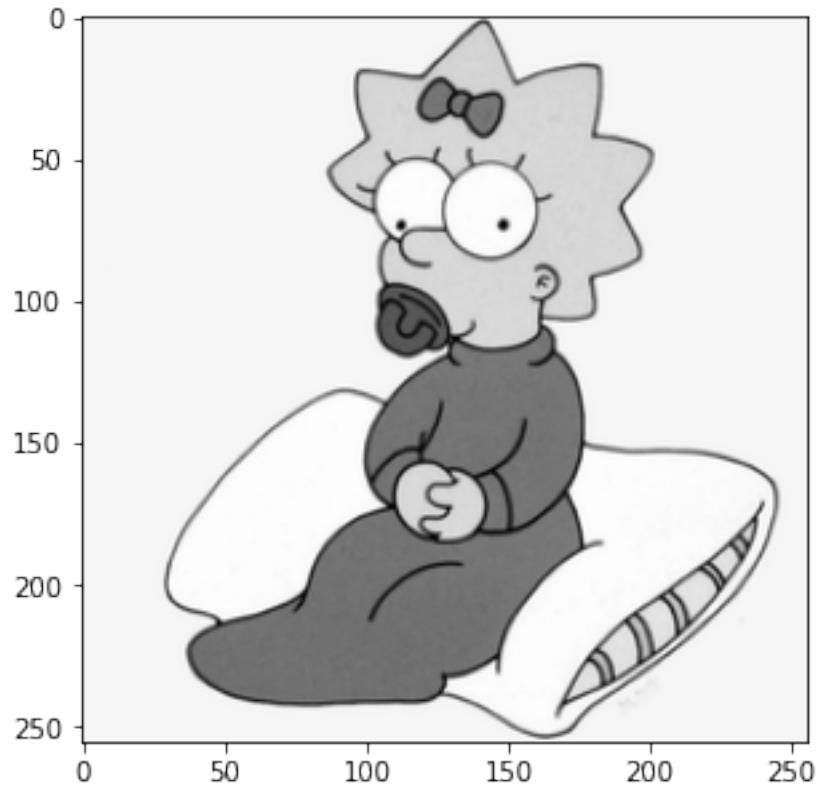


```
[ ]: for index, image in enumerate(grayScaleMaggie):
    image = np.array(image)
    norm = image/(image.max()/255)
    print(norm)
    #convert back to image
    im = Image.fromarray(norm)
    im = im.convert('RGB')
    im.save("normalizedMaggie/image-"+str(index)+".jpg")
    plt.imshow(im, cmap=plt.cm.gray)
    plt.show()
```

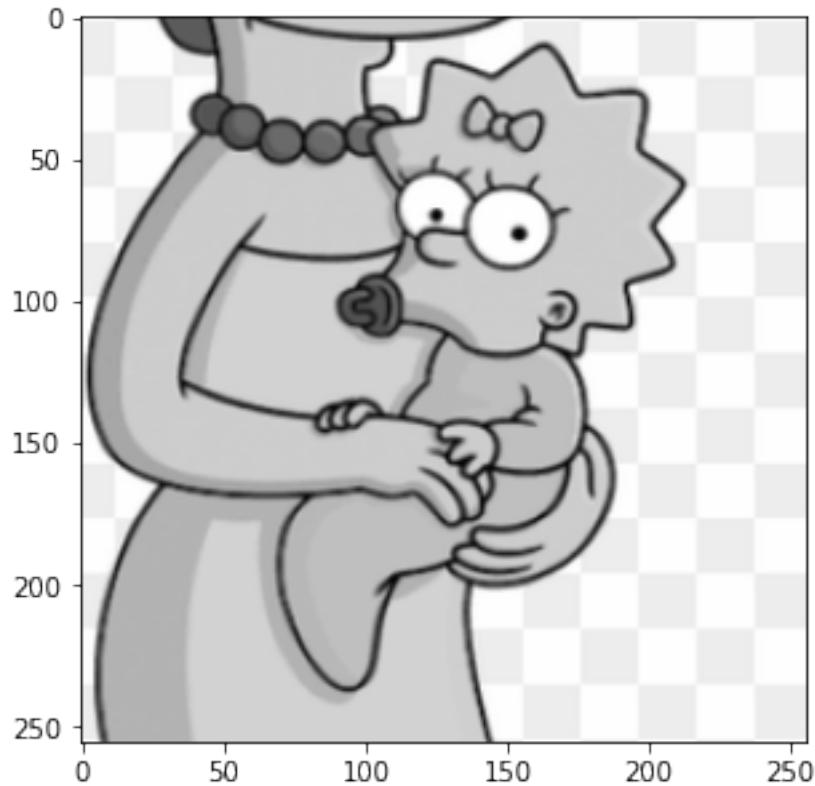
```
[[255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 ...
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]]
```



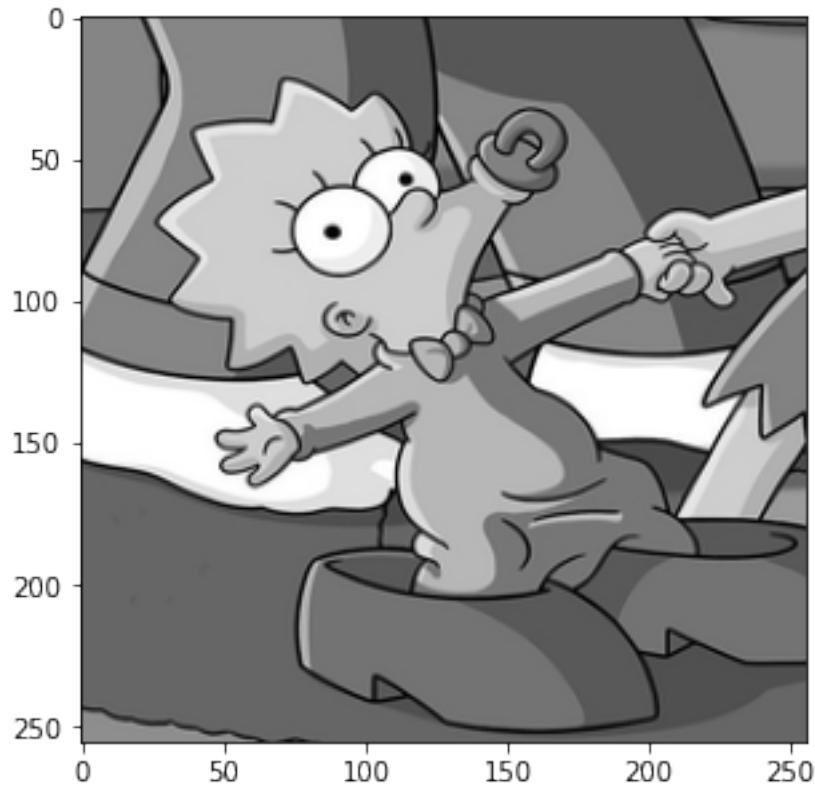
```
[[246. 246. 246. ... 246. 246. 246.]  
 [246. 246. 246. ... 246. 246. 246.]  
 [246. 246. 246. ... 246. 246. 246.]  
 ...  
 [246. 246. 246. ... 246. 246. 246.]  
 [246. 246. 246. ... 246. 246. 246.]  
 [246. 246. 246. ... 246. 246. 246.]]
```



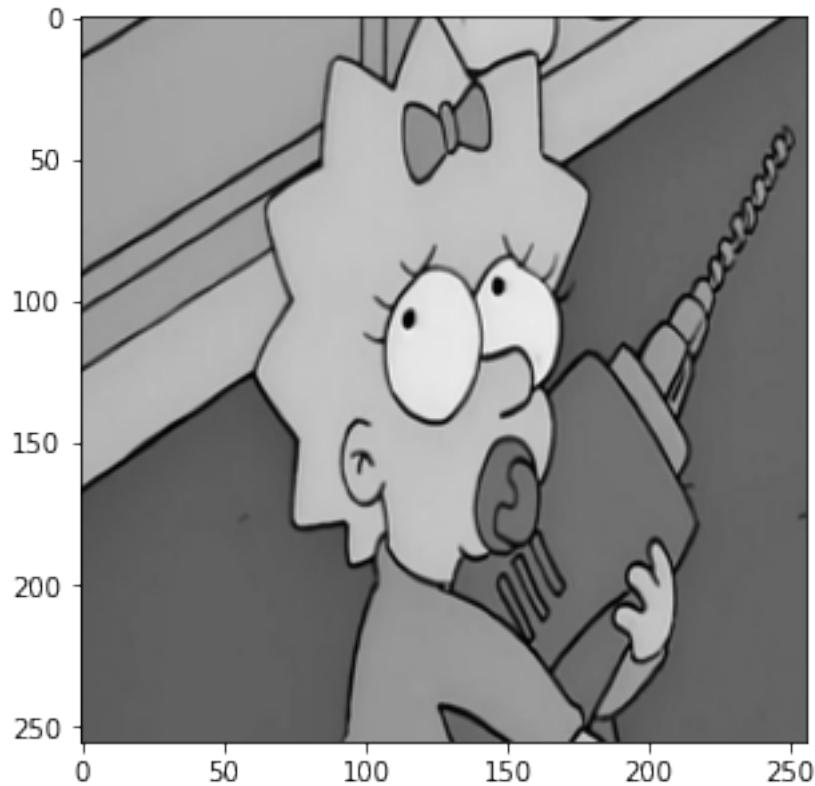
```
[[254. 254. 254. ... 254. 254. 254.]  
 [253. 253. 253. ... 253. 253. 253.]  
 [237. 237. 237. ... 237. 237. 237.]  
 ...  
 [237. 237. 237. ... 237. 237. 237.]  
 [238. 238. 238. ... 238. 238. 238.]  
 [254. 254. 254. ... 254. 254. 254.]]
```



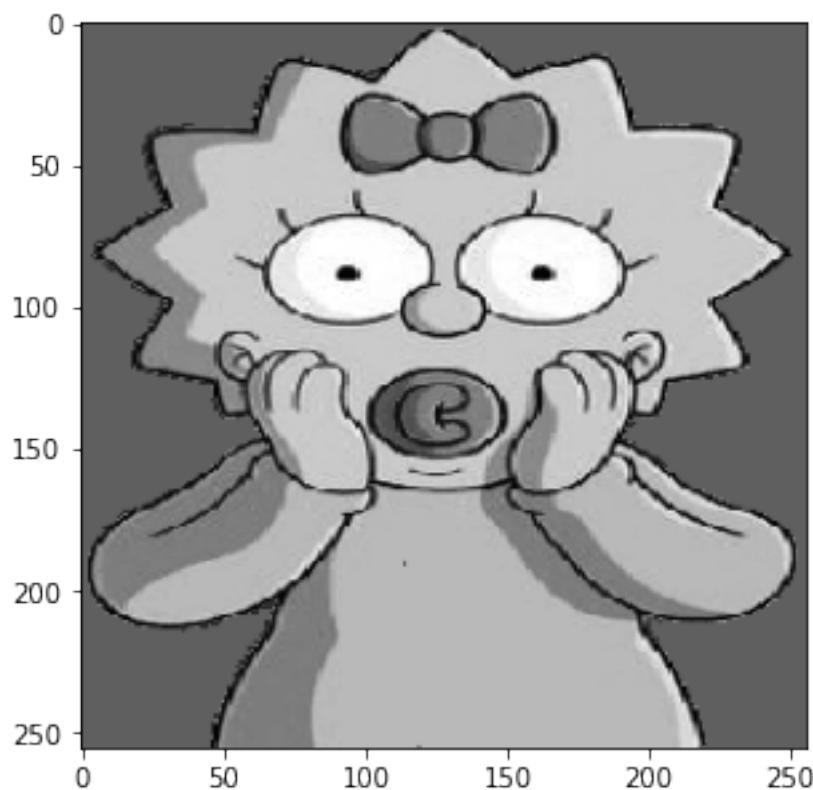
```
[[133. 133. 133. ... 100. 100. 100.]  
 [133. 133. 133. ... 110. 110. 110.]  
 [133. 133. 133. ... 64. 69. 73.]  
 ...  
 [142. 142. 142. ... 102. 102. 102.]  
 [142. 142. 142. ... 102. 102. 102.]  
 [142. 142. 142. ... 102. 102. 102.]]
```



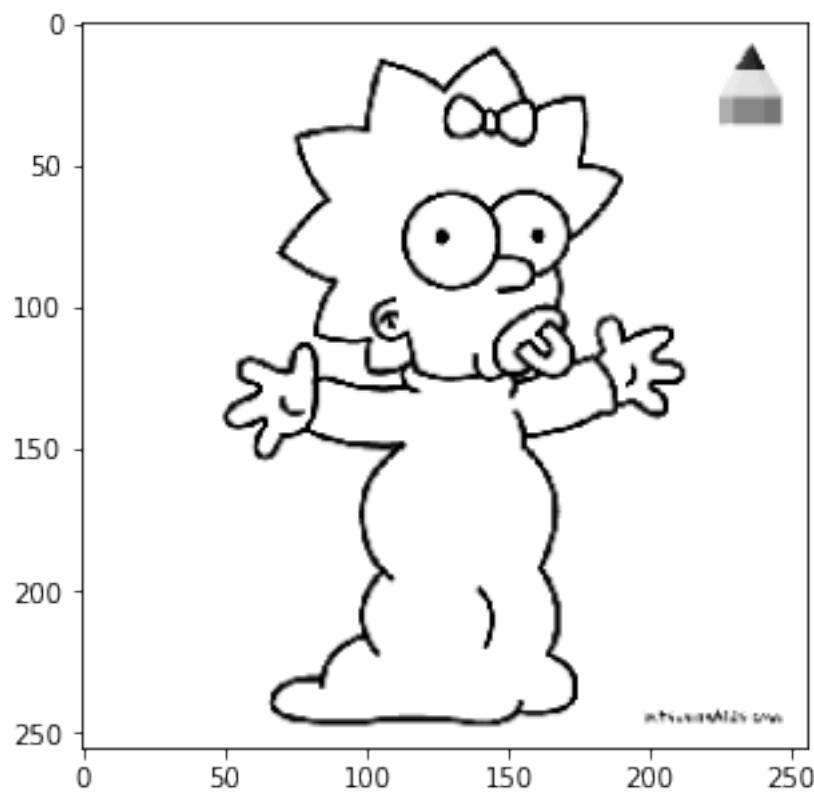
```
[[149.8125 161.5    160.4375 ... 97.75    97.75    96.6875]
 [160.4375 160.4375 160.4375 ... 97.75    97.75    97.75   ]
 [159.375   158.3125 159.375  ... 97.75    96.6875  97.75   ]
 ...
 [ 91.375   91.375   92.4375 ... 100.9375 100.9375 100.9375]
 [ 91.375   91.375   92.4375 ... 100.9375 100.9375 100.9375]
 [ 91.375   91.375   92.4375 ... 100.9375 100.9375 100.9375]]
```



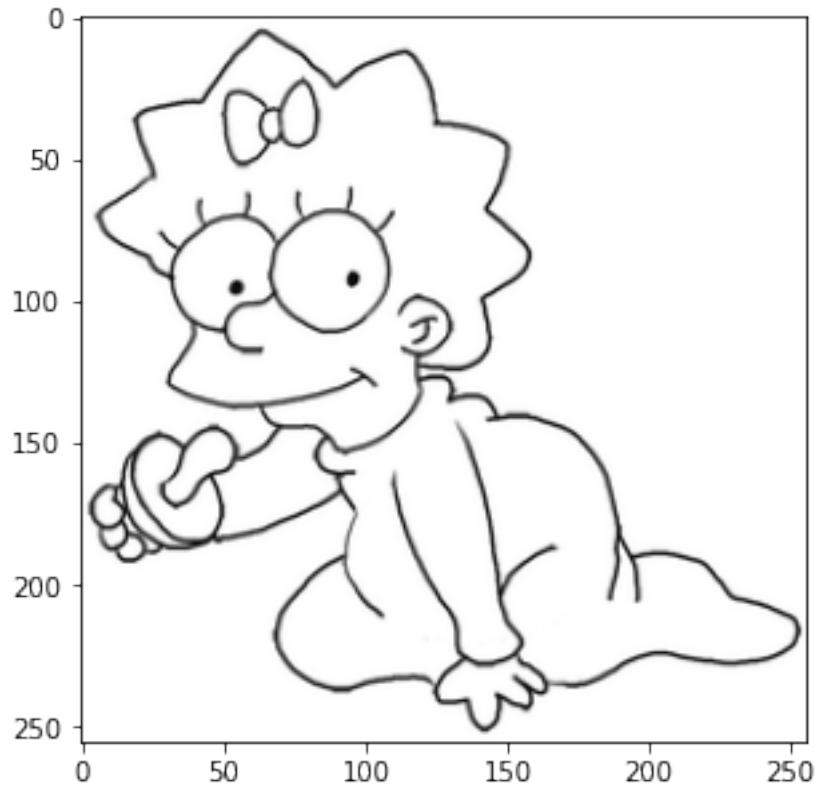
```
[[95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]  
 ...  
 [95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]]
```



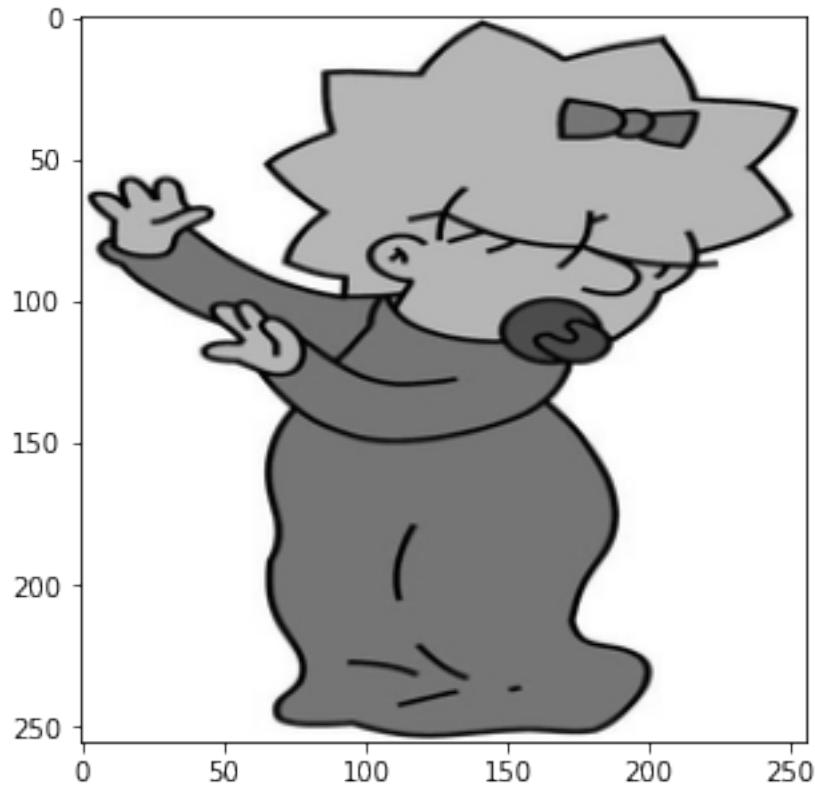
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



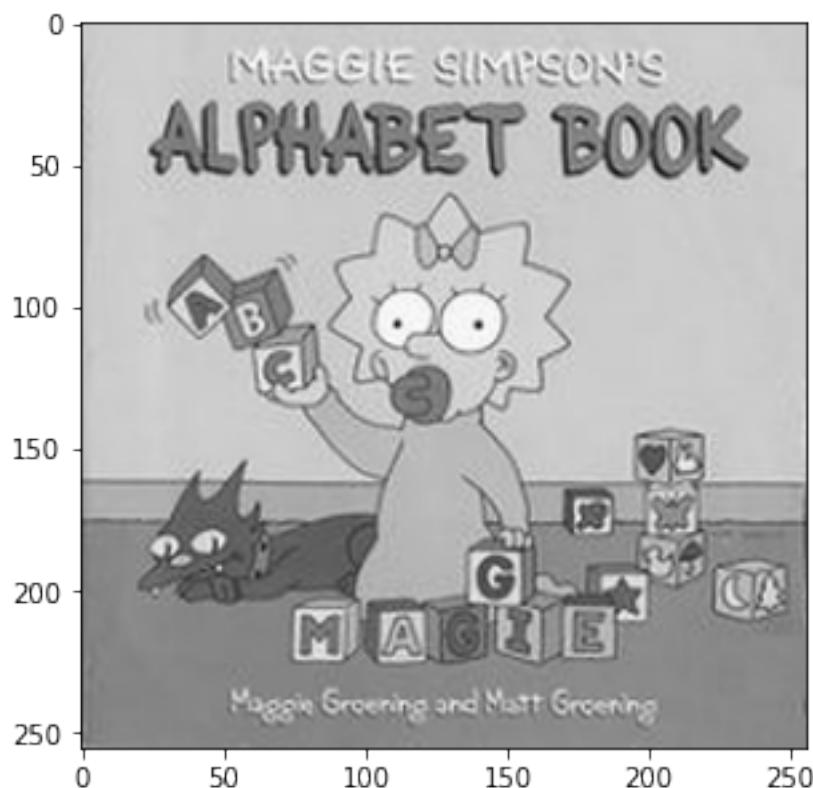
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



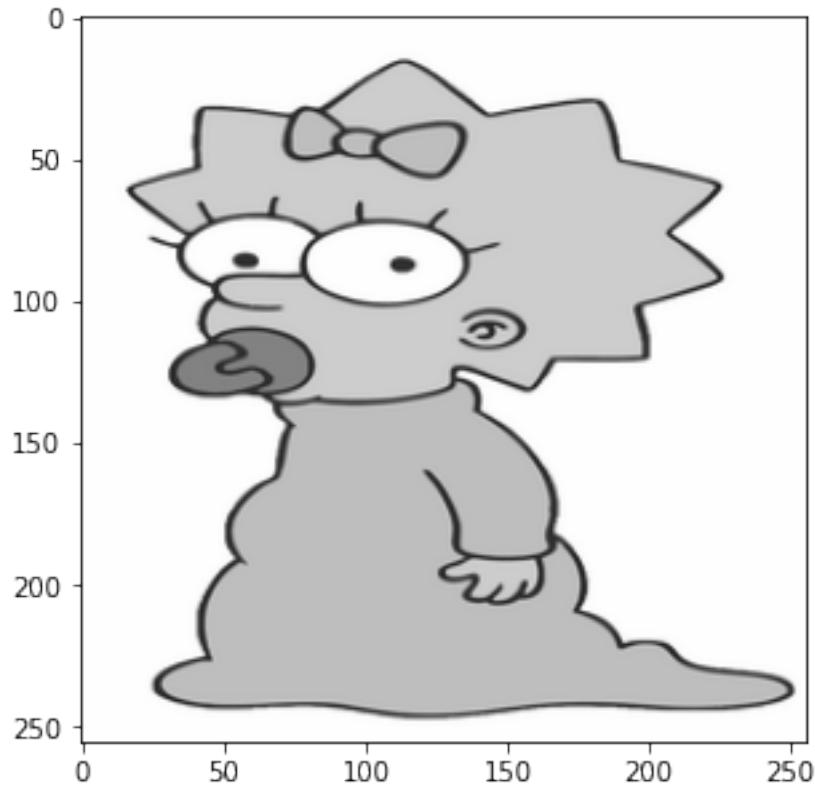
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



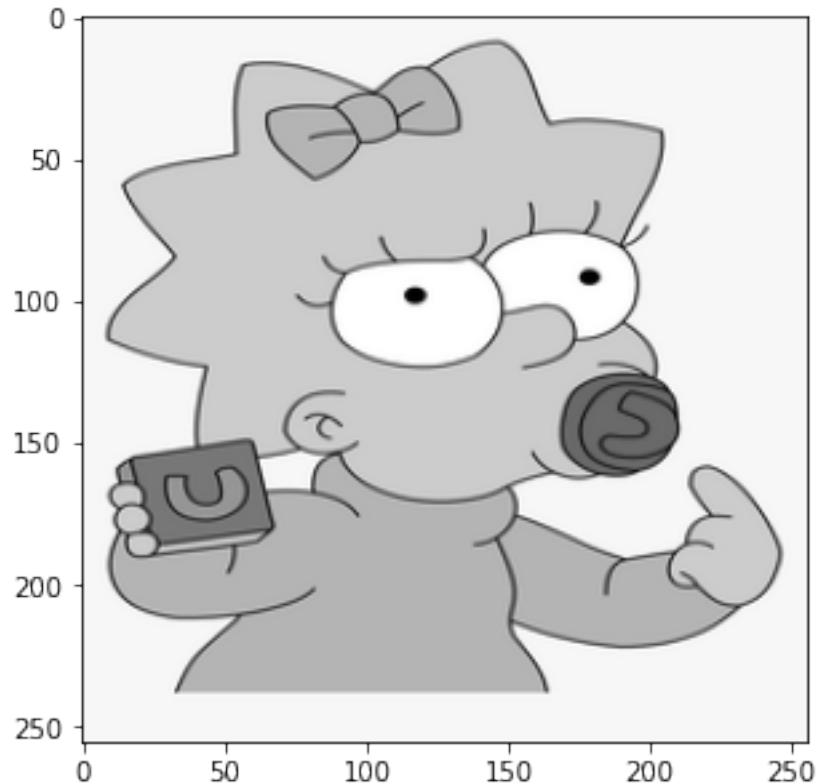
```
[[209. 205. 201. ... 203. 203. 202.]  
 [208. 205. 201. ... 202. 202. 202.]  
 [208. 205. 201. ... 202. 202. 202.]  
 ...  
 [139. 131. 119. ... 135. 135. 136.]  
 [176. 154. 129. ... 135. 135. 136.]  
 [202. 171. 140. ... 135. 135. 135.]]
```



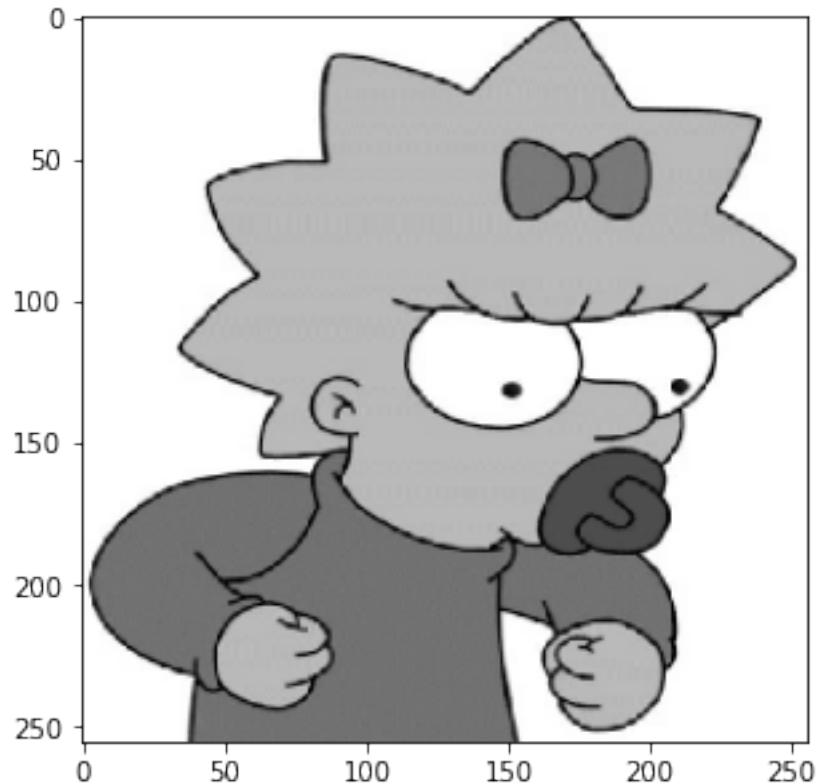
[[254. 254. 254. ... 254. 254. 254.]
[254. 254. 254. ... 254. 254. 254.]
[254. 254. 254. ... 254. 254. 254.]
...
[254. 254. 254. ... 254. 254. 254.]
[254. 254. 254. ... 254. 254. 254.]
[254. 254. 254. ... 254. 254. 254.]]



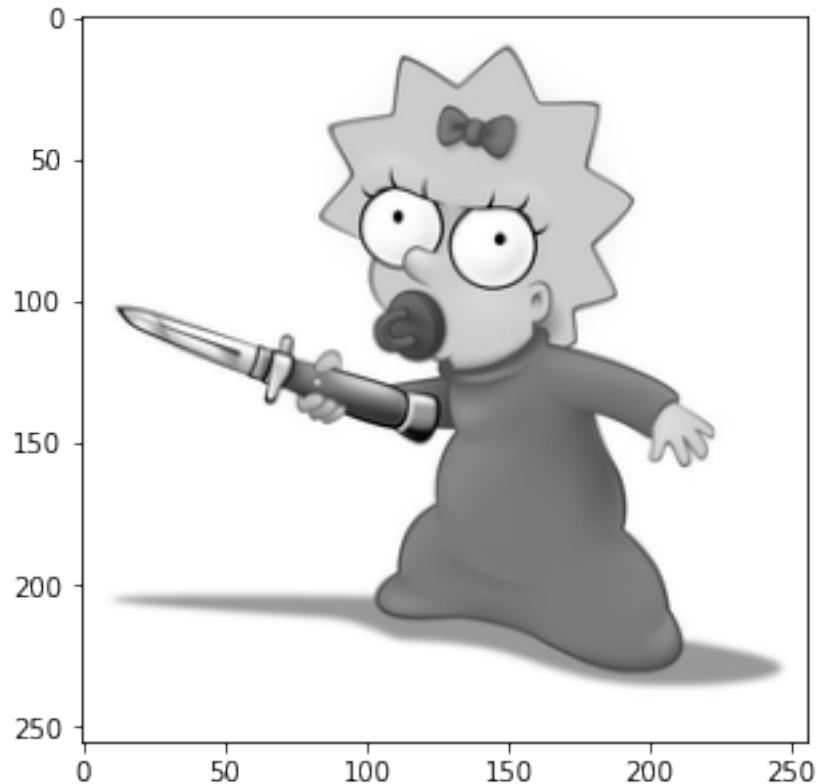
```
[[247. 247. 247. ... 247. 247. 247.]  
 [247. 247. 247. ... 247. 247. 247.]  
 [247. 247. 247. ... 247. 247. 247.]  
 ...  
 [247. 247. 247. ... 247. 247. 247.]  
 [247. 247. 247. ... 247. 247. 247.]  
 [247. 247. 247. ... 247. 247. 247.]]
```



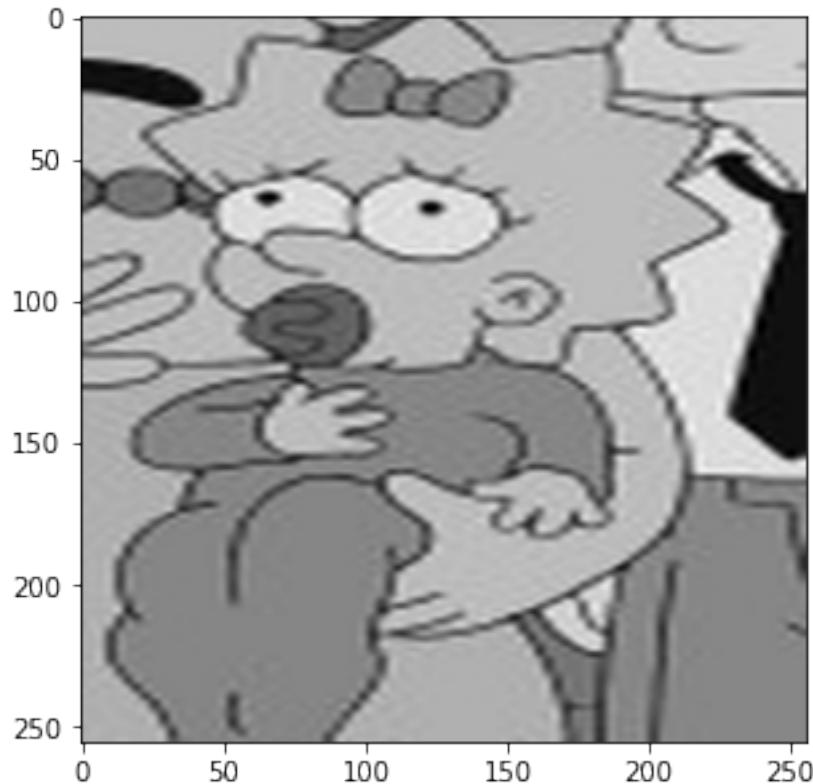
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



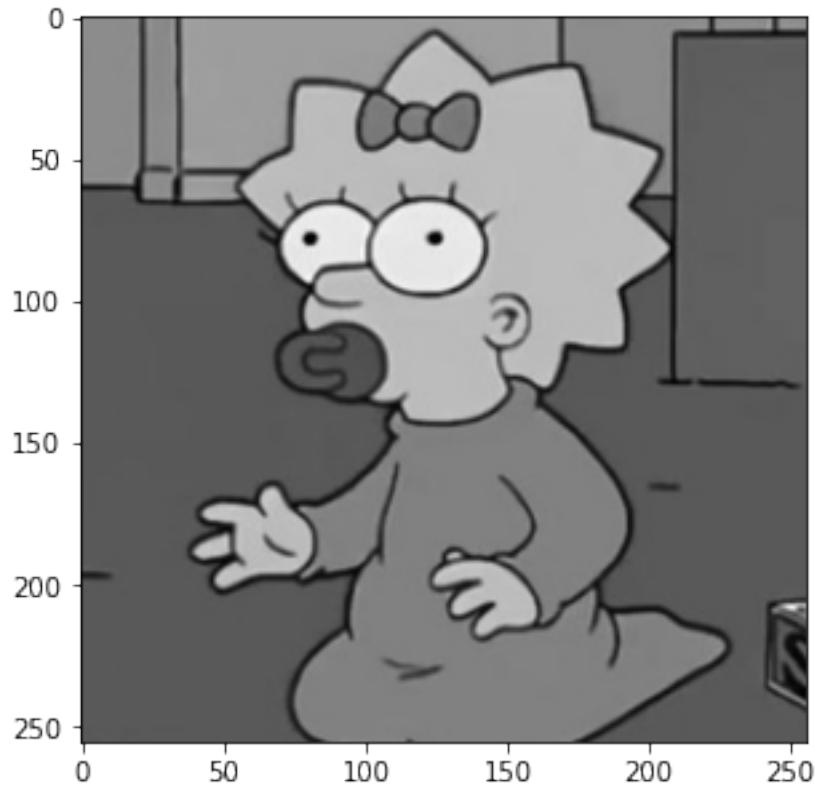
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



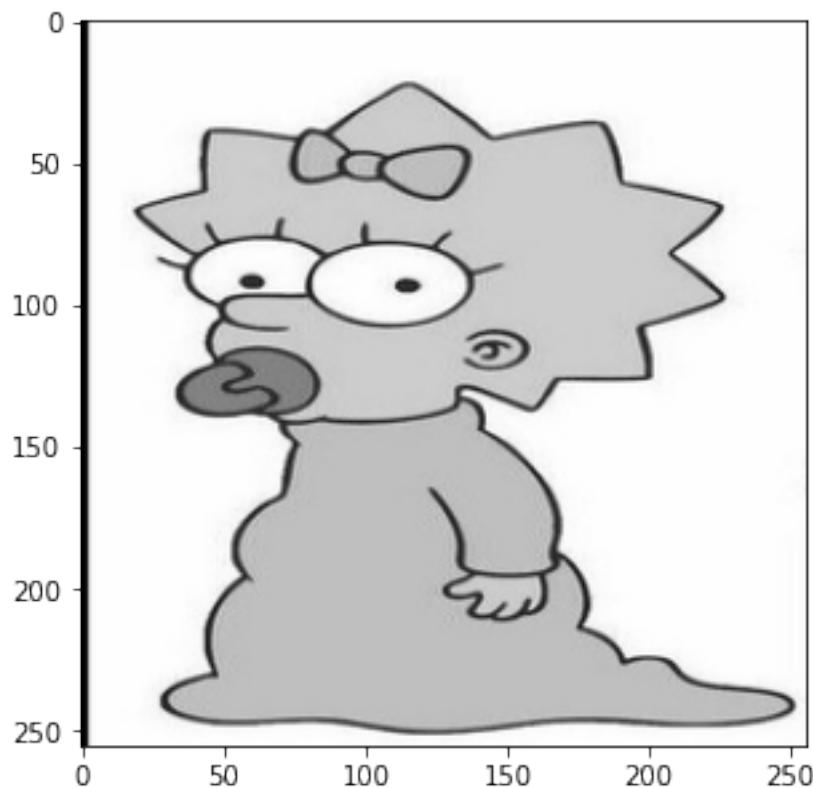
```
[[189.69512195 189.69512195 189.69512195 ... 208.35365854 208.35365854  
208.35365854]  
[189.69512195 189.69512195 189.69512195 ... 208.35365854 208.35365854  
208.35365854]  
[189.69512195 189.69512195 189.69512195 ... 208.35365854 208.35365854  
208.35365854]  
...  
[174.14634146 175.18292683 176.2195122 ... 135.79268293 135.79268293  
135.79268293]  
[175.18292683 176.2195122 177.25609756 ... 135.79268293 135.79268293  
135.79268293]  
[175.18292683 176.2195122 177.25609756 ... 135.79268293 135.79268293  
135.79268293]]
```



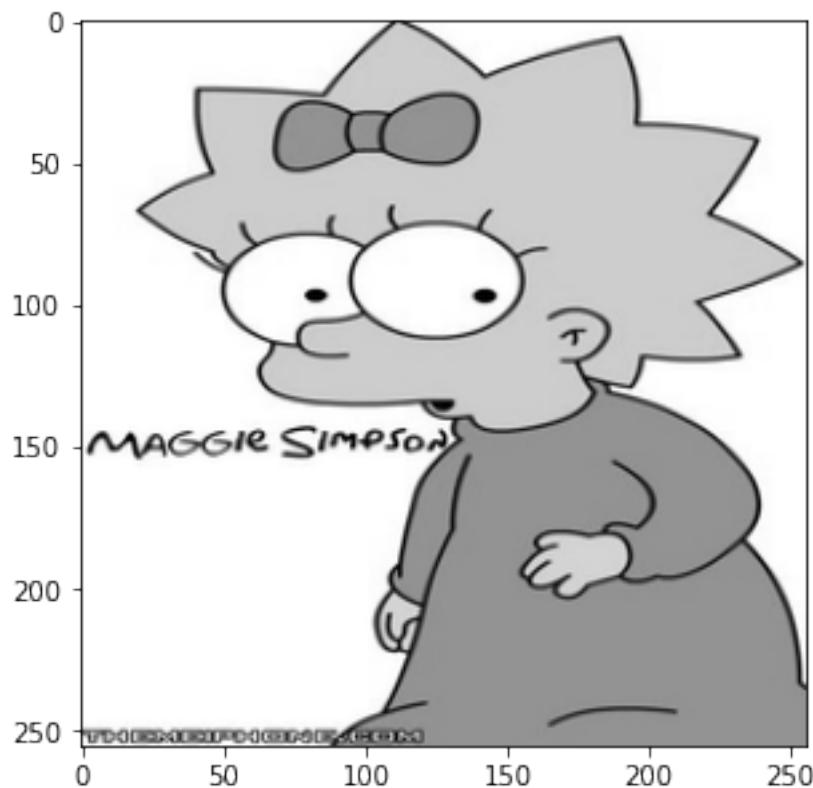
```
[[140.6547619  140.6547619  140.6547619 ... 142.67857143 142.67857143  
142.67857143]  
[140.6547619  140.6547619  140.6547619 ... 141.66666667 141.66666667  
141.66666667]  
[140.6547619  140.6547619  140.6547619 ... 140.6547619  140.6547619  
140.6547619 ]  
...  
[ 89.04761905  89.04761905  89.04761905 ...  89.04761905  89.04761905  
89.04761905]  
[ 89.04761905  89.04761905  89.04761905 ...  89.04761905  89.04761905  
89.04761905]  
[ 91.07142857  89.04761905  89.04761905 ...  89.04761905  89.04761905  
89.04761905]]
```



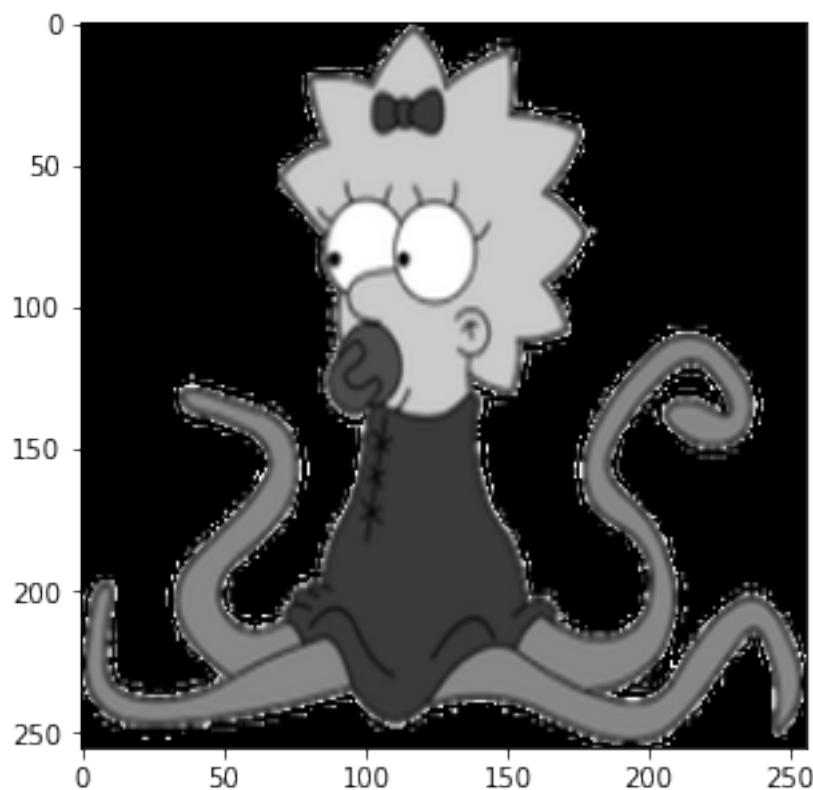
```
[[ 1.  0.  28. ... 254. 254. 254.]  
 [ 1.  0.  28. ... 254. 254. 254.]  
 [ 1.  0.  28. ... 254. 254. 254.]  
 ...  
 [ 1.  0.  27. ... 253. 252. 252.]  
 [ 1.  0.  27. ... 254. 252. 252.]  
 [ 1.  0.  27. ... 254. 251. 252.]]
```



```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [253. 255. 240. ... 147. 147. 147.]  
 [251. 246. 248. ... 147. 147. 147.]  
 [252. 251. 254. ... 147. 147. 147.]]
```



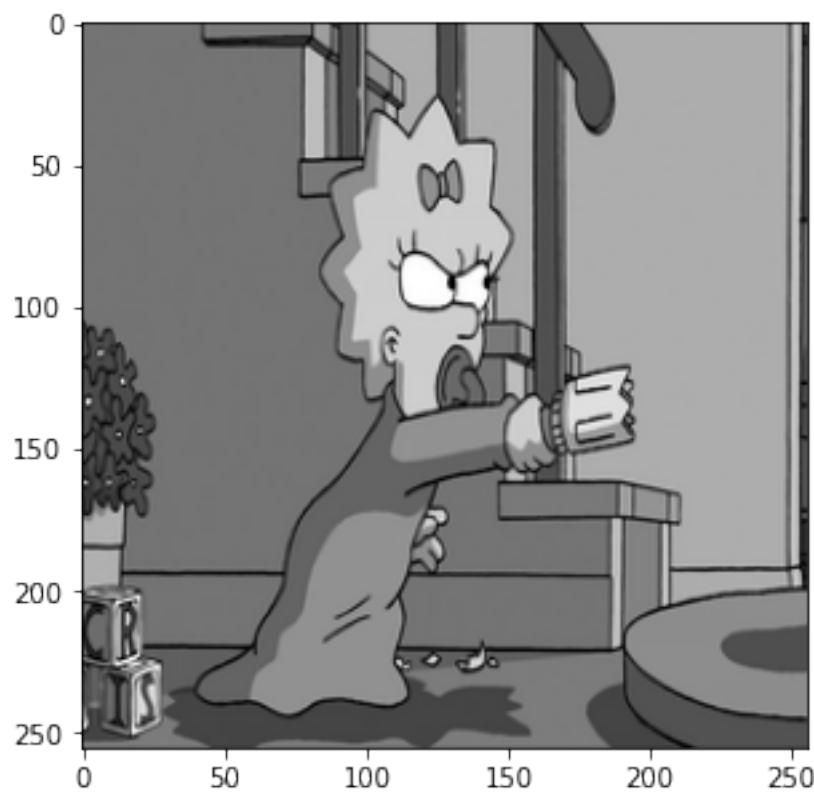
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



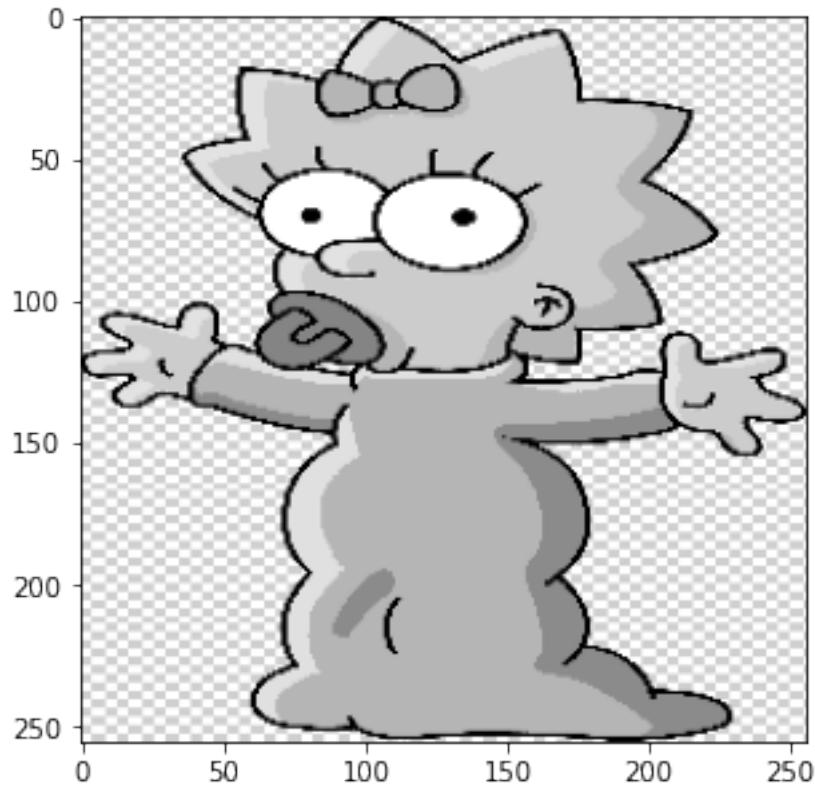
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



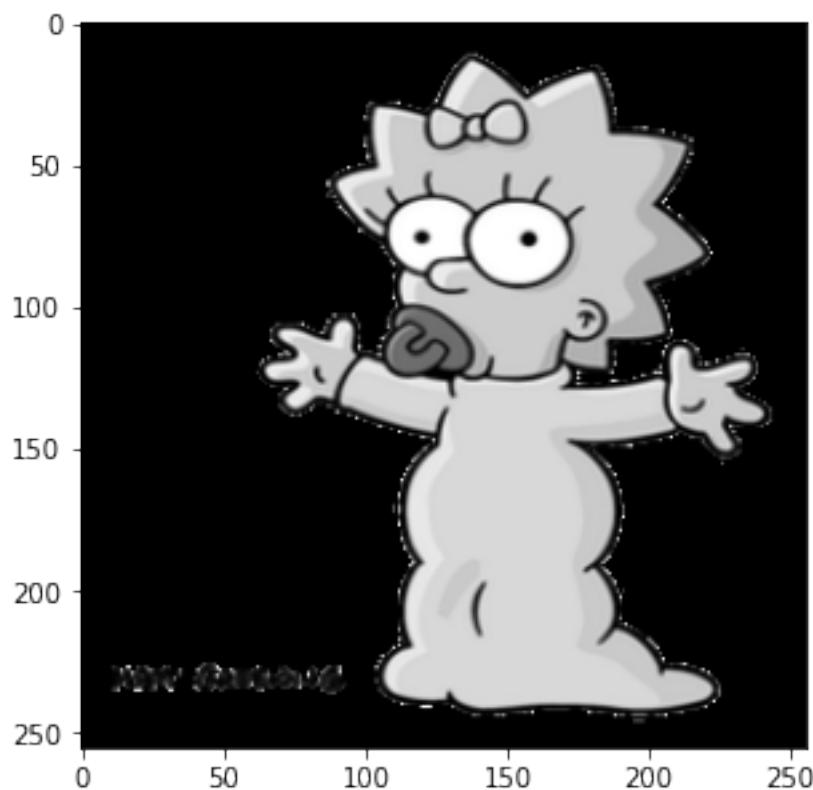
```
[[116. 116. 116. ... 174. 35. 80.]  
 [116. 116. 116. ... 174. 35. 79.]  
 [116. 116. 116. ... 175. 35. 79.]  
 ...  
 [ 87. 87. 87. ... 82. 82. 82.]  
 [128. 128. 128. ... 83. 83. 83.]  
 [109. 110. 110. ... 72. 72. 72.]]
```



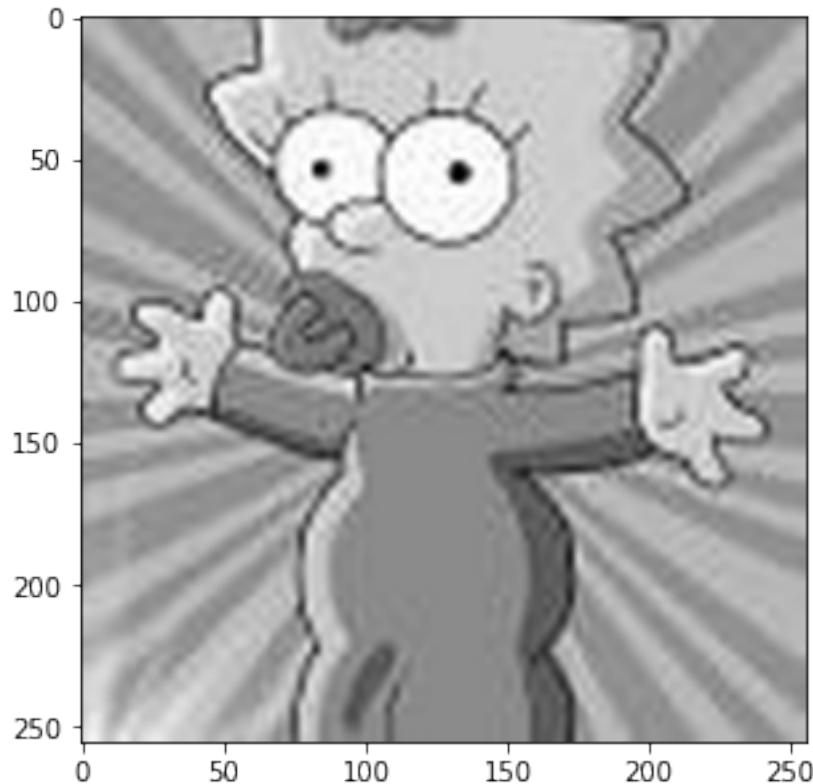
```
[[255. 255. 255. ... 255. 255. 207.]  
[255. 255. 255. ... 255. 255. 207.]  
[255. 255. 255. ... 255. 255. 207.]  
...  
[255. 255. 255. ... 255. 255. 207.]  
[255. 255. 255. ... 255. 255. 207.]  
[255. 255. 255. ... 255. 255. 207.]]
```



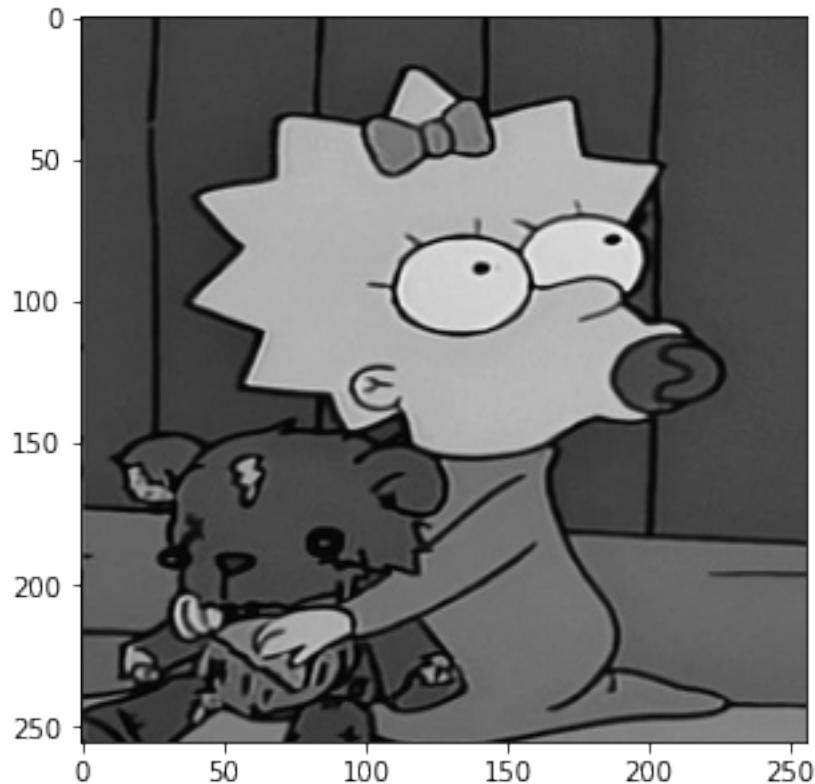
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



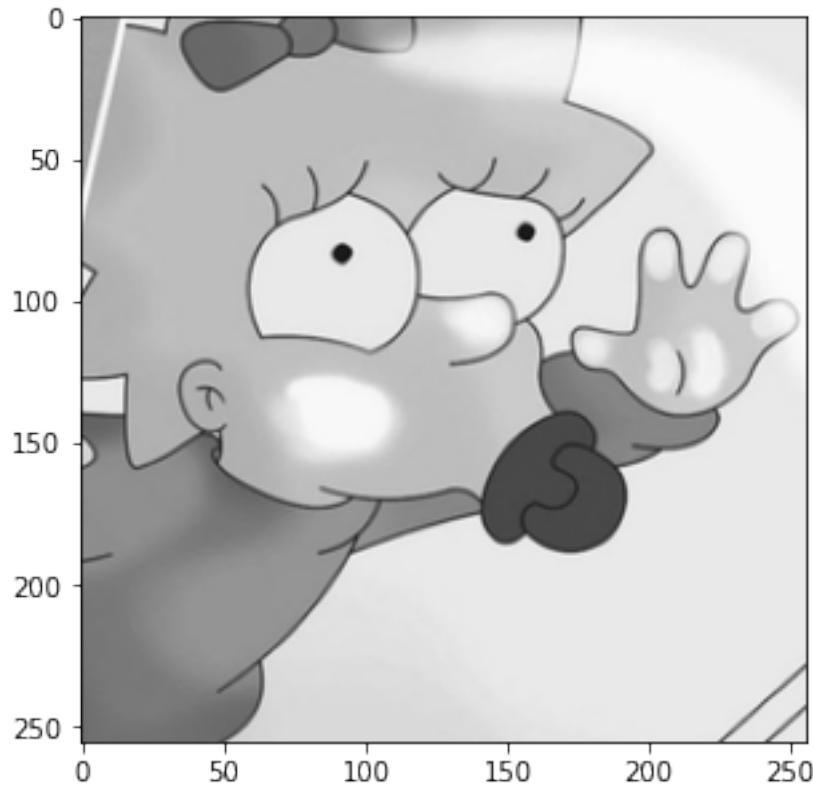
```
[[188. 187. 185. ... 146. 149. 150.]  
 [188. 187. 185. ... 150. 153. 154.]  
 [187. 186. 185. ... 156. 160. 160.]  
 ...  
 [221. 226. 232. ... 185. 185. 185.]  
 [221. 226. 233. ... 184. 184. 184.]  
 [221. 226. 233. ... 184. 184. 184.]]
```



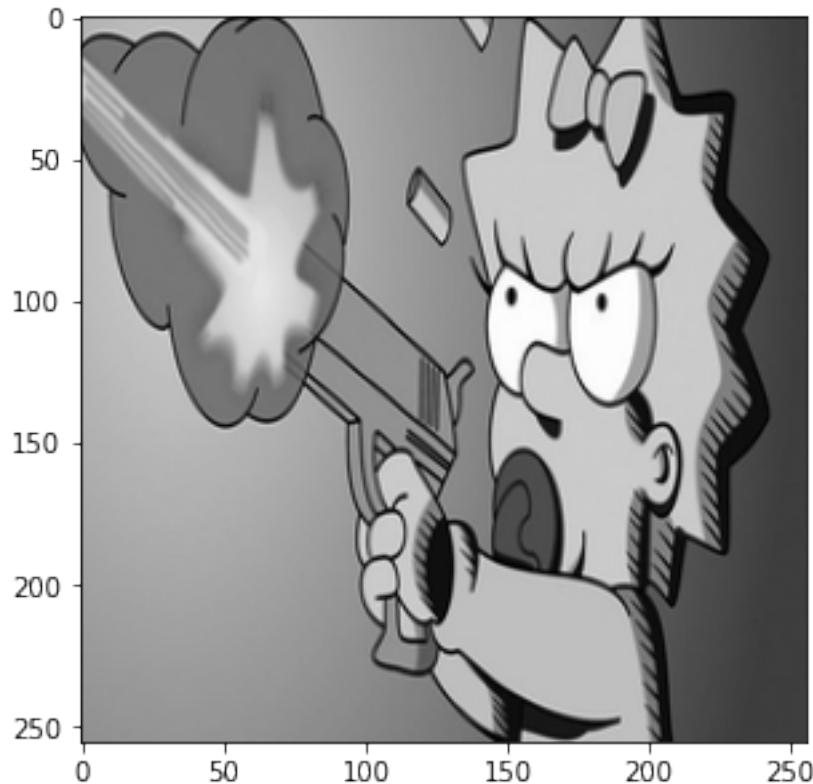
```
[[ 63.75      67.86290323  68.89112903 ...  69.91935484  70.94758065
  73.00403226]
 [ 64.77822581  69.91935484  71.97580645 ...  70.94758065  70.94758065
  71.97580645]
 [ 62.72177419  68.89112903  71.97580645 ...  69.91935484  71.97580645
  73.00403226]
 ...
 [ 70.94758065  83.28629032  71.97580645 ... 135.72580645 133.66935484
 132.64112903]
 [ 66.83467742  73.00403226  77.11693548 ... 135.72580645 133.66935484
 133.66935484]
 [ 65.80645161  40.10080645  67.86290323 ... 134.69758065 135.72580645
 135.72580645]]
```



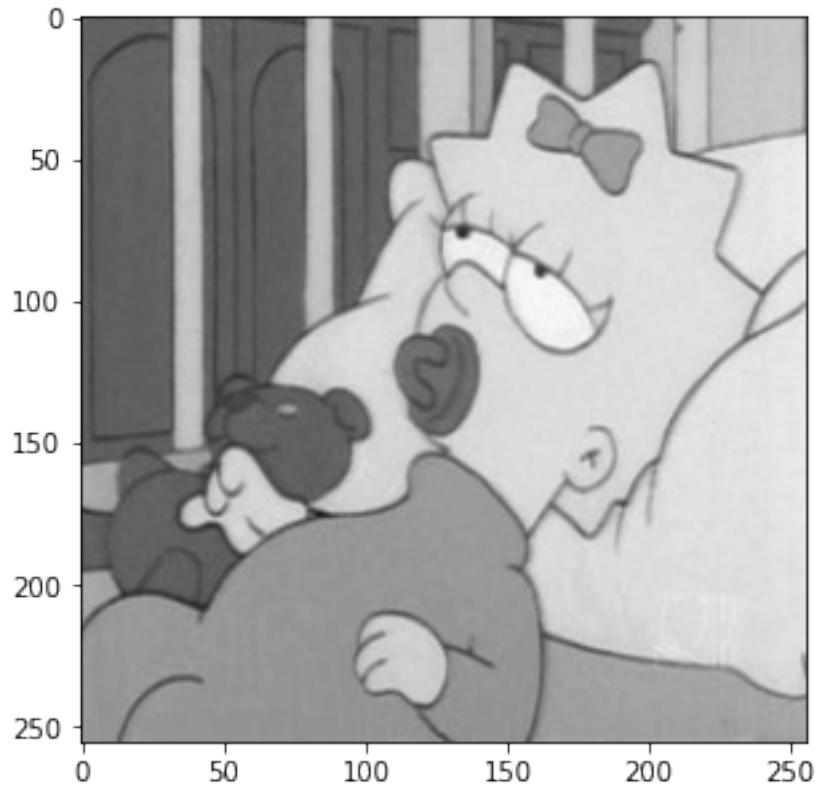
```
[[165.29644269 162.27272727 168.3201581 ... 232.82608696 232.82608696  
232.82608696]  
[165.29644269 162.27272727 162.27272727 ... 236.85770751 236.85770751  
236.85770751]  
[164.28853755 162.27272727 163.28063241 ... 232.82608696 232.82608696  
232.82608696]  
...  
[112.88537549 110.86956522 111.87747036 ... 233.83399209 233.83399209  
233.83399209]  
[111.87747036 110.86956522 110.86956522 ... 233.83399209 233.83399209  
233.83399209]  
[105.83003953 109.86166008 110.86956522 ... 233.83399209 233.83399209  
233.83399209]]
```



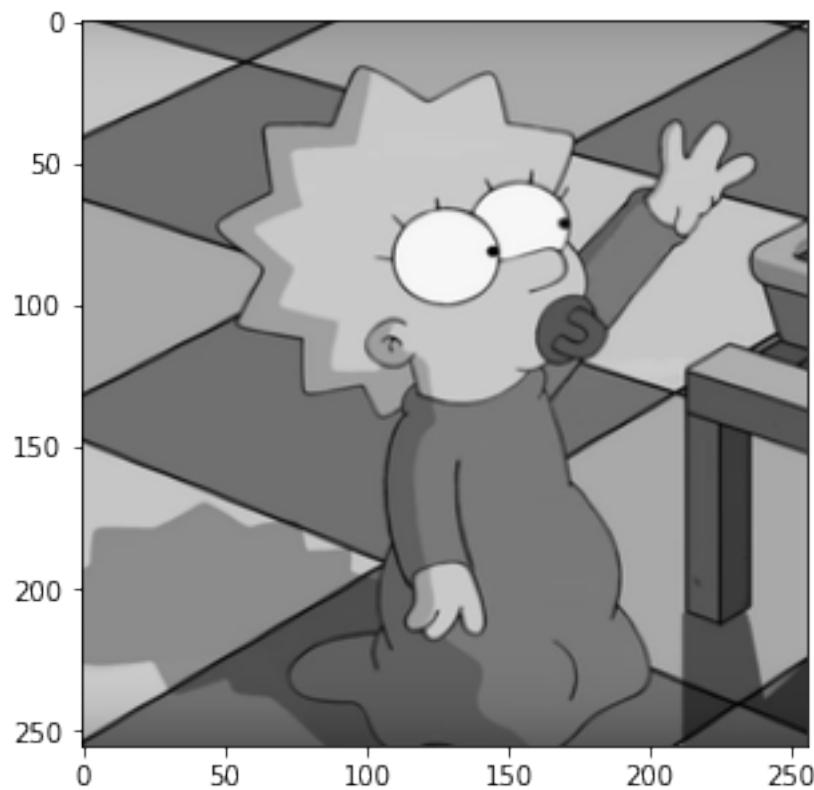
```
[[179. 179. 180. ... 62. 62. 63.]  
 [179. 179. 180. ... 62. 62. 63.]  
 [179. 180. 180. ... 62. 62. 63.]  
 ...  
 [144. 144. 144. ... 62. 62. 63.]  
 [144. 144. 145. ... 62. 62. 63.]  
 [143. 145. 147. ... 62. 62. 63.]]
```



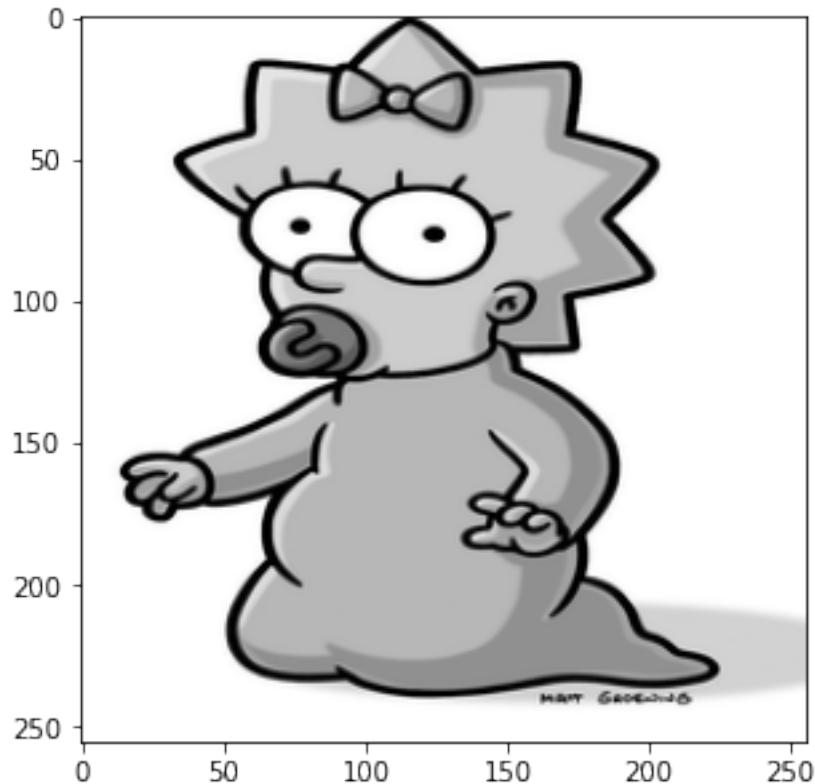
```
[[104.17021277 104.17021277 102.          ... 199.65957447 184.46808511
144.31914894]
[100.91489362 100.91489362 99.82978723 ... 199.65957447 184.46808511
144.31914894]
[ 99.82978723 100.91489362 100.91489362 ... 199.65957447 184.46808511
144.31914894]
...
[151.91489362 153.          151.91489362 ... 166.0212766 167.10638298
167.10638298]
[151.91489362 153.          151.91489362 ... 161.68085106 161.68085106
162.76595745]
[151.91489362 153.          151.91489362 ... 164.93617021 164.93617021
166.0212766 ]]
```



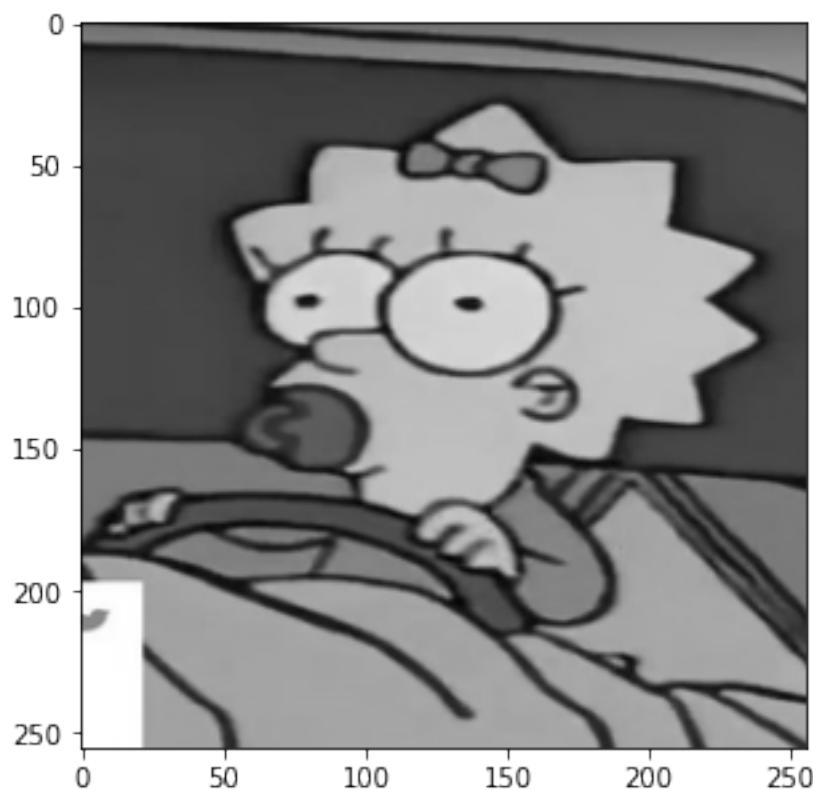
```
[[175.37549407 170.33596838 150.17786561 ... 100.79051383 70.55335968
 25.19762846]
[179.40711462 179.40711462 179.40711462 ... 41.32411067 22.17391304
 62.49011858]
[179.40711462 180.41501976 179.40711462 ... 4.03162055 90.71146245
 165.29644269]
...
[[128.00395257 52.41106719 12.09486166 ... 74.58498024 74.58498024
 75.59288538]
[ 18.14229249 30.23715415 74.58498024 ... 74.58498024 74.58498024
 73.5770751 ]
[ 14.11067194 68.53754941 92.72727273 ... 72.56916996 72.56916996
 72.56916996]]
```



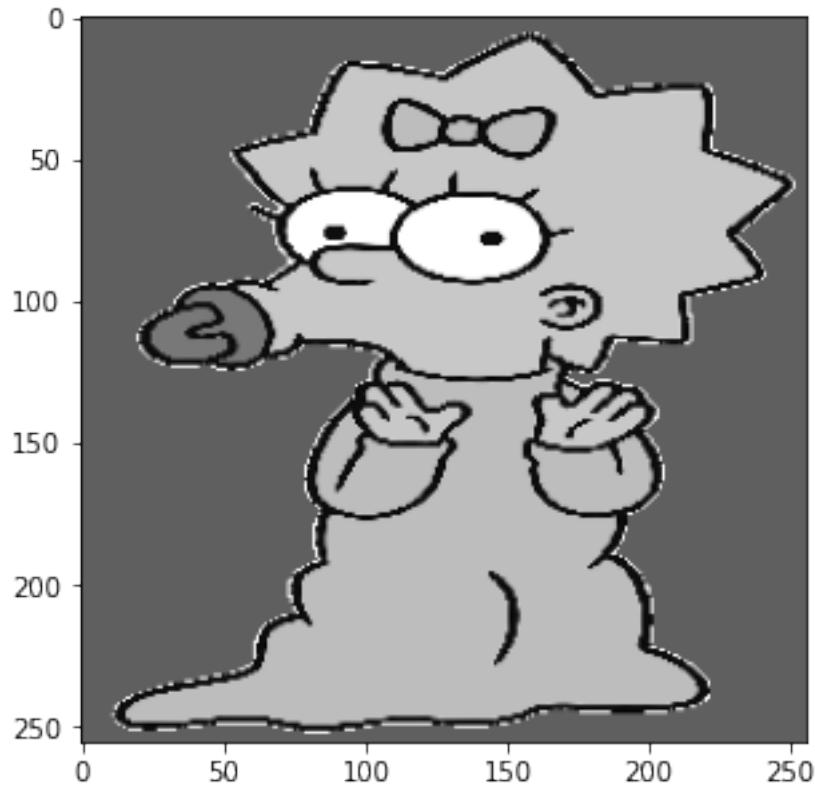
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



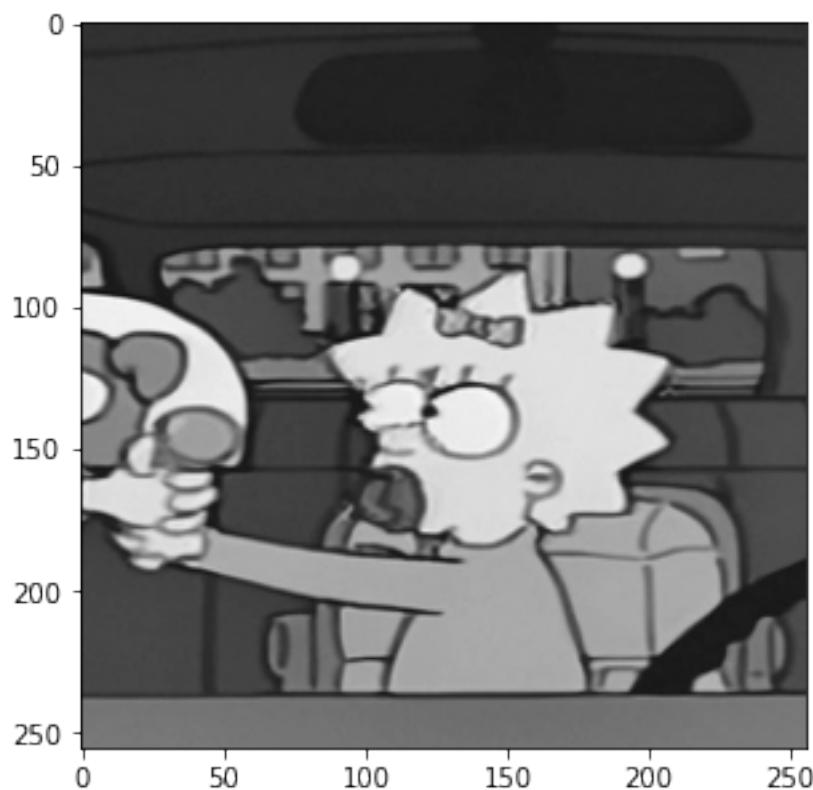
```
[[ 66.  65.  60. ... 113. 114. 114.]  
 [103. 102. 101. ... 116. 117. 117.]  
 [118. 116. 117. ... 118. 119. 119.]  
 ...  
 [255. 255. 255. ...  54.  77. 107.]  
 [255. 255. 255. ...   49.   47.   60.]  
 [255. 255. 255. ...   54.   45.   47.]]
```



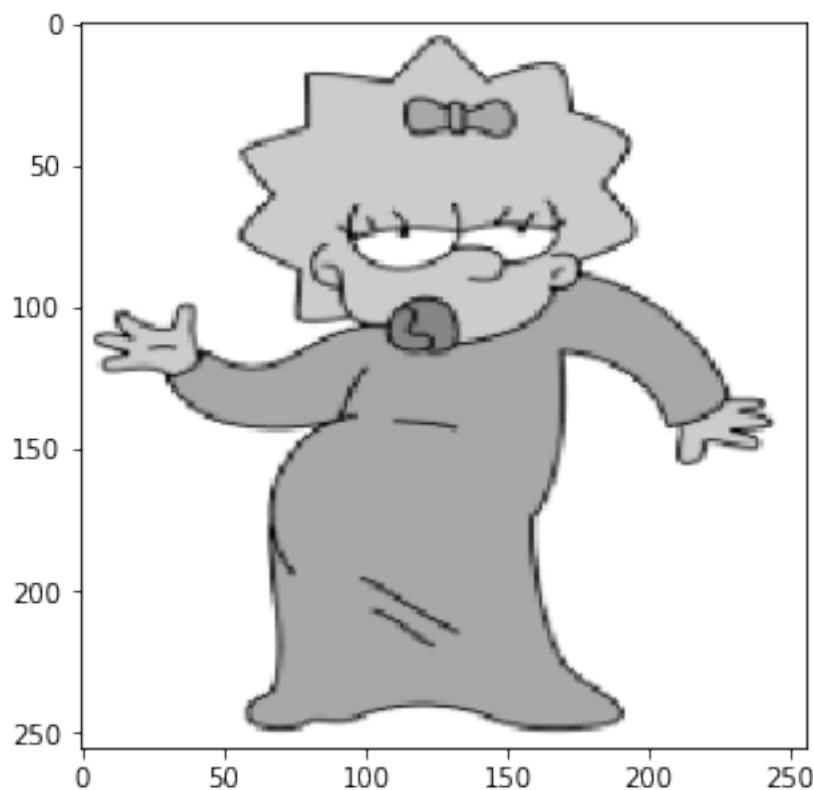
```
[[95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]  
 ...  
 [95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]  
 [95. 95. 95. ... 95. 95. 95.]]
```



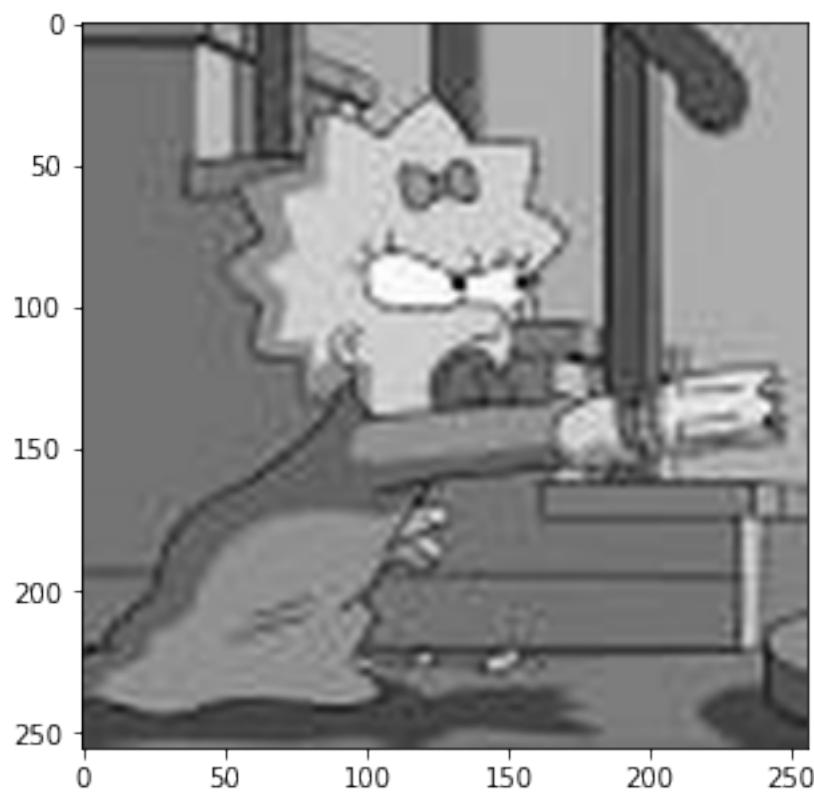
```
[[ 49.03846154  50.26442308  51.49038462 ... 49.03846154  50.26442308
  50.26442308]
 [ 49.03846154  52.71634615  51.49038462 ... 47.8125      47.8125
  50.26442308]
 [ 50.26442308  47.8125      50.26442308 ... 47.8125      47.8125
  49.03846154]
...
[120.14423077 121.37019231 122.59615385 ... 125.04807692 122.59615385
 123.82211538]
[121.37019231 122.59615385 122.59615385 ... 122.59615385 122.59615385
 125.04807692]
[120.14423077 123.82211538 123.82211538 ... 125.04807692 123.82211538
 125.04807692]]
```



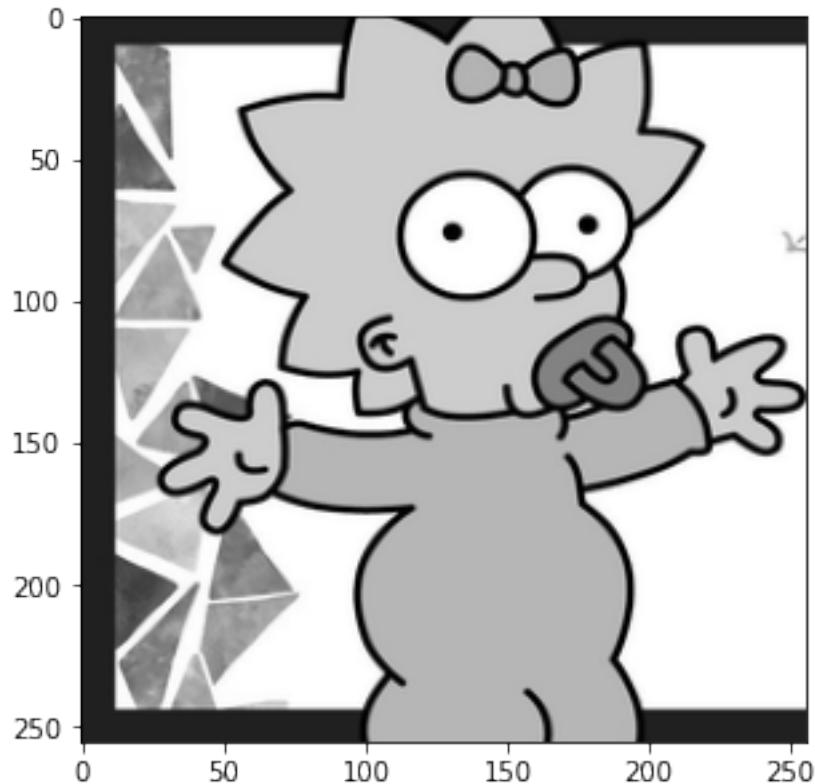
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



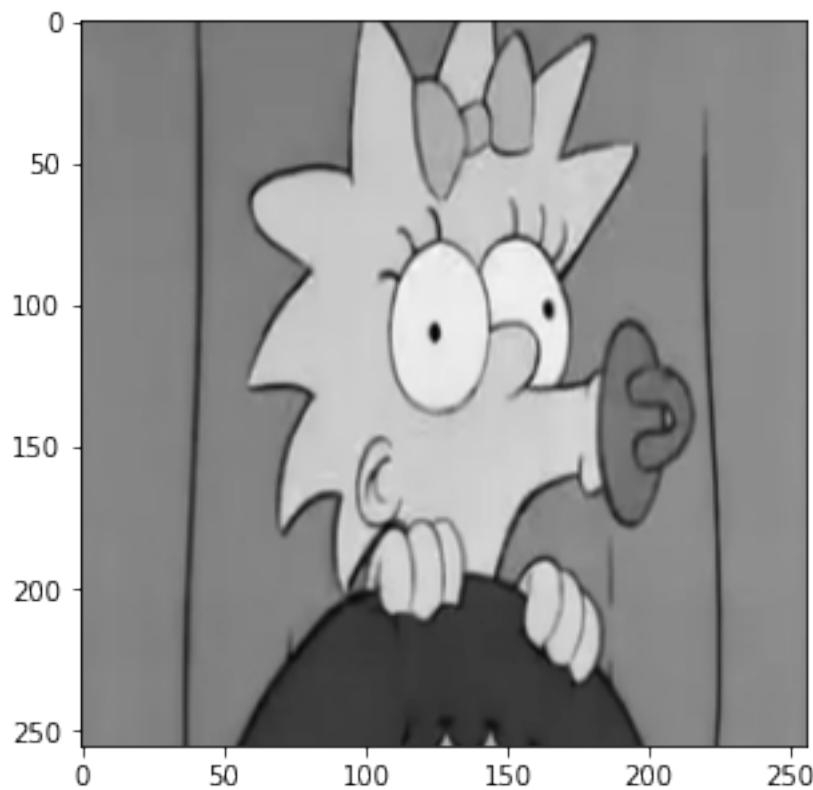
```
[[119. 119. 119. ... 173. 173. 173.]  
 [123. 123. 123. ... 173. 173. 173.]  
 [130. 130. 130. ... 173. 173. 173.]  
 ...  
 [ 72.  71.  70. ... 67.  67.  68.]  
 [ 72.  71.  70. ... 64.  64.  64.]  
 [ 72.  71.  70. ... 61.  61.  61.]]
```



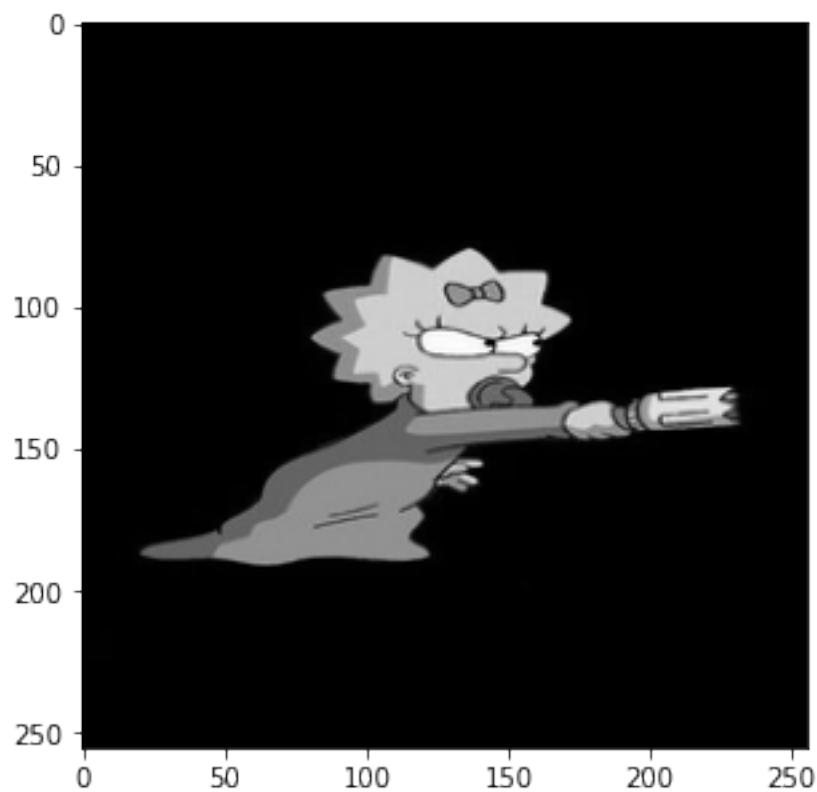
```
[[32. 32. 32. ... 32. 32. 32.]  
 [32. 32. 32. ... 32. 32. 32.]  
 [32. 32. 32. ... 32. 32. 32.]  
 ...  
 [32. 32. 32. ... 32. 32. 32.]  
 [32. 32. 32. ... 32. 32. 32.]  
 [32. 32. 32. ... 32. 32. 32.]]
```



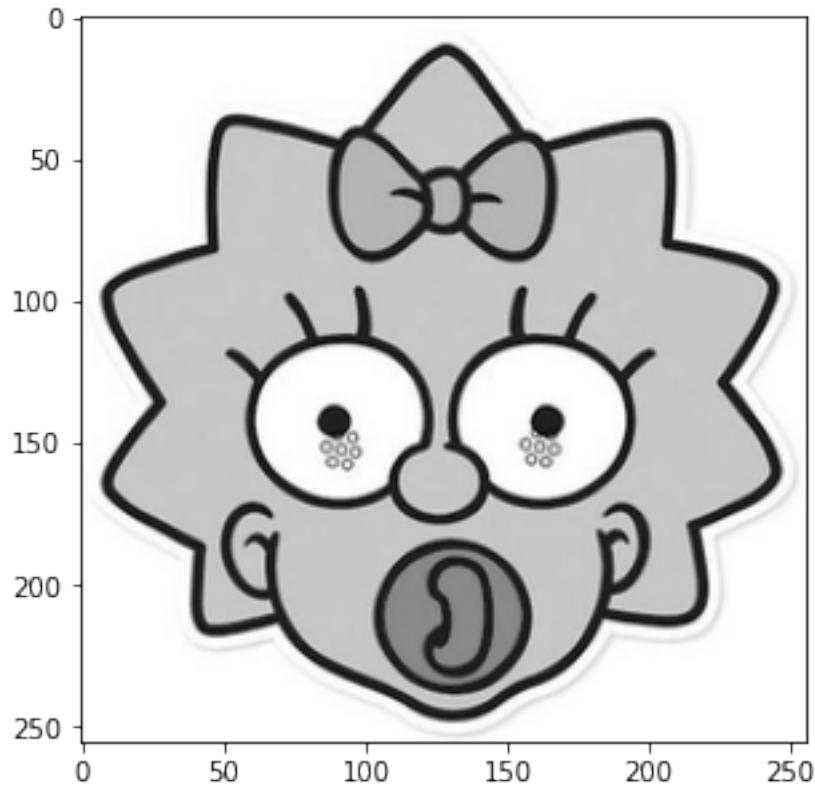
```
[[124.7639485 129.1416309 125.8583691 ... 140.08583691 141.18025751  
138.99141631]  
[124.7639485 129.1416309 128.0472103 ... 141.18025751 141.18025751  
138.99141631]  
[124.7639485 129.1416309 126.9527897 ... 141.18025751 140.08583691  
138.99141631]  
...  
[124.7639485 129.1416309 129.1416309 ... 138.99141631 136.80257511  
134.61373391]  
[124.7639485 129.1416309 129.1416309 ... 138.99141631 136.80257511  
134.61373391]  
[124.7639485 129.1416309 129.1416309 ... 138.99141631 136.80257511  
134.61373391]]
```



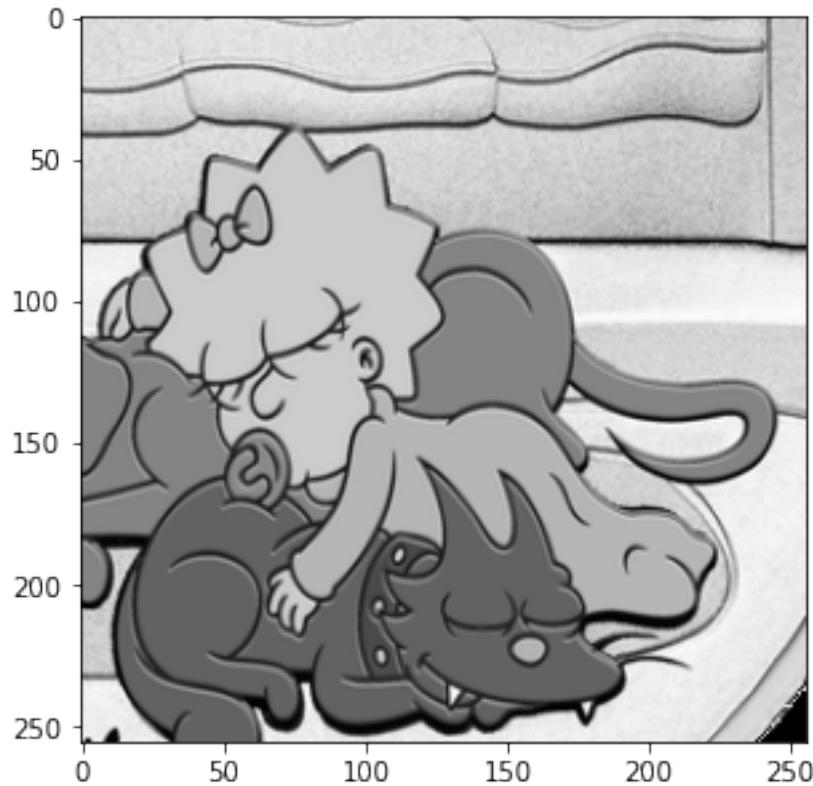
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



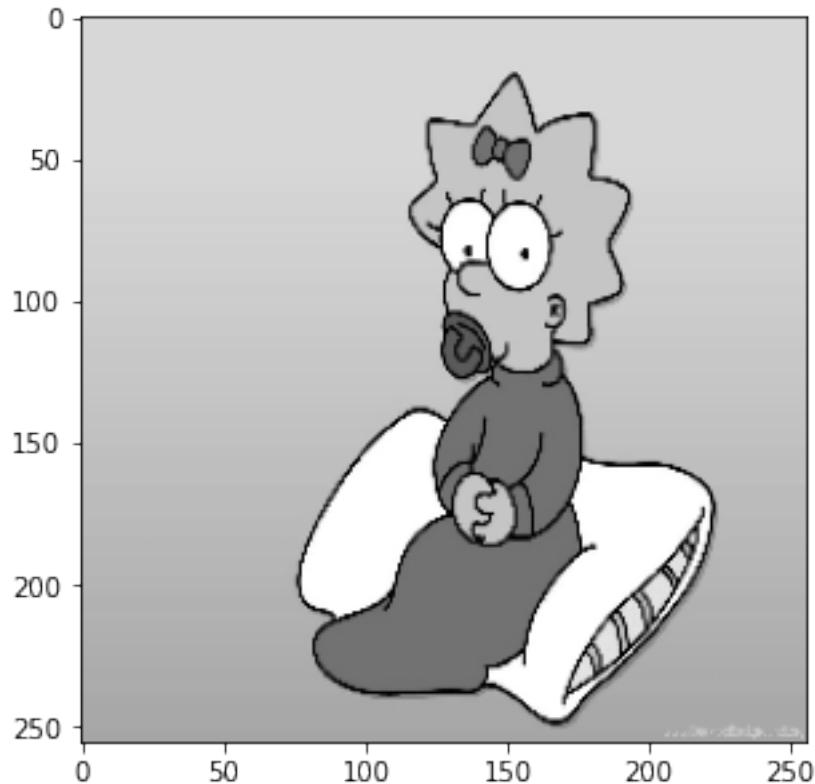
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



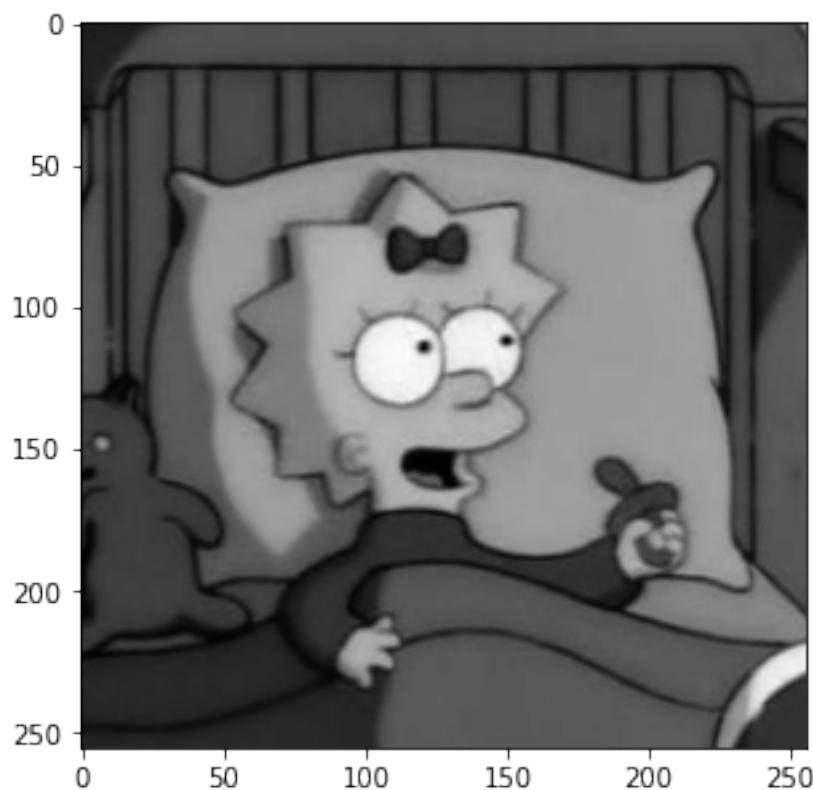
```
[[216. 227. 221. ... 206. 196. 203.]  
 [222. 227. 219. ... 194. 198. 209.]  
 [219. 224. 217. ... 198. 212. 212.]  
 ...  
 [235. 241. 255. ... 0. 0. 0.]  
 [239. 240. 134. ... 0. 0. 0.]  
 [254. 160. 0. ... 0. 0. 0.]]
```



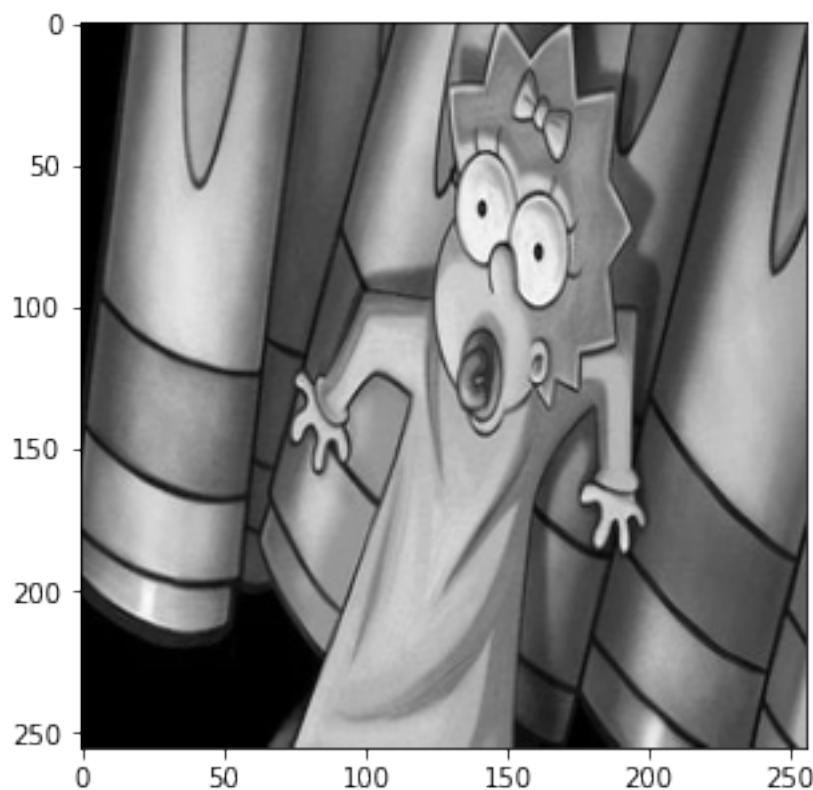
```
[[215. 215. 215. ... 215. 215. 215.]  
 [215. 215. 215. ... 215. 215. 215.]  
 [215. 215. 215. ... 215. 215. 215.]  
 ...  
 [167. 167. 167. ... 167. 198. 167.]  
 [166. 166. 166. ... 166. 198. 166.]  
 [165. 165. 165. ... 165. 165. 165.]]
```



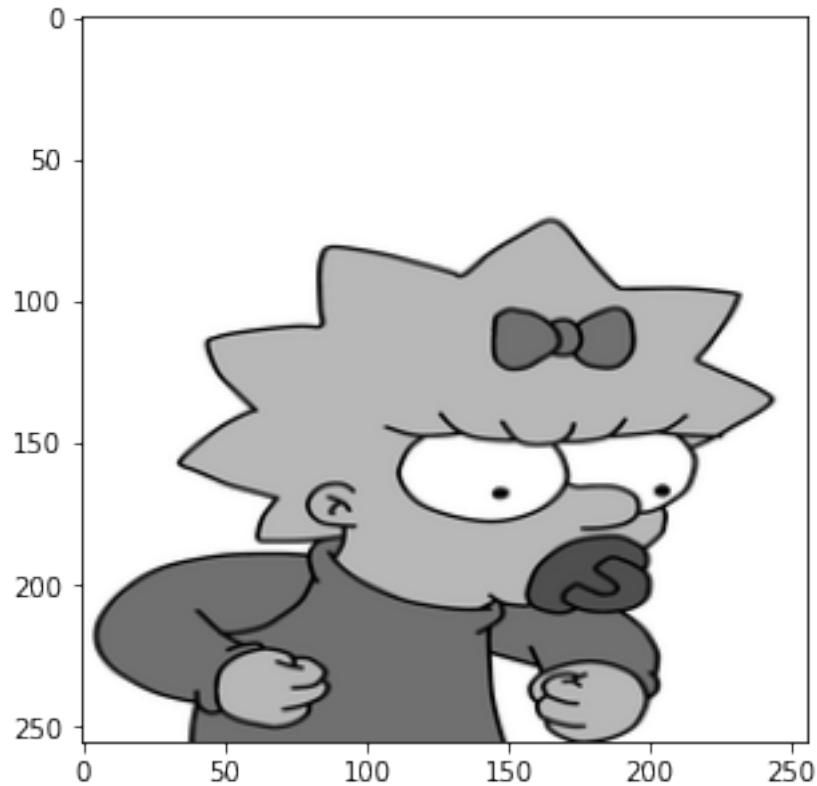
```
[[ 19.09090909  23.18181818  24.54545455 ...  91.36363636 113.18181818
 125.45454545]
 [ 21.81818182  25.90909091  28.63636364 ...   75.          105.
 121.36363636]
 [ 24.54545455  27.27272727   30.          ...  58.63636364  85.90909091
 109.09090909]
...
[ 39.54545455  38.18181818  38.18181818 ...  49.09090909  49.09090909
 49.09090909]
[ 35.45454545  36.81818182  36.81818182 ...  49.09090909  49.09090909
 49.09090909]
[ 35.45454545  35.45454545  36.81818182 ...  49.09090909  49.09090909
 49.09090909]]
```



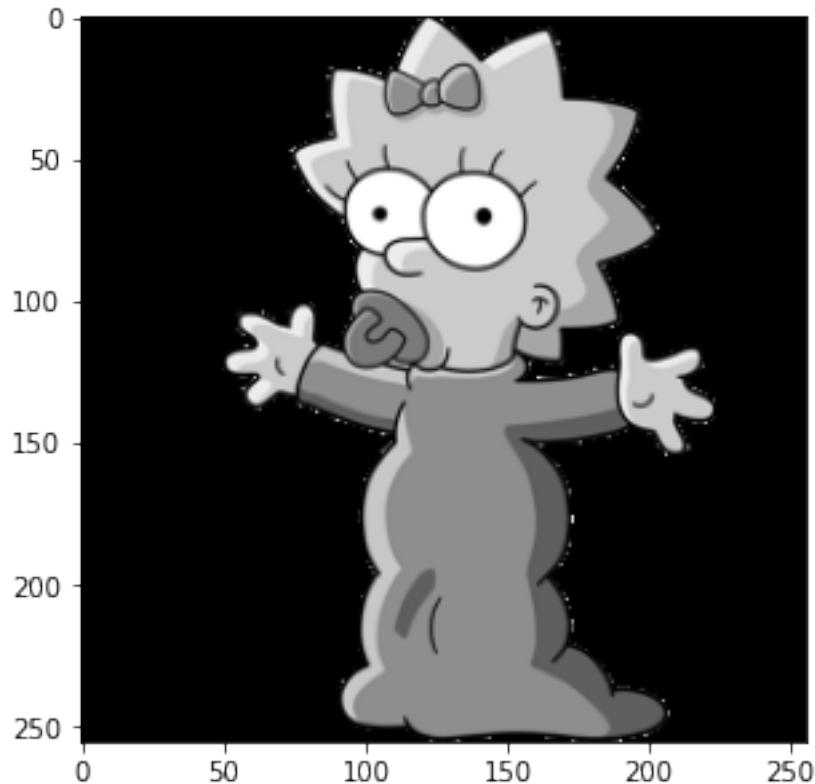
```
[[ 0.  0.  0. ... 119. 120. 118.]  
 [ 0.  0.  0. ... 120. 126. 109.]  
 [ 0.  0.  0. ... 121. 127.  87.]  
 ...  
 [ 0.  0.  0. ... 177. 178. 181.]  
 [ 0.  0.  0. ... 182. 179. 180.]  
 [ 0.  0.  0. ... 150. 179. 180.]]
```



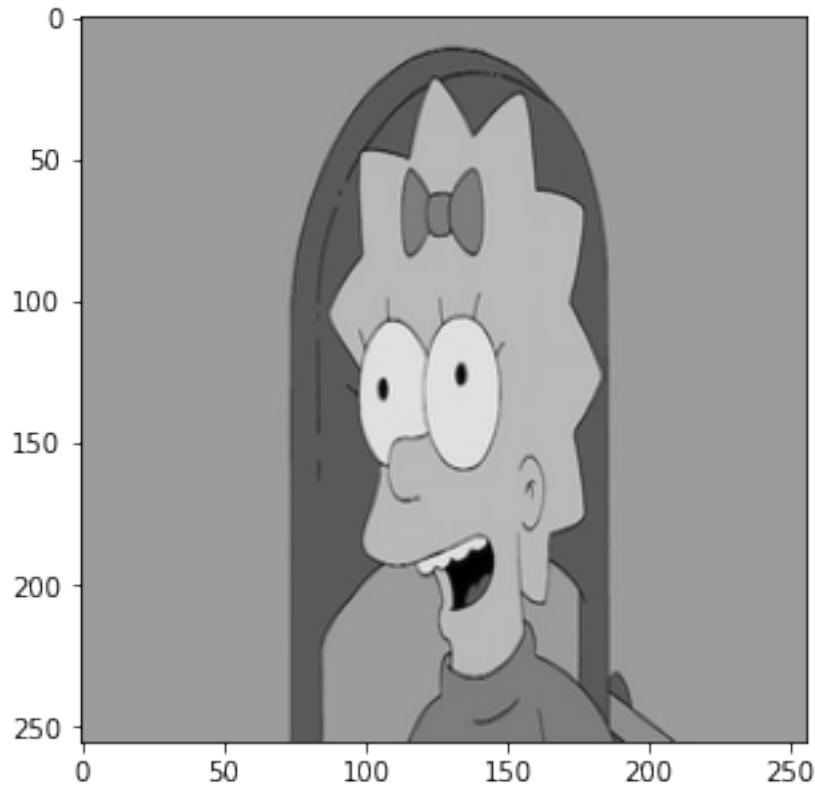
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



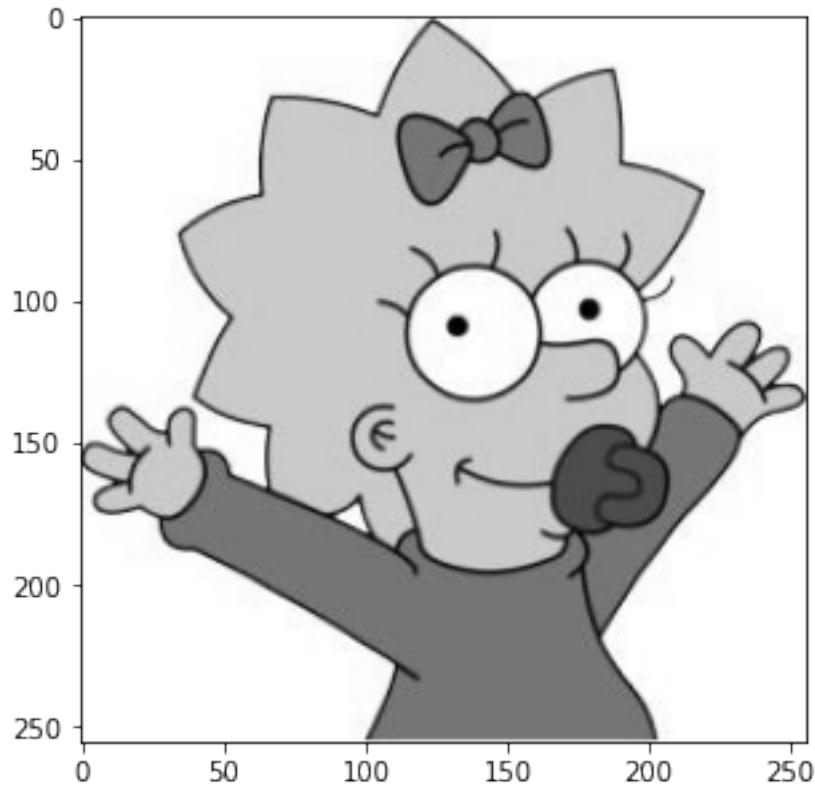
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



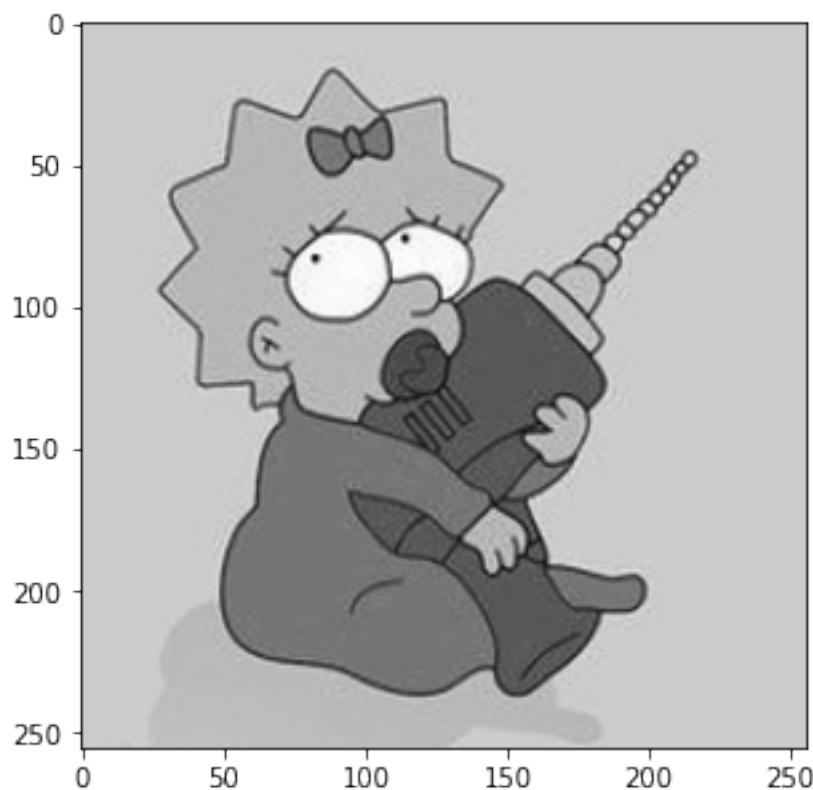
```
[[156.35802469 156.35802469 156.35802469 ... 156.35802469 156.35802469  
156.35802469]  
[156.35802469 156.35802469 156.35802469 ... 156.35802469 156.35802469  
156.35802469]  
[156.35802469 156.35802469 156.35802469 ... 156.35802469 156.35802469  
156.35802469]  
...  
[156.35802469 156.35802469 156.35802469 ... 156.35802469 156.35802469  
156.35802469]  
[156.35802469 156.35802469 156.35802469 ... 156.35802469 156.35802469  
156.35802469]  
[156.35802469 156.35802469 156.35802469 ... 156.35802469 156.35802469  
156.35802469]]
```



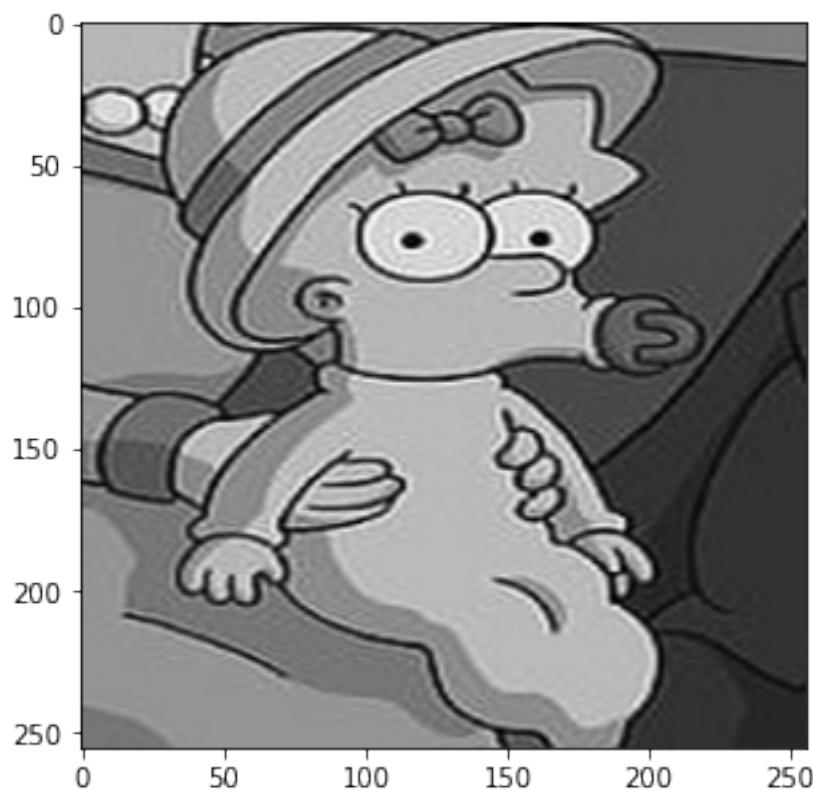
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



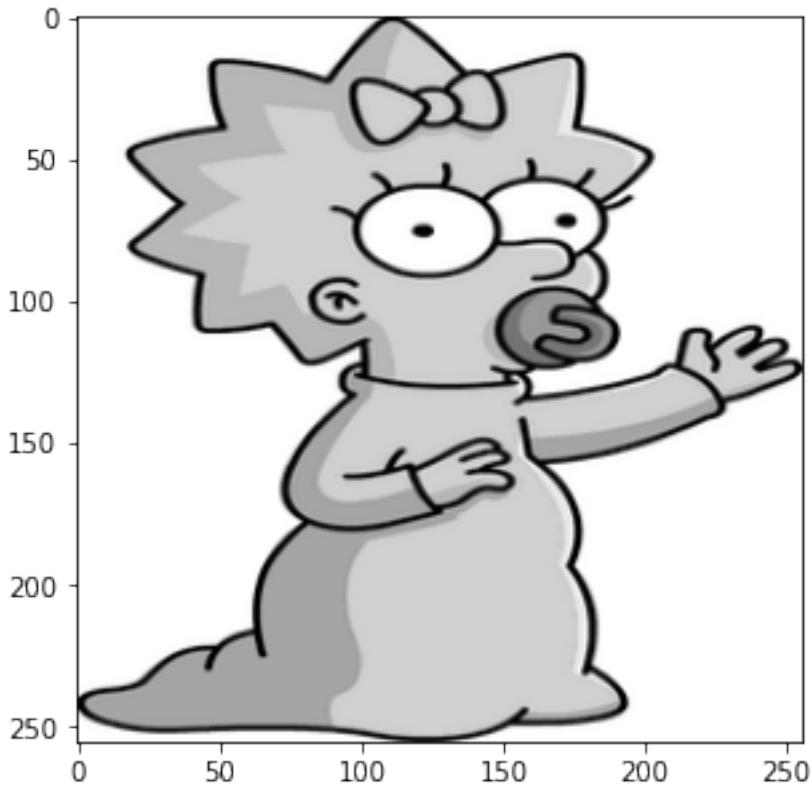
```
[[204. 204. 204. ... 204. 204. 204.]  
 [204. 204. 204. ... 204. 204. 204.]  
 [204. 204. 204. ... 204. 204. 204.]  
 ...  
 [205. 205. 204. ... 204. 204. 204.]  
 [202. 203. 204. ... 204. 204. 204.]  
 [195. 198. 202. ... 204. 204. 204.]]
```



```
[[183. 183. 183. ... 127. 127. 127.]  
 [183. 183. 183. ... 127. 127. 127.]  
 [183. 183. 183. ... 127. 127. 127.]  
 ...  
 [119. 119. 119. ... 39. 39. 39.]  
 [120. 120. 119. ... 39. 39. 39.]  
 [121. 121. 119. ... 39. 39. 39.]]
```

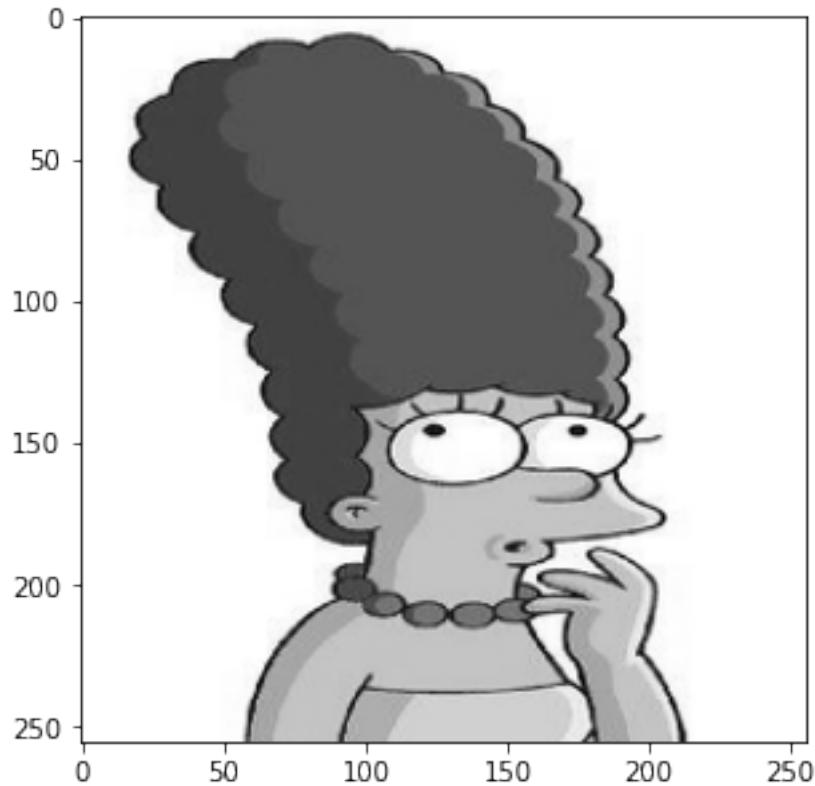


```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```

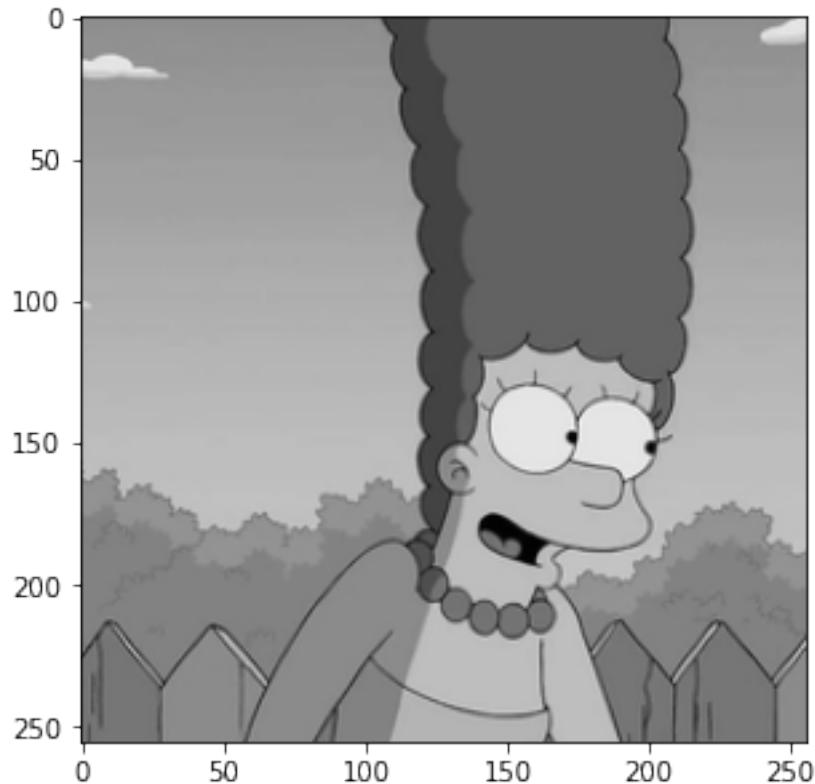


```
[ ]: for index, image in enumerate(grayScaleMarge):
    image = np.array(image)
    norm = image/(image.max()/255)
    print(norm)
    #convert back to image
    im = Image.fromarray(norm)
    im = im.convert('RGB')
    im.save("normalizedMarge/image-"+str(index)+".jpg")
    plt.imshow(im, cmap=plt.cm.gray)
    plt.show()
```

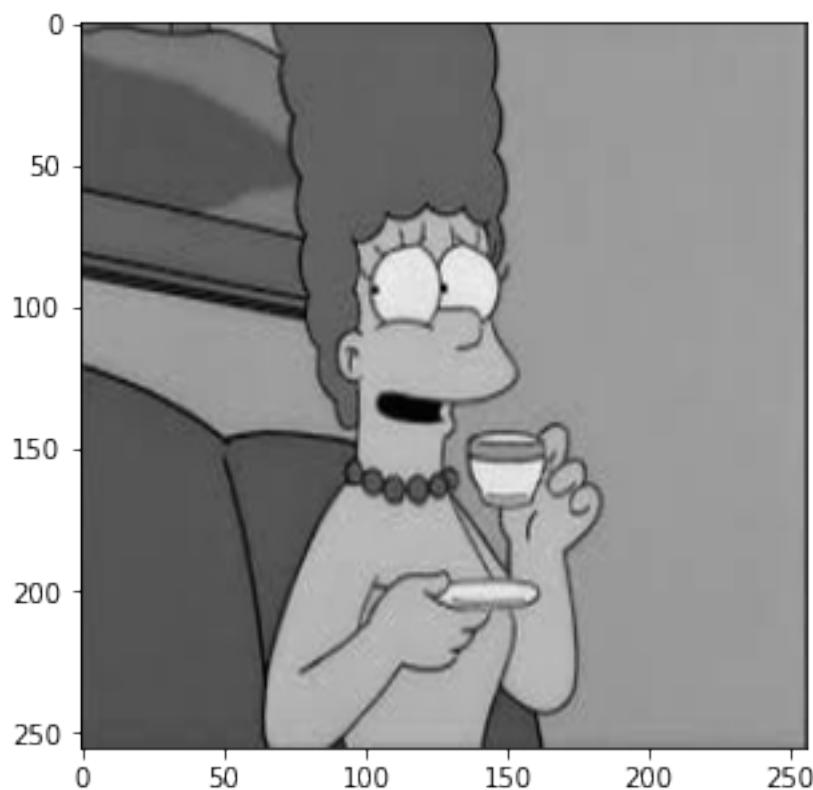
```
[[255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 ...
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]
 [255. 255. 255. ... 255. 255. 255.]]
```



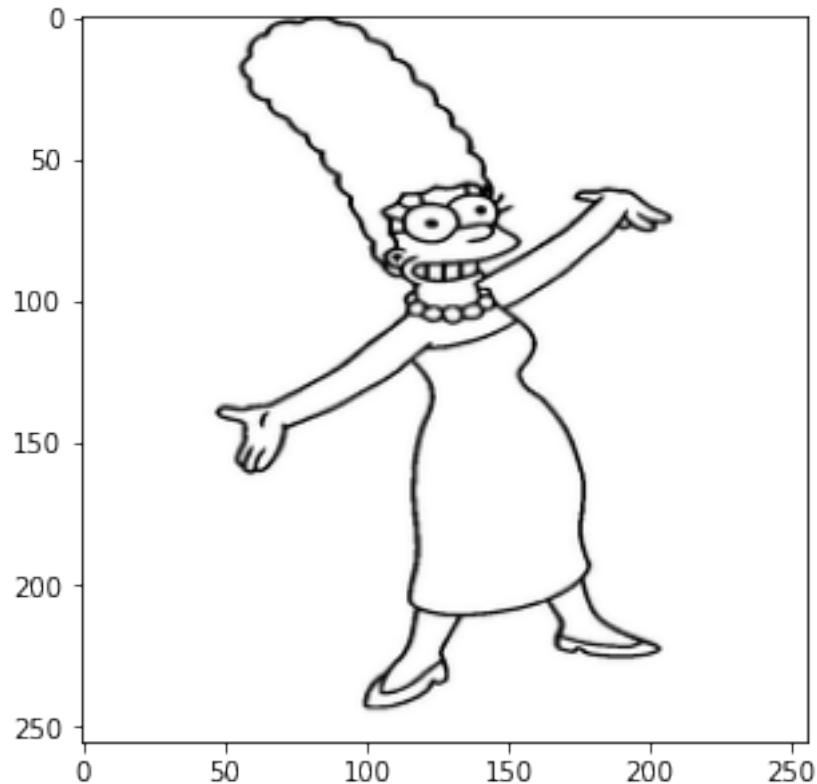
```
[[141.3253012 141.3253012 141.3253012 ... 176.14457831 185.36144578  
185.36144578]  
[141.3253012 141.3253012 141.3253012 ... 237.59036145 233.4939759  
232.46987952]  
[142.34939759 142.34939759 142.34939759 ... 227.34939759 227.34939759  
229.39759036]  
...  
[117.77108434 121.86746988 108.55421687 ... 139.27710843 139.27710843  
139.27710843]  
[117.77108434 121.86746988 108.55421687 ... 139.27710843 139.27710843  
139.27710843]  
[117.77108434 121.86746988 108.55421687 ... 139.27710843 139.27710843  
139.27710843]]
```



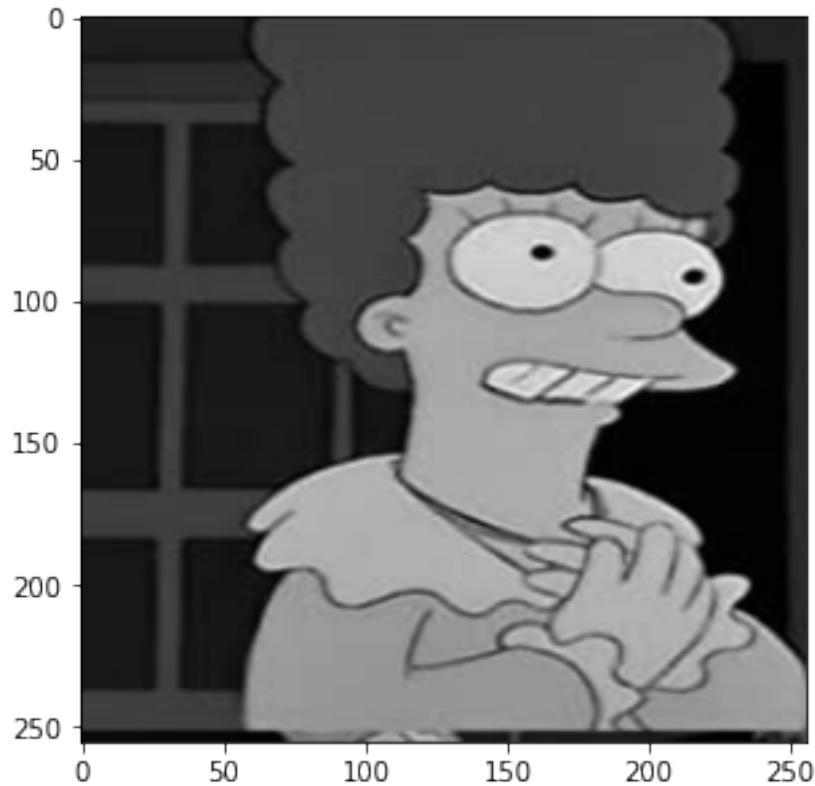
```
[[ 71.23481781 112.53036437 109.43319838 ... 158.98785425 165.18218623
  92.91497976]
 [ 77.4291498 119.75708502 117.69230769 ... 158.98785425 165.18218623
  92.91497976]
 [ 57.81376518 100.1417004 96.01214575 ... 158.98785425 165.18218623
  92.91497976]
 ...
 [ 49.55465587 77.4291498 80.52631579 ... 154.8582996 155.89068826
  94.97975709]
 [ 49.55465587 76.39676113 79.49392713 ... 144.53441296 145.56680162
  84.65587045]
 [ 49.55465587 76.39676113 79.49392713 ... 138.34008097 139.37246964
  78.46153846]]
```



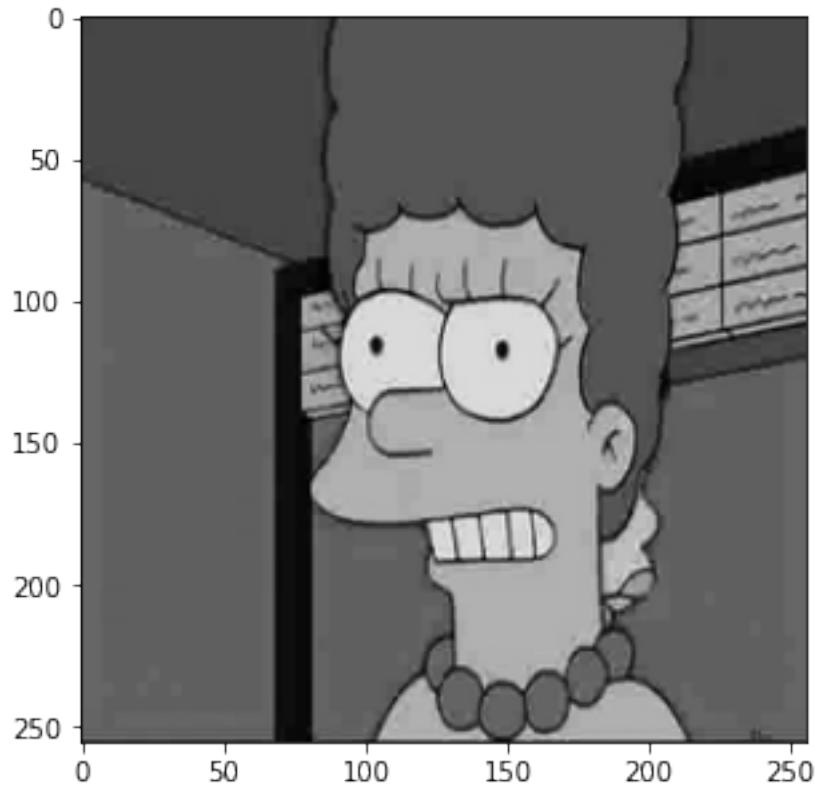
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



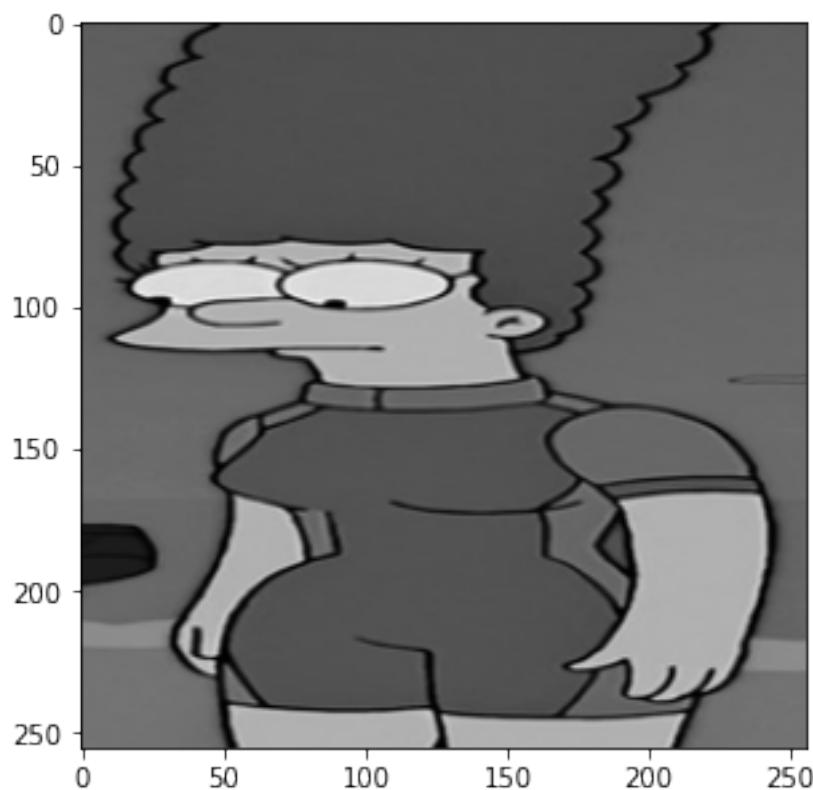
```
[[35.93181818 35.93181818 35.93181818 ... 34.77272727 34.77272727  
35.93181818]  
[34.77272727 34.77272727 34.77272727 ... 34.77272727 34.77272727  
35.93181818]  
[32.45454545 32.45454545 32.45454545 ... 34.77272727 34.77272727  
35.93181818]  
...  
[ 9.27272727 9.27272727 9.27272727 ... 64.90909091 76.5  
79.97727273]  
[10.43181818 10.43181818 10.43181818 ... 48.68181818 67.22727273  
74.18181818]  
[10.43181818 10.43181818 10.43181818 ... 38.25 60.27272727  
69.54545455]]
```



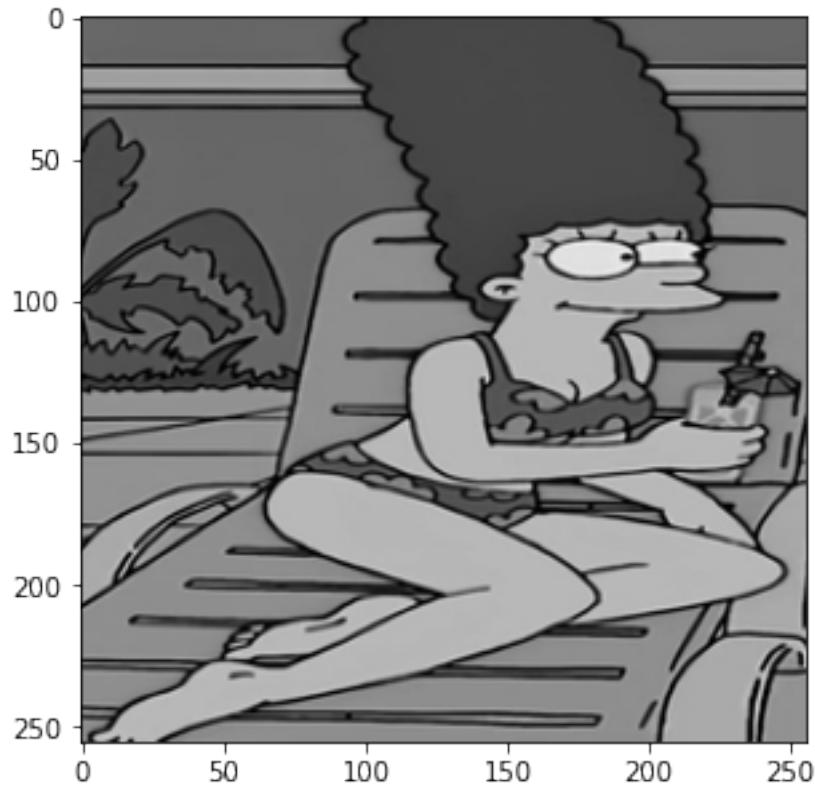
```
[[ 67.32  67.32  67.32 ...  70.38  70.38  70.38]
 [ 67.32  67.32  67.32 ...  70.38  70.38  70.38]
 [ 67.32  67.32  67.32 ...  70.38  70.38  70.38]
 ...
 [ 92.82  92.82  92.82 ...  92.82  92.82  92.82]
 [111.18  111.18  111.18 ... 111.18  111.18  111.18]
 [ 58.14  58.14  58.14 ...  58.14  58.14  58.14]]
```



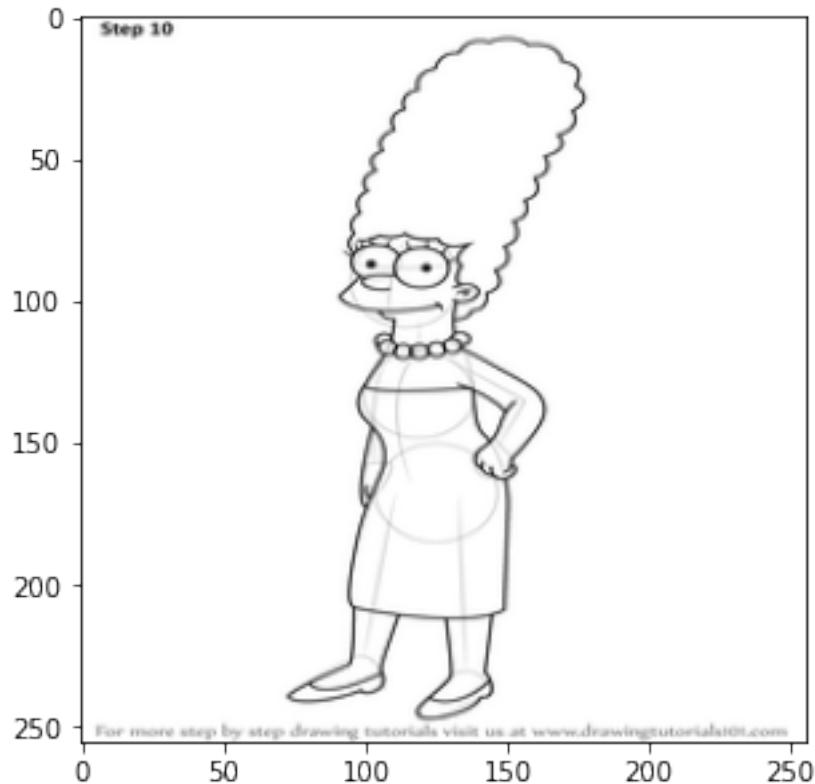
```
[[ 91.51209677  91.51209677  91.51209677 ...  91.51209677  91.51209677  
  92.54032258]  
 [ 91.51209677  91.51209677  92.54032258 ...  93.56854839  92.54032258  
  92.54032258]  
 [ 94.59677419  94.59677419  93.56854839 ...  93.56854839  93.56854839  
  93.56854839]  
 ...  
 [114.13306452 115.16129032 115.16129032 ... 114.13306452 114.13306452  
  115.16129032]  
 [116.18951613 116.18951613 115.16129032 ... 115.16129032 115.16129032  
  115.16129032]  
 [116.18951613 116.18951613 116.18951613 ... 116.18951613 116.18951613  
  116.18951613]]
```



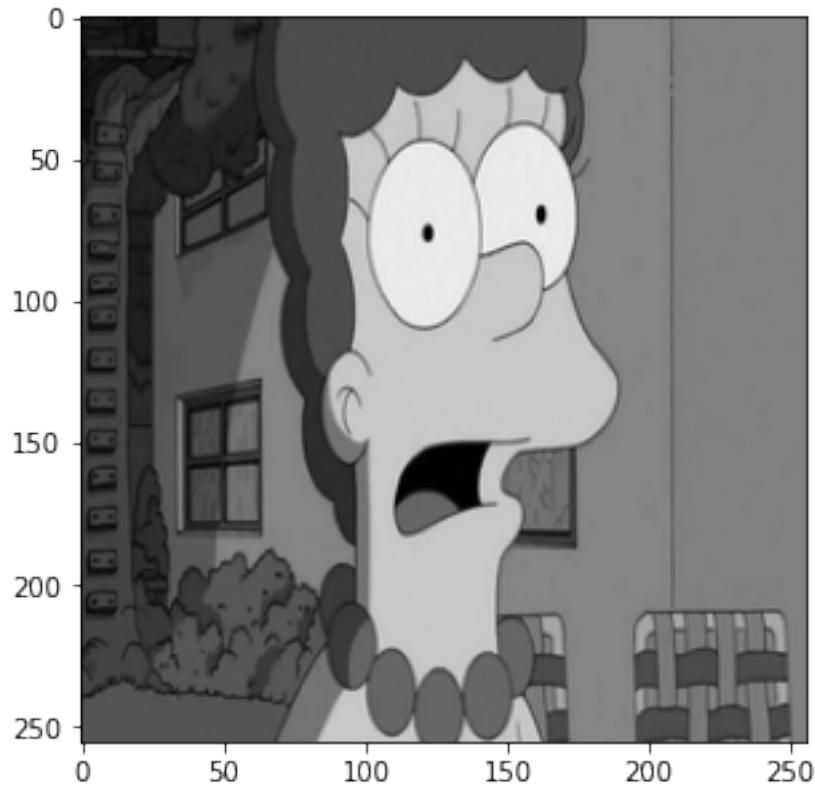
```
[[105. 110. 106. ... 102. 102. 102.]  
 [105. 110. 106. ... 102. 102. 102.]  
 [103. 109. 103. ... 102. 102. 102.]  
 ...  
 [139. 143. 112. ... 146. 146. 146.]  
 [147. 154. 115. ... 147. 147. 147.]  
 [113. 125. 106. ... 147. 147. 147.]]
```



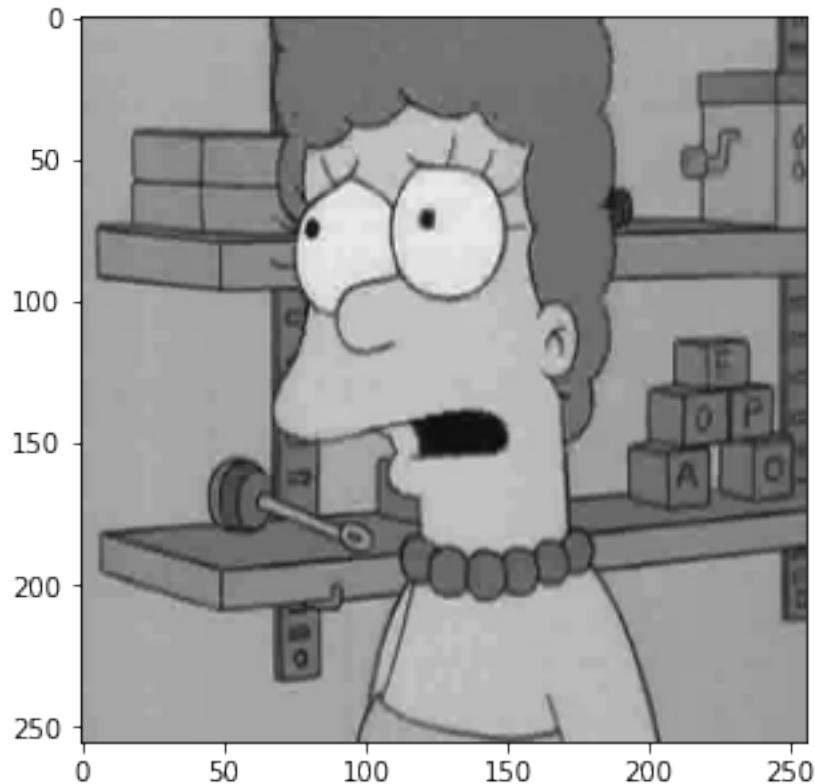
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]]
```



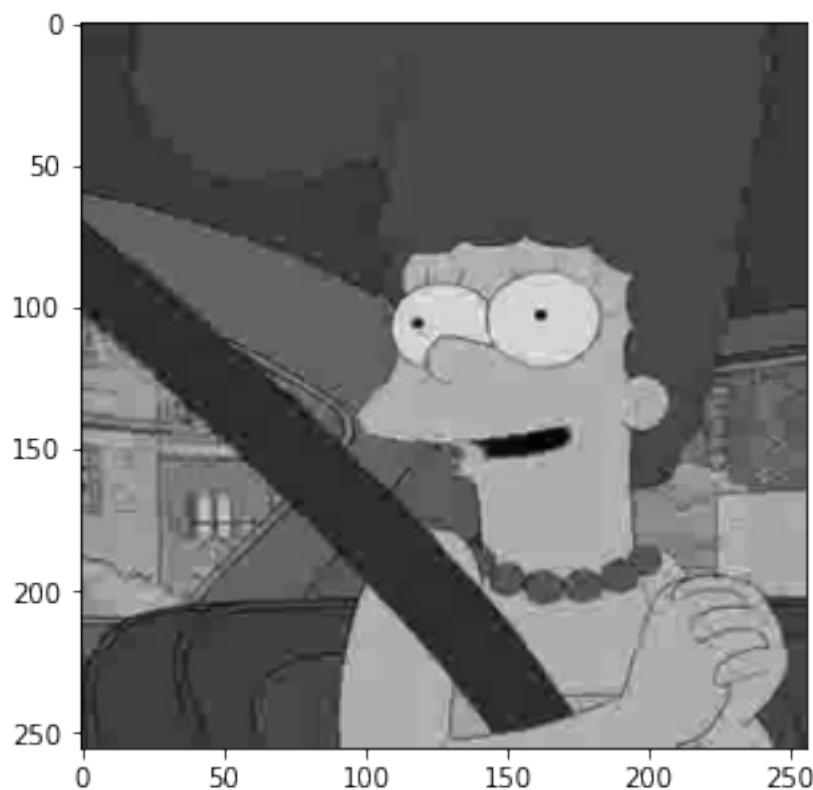
```
[[ 18.84782609  12.19565217  18.84782609 ... 129.7173913  129.7173913
 129.7173913 ]
 [ 26.60869565  28.82608696  32.15217391 ... 129.7173913  129.7173913
 129.7173913 ]
 [ 26.60869565  33.26086957  32.15217391 ... 129.7173913  129.7173913
 129.7173913 ]
 ...
 [ 85.36956522  85.36956522  85.36956522 ... 127.5          130.82608696
 130.82608696]
 [ 84.26086957  85.36956522  85.36956522 ... 128.60869565 130.82608696
 129.7173913 ]
 [ 73.17391304  86.47826087  85.36956522 ... 128.60869565 129.7173913
 128.60869565]]
```



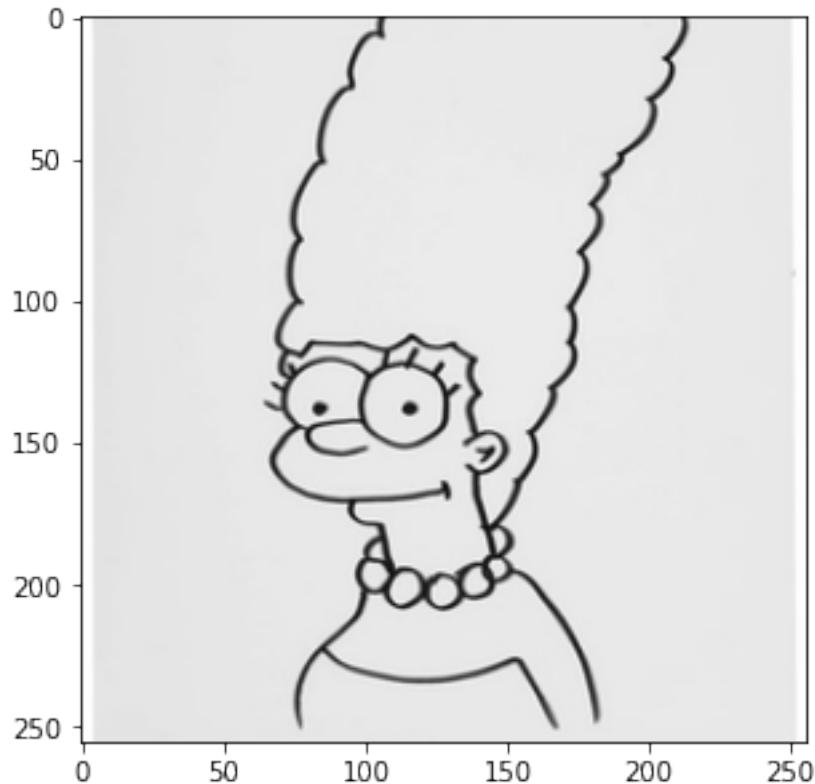
```
[[ 77.51101322  75.26431718  74.14096916 ...  97.73127753  51.67400881
  106.71806167]
 [166.25550661 167.37885463 167.37885463 ...  95.4845815   61.78414097
 101.10132159]
 [174.11894273 175.24229075 174.11894273 ...  93.23788546  78.63436123
 94.36123348]
...
[160.63876652 161.76211454 160.63876652 ... 155.02202643 155.02202643
 155.02202643]
[160.63876652 161.76211454 160.63876652 ... 155.02202643 155.02202643
 155.02202643]
[160.63876652 161.76211454 160.63876652 ... 155.02202643 155.02202643
 155.02202643]]
```



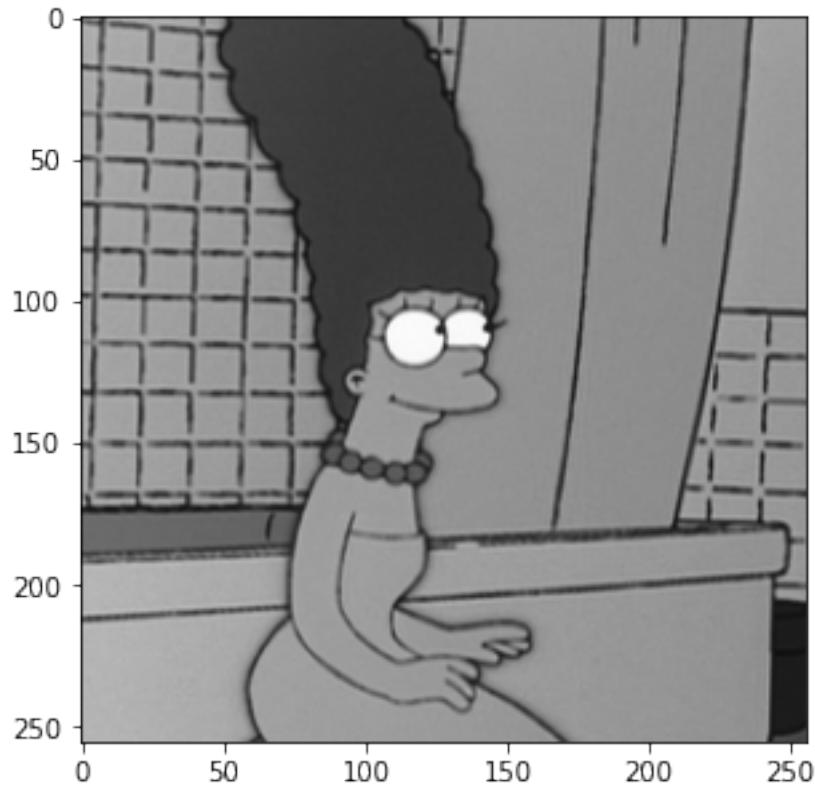
```
[[60.23622047 60.23622047 60.23622047 ... 60.23622047 60.23622047  
60.23622047]  
[60.23622047 60.23622047 60.23622047 ... 60.23622047 60.23622047  
60.23622047]  
[60.23622047 60.23622047 60.23622047 ... 60.23622047 60.23622047  
60.23622047]  
...  
[71.27952756 55.21653543 40.15748031 ... 60.23622047 53.20866142  
50.19685039]  
[71.27952756 55.21653543 40.15748031 ... 60.23622047 53.20866142  
50.19685039]  
[71.27952756 55.21653543 40.15748031 ... 60.23622047 53.20866142  
50.19685039]]
```



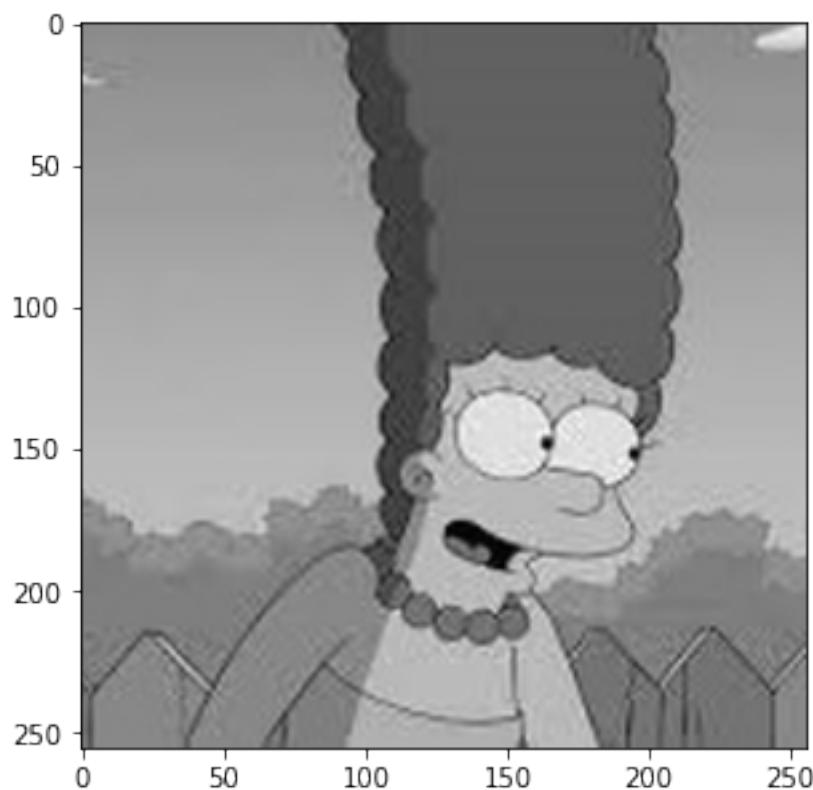
```
[[254. 254. 255. ... 255. 254. 255.]  
 [254. 254. 255. ... 255. 254. 255.]  
 [254. 254. 255. ... 255. 254. 255.]  
 ...  
 [254. 255. 254. ... 255. 254. 255.]  
 [254. 255. 254. ... 255. 254. 255.]  
 [254. 255. 254. ... 255. 254. 255.]]
```



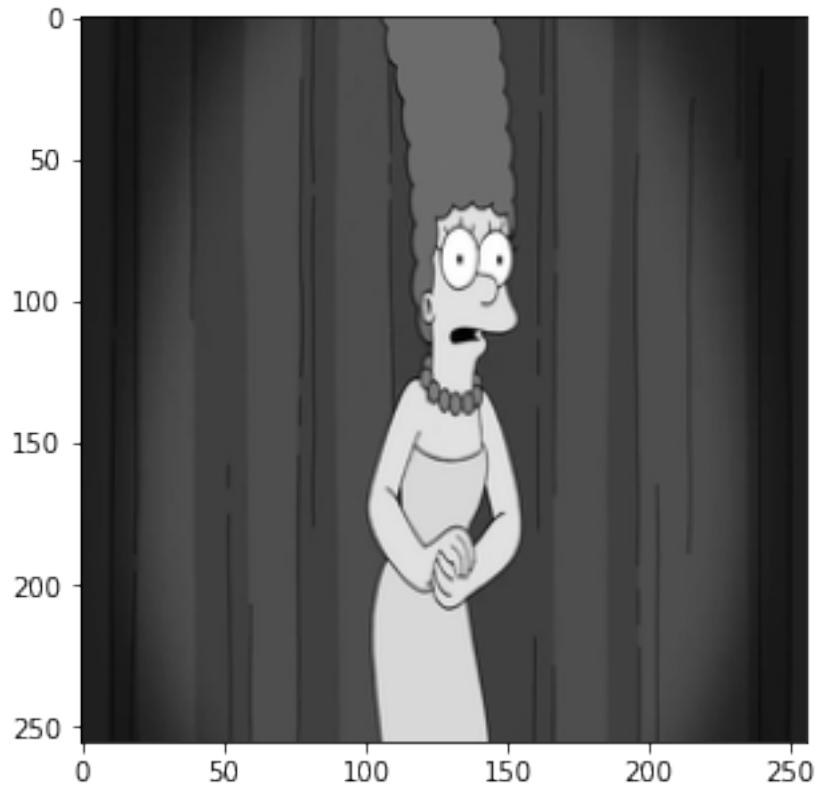
```
[[160.62992126 166.65354331 168.66141732 ... 158.62204724 159.62598425
158.62204724]
[108.42519685 126.49606299 123.48425197 ... 158.62204724 158.62204724
160.62992126]
[103.40551181 117.46062992 102.4015748 ... 160.62992126 159.62598425
160.62992126]
...
[185.72834646 180.70866142 180.70866142 ... 73.28740157 74.29133858
72.28346457]
[186.73228346 181.71259843 182.71653543 ... 67.26377953 67.26377953
69.27165354]
[183.72047244 181.71259843 183.72047244 ... 68.26771654 70.27559055
70.27559055]]
```



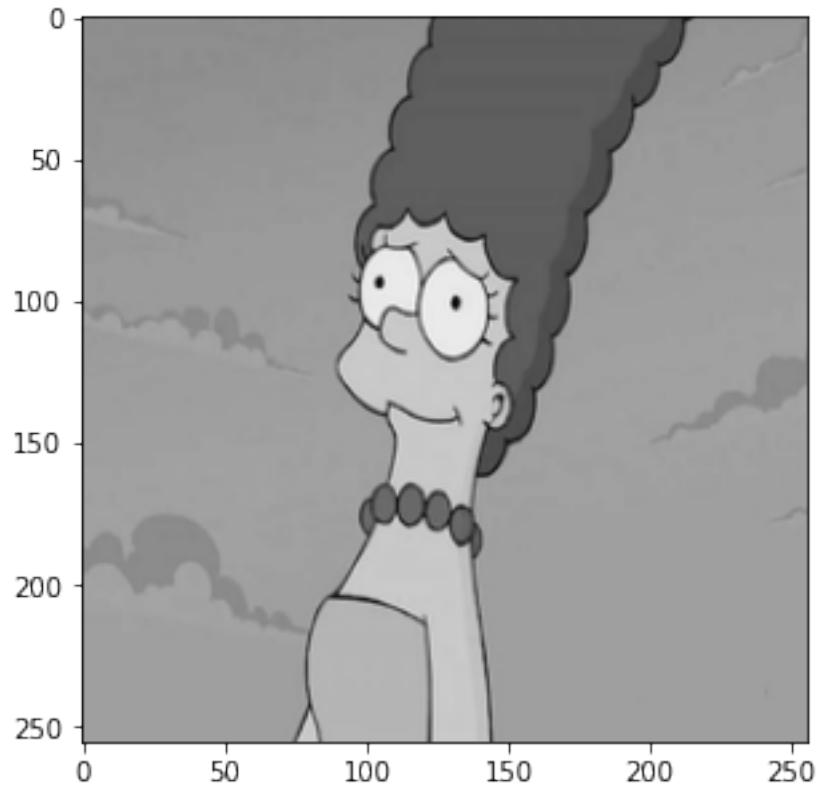
```
[[138. 138. 138. ... 176. 186. 188.]  
 [138. 138. 138. ... 206. 208. 206.]  
 [138. 138. 138. ... 241. 234. 229.]  
 ...  
 [119. 124. 114. ... 136. 136. 136.]  
 [119. 124. 114. ... 136. 136. 136.]  
 [119. 124. 114. ... 136. 136. 136.]]
```



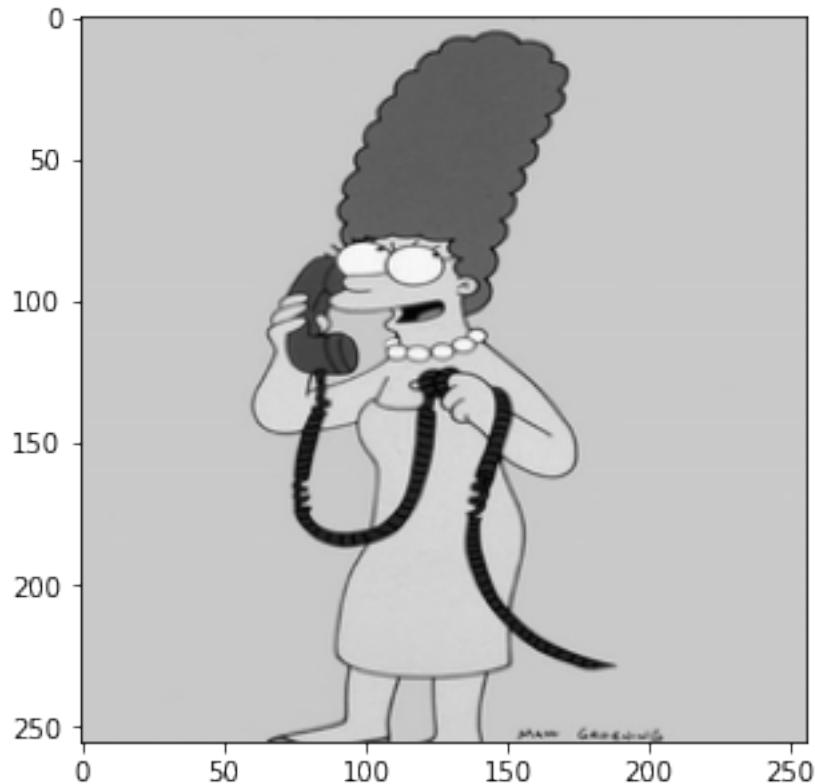
```
[[31. 31. 31. ... 31. 31. 31.]
 [31. 31. 31. ... 31. 31. 31.]
 [31. 31. 31. ... 31. 31. 31.]
 ...
 [31. 31. 31. ... 31. 31. 31.]
 [31. 31. 31. ... 31. 31. 31.]
 [31. 31. 31. ... 31. 31. 31.]]
```



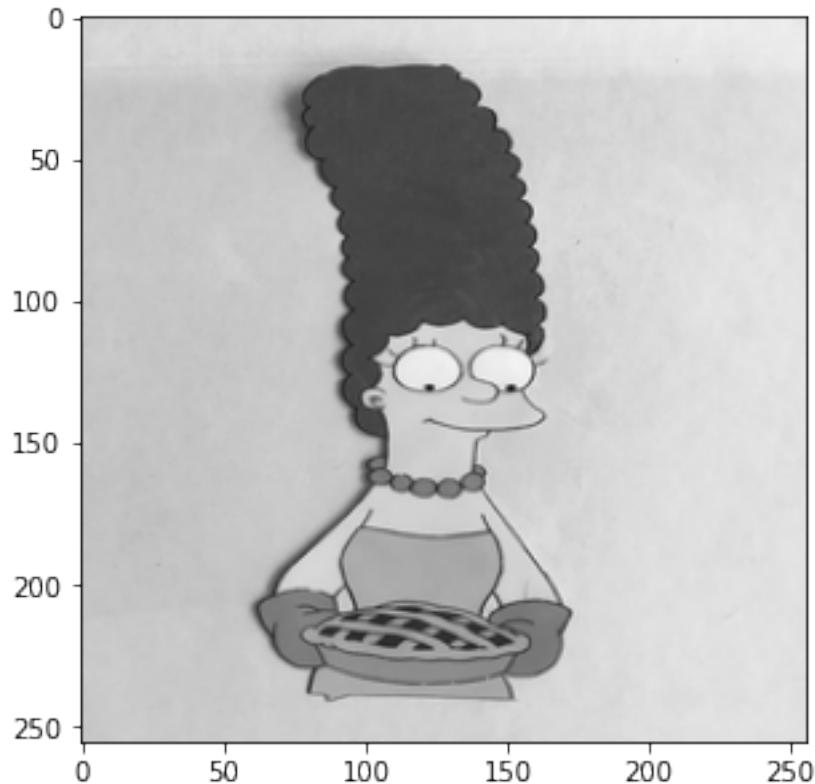
```
[[143.69047619 143.69047619 143.69047619 ... 146.72619048 145.71428571
146.72619048]
[144.70238095 144.70238095 144.70238095 ... 147.73809524 149.76190476
147.73809524]
[143.69047619 143.69047619 143.69047619 ... 148.75          145.71428571
141.66666667]
...
[159.88095238 159.88095238 159.88095238 ... 159.88095238 159.88095238
159.88095238]
[159.88095238 159.88095238 159.88095238 ... 159.88095238 159.88095238
159.88095238]
[159.88095238 159.88095238 159.88095238 ... 159.88095238 159.88095238
159.88095238]]
```



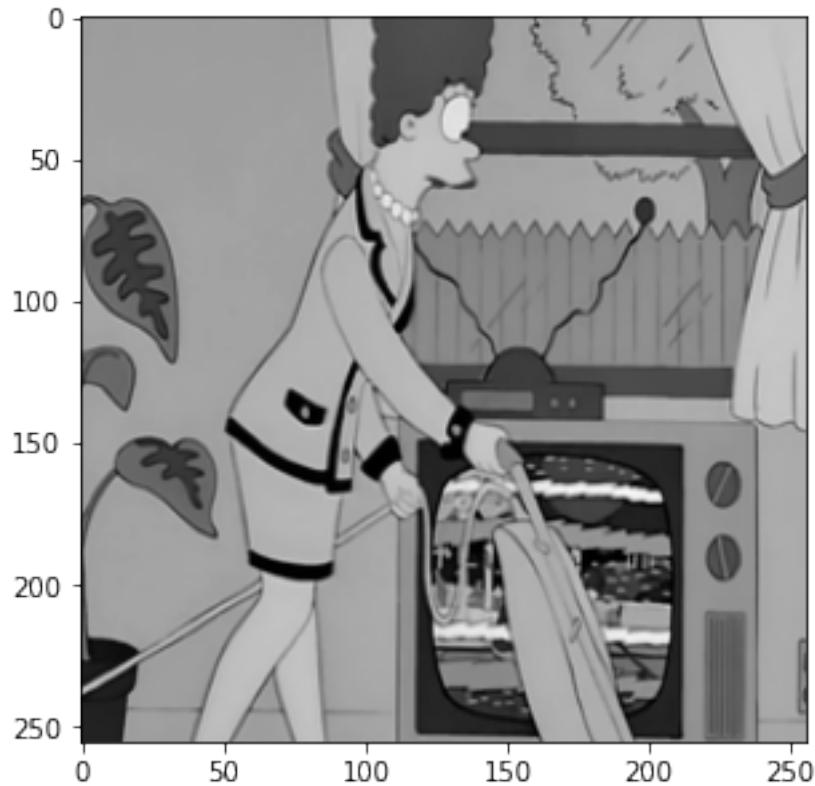
```
[[202. 202. 202. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]  
 [202. 202. 202. ... 201. 201. 201.]  
 ...  
 [202. 202. 201. ... 201. 201. 201.]  
 [201. 201. 201. ... 201. 201. 201.]  
 [202. 202. 202. ... 201. 201. 201.]]
```



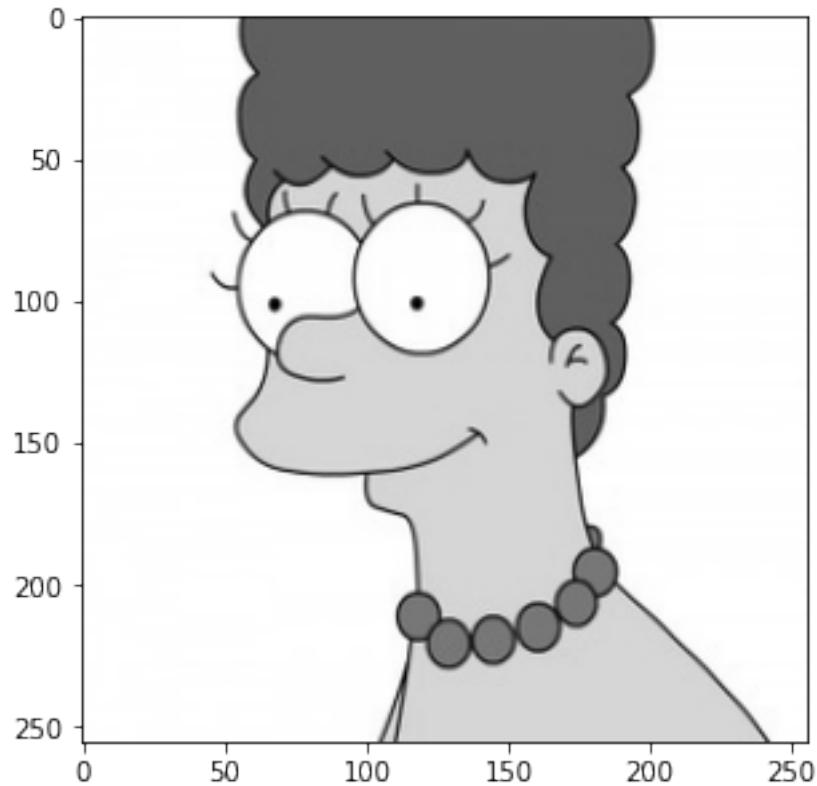
```
[[210.92592593 214.07407407 214.07407407 ... 234.01234568 232.96296296
 232.96296296]
[209.87654321 211.97530864 211.97530864 ... 232.96296296 232.96296296
 232.96296296]
[209.87654321 209.87654321 210.92592593 ... 231.91358025 232.96296296
 232.96296296]
...
[201.48148148 198.33333333 199.38271605 ... 224.56790123 224.56790123
 224.56790123]
[201.48148148 199.38271605 199.38271605 ... 224.56790123 224.56790123
 224.56790123]
[200.43209877 200.43209877 200.43209877 ... 224.56790123 224.56790123
 222.4691358 ]]
```



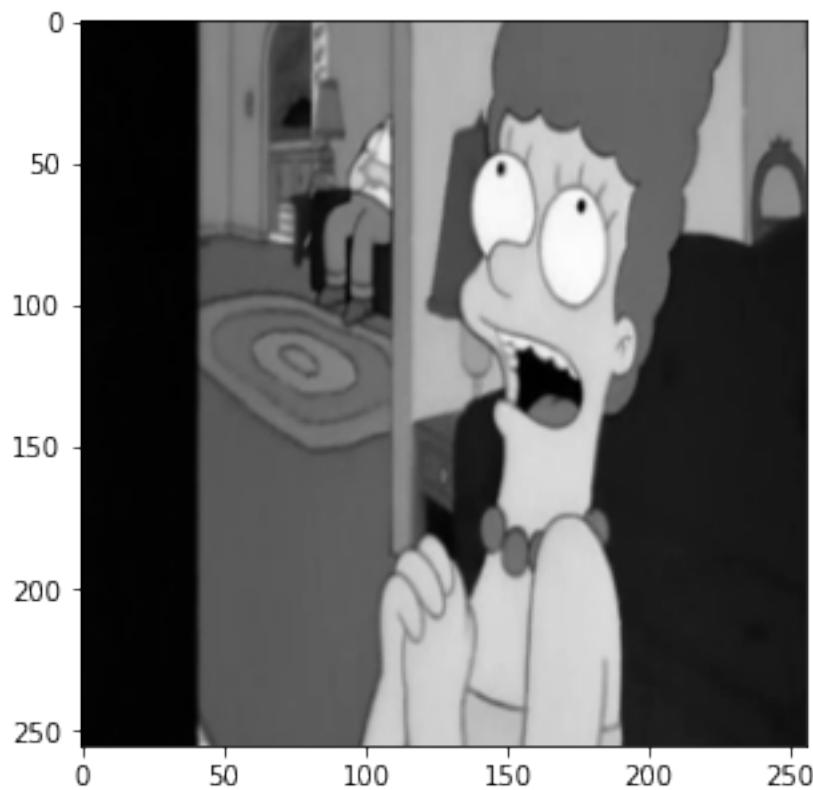
```
[[156.76229508 156.76229508 156.76229508 ... 194.3852459 194.3852459  
194.3852459 ]  
[156.76229508 156.76229508 156.76229508 ... 194.3852459 194.3852459  
194.3852459 ]  
[156.76229508 156.76229508 156.76229508 ... 194.3852459 194.3852459  
194.3852459 ]  
...  
[ 34.48770492 34.48770492 34.48770492 ... 167.21311475 168.25819672  
168.25819672]  
[ 34.48770492 34.48770492 34.48770492 ... 167.21311475 167.21311475  
168.25819672]  
[ 34.48770492 34.48770492 34.48770492 ... 167.21311475 168.25819672  
168.25819672]]
```



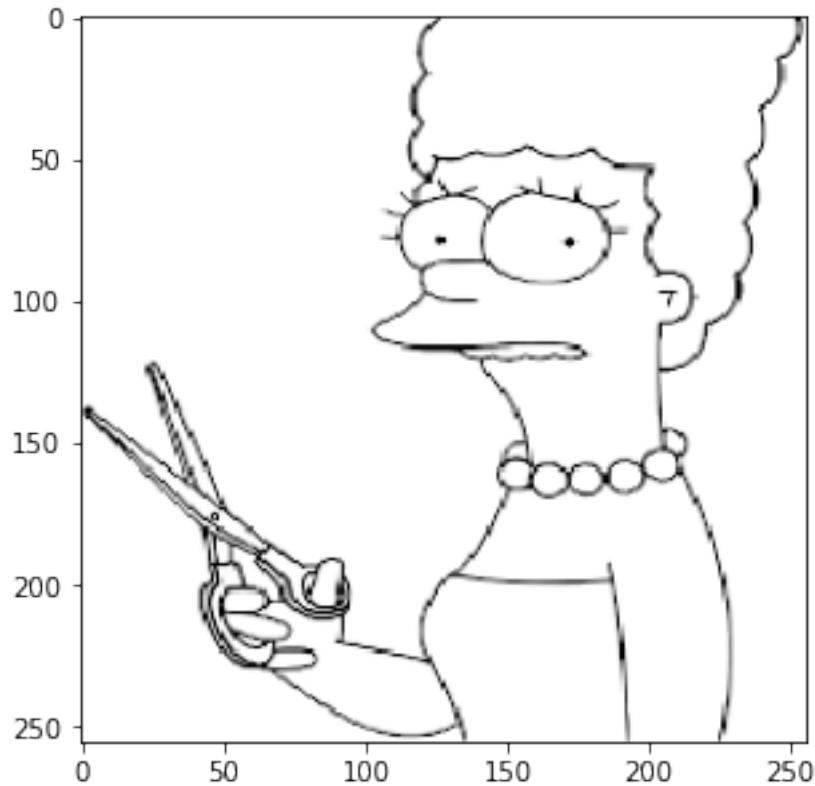
```
[[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
[255. 255. 255. ... 255. 255. 255.]  
...  
[254. 254. 254. ... 255. 255. 255.]  
[254. 254. 254. ... 255. 255. 255.]  
[254. 254. 254. ... 255. 255. 255.]]
```



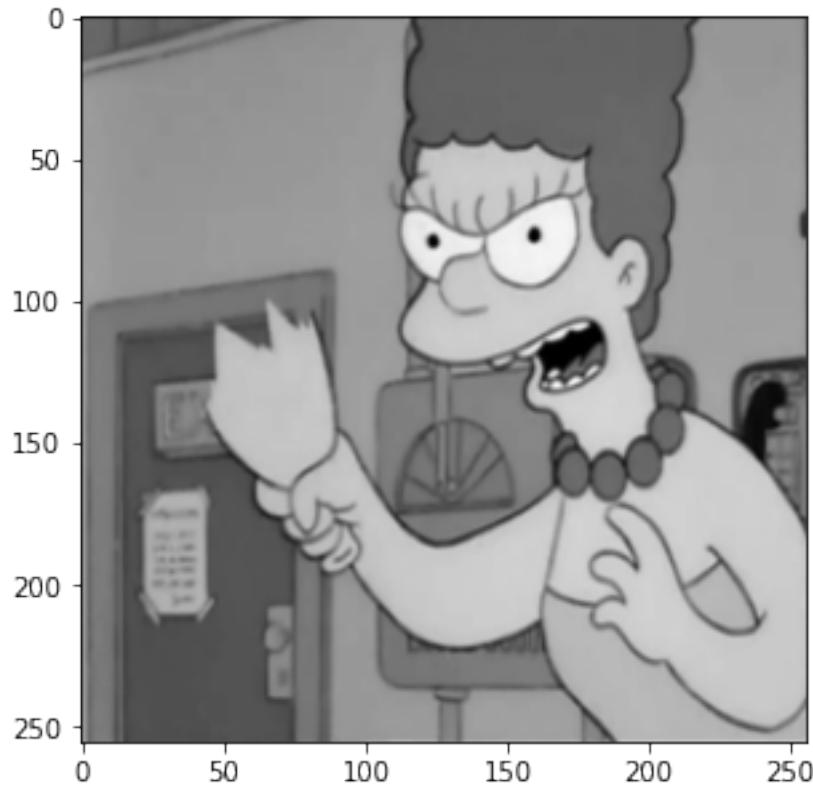
```
[[ 3.49315068  3.49315068  3.49315068 ... 137.39726027 135.06849315
  115.2739726 ]
 [ 3.49315068  3.49315068  3.49315068 ... 137.39726027 135.06849315
  115.2739726 ]
 [ 3.49315068  3.49315068  3.49315068 ... 137.39726027 135.06849315
  115.2739726 ]
 ...
 [ 3.49315068  3.49315068  4.65753425 ... 18.63013699 18.63013699
  18.63013699]
 [ 3.49315068  3.49315068  4.65753425 ... 17.46575342 18.63013699
  18.63013699]
 [ 3.49315068  3.49315068  4.65753425 ... 18.63013699 19.79452055
  19.79452055]]
```



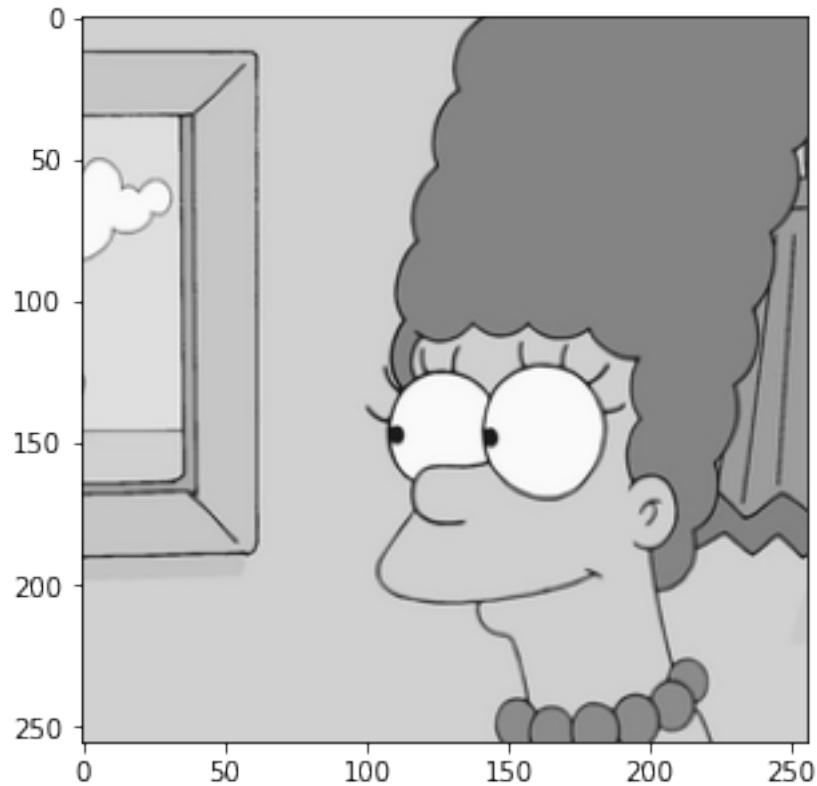
```
[[255. 255. 255. ... 50. 255. 255.]  
 [255. 255. 255. ... 0. 203. 255.]  
 [255. 255. 255. ... 117. 148. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



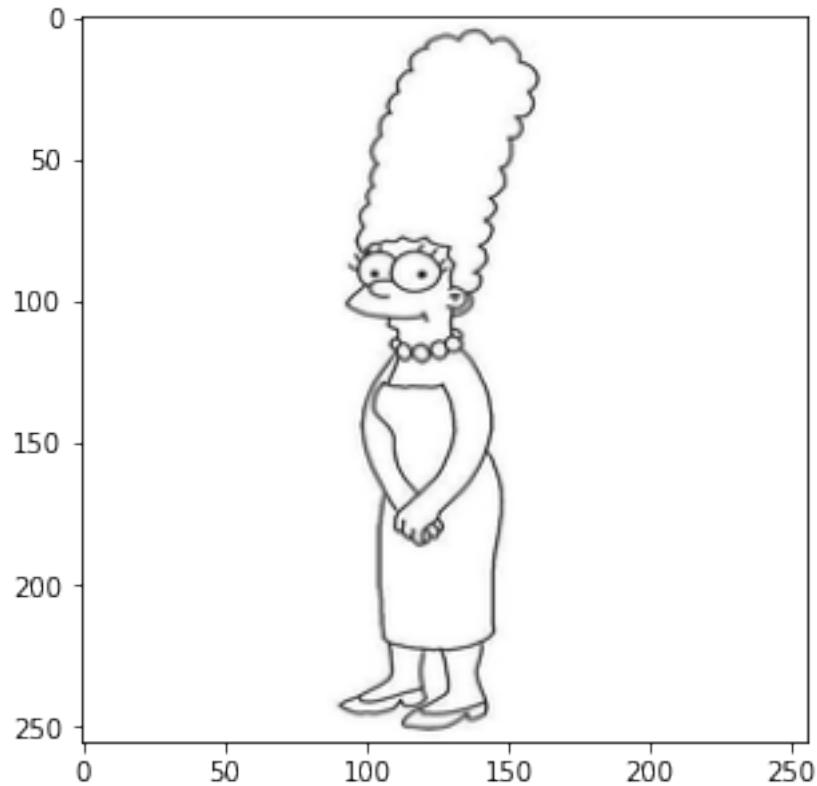
```
[[ 53.37209302  55.74418605  56.93023256 ...  80.65116279  79.46511628
  65.23255814]
 [ 97.25581395 100.81395349 103.18604651 ... 147.06976744 143.51162791
 119.79069767]
 [ 99.62790698 104.37209302 106.74418605 ... 149.44186047 147.06976744
 119.79069767]
...
[157.74418605 156.55813953 130.46511628 ... 181.46511628 177.90697674
179.09302326]
[158.93023256 157.74418605 130.46511628 ... 180.27906977 180.27906977
181.46511628]
[160.11627907 157.74418605 131.65116279 ... 193.3255814 193.3255814
193.3255814 ]]
```



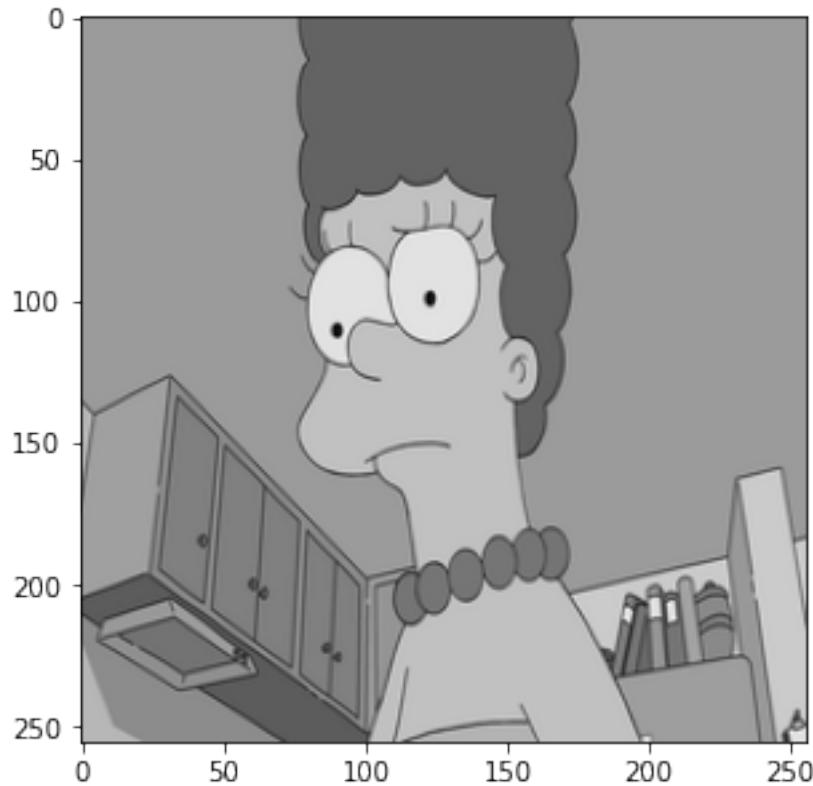
```
[[209. 209. 209. ... 133. 133. 133.]  
 [208. 208. 208. ... 133. 132. 133.]  
 [209. 209. 209. ... 133. 132. 133.]  
 ...  
 [209. 209. 209. ... 209. 209. 209.]  
 [209. 209. 209. ... 209. 209. 209.]  
 [209. 209. 209. ... 209. 209. 209.]]
```



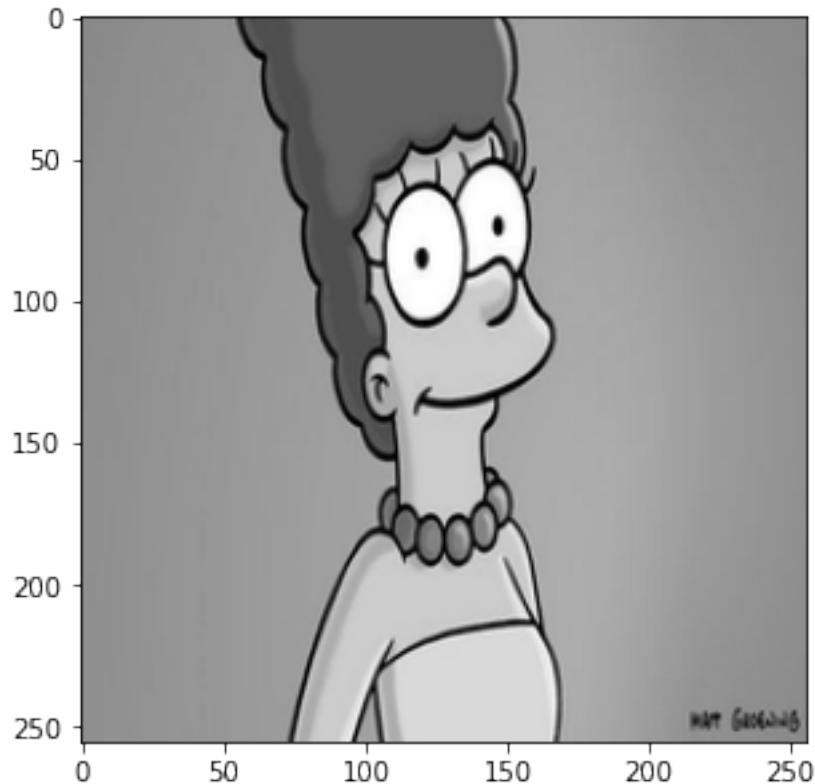
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



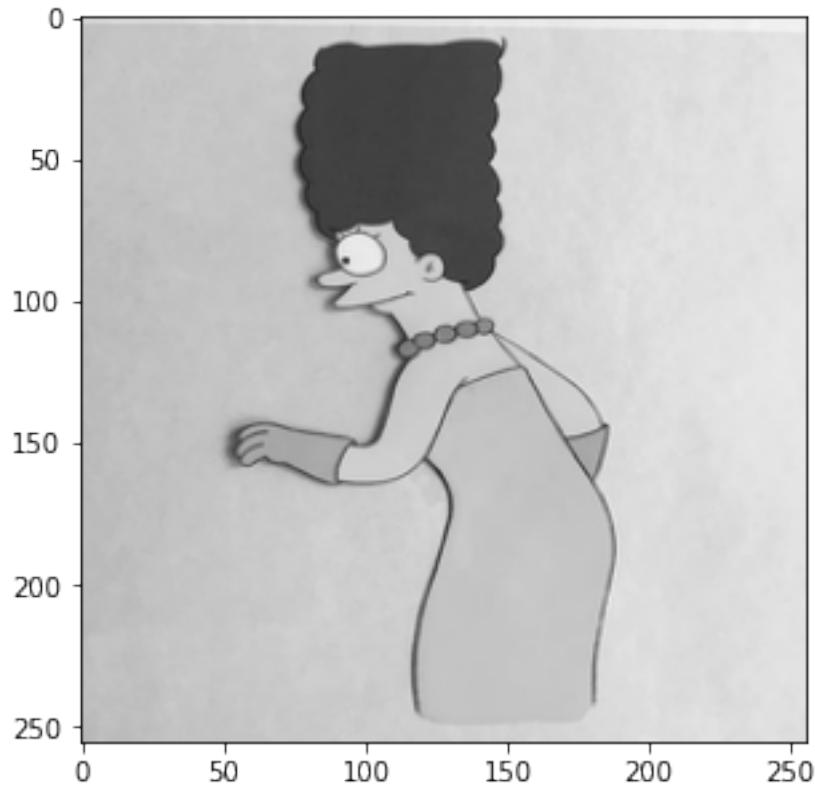
```
[[155.53941909 155.53941909 155.53941909 ... 155.53941909 155.53941909  
155.53941909]  
[155.53941909 155.53941909 155.53941909 ... 155.53941909 155.53941909  
155.53941909]  
[155.53941909 155.53941909 155.53941909 ... 155.53941909 155.53941909  
155.53941909]  
...  
[183.04979253 183.04979253 183.04979253 ... 223.25726141 224.3153527  
226.43153527]  
[183.04979253 183.04979253 183.04979253 ... 226.43153527 226.43153527  
226.43153527]  
[183.04979253 183.04979253 183.04979253 ... 226.43153527 226.43153527  
226.43153527]]
```



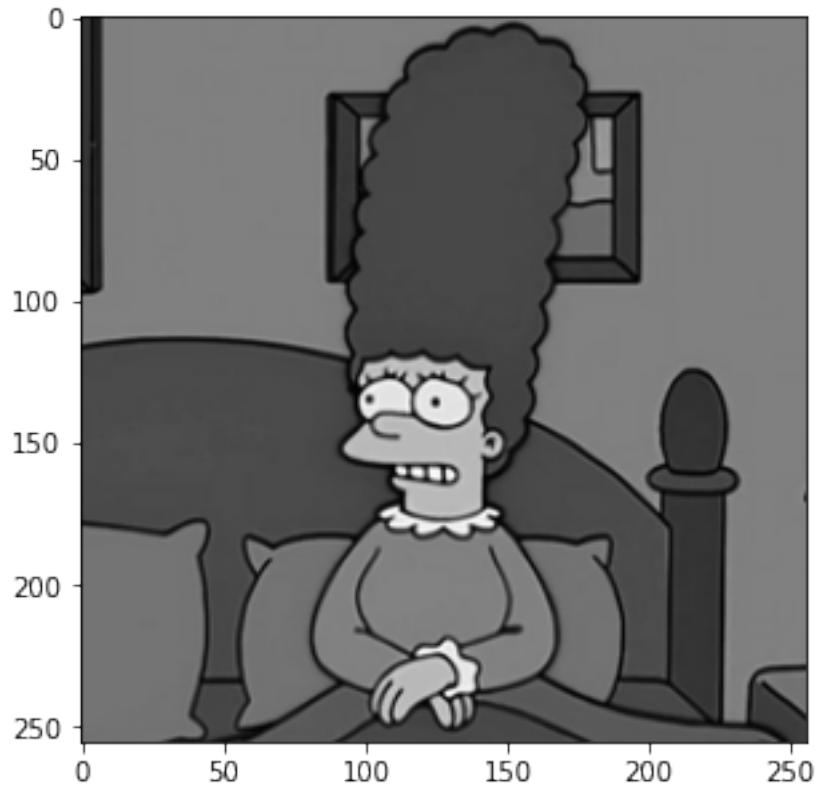
```
[[137. 137. 137. ... 136. 137. 137.]  
 [137. 137. 137. ... 136. 137. 137.]  
 [137. 137. 137. ... 136. 137. 137.]  
 ...  
 [143. 143. 143. ... 138. 139. 139.]  
 [143. 143. 143. ... 141. 139. 139.]  
 [143. 143. 143. ... 139. 139. 139.]]
```



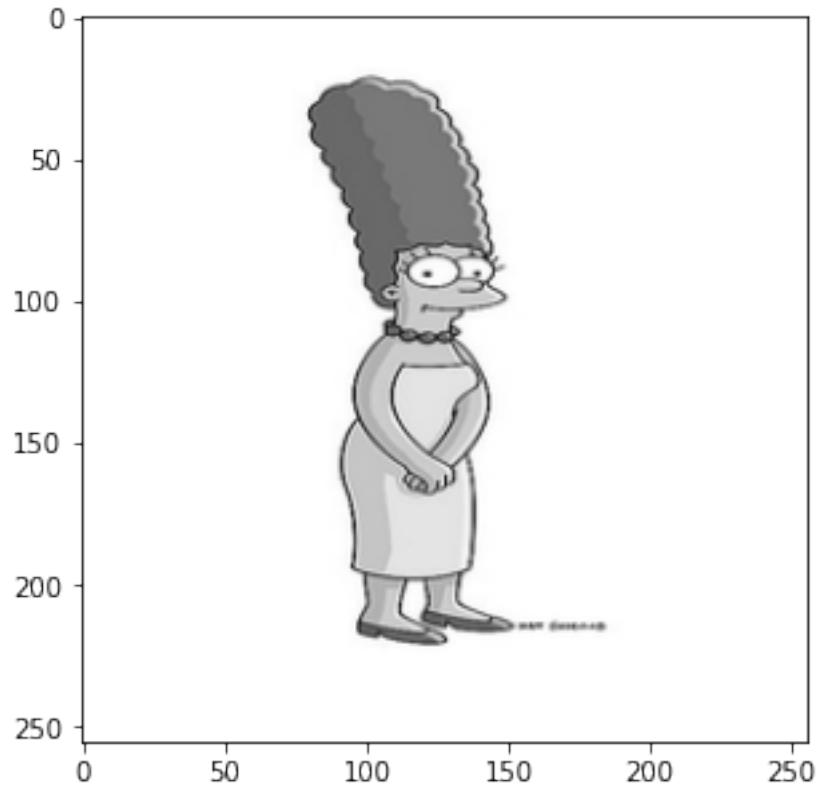
```
[[213.70445344 212.67206478 215.76923077 ... 239.51417004 239.51417004  
239.51417004]  
[215.76923077 213.70445344 215.76923077 ... 239.51417004 239.51417004  
239.51417004]  
[209.57489879 209.57489879 214.73684211 ... 240.5465587 239.51417004  
239.51417004]  
...  
[184.79757085 183.76518219 183.76518219 ... 222.99595142 221.96356275  
221.96356275]  
[184.79757085 183.76518219 183.76518219 ... 224.02834008 221.96356275  
221.96356275]  
[185.82995951 184.79757085 184.79757085 ... 222.99595142 220.93117409  
220.93117409]]
```



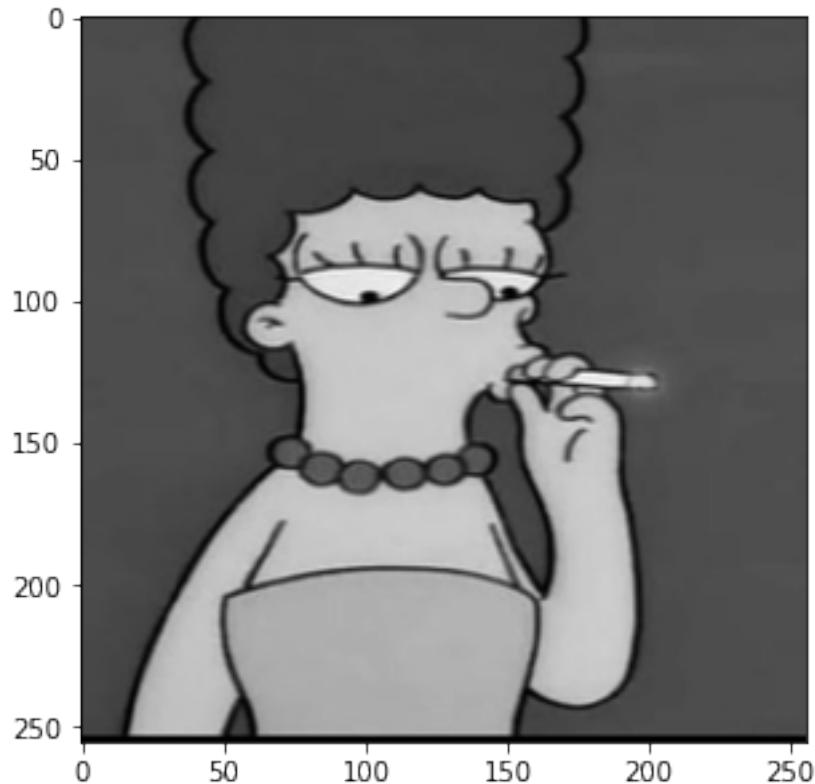
```
[[ 52.  52.  55. ... 128. 128. 127.]  
 [ 52.  52.  55. ... 128. 128. 127.]  
 [ 52.  52.  55. ... 128. 128. 127.]  
 ...  
 [115. 115. 114. ... 61.  62.  61.]  
 [113. 114. 113. ... 61.  62.  63.]  
 [113. 114. 113. ... 61.  61.  62.]]
```



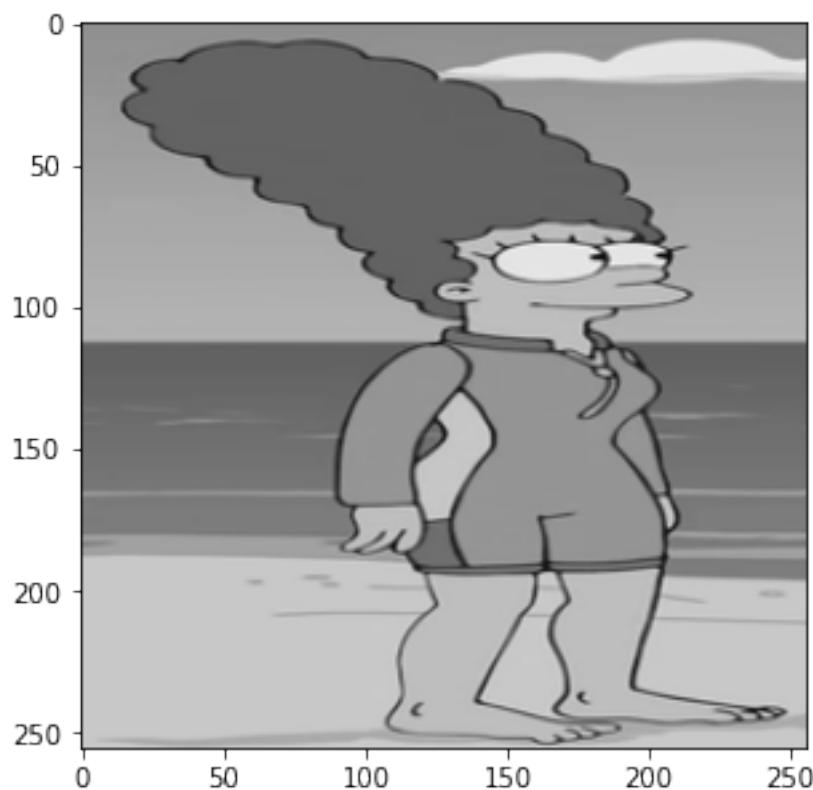
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



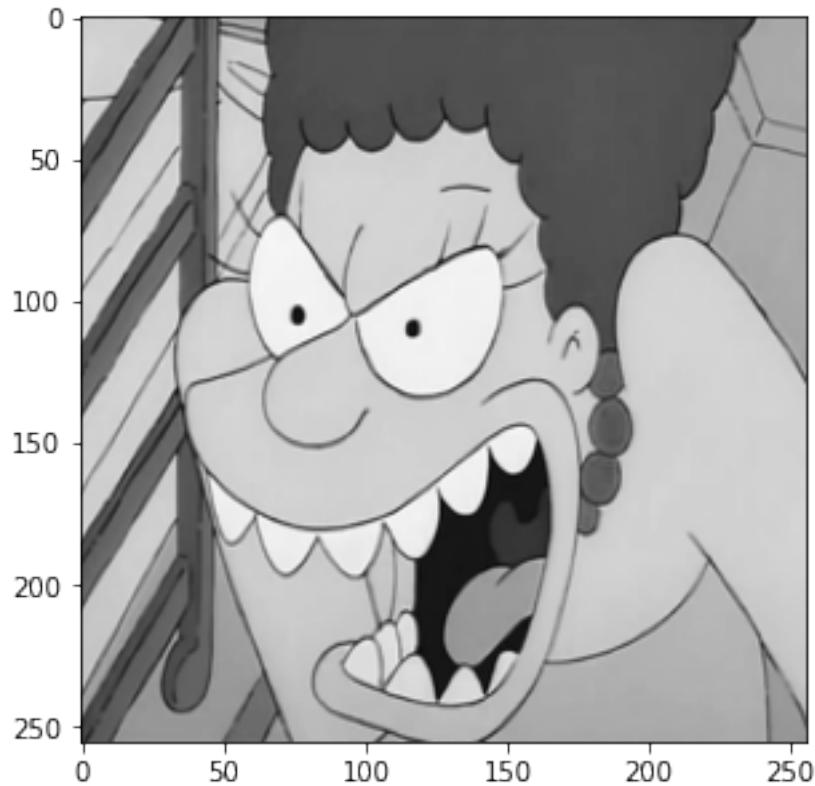
```
[[75.75313808 75.75313808 75.75313808 ... 75.75313808 75.75313808  
75.75313808]  
[76.82008368 76.82008368 75.75313808 ... 75.75313808 75.75313808  
75.75313808]  
[74.68619247 74.68619247 75.75313808 ... 75.75313808 75.75313808  
75.75313808]  
...  
[45.87866109 45.87866109 44.81171548 ... 51.21338912 52.28033473  
52.28033473]  
[ 0.          0.          0.          ... 0.          0.  
 0.          ]  
[ 0.          0.          0.          ... 0.          0.  
 0.          ]]
```



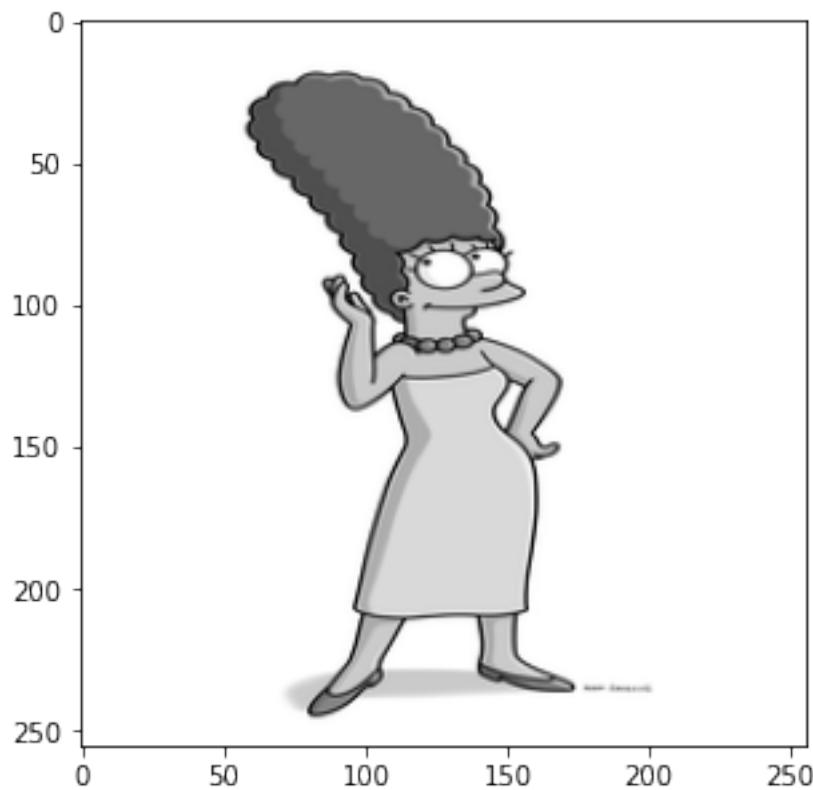
```
[[143.72727273 143.72727273 143.72727273 ... 143.72727273 143.72727273  
143.72727273]  
[143.72727273 143.72727273 143.72727273 ... 143.72727273 143.72727273  
143.72727273]  
[144.88636364 144.88636364 144.88636364 ... 144.88636364 144.88636364  
144.88636364]  
...  
[200.52272727 200.52272727 200.52272727 ... 200.52272727 200.52272727  
200.52272727]  
[200.52272727 200.52272727 200.52272727 ... 200.52272727 200.52272727  
200.52272727]  
[200.52272727 200.52272727 200.52272727 ... 200.52272727 200.52272727  
200.52272727]]
```



```
[[ 97. 178. 204. ... 169. 171. 172.]  
 [143. 204. 208. ... 171. 172. 173.]  
 [193. 207. 205. ... 168. 169. 169.]  
 ...  
 [ 68.  73.  52. ... 151. 152. 151.]  
 [ 70.  68.  36. ... 149. 150. 150.]  
 [ 75.  54.  41. ... 149. 149. 149.]]
```



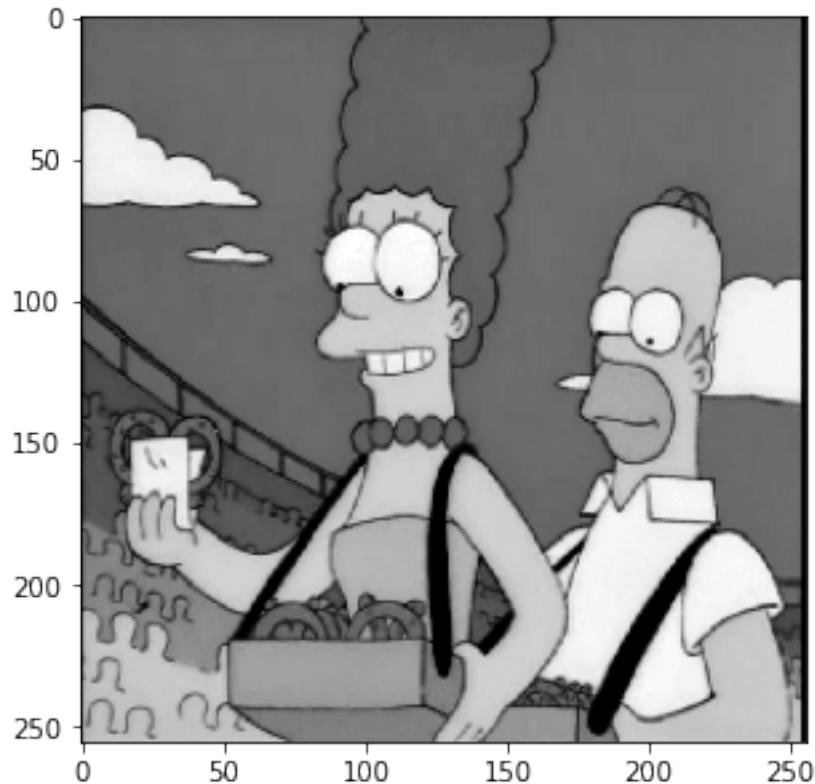
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



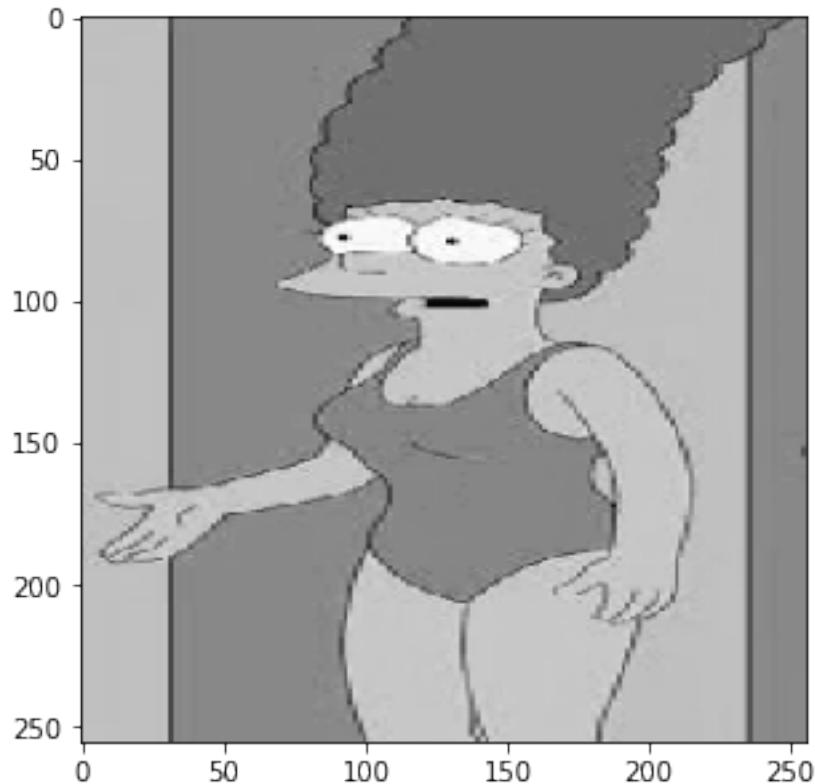
```
[[154. 154. 154. ... 151. 151. 151.]  
 [154. 154. 155. ... 151. 151. 151.]  
 [154. 154. 155. ... 151. 151. 151.]  
 ...  
 [160. 160. 160. ... 156. 156. 156.]  
 [160. 160. 159. ... 156. 156. 156.]  
 [160. 160. 159. ... 156. 156. 156.]]
```



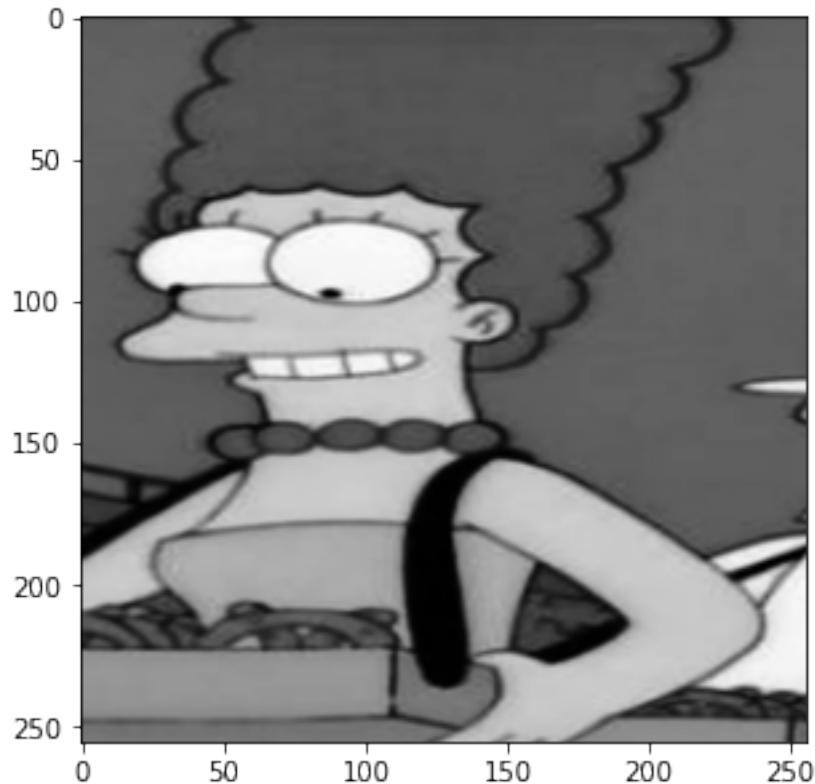
```
[[ 27.27906977  85.39534884  98.44186047 ... 107.93023256  14.23255814
  0.          ]
 [ 27.27906977  85.39534884  98.44186047 ... 107.93023256  14.23255814
  0.          ]
 [ 27.27906977  85.39534884  98.44186047 ... 109.11627907  14.23255814
  0.          ]
 ...
 [ 72.34883721 163.6744186 187.39534884 ... 199.25581395  17.79069767
  0.          ]
 [ 72.34883721 169.60465116 187.39534884 ... 199.25581395  17.79069767
  0.          ]
 [ 72.34883721 170.79069767 187.39534884 ... 199.25581395  17.79069767
  0.          ]]
```



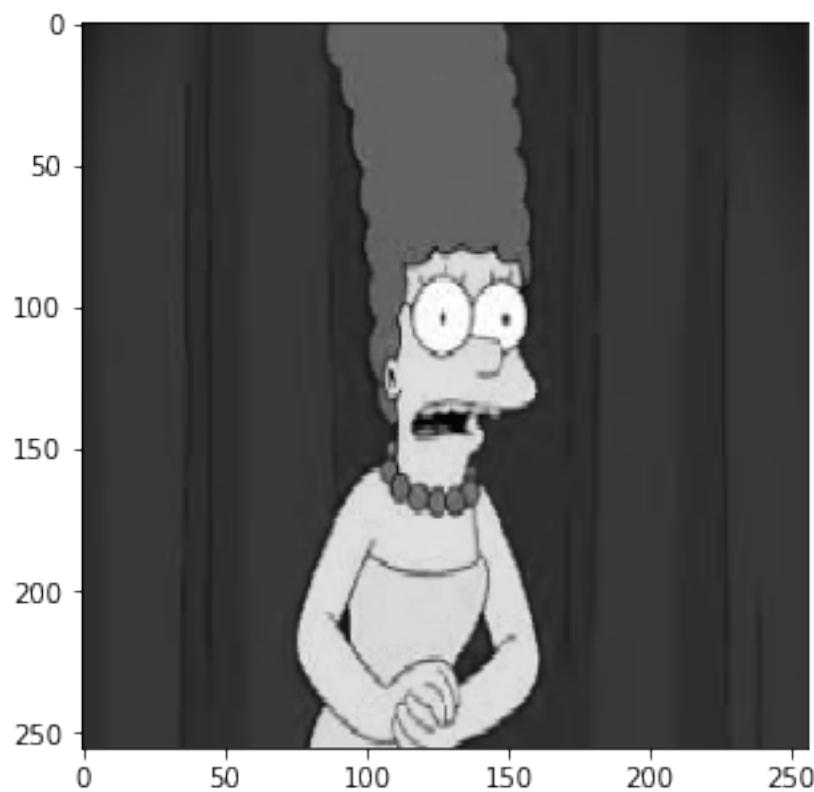
```
[[192.26190476 192.26190476 192.26190476 ... 139.64285714 123.45238095  
123.45238095]  
[192.26190476 192.26190476 192.26190476 ... 137.61904762 132.55952381  
132.55952381]  
[192.26190476 192.26190476 192.26190476 ... 137.61904762 145.71428571  
145.71428571]  
...  
[193.27380952 193.27380952 193.27380952 ... 137.61904762 137.61904762  
137.61904762]  
[193.27380952 193.27380952 193.27380952 ... 137.61904762 137.61904762  
137.61904762]  
[193.27380952 193.27380952 193.27380952 ... 137.61904762 137.61904762  
137.61904762]]
```



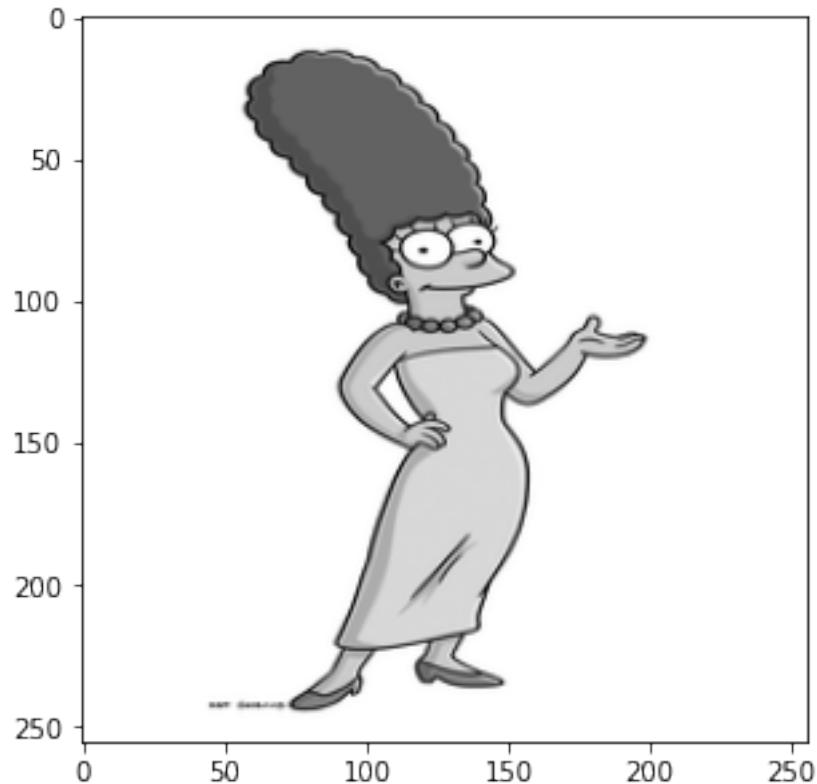
```
[[ 88.34645669  89.3503937   91.35826772 ...  92.36220472  92.36220472
  92.36220472]
 [ 88.34645669  89.3503937   90.35433071 ...  93.36614173  93.36614173
  93.36614173]
 [ 92.36220472  93.36614173  95.37401575 ...  95.37401575  95.37401575
  95.37401575]
 ...
 [[118.46456693 118.46456693 118.46456693 ... 106.41732283  99.38976378
  102.4015748 ]
 [119.46850394 119.46850394 119.46850394 ... 108.42519685 103.40551181
 107.42125984]
 [119.46850394 119.46850394 119.46850394 ... 108.42519685 103.40551181
 108.42519685]]
```



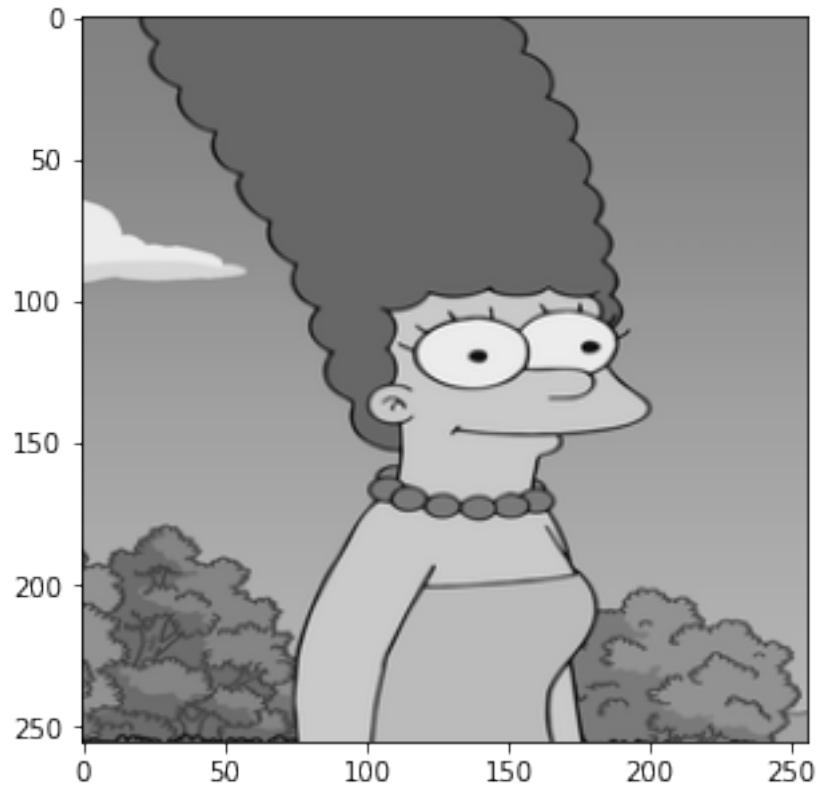
```
[[33.26086957 32.25296443 32.25296443 ... 30.23715415 30.23715415  
30.23715415]  
[33.26086957 33.26086957 33.26086957 ... 30.23715415 30.23715415  
30.23715415]  
[33.26086957 33.26086957 33.26086957 ... 30.23715415 30.23715415  
30.23715415]  
...  
[47.3715415 47.3715415 47.3715415 ... 56.44268775 56.44268775  
56.44268775]  
[47.3715415 47.3715415 47.3715415 ... 56.44268775 56.44268775  
56.44268775]  
[47.3715415 47.3715415 47.3715415 ... 56.44268775 56.44268775  
56.44268775]]
```



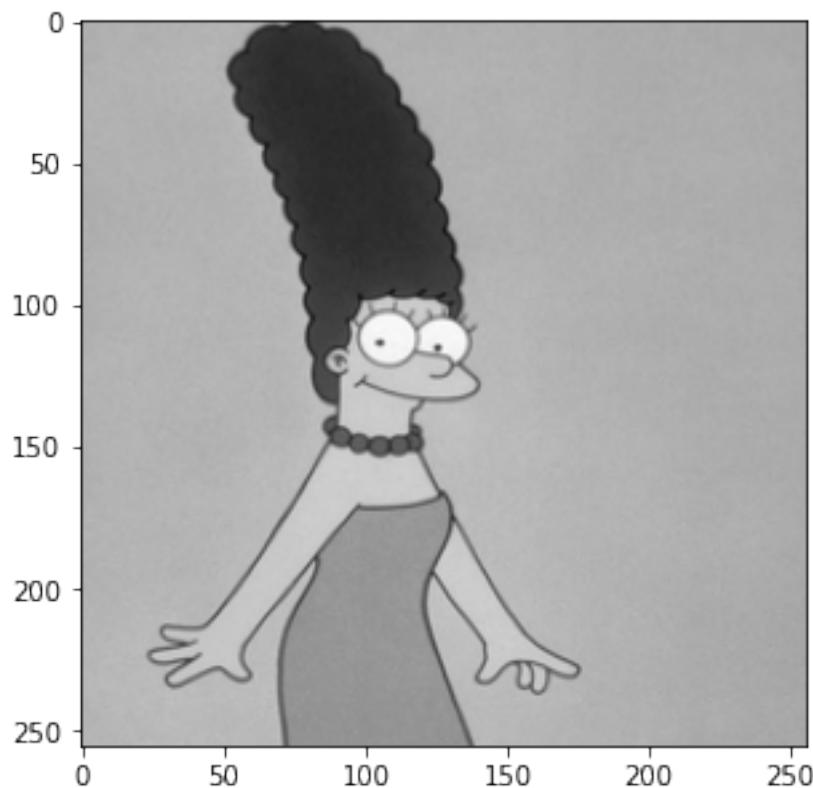
```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



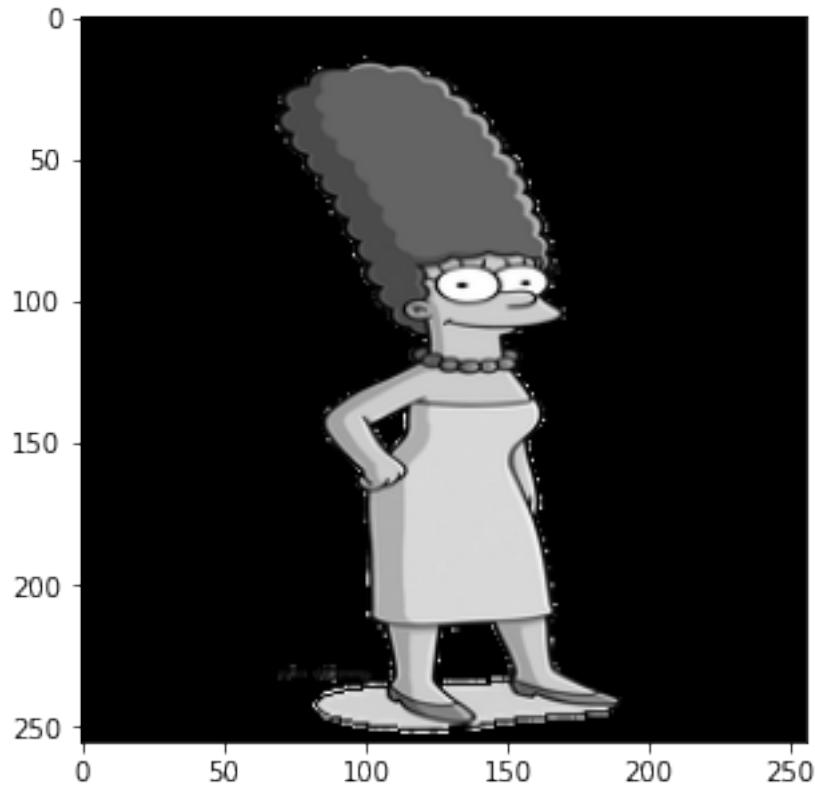
```
[[130.25974026 130.25974026 130.25974026 ... 130.25974026 130.25974026  
130.25974026]  
[130.25974026 130.25974026 130.25974026 ... 130.25974026 130.25974026  
130.25974026]  
[130.25974026 130.25974026 130.25974026 ... 130.25974026 130.25974026  
130.25974026]  
...  
[131.36363636 100.45454545 89.41558442 ... 162.27272727 162.27272727  
162.27272727]  
[ 90.51948052  81.68831169  82.79220779 ... 162.27272727 162.27272727  
162.27272727]  
[ 26.49350649  34.22077922  38.63636364 ... 161.16883117 162.27272727  
162.27272727]]
```



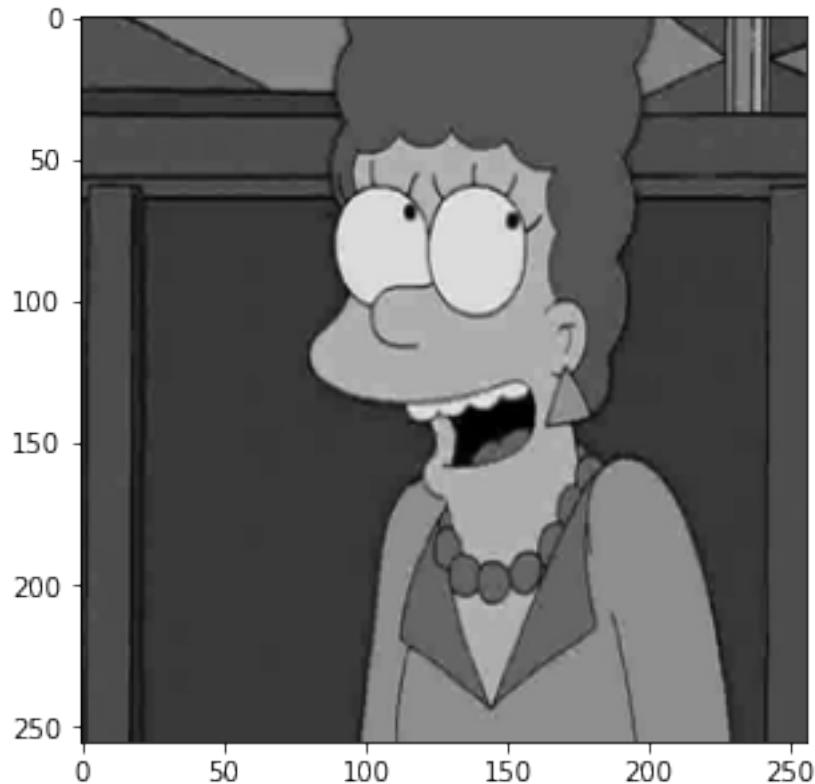
```
[[172.38 170.34 173.4 ... 169.32 169.32 168.3 ]
 [172.38 171.36 174.42 ... 169.32 168.3 170.34]
 [175.44 175.44 176.46 ... 170.34 169.32 170.34]
 ...
 [179.52 181.56 182.58 ... 171.36 169.32 165.24]
 [183.6 181.56 183.6 ... 169.32 169.32 165.24]
 [185.64 182.58 183.6 ... 169.32 168.3 168.3 ]]
```



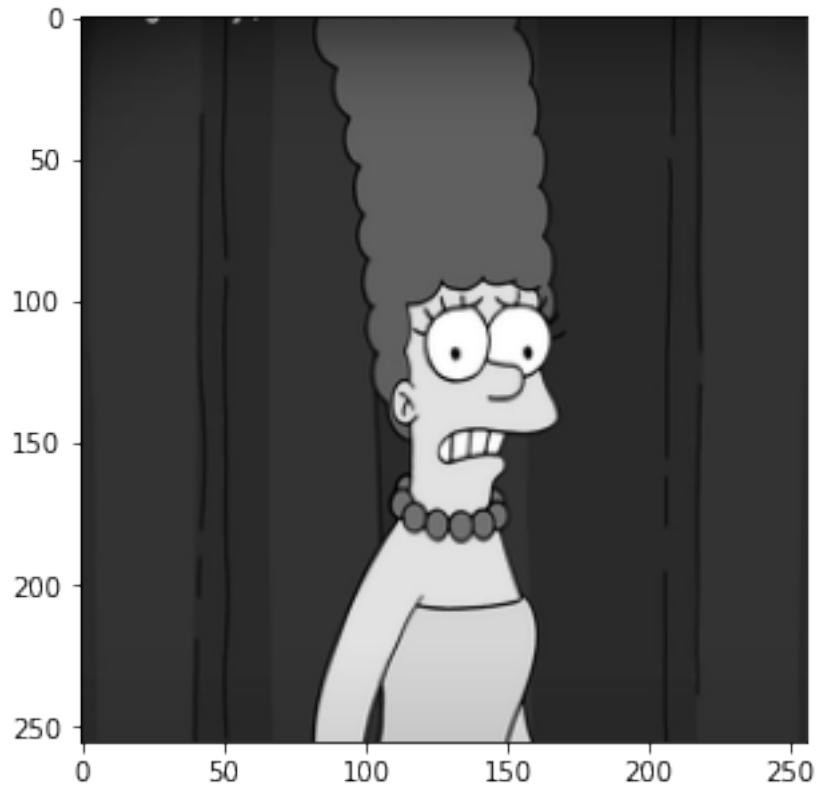
```
[[0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 0. 0.]]
```



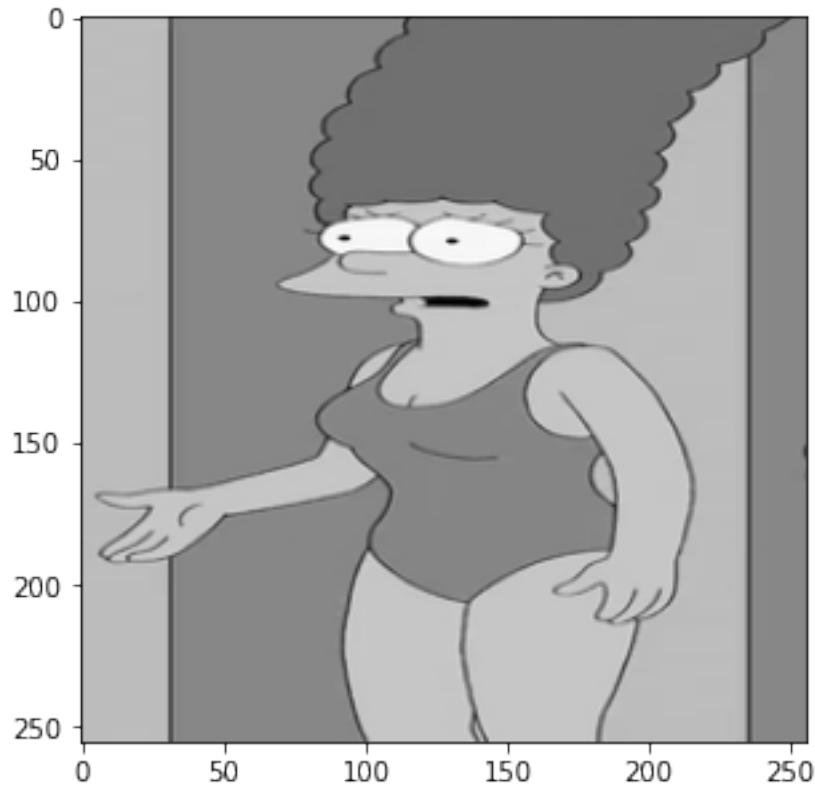
```
[[82.92682927 82.92682927 82.92682927 ... 75.67073171 75.67073171  
75.67073171]  
[82.92682927 82.92682927 82.92682927 ... 75.67073171 75.67073171  
75.67073171]  
[82.92682927 82.92682927 82.92682927 ... 75.67073171 75.67073171  
75.67073171]  
...  
[54.93902439 37.31707317 44.57317073 ... 76.70731707 76.70731707  
76.70731707]  
[54.93902439 37.31707317 44.57317073 ... 76.70731707 76.70731707  
76.70731707]  
[54.93902439 37.31707317 44.57317073 ... 76.70731707 76.70731707  
76.70731707]]
```



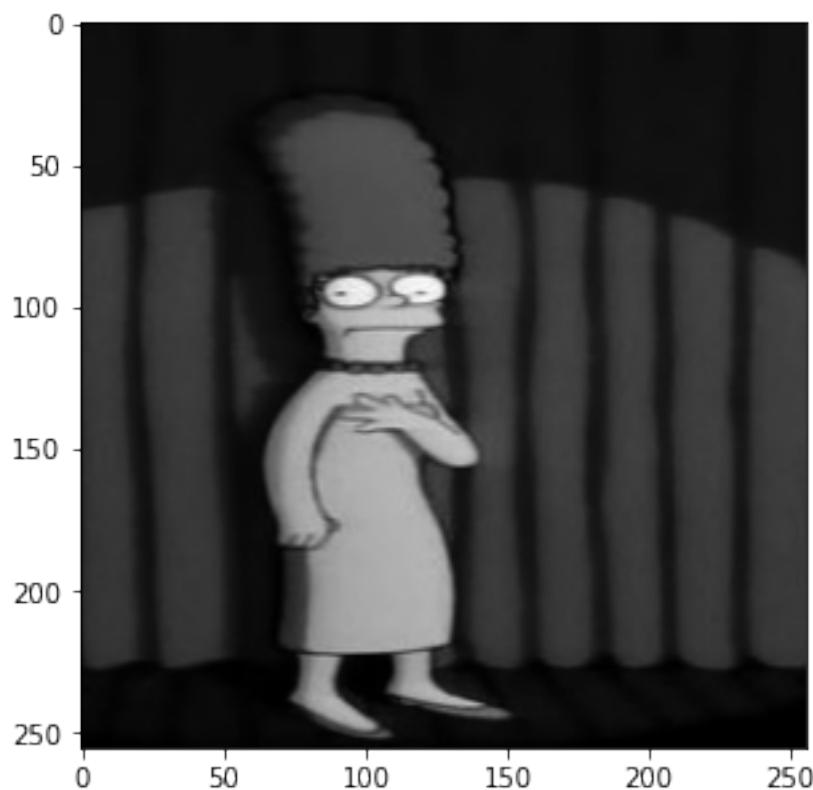
```
[[11. 11. 11. ... 19. 19. 18.]  
 [19. 19. 19. ... 32. 31. 31.]  
 [18. 19. 19. ... 31. 30. 30.]  
 ...  
 [40. 40. 40. ... 44. 39. 40.]  
 [40. 40. 40. ... 43. 39. 40.]  
 [40. 40. 40. ... 43. 40. 40.]]
```



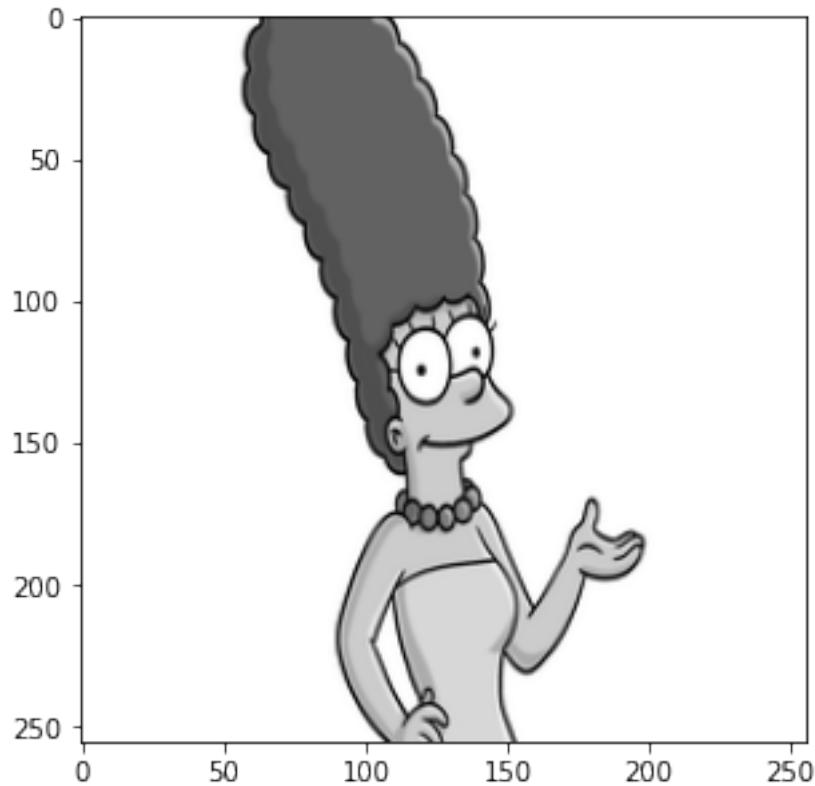
```
[[188. 188. 188. ... 134. 134. 135.]  
 [188. 188. 188. ... 135. 135. 135.]  
 [188. 188. 188. ... 136. 135. 135.]  
 ...  
 [188. 188. 188. ... 135. 135. 135.]  
 [188. 188. 188. ... 135. 135. 135.]  
 [188. 188. 188. ... 135. 135. 135.]]
```



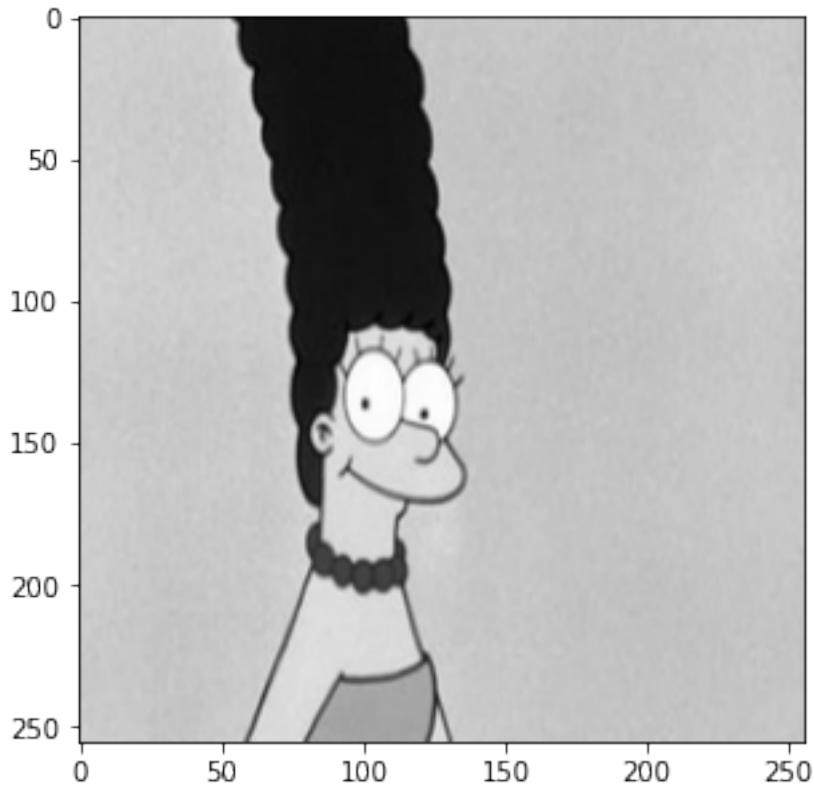
```
[[14. 15. 15. ... 16. 16. 16.]
 [14. 14. 14. ... 16. 16. 16.]
 [14. 14. 14. ... 16. 15. 15.]
 ...
 [ 2.  2.  2. ... 0.  0.  0.]
 [ 1.  1.  1. ... 0.  0.  0.]
 [ 0.  1.  1. ... 0.  0.  0.]]
```



```
[[255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 ...  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]  
 [255. 255. 255. ... 255. 255. 255.]]
```



```
[[205. 205. 205. ... 202. 201. 200.]  
 [204. 205. 206. ... 202. 200. 199.]  
 [204. 205. 206. ... 201. 200. 199.]  
 ...  
 [206. 206. 207. ... 198. 198. 195.]  
 [207. 206. 205. ... 198. 200. 198.]  
 [209. 206. 203. ... 198. 201. 201.]]
```



6 Development of initial CNN Topology

```
[3]: # fix random seed for reproducibility
seed = 1
np.random.seed(seed)
tf.random.set_seed(seed)

X = []
Y = []
for filename in glob.glob('normalizedBart/*.jpg'):
    im=Image.open(filename).convert('L')
    arr = np.array(im)
    X.append(arr)
    Y.append(1)  # Bart

for filename in glob.glob('normalizedHomer/*.jpg'):
    im=Image.open(filename).convert('L')
    arr = np.array(im)
    X.append(arr)
```

```

Y.append(2)  # Homer

for filename in glob.glob('normalizedLisa/*.jpg'):
    im=Image.open(filename).convert('L')
    arr = np.array(im)
    X.append(arr)
    Y.append(3)  # Lisa

for filename in glob.glob('normalizedMaggie/*.jpg'):
    im=Image.open(filename).convert('L')
    arr = np.array(im)
    X.append(arr)
    Y.append(4)  # Maggie

for filename in glob.glob('normalizedMarge/*.jpg'):
    im=Image.open(filename).convert('L')
    arr = np.array(im)
    X.append(arr)
    Y.append(5)  # Marge

# Convert to NP array
X = np.array(X)

# reshape to be [samples][channels][width][height]
X = X.reshape(X.shape[0], 1, 256, 256).astype('float32')

# Normalize the data
X = X /255

# Encode outputs
Y = np.array(Y)

# randomize the data set - numpy arrays
randomize = np.arange(len(X))
np.random.shuffle(randomize)
X = X[randomize]
Y = Y[randomize]

Y = tf.keras.utils.to_categorical(Y)
num_classes = Y.shape[1]

```

Investigation 1: Starting off with default sizes. Two convolutional layers by default. Changed input shape to match resolution of images.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", input_shape=(1, 256, 256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(128, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

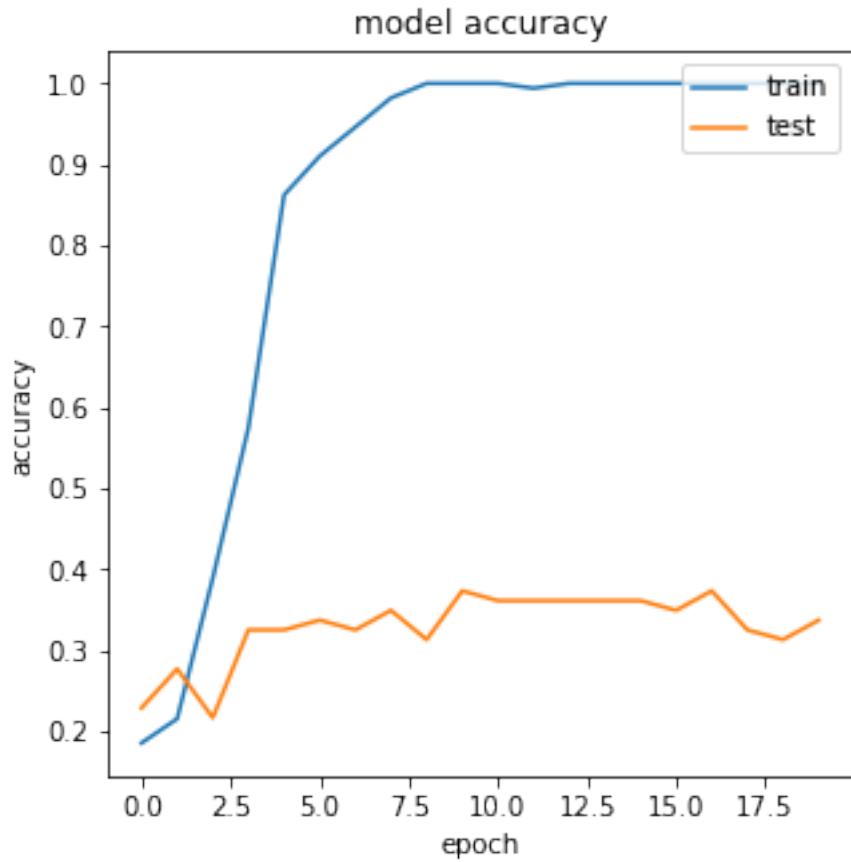
modelC.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50, verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()
```

Epoch 1/20
4/4 [=====] - 1s 191ms/step - loss: 9.4035 - acc: 0.1996 - val_loss: 5.6219 - val_acc: 0.2289
Epoch 2/20
4/4 [=====] - 1s 134ms/step - loss: 5.3995 - acc: 0.2276 - val_loss: 1.6844 - val_acc: 0.2771
Epoch 3/20
4/4 [=====] - 1s 138ms/step - loss: 1.6287 - acc: 0.3597 - val_loss: 1.7581 - val_acc: 0.2169
Epoch 4/20
4/4 [=====] - 1s 136ms/step - loss: 1.3321 - acc: 0.5053 - val_loss: 1.6702 - val_acc: 0.3253
Epoch 5/20
4/4 [=====] - 1s 136ms/step - loss: 0.7512 - acc: 0.8402 - val_loss: 2.0488 - val_acc: 0.3253
Epoch 6/20

```
4/4 [=====] - 1s 137ms/step - loss: 0.3857 - acc: 0.9021 - val_loss: 1.8286 - val_acc: 0.3373
Epoch 7/20
4/4 [=====] - 1s 140ms/step - loss: 0.1945 - acc: 0.9424 - val_loss: 2.1625 - val_acc: 0.3253
Epoch 8/20
4/4 [=====] - 1s 138ms/step - loss: 0.0812 - acc: 0.9848 - val_loss: 2.5488 - val_acc: 0.3494
Epoch 9/20
4/4 [=====] - 1s 140ms/step - loss: 0.0171 - acc: 1.0000 - val_loss: 3.2640 - val_acc: 0.3133
Epoch 10/20
4/4 [=====] - 1s 140ms/step - loss: 0.0119 - acc: 1.0000 - val_loss: 3.2628 - val_acc: 0.3735
Epoch 11/20
4/4 [=====] - 1s 139ms/step - loss: 0.0037 - acc: 1.0000 - val_loss: 3.4149 - val_acc: 0.3614
Epoch 12/20
4/4 [=====] - 1s 139ms/step - loss: 0.0192 - acc: 0.9943 - val_loss: 3.6721 - val_acc: 0.3614
Epoch 13/20
4/4 [=====] - 1s 138ms/step - loss: 0.0078 - acc: 1.0000 - val_loss: 3.6537 - val_acc: 0.3614
Epoch 14/20
4/4 [=====] - 1s 140ms/step - loss: 0.0040 - acc: 1.0000 - val_loss: 2.9220 - val_acc: 0.3614
Epoch 15/20
4/4 [=====] - 1s 139ms/step - loss: 0.0074 - acc: 1.0000 - val_loss: 3.2583 - val_acc: 0.3614
Epoch 16/20
4/4 [=====] - 1s 139ms/step - loss: 0.0110 - acc: 1.0000 - val_loss: 3.2724 - val_acc: 0.3494
Epoch 17/20
4/4 [=====] - 1s 138ms/step - loss: 0.0034 - acc: 1.0000 - val_loss: 3.2517 - val_acc: 0.3735
Epoch 18/20
4/4 [=====] - 1s 138ms/step - loss: 0.0013 - acc: 1.0000 - val_loss: 3.7352 - val_acc: 0.3253
Epoch 19/20
4/4 [=====] - 1s 137ms/step - loss: 7.2620e-04 - acc: 1.0000 - val_loss: 4.0344 - val_acc: 0.3133
Epoch 20/20
4/4 [=====] - 1s 134ms/step - loss: 4.0388e-04 - acc: 1.0000 - val_loss: 4.1521 - val_acc: 0.3373
```



Investigation 2: Changed the amount of nodes to 16 and 32 in the first convolutional layers. Dense layer is 64 as well. A 9% increase in accuracy.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(16, (3, 3), strides=1, padding="valid", input_shape=(1, 256, 256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(64, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))
```

```

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

                metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50,  

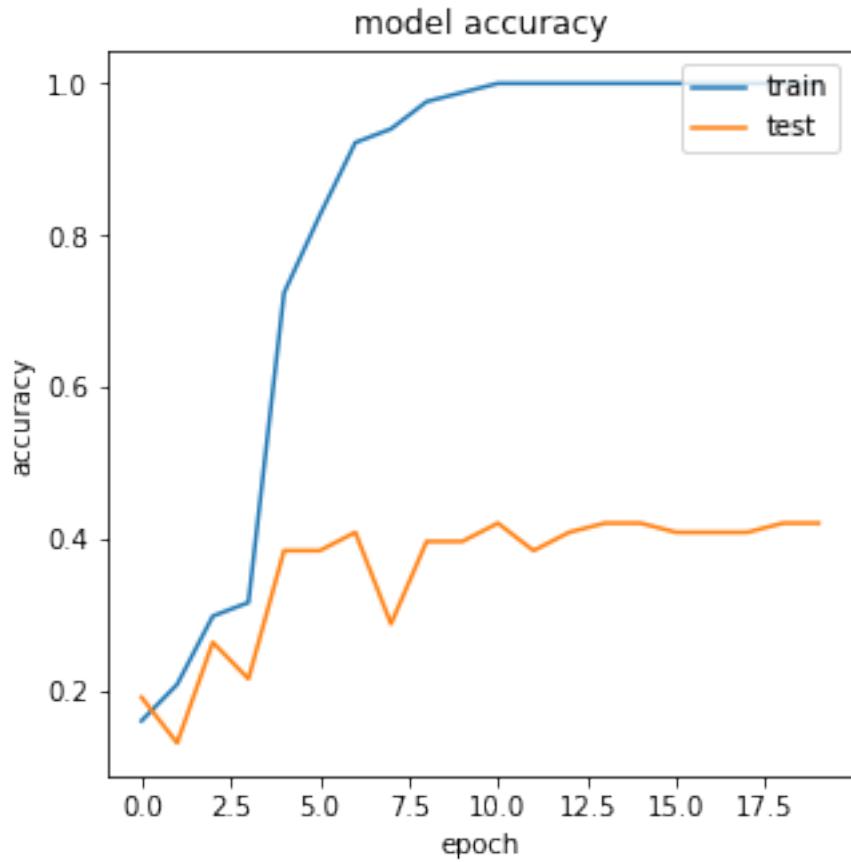
                      verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

Epoch 1/20
4/4 [=====] - 2s 222ms/step - loss: 4.0259 - acc:
0.1680 - val_loss: 7.7637 - val_acc: 0.1928
Epoch 2/20
4/4 [=====] - 0s 74ms/step - loss: 6.5110 - acc: 0.2125
- val_loss: 2.5053 - val_acc: 0.1325
Epoch 3/20
4/4 [=====] - 0s 75ms/step - loss: 2.1388 - acc: 0.2978
- val_loss: 1.6852 - val_acc: 0.2651
Epoch 4/20
4/4 [=====] - 0s 77ms/step - loss: 1.4589 - acc: 0.3003
- val_loss: 1.6034 - val_acc: 0.2169
Epoch 5/20
4/4 [=====] - 0s 77ms/step - loss: 1.1182 - acc: 0.7145
- val_loss: 1.5998 - val_acc: 0.3855
Epoch 6/20
4/4 [=====] - 0s 74ms/step - loss: 0.7579 - acc: 0.8059
- val_loss: 1.5343 - val_acc: 0.3855
Epoch 7/20
4/4 [=====] - 0s 77ms/step - loss: 0.4503 - acc: 0.9069
- val_loss: 1.6748 - val_acc: 0.4096
Epoch 8/20
4/4 [=====] - 0s 77ms/step - loss: 0.2719 - acc: 0.9407
- val_loss: 1.8937 - val_acc: 0.2892
Epoch 9/20
4/4 [=====] - 0s 78ms/step - loss: 0.1740 - acc: 0.9804
- val_loss: 1.8488 - val_acc: 0.3976
Epoch 10/20
4/4 [=====] - 0s 78ms/step - loss: 0.1051 - acc: 0.9845
- val_loss: 1.8951 - val_acc: 0.3976
Epoch 11/20

```
4/4 [=====] - 0s 78ms/step - loss: 0.0554 - acc: 1.0000
- val_loss: 1.9065 - val_acc: 0.4217
Epoch 12/20
4/4 [=====] - 0s 80ms/step - loss: 0.0322 - acc: 1.0000
- val_loss: 2.0736 - val_acc: 0.3855
Epoch 13/20
4/4 [=====] - 0s 79ms/step - loss: 0.0162 - acc: 1.0000
- val_loss: 2.2603 - val_acc: 0.4096
Epoch 14/20
4/4 [=====] - 0s 80ms/step - loss: 0.0097 - acc: 1.0000
- val_loss: 2.4381 - val_acc: 0.4217
Epoch 15/20
4/4 [=====] - 0s 77ms/step - loss: 0.0063 - acc: 1.0000
- val_loss: 2.5196 - val_acc: 0.4217
Epoch 16/20
4/4 [=====] - 0s 78ms/step - loss: 0.0044 - acc: 1.0000
- val_loss: 2.5102 - val_acc: 0.4096
Epoch 17/20
4/4 [=====] - 0s 76ms/step - loss: 0.0034 - acc: 1.0000
- val_loss: 2.5679 - val_acc: 0.4096
Epoch 18/20
4/4 [=====] - 0s 78ms/step - loss: 0.0027 - acc: 1.0000
- val_loss: 2.6495 - val_acc: 0.4096
Epoch 19/20
4/4 [=====] - 0s 76ms/step - loss: 0.0026 - acc: 1.0000
- val_loss: 2.6936 - val_acc: 0.4217
Epoch 20/20
4/4 [=====] - 0s 75ms/step - loss: 0.0016 - acc: 1.0000
- val_loss: 2.7480 - val_acc: 0.4217
```



Investigation 3: Decreased nodes to 8 and 16 respectively, as well as the dense layer having 32 nodes. A decrease of 4% test accuracy.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", input_shape=(1, 256, ↴256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴data_format='channels_first'))

modelC.add(Conv2D(16, (3, 3), strides=1, padding="valid", activation='relu', ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(32, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))
```

```

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

                metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50,  

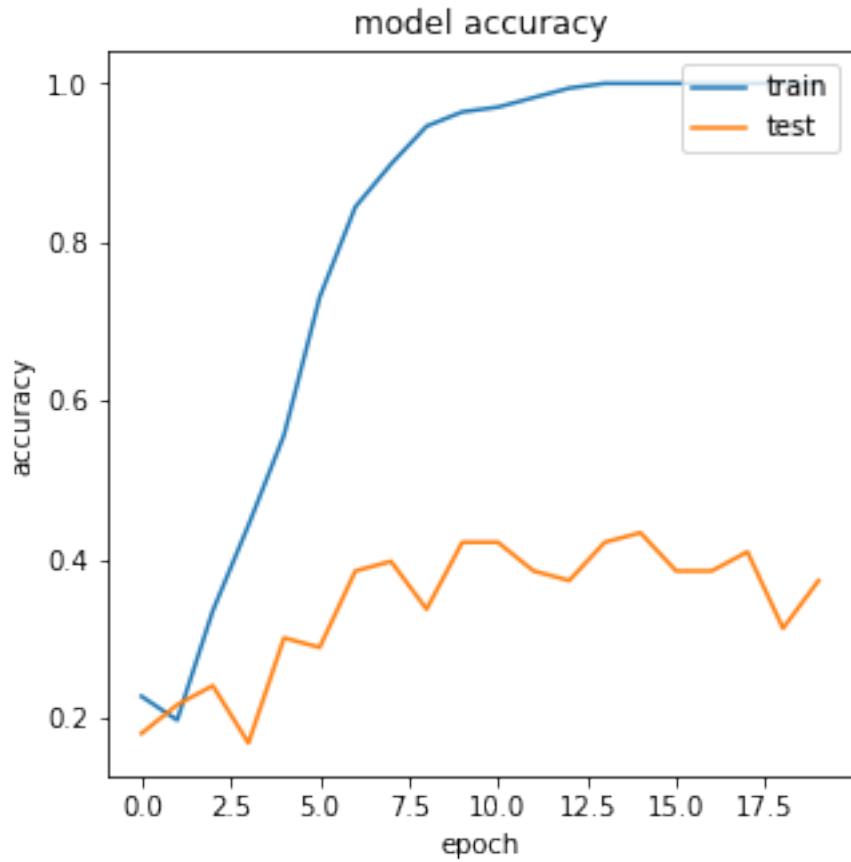
                      verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

Epoch 1/20
4/4 [=====] - 1s 100ms/step - loss: 2.1669 - acc:
0.2170 - val_loss: 1.7091 - val_acc: 0.1807
Epoch 2/20
4/4 [=====] - 0s 50ms/step - loss: 1.6985 - acc: 0.2004
- val_loss: 1.6892 - val_acc: 0.2169
Epoch 3/20
4/4 [=====] - 0s 54ms/step - loss: 1.5251 - acc: 0.3628
- val_loss: 1.6685 - val_acc: 0.2410
Epoch 4/20
4/4 [=====] - 0s 52ms/step - loss: 1.4176 - acc: 0.4346
- val_loss: 1.6311 - val_acc: 0.1687
Epoch 5/20
4/4 [=====] - 0s 51ms/step - loss: 1.2375 - acc: 0.5154
- val_loss: 1.6801 - val_acc: 0.3012
Epoch 6/20
4/4 [=====] - 0s 51ms/step - loss: 1.0393 - acc: 0.7189
- val_loss: 1.5647 - val_acc: 0.2892
Epoch 7/20
4/4 [=====] - 0s 48ms/step - loss: 0.8250 - acc: 0.8471
- val_loss: 1.5394 - val_acc: 0.3855
Epoch 8/20
4/4 [=====] - 0s 49ms/step - loss: 0.6573 - acc: 0.9033
- val_loss: 1.6267 - val_acc: 0.3976
Epoch 9/20
4/4 [=====] - 0s 50ms/step - loss: 0.4596 - acc: 0.9458
- val_loss: 1.5725 - val_acc: 0.3373
Epoch 10/20
4/4 [=====] - 0s 49ms/step - loss: 0.3466 - acc: 0.9516
- val_loss: 1.5956 - val_acc: 0.4217
Epoch 11/20

```
4/4 [=====] - 0s 52ms/step - loss: 0.2577 - acc: 0.9734
- val_loss: 1.7201 - val_acc: 0.4217
Epoch 12/20
4/4 [=====] - 0s 49ms/step - loss: 0.1893 - acc: 0.9808
- val_loss: 1.7019 - val_acc: 0.3855
Epoch 13/20
4/4 [=====] - 0s 48ms/step - loss: 0.1367 - acc: 0.9963
- val_loss: 1.8967 - val_acc: 0.3735
Epoch 14/20
4/4 [=====] - 0s 49ms/step - loss: 0.0954 - acc: 1.0000
- val_loss: 2.0148 - val_acc: 0.4217
Epoch 15/20
4/4 [=====] - 0s 51ms/step - loss: 0.0673 - acc: 1.0000
- val_loss: 2.0394 - val_acc: 0.4337
Epoch 16/20
4/4 [=====] - 0s 49ms/step - loss: 0.0419 - acc: 1.0000
- val_loss: 2.0608 - val_acc: 0.3855
Epoch 17/20
4/4 [=====] - 0s 50ms/step - loss: 0.0327 - acc: 1.0000
- val_loss: 2.1699 - val_acc: 0.3855
Epoch 18/20
4/4 [=====] - 0s 49ms/step - loss: 0.0219 - acc: 1.0000
- val_loss: 2.3655 - val_acc: 0.4096
Epoch 19/20
4/4 [=====] - 0s 50ms/step - loss: 0.0193 - acc: 1.0000
- val_loss: 2.6067 - val_acc: 0.3133
Epoch 20/20
4/4 [=====] - 0s 49ms/step - loss: 0.0127 - acc: 1.0000
- val_loss: 2.7021 - val_acc: 0.3735
```



Investigation 4: A vast increase in nodes. A vast decrease to accuracy.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", input_shape=(1, 256,
    ↴256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",
    ↴data_format='channels_first'))

modelC.add(Conv2D(128, (3, 3), strides=1, padding="valid", activation='relu',
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",
    ↴data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(256, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam',
    ↴metrics=['acc'])
```

```

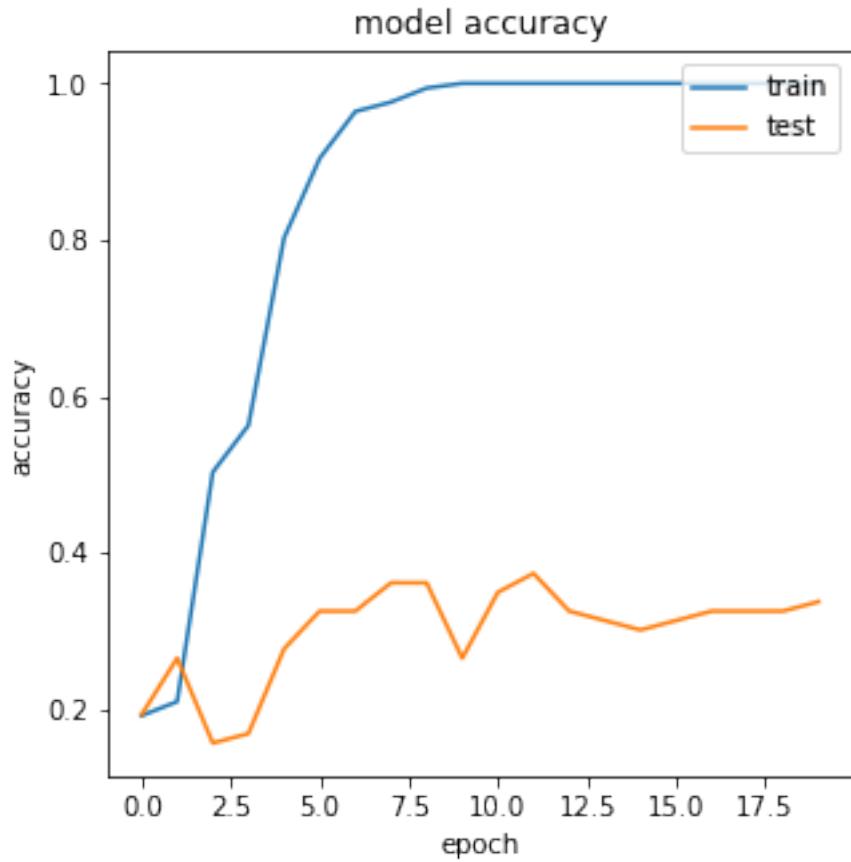
# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50, verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

Epoch 1/20
4/4 [=====] - 5s 843ms/step - loss: 11.3440 - acc: 0.2053 - val_loss: 11.1170 - val_acc: 0.1928
Epoch 2/20
4/4 [=====] - 1s 283ms/step - loss: 8.9714 - acc: 0.2178 - val_loss: 1.7296 - val_acc: 0.2651
Epoch 3/20
4/4 [=====] - 1s 285ms/step - loss: 1.5386 - acc: 0.5592 - val_loss: 1.9099 - val_acc: 0.1566
Epoch 4/20
4/4 [=====] - 1s 282ms/step - loss: 1.2310 - acc: 0.4858 - val_loss: 2.0199 - val_acc: 0.1687
Epoch 5/20
4/4 [=====] - 1s 281ms/step - loss: 0.7709 - acc: 0.7323 - val_loss: 1.9223 - val_acc: 0.2771
Epoch 6/20
4/4 [=====] - 1s 282ms/step - loss: 0.2988 - acc: 0.9337 - val_loss: 2.6637 - val_acc: 0.3253
Epoch 7/20
4/4 [=====] - 1s 280ms/step - loss: 0.1679 - acc: 0.9496 - val_loss: 3.0513 - val_acc: 0.3253
Epoch 8/20
4/4 [=====] - 1s 278ms/step - loss: 0.0991 - acc: 0.9751 - val_loss: 2.4566 - val_acc: 0.3614
Epoch 9/20
4/4 [=====] - 1s 283ms/step - loss: 0.0705 - acc: 0.9963 - val_loss: 2.9849 - val_acc: 0.3614
Epoch 10/20
4/4 [=====] - 1s 284ms/step - loss: 0.0271 - acc: 1.0000 - val_loss: 3.7372 - val_acc: 0.2651
Epoch 11/20
4/4 [=====] - 1s 279ms/step - loss: 0.0086 - acc: 1.0000 - val_loss: 4.7663 - val_acc: 0.3494
Epoch 12/20

```
4/4 [=====] - 1s 282ms/step - loss: 0.0058 - acc: 1.0000 - val_loss: 4.4361 - val_acc: 0.3735
Epoch 13/20
4/4 [=====] - 1s 281ms/step - loss: 0.0011 - acc: 1.0000 - val_loss: 4.1854 - val_acc: 0.3253
Epoch 14/20
4/4 [=====] - 1s 282ms/step - loss: 0.0011 - acc: 1.0000 - val_loss: 4.2025 - val_acc: 0.3133
Epoch 15/20
4/4 [=====] - 1s 282ms/step - loss: 5.8341e-04 - acc: 1.0000 - val_loss: 4.2830 - val_acc: 0.3012
Epoch 16/20
4/4 [=====] - 1s 282ms/step - loss: 4.9809e-04 - acc: 1.0000 - val_loss: 4.3661 - val_acc: 0.3133
Epoch 17/20
4/4 [=====] - 1s 287ms/step - loss: 3.1032e-04 - acc: 1.0000 - val_loss: 4.4410 - val_acc: 0.3253
Epoch 18/20
4/4 [=====] - 1s 280ms/step - loss: 1.7431e-04 - acc: 1.0000 - val_loss: 4.5180 - val_acc: 0.3253
Epoch 19/20
4/4 [=====] - 1s 280ms/step - loss: 1.4159e-04 - acc: 1.0000 - val_loss: 4.5961 - val_acc: 0.3253
Epoch 20/20
4/4 [=====] - 1s 281ms/step - loss: 5.7970e-05 - acc: 1.0000 - val_loss: 4.6719 - val_acc: 0.3373
```



Investigation 5: Building off our best investigation so far, we add another layer as well as increase the nodes in the dense layer. but it doesn't improve our accuracy.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(16, (3, 3), strides=1, padding="valid", input_shape=(1, 256, ↴256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴data_format='channels_first'))

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", input_shape=(1, 256, ↴256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴data_format='channels_first'))

modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", activation='relu', ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴data_format='channels_first'))
```

```

modelC.add(Flatten())
modelC.add(Dense(128, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

    ↪metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50,  

    ↪verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

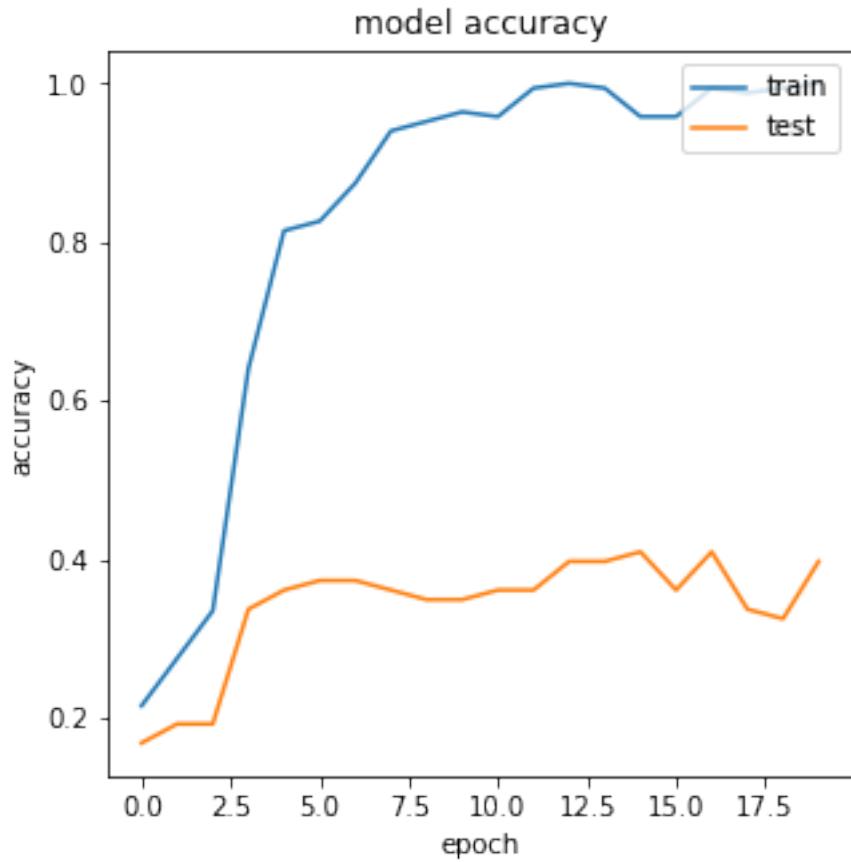
```

```

Epoch 1/20
4/4 [=====] - 1s 171ms/step - loss: 1.8863 - acc: 0.2009 - val_loss: 1.7045 - val_acc: 0.1687
Epoch 2/20
4/4 [=====] - 0s 85ms/step - loss: 1.6335 - acc: 0.2535 - val_loss: 1.6477 - val_acc: 0.1928
Epoch 3/20
4/4 [=====] - 0s 85ms/step - loss: 1.4772 - acc: 0.3468 - val_loss: 1.5750 - val_acc: 0.1928
Epoch 4/20
4/4 [=====] - 0s 83ms/step - loss: 1.2465 - acc: 0.5883 - val_loss: 1.5082 - val_acc: 0.3373
Epoch 5/20
4/4 [=====] - 0s 83ms/step - loss: 0.8650 - acc: 0.8197 - val_loss: 1.7994 - val_acc: 0.3614
Epoch 6/20
4/4 [=====] - 0s 85ms/step - loss: 0.5699 - acc: 0.8265 - val_loss: 2.2292 - val_acc: 0.3735
Epoch 7/20
4/4 [=====] - 0s 84ms/step - loss: 0.4337 - acc: 0.8577 - val_loss: 2.5426 - val_acc: 0.3735
Epoch 8/20
4/4 [=====] - 0s 84ms/step - loss: 0.2225 - acc: 0.9367 - val_loss: 2.5965 - val_acc: 0.3614
Epoch 9/20
4/4 [=====] - 0s 83ms/step - loss: 0.1900 - acc: 0.9435 - val_loss: 3.0521 - val_acc: 0.3494

```

```
Epoch 10/20
4/4 [=====] - 0s 84ms/step - loss: 0.1186 - acc: 0.9636
- val_loss: 3.3983 - val_acc: 0.3494
Epoch 11/20
4/4 [=====] - 0s 83ms/step - loss: 0.1132 - acc: 0.9599
- val_loss: 3.1720 - val_acc: 0.3614
Epoch 12/20
4/4 [=====] - 0s 84ms/step - loss: 0.0389 - acc: 0.9943
- val_loss: 3.9200 - val_acc: 0.3614
Epoch 13/20
4/4 [=====] - 0s 83ms/step - loss: 0.0325 - acc: 1.0000
- val_loss: 4.1806 - val_acc: 0.3976
Epoch 14/20
4/4 [=====] - 0s 83ms/step - loss: 0.0210 - acc: 0.9976
- val_loss: 3.8907 - val_acc: 0.3976
Epoch 15/20
4/4 [=====] - 0s 82ms/step - loss: 0.1522 - acc: 0.9739
- val_loss: 3.9977 - val_acc: 0.4096
Epoch 16/20
4/4 [=====] - 0s 85ms/step - loss: 0.0884 - acc: 0.9692
- val_loss: 3.7131 - val_acc: 0.3614
Epoch 17/20
4/4 [=====] - 0s 82ms/step - loss: 0.0478 - acc: 0.9976
- val_loss: 2.8865 - val_acc: 0.4096
Epoch 18/20
4/4 [=====] - 0s 84ms/step - loss: 0.0393 - acc: 0.9919
- val_loss: 3.9719 - val_acc: 0.3373
Epoch 19/20
4/4 [=====] - 0s 84ms/step - loss: 0.0378 - acc: 0.9976
- val_loss: 4.8273 - val_acc: 0.3253
Epoch 20/20
4/4 [=====] - 0s 81ms/step - loss: 0.0086 - acc: 1.0000
- val_loss: 4.5242 - val_acc: 0.3976
```



Investigation 6: Decrease to nodes and removal of extra layer.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(20, (3, 3), strides=1, padding="valid", input_shape=(1, 256,
    ↪256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",
    ↪data_format='channels_first'))

modelC.add(Conv2D(30, (3, 3), strides=1, padding="valid", activation='relu',
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",
    ↪data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(60, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam',
    ↪metrics=['acc'])
```

```

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50, verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

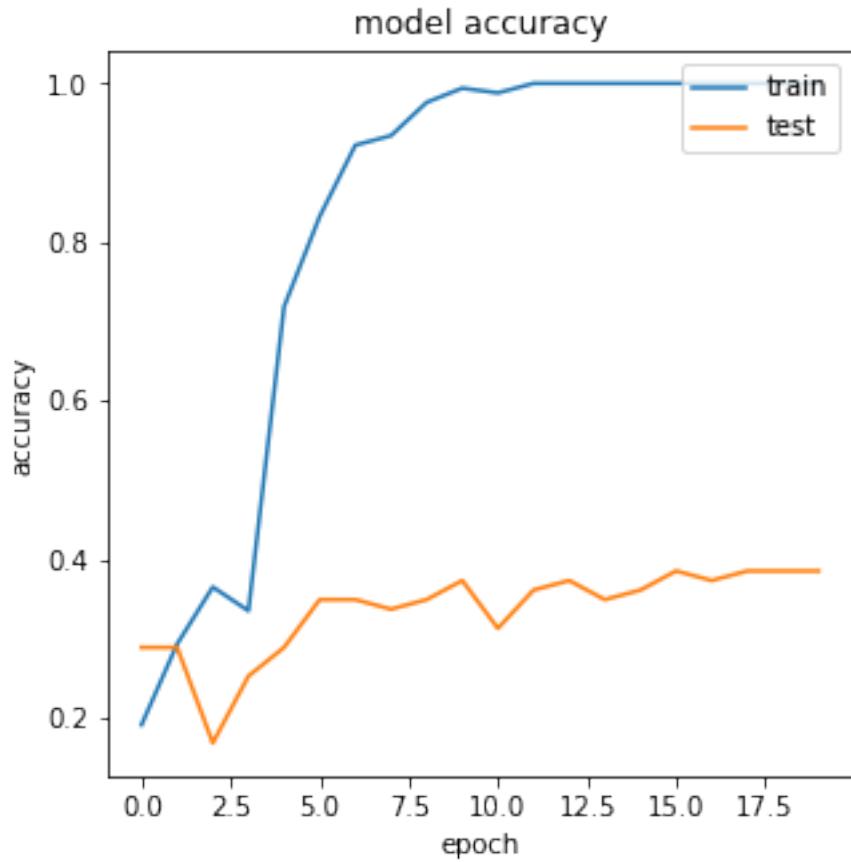
```

```

Epoch 1/20
4/4 [=====] - 2s 245ms/step - loss: 6.1558 - acc: 0.1606 - val_loss: 3.8446 - val_acc: 0.2892
Epoch 2/20
4/4 [=====] - 0s 83ms/step - loss: 4.0296 - acc: 0.2927 - val_loss: 1.7281 - val_acc: 0.2892
Epoch 3/20
4/4 [=====] - 0s 82ms/step - loss: 1.5298 - acc: 0.3888 - val_loss: 1.8839 - val_acc: 0.1687
Epoch 4/20
4/4 [=====] - 0s 81ms/step - loss: 1.4823 - acc: 0.2915 - val_loss: 1.6063 - val_acc: 0.2530
Epoch 5/20
4/4 [=====] - 0s 83ms/step - loss: 1.1050 - acc: 0.6961 - val_loss: 1.6521 - val_acc: 0.2892
Epoch 6/20
4/4 [=====] - 0s 84ms/step - loss: 0.8285 - acc: 0.8496 - val_loss: 1.6030 - val_acc: 0.3494
Epoch 7/20
4/4 [=====] - 0s 85ms/step - loss: 0.5282 - acc: 0.9182 - val_loss: 1.6907 - val_acc: 0.3494
Epoch 8/20
4/4 [=====] - 0s 84ms/step - loss: 0.3740 - acc: 0.9270 - val_loss: 1.7383 - val_acc: 0.3373
Epoch 9/20
4/4 [=====] - 0s 82ms/step - loss: 0.2062 - acc: 0.9824 - val_loss: 1.8812 - val_acc: 0.3494
Epoch 10/20
4/4 [=====] - 0s 85ms/step - loss: 0.1378 - acc: 0.9903 - val_loss: 2.1588 - val_acc: 0.3735
Epoch 11/20
4/4 [=====] - 0s 85ms/step - loss: 0.0669 - acc: 0.9905 - val_loss: 2.1512 - val_acc: 0.3133
Epoch 12/20

```

```
4/4 [=====] - 0s 83ms/step - loss: 0.0421 - acc: 1.0000
- val_loss: 2.3286 - val_acc: 0.3614
Epoch 13/20
4/4 [=====] - 0s 84ms/step - loss: 0.0232 - acc: 1.0000
- val_loss: 2.7627 - val_acc: 0.3735
Epoch 14/20
4/4 [=====] - 0s 85ms/step - loss: 0.0128 - acc: 1.0000
- val_loss: 2.9423 - val_acc: 0.3494
Epoch 15/20
4/4 [=====] - 0s 85ms/step - loss: 0.0086 - acc: 1.0000
- val_loss: 2.7577 - val_acc: 0.3614
Epoch 16/20
4/4 [=====] - 0s 85ms/step - loss: 0.0046 - acc: 1.0000
- val_loss: 2.6380 - val_acc: 0.3855
Epoch 17/20
4/4 [=====] - 0s 82ms/step - loss: 0.0035 - acc: 1.0000
- val_loss: 2.7033 - val_acc: 0.3735
Epoch 18/20
4/4 [=====] - 0s 85ms/step - loss: 0.0036 - acc: 1.0000
- val_loss: 2.7862 - val_acc: 0.3855
Epoch 19/20
4/4 [=====] - 0s 85ms/step - loss: 0.0023 - acc: 1.0000
- val_loss: 2.8895 - val_acc: 0.3855
Epoch 20/20
4/4 [=====] - 0s 80ms/step - loss: 0.0014 - acc: 1.0000
- val_loss: 3.0208 - val_acc: 0.3855
```



Investigation 7: Added back extra layer and scaled back numbers. New best test accuracy. A 2.5% increase.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", input_shape=(1, 256, ↴
    ↴256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(16, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))
```

```

modelC.add(Flatten())
modelC.add(Dense(64, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

    ↪metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50,  

    ↪verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

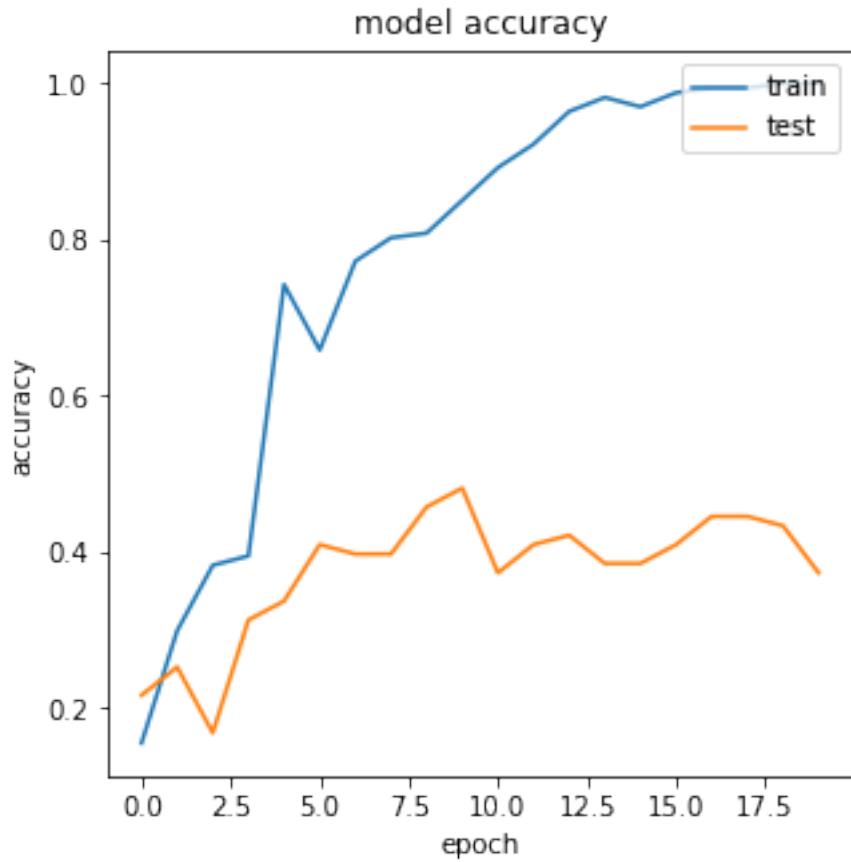
```

```

Epoch 1/20
4/4 [=====] - 1s 103ms/step - loss: 1.8618 - acc: 0.1389 - val_loss: 1.6965 - val_acc: 0.2169
Epoch 2/20
4/4 [=====] - 0s 57ms/step - loss: 1.6750 - acc: 0.2724 - val_loss: 1.5849 - val_acc: 0.2530
Epoch 3/20
4/4 [=====] - 0s 52ms/step - loss: 1.4935 - acc: 0.4506 - val_loss: 1.6449 - val_acc: 0.1687
Epoch 4/20
4/4 [=====] - 0s 53ms/step - loss: 1.4238 - acc: 0.3314 - val_loss: 1.5219 - val_acc: 0.3133
Epoch 5/20
4/4 [=====] - 0s 57ms/step - loss: 1.2075 - acc: 0.7483 - val_loss: 1.5468 - val_acc: 0.3373
Epoch 6/20
4/4 [=====] - 0s 55ms/step - loss: 0.9911 - acc: 0.6568 - val_loss: 1.4835 - val_acc: 0.4096
Epoch 7/20
4/4 [=====] - 0s 54ms/step - loss: 0.7600 - acc: 0.7756 - val_loss: 1.6836 - val_acc: 0.3976
Epoch 8/20
4/4 [=====] - 0s 56ms/step - loss: 0.6327 - acc: 0.7963 - val_loss: 1.7779 - val_acc: 0.3976
Epoch 9/20
4/4 [=====] - 0s 52ms/step - loss: 0.5187 - acc: 0.8127 - val_loss: 1.7478 - val_acc: 0.4578

```

```
Epoch 10/20
4/4 [=====] - 0s 58ms/step - loss: 0.3933 - acc: 0.8788
- val_loss: 1.8656 - val_acc: 0.4819
Epoch 11/20
4/4 [=====] - 0s 53ms/step - loss: 0.3157 - acc: 0.9049
- val_loss: 1.9827 - val_acc: 0.3735
Epoch 12/20
4/4 [=====] - 0s 54ms/step - loss: 0.2916 - acc: 0.8975
- val_loss: 2.1006 - val_acc: 0.4096
Epoch 13/20
4/4 [=====] - 0s 55ms/step - loss: 0.2348 - acc: 0.9630
- val_loss: 2.2602 - val_acc: 0.4217
Epoch 14/20
4/4 [=====] - 0s 55ms/step - loss: 0.1381 - acc: 0.9808
- val_loss: 2.1369 - val_acc: 0.3855
Epoch 15/20
4/4 [=====] - 0s 52ms/step - loss: 0.1184 - acc: 0.9634
- val_loss: 2.2553 - val_acc: 0.3855
Epoch 16/20
4/4 [=====] - 0s 54ms/step - loss: 0.0644 - acc: 0.9865
- val_loss: 2.4938 - val_acc: 0.4096
Epoch 17/20
4/4 [=====] - 0s 54ms/step - loss: 0.0359 - acc: 0.9963
- val_loss: 2.8907 - val_acc: 0.4458
Epoch 18/20
4/4 [=====] - 0s 55ms/step - loss: 0.0345 - acc: 0.9943
- val_loss: 3.1842 - val_acc: 0.4458
Epoch 19/20
4/4 [=====] - 0s 53ms/step - loss: 0.0192 - acc: 1.0000
- val_loss: 3.2924 - val_acc: 0.4337
Epoch 20/20
4/4 [=====] - 0s 51ms/step - loss: 0.0087 - acc: 1.0000
- val_loss: 3.4313 - val_acc: 0.3735
```



Investigation 8: Added another extra layer and scaled back numbers again. New best test accuracy. A 1.2% increase.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(4, (3, 3), strides=1, padding="valid", input_shape=(1, 256, ↴
    ↴256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(16, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))
```

```

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Flatten())  

modelC.add(Dense(64, activation='relu'))  

modelC.add(Dense(num_classes, activation='softmax'))  

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

    ↪metrics=['acc'])  

# Fit the model  

history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50,  

    ↪verbose=1)  

# summarize history for accuracy  

plt.plot(history.history['acc'])  

plt.plot(history.history['val_acc'])  

plt.title('model accuracy')  

plt.ylabel('accuracy')  

plt.xlabel('epoch')  

plt.legend(['train', 'test'], loc='upper right')  

plt.show()

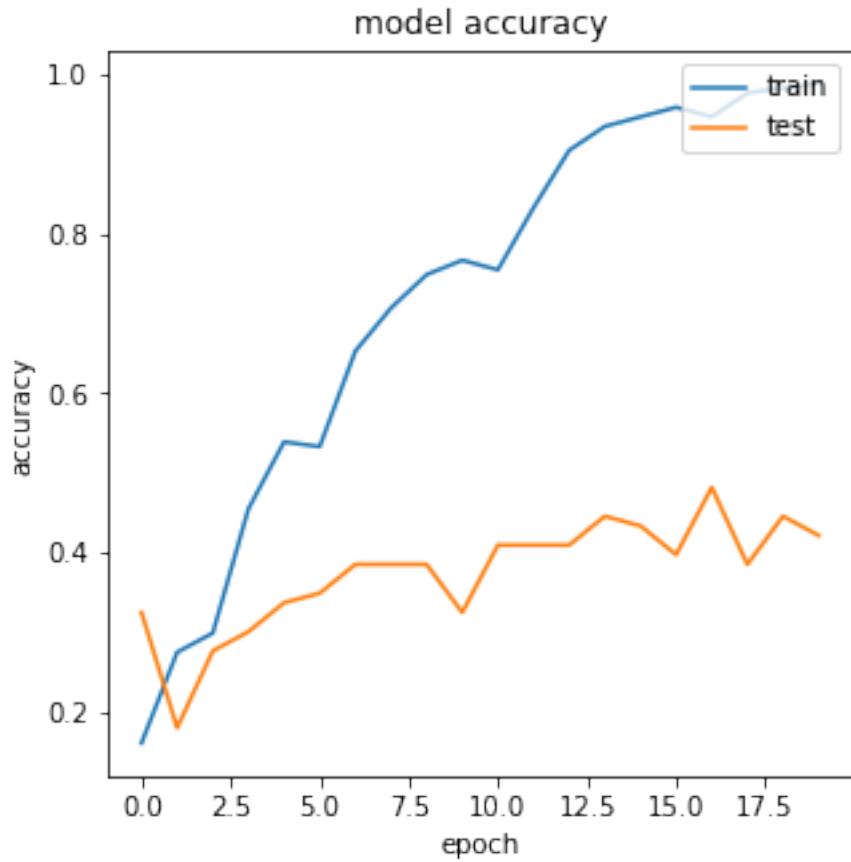
```

```

Epoch 1/20
4/4 [=====] - 1s 92ms/step - loss: 1.7444 - acc: 0.1487
- val_loss: 1.6005 - val_acc: 0.3253
Epoch 2/20
4/4 [=====] - 0s 43ms/step - loss: 1.5815 - acc: 0.3268
- val_loss: 1.6339 - val_acc: 0.1807
Epoch 3/20
4/4 [=====] - 0s 43ms/step - loss: 1.4769 - acc: 0.2951
- val_loss: 1.5222 - val_acc: 0.2771
Epoch 4/20
4/4 [=====] - 0s 45ms/step - loss: 1.3741 - acc: 0.4407
- val_loss: 1.5219 - val_acc: 0.3012
Epoch 5/20
4/4 [=====] - 0s 42ms/step - loss: 1.1805 - acc: 0.5129
- val_loss: 1.6295 - val_acc: 0.3373
Epoch 6/20
4/4 [=====] - 0s 45ms/step - loss: 1.1057 - acc: 0.5132
- val_loss: 1.5971 - val_acc: 0.3494
Epoch 7/20
4/4 [=====] - 0s 46ms/step - loss: 0.9678 - acc: 0.6477
- val_loss: 1.8115 - val_acc: 0.3855
Epoch 8/20

```

```
4/4 [=====] - 0s 43ms/step - loss: 0.8723 - acc: 0.6660
- val_loss: 1.6610 - val_acc: 0.3855
Epoch 9/20
4/4 [=====] - 0s 44ms/step - loss: 0.7173 - acc: 0.7487
- val_loss: 1.9381 - val_acc: 0.3855
Epoch 10/20
4/4 [=====] - 0s 44ms/step - loss: 0.5894 - acc: 0.7799
- val_loss: 2.1864 - val_acc: 0.3253
Epoch 11/20
4/4 [=====] - 0s 44ms/step - loss: 0.7146 - acc: 0.7085
- val_loss: 1.9486 - val_acc: 0.4096
Epoch 12/20
4/4 [=====] - 0s 43ms/step - loss: 0.5087 - acc: 0.7989
- val_loss: 1.8992 - val_acc: 0.4096
Epoch 13/20
4/4 [=====] - 0s 49ms/step - loss: 0.3648 - acc: 0.9057
- val_loss: 2.0109 - val_acc: 0.4096
Epoch 14/20
4/4 [=====] - 0s 42ms/step - loss: 0.2544 - acc: 0.9323
- val_loss: 2.1283 - val_acc: 0.4458
Epoch 15/20
4/4 [=====] - 0s 45ms/step - loss: 0.2220 - acc: 0.9411
- val_loss: 2.3357 - val_acc: 0.4337
Epoch 16/20
4/4 [=====] - 0s 45ms/step - loss: 0.1599 - acc: 0.9639
- val_loss: 2.5217 - val_acc: 0.3976
Epoch 17/20
4/4 [=====] - 0s 46ms/step - loss: 0.1682 - acc: 0.9484
- val_loss: 2.8974 - val_acc: 0.4819
Epoch 18/20
4/4 [=====] - 0s 43ms/step - loss: 0.1394 - acc: 0.9764
- val_loss: 2.7039 - val_acc: 0.3855
Epoch 19/20
4/4 [=====] - 0s 44ms/step - loss: 0.1225 - acc: 0.9768
- val_loss: 2.7297 - val_acc: 0.4458
Epoch 20/20
4/4 [=====] - 0s 42ms/step - loss: 0.0863 - acc: 0.9865
- val_loss: 2.7274 - val_acc: 0.4217
```



Investigation 9: Added another layer to see the effect. Decreases accuracy. Will try one more with adjusted nodes to see if four is the maximum.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(2, (3, 3), strides=1, padding="valid", input_shape=(1, 256, ↴
    ↴256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(4, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))
```

```

modelC.add(Conv2D(16, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Flatten())  

modelC.add(Dense(64, activation='relu'))  

modelC.add(Dense(num_classes, activation='softmax'))  

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

    ↪metrics=['acc'])  

# Fit the model  

history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50,  

    ↪verbose=1)  

# summarize history for accuracy  

plt.plot(history.history['acc'])  

plt.plot(history.history['val_acc'])  

plt.title('model accuracy')  

plt.ylabel('accuracy')  

plt.xlabel('epoch')  

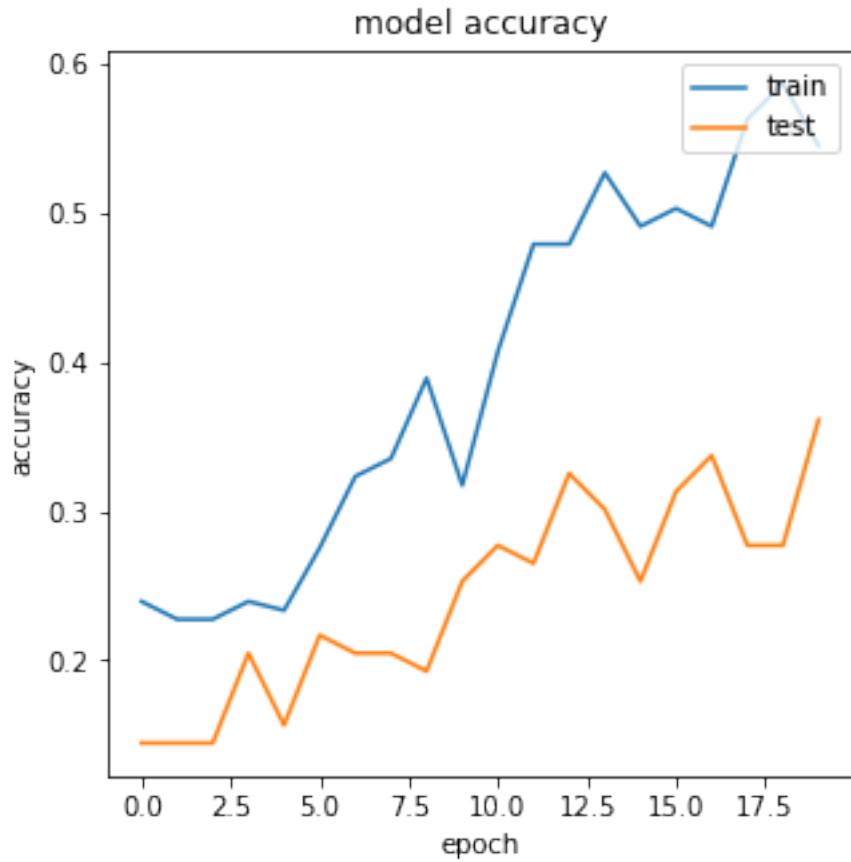
plt.legend(['train', 'test'], loc='upper right')  

plt.show()

```

Epoch 1/20
4/4 [=====] - 1s 115ms/step - loss: 1.7831 - acc: 0.2558 - val_loss: 1.7687 - val_acc: 0.1446
Epoch 2/20
4/4 [=====] - 0s 26ms/step - loss: 1.7308 - acc: 0.2317 - val_loss: 1.7266 - val_acc: 0.1446
Epoch 3/20
4/4 [=====] - 0s 25ms/step - loss: 1.6674 - acc: 0.2024 - val_loss: 1.6740 - val_acc: 0.1446
Epoch 4/20
4/4 [=====] - 0s 30ms/step - loss: 1.6103 - acc: 0.2165 - val_loss: 1.6514 - val_acc: 0.2048
Epoch 5/20
4/4 [=====] - 0s 26ms/step - loss: 1.5829 - acc: 0.2267 - val_loss: 1.6476 - val_acc: 0.1566
Epoch 6/20

```
4/4 [=====] - 0s 27ms/step - loss: 1.5617 - acc: 0.2742
- val_loss: 1.6100 - val_acc: 0.2169
Epoch 7/20
4/4 [=====] - 0s 26ms/step - loss: 1.5443 - acc: 0.3253
- val_loss: 1.6804 - val_acc: 0.2048
Epoch 8/20
4/4 [=====] - 0s 26ms/step - loss: 1.5122 - acc: 0.3408
- val_loss: 1.5899 - val_acc: 0.2048
Epoch 9/20
4/4 [=====] - 0s 26ms/step - loss: 1.5010 - acc: 0.3844
- val_loss: 1.6072 - val_acc: 0.1928
Epoch 10/20
4/4 [=====] - 0s 27ms/step - loss: 1.4856 - acc: 0.3076
- val_loss: 1.5511 - val_acc: 0.2530
Epoch 11/20
4/4 [=====] - 0s 26ms/step - loss: 1.4315 - acc: 0.3962
- val_loss: 1.5551 - val_acc: 0.2771
Epoch 12/20
4/4 [=====] - 0s 26ms/step - loss: 1.3346 - acc: 0.4970
- val_loss: 1.5419 - val_acc: 0.2651
Epoch 13/20
4/4 [=====] - 0s 28ms/step - loss: 1.3267 - acc: 0.4770
- val_loss: 1.5257 - val_acc: 0.3253
Epoch 14/20
4/4 [=====] - 0s 29ms/step - loss: 1.2238 - acc: 0.5354
- val_loss: 1.5733 - val_acc: 0.3012
Epoch 15/20
4/4 [=====] - 0s 28ms/step - loss: 1.2583 - acc: 0.4931
- val_loss: 1.5722 - val_acc: 0.2530
Epoch 16/20
4/4 [=====] - 0s 25ms/step - loss: 1.2770 - acc: 0.4712
- val_loss: 1.5496 - val_acc: 0.3133
Epoch 17/20
4/4 [=====] - 0s 26ms/step - loss: 1.2091 - acc: 0.5144
- val_loss: 1.5076 - val_acc: 0.3373
Epoch 18/20
4/4 [=====] - 0s 27ms/step - loss: 1.1665 - acc: 0.6098
- val_loss: 1.5655 - val_acc: 0.2771
Epoch 19/20
4/4 [=====] - 0s 27ms/step - loss: 1.0718 - acc: 0.6047
- val_loss: 1.5220 - val_acc: 0.2771
Epoch 20/20
4/4 [=====] - 0s 28ms/step - loss: 1.1526 - acc: 0.5093
- val_loss: 1.4972 - val_acc: 0.3614
```



Investigation 10: Five layers with adjusted nodes gives us 1.6% increase in accuracy.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(4, (3, 3), strides=1, padding="valid", input_shape=(1, 256, 256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(16, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))
```

```

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Flatten())  

modelC.add(Dense(128, activation='relu'))  

modelC.add(Dense(num_classes, activation='softmax'))  

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

    ↪metrics=['acc'])  

# Fit the model  

history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50,  

    ↪verbose=1)  

# summarize history for accuracy  

plt.plot(history.history['acc'])  

plt.plot(history.history['val_acc'])  

plt.title('model accuracy')  

plt.ylabel('accuracy')  

plt.xlabel('epoch')  

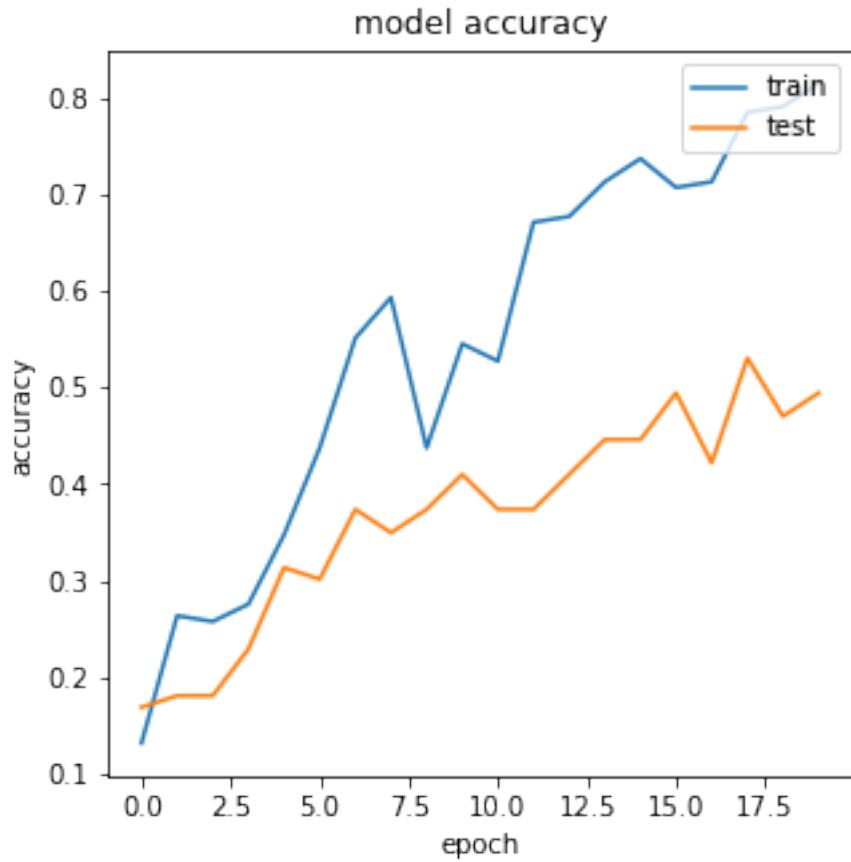
plt.legend(['train', 'test'], loc='upper right')  

plt.show()

```

Epoch 1/20
4/4 [=====] - 1s 100ms/step - loss: 1.7652 - acc: 0.1000 - val_loss: 1.6739 - val_acc: 0.1687
Epoch 2/20
4/4 [=====] - 0s 49ms/step - loss: 1.6138 - acc: 0.2707 - val_loss: 1.6295 - val_acc: 0.1807
Epoch 3/20
4/4 [=====] - 0s 47ms/step - loss: 1.5750 - acc: 0.2477 - val_loss: 1.5964 - val_acc: 0.1807
Epoch 4/20
4/4 [=====] - 0s 44ms/step - loss: 1.5141 - acc: 0.2875 - val_loss: 1.5595 - val_acc: 0.2289
Epoch 5/20
4/4 [=====] - 0s 45ms/step - loss: 1.4538 - acc: 0.3209 - val_loss: 1.6175 - val_acc: 0.3133
Epoch 6/20

```
4/4 [=====] - 0s 44ms/step - loss: 1.4260 - acc: 0.4449
- val_loss: 1.4971 - val_acc: 0.3012
Epoch 7/20
4/4 [=====] - 0s 44ms/step - loss: 1.2809 - acc: 0.5477
- val_loss: 1.4930 - val_acc: 0.3735
Epoch 8/20
4/4 [=====] - 0s 42ms/step - loss: 1.1464 - acc: 0.5738
- val_loss: 1.7164 - val_acc: 0.3494
Epoch 9/20
4/4 [=====] - 0s 43ms/step - loss: 1.3055 - acc: 0.4489
- val_loss: 1.6030 - val_acc: 0.3735
Epoch 10/20
4/4 [=====] - 0s 42ms/step - loss: 1.0836 - acc: 0.5760
- val_loss: 1.4794 - val_acc: 0.4096
Epoch 11/20
4/4 [=====] - 0s 43ms/step - loss: 1.1290 - acc: 0.5248
- val_loss: 1.5224 - val_acc: 0.3735
Epoch 12/20
4/4 [=====] - 0s 43ms/step - loss: 0.9438 - acc: 0.6243
- val_loss: 1.4797 - val_acc: 0.3735
Epoch 13/20
4/4 [=====] - 0s 45ms/step - loss: 0.8906 - acc: 0.6453
- val_loss: 1.4353 - val_acc: 0.4096
Epoch 14/20
4/4 [=====] - 0s 45ms/step - loss: 0.7884 - acc: 0.7024
- val_loss: 1.4410 - val_acc: 0.4458
Epoch 15/20
4/4 [=====] - 0s 42ms/step - loss: 0.6859 - acc: 0.7513
- val_loss: 1.6569 - val_acc: 0.4458
Epoch 16/20
4/4 [=====] - 0s 43ms/step - loss: 0.7669 - acc: 0.7033
- val_loss: 1.5047 - val_acc: 0.4940
Epoch 17/20
4/4 [=====] - 0s 45ms/step - loss: 0.5884 - acc: 0.7290
- val_loss: 1.6586 - val_acc: 0.4217
Epoch 18/20
4/4 [=====] - 0s 43ms/step - loss: 0.6144 - acc: 0.7671
- val_loss: 1.5422 - val_acc: 0.5301
Epoch 19/20
4/4 [=====] - 0s 46ms/step - loss: 0.5786 - acc: 0.7695
- val_loss: 1.5321 - val_acc: 0.4699
Epoch 20/20
4/4 [=====] - 0s 43ms/step - loss: 0.4509 - acc: 0.8344
- val_loss: 1.5923 - val_acc: 0.4940
```



Investigation 11: Added another layer. Six layers with adjusted nodes gives us 1.8% increase in accuracy.

```
[15]: modelC = Sequential()
modelC.add(Conv2D(4, (3, 3), strides=1, padding="valid", input_shape=(1, 256, ↴
    ↴256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))
```

```

modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(128, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(256, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Flatten())  

modelC.add(Dense(512, activation='relu'))  

modelC.add(Dense(num_classes, activation='softmax'))  

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

    ↪metrics=['acc'])  

# Fit the model  

history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50,  

    ↪verbose=1)  

# summarize history for accuracy  

plt.plot(history.history['acc'])  

plt.plot(history.history['val_acc'])  

plt.title('model accuracy')  

plt.ylabel('accuracy')  

plt.xlabel('epoch')  

plt.legend(['train', 'test'], loc='upper right')  

plt.show()

```

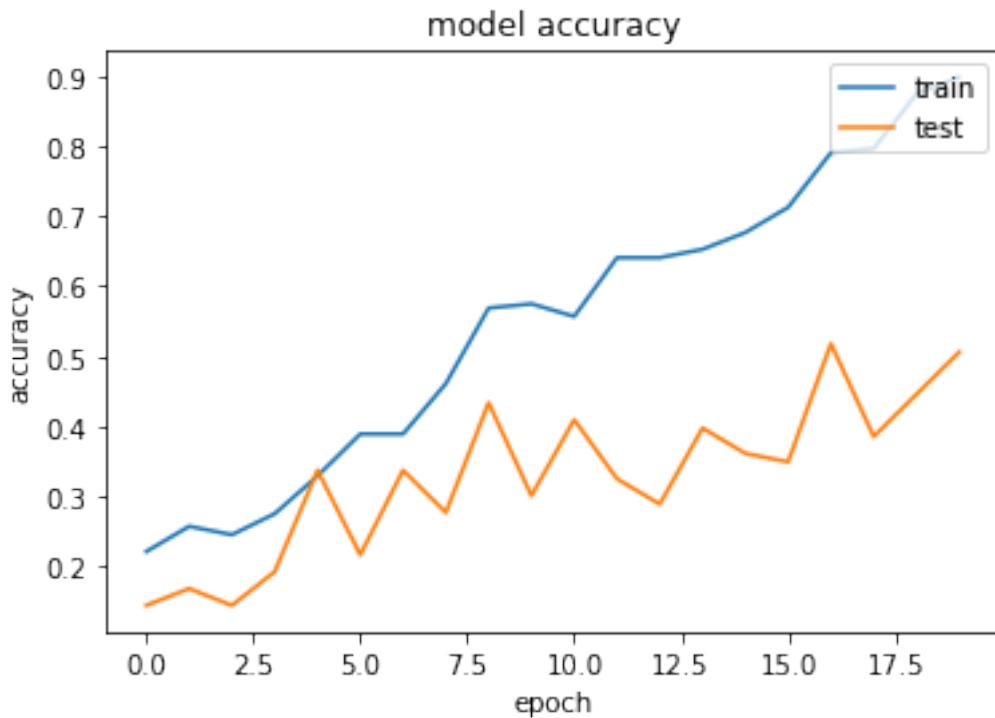
```

Epoch 1/20
4/4 [=====] - 1s 98ms/step - loss: 1.7592 - acc: 0.2220
- val_loss: 1.6979 - val_acc: 0.1446
Epoch 2/20
4/4 [=====] - 0s 46ms/step - loss: 1.6332 - acc: 0.2630
- val_loss: 1.6011 - val_acc: 0.1687
Epoch 3/20
4/4 [=====] - 0s 42ms/step - loss: 1.5877 - acc: 0.2429
- val_loss: 1.6895 - val_acc: 0.1446
Epoch 4/20
4/4 [=====] - 0s 42ms/step - loss: 1.5908 - acc: 0.2408

```

```
- val_loss: 1.6382 - val_acc: 0.1928
Epoch 5/20
4/4 [=====] - 0s 38ms/step - loss: 1.5061 - acc: 0.3311
- val_loss: 1.5363 - val_acc: 0.3373
Epoch 6/20
4/4 [=====] - 0s 38ms/step - loss: 1.4275 - acc: 0.4224
- val_loss: 1.6035 - val_acc: 0.2169
Epoch 7/20
4/4 [=====] - 0s 38ms/step - loss: 1.3430 - acc: 0.4124
- val_loss: 1.5043 - val_acc: 0.3373
Epoch 8/20
4/4 [=====] - 0s 38ms/step - loss: 1.2984 - acc: 0.4451
- val_loss: 1.5920 - val_acc: 0.2771
Epoch 9/20
4/4 [=====] - 0s 38ms/step - loss: 1.1719 - acc: 0.5875
- val_loss: 1.5467 - val_acc: 0.4337
Epoch 10/20
4/4 [=====] - 0s 37ms/step - loss: 1.1247 - acc: 0.5833
- val_loss: 1.6169 - val_acc: 0.3012
Epoch 11/20
4/4 [=====] - 0s 36ms/step - loss: 1.0923 - acc: 0.6074
- val_loss: 1.4458 - val_acc: 0.4096
Epoch 12/20
4/4 [=====] - 0s 35ms/step - loss: 1.0316 - acc: 0.6396
- val_loss: 1.6636 - val_acc: 0.3253
Epoch 13/20
4/4 [=====] - 0s 36ms/step - loss: 0.9437 - acc: 0.6423
- val_loss: 1.9154 - val_acc: 0.2892
Epoch 14/20
4/4 [=====] - 0s 36ms/step - loss: 0.9944 - acc: 0.5991
- val_loss: 1.8513 - val_acc: 0.3976
Epoch 15/20
4/4 [=====] - 0s 38ms/step - loss: 0.9295 - acc: 0.6560
- val_loss: 1.6495 - val_acc: 0.3614
Epoch 16/20
4/4 [=====] - 0s 38ms/step - loss: 0.7182 - acc: 0.7297
- val_loss: 1.7223 - val_acc: 0.3494
Epoch 17/20
4/4 [=====] - 0s 40ms/step - loss: 0.6253 - acc: 0.7902
- val_loss: 1.6947 - val_acc: 0.5181
Epoch 18/20
4/4 [=====] - 0s 38ms/step - loss: 0.5714 - acc: 0.7806
- val_loss: 1.8602 - val_acc: 0.3855
Epoch 19/20
4/4 [=====] - 0s 37ms/step - loss: 0.4620 - acc: 0.8750
- val_loss: 1.8403 - val_acc: 0.4458
Epoch 20/20
4/4 [=====] - 0s 36ms/step - loss: 0.2842 - acc: 0.9193
```

- val_loss: 2.0012 - val_acc: 0.5060



Investigation 12: Six layers with increased nodes compared to last investigation. A decrease in accuracy overall.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", input_shape=(1, 256, 256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(16, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
```

```

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(128, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(256, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(512, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam', ↴
    ↪metrics=['acc'])

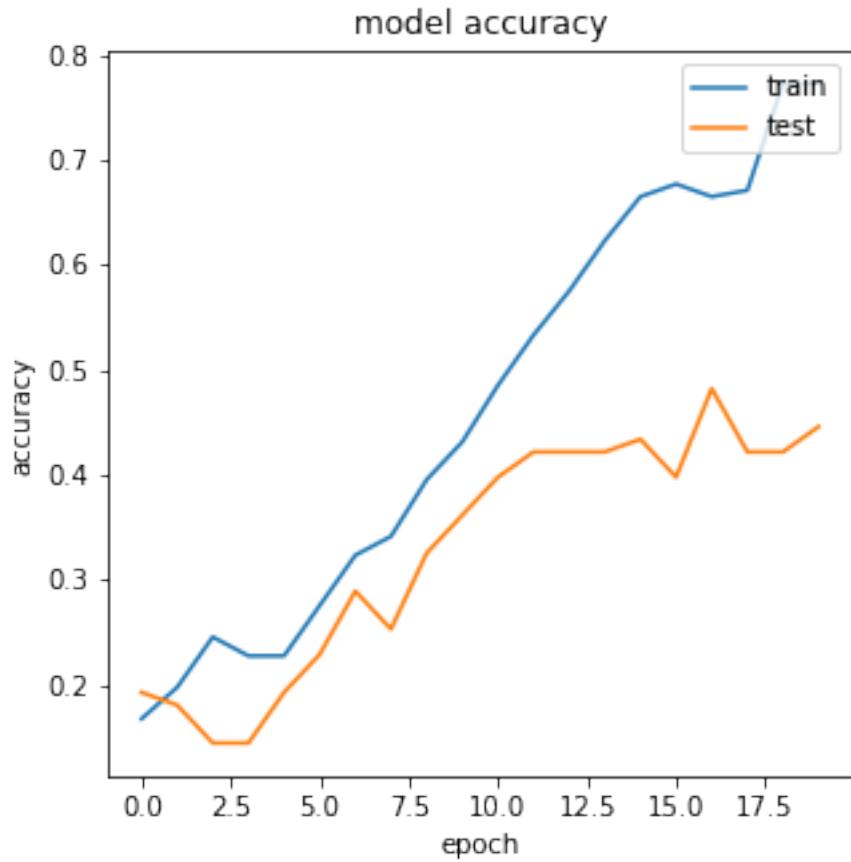
# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50, ↴
    ↪verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

Epoch 1/20
4/4 [=====] - 1s 113ms/step - loss: 1.7446 - acc: 0.1504 - val_loss: 1.6401 - val_acc: 0.1928
Epoch 2/20
4/4 [=====] - 0s 62ms/step - loss: 1.6293 - acc: 0.2137 - val_loss: 1.6483 - val_acc: 0.1807
Epoch 3/20
4/4 [=====] - 0s 57ms/step - loss: 1.6059 - acc: 0.2429 - val_loss: 1.7127 - val_acc: 0.1446
Epoch 4/20
4/4 [=====] - 0s 58ms/step - loss: 1.6316 - acc: 0.2097 - val_loss: 1.6649 - val_acc: 0.1446
Epoch 5/20

```
4/4 [=====] - 0s 59ms/step - loss: 1.5970 - acc: 0.2357
- val_loss: 1.6201 - val_acc: 0.1928
Epoch 6/20
4/4 [=====] - 0s 59ms/step - loss: 1.5692 - acc: 0.2742
- val_loss: 1.5826 - val_acc: 0.2289
Epoch 7/20
4/4 [=====] - 0s 56ms/step - loss: 1.5254 - acc: 0.3133
- val_loss: 1.6873 - val_acc: 0.2892
Epoch 8/20
4/4 [=====] - 0s 55ms/step - loss: 1.5286 - acc: 0.3405
- val_loss: 1.5389 - val_acc: 0.2530
Epoch 9/20
4/4 [=====] - 0s 59ms/step - loss: 1.4771 - acc: 0.3668
- val_loss: 1.5969 - val_acc: 0.3253
Epoch 10/20
4/4 [=====] - 0s 58ms/step - loss: 1.3687 - acc: 0.4278
- val_loss: 1.4448 - val_acc: 0.3614
Epoch 11/20
4/4 [=====] - 0s 57ms/step - loss: 1.3374 - acc: 0.4933
- val_loss: 1.4624 - val_acc: 0.3976
Epoch 12/20
4/4 [=====] - 0s 57ms/step - loss: 1.1917 - acc: 0.5138
- val_loss: 1.4398 - val_acc: 0.4217
Epoch 13/20
4/4 [=====] - 0s 62ms/step - loss: 1.1322 - acc: 0.5733
- val_loss: 1.4856 - val_acc: 0.4217
Epoch 14/20
4/4 [=====] - 0s 56ms/step - loss: 1.0614 - acc: 0.6144
- val_loss: 1.4609 - val_acc: 0.4217
Epoch 15/20
4/4 [=====] - 0s 57ms/step - loss: 0.8558 - acc: 0.6832
- val_loss: 1.4923 - val_acc: 0.4337
Epoch 16/20
4/4 [=====] - 0s 57ms/step - loss: 0.7789 - acc: 0.6873
- val_loss: 1.8216 - val_acc: 0.3976
Epoch 17/20
4/4 [=====] - 0s 56ms/step - loss: 0.8484 - acc: 0.6625
- val_loss: 1.6583 - val_acc: 0.4819
Epoch 18/20
4/4 [=====] - 0s 55ms/step - loss: 0.8591 - acc: 0.6329
- val_loss: 1.7475 - val_acc: 0.4217
Epoch 19/20
4/4 [=====] - 0s 58ms/step - loss: 0.6525 - acc: 0.7703
- val_loss: 1.7840 - val_acc: 0.4217
Epoch 20/20
4/4 [=====] - 0s 55ms/step - loss: 0.5916 - acc: 0.7770
- val_loss: 1.6626 - val_acc: 0.4458
```



Six layers are the max we can go.

7 Hyperparameter tuning of batch size, epochs and kernel filter sizes (batch sizes x 3 (suitable epochs), and two different kernel sizes)

For our hyperparameter tuning, we are using Investigation 11 as it provided the best test performance of all our investigations.

Investigation 1: Increase in epochs and batch size. Makes no substantial difference to accuracy.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(4, (3, 3), strides=1, padding="valid", input_shape=(1, 256, 256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
```

```

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(128, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(256, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(512, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam', ↴
    ↪metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=30, batch_size=128, ↴
    ↪verbose=1)

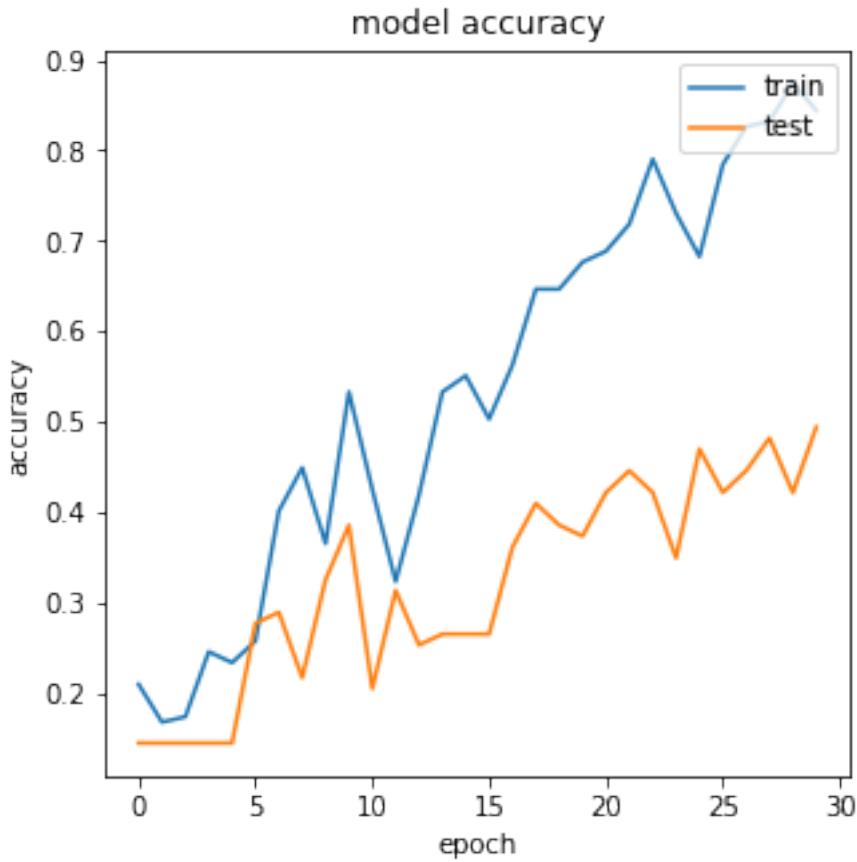
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

Epoch 1/30
2/2 [=====] - 1s 239ms/step - loss: 1.7845 - acc: 0.2048 - val_loss: 1.6947 - val_acc: 0.1446

```
Epoch 2/30
2/2 [=====] - 0s 75ms/step - loss: 1.6462 - acc: 0.1717
- val_loss: 1.8355 - val_acc: 0.1446
Epoch 3/30
2/2 [=====] - 0s 78ms/step - loss: 1.6993 - acc: 0.1757
- val_loss: 1.6300 - val_acc: 0.1446
Epoch 4/30
2/2 [=====] - 0s 69ms/step - loss: 1.5944 - acc: 0.2314
- val_loss: 1.6236 - val_acc: 0.1446
Epoch 5/30
2/2 [=====] - 0s 68ms/step - loss: 1.5889 - acc: 0.2364
- val_loss: 1.6122 - val_acc: 0.1446
Epoch 6/30
2/2 [=====] - 0s 69ms/step - loss: 1.5685 - acc: 0.2602
- val_loss: 1.6138 - val_acc: 0.2771
Epoch 7/30
2/2 [=====] - 0s 67ms/step - loss: 1.5506 - acc: 0.4159
- val_loss: 1.6342 - val_acc: 0.2892
Epoch 8/30
2/2 [=====] - 0s 64ms/step - loss: 1.5317 - acc: 0.4504
- val_loss: 1.6314 - val_acc: 0.2169
Epoch 9/30
2/2 [=====] - 0s 64ms/step - loss: 1.4838 - acc: 0.3789
- val_loss: 1.5245 - val_acc: 0.3253
Epoch 10/30
2/2 [=====] - 0s 66ms/step - loss: 1.4350 - acc: 0.5272
- val_loss: 1.5033 - val_acc: 0.3855
Epoch 11/30
2/2 [=====] - 0s 68ms/step - loss: 1.3884 - acc: 0.4241
- val_loss: 1.9000 - val_acc: 0.2048
Epoch 12/30
2/2 [=====] - 0s 69ms/step - loss: 1.5424 - acc: 0.3145
- val_loss: 1.5361 - val_acc: 0.3133
Epoch 13/30
2/2 [=====] - 0s 65ms/step - loss: 1.3110 - acc: 0.4070
- val_loss: 1.5399 - val_acc: 0.2530
Epoch 14/30
2/2 [=====] - 0s 66ms/step - loss: 1.2375 - acc: 0.5272
- val_loss: 1.5434 - val_acc: 0.2651
Epoch 15/30
2/2 [=====] - 0s 66ms/step - loss: 1.2048 - acc: 0.5443
- val_loss: 1.5804 - val_acc: 0.2651
Epoch 16/30
2/2 [=====] - 0s 67ms/step - loss: 1.1501 - acc: 0.4994
- val_loss: 1.6220 - val_acc: 0.2651
Epoch 17/30
2/2 [=====] - 0s 67ms/step - loss: 1.0872 - acc: 0.5706
- val_loss: 1.5730 - val_acc: 0.3614
```

```
Epoch 18/30
2/2 [=====] - 0s 65ms/step - loss: 0.9907 - acc: 0.6395
- val_loss: 1.5338 - val_acc: 0.4096
Epoch 19/30
2/2 [=====] - 0s 69ms/step - loss: 0.9289 - acc: 0.6577
- val_loss: 1.6244 - val_acc: 0.3855
Epoch 20/30
2/2 [=====] - 0s 68ms/step - loss: 0.8247 - acc: 0.6829
- val_loss: 1.6412 - val_acc: 0.3735
Epoch 21/30
2/2 [=====] - 0s 68ms/step - loss: 0.8341 - acc: 0.6830
- val_loss: 1.6499 - val_acc: 0.4217
Epoch 22/30
2/2 [=====] - 0s 65ms/step - loss: 0.7692 - acc: 0.7108
- val_loss: 1.6283 - val_acc: 0.4458
Epoch 23/30
2/2 [=====] - 0s 74ms/step - loss: 0.6864 - acc: 0.7874
- val_loss: 1.6935 - val_acc: 0.4217
Epoch 24/30
2/2 [=====] - 0s 68ms/step - loss: 0.6869 - acc: 0.7318
- val_loss: 2.0524 - val_acc: 0.3494
Epoch 25/30
2/2 [=====] - 0s 65ms/step - loss: 0.8167 - acc: 0.6817
- val_loss: 1.5362 - val_acc: 0.4699
Epoch 26/30
2/2 [=====] - 0s 67ms/step - loss: 0.6642 - acc: 0.7834
- val_loss: 1.6594 - val_acc: 0.4217
Epoch 27/30
2/2 [=====] - 0s 70ms/step - loss: 0.5884 - acc: 0.8269
- val_loss: 1.8281 - val_acc: 0.4458
Epoch 28/30
2/2 [=====] - 0s 67ms/step - loss: 0.5553 - acc: 0.8283
- val_loss: 1.7207 - val_acc: 0.4819
Epoch 29/30
2/2 [=====] - 0s 84ms/step - loss: 0.4422 - acc: 0.8771
- val_loss: 1.9085 - val_acc: 0.4217
Epoch 30/30
2/2 [=====] - 0s 65ms/step - loss: 0.4695 - acc: 0.8415
- val_loss: 1.9222 - val_acc: 0.4940
```



Investigation 2: Increased epochs to 40 and batch_size to 1000. Slightly worse performance.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(4, (3, 3), strides=1, padding="valid", input_shape=(1, 256, 256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))
```

```

modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(128, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(256, (3, 3), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Flatten())  

modelC.add(Dense(512, activation='relu'))  

modelC.add(Dense(num_classes, activation='softmax'))  

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

    ↪metrics=['acc'])  

# Fit the model  

history = modelC.fit(X, Y, validation_split=0.33, epochs=40, batch_size=1000,  

    ↪verbose=1)  

# summarize history for accuracy  

plt.plot(history.history['acc'])  

plt.plot(history.history['val_acc'])  

plt.title('model accuracy')  

plt.ylabel('accuracy')  

plt.xlabel('epoch')  

plt.legend(['train', 'test'], loc='upper right')  

plt.show()

```

```

Epoch 1/40
1/1 [=====] - 1s 815ms/step - loss: 1.7950 - acc: 0.1976 - val_loss: 1.7404 - val_acc: 0.2048
Epoch 2/40
1/1 [=====] - 0s 139ms/step - loss: 1.7239 - acc: 0.1976 - val_loss: 1.6995 - val_acc: 0.2048
Epoch 3/40
1/1 [=====] - 0s 142ms/step - loss: 1.6468 - acc: 0.1976 - val_loss: 1.7238 - val_acc: 0.1446
Epoch 4/40
1/1 [=====] - 0s 130ms/step - loss: 1.6173 - acc:

```

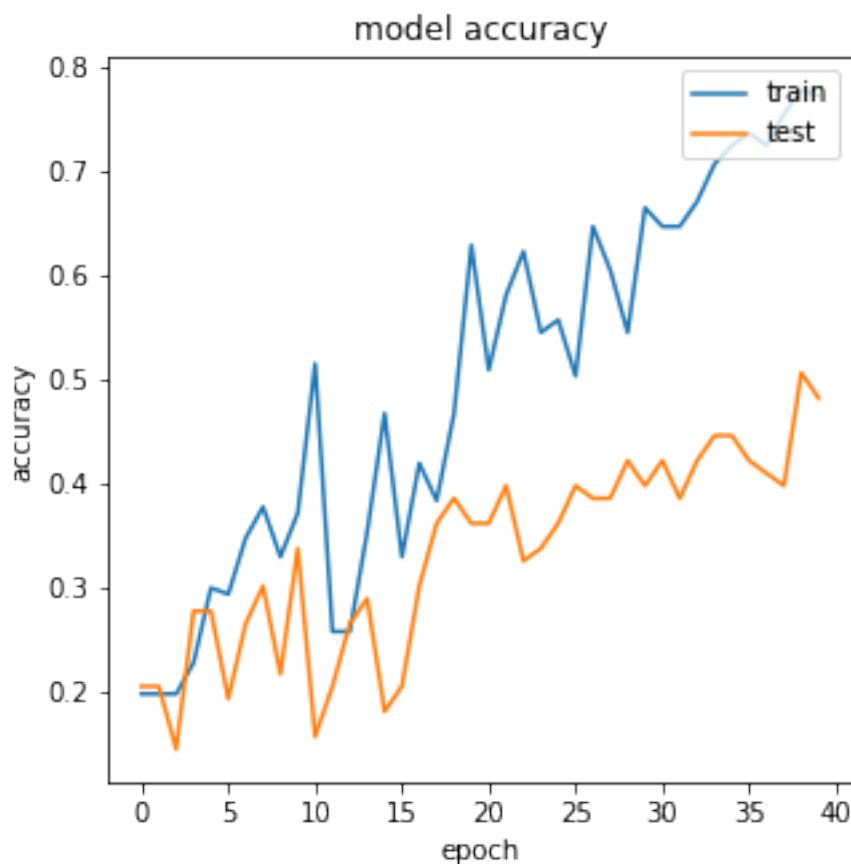
```
0.2275 - val_loss: 1.6052 - val_acc: 0.2771
Epoch 5/40
1/1 [=====] - 0s 132ms/step - loss: 1.5950 - acc:
0.2994 - val_loss: 1.6005 - val_acc: 0.2771
Epoch 6/40
1/1 [=====] - 0s 121ms/step - loss: 1.5982 - acc:
0.2934 - val_loss: 1.6804 - val_acc: 0.1928
Epoch 7/40
1/1 [=====] - 0s 126ms/step - loss: 1.5828 - acc:
0.3473 - val_loss: 1.6919 - val_acc: 0.2651
Epoch 8/40
1/1 [=====] - 0s 120ms/step - loss: 1.5694 - acc:
0.3772 - val_loss: 1.5672 - val_acc: 0.3012
Epoch 9/40
1/1 [=====] - 0s 125ms/step - loss: 1.5463 - acc:
0.3293 - val_loss: 1.5785 - val_acc: 0.2169
Epoch 10/40
1/1 [=====] - 0s 117ms/step - loss: 1.5186 - acc:
0.3713 - val_loss: 1.6283 - val_acc: 0.3373
Epoch 11/40
1/1 [=====] - 0s 116ms/step - loss: 1.5014 - acc:
0.5150 - val_loss: 1.7302 - val_acc: 0.1566
Epoch 12/40
1/1 [=====] - 0s 115ms/step - loss: 1.5257 - acc:
0.2575 - val_loss: 1.5663 - val_acc: 0.2048
Epoch 13/40
1/1 [=====] - 0s 111ms/step - loss: 1.5813 - acc:
0.2575 - val_loss: 1.5819 - val_acc: 0.2651
Epoch 14/40
1/1 [=====] - 0s 116ms/step - loss: 1.5235 - acc:
0.3533 - val_loss: 1.6037 - val_acc: 0.2892
Epoch 15/40
1/1 [=====] - 0s 120ms/step - loss: 1.4557 - acc:
0.4671 - val_loss: 1.6414 - val_acc: 0.1807
Epoch 16/40
1/1 [=====] - 0s 126ms/step - loss: 1.4523 - acc:
0.3293 - val_loss: 1.5777 - val_acc: 0.2048
Epoch 17/40
1/1 [=====] - 0s 113ms/step - loss: 1.3969 - acc:
0.4192 - val_loss: 1.5660 - val_acc: 0.3012
Epoch 18/40
1/1 [=====] - 0s 109ms/step - loss: 1.3934 - acc:
0.3832 - val_loss: 1.5280 - val_acc: 0.3614
Epoch 19/40
1/1 [=====] - 0s 118ms/step - loss: 1.3486 - acc:
0.4671 - val_loss: 1.4641 - val_acc: 0.3855
Epoch 20/40
1/1 [=====] - 0s 112ms/step - loss: 1.2643 - acc:
```

```
0.6287 - val_loss: 1.4826 - val_acc: 0.3614
Epoch 21/40
1/1 [=====] - 0s 107ms/step - loss: 1.2376 - acc:
0.5090 - val_loss: 1.5120 - val_acc: 0.3614
Epoch 22/40
1/1 [=====] - 0s 119ms/step - loss: 1.1814 - acc:
0.5808 - val_loss: 1.4436 - val_acc: 0.3976
Epoch 23/40
1/1 [=====] - 0s 113ms/step - loss: 1.0796 - acc:
0.6228 - val_loss: 1.4893 - val_acc: 0.3253
Epoch 24/40
1/1 [=====] - 0s 121ms/step - loss: 1.0896 - acc:
0.5449 - val_loss: 1.5590 - val_acc: 0.3373
Epoch 25/40
1/1 [=====] - 0s 120ms/step - loss: 1.1696 - acc:
0.5569 - val_loss: 1.6872 - val_acc: 0.3614
Epoch 26/40
1/1 [=====] - 0s 112ms/step - loss: 1.2217 - acc:
0.5030 - val_loss: 1.5362 - val_acc: 0.3976
Epoch 27/40
1/1 [=====] - 0s 115ms/step - loss: 1.0445 - acc:
0.6467 - val_loss: 1.4719 - val_acc: 0.3855
Epoch 28/40
1/1 [=====] - 0s 109ms/step - loss: 1.0001 - acc:
0.6048 - val_loss: 1.5165 - val_acc: 0.3855
Epoch 29/40
1/1 [=====] - 0s 114ms/step - loss: 1.0802 - acc:
0.5449 - val_loss: 1.3935 - val_acc: 0.4217
Epoch 30/40
1/1 [=====] - 0s 125ms/step - loss: 0.9197 - acc:
0.6647 - val_loss: 1.5130 - val_acc: 0.3976
Epoch 31/40
1/1 [=====] - 0s 120ms/step - loss: 0.9539 - acc:
0.6467 - val_loss: 1.5551 - val_acc: 0.4217
Epoch 32/40
1/1 [=====] - 0s 121ms/step - loss: 0.9213 - acc:
0.6467 - val_loss: 1.5806 - val_acc: 0.3855
Epoch 33/40
1/1 [=====] - 0s 116ms/step - loss: 0.8686 - acc:
0.6707 - val_loss: 1.5395 - val_acc: 0.4217
Epoch 34/40
1/1 [=====] - 0s 111ms/step - loss: 0.8177 - acc:
0.7066 - val_loss: 1.5085 - val_acc: 0.4458
Epoch 35/40
1/1 [=====] - 0s 112ms/step - loss: 0.8092 - acc:
0.7246 - val_loss: 1.4932 - val_acc: 0.4458
Epoch 36/40
1/1 [=====] - 0s 124ms/step - loss: 0.7746 - acc:
```

```

0.7365 - val_loss: 1.5680 - val_acc: 0.4217
Epoch 37/40
1/1 [=====] - 0s 119ms/step - loss: 0.7332 - acc:
0.7246 - val_loss: 1.6686 - val_acc: 0.4096
Epoch 38/40
1/1 [=====] - 0s 112ms/step - loss: 0.7061 - acc:
0.7545 - val_loss: 1.6404 - val_acc: 0.3976
Epoch 39/40
1/1 [=====] - 0s 110ms/step - loss: 0.6927 - acc:
0.7784 - val_loss: 1.4971 - val_acc: 0.5060
Epoch 40/40
1/1 [=====] - 0s 107ms/step - loss: 0.6560 - acc:
0.7725 - val_loss: 1.5391 - val_acc: 0.4819

```



Investigation 2: Decreased batch size to 800, epochs to 30, worse performance.

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(4, (3, 3), strides=1, padding="valid", input_shape=(1, 256, 256), activation='relu', data_format='channels_first'))
```

```

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(128, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(256, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(512, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam', ↴
    ↪metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=30, batch_size=800, ↴
    ↪verbose=1)

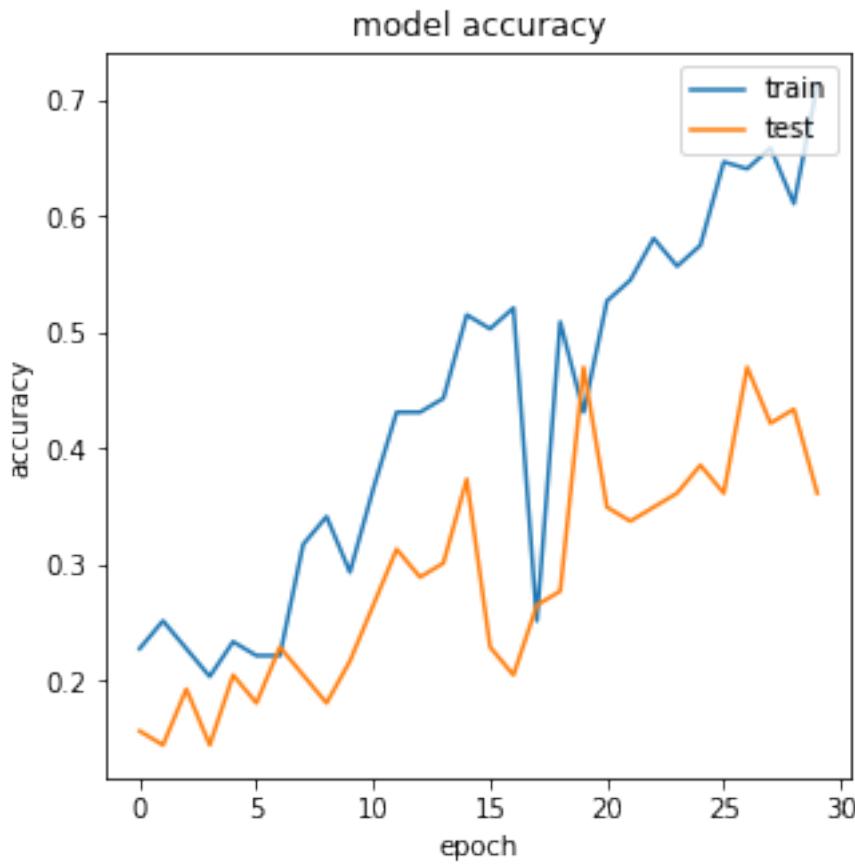
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')

```

```
plt.legend(['train', 'test'], loc='upper right')
plt.show()
```

Epoch 1/30
1/1 [=====] - 1s 800ms/step - loss: 1.8106 - acc:
0.2275 - val_loss: 1.6962 - val_acc: 0.1566
Epoch 2/30
1/1 [=====] - 0s 154ms/step - loss: 1.6754 - acc:
0.2515 - val_loss: 1.6783 - val_acc: 0.1446
Epoch 3/30
1/1 [=====] - 0s 134ms/step - loss: 1.6082 - acc:
0.2275 - val_loss: 1.6242 - val_acc: 0.1928
Epoch 4/30
1/1 [=====] - 0s 123ms/step - loss: 1.6086 - acc:
0.2036 - val_loss: 1.6775 - val_acc: 0.1446
Epoch 5/30
1/1 [=====] - 0s 120ms/step - loss: 1.6205 - acc:
0.2335 - val_loss: 1.6519 - val_acc: 0.2048
Epoch 6/30
1/1 [=====] - 0s 110ms/step - loss: 1.6235 - acc:
0.2216 - val_loss: 1.6582 - val_acc: 0.1807
Epoch 7/30
1/1 [=====] - 0s 110ms/step - loss: 1.5799 - acc:
0.2216 - val_loss: 1.6519 - val_acc: 0.2289
Epoch 8/30
1/1 [=====] - 0s 119ms/step - loss: 1.5762 - acc:
0.3174 - val_loss: 1.6154 - val_acc: 0.2048
Epoch 9/30
1/1 [=====] - 0s 114ms/step - loss: 1.5697 - acc:
0.3413 - val_loss: 1.5894 - val_acc: 0.1807
Epoch 10/30
1/1 [=====] - 0s 112ms/step - loss: 1.5472 - acc:
0.2934 - val_loss: 1.5919 - val_acc: 0.2169
Epoch 11/30
1/1 [=====] - 0s 114ms/step - loss: 1.5215 - acc:
0.3653 - val_loss: 1.6192 - val_acc: 0.2651
Epoch 12/30
1/1 [=====] - 0s 121ms/step - loss: 1.5080 - acc:
0.4311 - val_loss: 1.6140 - val_acc: 0.3133
Epoch 13/30
1/1 [=====] - 0s 117ms/step - loss: 1.4818 - acc:
0.4311 - val_loss: 1.5734 - val_acc: 0.2892
Epoch 14/30
1/1 [=====] - 0s 122ms/step - loss: 1.4329 - acc:
0.4431 - val_loss: 1.5614 - val_acc: 0.3012
Epoch 15/30
1/1 [=====] - 0s 112ms/step - loss: 1.3828 - acc:
0.5150 - val_loss: 1.4677 - val_acc: 0.3735

```
Epoch 16/30
1/1 [=====] - 0s 119ms/step - loss: 1.3418 - acc: 0.5030 - val_loss: 1.6590 - val_acc: 0.2289
Epoch 17/30
1/1 [=====] - 0s 118ms/step - loss: 1.3278 - acc: 0.5210 - val_loss: 1.7218 - val_acc: 0.2048
Epoch 18/30
1/1 [=====] - 0s 119ms/step - loss: 1.4827 - acc: 0.2515 - val_loss: 1.5760 - val_acc: 0.2651
Epoch 19/30
1/1 [=====] - 0s 109ms/step - loss: 1.2434 - acc: 0.5090 - val_loss: 1.6230 - val_acc: 0.2771
Epoch 20/30
1/1 [=====] - 0s 118ms/step - loss: 1.3800 - acc: 0.4311 - val_loss: 1.4509 - val_acc: 0.4699
Epoch 21/30
1/1 [=====] - 0s 114ms/step - loss: 1.2527 - acc: 0.5269 - val_loss: 1.4489 - val_acc: 0.3494
Epoch 22/30
1/1 [=====] - 0s 115ms/step - loss: 1.1908 - acc: 0.5449 - val_loss: 1.5008 - val_acc: 0.3373
Epoch 23/30
1/1 [=====] - 0s 112ms/step - loss: 1.1353 - acc: 0.5808 - val_loss: 1.5755 - val_acc: 0.3494
Epoch 24/30
1/1 [=====] - 0s 111ms/step - loss: 1.1254 - acc: 0.5569 - val_loss: 1.6064 - val_acc: 0.3614
Epoch 25/30
1/1 [=====] - 0s 112ms/step - loss: 1.1176 - acc: 0.5749 - val_loss: 1.5029 - val_acc: 0.3855
Epoch 26/30
1/1 [=====] - 0s 112ms/step - loss: 1.0186 - acc: 0.6467 - val_loss: 1.4207 - val_acc: 0.3614
Epoch 27/30
1/1 [=====] - 0s 122ms/step - loss: 0.9787 - acc: 0.6407 - val_loss: 1.3622 - val_acc: 0.4699
Epoch 28/30
1/1 [=====] - 0s 115ms/step - loss: 0.9277 - acc: 0.6587 - val_loss: 1.4397 - val_acc: 0.4217
Epoch 29/30
1/1 [=====] - 0s 112ms/step - loss: 0.9243 - acc: 0.6108 - val_loss: 1.4152 - val_acc: 0.4337
Epoch 30/30
1/1 [=====] - 0s 118ms/step - loss: 0.8261 - acc: 0.7126 - val_loss: 1.5049 - val_acc: 0.3614
```



Investigation 3: Modified batch size and kernel size. Slightly worse performance.

```
[ ]: modelC = Sequential()

modelC.add(Conv2D(8, (4, 4), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))

modelC.add(Conv2D(32, (4, 4), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))

modelC.add(Conv2D(64, (4, 4), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))
```

```

modelC.add(Conv2D(128, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(256, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(512, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam', ↴
    ↪metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=30, batch_size=8, ↴
    ↪verbose=1)

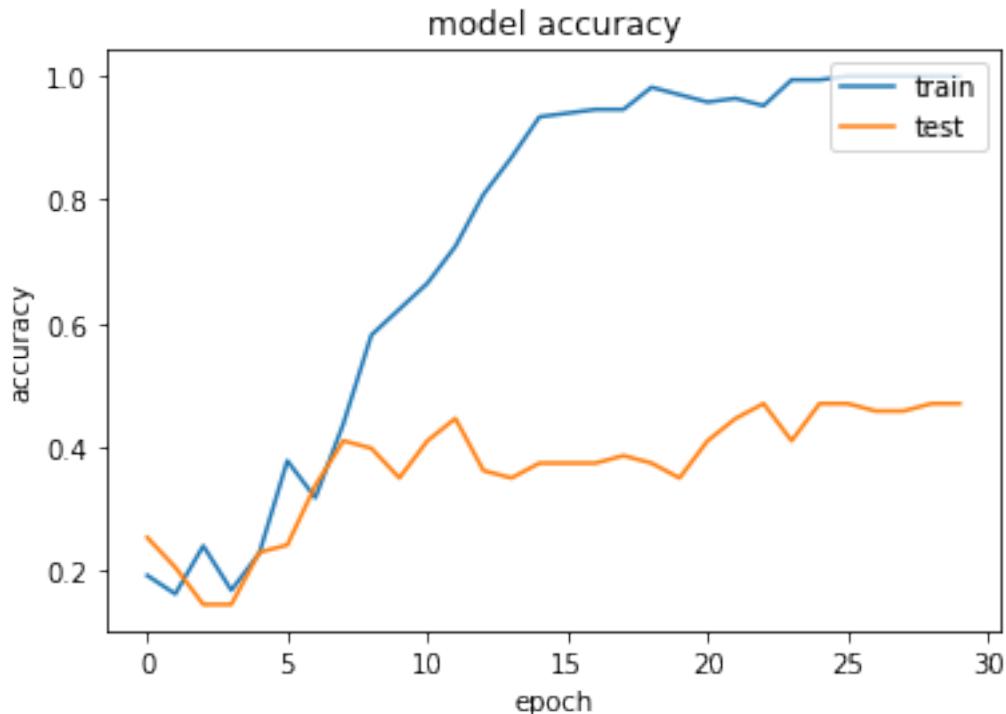
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

Epoch 1/30
21/21 [=====] - 1s 28ms/step - loss: 1.8504 - acc:
0.2185 - val_loss: 1.6758 - val_acc: 0.2530
Epoch 2/30
21/21 [=====] - 0s 17ms/step - loss: 1.6516 - acc:
0.2109 - val_loss: 1.6137 - val_acc: 0.2048
Epoch 3/30
21/21 [=====] - 0s 16ms/step - loss: 1.6283 - acc:
0.2091 - val_loss: 1.6549 - val_acc: 0.1446
Epoch 4/30
21/21 [=====] - 0s 16ms/step - loss: 1.6094 - acc:
0.1789 - val_loss: 1.6307 - val_acc: 0.1446
Epoch 5/30
21/21 [=====] - 0s 16ms/step - loss: 1.6119 - acc:
0.2527 - val_loss: 1.6054 - val_acc: 0.2289
Epoch 6/30

```
21/21 [=====] - 0s 16ms/step - loss: 1.5020 - acc:  
0.4129 - val_loss: 1.6611 - val_acc: 0.2410  
Epoch 7/30  
21/21 [=====] - 0s 16ms/step - loss: 1.5702 - acc:  
0.2860 - val_loss: 1.5450 - val_acc: 0.3373  
Epoch 8/30  
21/21 [=====] - 0s 16ms/step - loss: 1.4194 - acc:  
0.3959 - val_loss: 1.4178 - val_acc: 0.4096  
Epoch 9/30  
21/21 [=====] - 0s 17ms/step - loss: 1.1361 - acc:  
0.5754 - val_loss: 1.4500 - val_acc: 0.3976  
Epoch 10/30  
21/21 [=====] - 0s 16ms/step - loss: 1.0582 - acc:  
0.6224 - val_loss: 1.9171 - val_acc: 0.3494  
Epoch 11/30  
21/21 [=====] - 0s 16ms/step - loss: 0.9806 - acc:  
0.6638 - val_loss: 1.7524 - val_acc: 0.4096  
Epoch 12/30  
21/21 [=====] - 0s 17ms/step - loss: 0.7462 - acc:  
0.7010 - val_loss: 1.9087 - val_acc: 0.4458  
Epoch 13/30  
21/21 [=====] - 0s 16ms/step - loss: 0.5093 - acc:  
0.8030 - val_loss: 2.7133 - val_acc: 0.3614  
Epoch 14/30  
21/21 [=====] - 0s 16ms/step - loss: 0.5001 - acc:  
0.8387 - val_loss: 2.6932 - val_acc: 0.3494  
Epoch 15/30  
21/21 [=====] - 0s 16ms/step - loss: 0.3204 - acc:  
0.9338 - val_loss: 2.9119 - val_acc: 0.3735  
Epoch 16/30  
21/21 [=====] - 0s 16ms/step - loss: 0.0987 - acc:  
0.9708 - val_loss: 4.4458 - val_acc: 0.3735  
Epoch 17/30  
21/21 [=====] - 0s 16ms/step - loss: 0.1869 - acc:  
0.9355 - val_loss: 3.9053 - val_acc: 0.3735  
Epoch 18/30  
21/21 [=====] - 0s 16ms/step - loss: 0.0861 - acc:  
0.9625 - val_loss: 4.6671 - val_acc: 0.3855  
Epoch 19/30  
21/21 [=====] - 0s 16ms/step - loss: 0.0885 - acc:  
0.9839 - val_loss: 5.3662 - val_acc: 0.3735  
Epoch 20/30  
21/21 [=====] - 0s 16ms/step - loss: 0.0509 - acc:  
0.9774 - val_loss: 4.2401 - val_acc: 0.3494  
Epoch 21/30  
21/21 [=====] - 0s 16ms/step - loss: 0.1002 - acc:  
0.9332 - val_loss: 6.1974 - val_acc: 0.4096  
Epoch 22/30
```

```
21/21 [=====] - 0s 16ms/step - loss: 0.0844 - acc: 0.9681 - val_loss: 5.2397 - val_acc: 0.4458
Epoch 23/30
21/21 [=====] - 0s 16ms/step - loss: 0.1311 - acc: 0.9529 - val_loss: 3.8698 - val_acc: 0.4699
Epoch 24/30
21/21 [=====] - 0s 16ms/step - loss: 0.0186 - acc: 0.9979 - val_loss: 4.7096 - val_acc: 0.4096
Epoch 25/30
21/21 [=====] - 0s 16ms/step - loss: 0.0183 - acc: 0.9894 - val_loss: 5.6084 - val_acc: 0.4699
Epoch 26/30
21/21 [=====] - 0s 16ms/step - loss: 0.0026 - acc: 1.0000 - val_loss: 6.0788 - val_acc: 0.4699
Epoch 27/30
21/21 [=====] - 0s 16ms/step - loss: 9.6032e-04 - acc: 1.0000 - val_loss: 6.1025 - val_acc: 0.4578
Epoch 28/30
21/21 [=====] - 0s 16ms/step - loss: 7.7097e-04 - acc: 1.0000 - val_loss: 6.1511 - val_acc: 0.4578
Epoch 29/30
21/21 [=====] - 0s 16ms/step - loss: 3.8923e-04 - acc: 1.0000 - val_loss: 6.1914 - val_acc: 0.4699
Epoch 30/30
21/21 [=====] - 0s 16ms/step - loss: 5.8277e-04 - acc: 1.0000 - val_loss: 6.2822 - val_acc: 0.4699
```



Investigation 4: Added back initial layer, changed batch size and epochs. Better performance.

```
[ ]: modelC = Sequential()

modelC.add(Conv2D(4, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(8, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(32, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(64, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(128, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(256, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(512, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam', ↴
    ↴metrics=['acc'])

# Fit the model
```

```

history = modelC.fit(X, Y, validation_split=0.33, epochs=40, batch_size=8,verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

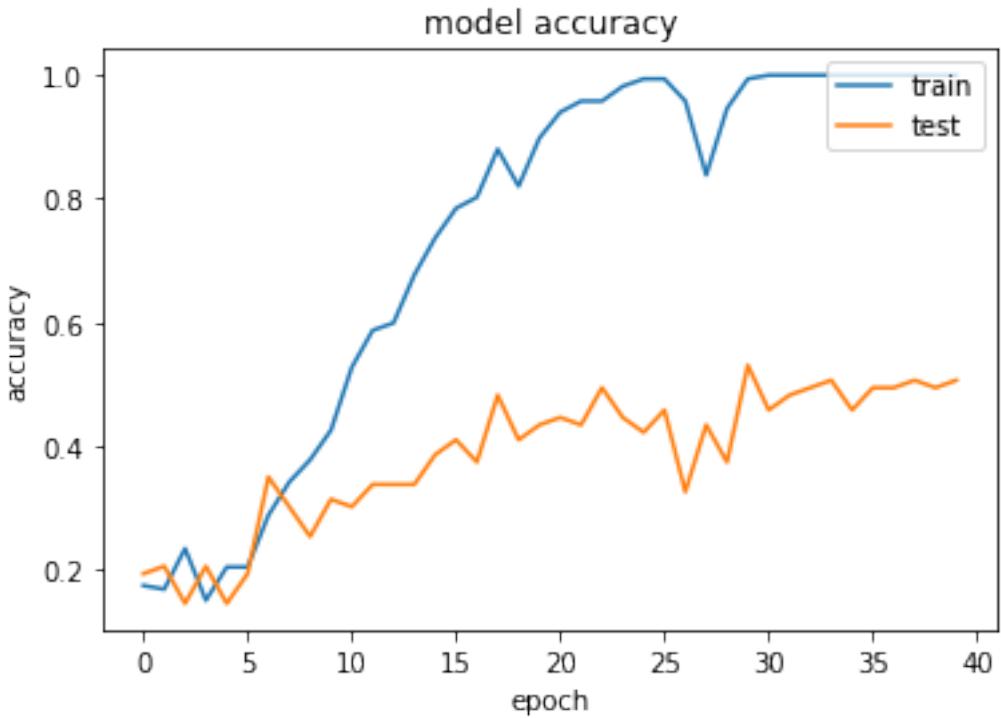
```

Epoch 1/40
21/21 [=====] - 1s 23ms/step - loss: 1.7884 - acc: 0.1585 - val_loss: 1.6729 - val_acc: 0.1928
Epoch 2/40
21/21 [=====] - 0s 13ms/step - loss: 1.6598 - acc: 0.2026 - val_loss: 1.6298 - val_acc: 0.2048
Epoch 3/40
21/21 [=====] - 0s 12ms/step - loss: 1.6351 - acc: 0.2077 - val_loss: 1.6500 - val_acc: 0.1446
Epoch 4/40
21/21 [=====] - 0s 12ms/step - loss: 1.6152 - acc: 0.1767 - val_loss: 1.6500 - val_acc: 0.2048
Epoch 5/40
21/21 [=====] - 0s 10ms/step - loss: 1.6208 - acc: 0.1744 - val_loss: 1.6327 - val_acc: 0.1446
Epoch 6/40
21/21 [=====] - 0s 10ms/step - loss: 1.6106 - acc: 0.2143 - val_loss: 1.6296 - val_acc: 0.1928
Epoch 7/40
21/21 [=====] - 0s 10ms/step - loss: 1.6007 - acc: 0.2730 - val_loss: 1.5851 - val_acc: 0.3494
Epoch 8/40
21/21 [=====] - 0s 11ms/step - loss: 1.5598 - acc: 0.3350 - val_loss: 1.5252 - val_acc: 0.3012
Epoch 9/40
21/21 [=====] - 0s 11ms/step - loss: 1.4053 - acc: 0.3567 - val_loss: 1.5522 - val_acc: 0.2530
Epoch 10/40
21/21 [=====] - 0s 10ms/step - loss: 1.3964 - acc: 0.3730 - val_loss: 1.4671 - val_acc: 0.3133
Epoch 11/40
21/21 [=====] - 0s 10ms/step - loss: 1.1940 - acc: 0.5344 - val_loss: 1.5579 - val_acc: 0.3012
Epoch 12/40
21/21 [=====] - 0s 10ms/step - loss: 1.0748 - acc: 0.5321 - val_loss: 1.6862 - val_acc: 0.3373

```

```
Epoch 13/40
21/21 [=====] - 0s 10ms/step - loss: 0.8997 - acc: 0.5951 - val_loss: 1.5976 - val_acc: 0.3373
Epoch 14/40
21/21 [=====] - 0s 10ms/step - loss: 0.8709 - acc: 0.6837 - val_loss: 1.8450 - val_acc: 0.3373
Epoch 15/40
21/21 [=====] - 0s 10ms/step - loss: 0.7283 - acc: 0.7035 - val_loss: 2.1456 - val_acc: 0.3855
Epoch 16/40
21/21 [=====] - 0s 11ms/step - loss: 0.5199 - acc: 0.8038 - val_loss: 2.3043 - val_acc: 0.4096
Epoch 17/40
21/21 [=====] - 0s 11ms/step - loss: 0.4954 - acc: 0.8153 - val_loss: 2.5469 - val_acc: 0.3735
Epoch 18/40
21/21 [=====] - 0s 11ms/step - loss: 0.4578 - acc: 0.8398 - val_loss: 2.2559 - val_acc: 0.4819
Epoch 19/40
21/21 [=====] - 0s 10ms/step - loss: 0.3982 - acc: 0.8767 - val_loss: 1.7433 - val_acc: 0.4096
Epoch 20/40
21/21 [=====] - 0s 10ms/step - loss: 0.4433 - acc: 0.8768 - val_loss: 2.0413 - val_acc: 0.4337
Epoch 21/40
21/21 [=====] - 0s 10ms/step - loss: 0.1804 - acc: 0.9388 - val_loss: 2.9809 - val_acc: 0.4458
Epoch 22/40
21/21 [=====] - 0s 11ms/step - loss: 0.0878 - acc: 0.9538 - val_loss: 2.9909 - val_acc: 0.4337
Epoch 23/40
21/21 [=====] - 0s 11ms/step - loss: 0.0951 - acc: 0.9739 - val_loss: 2.9043 - val_acc: 0.4940
Epoch 24/40
21/21 [=====] - 0s 11ms/step - loss: 0.0944 - acc: 0.9755 - val_loss: 3.0928 - val_acc: 0.4458
Epoch 25/40
21/21 [=====] - 0s 10ms/step - loss: 0.0285 - acc: 0.9992 - val_loss: 4.2424 - val_acc: 0.4217
Epoch 26/40
21/21 [=====] - 0s 11ms/step - loss: 0.0080 - acc: 0.9989 - val_loss: 3.6208 - val_acc: 0.4578
Epoch 27/40
21/21 [=====] - 0s 10ms/step - loss: 0.1107 - acc: 0.9619 - val_loss: 6.9845 - val_acc: 0.3253
Epoch 28/40
21/21 [=====] - 0s 10ms/step - loss: 0.8281 - acc: 0.8299 - val_loss: 2.1689 - val_acc: 0.4337
```

```
Epoch 29/40
21/21 [=====] - 0s 11ms/step - loss: 0.2508 - acc: 0.9129 - val_loss: 2.8568 - val_acc: 0.3735
Epoch 30/40
21/21 [=====] - 0s 10ms/step - loss: 0.0548 - acc: 0.9929 - val_loss: 2.9030 - val_acc: 0.5301
Epoch 31/40
21/21 [=====] - 0s 11ms/step - loss: 0.0122 - acc: 1.0000 - val_loss: 3.5913 - val_acc: 0.4578
Epoch 32/40
21/21 [=====] - 0s 10ms/step - loss: 0.0042 - acc: 1.0000 - val_loss: 4.0779 - val_acc: 0.4819
Epoch 33/40
21/21 [=====] - 0s 10ms/step - loss: 9.9972e-04 - acc: 1.0000 - val_loss: 4.3346 - val_acc: 0.4940
Epoch 34/40
21/21 [=====] - 0s 10ms/step - loss: 5.1091e-04 - acc: 1.0000 - val_loss: 4.4846 - val_acc: 0.5060
Epoch 35/40
21/21 [=====] - 0s 11ms/step - loss: 2.8626e-04 - acc: 1.0000 - val_loss: 4.5702 - val_acc: 0.4578
Epoch 36/40
21/21 [=====] - 0s 10ms/step - loss: 2.8625e-04 - acc: 1.0000 - val_loss: 4.7800 - val_acc: 0.4940
Epoch 37/40
21/21 [=====] - 0s 10ms/step - loss: 2.0108e-04 - acc: 1.0000 - val_loss: 5.0266 - val_acc: 0.4940
Epoch 38/40
21/21 [=====] - 0s 10ms/step - loss: 1.5026e-04 - acc: 1.0000 - val_loss: 5.3293 - val_acc: 0.5060
Epoch 39/40
21/21 [=====] - 0s 10ms/step - loss: 7.6990e-05 - acc: 1.0000 - val_loss: 5.4187 - val_acc: 0.4940
Epoch 40/40
21/21 [=====] - 0s 11ms/step - loss: 3.5817e-05 - acc: 1.0000 - val_loss: 5.6755 - val_acc: 0.5060
```



Investigation 5: Removed initial layer again, changed nodes, epochs and kernel size again. Not great performance at all.

```
[ ]: modelC = Sequential()

modelC.add(Conv2D(16, (8, 8), strides=1, padding="valid", activation='relu',
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",
    ↪data_format='channels_first'))

modelC.add(Conv2D(32, (8, 8), strides=1, padding="valid", activation='relu',
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",
    ↪data_format='channels_first'))

modelC.add(Conv2D(64, (8, 8), strides=1, padding="valid", activation='relu',
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",
    ↪data_format='channels_first'))

modelC.add(Conv2D(128, (8, 8), strides=1, padding="valid", activation='relu',
    ↪data_format='channels_first'))
```

```

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  
  

modelC.add(Conv2D(256, (8, 8), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  
  

modelC.add(Flatten())  

modelC.add(Dense(512, activation='relu'))  

modelC.add(Dense(num_classes, activation='softmax'))  
  

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

    ↪metrics=['acc'])  
  

# Fit the model  

history = modelC.fit(X, Y, validation_split=0.33, epochs=40, batch_size=8,  

    ↪verbose=1)  
  

# summarize history for accuracy  

plt.plot(history.history['acc'])  

plt.plot(history.history['val_acc'])  

plt.title('model accuracy')  

plt.ylabel('accuracy')  

plt.xlabel('epoch')  

plt.legend(['train', 'test'], loc='upper right')  

plt.show()

```

Epoch 1/40
21/21 [=====] - 1s 35ms/step - loss: 1.7707 - acc:
0.1763 - val_loss: 1.6745 - val_acc: 0.1807
Epoch 2/40
21/21 [=====] - 0s 24ms/step - loss: 1.6309 - acc:
0.2012 - val_loss: 1.6359 - val_acc: 0.2169
Epoch 3/40
21/21 [=====] - 0s 24ms/step - loss: 1.6303 - acc:
0.2080 - val_loss: 1.6566 - val_acc: 0.1807
Epoch 4/40
21/21 [=====] - 0s 24ms/step - loss: 1.6060 - acc:
0.3093 - val_loss: 1.6506 - val_acc: 0.1325
Epoch 5/40
21/21 [=====] - 0s 24ms/step - loss: 1.6393 - acc:
0.2621 - val_loss: 1.6621 - val_acc: 0.1446
Epoch 6/40
21/21 [=====] - 0s 24ms/step - loss: 1.6200 - acc:
0.2295 - val_loss: 1.6609 - val_acc: 0.1807

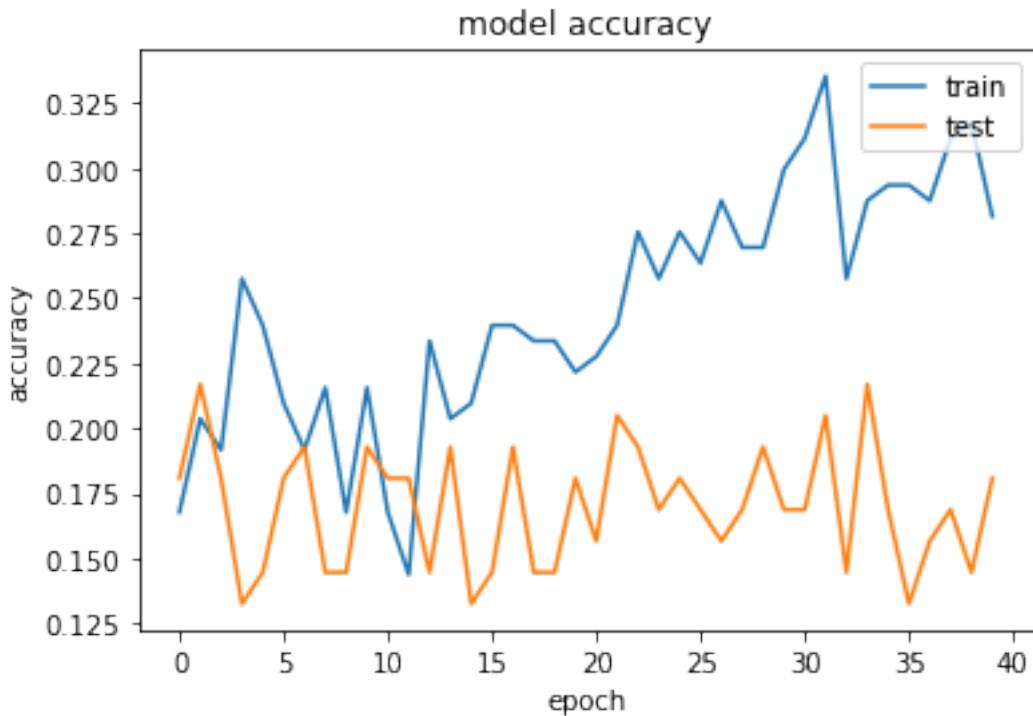
Epoch 7/40
21/21 [=====] - 0s 24ms/step - loss: 1.6134 - acc:
0.2070 - val_loss: 1.6342 - val_acc: 0.1928
Epoch 8/40
21/21 [=====] - 0s 23ms/step - loss: 1.6093 - acc:
0.2477 - val_loss: 1.6417 - val_acc: 0.1446
Epoch 9/40
21/21 [=====] - 0s 24ms/step - loss: 1.6022 - acc:
0.2129 - val_loss: 1.6434 - val_acc: 0.1446
Epoch 10/40
21/21 [=====] - 0s 24ms/step - loss: 1.6097 - acc:
0.2376 - val_loss: 1.6365 - val_acc: 0.1928
Epoch 11/40
21/21 [=====] - 0s 24ms/step - loss: 1.6127 - acc:
0.1874 - val_loss: 1.6319 - val_acc: 0.1807
Epoch 12/40
21/21 [=====] - 0s 24ms/step - loss: 1.6075 - acc:
0.1495 - val_loss: 1.6465 - val_acc: 0.1807
Epoch 13/40
21/21 [=====] - 0s 23ms/step - loss: 1.6264 - acc:
0.2241 - val_loss: 1.6444 - val_acc: 0.1446
Epoch 14/40
21/21 [=====] - 0s 23ms/step - loss: 1.6140 - acc:
0.2208 - val_loss: 1.6451 - val_acc: 0.1928
Epoch 15/40
21/21 [=====] - 0s 23ms/step - loss: 1.5917 - acc:
0.2283 - val_loss: 1.6556 - val_acc: 0.1325
Epoch 16/40
21/21 [=====] - 0s 24ms/step - loss: 1.5960 - acc:
0.2713 - val_loss: 1.6724 - val_acc: 0.1446
Epoch 17/40
21/21 [=====] - 0s 24ms/step - loss: 1.5902 - acc:
0.2393 - val_loss: 1.6744 - val_acc: 0.1928
Epoch 18/40
21/21 [=====] - 0s 23ms/step - loss: 1.5695 - acc:
0.2431 - val_loss: 1.6881 - val_acc: 0.1446
Epoch 19/40
21/21 [=====] - 0s 24ms/step - loss: 1.5963 - acc:
0.2160 - val_loss: 1.6750 - val_acc: 0.1446
Epoch 20/40
21/21 [=====] - 0s 24ms/step - loss: 1.5850 - acc:
0.2320 - val_loss: 1.7130 - val_acc: 0.1807
Epoch 21/40
21/21 [=====] - 0s 24ms/step - loss: 1.5598 - acc:
0.2518 - val_loss: 1.6908 - val_acc: 0.1566
Epoch 22/40
21/21 [=====] - 0s 24ms/step - loss: 1.5181 - acc:
0.2399 - val_loss: 1.7250 - val_acc: 0.2048

```
Epoch 23/40
21/21 [=====] - 0s 24ms/step - loss: 1.5046 - acc: 0.2985 - val_loss: 1.7841 - val_acc: 0.1928
Epoch 24/40
21/21 [=====] - 0s 24ms/step - loss: 1.5311 - acc: 0.2666 - val_loss: 1.8885 - val_acc: 0.1687
Epoch 25/40
21/21 [=====] - 0s 24ms/step - loss: 1.5132 - acc: 0.3054 - val_loss: 1.6836 - val_acc: 0.1807
Epoch 26/40
21/21 [=====] - 0s 24ms/step - loss: 1.5764 - acc: 0.2868 - val_loss: 1.6714 - val_acc: 0.1687
Epoch 27/40
21/21 [=====] - 0s 23ms/step - loss: 1.5009 - acc: 0.2927 - val_loss: 1.9955 - val_acc: 0.1566
Epoch 28/40
21/21 [=====] - 0s 24ms/step - loss: 1.4764 - acc: 0.3003 - val_loss: 1.6428 - val_acc: 0.1687
Epoch 29/40
21/21 [=====] - 0s 24ms/step - loss: 1.5005 - acc: 0.2813 - val_loss: 1.6883 - val_acc: 0.1928
Epoch 30/40
21/21 [=====] - 0s 24ms/step - loss: 1.4231 - acc: 0.3364 - val_loss: 1.7773 - val_acc: 0.1687
Epoch 31/40
21/21 [=====] - 0s 24ms/step - loss: 1.4854 - acc: 0.3486 - val_loss: 1.8669 - val_acc: 0.1687
Epoch 32/40
21/21 [=====] - 0s 24ms/step - loss: 1.4449 - acc: 0.3121 - val_loss: 1.9696 - val_acc: 0.2048
Epoch 33/40
21/21 [=====] - 0s 24ms/step - loss: 1.6098 - acc: 0.2556 - val_loss: 1.6863 - val_acc: 0.1446
Epoch 34/40
21/21 [=====] - 0s 24ms/step - loss: 1.5423 - acc: 0.3157 - val_loss: 2.0534 - val_acc: 0.2169
Epoch 35/40
21/21 [=====] - 0s 24ms/step - loss: 1.5467 - acc: 0.2895 - val_loss: 2.1040 - val_acc: 0.1687
Epoch 36/40
21/21 [=====] - 0s 24ms/step - loss: 1.4720 - acc: 0.3497 - val_loss: 3.6816 - val_acc: 0.1325
Epoch 37/40
21/21 [=====] - 0s 24ms/step - loss: 1.6216 - acc: 0.2273 - val_loss: 1.9887 - val_acc: 0.1566
Epoch 38/40
21/21 [=====] - 0s 24ms/step - loss: 1.4159 - acc: 0.3289 - val_loss: 1.9907 - val_acc: 0.1687
```

```

Epoch 39/40
21/21 [=====] - 0s 24ms/step - loss: 1.4800 - acc: 0.2957 - val_loss: 2.1775 - val_acc: 0.1446
Epoch 40/40
21/21 [=====] - 0s 24ms/step - loss: 1.4175 - acc: 0.3062 - val_loss: 2.2529 - val_acc: 0.1807

```



Investigation 4 provides the best performance, so we will use this to add depth to the network.

We changed:

Batch sizes & Epochs: **Four times**

Kernel filter sizes: **Two times**

8 Development of two further deeper CNN topologies (based on best hyperparameters from the previous model)

Investigation 1: Added another dense layer. Slightly worse performance.

```
[ ]: modelC = Sequential()
```

```

modelC.add(Conv2D(4, (4, 4), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(8, (4, 4), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(32, (4, 4), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(64, (4, 4), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Conv2D(128, (4, 4), strides=1, padding="valid", activation='relu',  

    ↪data_format='channels_first'))  

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  

    ↪data_format='channels_first'))  

modelC.add(Flatten())  

modelC.add(Dense(64, activation='relu'))  

modelC.add(Dense(128, activation='relu'))  

modelC.add(Dense(num_classes, activation='softmax'))  

modelC.compile(loss='categorical_crossentropy', optimizer='adam',  

    ↪metrics=['acc'])  

# Fit the model  

history = modelC.fit(X, Y, validation_split=0.33, epochs=30, batch_size=8,  

    ↪verbose=1)  

# summarize history for accuracy  

plt.plot(history.history['acc'])  

plt.plot(history.history['val_acc'])

```

```

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

Epoch 1/30
21/21 [=====] - 1s 24ms/step - loss: 1.8011 - acc:
0.1847 - val_loss: 1.7063 - val_acc: 0.1807

Epoch 2/30
21/21 [=====] - 0s 14ms/step - loss: 1.6737 - acc:
0.1956 - val_loss: 1.6271 - val_acc: 0.2048

Epoch 3/30
21/21 [=====] - 0s 11ms/step - loss: 1.6438 - acc:
0.1924 - val_loss: 1.6603 - val_acc: 0.1446

Epoch 4/30
21/21 [=====] - 0s 11ms/step - loss: 1.6213 - acc:
0.1686 - val_loss: 1.6431 - val_acc: 0.2048

Epoch 5/30
21/21 [=====] - 0s 11ms/step - loss: 1.6276 - acc:
0.1897 - val_loss: 1.6373 - val_acc: 0.1446

Epoch 6/30
21/21 [=====] - 0s 12ms/step - loss: 1.6122 - acc:
0.2099 - val_loss: 1.6326 - val_acc: 0.1807

Epoch 7/30
21/21 [=====] - 0s 11ms/step - loss: 1.6077 - acc:
0.2829 - val_loss: 1.6086 - val_acc: 0.2169

Epoch 8/30
21/21 [=====] - 0s 12ms/step - loss: 1.6045 - acc:
0.3053 - val_loss: 1.6393 - val_acc: 0.1446

Epoch 9/30
21/21 [=====] - 0s 12ms/step - loss: 1.5699 - acc:
0.2798 - val_loss: 1.6146 - val_acc: 0.2289

Epoch 10/30
21/21 [=====] - 0s 12ms/step - loss: 1.5887 - acc:
0.1993 - val_loss: 1.6566 - val_acc: 0.1325

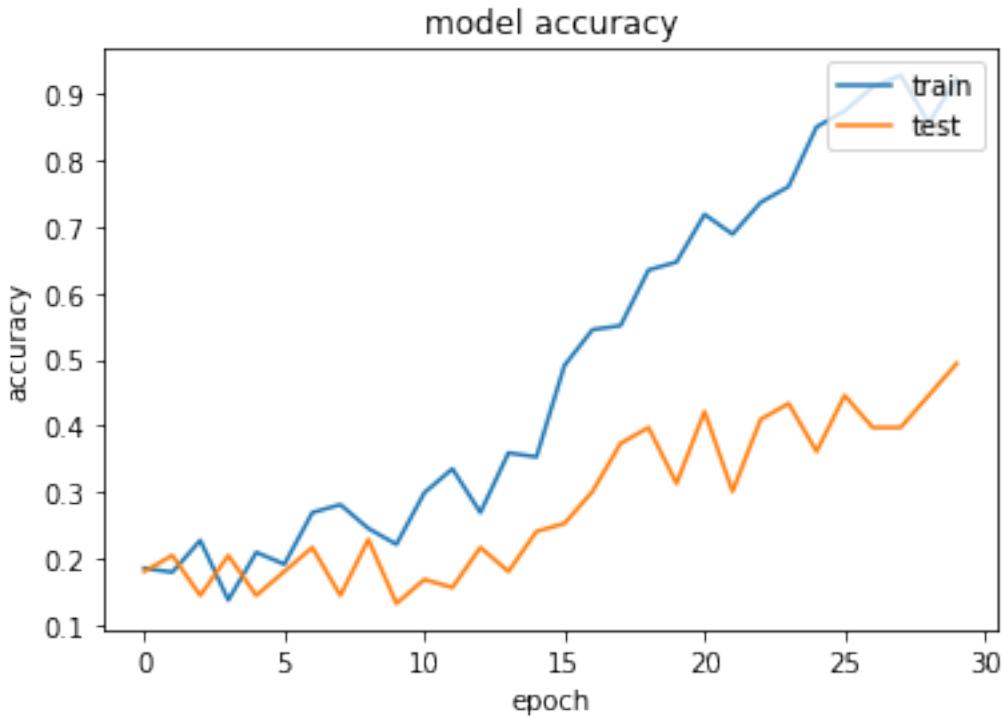
Epoch 11/30
21/21 [=====] - 0s 12ms/step - loss: 1.5335 - acc:
0.3249 - val_loss: 1.6898 - val_acc: 0.1687

Epoch 12/30
21/21 [=====] - 0s 11ms/step - loss: 1.5573 - acc:
0.3490 - val_loss: 1.8730 - val_acc: 0.1566

Epoch 13/30
21/21 [=====] - 0s 11ms/step - loss: 1.5437 - acc:
0.2350 - val_loss: 1.6643 - val_acc: 0.2169

Epoch 14/30
21/21 [=====] - 0s 11ms/step - loss: 1.4423 - acc:
0.3336 - val_loss: 1.8902 - val_acc: 0.1807

Epoch 15/30
21/21 [=====] - 0s 11ms/step - loss: 1.4640 - acc:
0.3245 - val_loss: 1.6163 - val_acc: 0.2410
Epoch 16/30
21/21 [=====] - 0s 11ms/step - loss: 1.1992 - acc:
0.5356 - val_loss: 1.5559 - val_acc: 0.2530
Epoch 17/30
21/21 [=====] - 0s 11ms/step - loss: 1.2053 - acc:
0.4788 - val_loss: 1.7071 - val_acc: 0.3012
Epoch 18/30
21/21 [=====] - 0s 11ms/step - loss: 1.1409 - acc:
0.5041 - val_loss: 1.8082 - val_acc: 0.3735
Epoch 19/30
21/21 [=====] - 0s 11ms/step - loss: 0.9573 - acc:
0.6334 - val_loss: 1.5922 - val_acc: 0.3976
Epoch 20/30
21/21 [=====] - 0s 11ms/step - loss: 0.9018 - acc:
0.6316 - val_loss: 1.7287 - val_acc: 0.3133
Epoch 21/30
21/21 [=====] - 0s 11ms/step - loss: 0.7440 - acc:
0.7372 - val_loss: 1.6076 - val_acc: 0.4217
Epoch 22/30
21/21 [=====] - 0s 11ms/step - loss: 0.7490 - acc:
0.7118 - val_loss: 1.6433 - val_acc: 0.3012
Epoch 23/30
21/21 [=====] - 0s 11ms/step - loss: 0.7895 - acc:
0.7116 - val_loss: 1.8753 - val_acc: 0.4096
Epoch 24/30
21/21 [=====] - 0s 11ms/step - loss: 0.6827 - acc:
0.7403 - val_loss: 1.6575 - val_acc: 0.4337
Epoch 25/30
21/21 [=====] - 0s 11ms/step - loss: 0.4071 - acc:
0.8963 - val_loss: 2.3683 - val_acc: 0.3614
Epoch 26/30
21/21 [=====] - 0s 11ms/step - loss: 0.3669 - acc:
0.8794 - val_loss: 2.0762 - val_acc: 0.4458
Epoch 27/30
21/21 [=====] - 0s 11ms/step - loss: 0.2821 - acc:
0.9113 - val_loss: 2.9019 - val_acc: 0.3976
Epoch 28/30
21/21 [=====] - 0s 11ms/step - loss: 0.1988 - acc:
0.9441 - val_loss: 2.7714 - val_acc: 0.3976
Epoch 29/30
21/21 [=====] - 0s 11ms/step - loss: 0.5035 - acc:
0.8244 - val_loss: 2.0897 - val_acc: 0.4458
Epoch 30/30
21/21 [=====] - 0s 11ms/step - loss: 0.2204 - acc:
0.9247 - val_loss: 2.7968 - val_acc: 0.4940



Investigation 2: Added more dense layers, again, worse performance

```
[4]: modelC = Sequential()

modelC.add(Conv2D(4, (4, 4), strides=1, padding="valid", activation='relu',  
    ↪data_format='channels_first'))  
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  
    ↪data_format='channels_first'))  
  
modelC.add(Conv2D(8, (4, 4), strides=1, padding="valid", activation='relu',  
    ↪data_format='channels_first'))  
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  
    ↪data_format='channels_first'))  
  
modelC.add(Conv2D(32, (4, 4), strides=1, padding="valid", activation='relu',  
    ↪data_format='channels_first'))  
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid",  
    ↪data_format='channels_first'))  
  
modelC.add(Conv2D(64, (4, 4), strides=1, padding="valid", activation='relu',  
    ↪data_format='channels_first'))
```

```

modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(128, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Conv2D(256, (4, 4), strides=1, padding="valid", activation='relu', ↴
    ↪data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↪data_format='channels_first'))

modelC.add(Flatten())

modelC.add(Dense(64, activation='relu'))
modelC.add(Dense(128, activation='relu'))
modelC.add(Dense(256, activation='relu'))
modelC.add(Dense(512, activation='relu'))
modelC.add(Dense(1024, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam', ↴
    ↪metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=30, batch_size=8, ↴
    ↪verbose=1)

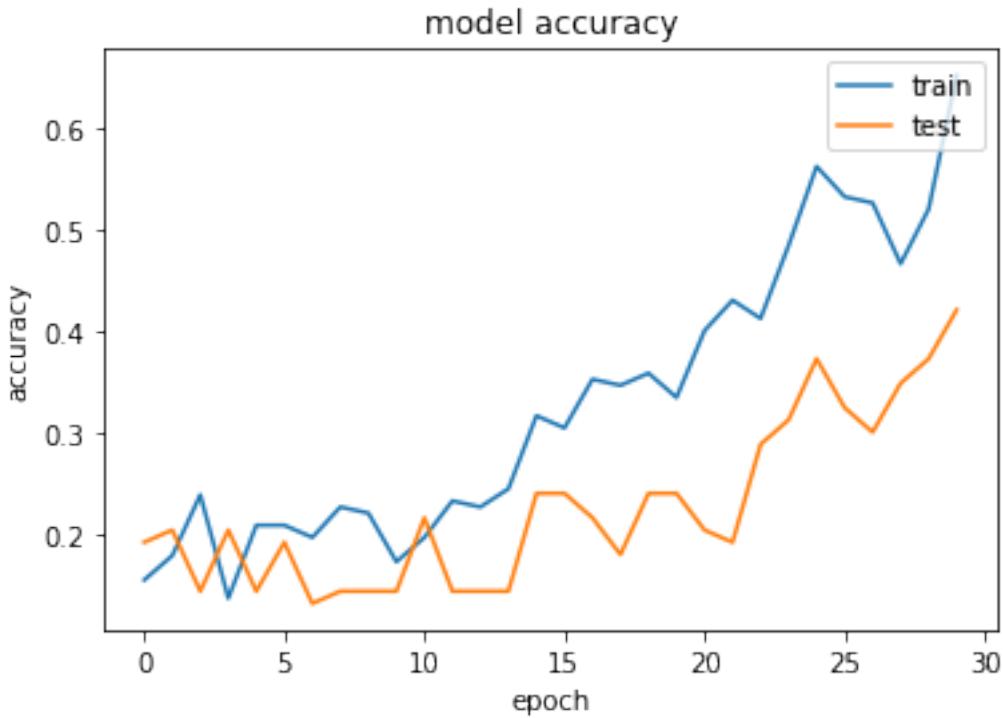
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

Epoch 1/30
21/21 [=====] - 1s 26ms/step - loss: 1.8113 - acc: 0.1472 - val_loss: 1.6807 - val_acc: 0.1928
Epoch 2/30
21/21 [=====] - 0s 14ms/step - loss: 1.6747 - acc: 0.1985 - val_loss: 1.6398 - val_acc: 0.2048
Epoch 3/30
21/21 [=====] - 0s 13ms/step - loss: 1.6397 - acc:

```
0.2182 - val_loss: 1.6564 - val_acc: 0.1446
Epoch 4/30
21/21 [=====] - 0s 12ms/step - loss: 1.6157 - acc:
0.1642 - val_loss: 1.6432 - val_acc: 0.2048
Epoch 5/30
21/21 [=====] - 0s 11ms/step - loss: 1.6180 - acc:
0.1897 - val_loss: 1.6407 - val_acc: 0.1446
Epoch 6/30
21/21 [=====] - 0s 12ms/step - loss: 1.6119 - acc:
0.2286 - val_loss: 1.6461 - val_acc: 0.1928
Epoch 7/30
21/21 [=====] - 0s 11ms/step - loss: 1.6126 - acc:
0.2134 - val_loss: 1.6317 - val_acc: 0.1325
Epoch 8/30
21/21 [=====] - 0s 11ms/step - loss: 1.6086 - acc:
0.2361 - val_loss: 1.6437 - val_acc: 0.1446
Epoch 9/30
21/21 [=====] - 0s 11ms/step - loss: 1.5984 - acc:
0.2482 - val_loss: 1.6383 - val_acc: 0.1446
Epoch 10/30
21/21 [=====] - 0s 11ms/step - loss: 1.6061 - acc:
0.1613 - val_loss: 1.6324 - val_acc: 0.1446
Epoch 11/30
21/21 [=====] - 0s 11ms/step - loss: 1.6092 - acc:
0.2319 - val_loss: 1.6352 - val_acc: 0.2169
Epoch 12/30
21/21 [=====] - 0s 11ms/step - loss: 1.6096 - acc:
0.2589 - val_loss: 1.6500 - val_acc: 0.1446
Epoch 13/30
21/21 [=====] - 0s 11ms/step - loss: 1.6343 - acc:
0.1851 - val_loss: 1.6364 - val_acc: 0.1446
Epoch 14/30
21/21 [=====] - 0s 11ms/step - loss: 1.5967 - acc:
0.2361 - val_loss: 2.1588 - val_acc: 0.1446
Epoch 15/30
21/21 [=====] - 0s 11ms/step - loss: 1.6388 - acc:
0.2601 - val_loss: 1.5995 - val_acc: 0.2410
Epoch 16/30
21/21 [=====] - 0s 11ms/step - loss: 1.5229 - acc:
0.3685 - val_loss: 1.5911 - val_acc: 0.2410
Epoch 17/30
21/21 [=====] - 0s 11ms/step - loss: 1.5275 - acc:
0.3376 - val_loss: 1.6106 - val_acc: 0.2169
Epoch 18/30
21/21 [=====] - 0s 11ms/step - loss: 1.4864 - acc:
0.3031 - val_loss: 1.6693 - val_acc: 0.1807
Epoch 19/30
21/21 [=====] - 0s 11ms/step - loss: 1.4008 - acc:
```

```
0.3467 - val_loss: 1.6131 - val_acc: 0.2410
Epoch 20/30
21/21 [=====] - 0s 11ms/step - loss: 1.4776 - acc:
0.2869 - val_loss: 1.6073 - val_acc: 0.2410
Epoch 21/30
21/21 [=====] - 0s 11ms/step - loss: 1.4007 - acc:
0.3856 - val_loss: 1.5407 - val_acc: 0.2048
Epoch 22/30
21/21 [=====] - 0s 12ms/step - loss: 1.2032 - acc:
0.4562 - val_loss: 1.5640 - val_acc: 0.1928
Epoch 23/30
21/21 [=====] - 0s 12ms/step - loss: 1.2779 - acc:
0.3767 - val_loss: 1.6079 - val_acc: 0.2892
Epoch 24/30
21/21 [=====] - 0s 11ms/step - loss: 1.1768 - acc:
0.4534 - val_loss: 1.5499 - val_acc: 0.3133
Epoch 25/30
21/21 [=====] - 0s 11ms/step - loss: 0.9508 - acc:
0.6612 - val_loss: 1.6468 - val_acc: 0.3735
Epoch 26/30
21/21 [=====] - 0s 11ms/step - loss: 0.9788 - acc:
0.5849 - val_loss: 1.5795 - val_acc: 0.3253
Epoch 27/30
21/21 [=====] - 0s 11ms/step - loss: 1.0229 - acc:
0.5360 - val_loss: 1.7310 - val_acc: 0.3012
Epoch 28/30
21/21 [=====] - 0s 11ms/step - loss: 1.0763 - acc:
0.5249 - val_loss: 1.5134 - val_acc: 0.3494
Epoch 29/30
21/21 [=====] - 0s 11ms/step - loss: 1.0443 - acc:
0.4663 - val_loss: 1.4915 - val_acc: 0.3735
Epoch 30/30
21/21 [=====] - 0s 11ms/step - loss: 0.8670 - acc:
0.6937 - val_loss: 1.7153 - val_acc: 0.4217
```



Adding more dense layers seems to worsen the performance of my model. This is particularly strange as I would've thought it would have improved our accuracy.

9 Unseen Data

To grab unseen data for the model, I used my The Simpsons DVDs to grab frames from the following The Simpsons episodes: **'The Itchy & Scratchy & Poochie Show'**, **'Rosebud'**, **'Who Shot Mr Burns Part 1'** & **'Marge in Chains'**

This allowed me to capture frames using VLC Media Player where The Simpsons' characters, particular the titular family, are in focus, similar to the data the model has been trained with.

FINAL MODEL USAGE TO PREDICT UNSEEN DATA

```
[ ]: modelC = Sequential()
modelC.add(Conv2D(4, (3, 3), strides=1, padding="valid", input_shape=(1, 256, 256), activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))

modelC.add(Conv2D(8, (3, 3), strides=1, padding="valid", activation='relu', data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", data_format='channels_first'))
```

```

modelC.add(Conv2D(32, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(64, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(128, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Conv2D(256, (3, 3), strides=1, padding="valid", activation='relu', ↴
    ↴data_format='channels_first'))
modelC.add(MaxPooling2D(pool_size=(2, 2), padding="valid", ↴
    ↴data_format='channels_first'))

modelC.add(Flatten())
modelC.add(Dense(512, activation='relu'))
modelC.add(Dense(num_classes, activation='softmax'))

modelC.compile(loss='categorical_crossentropy', optimizer='adam', ↴
    ↴metrics=['acc'])

# Fit the model
history = modelC.fit(X, Y, validation_split=0.33, epochs=20, batch_size=50, ↴
    ↴verbose=1)

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

```

```

Epoch 1/20
4/4 [=====] - 1s 98ms/step - loss: 1.7592 - acc: 0.2220
- val_loss: 1.6979 - val_acc: 0.1446
Epoch 2/20
4/4 [=====] - 0s 46ms/step - loss: 1.6332 - acc: 0.2630

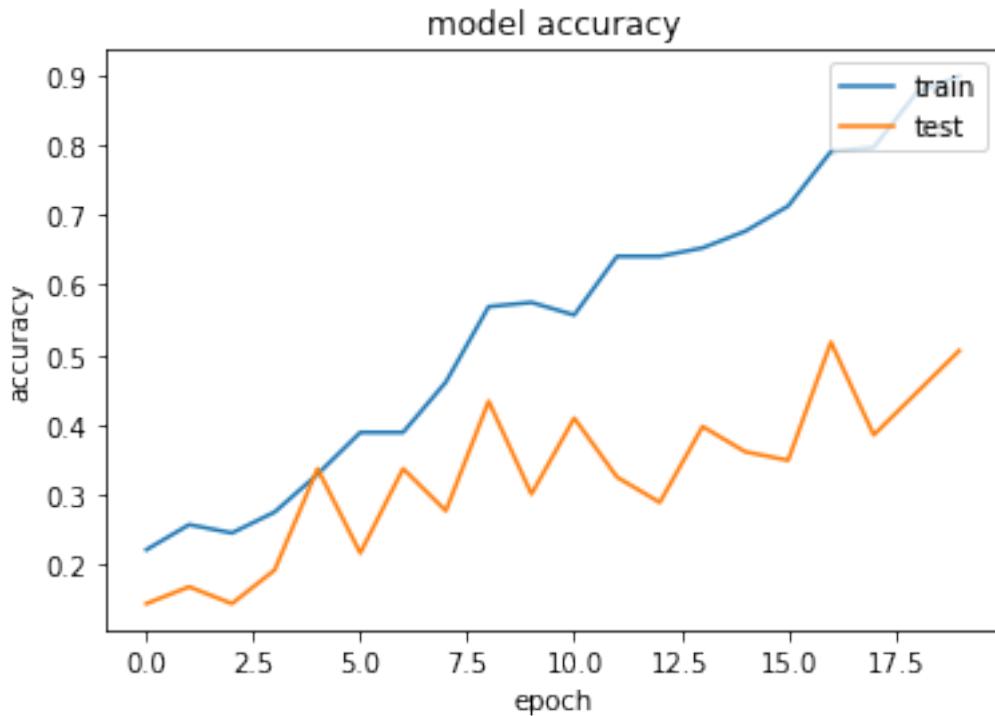
```

```
- val_loss: 1.6011 - val_acc: 0.1687
Epoch 3/20
4/4 [=====] - 0s 42ms/step - loss: 1.5877 - acc: 0.2429
- val_loss: 1.6895 - val_acc: 0.1446
Epoch 4/20
4/4 [=====] - 0s 42ms/step - loss: 1.5908 - acc: 0.2408
- val_loss: 1.6382 - val_acc: 0.1928
Epoch 5/20
4/4 [=====] - 0s 38ms/step - loss: 1.5061 - acc: 0.3311
- val_loss: 1.5363 - val_acc: 0.3373
Epoch 6/20
4/4 [=====] - 0s 38ms/step - loss: 1.4275 - acc: 0.4224
- val_loss: 1.6035 - val_acc: 0.2169
Epoch 7/20
4/4 [=====] - 0s 38ms/step - loss: 1.3430 - acc: 0.4124
- val_loss: 1.5043 - val_acc: 0.3373
Epoch 8/20
4/4 [=====] - 0s 38ms/step - loss: 1.2984 - acc: 0.4451
- val_loss: 1.5920 - val_acc: 0.2771
Epoch 9/20
4/4 [=====] - 0s 38ms/step - loss: 1.1719 - acc: 0.5875
- val_loss: 1.5467 - val_acc: 0.4337
Epoch 10/20
4/4 [=====] - 0s 37ms/step - loss: 1.1247 - acc: 0.5833
- val_loss: 1.6169 - val_acc: 0.3012
Epoch 11/20
4/4 [=====] - 0s 36ms/step - loss: 1.0923 - acc: 0.6074
- val_loss: 1.4458 - val_acc: 0.4096
Epoch 12/20
4/4 [=====] - 0s 35ms/step - loss: 1.0316 - acc: 0.6396
- val_loss: 1.6636 - val_acc: 0.3253
Epoch 13/20
4/4 [=====] - 0s 36ms/step - loss: 0.9437 - acc: 0.6423
- val_loss: 1.9154 - val_acc: 0.2892
Epoch 14/20
4/4 [=====] - 0s 36ms/step - loss: 0.9944 - acc: 0.5991
- val_loss: 1.8513 - val_acc: 0.3976
Epoch 15/20
4/4 [=====] - 0s 38ms/step - loss: 0.9295 - acc: 0.6560
- val_loss: 1.6495 - val_acc: 0.3614
Epoch 16/20
4/4 [=====] - 0s 38ms/step - loss: 0.7182 - acc: 0.7297
- val_loss: 1.7223 - val_acc: 0.3494
Epoch 17/20
4/4 [=====] - 0s 40ms/step - loss: 0.6253 - acc: 0.7902
- val_loss: 1.6947 - val_acc: 0.5181
Epoch 18/20
4/4 [=====] - 0s 38ms/step - loss: 0.5714 - acc: 0.7806
```

```

- val_loss: 1.8602 - val_acc: 0.3855
Epoch 19/20
4/4 [=====] - 0s 37ms/step - loss: 0.4620 - acc: 0.8750
- val_loss: 1.8403 - val_acc: 0.4458
Epoch 20/20
4/4 [=====] - 0s 36ms/step - loss: 0.2842 - acc: 0.9193
- val_loss: 2.0012 - val_acc: 0.5060

```

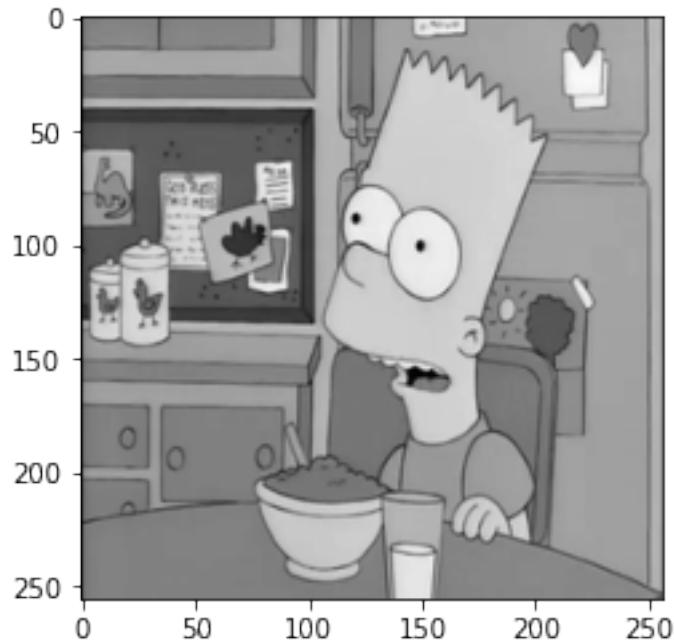


Bart

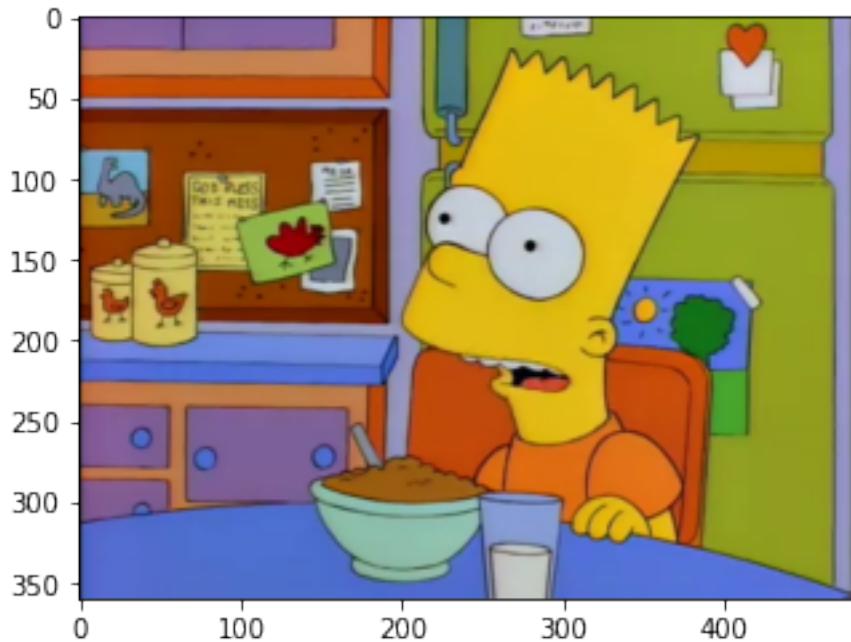
```
[21]: bart1 = Image.open("unseenData/Bart/1.png")
plt.imshow(bart1)
plt.show()
grayscaleBart1 = bart1.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleBart1, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleBart1))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
```

```
result = modelC.predict_classes(Xt)
plt.imshow(bart1)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





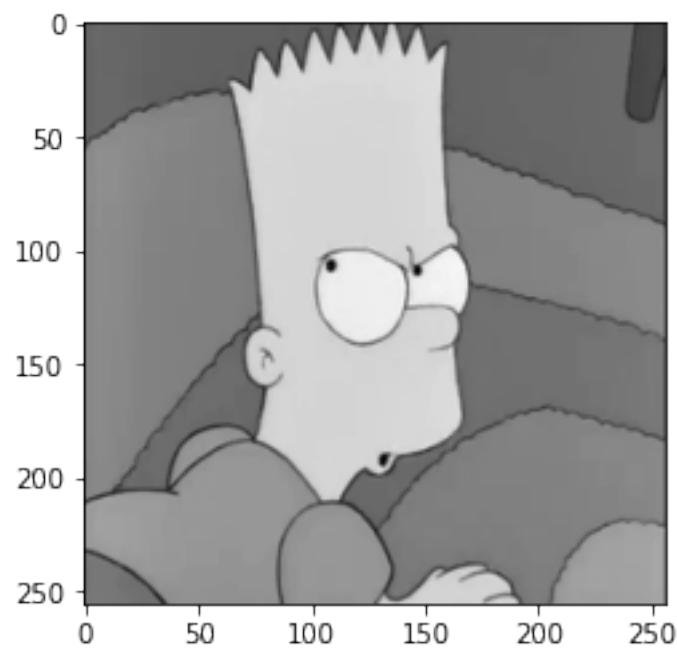
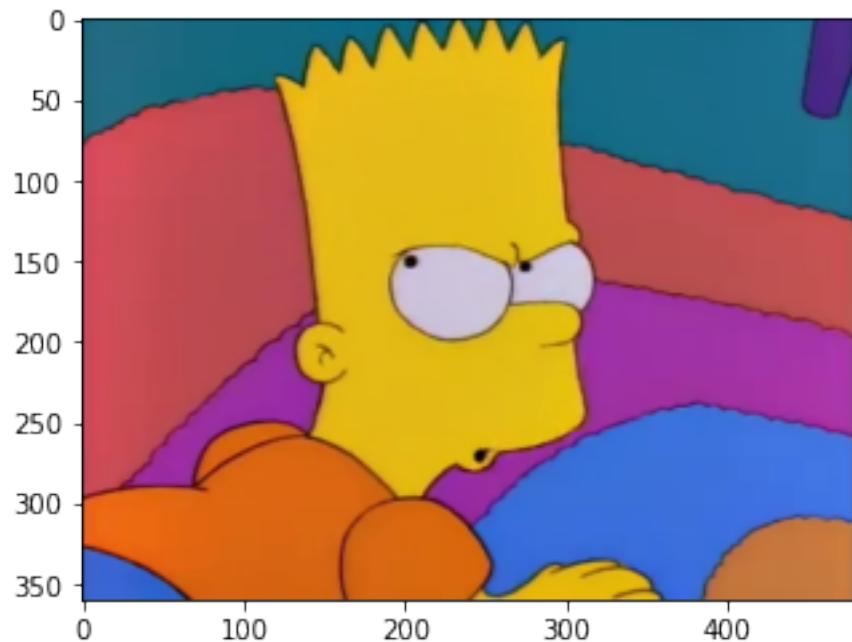
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



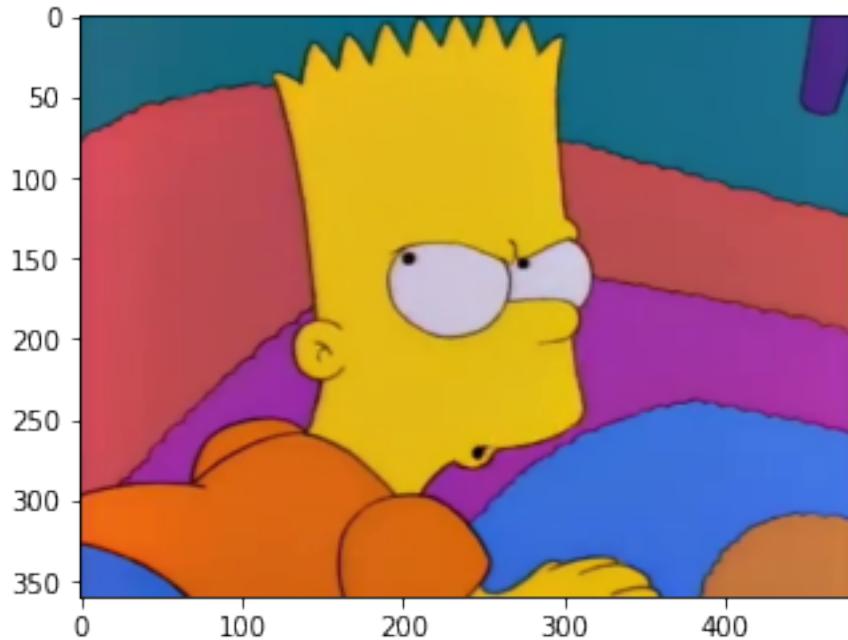
Bart

```
[23]: bart2 = Image.open("unseenData/Bart/2.png")
plt.imshow(bart2)
plt.show()
grayscaleBart2 = bart2.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleBart2, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleBart2))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(bart2)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



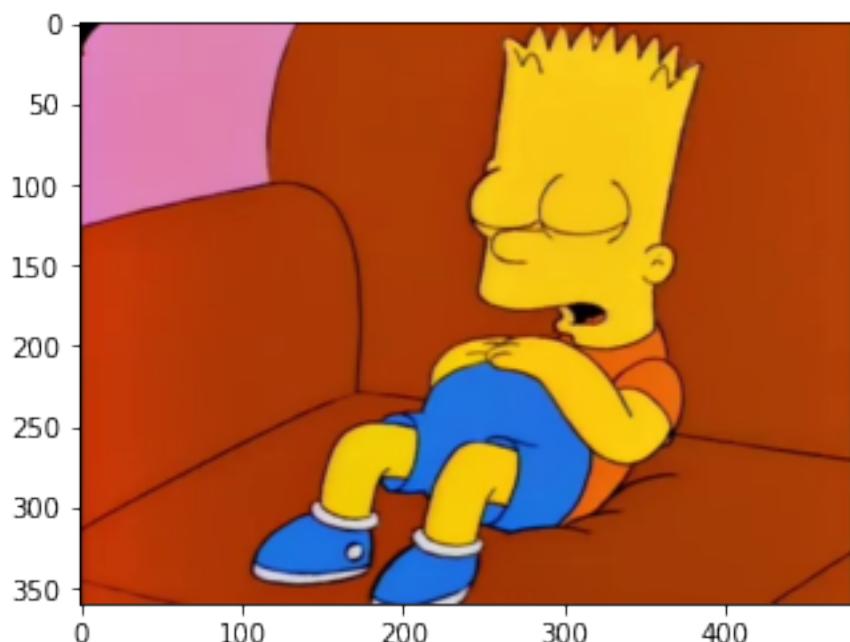
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```

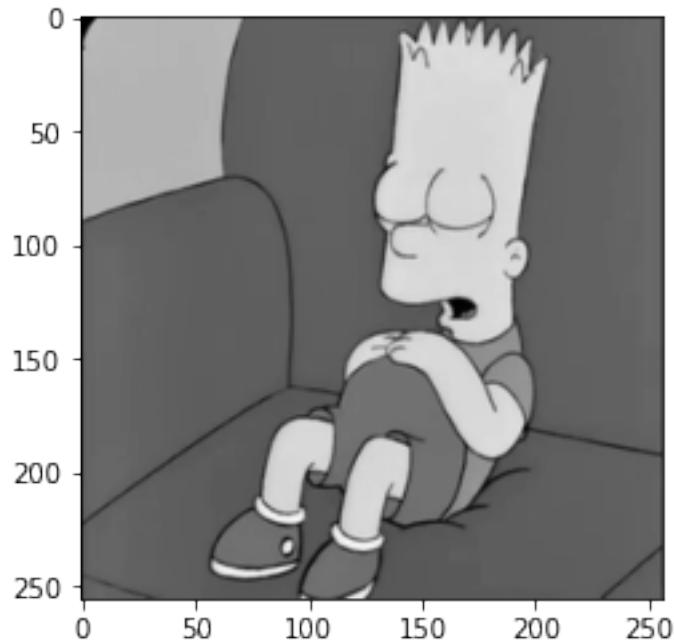


Marge

```
[24]: bart3 = Image.open("unseenData/Bart/3.png")
plt.imshow(bart3)
plt.show()
grayscaleBart3 = bart3.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleBart3, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleBart3))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
```

```
plt.imshow(bart3)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





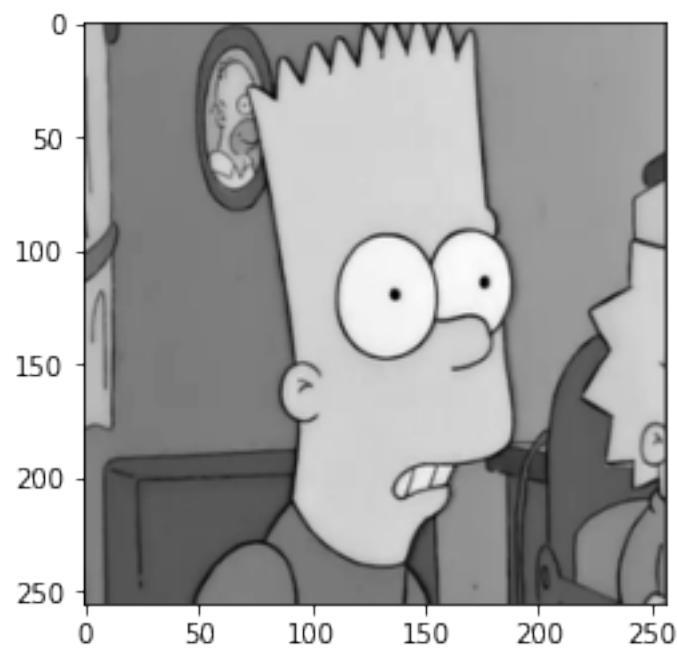
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



Bart

```
[27]: bart4 = Image.open("unseenData/Bart/4.png")
plt.imshow(bart4)
plt.show()
grayscaleBart4 = bart4.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleBart4, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleBart4))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(bart4)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```

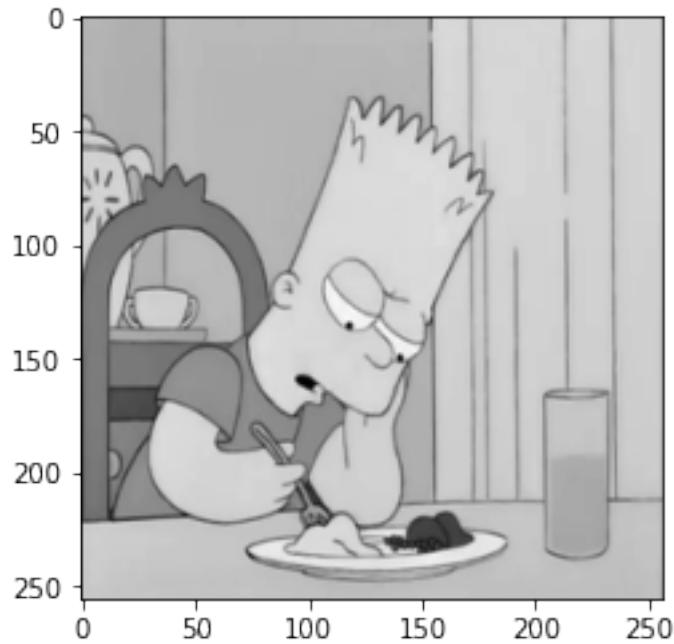


Marge

```
[29]: bart5 = Image.open("unseenData/Bart/5.png")
plt.imshow(bart5)
plt.show()
grayscaleBart5 = bart5.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleBart5, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleBart5))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
```

```
plt.imshow(bart5)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



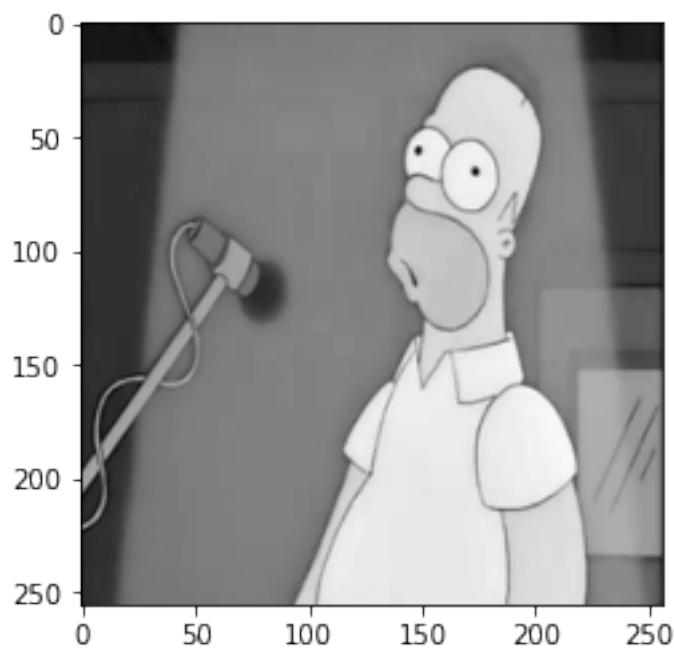
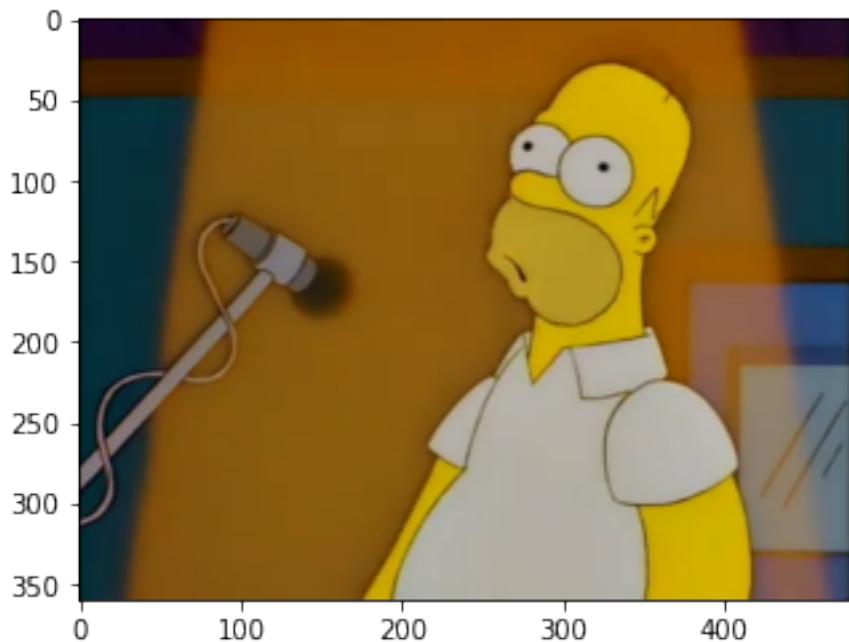
Bart

3/5 predictions Bart. Not bad!

Homer

```
[31]: homer1 = Image.open("unseenData/Homer/1.png")
plt.imshow(homer1)
plt.show()
grayscaleHomer1 = homer1.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleHomer1, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleHomer1))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(homer1)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
```

```
elif result[0] == 3:  
    print("Lisa")  
elif result[0] == 4:  
    print("Maggie")  
elif result[0] == 5:  
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```

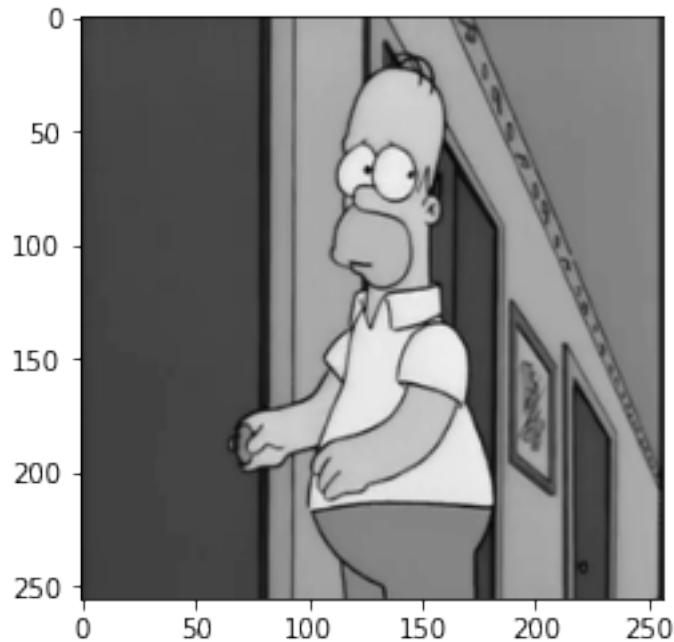


Homer

```
[32]: homer2 = Image.open("unseenData/Homer/2.png")
plt.imshow(homer2)
plt.show()
grayscaleHomer2 = homer2.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleHomer2, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleHomer2))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
```

```
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(homer2)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





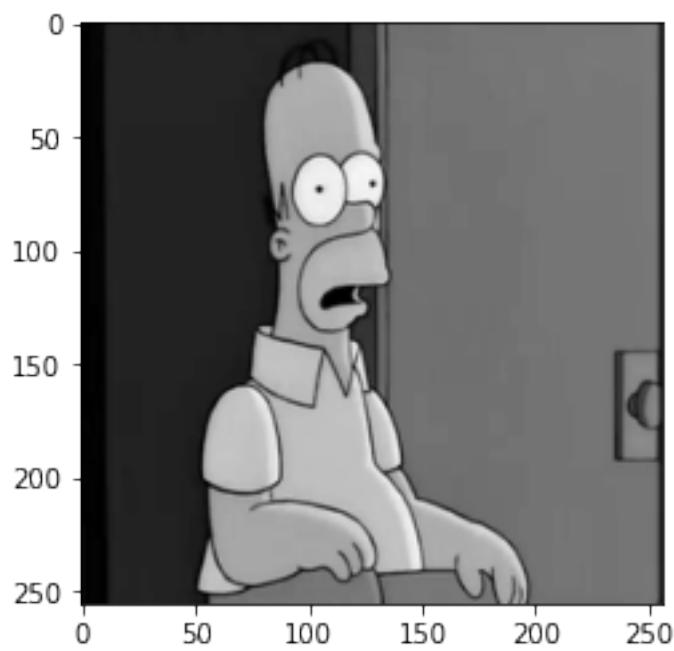
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



Homer

```
[34]: homer3 = Image.open("unseenData/Homer/3.png")
plt.imshow(homer3)
plt.show()
grayscaleHomer3 = homer3.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleHomer3, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleHomer3))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(homer3)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```

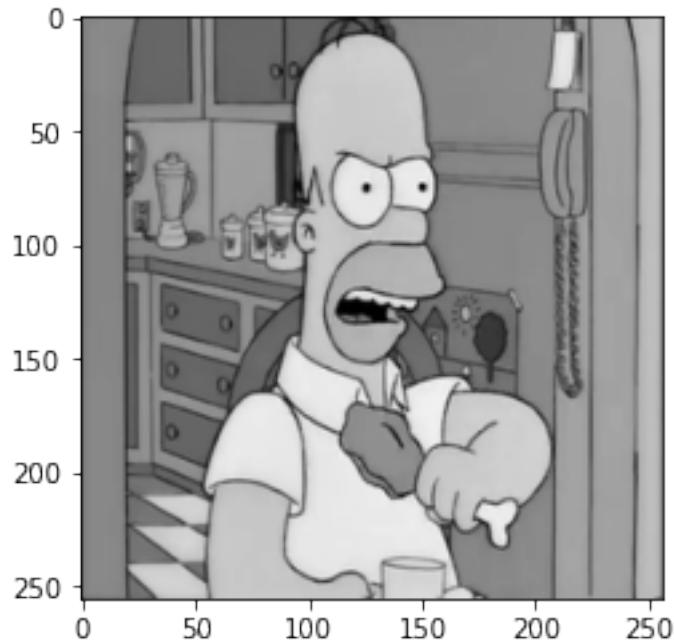


Marge

```
[37]: homer4 = Image.open("unseenData/Homer/4.png")
plt.imshow(homer4)
plt.show()
grayscaleHomer4 = homer4.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleHomer4, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleHomer4))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
```

```
plt.imshow(homer4)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





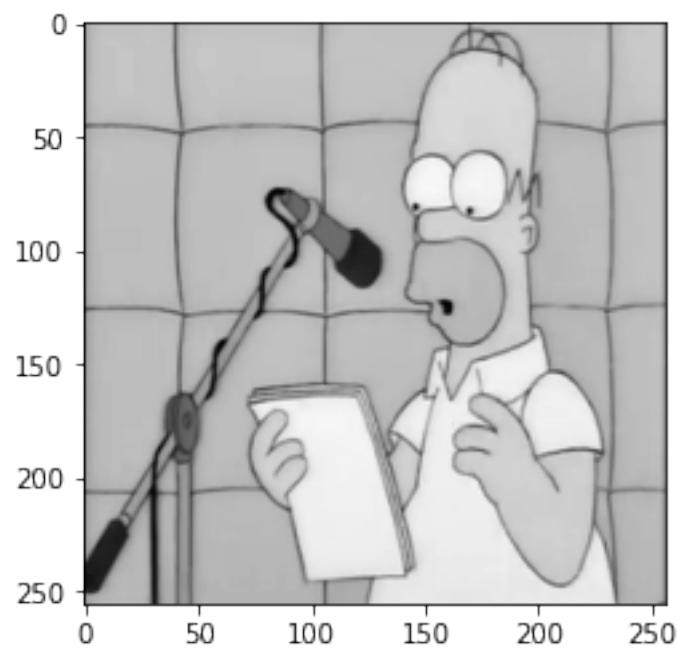
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



Homer

```
[38]: homer5 = Image.open("unseenData/Homer/5.png")
plt.imshow(homer5)
plt.show()
grayscaleHomer5 = homer5.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleHomer5, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleHomer5))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(homer5)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```



Maggie

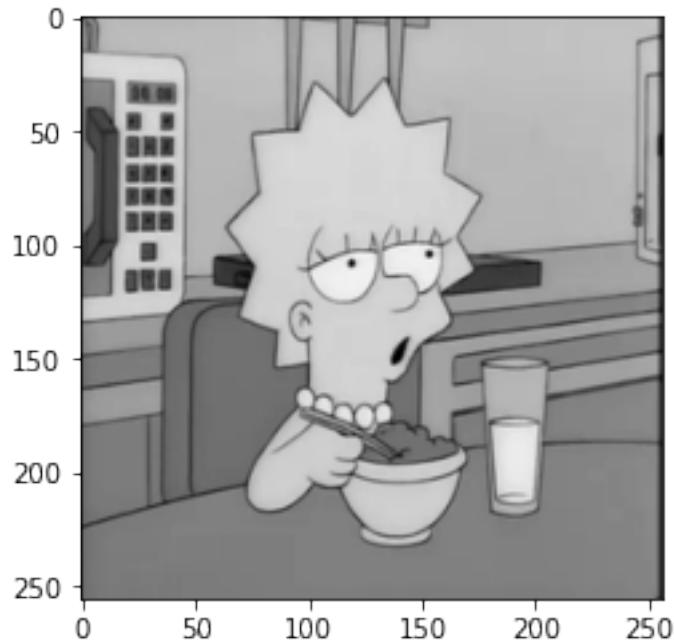
3/5 for Homer.

Lisa

```
[40]: lisa1 = Image.open("unseenData/Lisa/1.png")
plt.imshow(lisa1)
plt.show()
grayscaleLisa1 = lisa1.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleLisa1, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleLisa1))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
```

```
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(lisa1)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





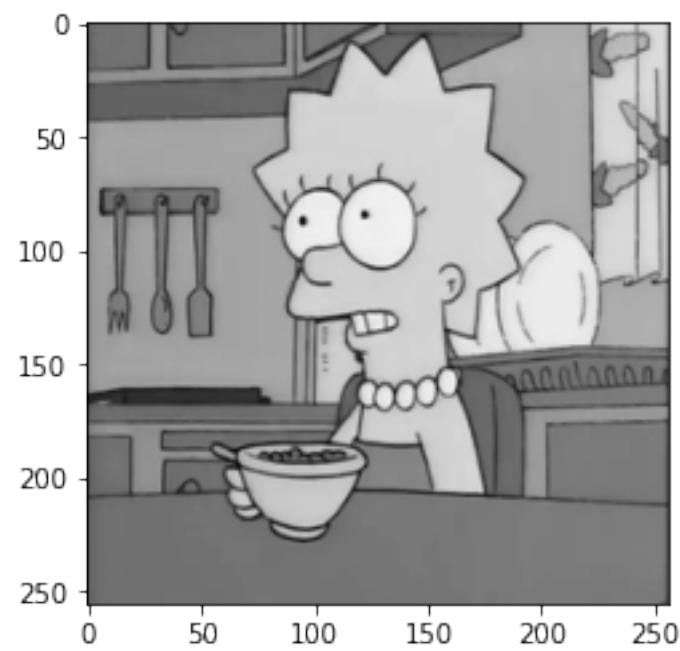
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



Lisa

```
[41]: lisa2 = Image.open("unseenData/Lisa/2.png")
plt.imshow(lisa2)
plt.show()
grayscaleLisa2 = lisa2.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleLisa2, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleLisa2))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(lisa2)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```

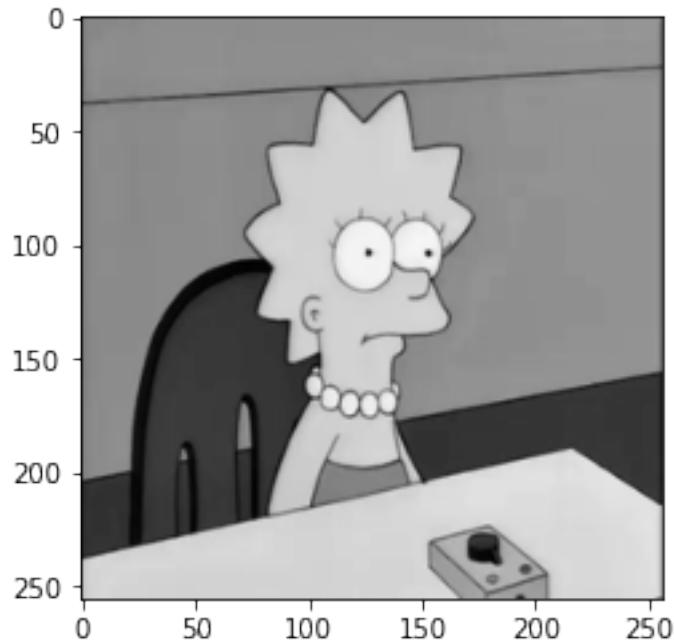


Lisa

```
[42]: lisa3 = Image.open("unseenData/Lisa/3.png")
plt.imshow(lisa3)
plt.show()
grayscaleLisa3 = lisa3.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleLisa3, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleLisa3))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
```

```
plt.imshow(lisa3)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





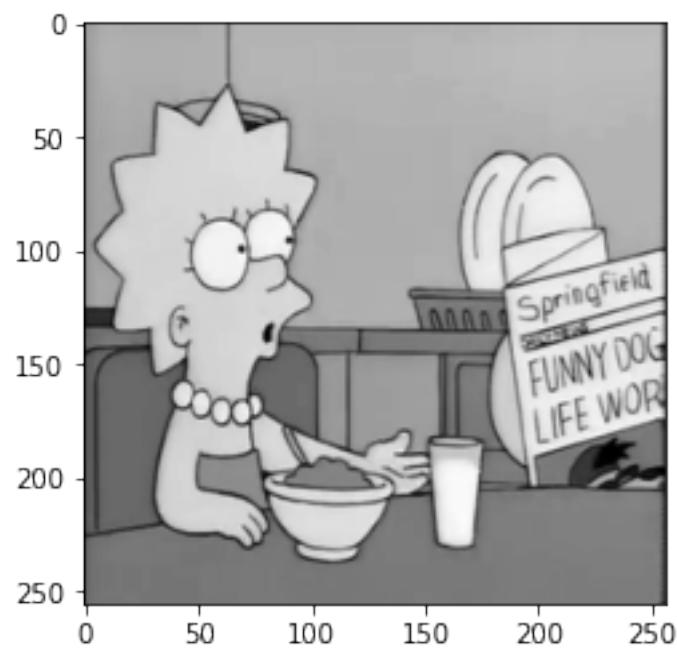
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



Homer

```
[43]: lisa4 = Image.open("unseenData/Lisa/4.png")
plt.imshow(lisa4)
plt.show()
grayscaleLisa4 = lisa4.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleLisa4, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleLisa4))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(lisa4)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```

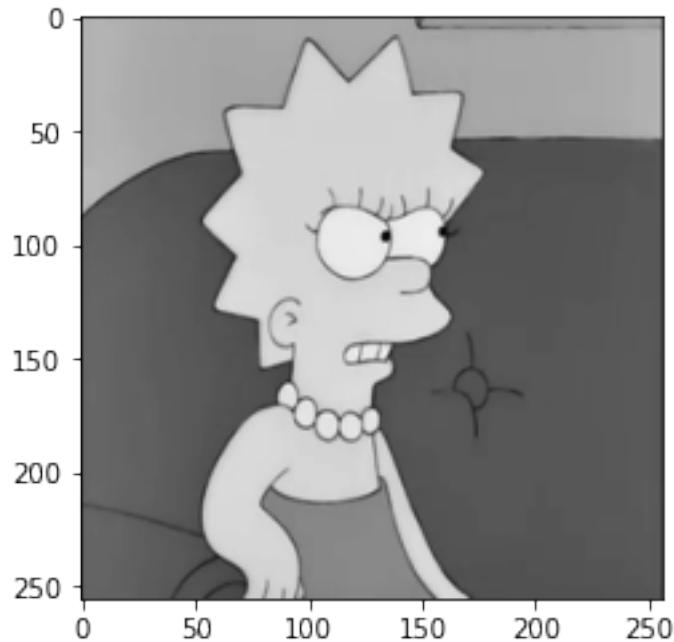


Lisa

```
[44]: lisa5 = Image.open("unseenData/Lisa/5.png")
plt.imshow(lisa5)
plt.show()
grayscaleLisa5 = lisa5.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleLisa5, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleLisa5))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
```

```
plt.imshow(lisa5)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



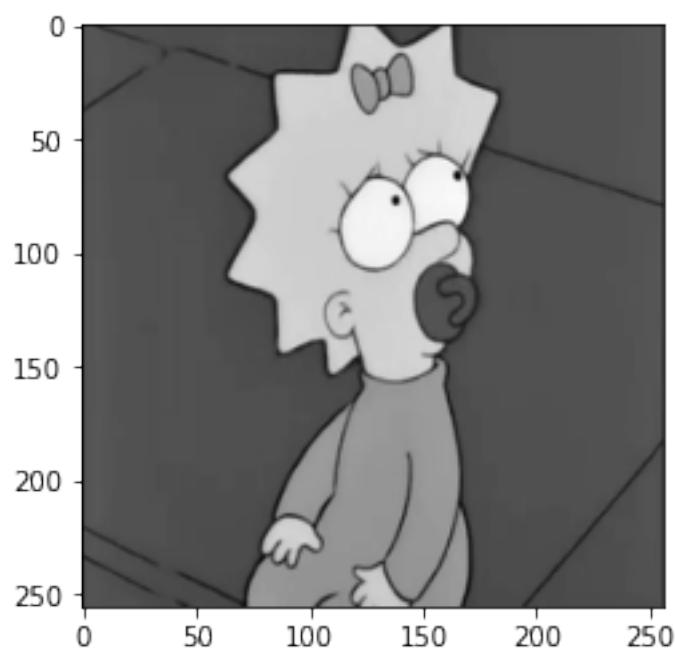
Lisa

4/5 for Lisa!

Maggie

```
[45]: maggie1 = Image.open("unseenData/Maggie/1.png")
plt.imshow(maggie1)
plt.show()
grayscaleMaggie1 = maggie1.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMaggie1, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMaggie1))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(maggie1)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
```

```
elif result[0] == 3:  
    print("Lisa")  
elif result[0] == 4:  
    print("Maggie")  
elif result[0] == 5:  
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```

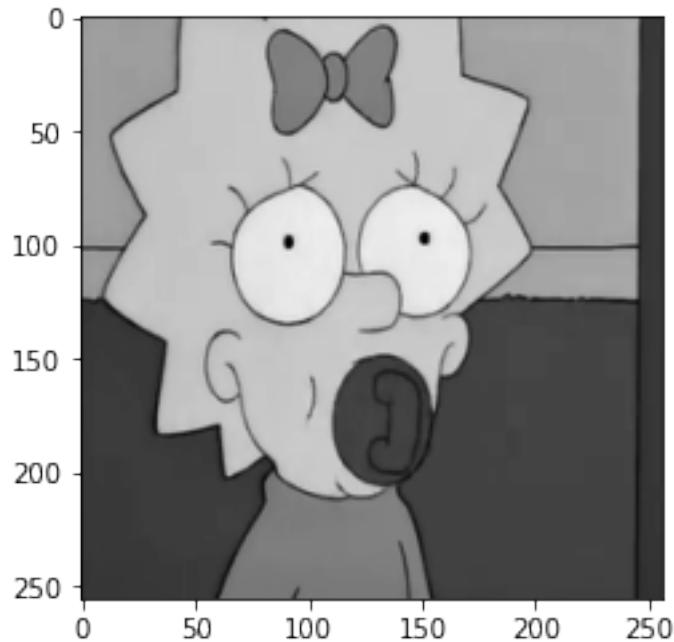


Marge

```
[46]: maggie2 = Image.open("unseenData/Maggie/2.png")
plt.imshow(maggie2)
plt.show()
grayscaleMaggie2 = maggie2.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMaggie2, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMaggie2))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
```

```
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(maggie2)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





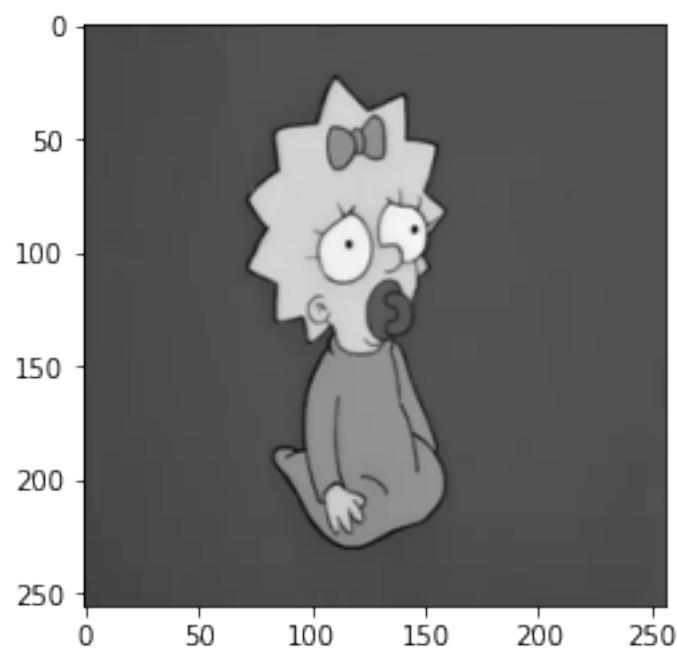
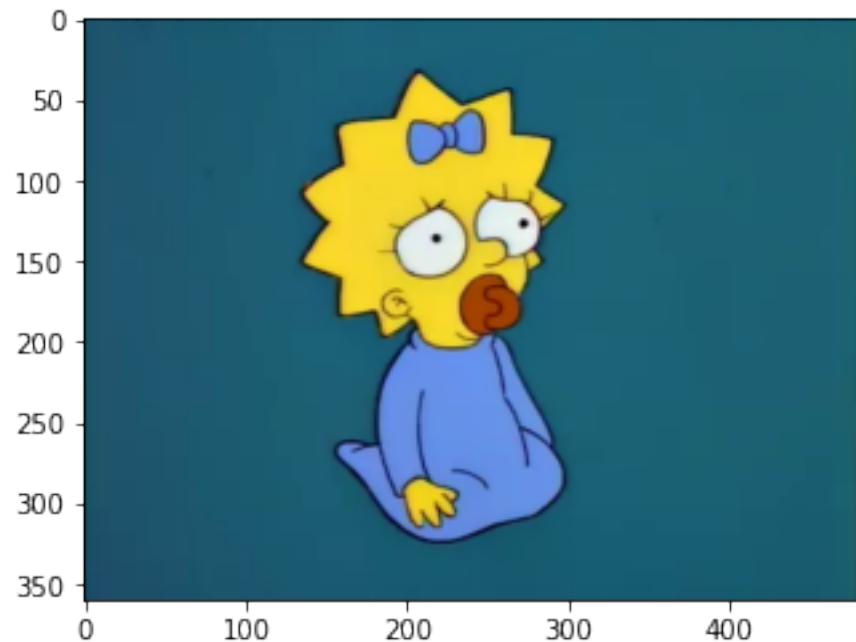
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)`, if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")`, if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



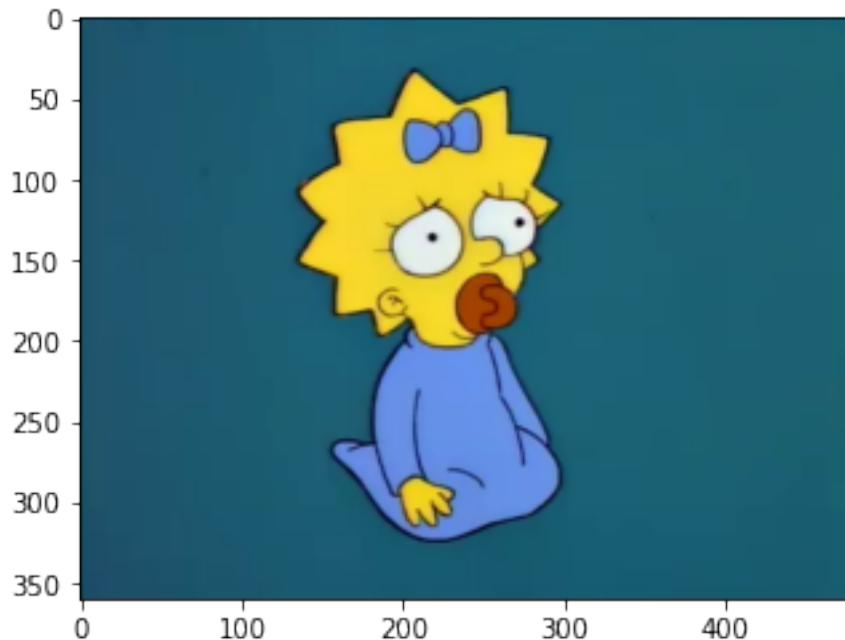
Lisa

```
[47]: maggie3 = Image.open("unseenData/Maggie/3.png")
plt.imshow(maggie3)
plt.show()
grayscaleMaggie3 = maggie3.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMaggie3, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMaggie3))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(maggie3)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```

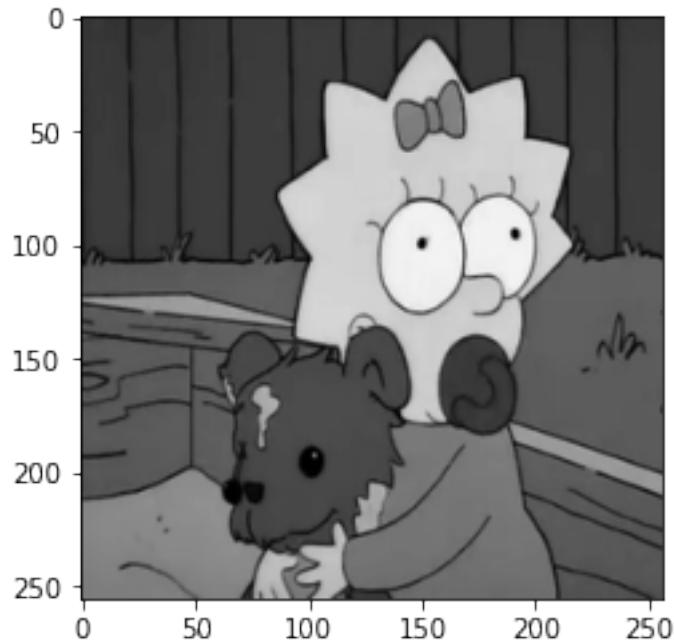


Marge

```
[48]: maggie4 = Image.open("unseenData/Maggie/4.png")
plt.imshow(maggie4)
plt.show()
grayscaleMaggie4 = maggie4.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMaggie4, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMaggie4))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
```

```
plt.imshow(maggie4)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





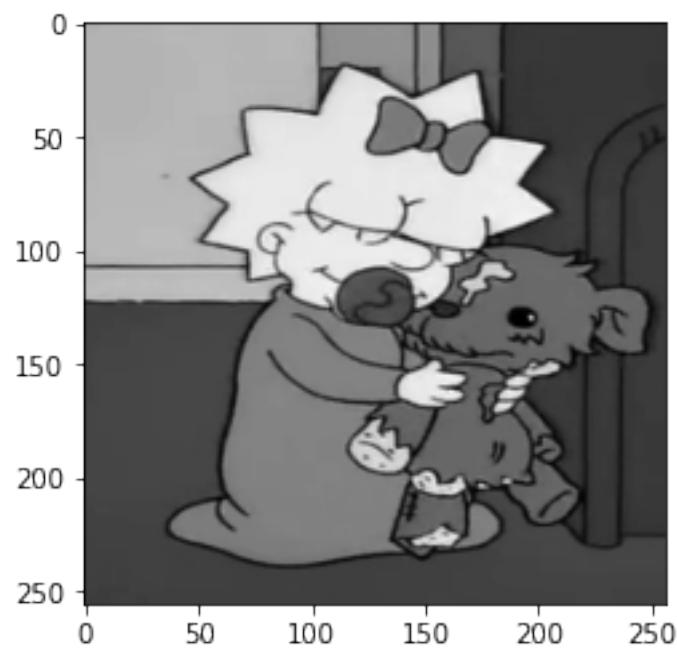
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



Maggie

```
[50]: maggie5 = Image.open("unseenData/Maggie/5.png")
plt.imshow(maggie5)
plt.show()
grayscaleMaggie5 = maggie5.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMaggie5, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMaggie5))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(maggie5)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```



Lisa

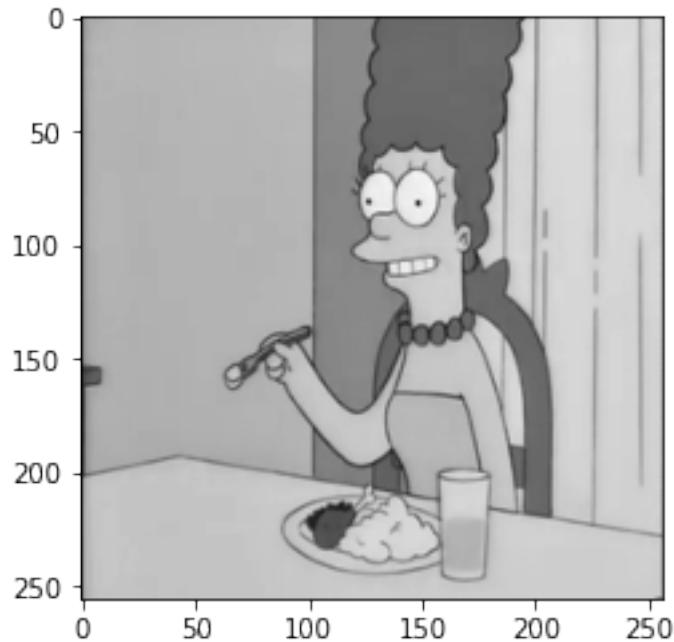
1/5 for Maggie, damn. Unfortunate.

Marge

```
[51]: marge1 = Image.open("unseenData/Marge/1.png")
plt.imshow(marge1)
plt.show()
grayscaleMarge1 = marge1.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMarge1, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMarge1))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
```

```
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(marge1)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





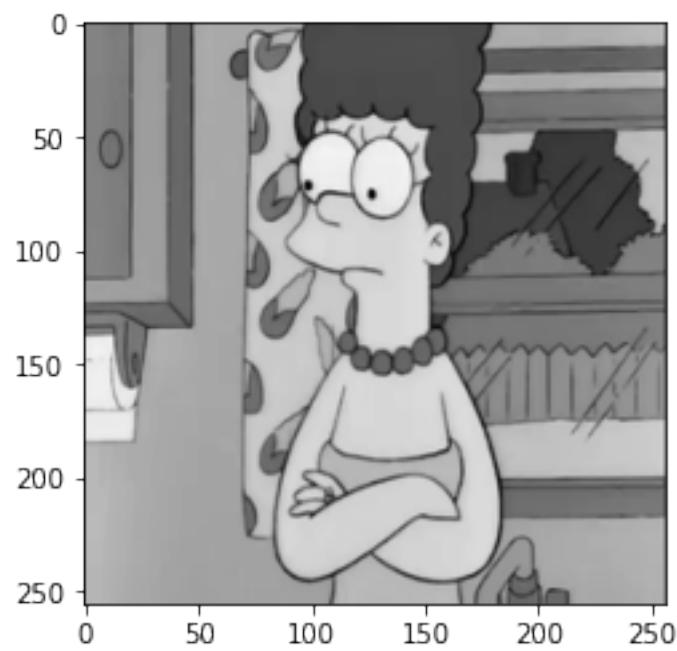
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



Marge

```
[52]: marge2 = Image.open("unseenData/Marge/2.png")
plt.imshow(marge2)
plt.show()
grayscaleMarge2 = marge2.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMarge2, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMarge2))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(marge2)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```



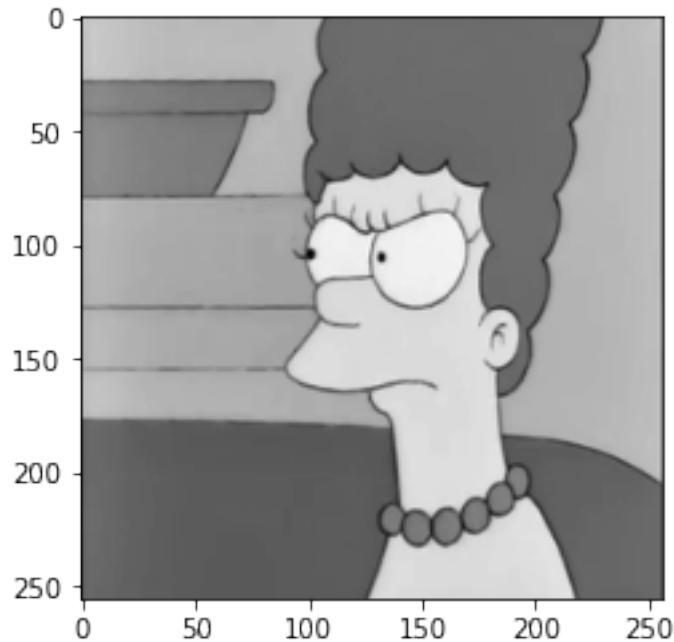
Homer

Don't think she'd be pleased with that prediction.

```
[53]: marge3 = Image.open("unseenData/Marge/3.png")
plt.imshow(marge3)
plt.show()
grayscaleMarge3 = marge3.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMarge3, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMarge3))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
```

```
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(marge3)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





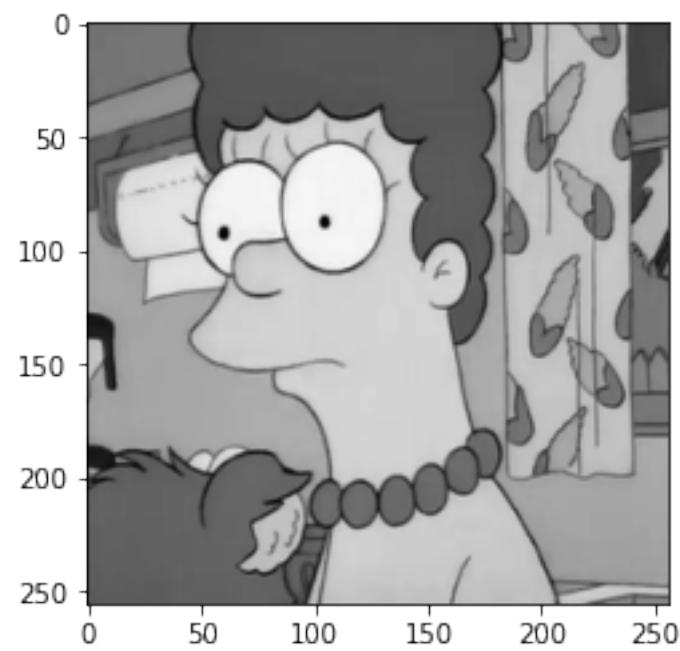
```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)`, if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")`, if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



Marge

```
[54]: marge4 = Image.open("unseenData/Marge/4.png")
plt.imshow(marge4)
plt.show()
grayscaleMarge4 = marge4.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMarge4, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMarge4))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
plt.imshow(marge4)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
```

```
print("Maggie")
elif result[0] == 5:
    print("Marge")
```



```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)` , if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")` , if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(``model.predict_classes()` is deprecated and '
```

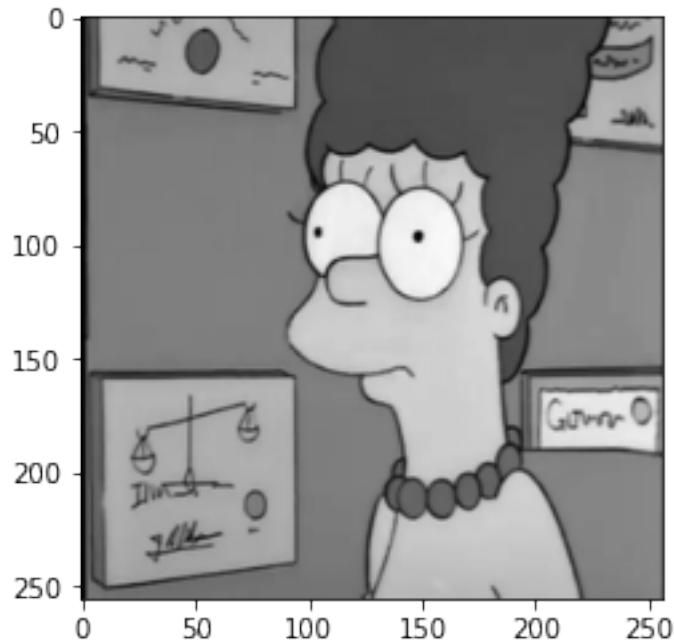


Bart

```
[55]: marge5 = Image.open("unseenData/Marge/5.png")
plt.imshow(marge5)
plt.show()
grayscaleMarge5 = marge5.resize((256,256),Image.ANTIALIAS).convert('L')
plt.imshow(grayscaleMarge5, cmap="gray")
plt.show()
Xt = []
Xt.append(np.array(grayscaleMarge5))
# Convert to NP array
Xt = np.array(Xt)
# Reshape 2D
Xt = Xt.reshape(Xt.shape[0], 1, 256, 256).astype('float32')
# Normalize the data
Xt = Xt /255
result = modelC.predict_classes(Xt)
```

```
plt.imshow(marge5)
plt.show()
if result[0] == 1:
    print("Bart")
elif result[0] == 2:
    print("Homer")
elif result[0] == 3:
    print("Lisa")
elif result[0] == 4:
    print("Maggie")
elif result[0] == 5:
    print("Marge")
```





```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead: * `np.argmax(model.predict(x), axis=-1)`, if your model
does multi-class classification (e.g. if it uses a `softmax` last-layer
activation). * `(model.predict(x) > 0.5).astype("int32")`, if your model does
binary classification (e.g. if it uses a `sigmoid` last-layer activation).
warnings.warn(`model.predict_classes()` is deprecated and '
```



Marge

3/5 for Marge.

10 Calculation & Reporting of Performance

Bart: 3/5 Predictions

Homer: 3/5 Predictions

Lisa: 4/5 Predictions

Maggie: 1/5 Predictions

Marge: 3/5 Predictions

That is 13/25 predictions correct. That's a 52% accuracy. Considering the initial model's accuracy is 50.6%, that is pretty damn close with real unseen data!

I personally think with Maggie, the model failed to predict her features because she is very similar looking to both Lisa and Marge. She has the hair of Lisa and the same eyelashes as Marge, so she's the weird inbetween with little distinguishing features besides her soother.

On a side note, I wonder if the boosted quality of the images hampered the performance of the model. If we'd gone with a default of 26*26, we would just be seeing blobs of characters, with little distinguishing features. However, with the heightened resolution, the model has more of a right to be picky – so I question if this is why we struggled to get a test accuracy of more than 50%.