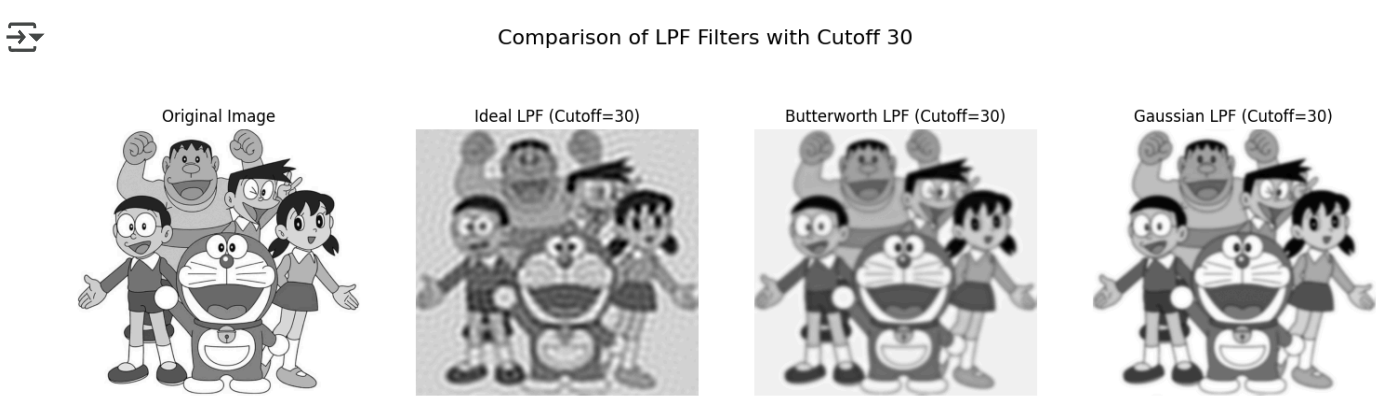
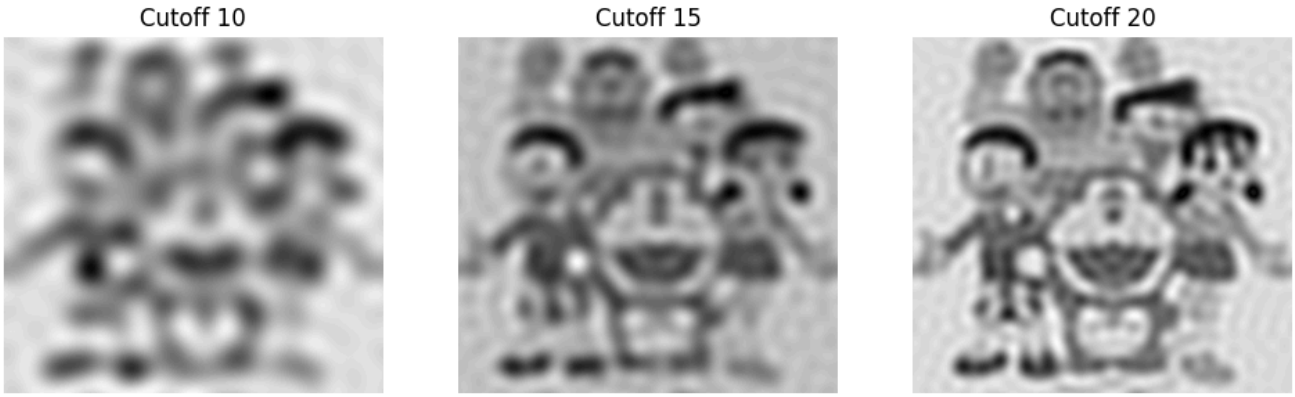
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def ideal_low_pass_filter(shape, cutoff):
    rows, cols = shape
    crow, ccol = rows // 2, cols // 2
    mask = np.zeros((rows, cols), np.uint8)
    for i in range(rows):
        for j in range(cols):
            if np.sqrt((i - crow) ** 2 + (j - ccol) ** 2) <= cutoff:
                mask[i, j] = 1
    return mask

def butterworth_low_pass_filter(shape, cutoff, order=2):
    rows, cols = shape
    crow, ccol = rows // 2, cols // 2
    mask = np.zeros((rows, cols), np.float32)
    for i in range(rows):
        for j in range(cols):
            d = np.sqrt((i - crow) ** 2 + (j - ccol) ** 2)
            mask[i, j] = 1 / (1 + (d / cutoff) ** (2 * order))
    return mask

def gaussian_low_pass_filter(shape, cutoff):
    rows, cols = shape
    crow, ccol = rows // 2, cols // 2
    mask = np.zeros((rows, cols), np.float32)
    for i in range(rows):
        for j in range(cols):
            d = np.sqrt((i - crow) ** 2 + (j - ccol) ** 2)
            mask[i, j] = np.exp(-(d ** 2) / (2 * (cutoff ** 2)))
    return mask

def apply_filter(image, filter_mask):
    dft = np.fft.fft2(image)
    dft_shift = np.fft.fftshift(dft)
    filtered_dft = dft_shift * filter_mask
    dft_inverse_shift = np.fft.ifftshift(filtered_dft)
    filtered_image = np.fft.ifft2(dft_inverse_shift)
    return np.abs(filtered_image)



image = cv2.imread('/content/Doremon.jpg', cv2.IMREAD_GRAYSCALE)

cutoff_values = [5, 15, 30, 90, 120]
order = 2

cutoff = 30
ideal_filtered = apply_filter(image, ideal_low_pass_filter(image.shape, cutoff))
butterworth_filtered = apply_filter(image, butterworth_low_pass_filter(image.shape, cutoff, order))
gaussian_filtered = apply_filter(image, gaussian_low_pass_filter(image.shape, cutoff))

ringing_cutoffs = [10, 15, 20]
ringing_results = [apply_filter(image, ideal_low_pass_filter(image.shape, c)) for c in ringing_cutof

butterworth_results = [apply_filter(image, butterworth_low_pass_filter(image.shape, c, order)) for c
```

```
gaussian_results = [apply_filter(image, gaussian_low_pass_filter(image.shape, c)) for c in cutoff_va


fig, axes = plt.subplots(1, 4, figsize=(18, 5))
plt.subplots_adjust(top=0.85)
axes[0].imshow(image, cmap='gray')
axes[0].set_title("Original Image")
axes[0].axis('off')

axes[1].imshow(ideal_filtered, cmap='gray')
axes[1].set_title("Ideal LPF (Cutoff=30)")
axes[1].axis('off')

axes[2].imshow(butterworth_filtered, cmap='gray')
axes[2].set_title("Butterworth LPF (Cutoff=30)")
axes[2].axis('off')

axes[3].imshow(gaussian_filtered, cmap='gray')
axes[3].set_title("Gaussian LPF (Cutoff=30)")
axes[3].axis('off')

plt.suptitle("Comparison of LPF Filters with Cutoff 30", fontsize=16)
plt.show()
```




```
fig, axes = plt.subplots(1, 3, figsize=(12, 4))
plt.subplots_adjust(top=0.85)
for i, c in enumerate(ringing_cutoffs):
    axes[i].imshow(ringing_results[i], cmap='gray')
    axes[i].set_title(f"Cutoff {c}", fontsize=12)
    axes[i].axis('off')

plt.suptitle("Ringing Effect in Ideal LPF", fontsize=16)
plt.show()
```




```
fig, axes = plt.subplots(1, 5, figsize=(18, 5))
plt.subplots_adjust(top=0.85)
for i, c in enumerate(cutoff_values):
    axes[i].imshow(butterworth_results[i], cmap='gray')
    axes[i].set_title(f"Cutoff {c}", fontsize=12)
    axes[i].axis('off')

plt.suptitle("Butterworth LPF for Different Cutoff Frequencies", fontsize=16)
plt.show()
```


```
fig, axes = plt.subplots(1, 5, figsize=(18, 5))
plt.subplots_adjust(top=0.85)
for i, c in enumerate(cutoff_values):
    axes[i].imshow(gaussian_results[i], cmap='gray')
    axes[i].set_title(f"Cutoff {c}", fontsize=12)
    axes[i].axis('off')

plt.suptitle("Gaussian LPF for Different Cutoff Frequencies", fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.subplots_adjust(top=0.92)
plt.show()
```


Gaussian LPF for Different Cutoff Frequencies
Cutoff 5
Cutoff 15
Cutoff 30
Cutoff 90
Cutoff 120