- Shin’s Lab -

Python for Data Visualization

Python for Data Visualization

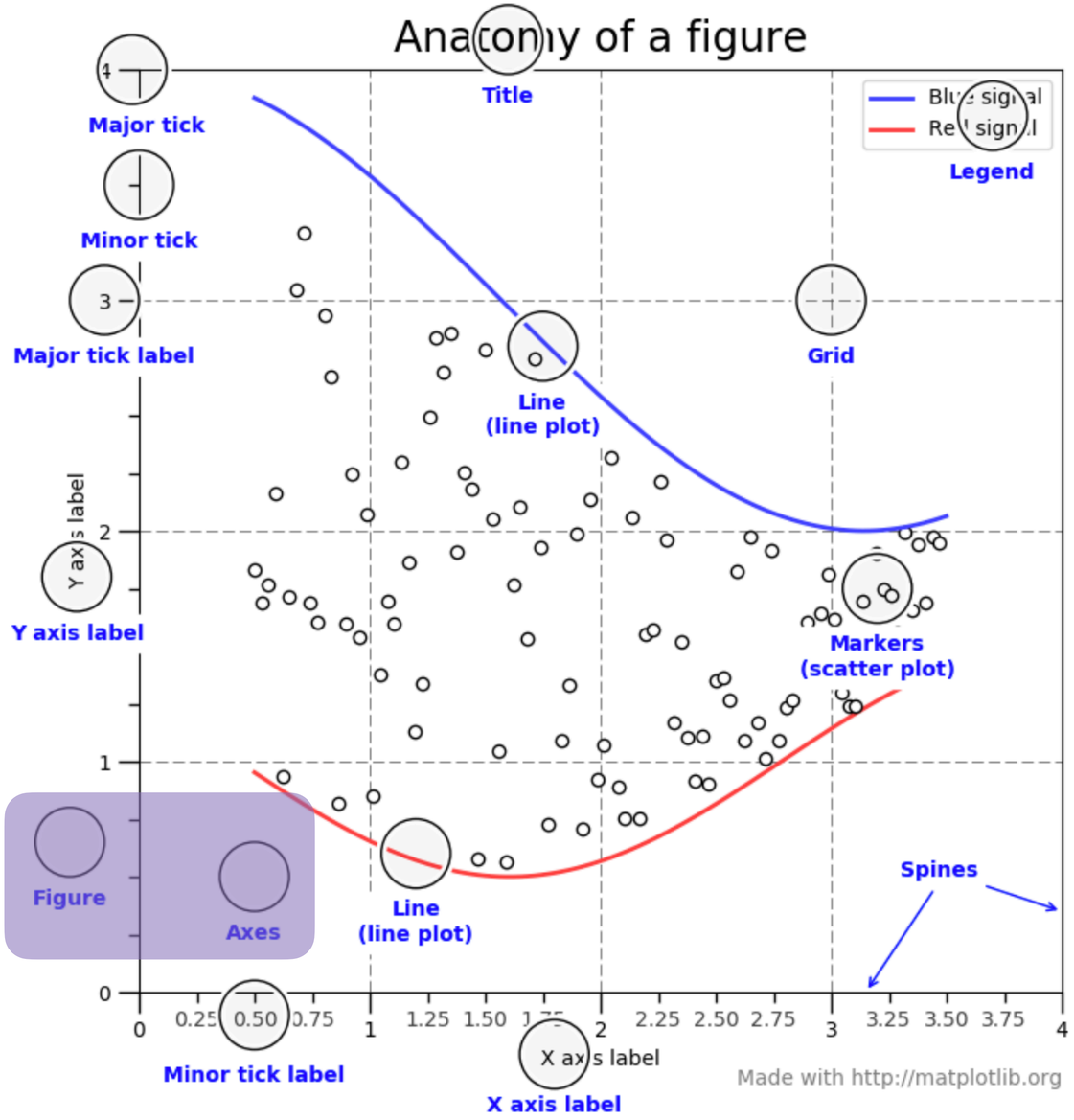
-Chapter.1 Matplotlib Anatomy -

Python for Data Visualization

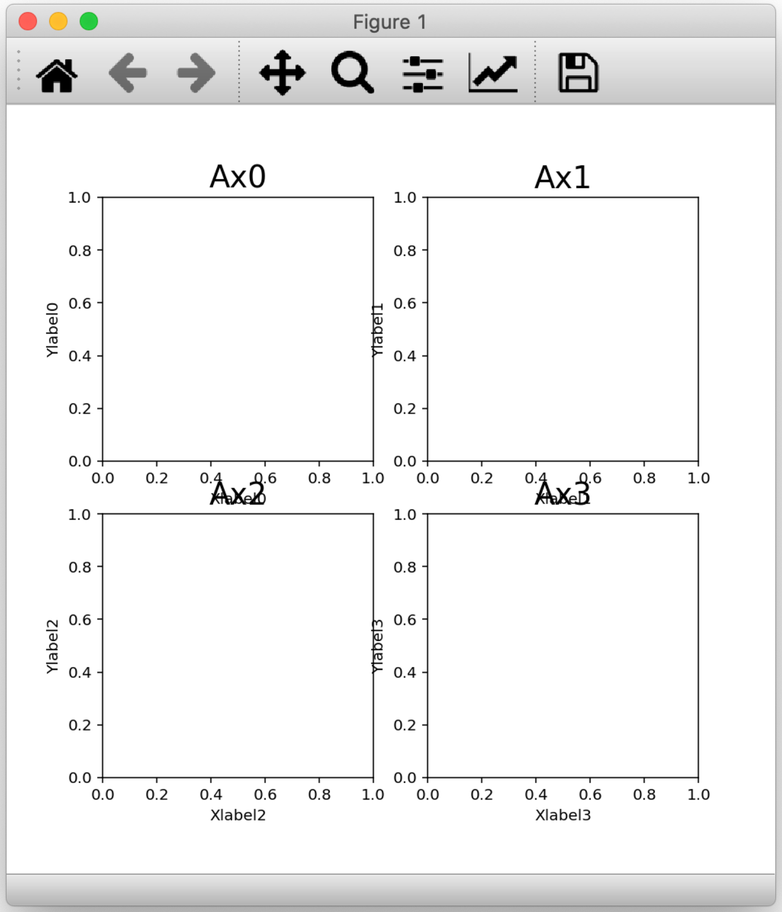
-Chapter.1 Matplotlib Anatomy -

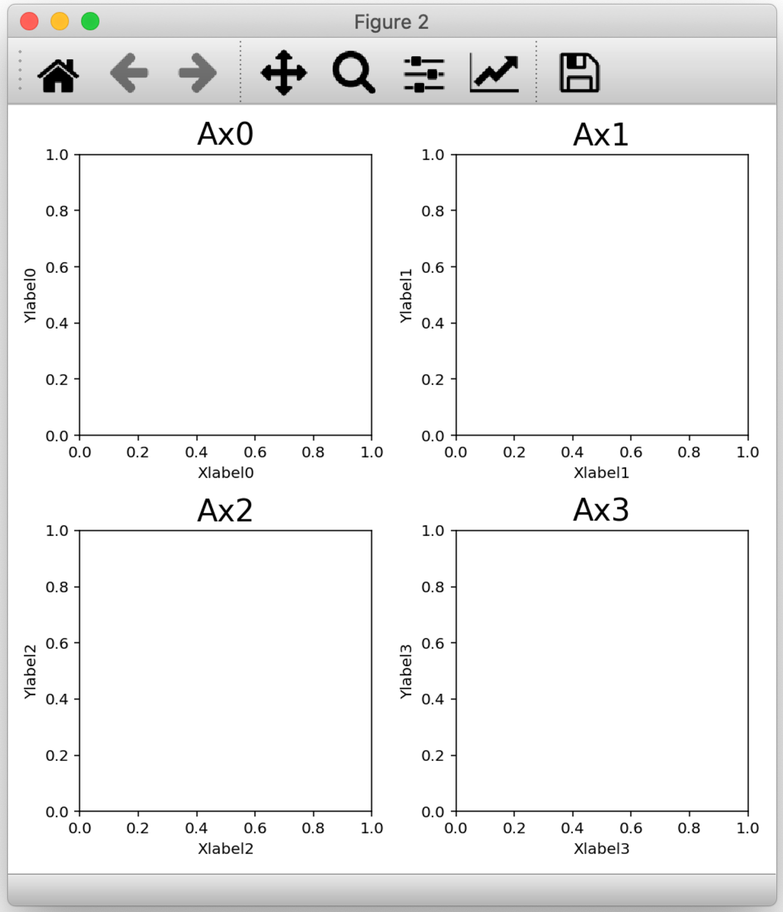
1-02. Axes Customizing

1. fig.tight_layout(Axes Layout Adjustment)
2. fig.subplots_adjust(More Customized Layout)
3. fig.subplots_adjust(Practice)
4. Axis Sharing
5. Axis Sharing(Practice)
6. ax.twinx(Different Y Values)
7. ax.set_yscale(Axis Scale)


Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Minor tick label
Line
(line plot)
Grid
Legend
Markers
(scatter plot)
Y axis label
X axis label
Figure
Axes
Spines
Made with http://matplotlib.org

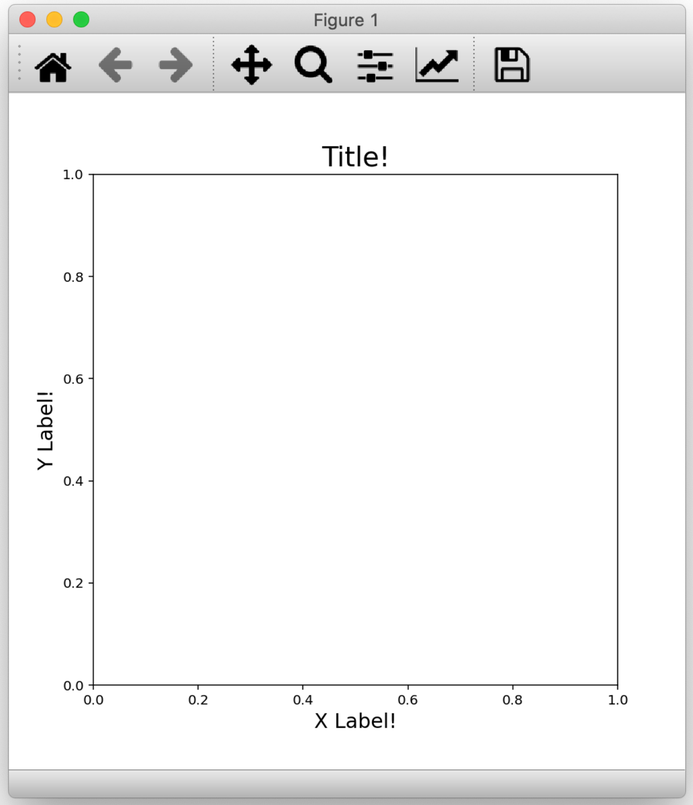
1. fig.tight_layout(Axes Layout Adjustment)





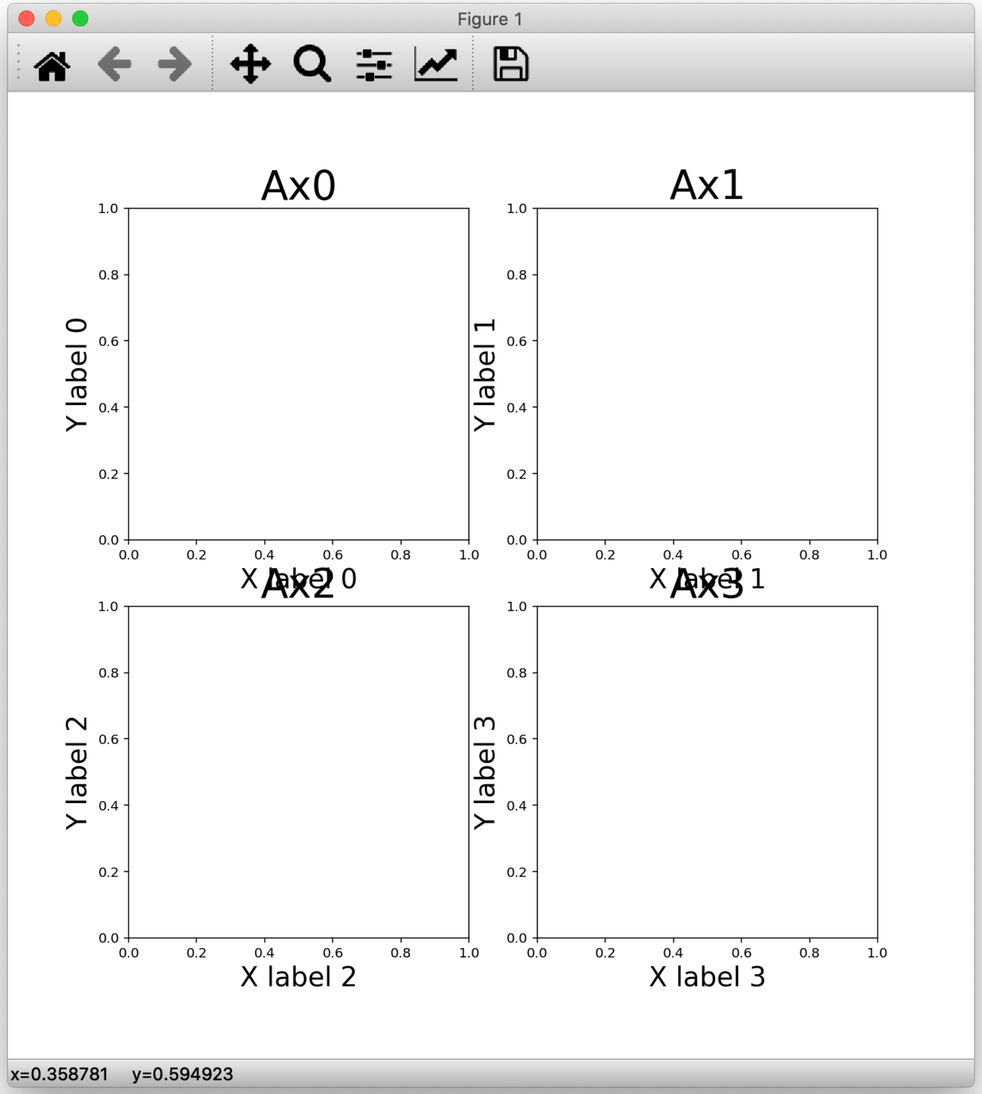
1. fig.tight_layout(ax.set_title, ax.set_xlabel, ax.set_ylabel)

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(7, 7))
ax.set_title('Title!', fontsize=20)
ax.set_xlabel('X Label!', fontsize=15)
ax.set_ylabel('Y Label!', fontsize=15)
```



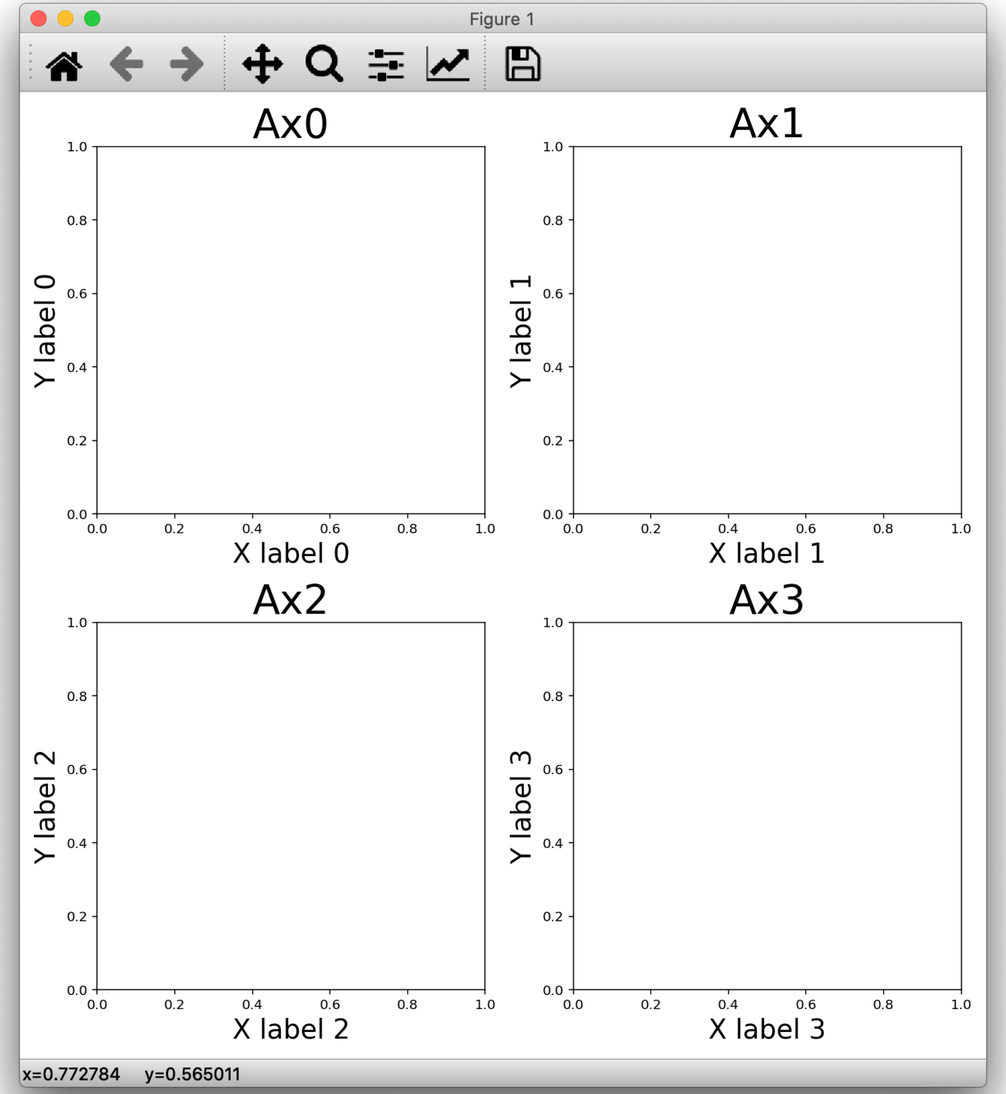
1. fig.tight_layout(ax.set_title, ax.set_xlabel, ax.set_ylabel)

```
title_list = ['Ax' + str(i) for i in range(4)]
xlabel_list = ['X label ' + str(i) for i in range(4)]
ylabel_list = ['Y label ' + str(i) for i in range(4)]

print(title_list)   ['Ax0', 'Ax1', 'Ax2', 'Ax3']
print(xlabel_list)  ['X label 0', 'X label 1', 'X label 2', 'X label 3']
print(ylabel_list)  ['Y label 0', 'Y label 1', 'Y label 2', 'Y label 3']


fig, axes = plt.subplots(2, 2, figsize=(10, 10))
for ax_idx, ax in enumerate(axes.flat):
    ax.set_title(title_list[ax_idx],
                 fontsize=30)
    ax.set_xlabel(xlabel_list[ax_idx],
                  fontsize=20)
    ax.set_ylabel(ylabel_list[ax_idx],
                  fontsize=20)
```



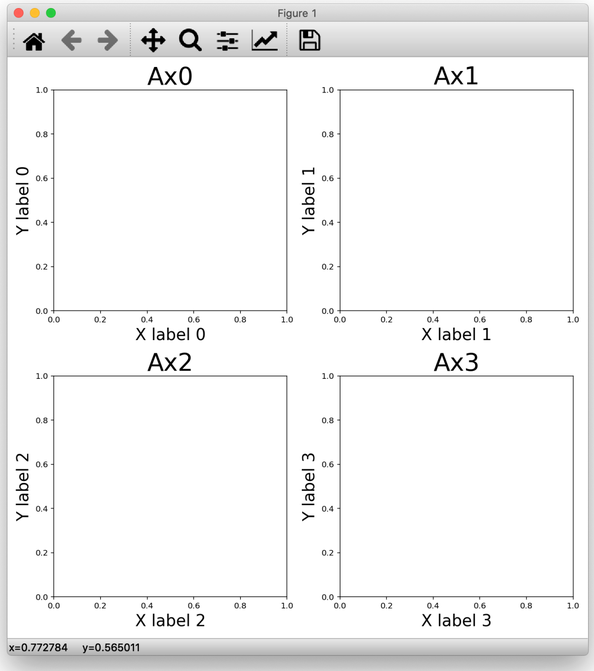
1. fig.tight_layout(Usage)

```
title_list = ['Ax' + str(i) for i in range(4)]
xlabel_list = ['X label ' + str(i) for i in range(4)]
ylabel_list = ['Y label ' + str(i) for i in range(4)]

fig, axes = plt.subplots(2, 2, figsize=(10, 10))
for ax_idx, ax in enumerate(axes.flat):
    ax.set_title(title_list[ax_idx],
                 fontsize=30)
    ax.set_xlabel(xlabel_list[ax_idx],
                  fontsize=20)
    ax.set_ylabel(ylabel_list[ax_idx],
                  fontsize=20)

fig.tight_layout()
```

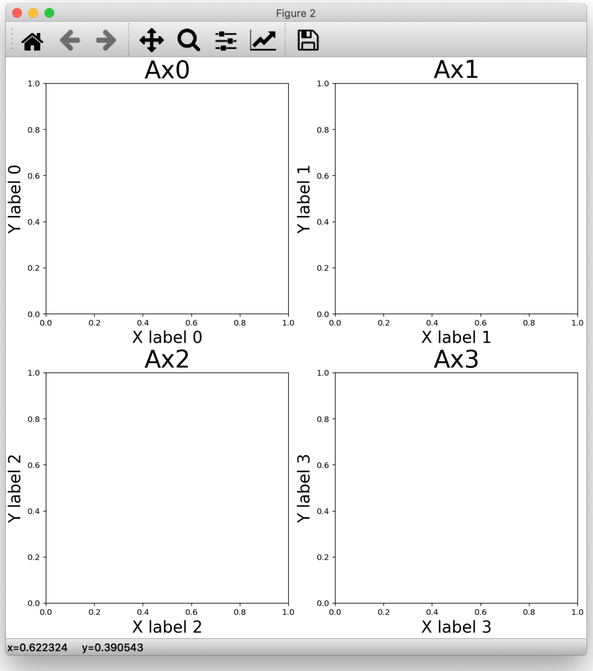


1. fig.tight_layout(pad Argument)

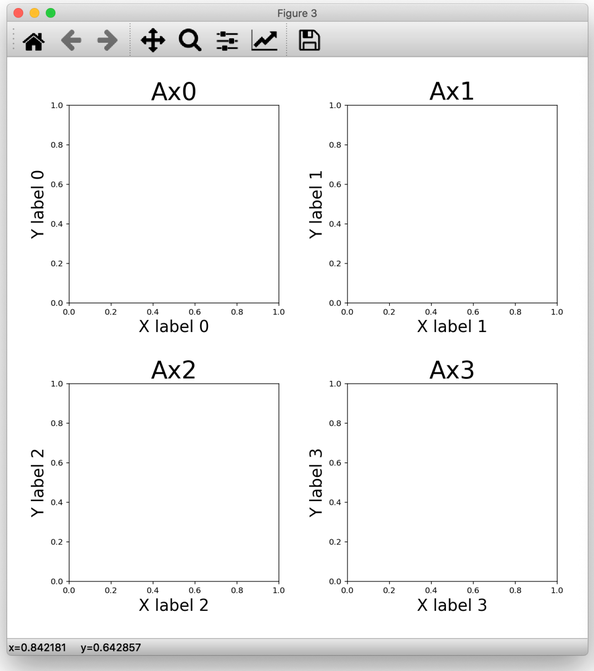
`fig.tight_layout()`



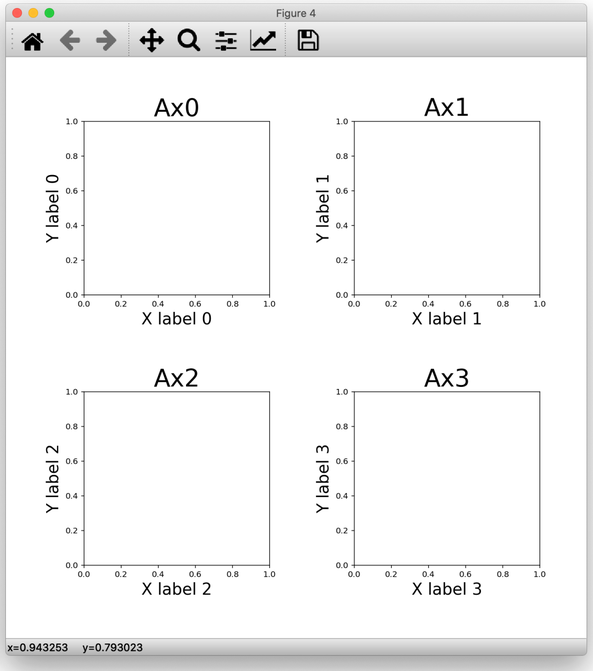
`fig.tight_layout(pad=0.3)`



`fig.tight_layout(pad=3)`



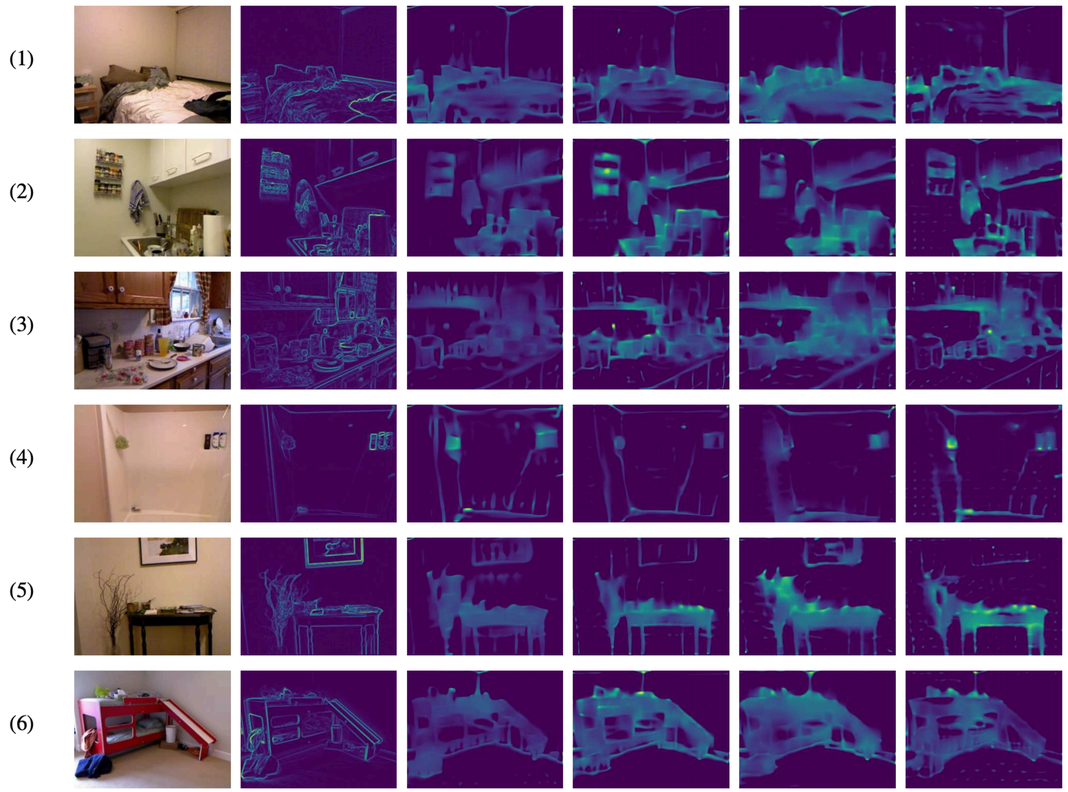
`fig.tight_layout(pad=5)`

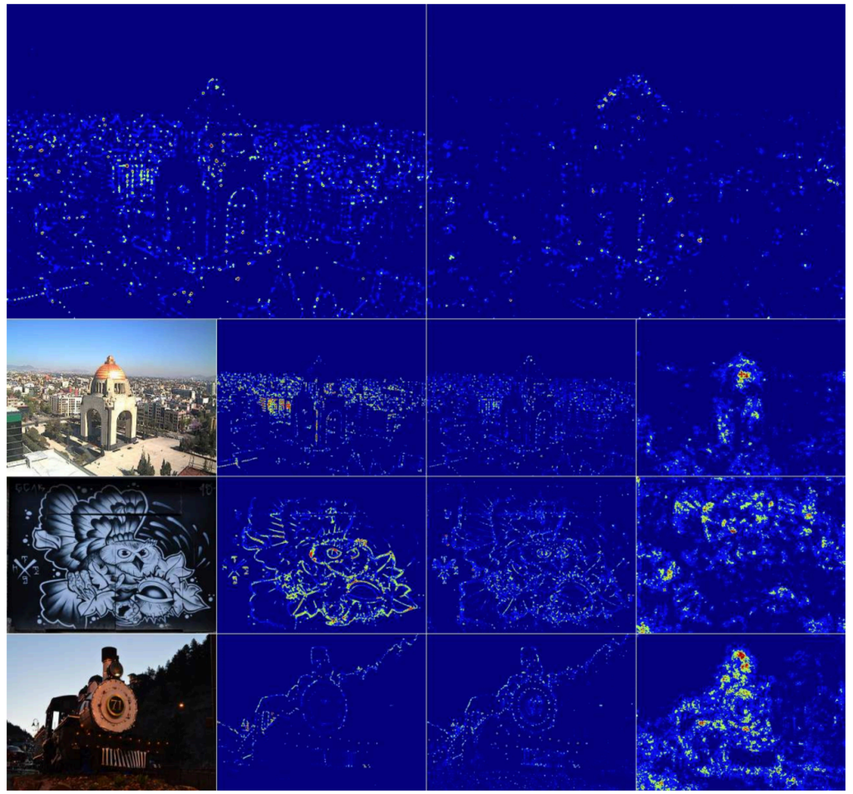


2. fig.subplots_adjust(More Customized Layout)

Visualization of Convolutional Neural Networks for Monocular Depth Estimation

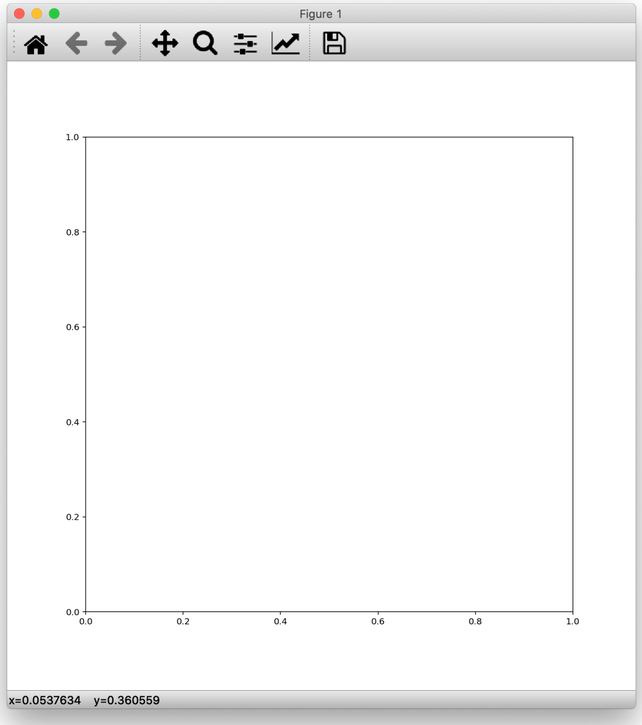
Junjie Hu[1,2] Yan Zhang[2] Takayuki Okatani[1,2]
[1] Graduate School of Information Sciences, Tohoku University, Japan
[2] Center for Advanced Intelligence Project, RIKEN, Japan
{junjie.hu, zhang, okatani}@vision.is.tohoku.ac.jp



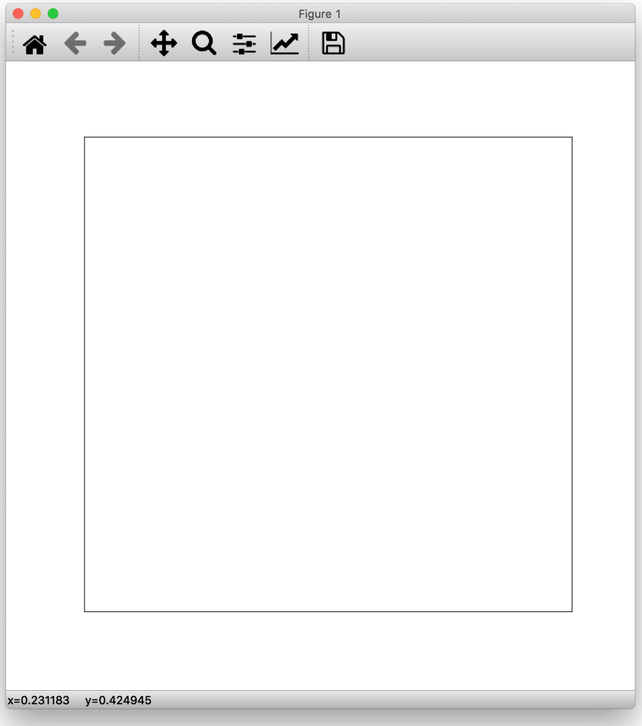


2. fig.subplots_adjust(axis.set_visible)

```
fig, ax = plt.subplots(figsize=(10, 10))
```



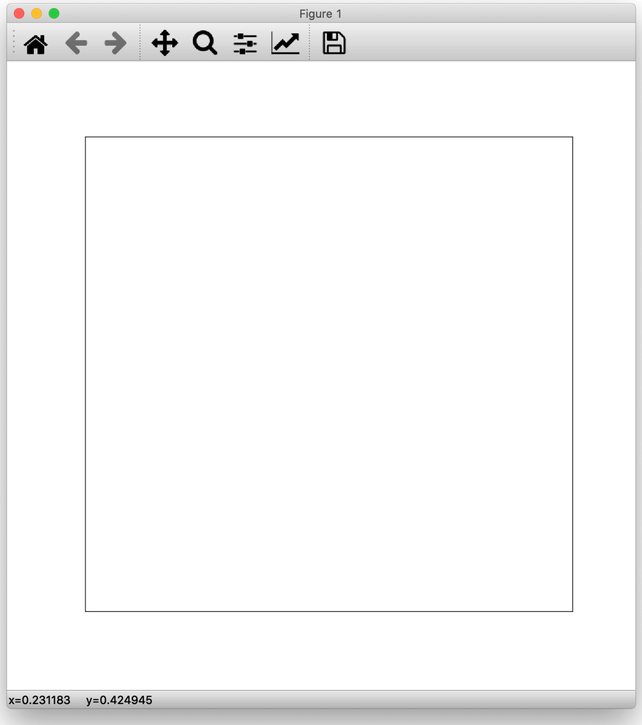

```
fig, ax = plt.subplots(figsize=(10, 10))

xaxis = ax.get_xaxis()
yaxis = ax.get_yaxis()

xaxis.set_visible(False)
yaxis.set_visible(False)
```



2. fig.subplots_adjust(axis.set_visible)

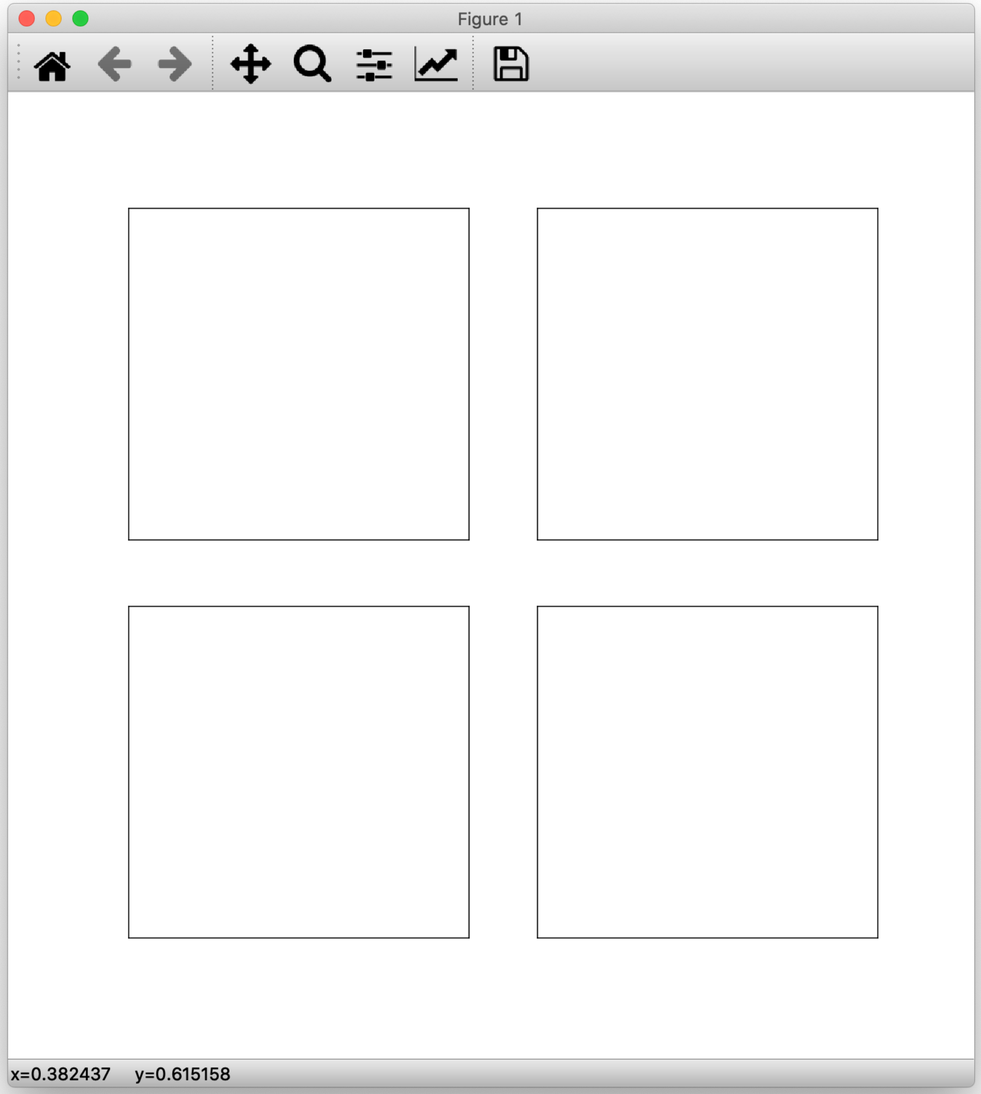
```
fig, ax = plt.subplots(figsize=(10, 10))

xaxis = ax.get_xaxis()
yaxis = ax.get_yaxis()

xaxis.set_visible(False)
yaxis.set_visible(False)
```

```
fig, ax = plt.subplots(figsize=(10, 10))

ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
```

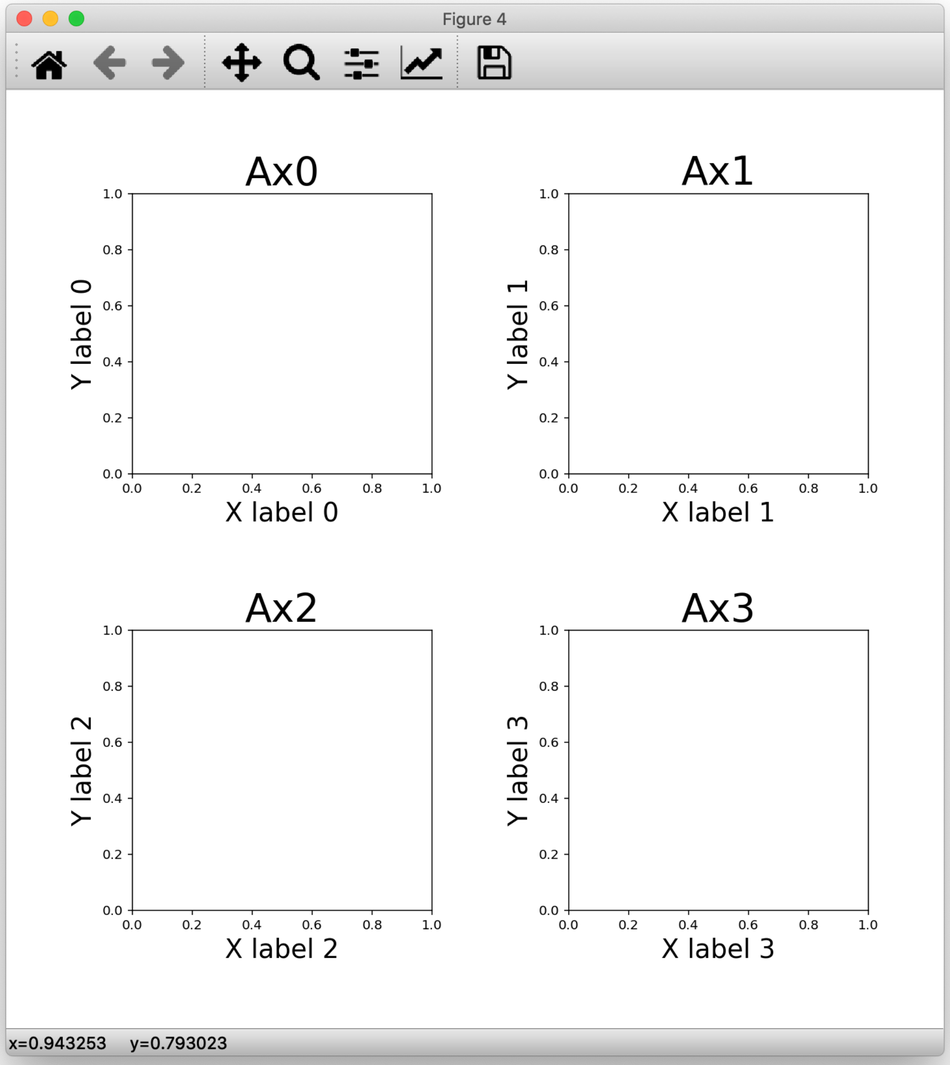


2. fig.subplots_adjust(axis.set_visible)

```
fig, axes = plt.subplots(2, 2, figsize=(10, 10))

for ax_idx, ax in enumerate(axes.flat):
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```



2. fig.subplots_adjust(Layout Parameters)



2. fig.subplots_adjust(Document)

matplotlib.pyplot.subplots_adjust

matplotlib.pyplot.subplots_adjust(*left=None*, *bottom=None*, *right=None*, *top=None*, *wspace=None*, *hspace=None*) [source]

Adjust the subplot layout parameters.

Unset parameters are left unmodified; initial values are given by `rcParams["figure.subplot.[name]"]`.

Parameters:

left : float, optional

The position of the left edge of the subplots, as a fraction of the figure width.

right : float, optional

The position of the right edge of the subplots, as a fraction of the figure width.

bottom : float, optional

The position of the bottom edge of the subplots, as a fraction of the figure height.

top : float, optional

The position of the top edge of the subplots, as a fraction of the figure height.

wspace : float, optional

The width of the padding between subplots, as a fraction of the average axes width.

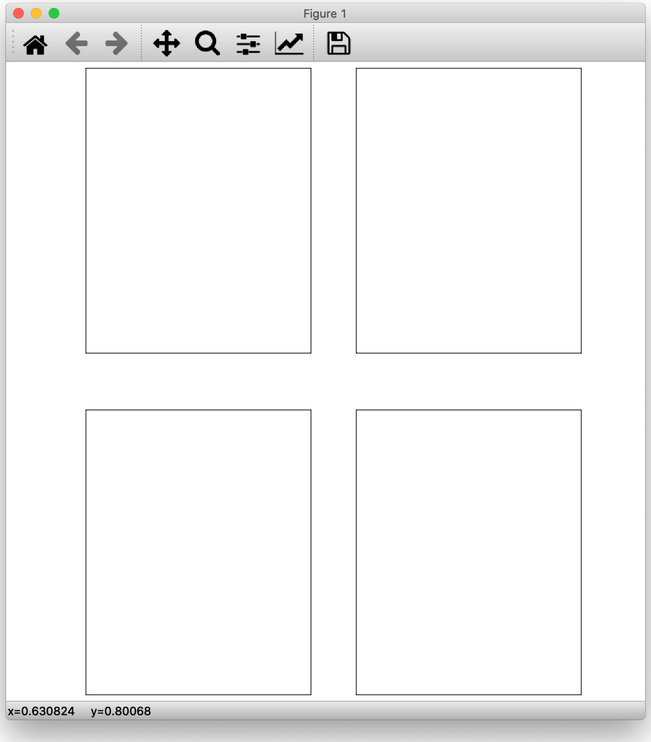
hspace : float, optional

The height of the padding between subplots, as a fraction of the average axes height.

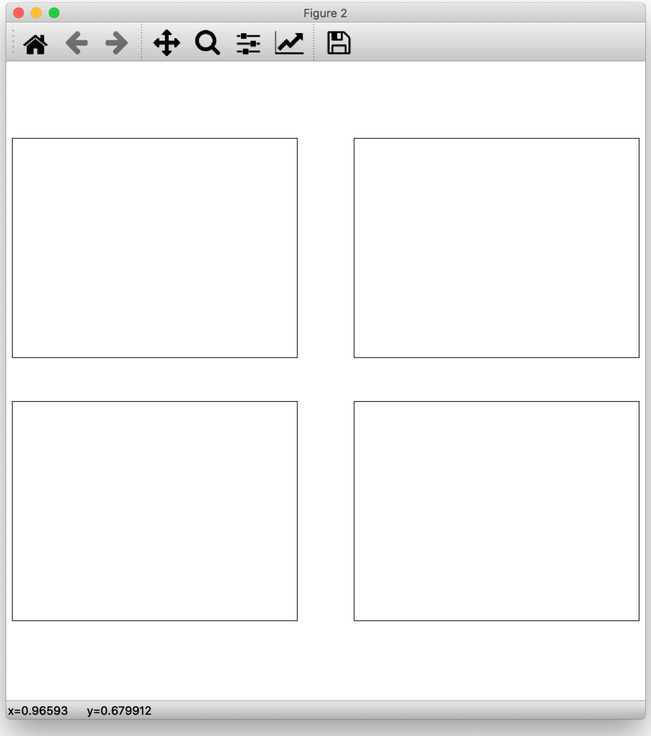
2. fig.subplots_adjust(Position Arguments)

```
fig, axes = plt.subplots(2, 2, figsize=(10, 10))

for ax_idx, ax in enumerate(axes.flat):
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

```
fig.subplots_adjust(bottom=0.01, top=0.99)
```



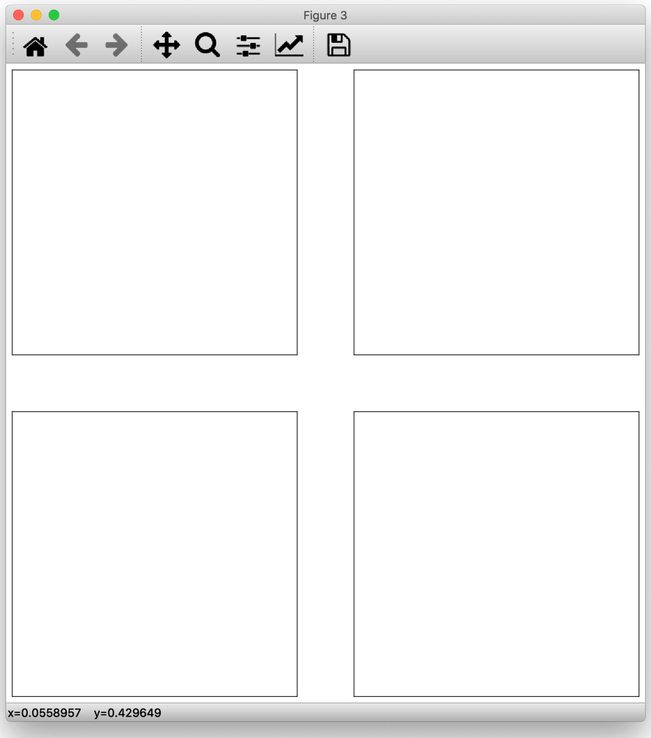

```
fig.subplots_adjust(left=0.01, right=0.99)
```



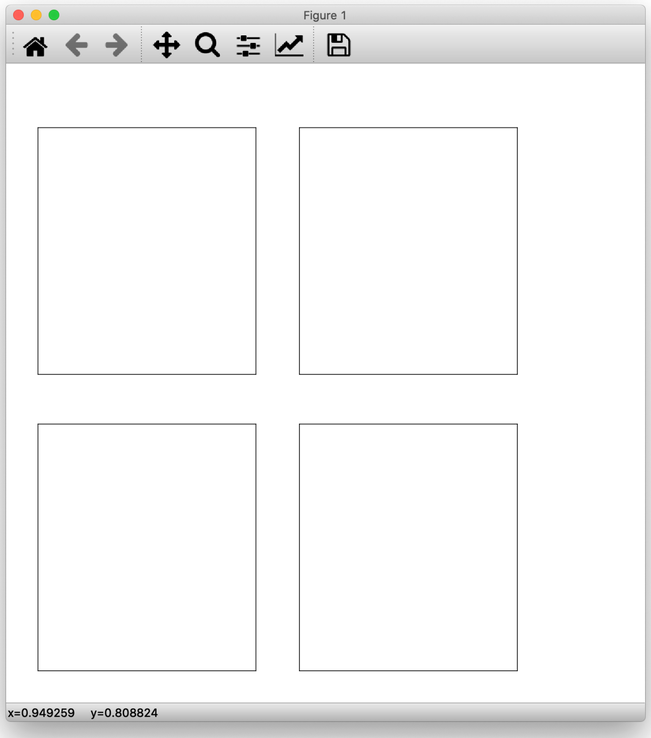
2. fig.subplots_adjust(Position Arguments)

```
fig, axes = plt.subplots(2, 2, figsize=(10, 10))

for ax_idx, ax in enumerate(axes.flat):
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

```
fig.subplots_adjust(bottom=0.01, top=0.99,
                    left=0.01, right=0.99)
```



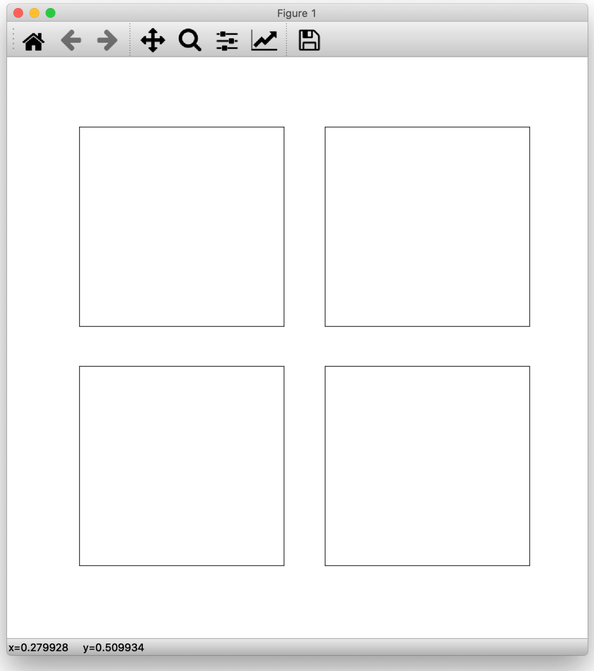

```
fig.subplots_adjust(bottom=0.05, top=0.9,
                    left=0.05, right=0.8)
```



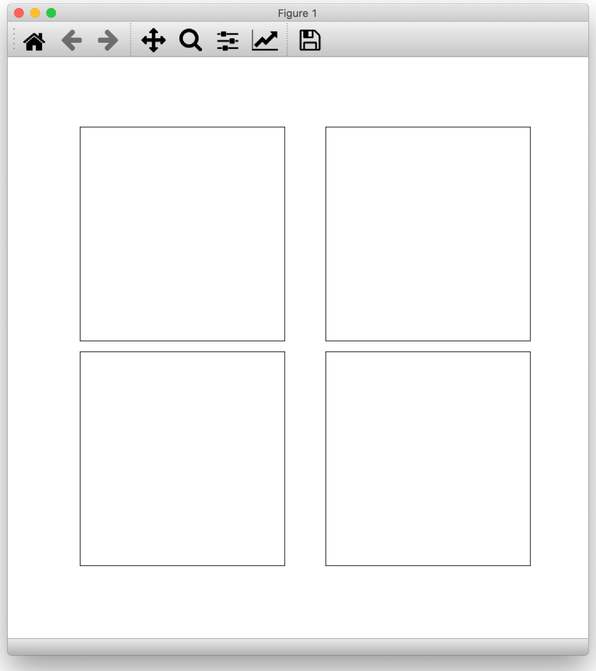
2. fig.subplots_adjust(Spacing Arguments)

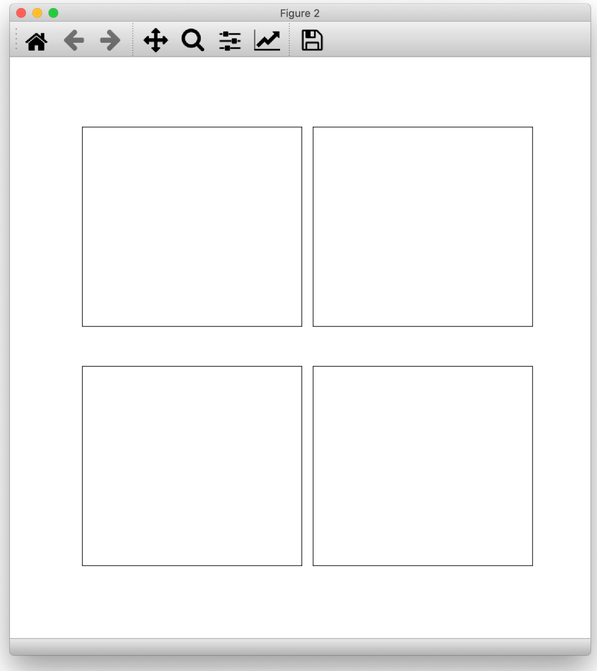
```
fig, axes = plt.subplots(2, 2, figsize=(10, 10))

for ax_idx, ax in enumerate(axes.flat):
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

```
fig.subplots_adjust(hspace=0.05)
```



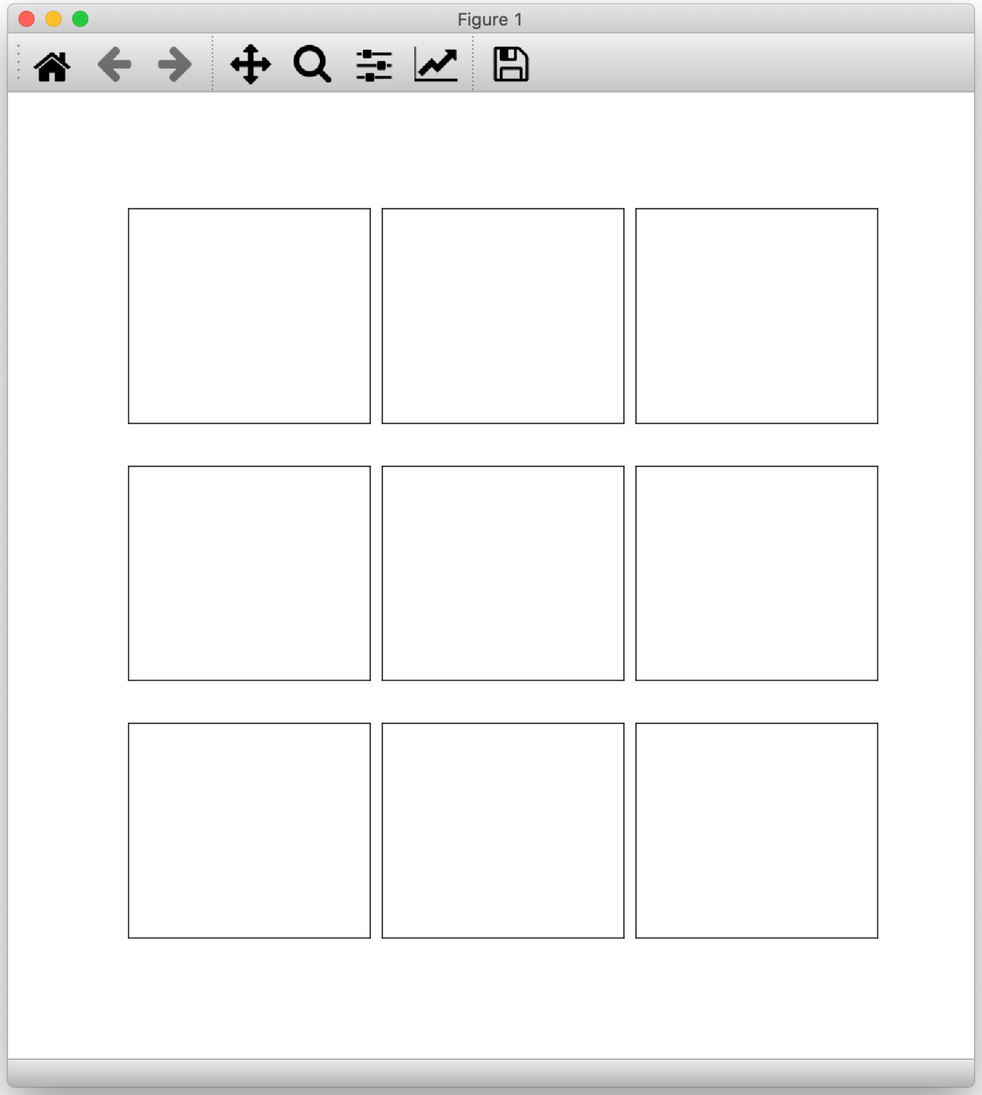

```
fig.subplots_adjust(wspace=0.05)
```





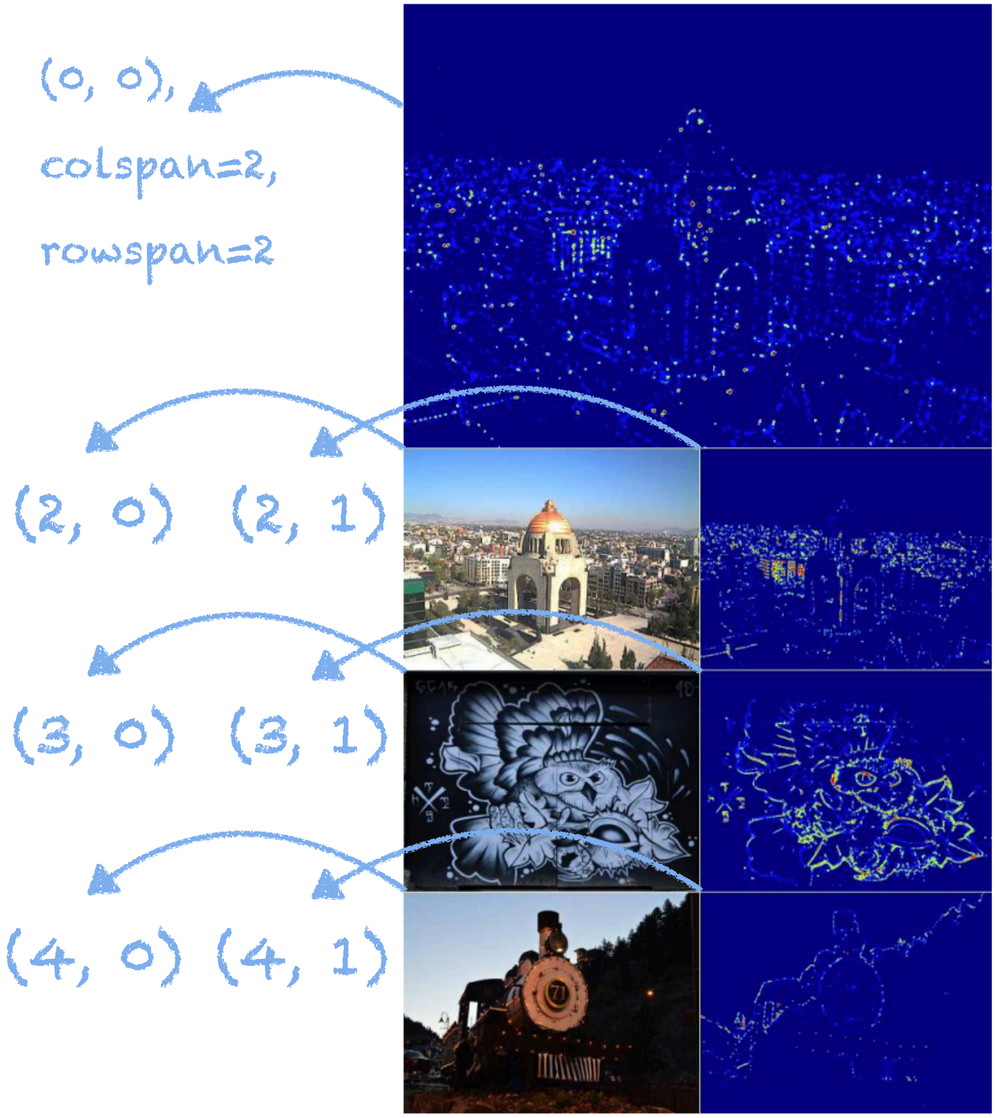
2. fig.subplots_adjust(Spacing Arguments)

```
fig, axes = plt.subplots(3, 3, figsize=(10, 10))

for ax_idx, ax in enumerate(axes.flat):
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

fig.subplots_adjust(hspace=0.2, wspace=0.05)
```



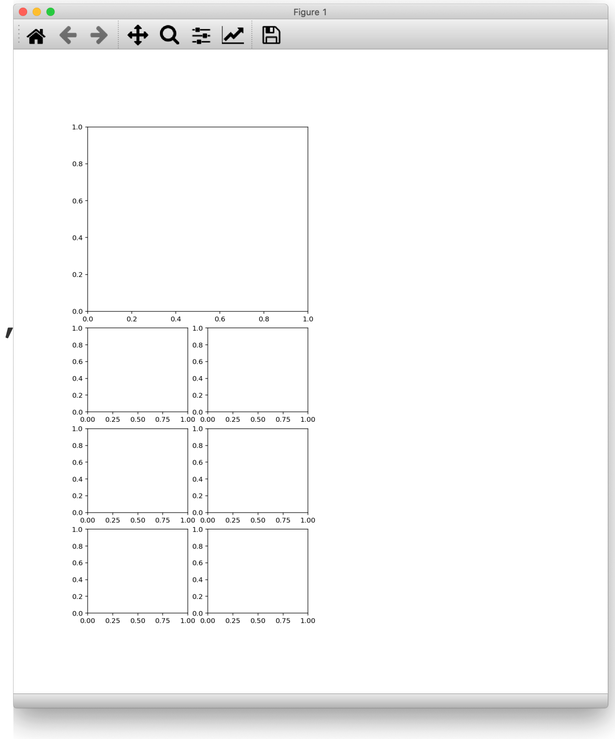
3. fig.subplots_adjust(Practice)

3. fig.subplots_adjust(Practice)

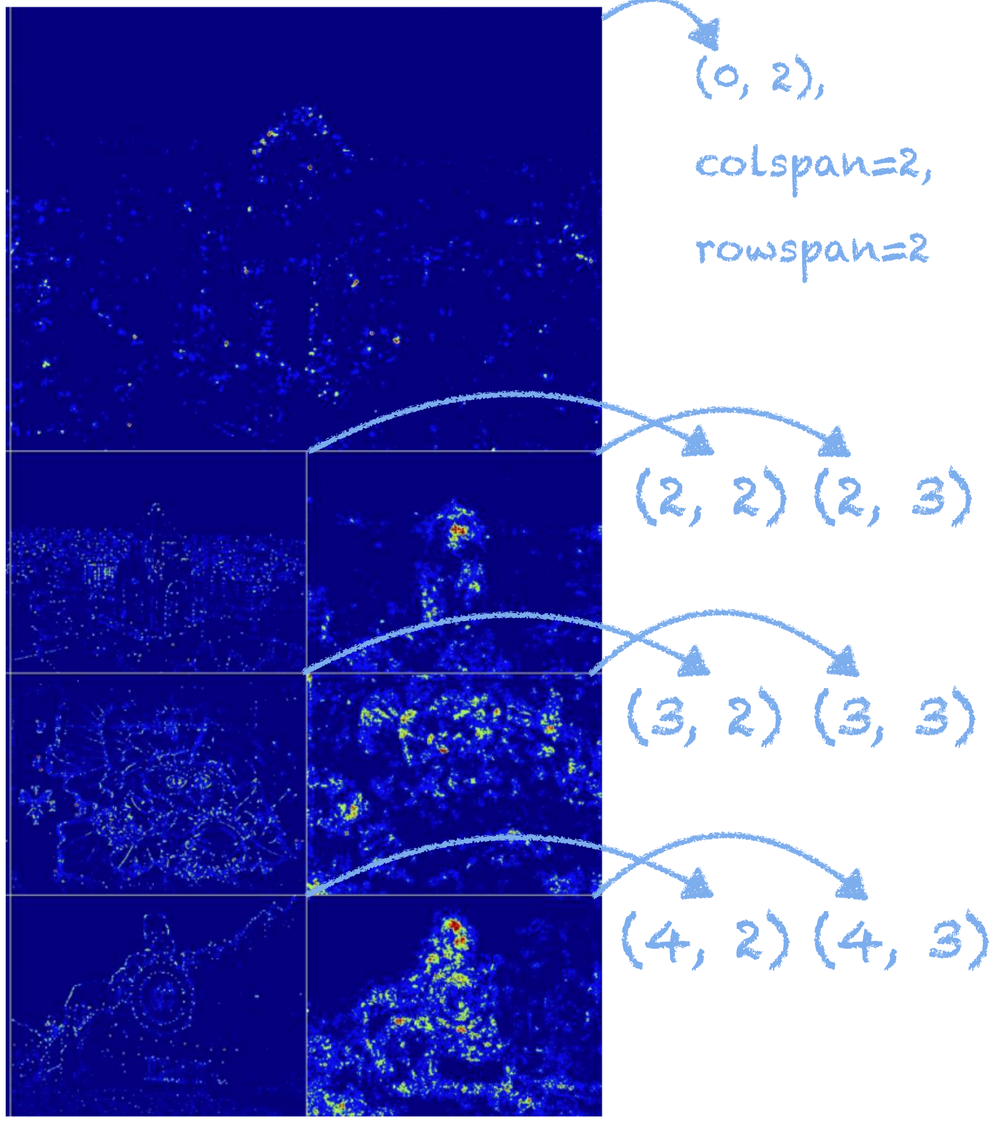



```
fig = plt.figure(figsize=(12, 13))

axes_g1 = np.empty(shape=(0,))
main = plt.subplot2grid((5, 4), (0, 0),
                        2, 2, fig=fig)

axes_g1 = np.append(axes_g1, main)

for r_idx in range(2, 2 + 3):
    for c_idx in range(2):
        ax = plt.subplot2grid((5, 4),
                              (r_idx, c_idx),
                              fig=fig)
        axes_g1 = np.append(axes_g1, ax)


axes_g1 = axes_g1.reshape(1, -1)
print(axes_g1.shape)
```

(1, 7)

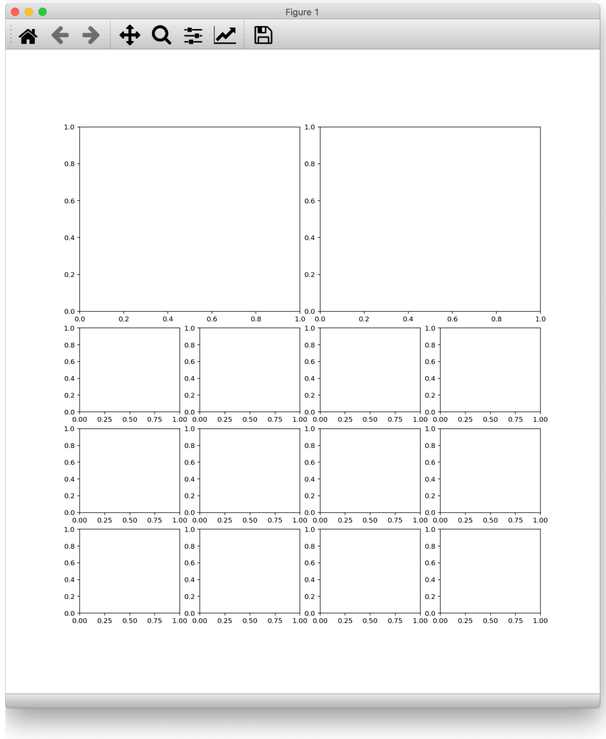


3. fig.subplots_adjust(Practice)




```
axes_g2 = np.empty(shape=(0,))
main = plt.subplot2grid((5, 4), (0, 2),
                        2, 2, fig=fig)

axes_g2 = np.append(axes_g2, main)
for r_idx in range(2, 2 + 3):
    for c_idx in range(2, 2 + 2):
        ax = plt.subplot2grid((5, 4),
                              (r_idx, c_idx),
                              fig=fig)
        axes_g2 = np.append(axes_g2, ax)

axes_g2 = axes_g2.reshape(1, -1)
print(axes_g2.shape)
```

(1, 7)

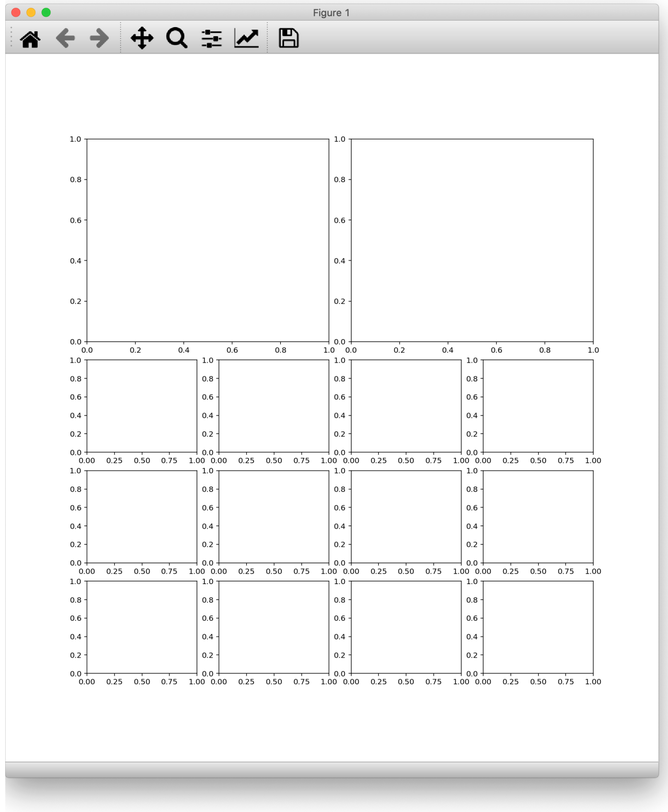


3. fig.subplots_adjust(Practice)

```
fig = plt.figure(figsize=(12, 13))

axes_g1 = np.empty(shape=(0,))

main = plt.subplot2grid((5, 4), (0, 0),
                        2, 2, fig=fig)

axes_g1 = np.append(axes_g1, main)

for r_idx in range(2, 2 + 3):
    for c_idx in range(2):
        ax = plt.subplot2grid((5, 4),
                              (r_idx, c_idx),
                              fig=fig)
        axes_g1 = np.append(axes_g1, ax)

axes_g1 = axes_g1.reshape(1, -1)
```

```
axes_g2 = np.empty(shape=(0,))

main = plt.subplot2grid((5, 4), (0, 2),
                        2, 2, fig=fig)

axes_g2 = np.append(axes_g2, main)

for r_idx in range(2, 2 + 3):
    for c_idx in range(2, 2 + 2):
        ax = plt.subplot2grid((5, 4),
                              (r_idx, c_idx),
                              fig=fig)
        axes_g2 = np.append(axes_g2, ax)

axes_g2 = axes_g2.reshape(1, -1)


axes = np.vstack((axes_g1, axes_g2))
print(axes.shape)
```

```
(2, 7)
```



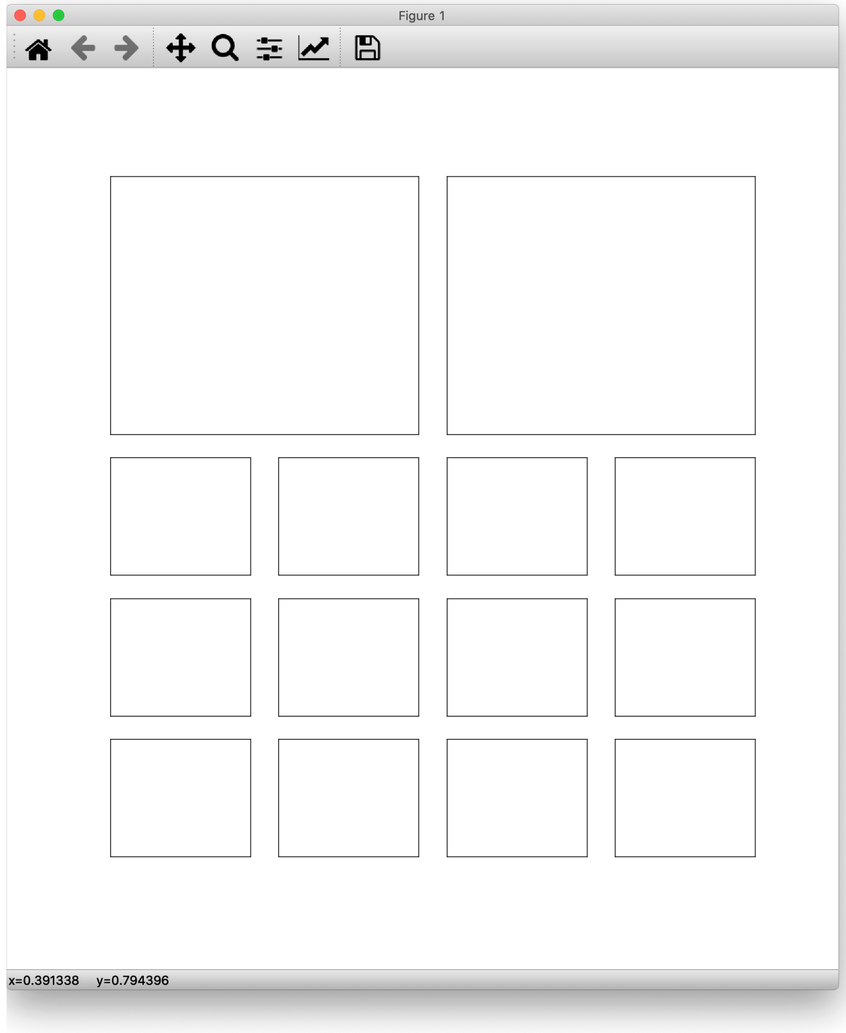
3. fig.subplots_adjust(Practice)

3. fig.subplots_adjust(Practice)

```
fig = plt.figure(figsize=(12, 13))

axes = np.empty(shape=(0, 7))
for g_idx in range(2):
    axes_g = np.empty(shape=(0,))

    main = plt.subplot2grid((5, 4), (0, 2*g_idx),
                            2, 2,
                            fig=fig)
    axes_g = np.append(axes_g, main)


    for r_idx in range(2, 2 + 3):
        for c_idx in range(2):
            ax = plt.subplot2grid((5, 4),
                                  (r_idx, c_idx + 2*g_idx),
                                  fig=fig)
            axes_g = np.append(axes_g, ax)

    axes_g = axes_g.reshape(1, -1)
    axes = np.vstack((axes, axes_g))
```

3. fig.subplots_adjust(Practice)

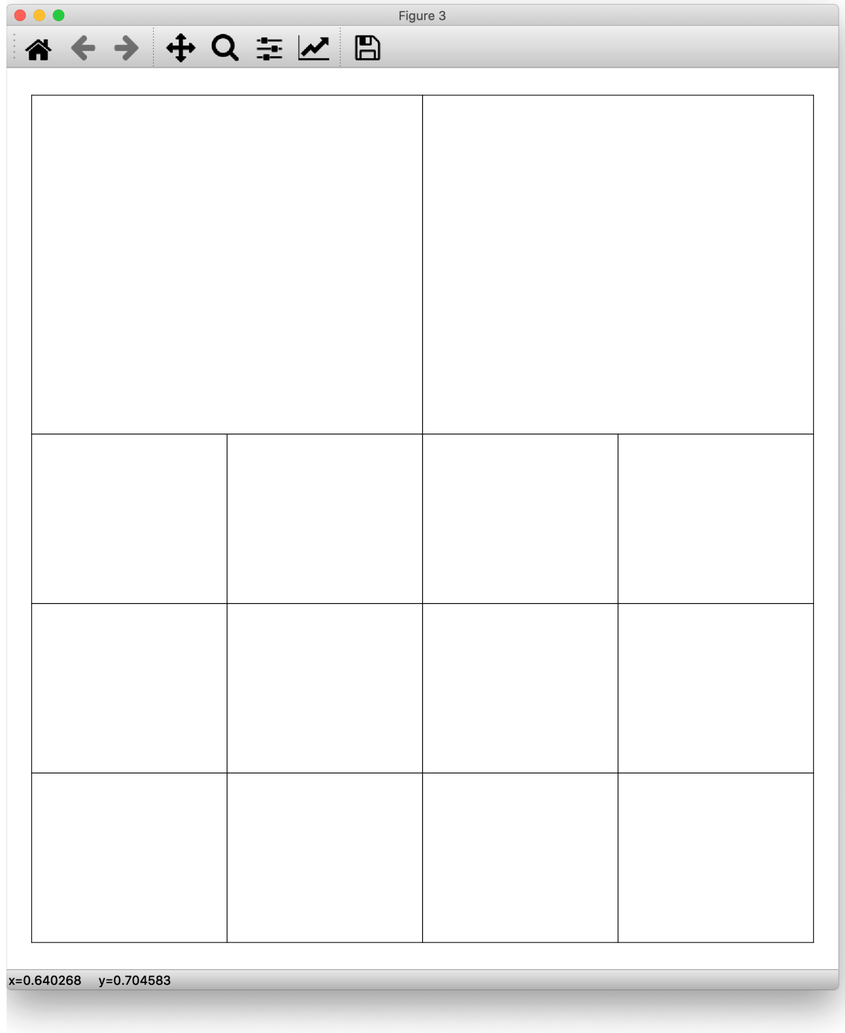
```
fig = plt.figure(figsize=(12, 13))
axes = np.empty(shape=(0, 7))

for g_idx in range(2):
    axes_g = np.empty(shape=(0,))
    main = plt.subplot2grid((5, 4), (0, 2*g_idx),
                            2, 2,
                            fig=fig)
    axes_g = np.append(axes_g, main)

    for r_idx in range(2, 2 + 3):
        for c_idx in range(2):
            ax = plt.subplot2grid((5, 4),
                                  (r_idx, c_idx + 2*g_idx),
                                  fig=fig)
            axes_g = np.append(axes_g, ax)
    axes_g = axes_g.reshape(1, -1)
    axes = np.vstack((axes, axes_g))

for ax_idx, ax in enumerate(axes.flat):
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```



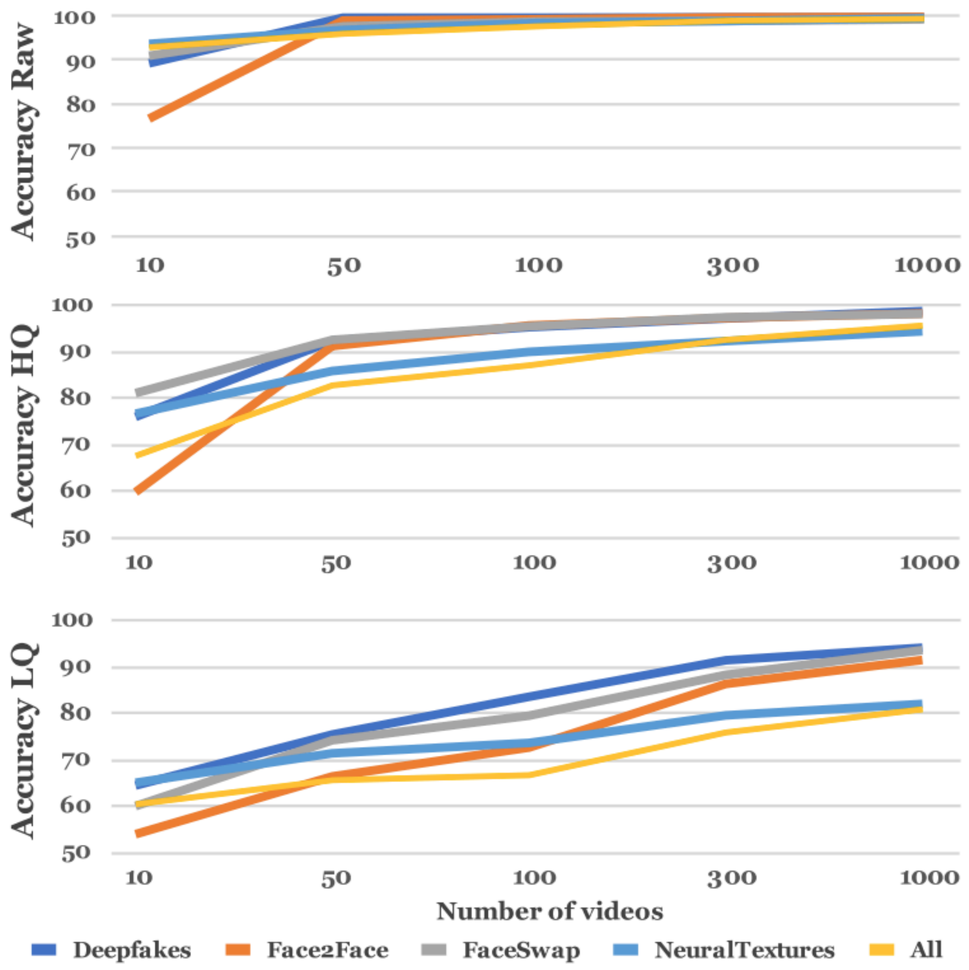
3. fig.subplots_adjust(Practice)

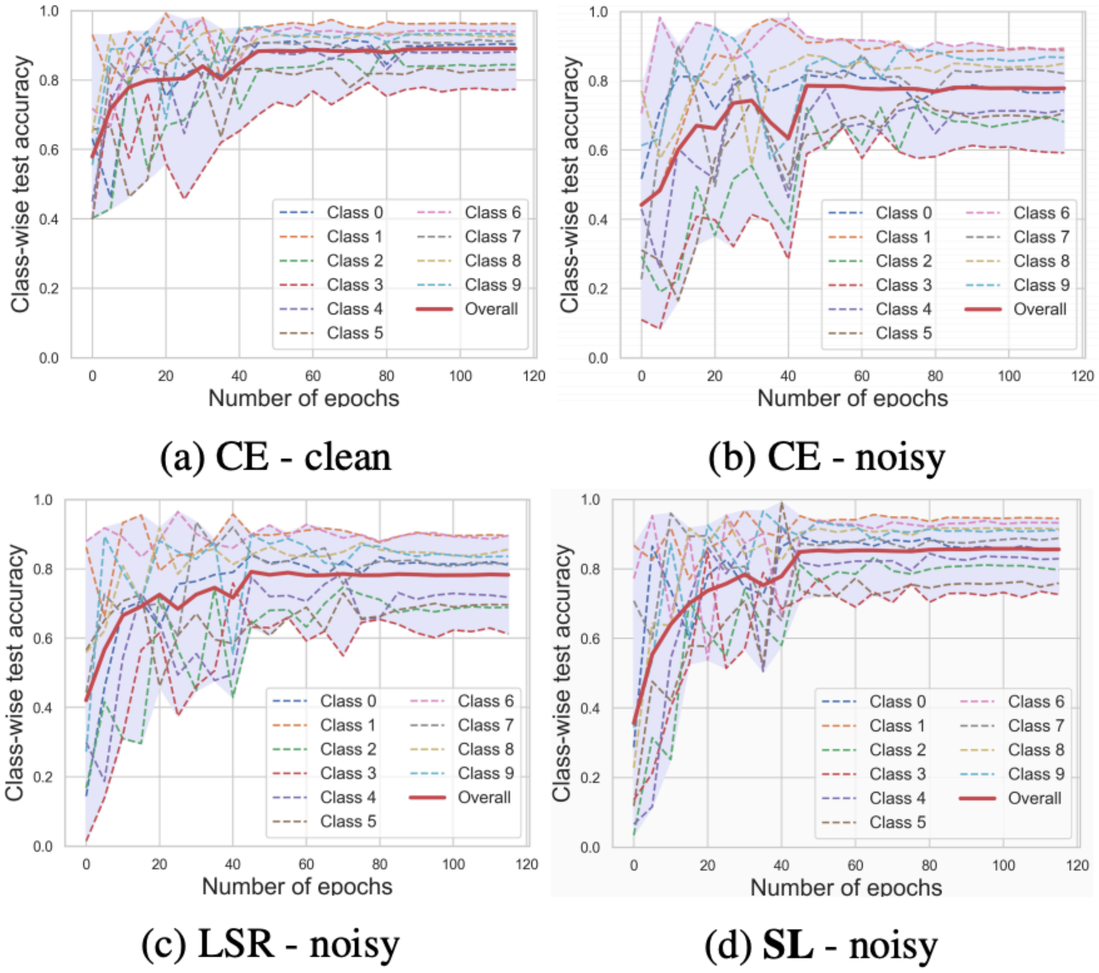
```
fig = plt.figure(figsize=(12, 13))
axes = np.empty(shape=(0, 7))

for g_idx in range(2):
    axes_g = np.empty(shape=(0,))
    main = plt.subplot2grid((5, 4), (0, 2*g_idx),
                            2, 2,
                            fig=fig)
    axes_g = np.append(axes_g, main)

    for r_idx in range(2, 2 + 3):
        for c_idx in range(2):
            ax = plt.subplot2grid((5, 4),
                                  (r_idx, c_idx + 2*g_idx),
                                  fig=fig)
            axes_g = np.append(axes_g, ax)
    axes_g = axes_g.reshape(1, -1)
    axes = np.vstack((axes, axes_g))

for ax_idx, ax in enumerate(axes.flat):
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

```
fig.subplots_adjust(bottom=0.03, top=0.97,
                    left=0.03, right=0.97,
                    hspace=0, wspace=0)
```



4. Axis Sharing





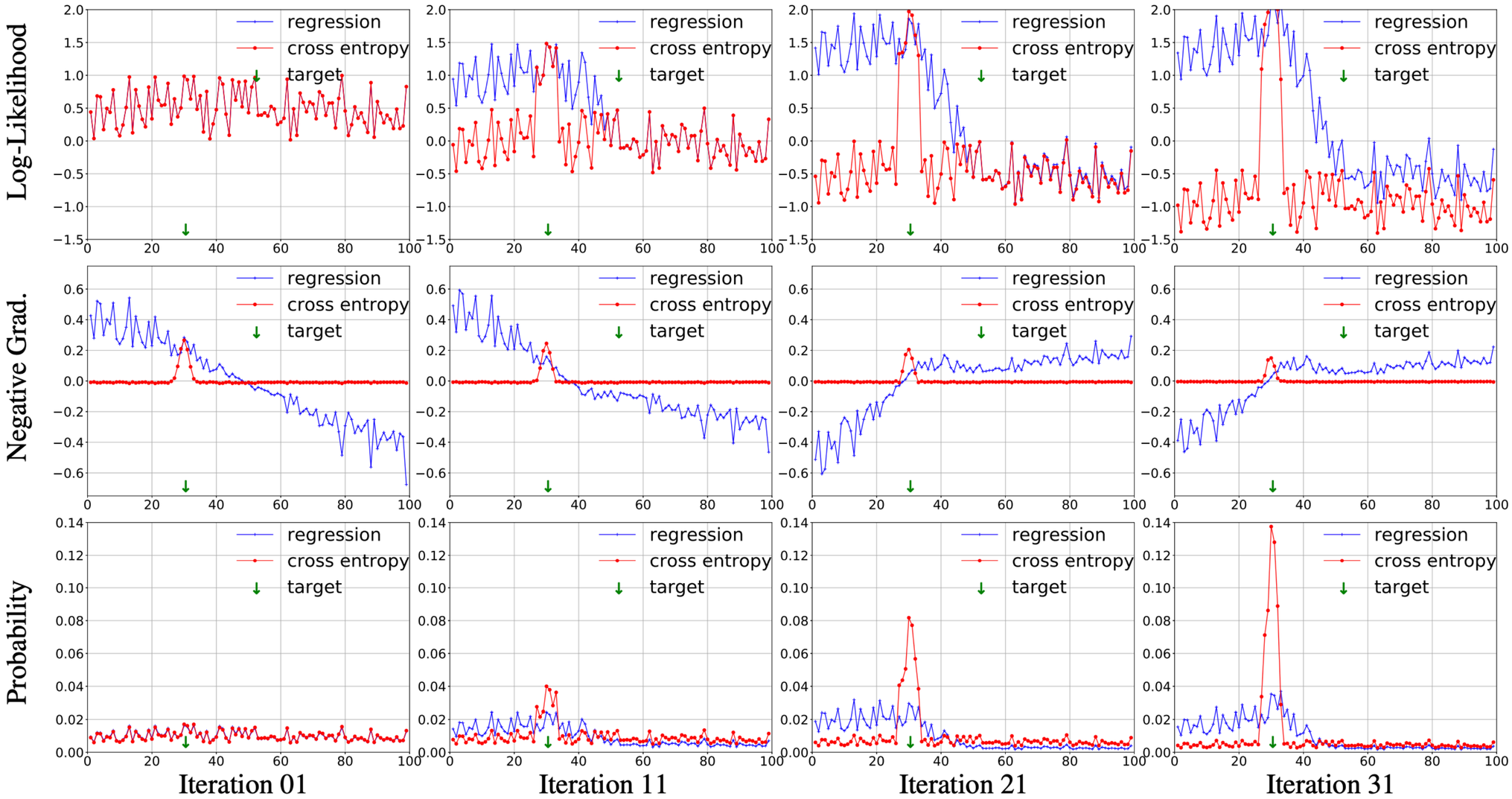
(a) CE - clean

(b) CE - noisy

(c) LSR - noisy

(d) **SL** - noisy

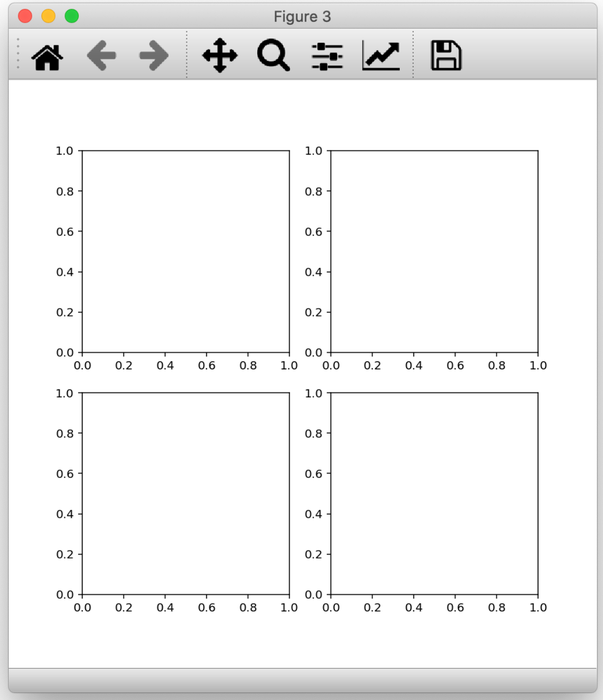
4. Axis Sharing



4. Axis Sharing

```
import matplotlib.pyplot as plt
import numpy as np

# Method 1
fig, axes = plt.subplots(2, 2,
                         figsize=(7, 7))


# Method 2
fig = plt.figure(figsize=(7, 7))
ax1_1 = fig.add_subplot(221)
ax1_2 = fig.add_subplot(222)
ax2_1 = fig.add_subplot(223)
ax2_3 = fig.add_subplot(224)


# Method 3
fig = plt.figure(figsize=(7, 7))
ax1_1 = plt.subplot2grid((2, 2), (0, 0), fig=fig)
ax1_2 = plt.subplot2grid((2, 2), (0, 1), fig=fig)
ax2_1 = plt.subplot2grid((2, 2), (1, 0), fig=fig)
ax2_2 = plt.subplot2grid((2, 2), (1, 1), fig=fig)
```

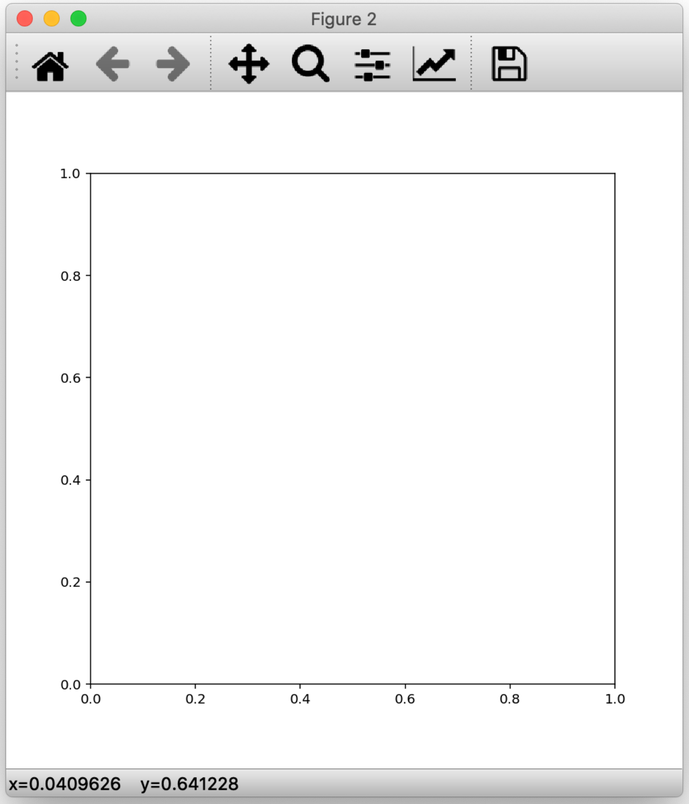
```
# Method 4
left, bottom = 0.1, 0.1
spacing = 0.1
height, width = 0.35, 0.35

rect1 = [left, bottom, width, height]
rect2 = [left+width+spacing, bottom, width, height]
rect3 = [left, bottom+height+spacing, width, height]
rect4 = [left+width+spacing,
         bottom+height+spacing,
         width, height]

fig = plt.figure(figsize=(7, 7))
ax2_1 = fig.add_axes(rect1)
ax2_2 = fig.add_axes(rect2)
ax1_1 = fig.add_axes(rect3)
ax1_2 = fig.add_axes(rect4)
```



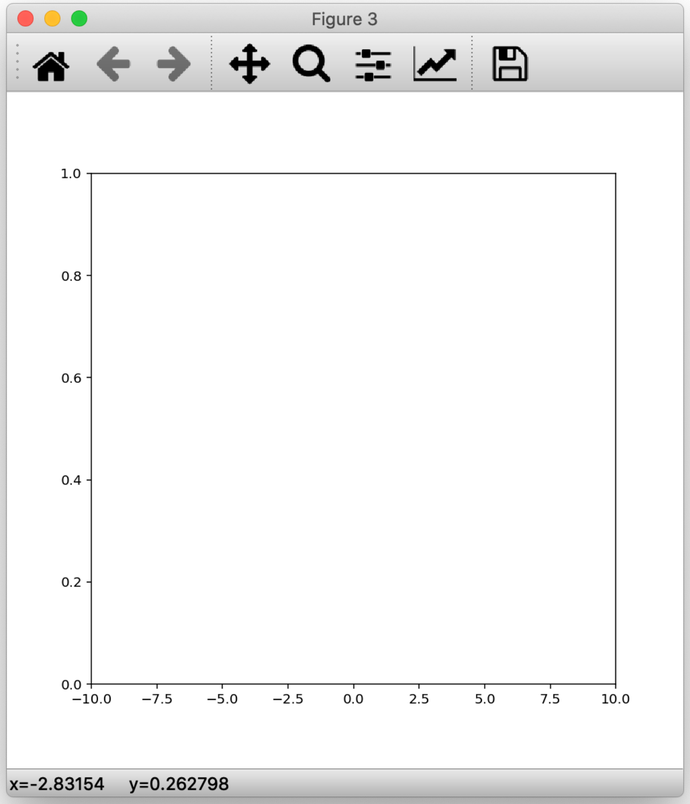
4. Axis Sharing(ax.set_xlim, ax.set_ylim)

```
import matplotlib.pyplot as plt
import numpy as np
```

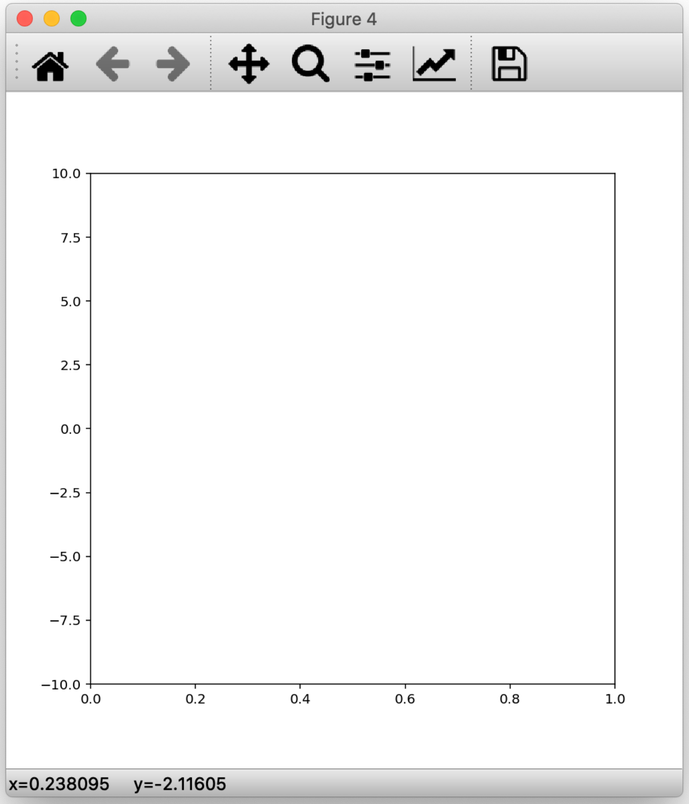
```
fig, ax = plt.subplots(figsize=(7, 7))
```



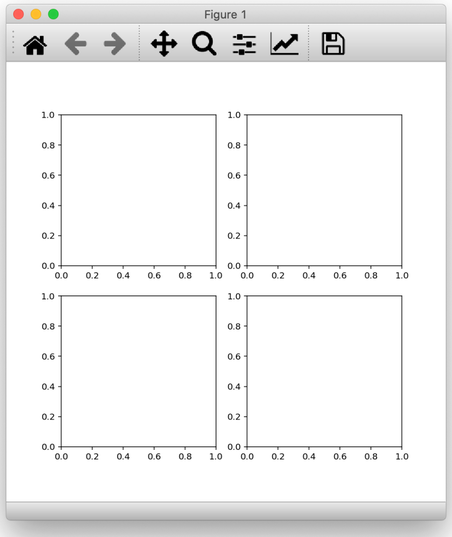

```
fig, ax = plt.subplots(figsize=(7, 7))
ax.set_xlim([-10, 10])
```

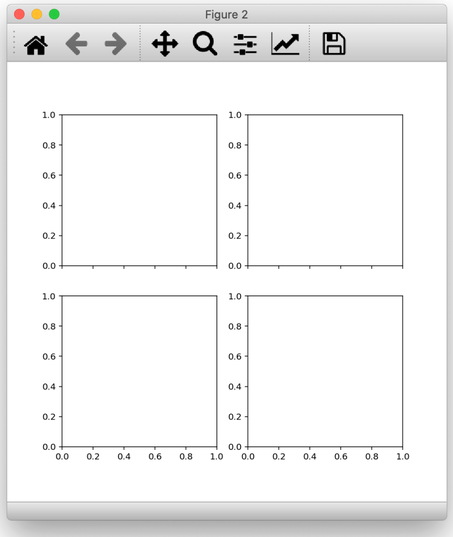


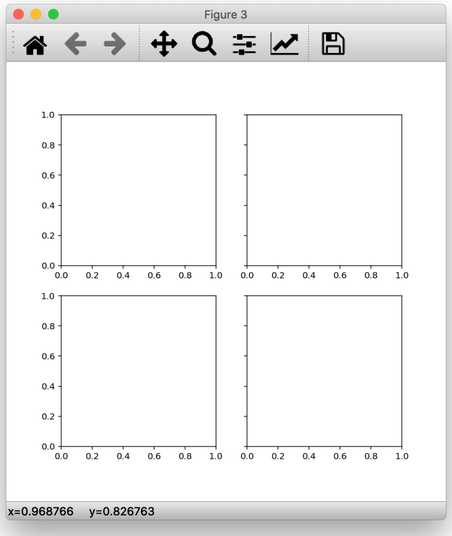

```
fig, ax = plt.subplots(figsize=(7, 7))
ax.set_ylim([-10, 10])
```

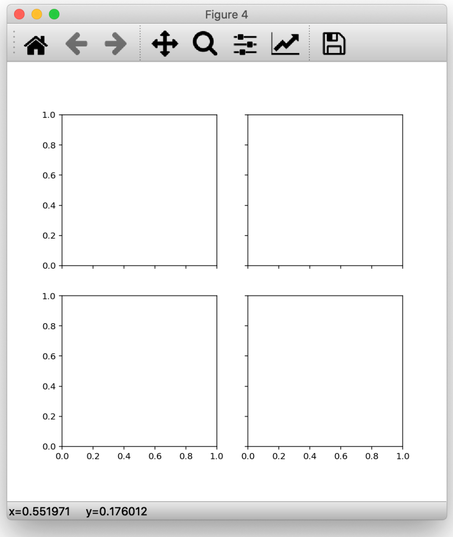


4. Axis Sharing(Using plt.subplots)









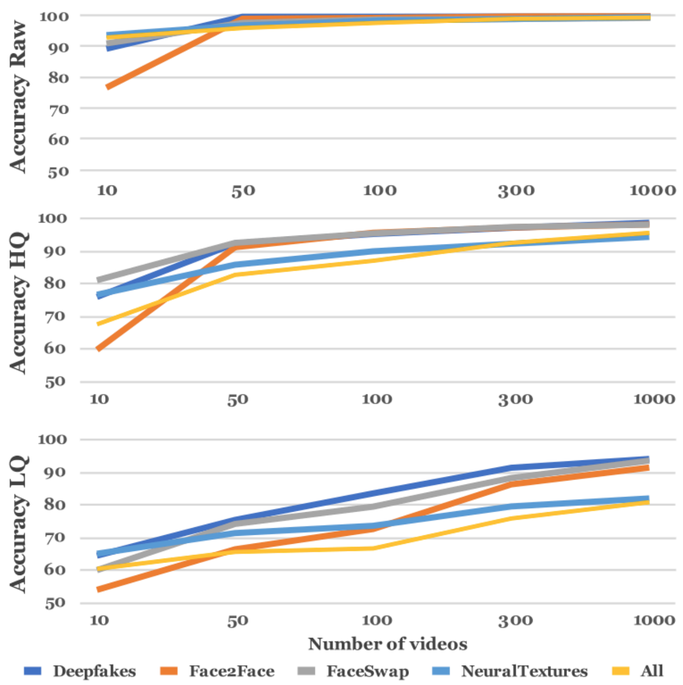

```
fig, axes = plt.subplots(2, 2,
                         figsize=(7, 7))
```

```
fig, axes = plt.subplots(2, 2,
                         figsize=(7, 7),
                         sharex=True)
```

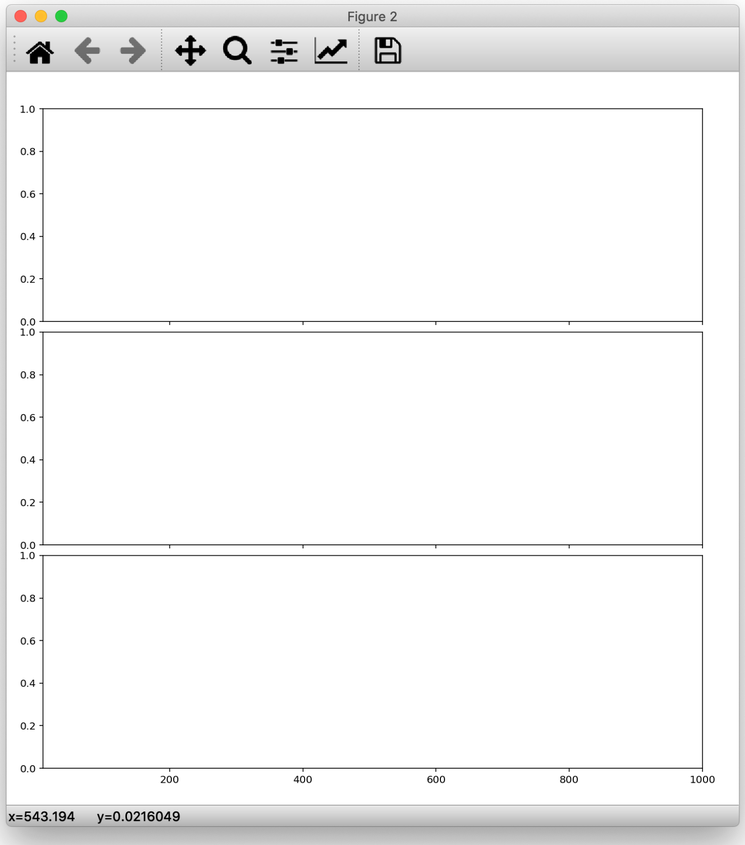
```
fig, axes = plt.subplots(2, 2,
                         figsize=(7, 7),
                         sharey=True)
```

```
fig, axes = plt.subplots(2, 2,
                         figsize=(7, 7),
                         sharex=True,
                         sharey=True)
```

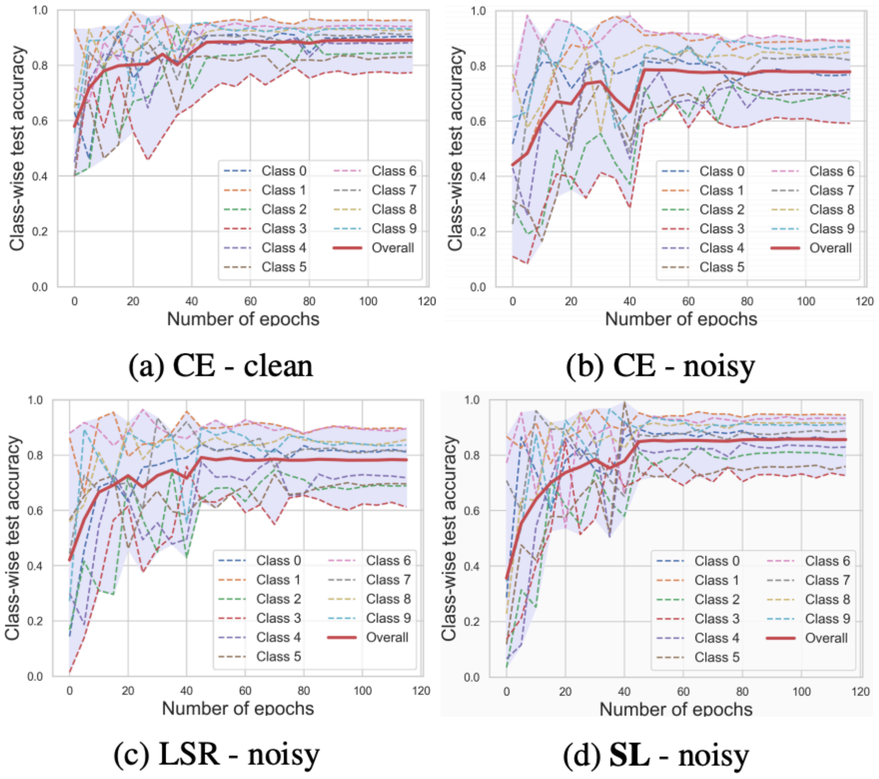
4. Axis Sharing(Using plt.subplots)




```
fig, axes = plt.subplots(3, 1,
                         figsize=(10, 10),
                         sharex=True)

fig.subplots_adjust(bottom=0.05, top=0.95,
                    left=0.05, right=0.95,
                    hspace=0.05)

axes[0].set_xlim([10, 1000])
```



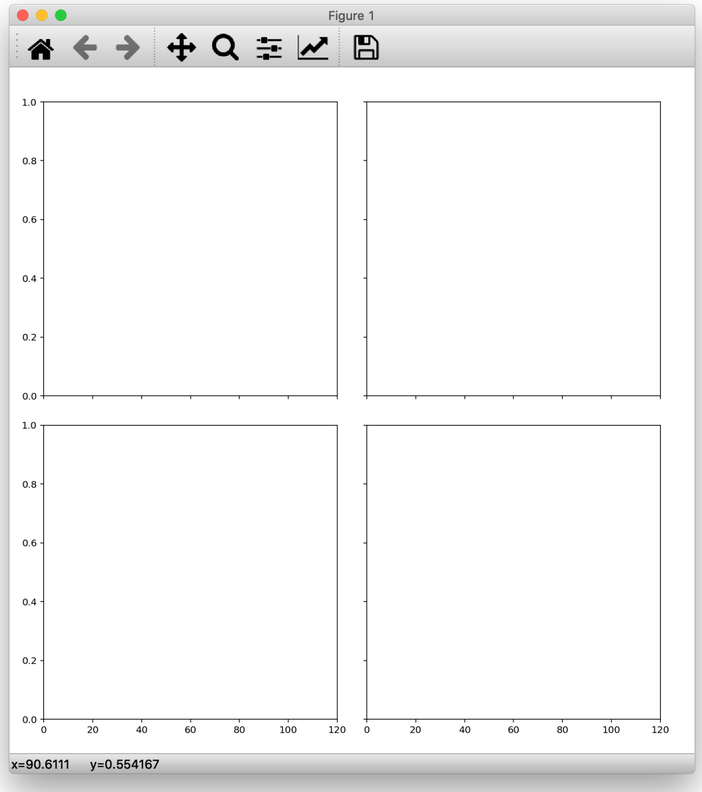
4. Axis Sharing(Using plt.subplots)



(a) CE - clean (b) CE - noisy

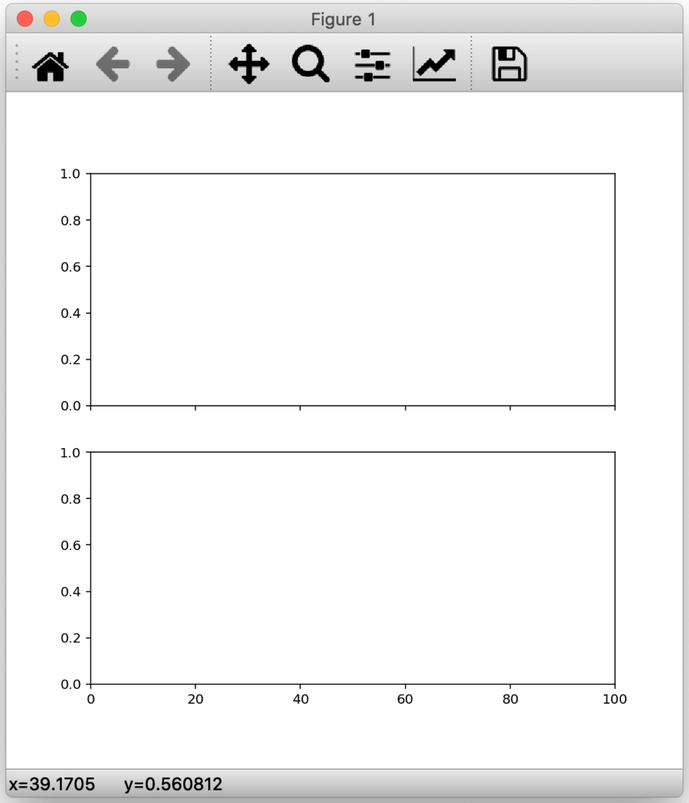
(c) LSR - noisy (d) **SL** - noisy

```
fig, axes = plt.subplots(2, 2,
                         figsize=(10, 10),
                         sharex=True,
                         sharey=True)

fig.subplots_adjust(bottom=0.05, top=0.95,
                    left=0.05, right=0.95,
                    hspace=0.1, wspace=0.1)

axes[0, 0].set_xlim([0, 120])
axes[0, 0].set_ylim([0, 1])
```

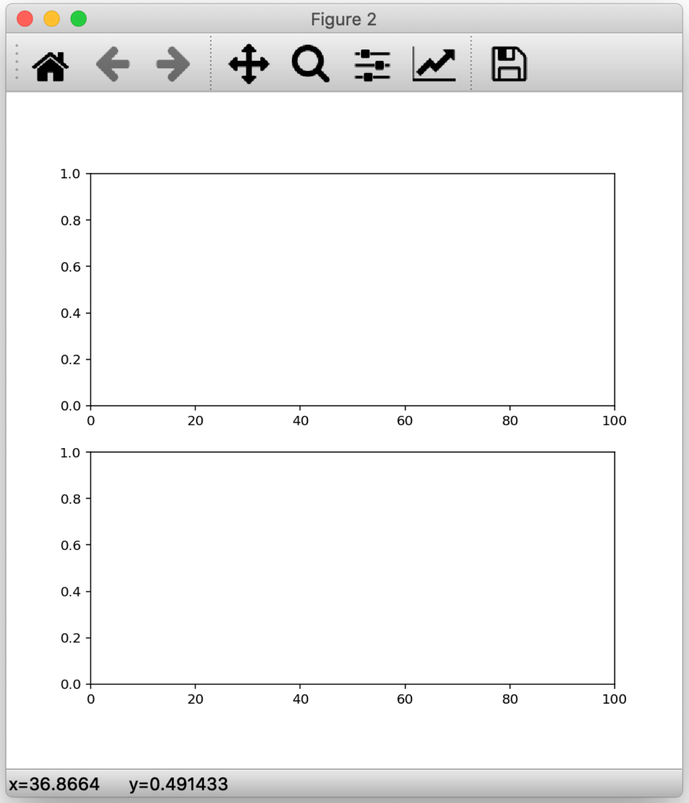


4. Axis Sharing(Using fig.add_subplot)

```
fig, axes = plt.subplots(2, 1,
                         figsize=(7, 7),
                         sharex=True)
axes[0].set_xlim([0, 100])
```

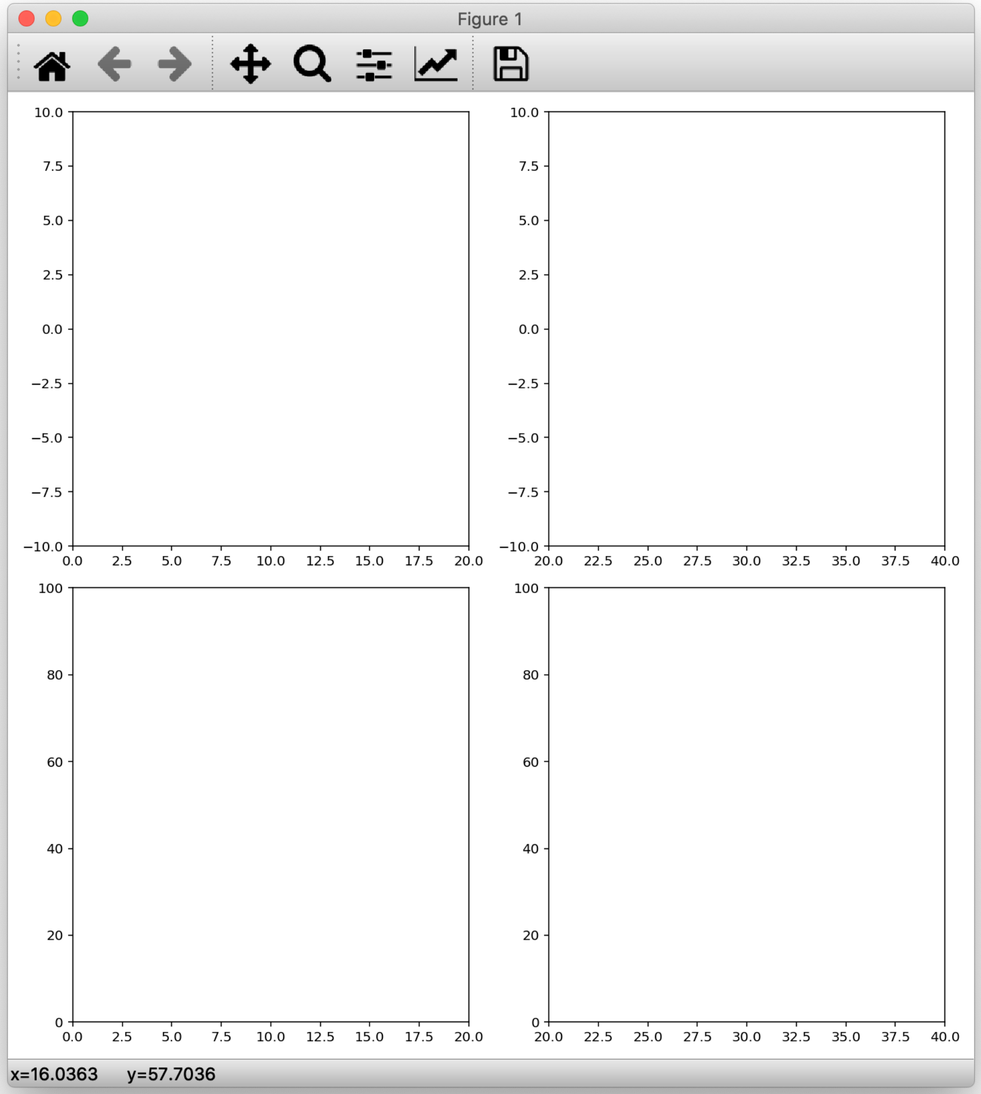



```
fig = plt.figure(figsize=(7,7))
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212, sharex=ax1)
ax2.set_xlim([0, 100])
```

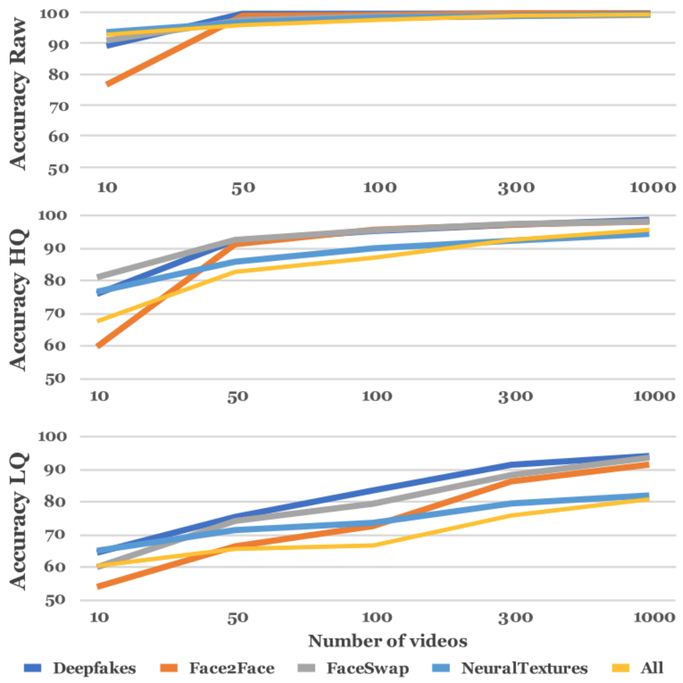


4. Axis Sharing(Using fig.add_subplot)

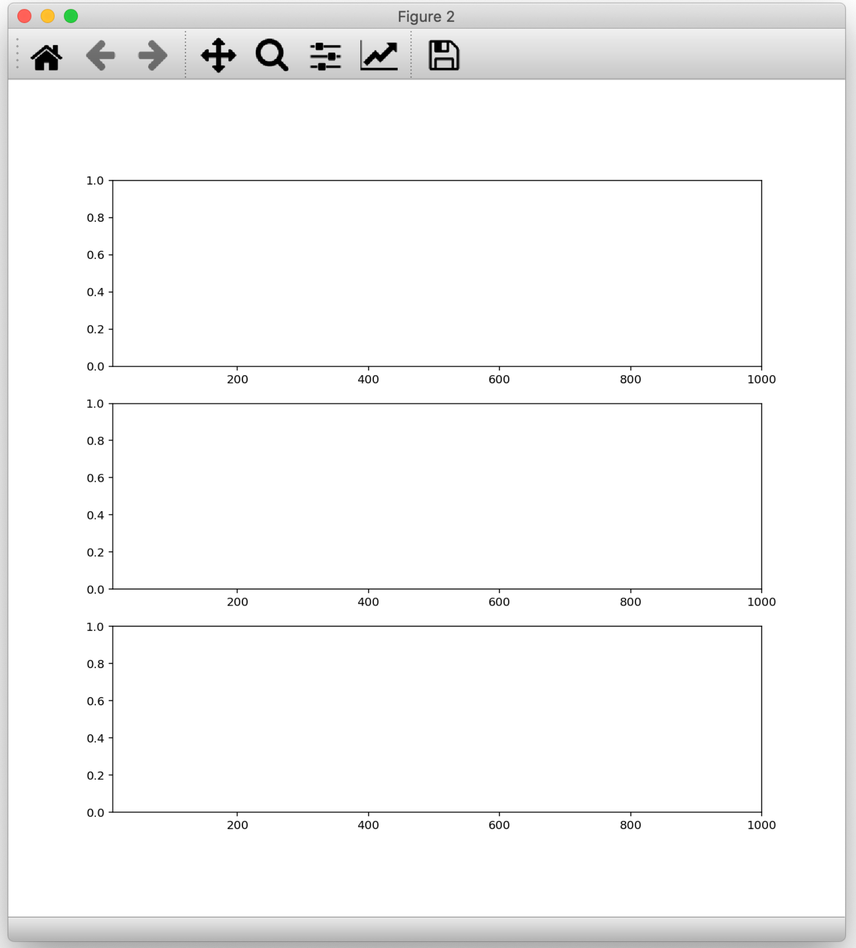
```
fig = plt.figure(figsize=(10, 10))

ax1_1 = fig.add_subplot(221)
ax1_2 = fig.add_subplot(222,
                        sharey=ax1_1)
ax2_1 = fig.add_subplot(223,
                        sharex=ax1_1)
ax2_2 = fig.add_subplot(224,
                        sharey=ax2_1,
                        sharex=ax1_2)


ax1_1.set_ylim([-10, 10])
ax2_1.set_ylim([0, 100])
ax2_1.set_xlim([0, 20])
ax2_2.set_xlim([20, 40])

fig.tight_layout()
```



4. Axis Sharing(Using fig.add_subplot)



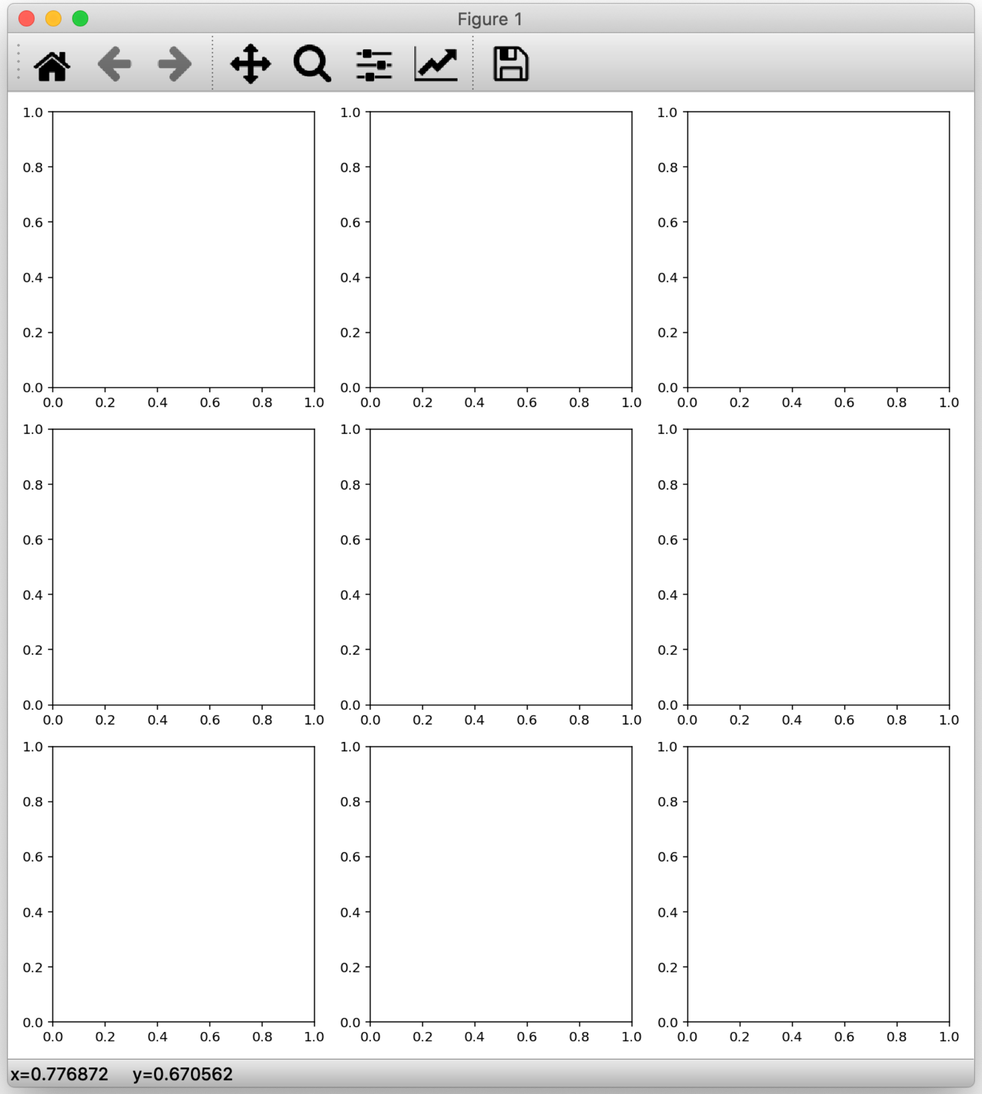

```
fig = plt.figure(figsize=(10, 10))

ax1 = fig.add_subplot(311)
ax2 = fig.add_subplot(312,
                      sharex=ax1)
ax3 = fig.add_subplot(313,
                      sharex=ax1)
ax3.set_xlim([10, 1000])
```




```
fig = plt.figure(figsize=(10, 10))
axes = np.empty(shape=(0,))

for ax_idx in range(1, 1+3):
    if ax_idx == 1:
        axes = np.append(axes,
                         fig.add_subplot(3, 1, ax_idx))
    else:
        axes = np.append(axes,
                         fig.add_subplot(3, 1, ax_idx,
                                         sharex=axes[0]))

axes[0].set_xlim([10, 1000])
```

4. Axis Sharing(Using fig.add_subplot)

```
fig = plt.figure(figsize=(10, 10))

n_row, n_col = 3, 3
axes = np.empty(shape=(0,3))

for r_idx in range(n_row):

    axes_row = np.empty(shape=(0,))
    for c_idx in range(n_col):
        ax = fig.add_subplot(n_row, n_col,
                             r_idx*n_row + c_idx + 1)
        axes_row = np.append(axes_row, ax)


    axes = np.vstack((axes, axes_row))
fig.tight_layout()
```



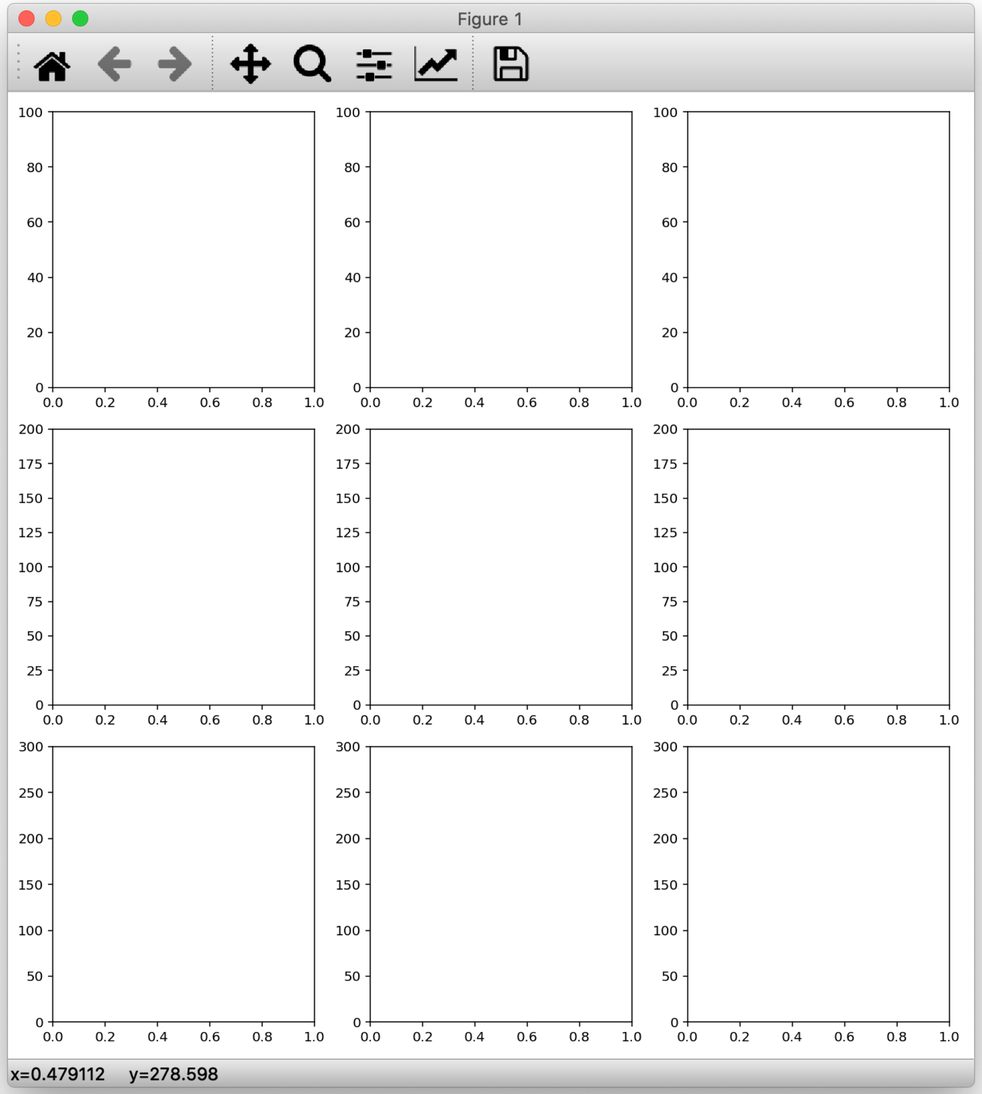
4. Axis Sharing(Using fig.add_subplot)

```
fig = plt.figure(figsize=(10, 10))

n_row, n_col = 3, 3
axes = np.empty(shape=(0,3))

for r_idx in range(n_row):

    axes_row = np.empty(shape=(0,))
    for c_idx in range(n_col):
        ax = fig.add_subplot(n_row, n_col,
                             r_idx*n_row + c_idx + 1)
        axes_row = np.append(axes_row, ax)

    axes = np.vstack((axes, axes_row))
fig.tight_layout()
```

```
fig = plt.figure(figsize=(10, 10))

n_row, n_col = 3, 3
axes = np.empty(shape=(0,3))

for r_idx in range(n_row):

    axes_row = np.empty(shape=(0,))
    for c_idx in range(n_col):
        if c_idx == 0:
            ax = fig.add_subplot(n_row, n_col,
                                 r_idx*n_row + c_idx + 1)
        else:
            ax = fig.add_subplot(n_row, n_col,
                                 r_idx*n_row + c_idx + 1,
                                 sharey=axes_row[0])
        axes_row = np.append(axes_row, ax)
    axes = np.vstack((axes, axes_row))

fig.tight_layout()

axes[0, 0].set_ylim([0, 100])
axes[1, 0].set_ylim([0, 200])
axes[2, 0].set_ylim([0, 300])
```

4. Axis Sharing(Using fig.add_subplot)

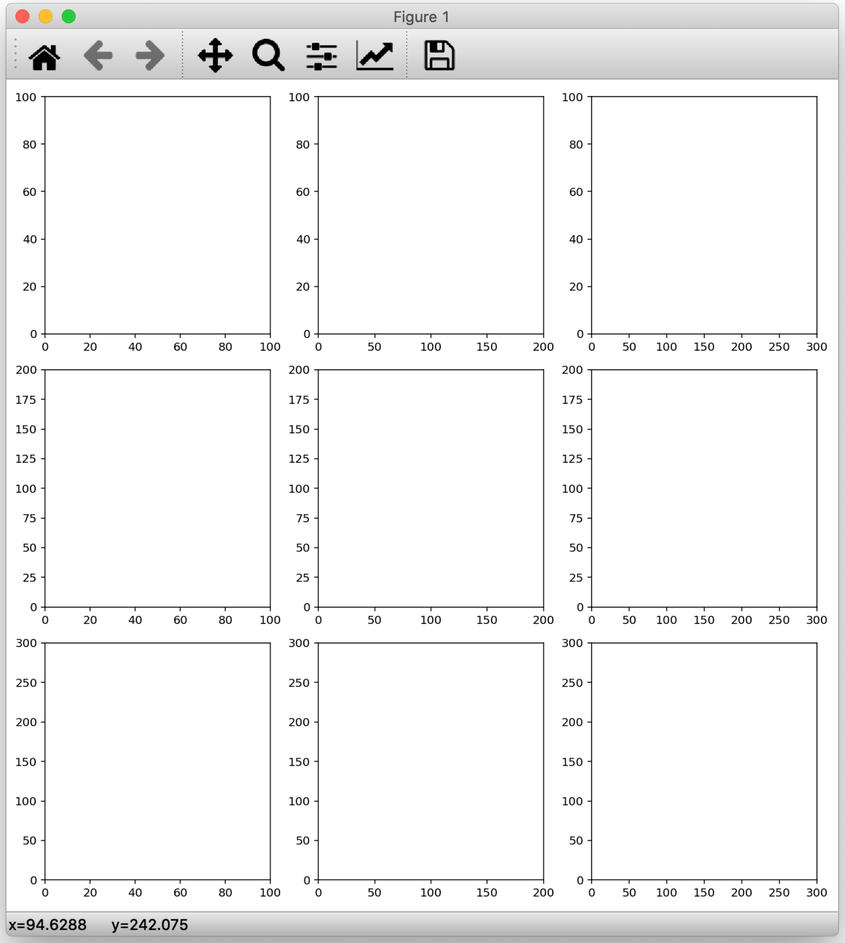
```
fig = plt.figure(figsize=(10, 10))

n_row, n_col = 3, 3
axes = np.empty(shape=(0,3))

for r_idx in range(n_row):

    axes_row = np.empty(shape=(0,))
    for c_idx in range(n_col):
        if c_idx == 0:
            ax = fig.add_subplot(n_row, n_col,
                                 r_idx*n_row + c_idx + 1)
        else:
            ax = fig.add_subplot(n_row, n_col,
                                 r_idx*n_row + c_idx + 1,
                                 sharey=axes_row[0])
        axes_row = np.append(axes_row, ax)
    axes = np.vstack((axes, axes_row))

fig.tight_layout()

axes[0, 0].set_ylim([0, 100])
axes[1, 0].set_ylim([0, 200])
axes[2, 0].set_ylim([0, 300])
```



4. Axis Sharing(Using fig.add_subplot)

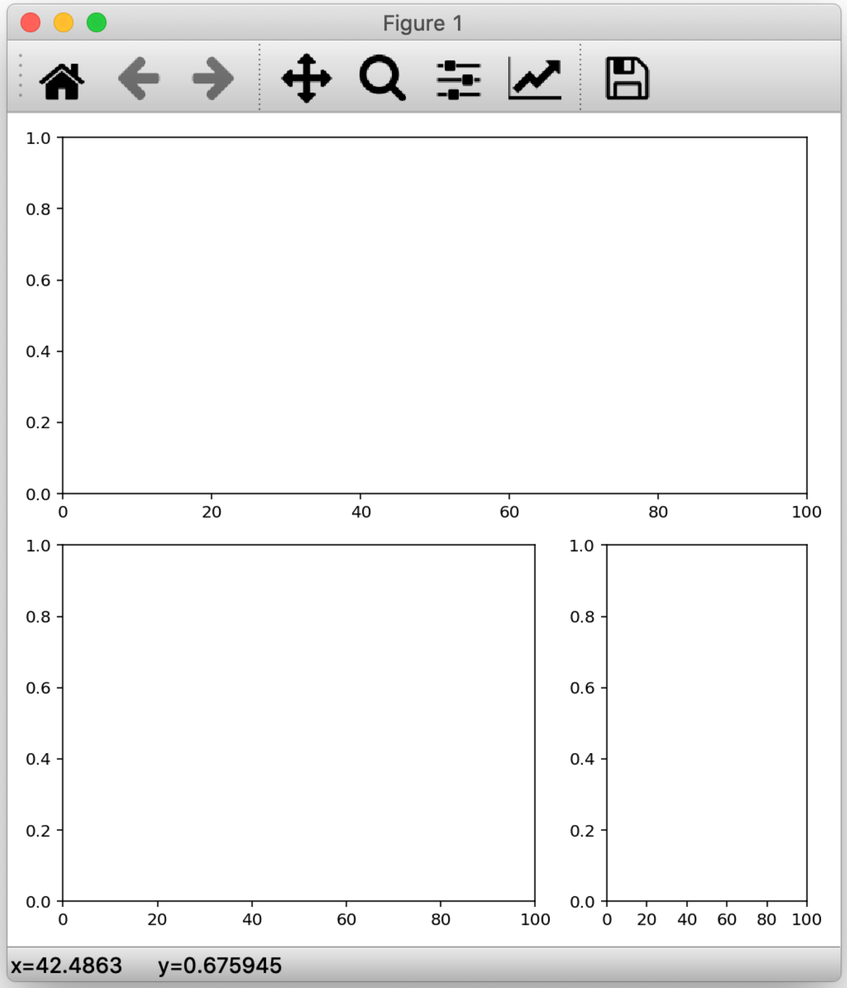
```
fig = plt.figure(figsize=(10, 10))

n_row, n_col = 3, 3
axes = np.empty(shape=(0,3))
for r_idx in range(n_row):
    axes_row = np.empty(shape=(0,))

    if r_idx == 0:
        for c_idx in range(n_col):
            if c_idx == 0:
                ax = fig.add_subplot(n_row, n_col,
                                     r_idx*n_row + c_idx + 1)
            else:
                ax = fig.add_subplot(n_row, n_col,
                                     r_idx*n_row + c_idx + 1,
                                     sharey=axes_row[0])
            axes_row = np.append(axes_row, ax)
    else:
        for c_idx in range(n_col):
            if c_idx == 0:
                ax = fig.add_subplot(n_row, n_col,
                                     r_idx*n_row + c_idx + 1,
                                     sharex=axes[0, c_idx])
            else:
                ax = fig.add_subplot(n_row, n_col,
                                     r_idx*n_row + c_idx + 1,
                                     sharey=axes_row[0],
                                     sharex=axes[0, c_idx])
             axes_row = np.append(axes_row, ax)

    axes = np.vstack((axes, axes_row))
```

```
fig.tight_layout()

axes[0, 0].set_ylim([0, 100])
axes[1, 0].set_ylim([0, 200])
axes[2, 0].set_ylim([0, 300])
axes[0, 0].set_xlim([0, 100])
axes[0, 1].set_xlim([0, 200])
axes[0, 2].set_xlim([0, 300])
```

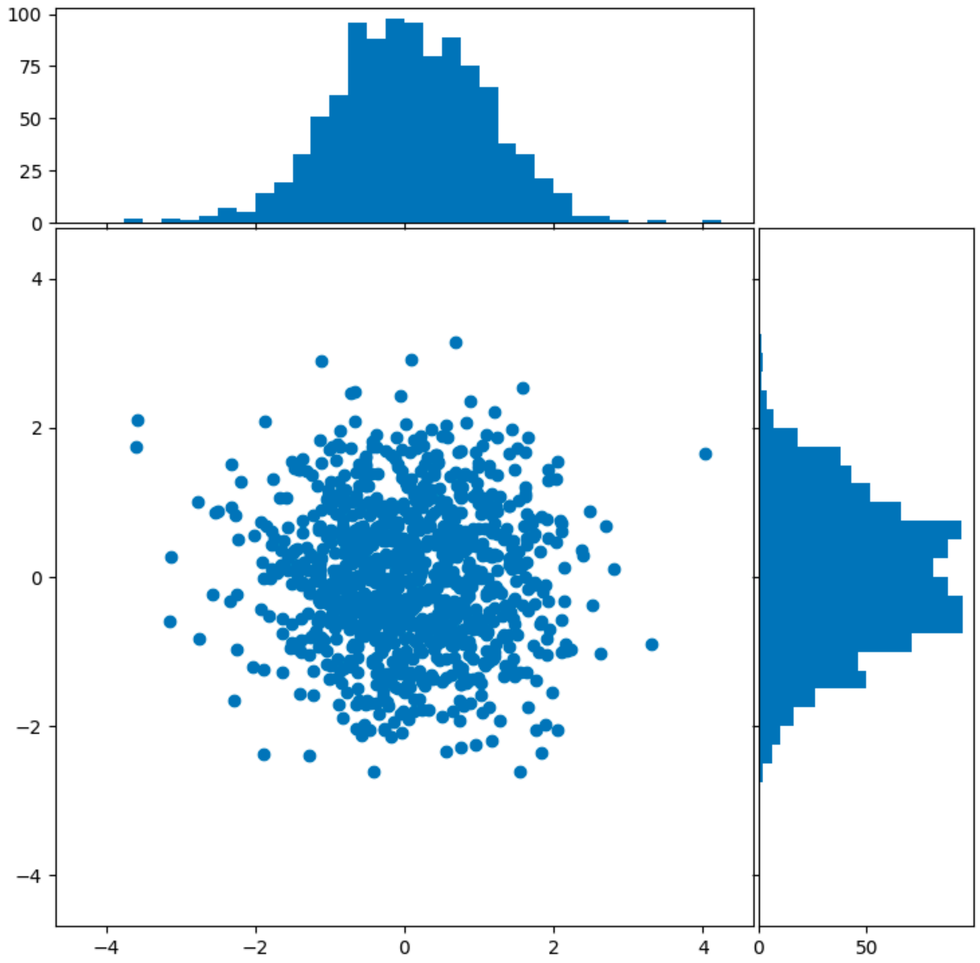


4. Axis Sharing(Using plt.subplot2grid)

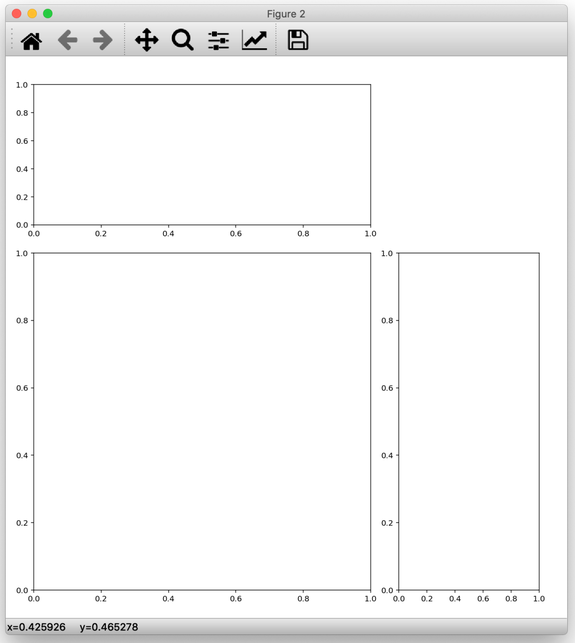
```
fig = plt.figure(figsize=(7, 7))

ax1_1 = plt.subplot2grid((2, 3), (0, 0),
                         colspan=3,
                         fig=fig)
ax2_1 = plt.subplot2grid((2, 3), (1, 0),
                         colspan=2,
                         fig=fig,
                         sharex=ax1_1)
ax2_2 = plt.subplot2grid((2, 3), (1, 2),
                         fig=fig,
                         sharex=ax1_1)

ax1_1.set_xlim([0, 100])
fig.tight_layout()
```



4. Axis Sharing(Using fig.add_axes)

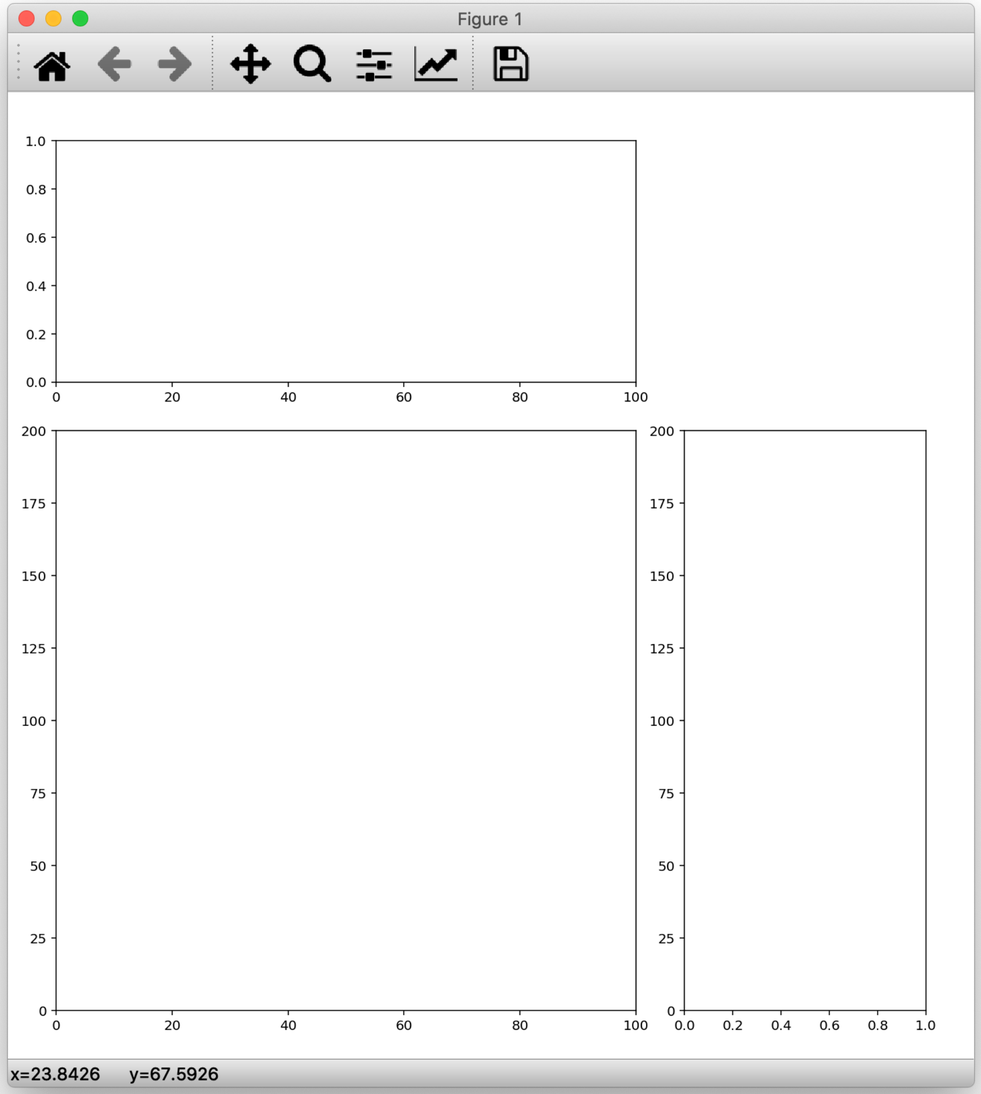



```
fig = plt.figure(figsize=(10, 10))

left, bottom = 0.05, 0.05
spacing = 0.05
width1, height1 = 0.6, 0.6
width2 = 1 - (2*left + spacing + width1)
height2 = 1 - (2*bottom + spacing + height1)

rect1 = [left, bottom, width1, height1]
rect2 = [left, bottom + height1 + spacing, width1, height2]
rect3 = [left + width1 + spacing, bottom, width2, height1]

ax1 = fig.add_axes(rect1)
ax2 = fig.add_axes(rect2)
ax3 = fig.add_axes(rect3)
```



4. Axis Sharing(Using plt.subplot2grid)

```
fig = plt.figure(figsize=(10, 10))

left, bottom = 0.05, 0.05
spacing = 0.05
width1, height1 = 0.6, 0.6
width2 = 1 - (2*left + spacing + width1)
height2 = 1 - (2*bottom + spacing + height1)

rect1 = [left, bottom, width1, height1]
rect2 = [left, bottom + height1 + spacing, width1, height2]
rect3 = [left + width1 + spacing, bottom, width2, height1]


ax1 = fig.add_axes(rect1)
ax2 = fig.add_axes(rect2,
                   sharex=ax1)
ax3 = fig.add_axes(rect3,
                   sharey=ax1)


ax1.set_xlim([0, 100])
ax1.set_ylim([0, 200])
```



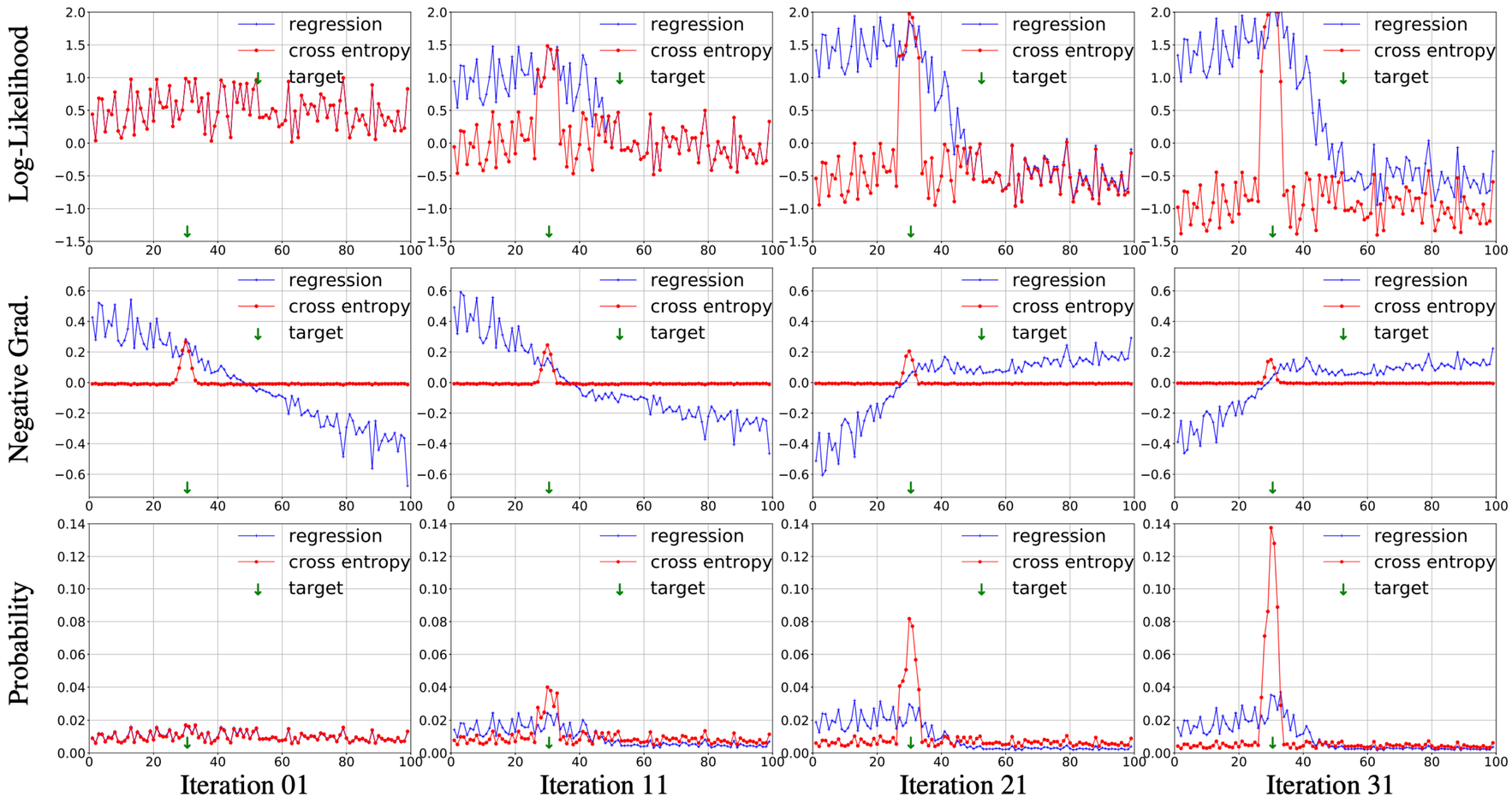
5. Axis Sharing(Practice)

On the Over-Smoothing Problem of CNN Based Disparity Estimation

Chuangrong Chen
Sun Yat-sen University
chenchr5@mail2.sysu.edu.cn

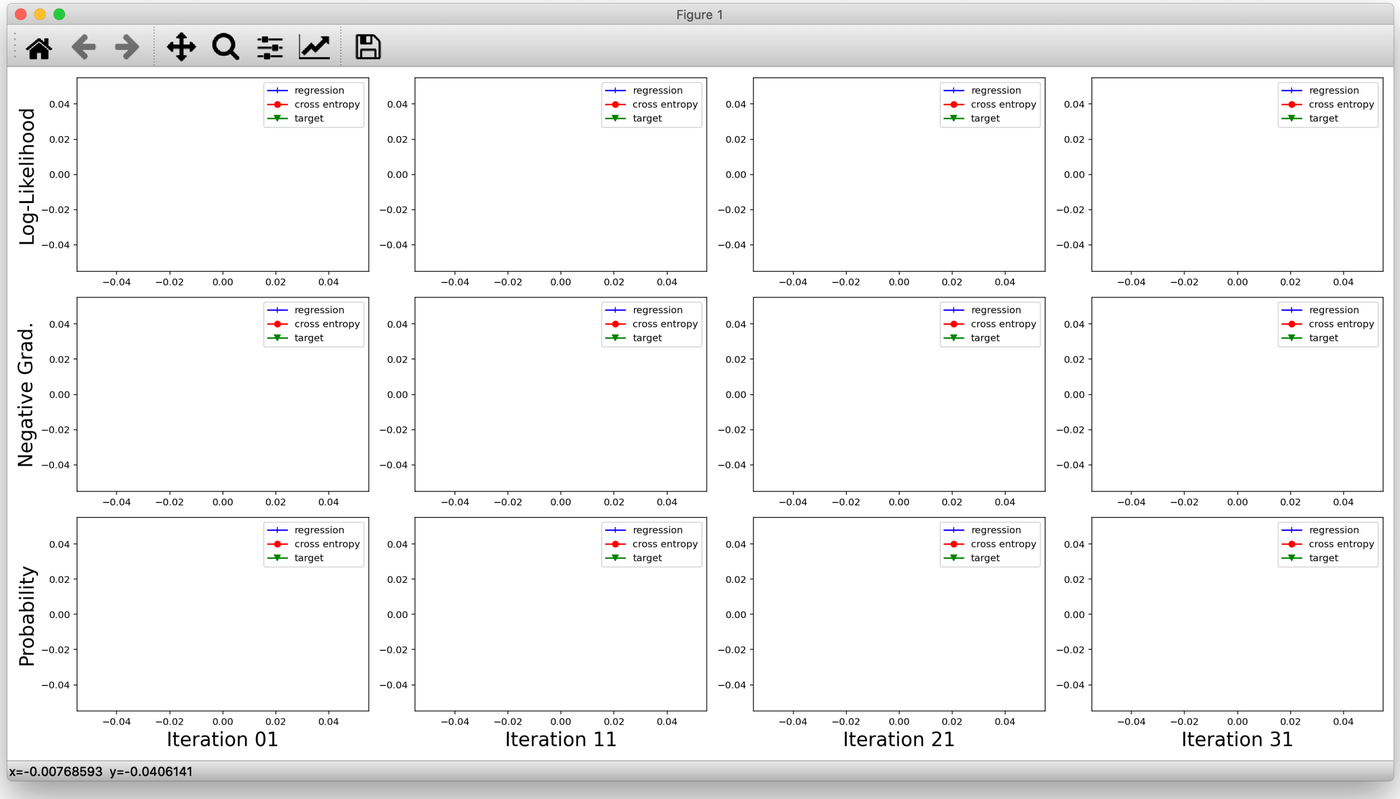
Xiaozhi Chen
DJI
cxz.thu@gmail.com

Hui Cheng*
Sun Yat-sen University
chengh9@mail.sysu.edu.cn



5. Axis Sharing(Practice)

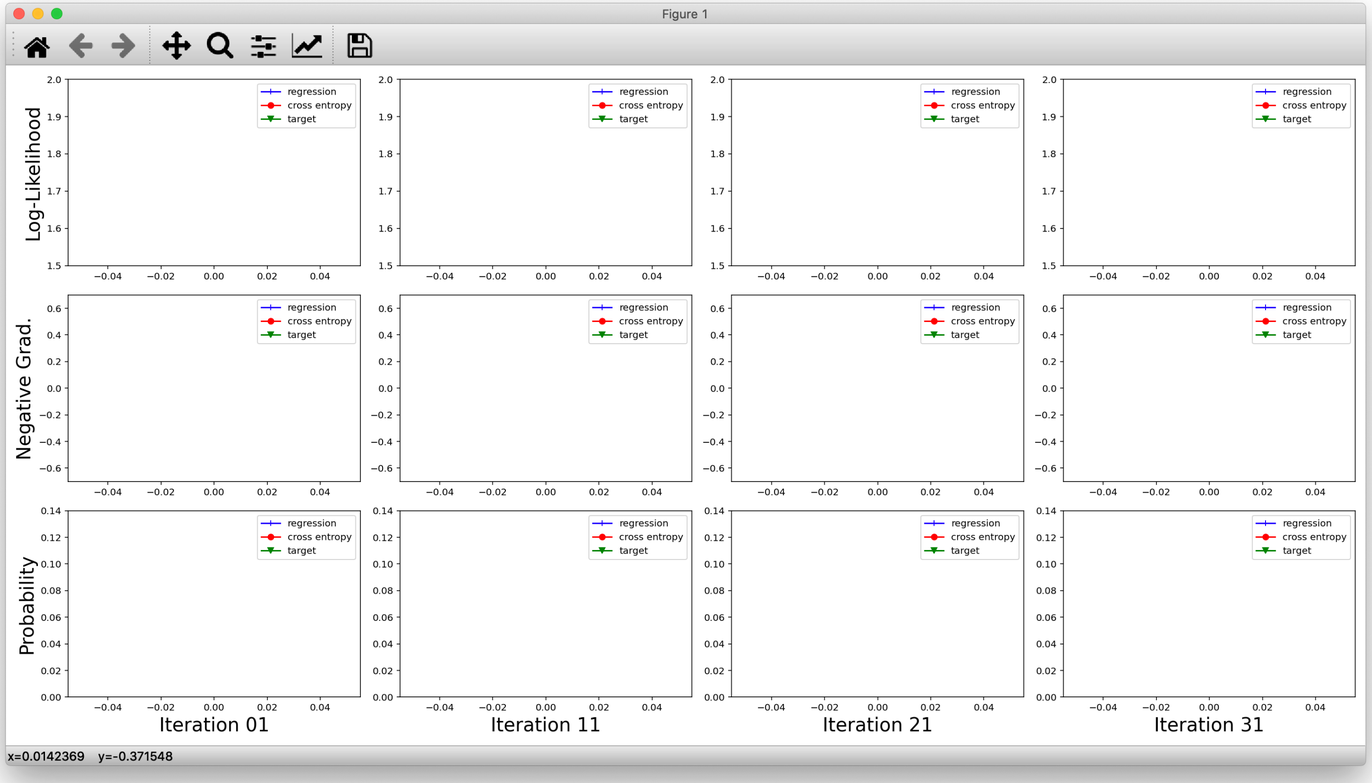
```
fig = plt.figure(figsize=(20, 10))

n_row, n_col = 3, 4
axes = np.empty(shape=(0, n_col))

xlabel_list = ['Iteration 01', 'Iteration 11', 'Iteration 21', 'Iteration 31']
ylabel_list = ['Log-Likelihood', 'Negative Grad.', 'Probability']

for r_idx in range(n_row):

    axes_row = np.empty(shape=(0,))
    for c_idx in range(n_col):
        ax = fig.add_subplot(n_row, n_col,
                             n_col*r_idx + c_idx + 1)

        # plotting
        ax.plot([], marker='|', color='b', label='regression')
        ax.plot([], marker='o', color='r', label='cross entropy')
        ax.plot([], marker='v', color='g', label='target')
        ax.legend()
        axes_row = np.append(axes_row, ax)

    axes = np.vstack((axes, axes_row))

# set xylabels
for ax_idx, ax in enumerate(axes.flat):
    if ax_idx % n_col == 0:
        ax.set_ylabel(ylabel_list[ax_idx // n_col],
                      fontsize=20)
    if ax_idx >= 2*n_col:
        ax.set_xlabel(xlabel_list[ax_idx - 2*n_col],
                      fontsize=20)
fig.tight_layout()
```



5. Axis Sharing(Practice)

```
fig = plt.figure(figsize=(20, 10))

n_row, n_col = 3, 4
axes = np.empty(shape=(0, n_col))

xlabel_list = ['Iteration 01', 'Iteration 11', 'Iteration 21', 'Iteration 31']
ylabel_list = ['Log-Likelihood', 'Negative Grad.', 'Probability']

for r_idx in range(n_row):
    axes_row = np.empty(shape=(0,))
    for c_idx in range(n_col):
        if c_idx == 0:
            ax = fig.add_subplot(n_row, n_col,
                                 n_col*r_idx + c_idx + 1)
        else:
            ax = fig.add_subplot(n_row, n_col,
                                 n_col*r_idx + c_idx + 1,
                                 sharey=axes_row[0])
        # plotting
        ax.plot([], marker='|', color='b', label='regression')
        ax.plot([], marker='o', color='r', label='cross entropy')
        ax.plot([], marker='v', color='g', label='target')
        ax.legend()

        axes_row = np.append(axes_row, ax)
    axes = np.vstack((axes, axes_row))

axes[0, 0].set_ylim([1.5, 2.0])
axes[1, 0].set_ylim([-0.7, 0.7])
axes[2, 0].set_ylim([0, 0.14])
```

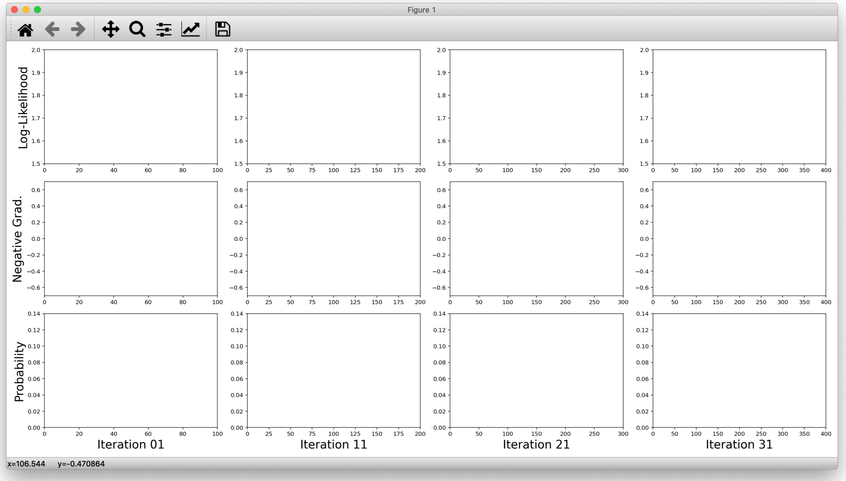
```
# set xylabels
for ax_idx, ax in enumerate(axes.flat):
    if ax_idx % n_col == 0:
        ax.set_ylabel(ylabel_list[ax_idx // n_col],
                      fontsize=20)
    if ax_idx >= 2*n_col:
        ax.set_xlabel(xlabel_list[ax_idx - 2*n_col],
                      fontsize=20)

fig.tight_layout()
```



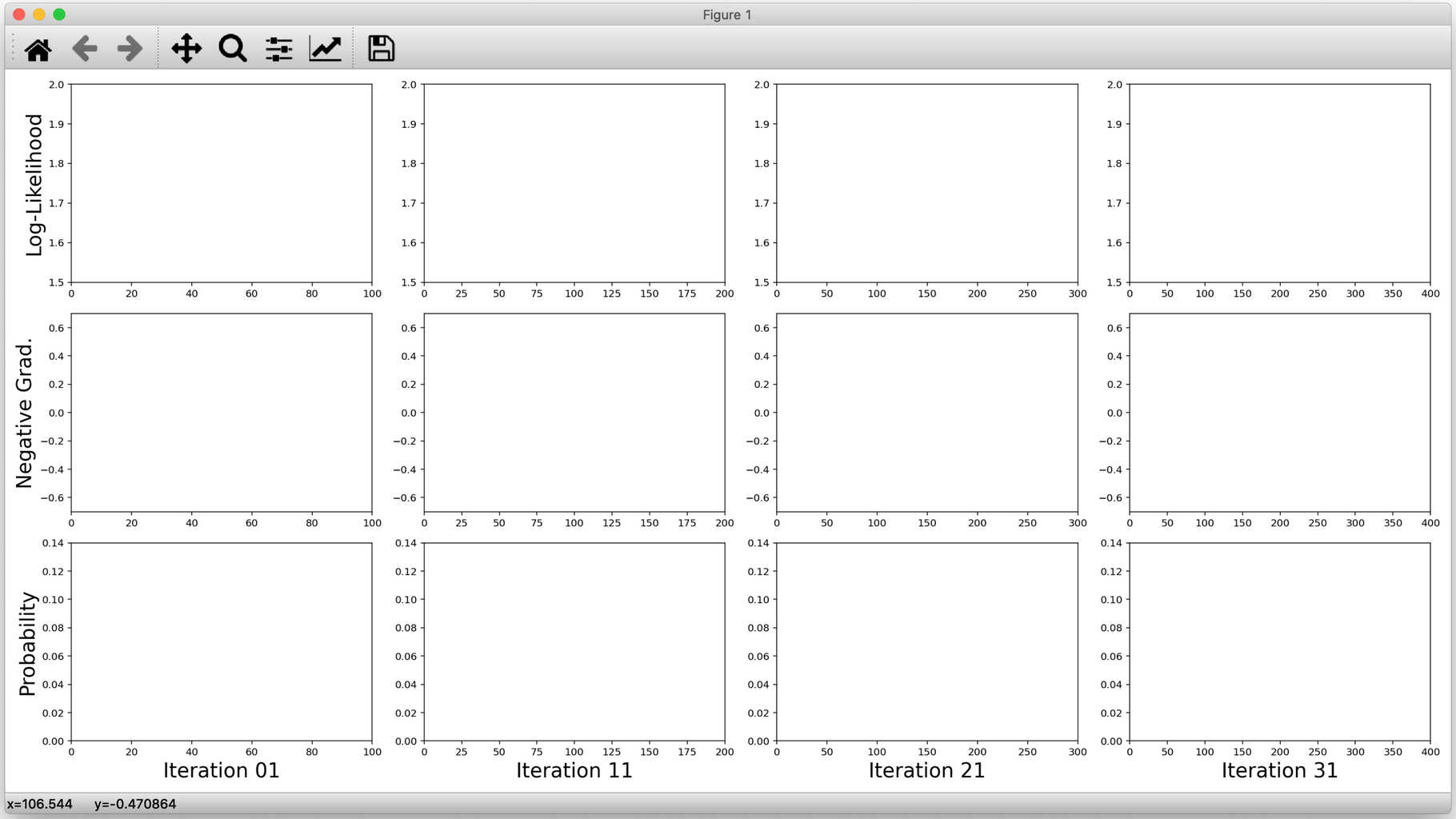
5. Axis Sharing(Practice)

```
fig = plt.figure(figsize=(20, 10))

n_row, n_col = 3, 4
axes = np.empty(shape=(0, n_col))

xlabel_list = ['Iteration 01', 'Iteration 11', 'Iteration 21', 'Iteration 31']
ylabel_list = ['Log-Likelihood', 'Negative Grad.', 'Probability']

for r_idx in range(n_row):
    axes_row = np.empty(shape=(0,))

    if r_idx == 0:
        for c_idx in range(n_col):
            if c_idx == 0:
                ax = fig.add_subplot(n_row, n_col,
                                     n_col*r_idx + c_idx + 1)
            else:
                ax = fig.add_subplot(n_row, n_col,
                                     n_col*r_idx + c_idx + 1,
                                     sharey=axes_row[0])
            axes_row = np.append(axes_row, ax)
    else:
        for c_idx in range(n_col):
            if c_idx == 0:
                ax = fig.add_subplot(n_row, n_col,
                                     n_col*r_idx + c_idx + 1,
                                     sharex=axes[0, c_idx])
            else:
                ax = fig.add_subplot(n_row, n_col,
                                     n_col*r_idx + c_idx + 1,
                                     sharey=axes_row[0],
                                     sharex=axes[0, c_idx])
            axes_row = np.append(axes_row, ax)
    axes = np.vstack((axes, axes_row))
```

```
axes[0, 0].set_ylim([1.5, 2.0])
axes[1, 0].set_ylim([-0.7, 0.7])
axes[2, 0].set_ylim([0, 0.14])
axes[0, 0].set_xlim([0, 100])
axes[0, 1].set_xlim([0, 200])
axes[0, 2].set_xlim([0, 300])
axes[0, 3].set_xlim([0, 400])

for ax_idx, ax in enumerate(axes.flat):
    if ax_idx % n_col == 0:
        ax.set_ylabel(ylabel_list[ax_idx // n_col],
                      fontsize=20)
    if ax_idx >= 2*n_col:
        ax.set_xlabel(xlabel_list[ax_idx - 2*n_col],
                      fontsize=20)
fig.tight_layout()
```

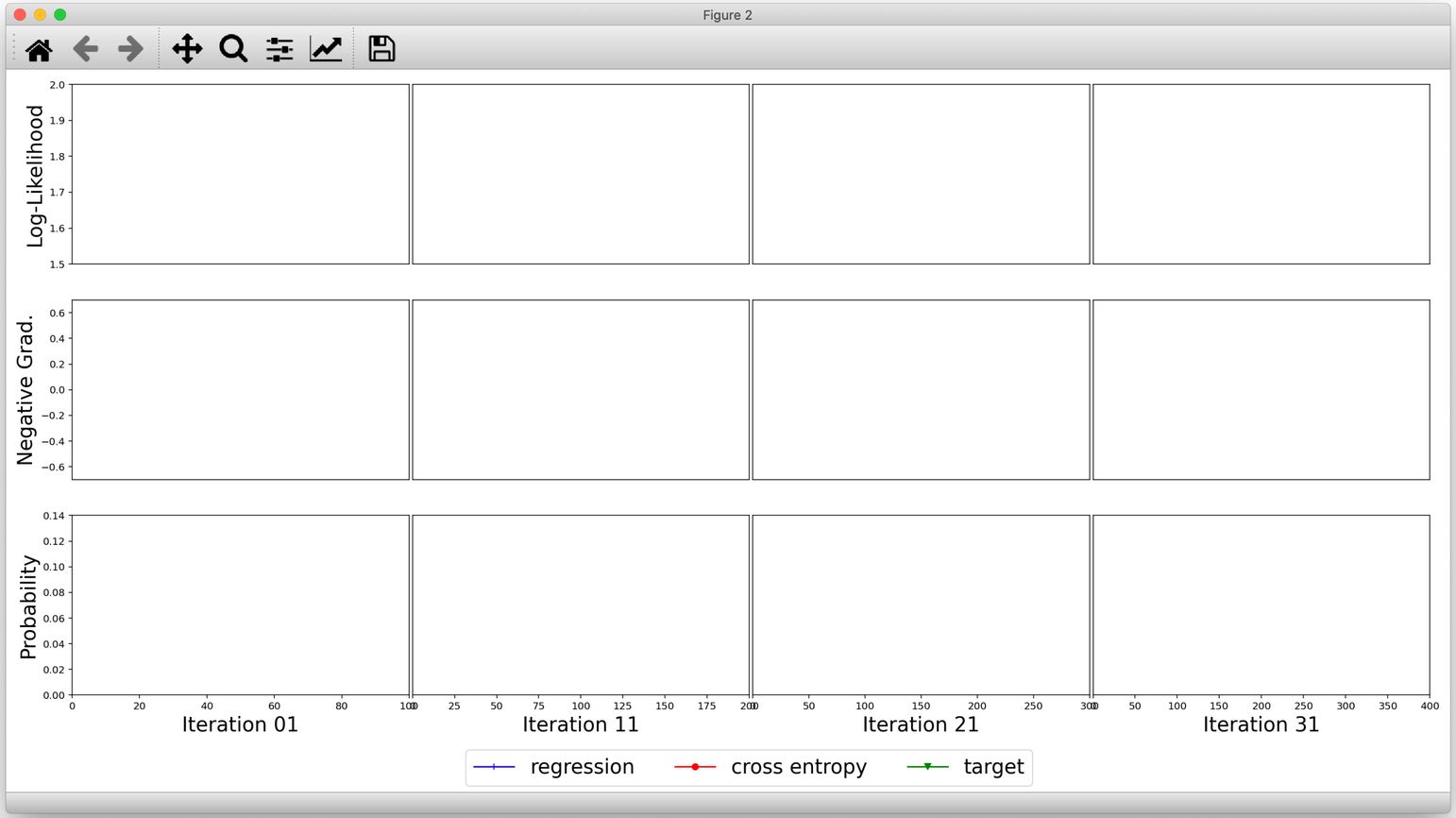


5. Axis Sharing(Practice)

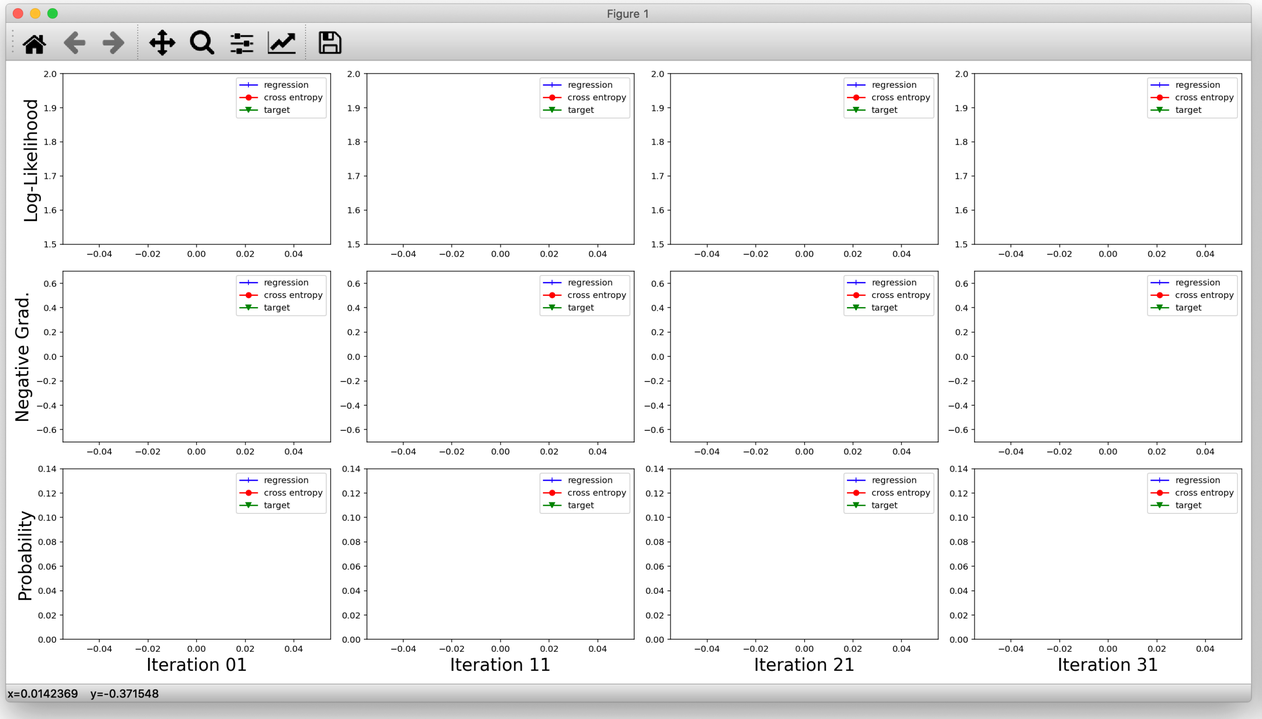


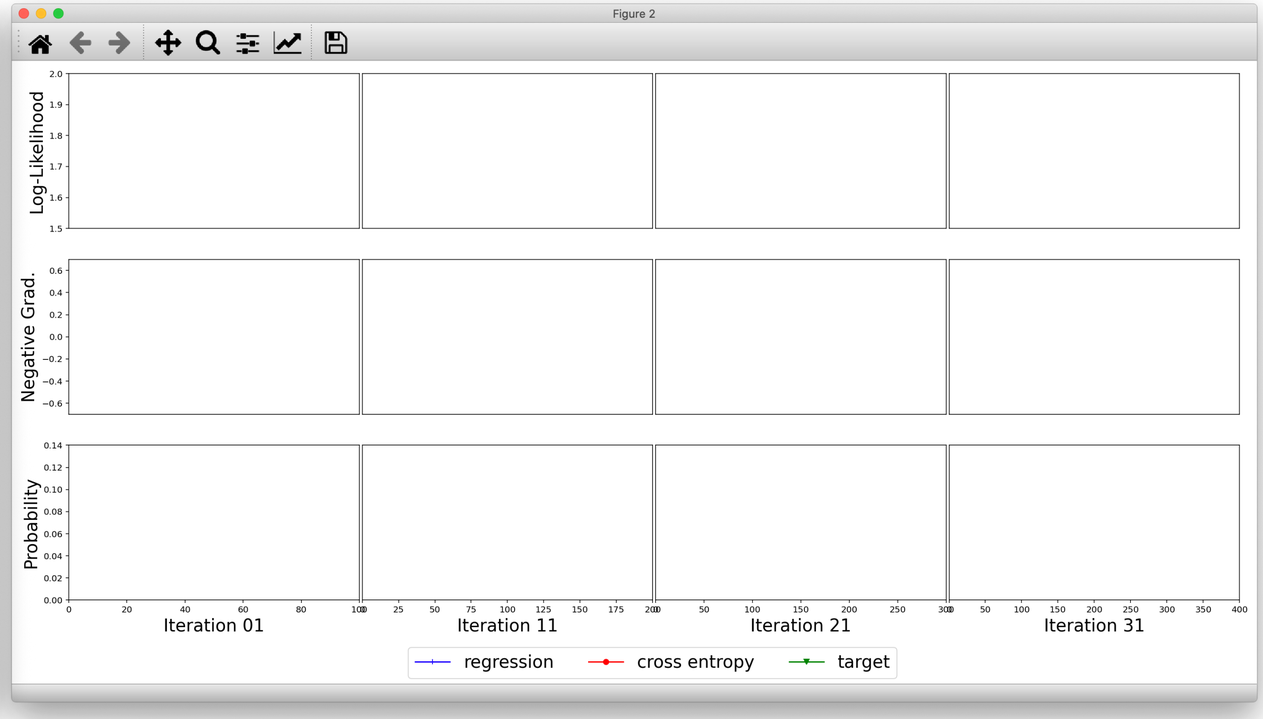
5. Axis Sharing(Practice)

```
for ax_idx, ax in enumerate(axes.flat):
    if ax_idx % n_col == 0:
        ax.set_ylabel(ylabel_list[ax_idx // n_col],
                      fontsize=20)
    if ax_idx >= 2*n_col:
        ax.set_xlabel(xlabel_list[ax_idx - 2*n_col],
                      fontsize=20)

    if ax_idx % n_col != 0:
        ax.get_yaxis().set_visible(False)
    if ax_idx <= n_col*2 - 1:
        ax.get_xaxis().set_visible(False)

axes[2, 1].plot([], color='b', marker='|',
                label='regression')
axes[2, 1].plot([], color='r', marker='o',
                label='cross entropy')
axes[2, 1].plot([], color='g', marker='v',
                label='target')

axes[2, 1].legend(loc='upper center',
                  bbox_to_anchor=(1, -0.25),
                  fontsize=20,
                  ncol=3)

fig.tight_layout()
fig.subplots_adjust(hspace=0.2, wspace=0.01)
```

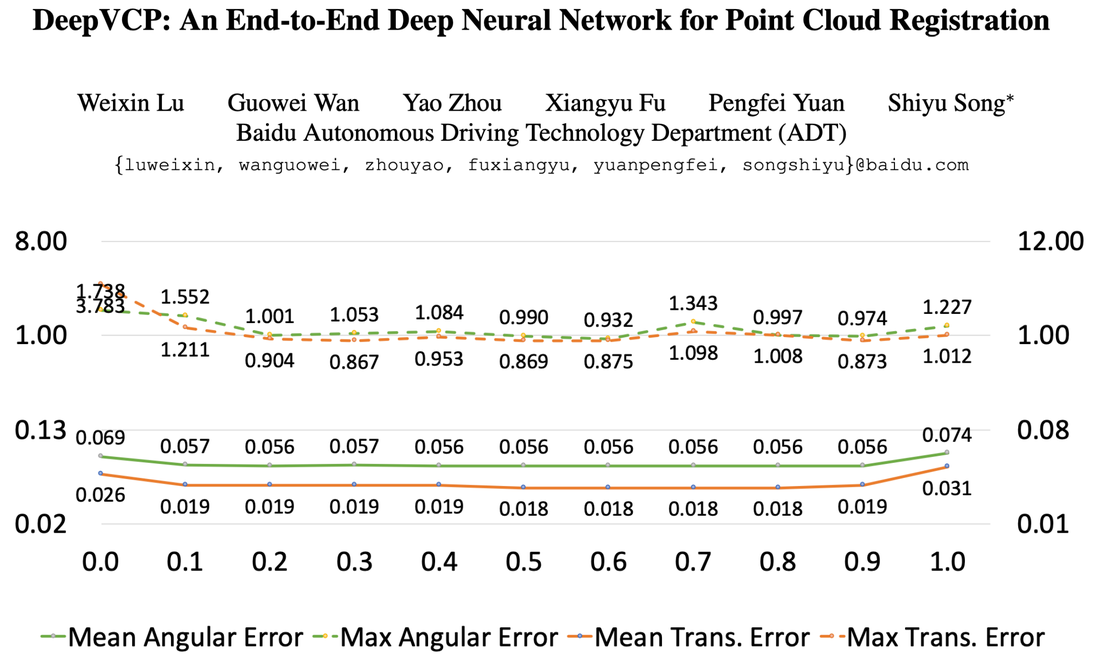


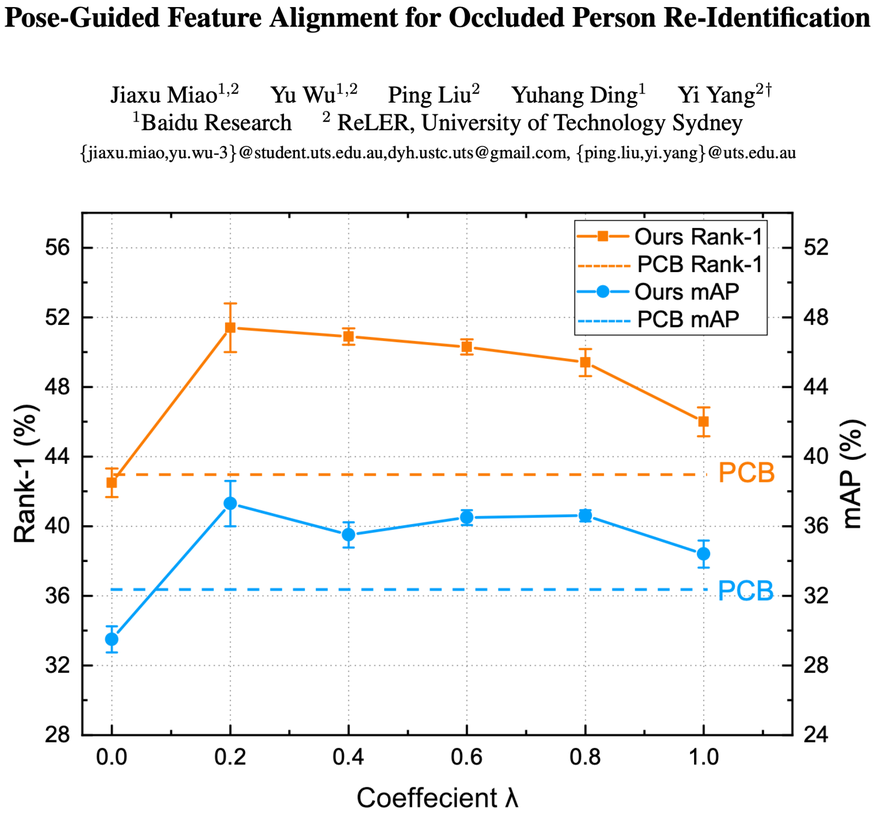
5. Axis Sharing(Practice)





6. ax.twinx(Different Y Values)

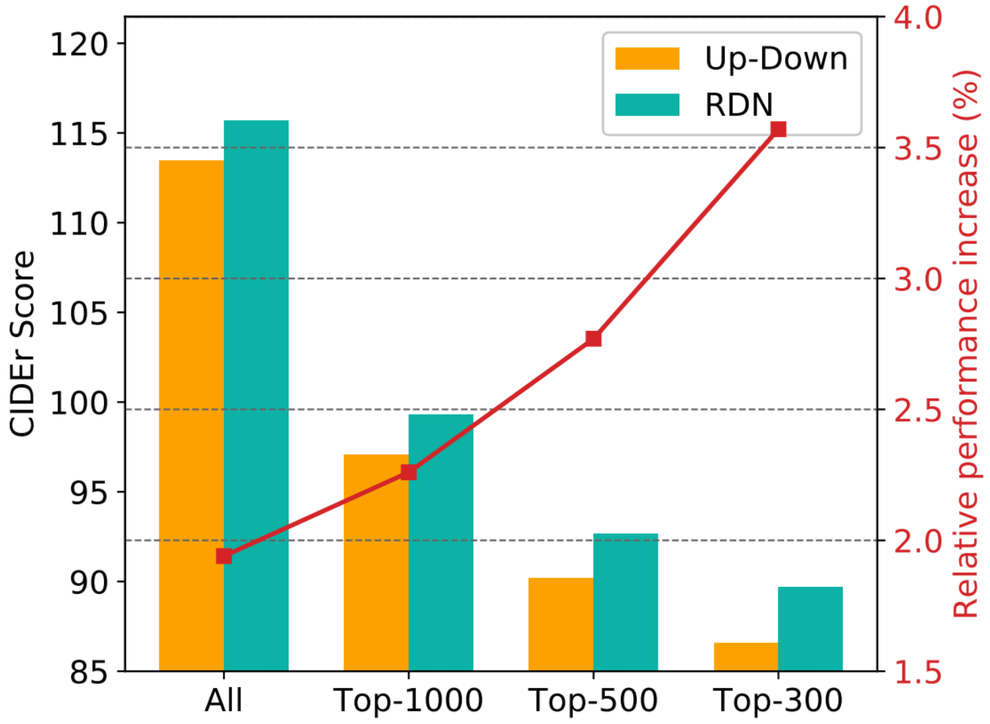


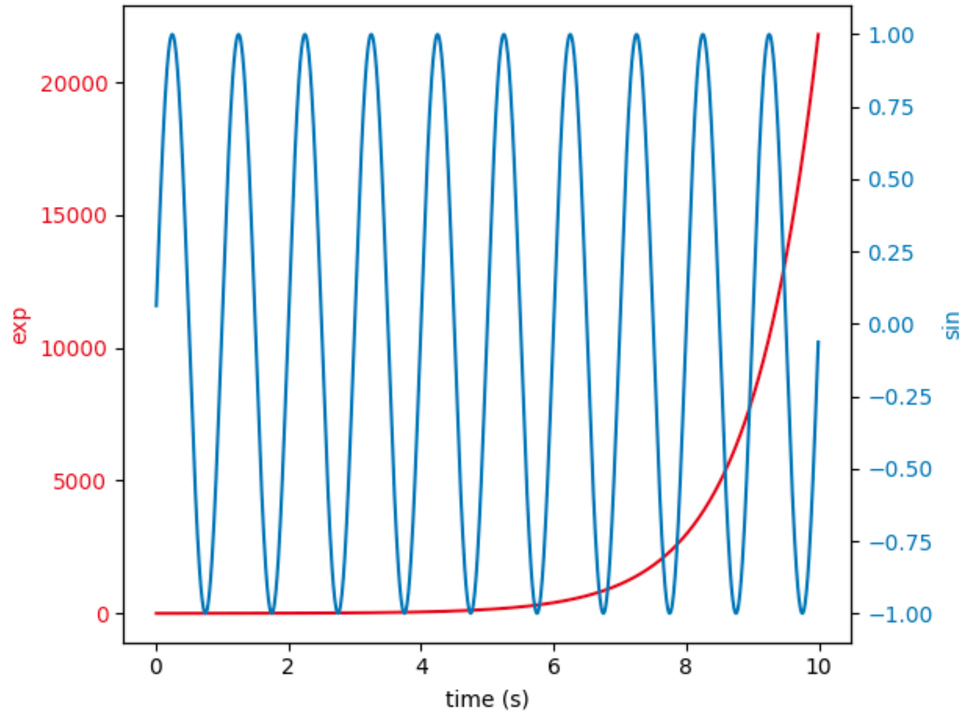


6. ax.twinx

Reflective Decoding Network for Image Captioning

Lei Ke[1], Wenjie Pei[2], Ruiyu Li[2], Xiaoyong Shen[2], Yu-Wing Tai[2]
[1]The Hong Kong University of Science and Technology, [2]Tencent
keleiwhu@gmail.com,wenjiecoder@outlook.com,{royryli,dylanshen,yuwingtai}@tencent.com

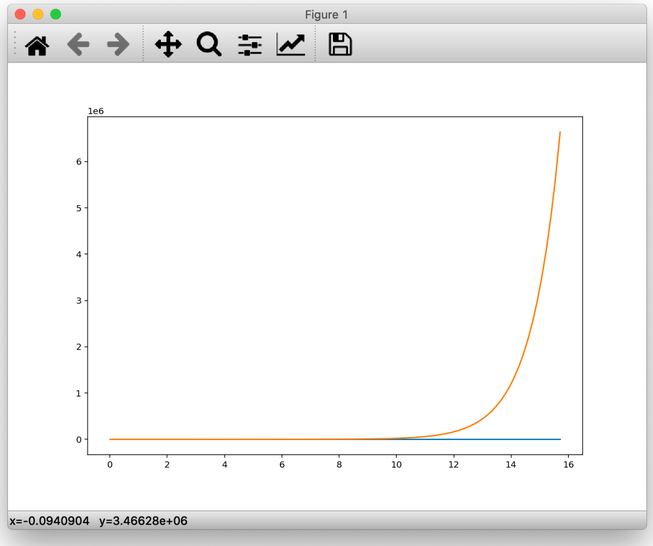




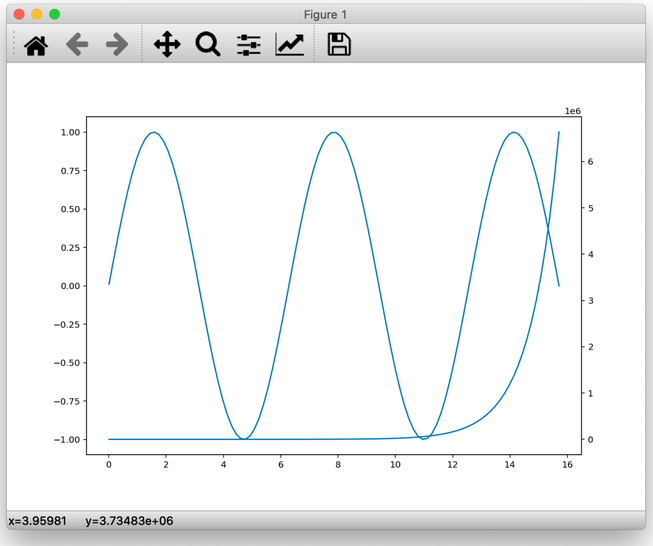
https://matplotlib.org/gallery/subplots_axes_and_figures/two_scales.html#sphx-glr-gallery-subplots-axes-and-figures-two-scales-py

6. ax.twinx

```
import matplotlib.pyplot as plt
import numpy as np

PI = np.pi
t = np.linspace(0.01, 5*PI, 100)
sin = np.sin(t)
exp = np.exp(t)

fig = plt.figure(figsize=(10, 7))
ax1 = fig.add_subplot()

ax1.plot(t, sin)
ax1.plot(t, exp)
```

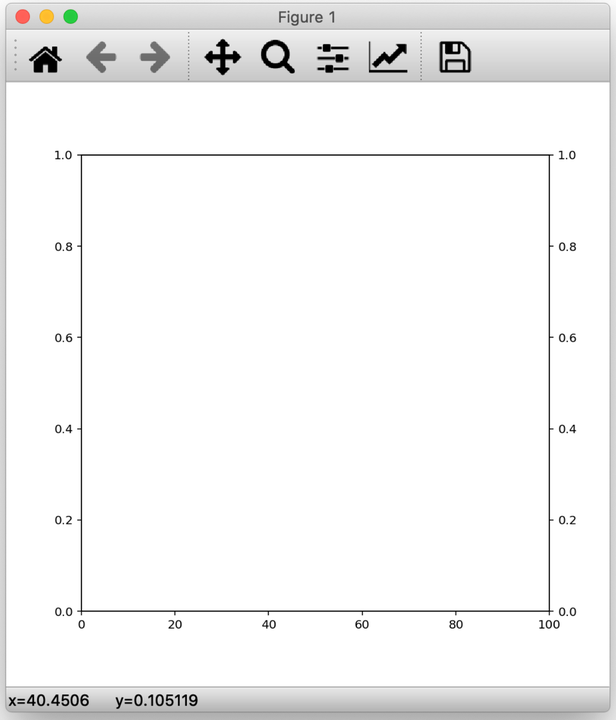



```
fig = plt.figure(figsize=(10, 7))
ax1 = fig.add_subplot()
ax1.plot(t, sin)

ax2 = ax1.twinx()
ax2.plot(t, exp)
```

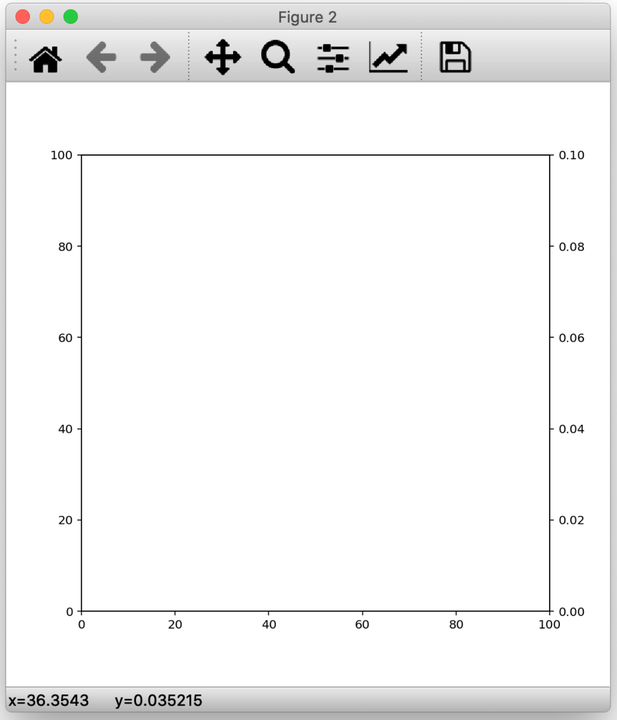


6. ax.twinx

```
fig = plt.figure(figsize=(10, 10))

ax1 = fig.add_subplot()
ax2 = ax1.twinx()

ax1.set_xlim([0, 100])
```

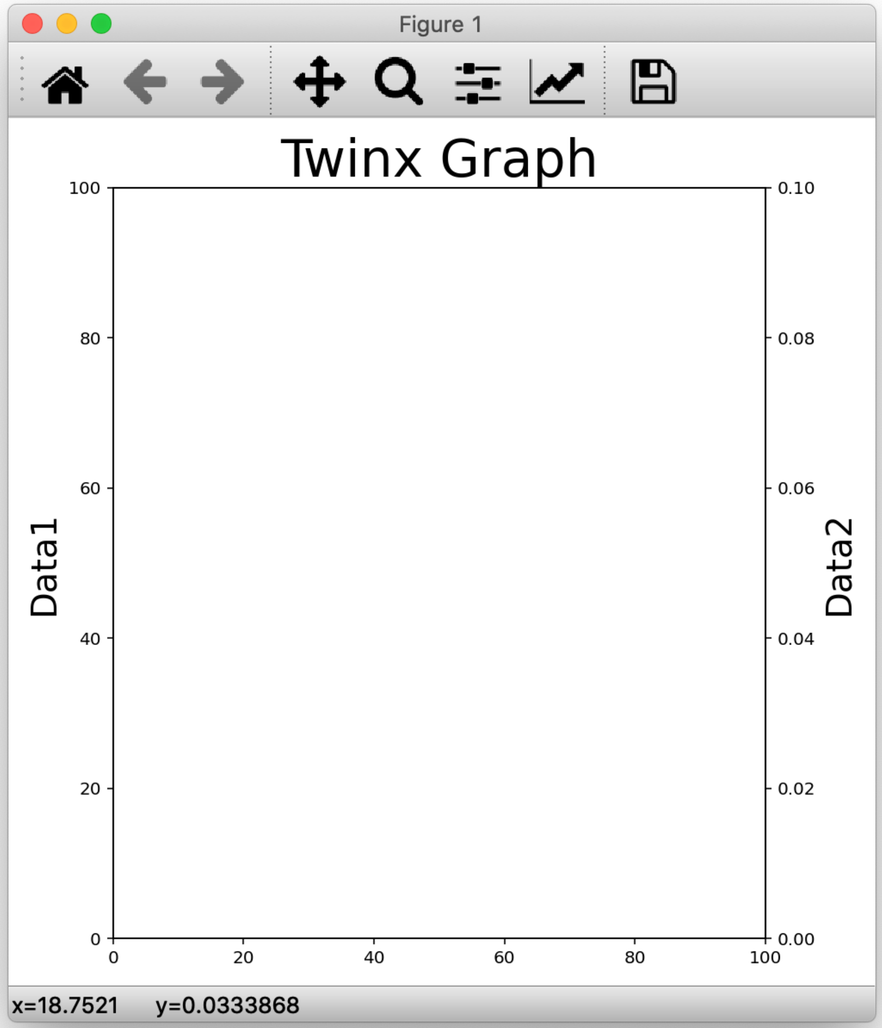



```
ax1.set_xlim([0, 100])

ax1.set_ylim([0, 100])
ax2.set_ylim([0, 0.1])
```



6. ax.twinx

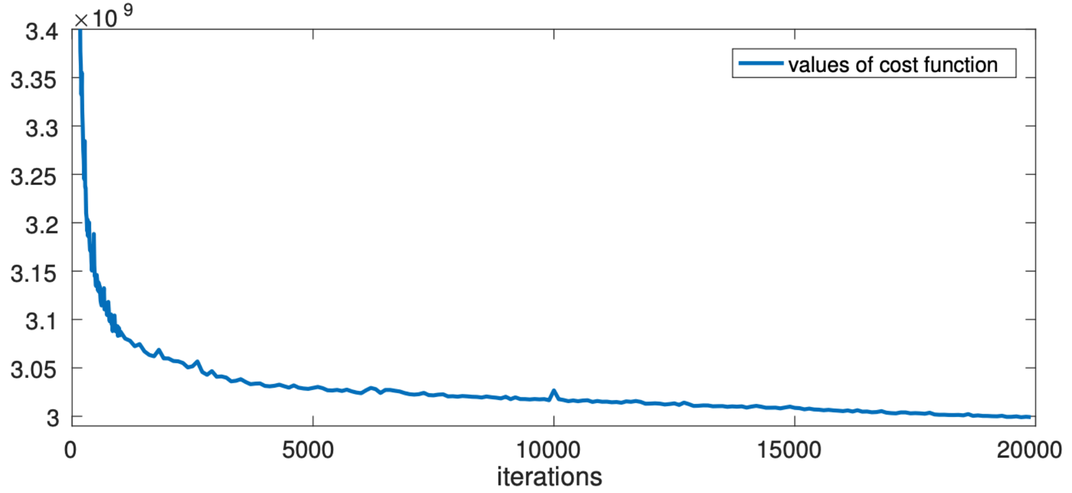
```
fig = plt.figure(figsize=(7, 7))

ax1 = fig.add_subplot()
ax2 = ax1.twinx()

ax1.set_xlim([0, 100])

ax1.set_ylim([0, 100])
ax2.set_ylim([0, 0.1])

ax1.set_title("Twinx Graph", fontsize=30)
ax1.set_ylabel("Data1", fontsize=20)
ax2.set_ylabel("Data2", fontsize=20)

fig.tight_layout()
```

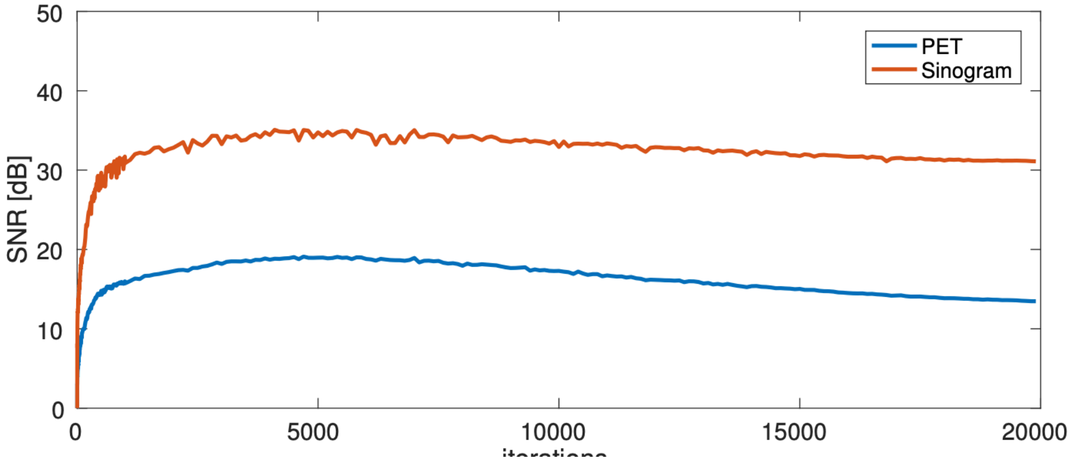


7. ax.set_yscale(Axis Scale)

Dynamic PET Image Reconstruction
Using Nonnegative Matrix Factorization Incorporated With Deep Image Prior

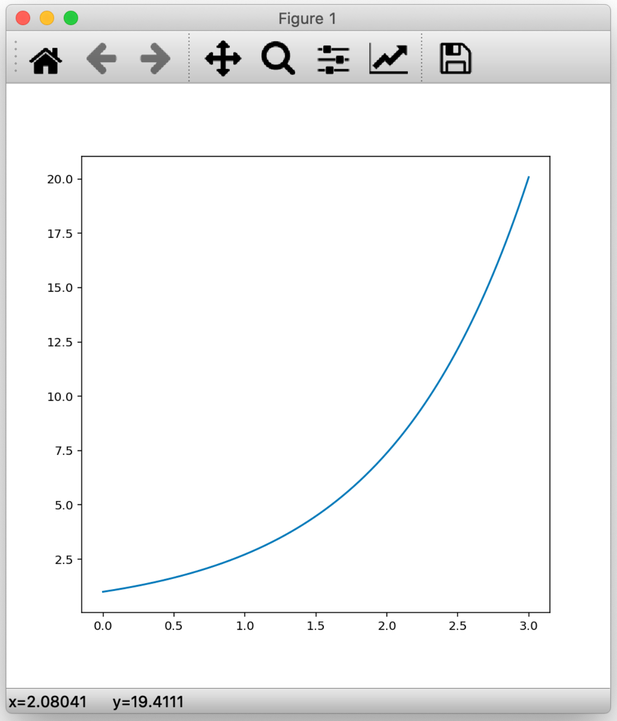
Tatsuya Yokota*,⋆, Kazuya Kawai*, Muneyuki Sakata†, Yuichi Kimura‡, and Hidekata Hontani*
* Nagoya Institute of Technology, Nagoya, Japan, {t.yokota, hontani}@nitech.ac.jp
⋆ RIKEN Center for Advanced Intelligence Project, Tokyo, Japan
† Tokyo Metropolitan Institute of Gerontology, Tokyo, Japan
‡ Kindai University, Wakayama, Japan



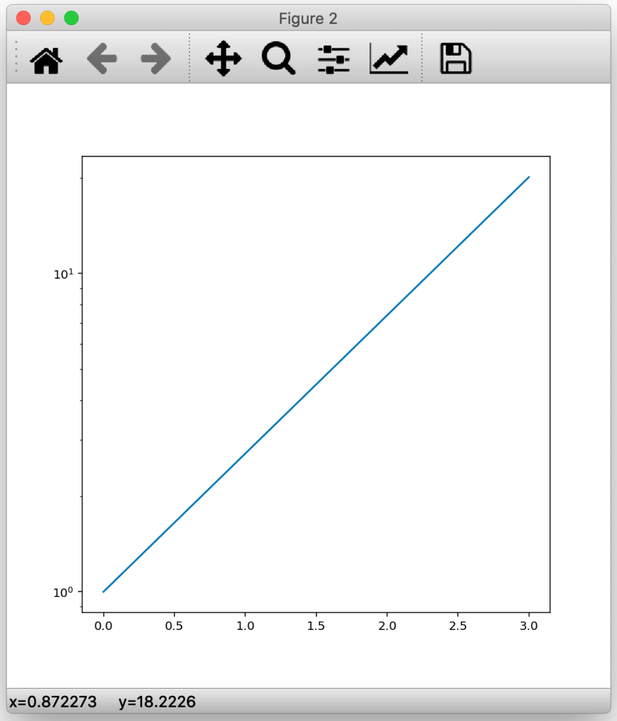


7. ax.set_yscale(Log Scale)

```
import matplotlib.pyplot as plt
import numpy as np

t = np.linspace(0, 3, 300)
exp = np.exp(t)

fig, ax = plt.subplots(figsize=(7, 7))
ax.plot(t, exp)
```

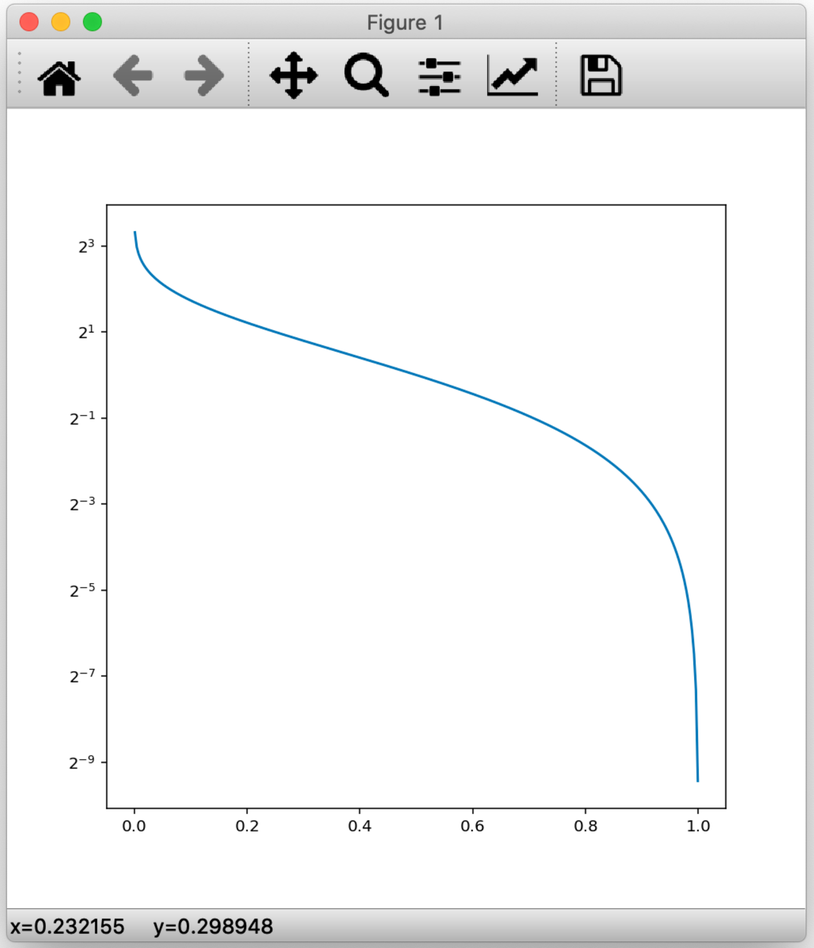



```
fig, ax = plt.subplots(figsize=(7, 7))
ax.set_yscale('log')
ax.plot(t, exp)
```



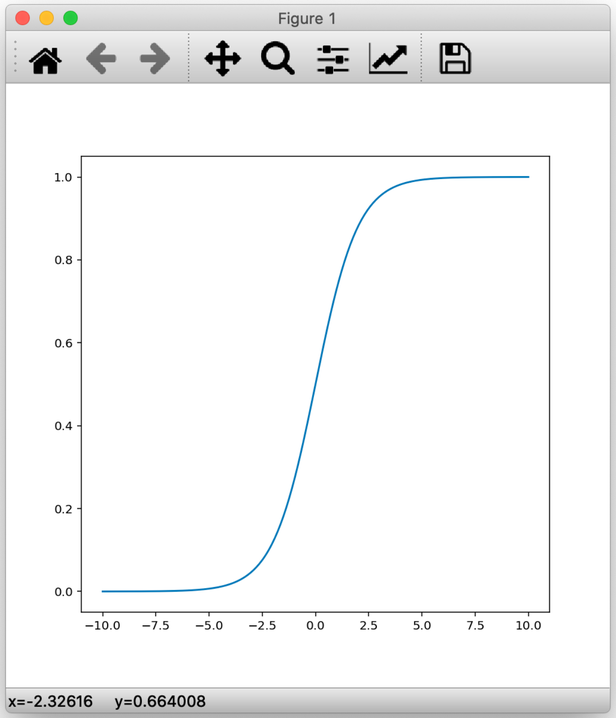
7. ax.set_yscale(Log Scale with Base2)

```
import matplotlib.pyplot as plt
import numpy as np

p = np.linspace(0.001, 0.999, 300)
information = -np.log2(p)

fig, ax = plt.subplots(figsize=(7, 7))
ax.set_yscale('log',
              basey=2)

ax.plot(p, information)
```

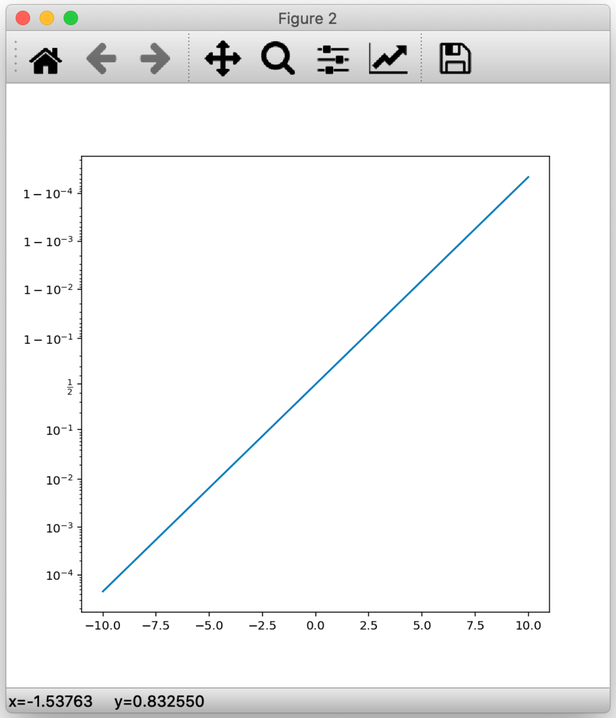


7. ax.set_yscale(Logit Scale)

```
logit = np.linspace(-10, 10, 300)
sigmoid = 1 / (1 + np.exp(-logit))

fig, ax = plt.subplots(figsize=(7, 7))
ax.plot(logit, sigmoid)
```




```
fig, ax = plt.subplots(figsize=(7, 7))
ax.set_yscale('logit')
ax.plot(logit, sigmoid)
```



Python for Data Visualization

-Chapter.1 Matplotlib Anatomy -

1-02. Axes Customizing

1. fig.tight_layout(Axes Layout Adjustment)
2. fig.subplots_adjust(More Customized Layout)
3. fig.subplots_adjust(Practice)
4. Axis Sharing
5. Axis Sharing(Practice)
6. ax.twinx(Different Y Values)
7. ax.set_yscale(Axis Scale)