```python
import torch
from torchvision.transforms import ToTensor, ToPILImage
from PIL import Image
import requests
from io import BytesIO

# Define the EDSR model architecture
class EDSR(torch.nn.Module):
    def __init__(self, num_channels=3, num_blocks=16, scaling_factor=4):
        super(EDSR, self).__init__()
        self.scaling_factor = scaling_factor
        self.input_conv = torch.nn.Conv2d(num_channels, 64, kernel_size=3, padding=1)
        self.residual_layers = torch.nn.Sequential(
            *[self._residual_block() for _ in range(num_blocks)]
        )
        self.upsample = torch.nn.Sequential(
            torch.nn.Conv2d(64, 64 * scaling_factor ** 2, kernel_size=3, padding=1),
            torch.nn.PixelShuffle(scaling_factor),
            torch.nn.Conv2d(64, num_channels, kernel_size=3, padding=1)
        )

    def _residual_block(self):
        block = torch.nn.Sequential(
            torch.nn.Conv2d(64, 64, kernel_size=3, padding=1),
            torch.nn.ReLU(inplace=True),
            torch.nn.Conv2d(64, 64, kernel_size=3, padding=1)
        )
        return block
    def forward(self, x):
        x = self.input_conv(x)
        res = self.residual_layers(x)
        x = x + res  # Skip connection
        x = self.upsample(x)
        return x

# Function to enhance the image resolution
def super_resolve_image(image_path, model, device):
    # Load the image
    image = Image.open(image_path).convert('RGB')

    # Transform to tensor
    image_tensor = ToTensor()(image).unsqueeze(0).to(device)

    # Super-resolve the image using the model
    with torch.no_grad():
        output_tensor = model(image_tensor).squeeze(0)

    # Convert back to image
    output_image = ToPILImage()(output_tensor.cpu())

    return output_image

if __name__ == "__main__":
    # Set device to GPU if available
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    # Load the pre-trained EDSR model
    scaling_factor = 4  # You can adjust the scaling factor (e.g., 2, 3, 4)
    edsr_model = EDSR(scaling_factor=scaling_factor).to(device)

    # Assume we have a pre-trained model; if not, we can use a randomly initialized model
    # edsr_model.load_state_dict(torch.load('path_to_pretrained_model.pth'))

    # Test image path
    image_url = "https://example.com/sample-image.jpg"  # Replace with your image URL or local path
    response = requests.get(image_url)
    img = Image.open(BytesIO(response.content))

    # Super-resolve the image
    sr_image = super_resolve_image(img, edsr_model, device)

    # Save the result
    sr_image.save("super_resolved_image.png")
    sr_image.show()
```