In [1]:

```python
import os
import numpy as np
from PIL import Image, ImageEnhance, ImageFilter
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import logging

# Set up logging to handle any corrupted images
logging.basicConfig(level=logging.INFO, filename='corrupted_images.log', f
logger = logging.getLogger()

```

In [2]:

```python
def load_image_safe(filepath, apply_processing=None):
    try:
        img = Image.open(filepath)
        img = img.resize((224, 224))

        # Apply specified processing
        if apply_processing == "contrast":
            img = ImageEnhance.Contrast(img).enhance(1.5)  # Increase cont
        elif apply_processing == "blur":
            img = img.filter(ImageFilter.GaussianBlur(1))  # Apply Gaussia
        elif apply_processing == "edge":
            img = img.filter(ImageFilter.EDGE_ENHANCE)     # Enhance edges

        img = img.convert("RGB")  # Ensure consistency in format
        return np.array(img) / 255.0  # Normalize
    except Exception as e:
        logger.info(f"Corrupted image skipped: {filepath} | Error: {e}")
        return None

```

In [3]:

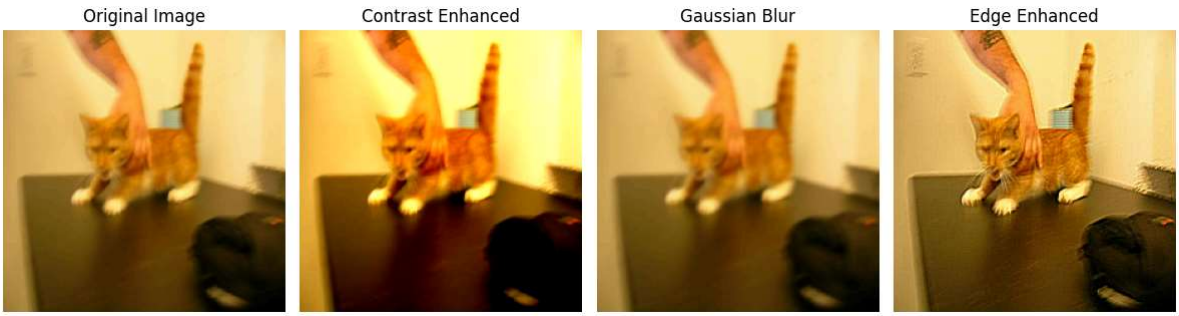
```
def safe_image_generator(directory, image_size=(224, 224), batch_size=32,
    datagen = ImageDataGenerator(rescale=1./255)
    generator = datagen.flow_from_directory(
        directory,
        target_size=image_size,
        batch_size=batch_size,
        class_mode=class_mode,
        shuffle=True
    )

    while True:
        batch_data, batch_labels = generator.next()
        valid_images = []
        valid_labels = []
        for i in range(len(batch_data)):
            img_path = generator.filepaths[generator.index_array[i]]
            img = load_image_safe(img_path, apply_processing)
            if img is not None:
                valid_images.append(img)
                valid_labels.append(batch_labels[i])
        if valid_images:
            yield np.array(valid_images), np.array(valid_labels)

```

In [4]:

```
def create_model():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['
    return model

```

In [5]:

```
# Load sample image from the dataset
sample_directory = 'C:/Users/asus/Downloads/archive/kagglecatsanddogs_3367
sample_image_path = os.path.join(sample_directory, os.listdir(sample_direc

# Apply different processing techniques
original_img = load_image_safe(sample_image_path)
contrast_img = load_image_safe(sample_image_path, apply_processing="contra
blurred_img = load_image_safe(sample_image_path, apply_processing="blur")
edge_img = load_image_safe(sample_image_path, apply_processing="edge")

# Plot all images side by side for comparison
plt.figure(figsize=(12, 6))

# Original Image
plt.subplot(1, 4, 1)
plt.imshow(original_img)
plt.title("Original Image")
plt.axis('off')

# Contrast-enhanced Image
plt.subplot(1, 4, 2)
plt.imshow(contrast_img)
plt.title("Contrast Enhanced")
plt.axis('off')

# Gaussian-blurred Image
plt.subplot(1, 4, 3)
plt.imshow(blurred_img)
plt.title("Gaussian Blur")
plt.axis('off')

# Edge-enhanced Image
plt.subplot(1, 4, 4)
plt.imshow(edge_img)
plt.title("Edge Enhanced")
plt.axis('off')

plt.tight_layout()
plt.show()

```



In [6]:

```
directory_path = 'C:/Users/asus/Downloads/archive/kagglecatsanddogs_3367a/
image_size = (224, 224)
batch_size = 64

# Initialize model
model_no_processing = create_model()

# Train and evaluate
train_data_gen = safe_image_generator(directory_path, image_size=image_siz
history_no_processing = model_no_processing.fit(train_data_gen, steps_per_

# Output final accuracy
accuracy_no_processing = history_no_processing.history['accuracy'][-1]
print(f"Final accuracy without preprocessing: {accuracy_no_processing:.2f}
```

```
Found 24959 images belonging to 2 classes.
Epoch 1/5
100/100 [==============================] - 210s 2s/step - loss: 0.9062 - accu
racy: 0.5023
Epoch 2/5
 81/100 [=======================>......] - ETA: 39s - loss: 0.6933 - accurac
y: 0.4909

c:\Users\asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\PIL\T
iffImagePlugin.py:864: UserWarning: Truncated File Read
  warnings.warn(str(msg))

100/100 [==============================] - 209s 2s/step - loss: 0.6933 - accu
racy: 0.4917
Epoch 3/5
100/100 [==============================] - 205s 2s/step - loss: 0.6933 - accu
racy: 0.4997
Epoch 4/5
100/100 [==============================] - 198s 2s/step - loss: 0.6932 - accu
racy: 0.4976
Epoch 5/5
100/100 [==============================] - 185s 2s/step - loss: 0.6932 - accu
racy: 0.5028
Final accuracy without preprocessing: 0.50
```

In [7]:

```
# Initialize model
model_contrast = create_model()

# Train and evaluate
train_data_gen_contrast = safe_image_generator(directory_path, image_size=
history_contrast = model_contrast.fit(train_data_gen_contrast, steps_per_e

# Output final accuracy
accuracy_contrast = history_contrast.history['accuracy'][-1]
print(f"Final accuracy with contrast enhancement: {accuracy_contrast:.2f}"

```

```
Found 24959 images belonging to 2 classes.
Epoch 1/5
100/100 [==============================] - 182s 2s/step - loss: 0.8995 - accu
racy: 0.4989
Epoch 2/5
100/100 [==============================] - 186s 2s/step - loss: 0.6933 - accu
racy: 0.5077
Epoch 3/5
100/100 [==============================] - 187s 2s/step - loss: 0.6934 - accu
racy: 0.4986
Epoch 4/5
100/100 [==============================] - 191s 2s/step - loss: 0.6931 - accu
racy: 0.5018
Epoch 5/5
100/100 [==============================] - 172s 2s/step - loss: 0.6937 - accu
racy: 0.5008
Final accuracy with contrast enhancement: 0.50
```

In [8]:

```
# Initialize model
model_blur = create_model()

# Train and evaluate
train_data_gen_blur = safe_image_generator(directory_path, image_size=imag
history_blur = model_blur.fit(train_data_gen_blur, steps_per_epoch=100, ep

# Output final accuracy
accuracy_blur = history_blur.history['accuracy'][-1]
print(f"Final accuracy with Gaussian blur: {accuracy_blur:.2f}")

```

```
Found 24959 images belonging to 2 classes.
Epoch 1/5
100/100 [==============================] - 178s 2s/step - loss: 0.8845 - accu
racy: 0.4998
Epoch 2/5
100/100 [==============================] - 174s 2s/step - loss: 0.6932 - accu
racy: 0.4961
Epoch 3/5
100/100 [==============================] - 188s 2s/step - loss: 0.6931 - accu
racy: 0.5088
Epoch 4/5
100/100 [==============================] - 194s 2s/step - loss: 0.6932 - accu
racy: 0.4999
Epoch 5/5
100/100 [==============================] - 176s 2s/step - loss: 0.6932 - accu
racy: 0.4972
Final accuracy with Gaussian blur: 0.50
```

In [9]:

```
# Initialize model
model_edge = create_model()

# Train and evaluate
train_data_gen_edge = safe_image_generator(directory_path, image_size=imag
history_edge = model_edge.fit(train_data_gen_edge, steps_per_epoch=100, ep

# Output final accuracy
accuracy_edge = history_edge.history['accuracy'][-1]
print(f"Final accuracy with edge enhancement: {accuracy_edge:.2f}")

```

```
Found 24959 images belonging to 2 classes.
Epoch 1/5
100/100 [==============================] - 172s 2s/step - loss: 1.2330 - accu
racy: 0.4992
Epoch 2/5
100/100 [==============================] - 189s 2s/step - loss: 0.6932 - accu
racy: 0.4953
Epoch 3/5
100/100 [==============================] - 176s 2s/step - loss: 0.6932 - accu
racy: 0.4903
Epoch 4/5
100/100 [==============================] - 179s 2s/step - loss: 0.6932 - accu
racy: 0.4971
Epoch 5/5
100/100 [==============================] - 169s 2s/step - loss: 0.6932 - accu
racy: 0.5036
Final accuracy with edge enhancement: 0.50
```

In [10]:

```
print(f"Accuracy without preprocessing: {accuracy_no_processing:.2f}")
print(f"Accuracy with contrast enhancement: {accuracy_contrast:.2f}")
print(f"Accuracy with Gaussian blur: {accuracy_blur:.2f}")
print(f"Accuracy with edge enhancement: {accuracy_edge:.2f}")

```

```
Accuracy without preprocessing: 0.50
Accuracy with contrast enhancement: 0.50
Accuracy with Gaussian blur: 0.50
Accuracy with edge enhancement: 0.50
```

In [ ]:

```

```