[18]:
```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[19]:
```
%matplotlib inline
```

## Obtención de patches de imágenes WSI (Whole Slide Images)

Las imágenes en formato *svs* son muy útiles para los médicos en ayuda al diagnóstico y para análisis individualizados debido a su alta resolución pero este formato no es válido como entrada a una red neuronal.

Por un lado es necesaria la conversión a un formato de una única capa como *png* o *jpg* en lugar del formato piramidal *svs*.

El otro problema es la resolución, de miles por miles de pixeles, una resolución de 1024x1024 ya significa que la capa de entrada de la red neuronal tiene más de un millón de neuronas; en en el caso de imágenes en blanco y negro, a color habría que multiplicar por 3 al tener 3 canales. Para solucionar este problema se divive cada imagen en *patches* (parches), cuadrados más pequeños sobre imagen original.

En este Notebook se realizará el parcheado de cada una de las imágenes descargadas anteriormente, guardando en formato .png cada uno de los *patches* obtenidos.

Para entender bien este concepto de *patching* sobre una imagen SVS se recomienda ver el ejemplo en el notebook *wsi_patching_example.ipynb* o su correspondiente html *wsi_patching_example.html*

### Librerías

[138]:
```
import yaml
import os
import openslide
import pandas as pd
pd.set_option('display.max_colwidth', -1)

import matplotlib.image as mpimg
from matplotlib import pyplot as plt

try:
    get_ipython()
    from tqdm import tqdm_notebook as tqdm
except:
    from tqdm import tqdm
```

[131]:
```
from wsi.slide import thumbnail
from wsi.patch import patch_slides
```

### Config

[133]:
```
with open('conf/user_conf.yaml', 'r') as f:
    conf = yaml.load(f)
```

### Creación de rutas

Se crean los directorios para guardar las imágenes de salida.

[9]:
```
slides_path = os.path.join(conf['data_path'], 'slides', 'svs')
```

[10]:
```
patches_path = os.path.join(conf['data_path'], 'slides', 'patches')

if not os.path.exists(patches_path):
    os.mkdir(patches_path)
```

[11]:
```
thumbnails_path = os.path.join(conf['data_path'], 'slides', 'thumbnail')

if not os.path.exists(thumbnails_path):
    os.mkdir(thumbnails_path)
```

**Lectura de DataFrame de slides**

Se lee el DataFrame de slides para obtener el nombre de todas las imágenes descargadas. Se filtra por imágenes contenidas en el directorio por si alguna no hubiera podido ser descargada.

[13]:
```
slides_df = pd.read_csv(os.path.join(conf['data_path'], 'slides_metadata.csv'),␣
↪sep='|')
slides_df = slides_df[slides_df['file_name'].isin(os.listdir(slides_path))]
```

[140]:
```
slides_df.head(2)
```

[140]:
```
                                file_id       case_id         sample_id  \
0  e1daa3de-3f76-44fb-ba6b-f60af4943ef3  TCGA-2J-AABR  TCGA-2J-AABR-01A
1  dfa7125a-e250-4abf-a528-608dded99751  TCGA-US-A77G  TCGA-US-A77G-11A

                  slide_id    data_type experimental_strategy data_format  \
0  TCGA-2J-AABR-01A-01-TS1  Slide Image  Tissue Slide          SVS
1  TCGA-US-A77G-11A-01-TSA  Slide Image  Tissue Slide          SVS

   file_size                    file_name primary_site  \
0  381.90     TCGA-2J-AABR-01A-01-TS1.svs  Pancreas
1  24.74      TCGA-US-A77G-11A-01-TSA.svs  Pancreas

                  disease_type          sample_type  is_ffpe  \
0  Ductal and Lobular Neoplasms  Primary Tumor        False
1  Ductal and Lobular Neoplasms  Solid Tissue Normal  False

   percent_normal_cells  percent_stromal_cells  percent_tumor_cells  \
0  65.0                  0.0                   35.0
1  15.0                  85.0                  0.0

   percent_tumor_nuclei
0  20.0
1  0.0
```

[15]:
```
slides_df.groupby(['experimental_strategy', 'sample_type']).size()
```

[15]:
```
experimental_strategy  sample_type
Tissue Slide           Metastatic              1
                       Primary Tumor         219
                       Solid Tissue Normal    37
dtype: int64
```

### Guarda miniaturas

Para facilitar el trabajo cuando se quiera visualizar una imagen al completo se guardará en formato .png una una imágen en baja resolución de la imagen SVS original. De esta manera no será necesario utilizar herramientas de lectura de imágenes SVS para hacerse una idea de la estructura completa de una lámina.

La resolución de esta imagen de sale se selecciona en el fichero de configuración bajo el parámetro *thumbnail_size*.

[135]:
```
conf['wsi']['thumbnail_size']
```

[135]:
```
1024
```

[ ]:
```
for file in tqdm(slides_df['file_name'].values, unit='file'):

    os_img = openslide.open_slide(os.path.join(slides_path, file))
    img = thumbnail(os_img, max_size=conf['wsi']['thumbnail_size'])

    img.save(os.path.join(thumbnails_path, file.replace('.svs', '.png')))
```

### Patching

Finalmente se realiza el parcheado de cada una de las imágenes. La función *patch_slides* recibe una lista de nombres de ficheros y almacena todos los *patches* generados en un directorio. Los parámetros que se pueden modificar sobre el parcheado son los siguienes:

- *patch_size*: tamaño en píxeles de cada uno de los cuadrados.
- *magnification*: nivel de aumento microscópico sobre el que realizar el parcheado. A mayor aumento mayor resolución de la imágen original y por tanto mayor número de parches.
- *white_pixel_threshold*: es el umbral del porcentaje máximo de píxeles blancos que puede tener un parche para ser aceptado (ver notebook the ejemplo).
- sampling: tasa de muestreo de los parches. Cuando se utilizan niveles de aumento elevados el número de parches es muy alto y se recomienda hacer un muestreo.

[151]:
```
conf['wsi']
```

[151]:
```
{'magnification': 10,
 'patch_size': 128,
 'thumbnail_size': 1024,
 'white_pixel_threshold': 20,
 'sampling': 0.2}
```

[141]:
```
slide_files = slides_df['file_name'].map(lambda x: os.path.join(slides_path, x))
slide_files.iloc[:5]
```

[141]:
```
0    /Users/portizdegalisteo/TFM_Data/slides/svs/TCGA-2J-AABR-01A-01-TS1.svs
1    /Users/portizdegalisteo/TFM_Data/slides/svs/TCGA-US-A77G-11A-01-TSA.svs
2    /Users/portizdegalisteo/TFM_Data/slides/svs/TCGA-S4-A8RM-01A-01-TSA.svs
3    /Users/portizdegalisteo/TFM_Data/slides/svs/TCGA-FZ-5926-01A-01-TS1.svs
4    /Users/portizdegalisteo/TFM_Data/slides/svs/TCGA-IB-A7LX-01A-01-TSA.svs
Name: file_name, dtype: object
```

[ ]:
```
results = patch_slides(slide_files, patches_path, conf['wsi']['patch_size'], ␣
↪conf['wsi']['magnification'],
                       conf['wsi']['white_pixel_threshold'], conf['wsi']['sampling'])
```

Además de guardar los parches la función también devolverá un DataFrame con un resumen de los resultados para cada imágen, indicando el total de parches generados y cuántos de ellos fueron guardados.

[146]:
```
results.head(5)
```

[146]:

| | file | total_patches | saved_patches | perc_saved_patches |
|---|---|---|---|---|
| 0 | TCGA-2J-AABR-01A-01-TS1.svs | 106 | 8 | 0.08 |
| 1 | TCGA-US-A77G-11A-01-TSA.svs | 25 | 0 | 0.00 |
| 2 | TCGA-S4-A8RM-01A-01-TSA.svs | 59 | 5 | 0.08 |
| 3 | TCGA-FZ-5926-01A-01-TS1.svs | 33 | 9 | 0.27 |
| 4 | TCGA-IB-A7LX-01A-01-TSA.svs | 59 | 12 | 0.20 |

[156]:
```
results['total_patches'].sum()
```

[156]:
```
150529
```

[158]:
```
results['saved_patches'].sum()
```

[158]:
```
22299
```

[160]:
```
round(results['perc_saved_patches'].mean(), 2)
```

[160]:
```
0.17
```

[20]:
```
results.to_csv(os.path.join(conf['data_path'], 'patching_results.csv'), sep='|',␣
↪index=False)
```