```
import gradio as gr
from PIL import Image
import torch
import torchvision.transforms as T
import numpy as np
from torchvision import models


# Define EfficientNetSegmentation class
class EfficientNetSegmentation(torch.nn.Module):
    def __init__(self, num_classes):
        super(EfficientNetSegmentation, self).__init__()
        self.backbone = models.efficientnet_b0(weights=models.EfficientNet_B0_Weights.IMAGENET1K_V1).features

        # Freeze the parameters in the backbone
        for param in self.backbone.parameters():
            param.requires_grad = False

        self.upsample = torch.nn.Sequential(
            torch.nn.Conv2d(1280, num_classes, kernel_size=1),
            torch.nn.Upsample(size=(512, 512), mode='bilinear', align_corners=False)
        )

    def forward(self, x):
        x = self.backbone(x)
        x = self.upsample(x)
        return x


# Load model
model_path = "model/best_model.pth"
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
num_classes = 4
model = EfficientNetSegmentation(num_classes=num_classes).to(device)

# Load the model state_dict
try:
    state_dict = torch.load(model_path, map_location=device)
    model.load_state_dict(state_dict, strict=False)
    model.eval()
    print("Model loaded successfully.")
except Exception as e:
    print(f"Error loading model: {e}")
    raise

# Define label colors
label_colors = {
    "Background": (0, 0, 0),
    "Grilled Chicken": (255, 0, 0),
    "Paneer": (0, 255, 0),
    "Eggplant": (0, 0, 255),
}

# Image processing function
def process_image(image):
    transform = T.Compose([
        T.Resize((512, 512)),
        T.ToTensor(),
        T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])
    input_tensor = transform(image).unsqueeze(0).to(device)
    with torch.no_grad():
        output = model(input_tensor).argmax(dim=1).cpu().numpy()[0]

    seg_image = np.zeros((output.shape[0], output.shape[1], 3), dtype=np.uint8)
    for class_name, color in label_colors.items():
        class_idx = list(label_colors.keys()).index(class_name)
        seg_image[output == class_idx] = color

    seg_image_pil = Image.fromarray(seg_image)
    return seg_image_pil

# CSS for interface
css = """
body {
    background: linear-gradient(135deg, #1e272e, #34495e);
    color: #ecf0f1;
    font-family: 'Roboto', sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
```

```
        flex-direction: column;
        align-items: center;
        justify-content: center;
        min-height: 100vh;
    }
    .title {
        font-size: 2.5rem;
        color: #00cec9;
        text-shadow: 2px 2px 4px #000000;
        font-family: 'Montserrat', sans-serif;
        font-weight: bold;
        margin-bottom: 15px;
        text-align: center;
    }
    .description {
        font-size: 1.2rem;
        color: #74b9ff;
        font-family: 'Arial', sans-serif;
        margin-bottom: 30px;
        text-align: center;
        line-height: 1.6;
    }
    .gr-button {
        background-color: #0984e3;
        color: white;
        border: none;
        border-radius: 25px;
        padding: 10px 20px;
        font-size: 16px;
        transition: all 0.3s ease-in-out;
        cursor: pointer;
    }
    .gr-button:hover {
        background-color: #74b9ff;
        box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.2);
        transform: scale(1.1);
    }
    .gr-input, .gr-output {
        border: 2px solid #00cec9;
        border-radius: 20px;
        padding: 10px;
        background-color: #34495e;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1), 0 1px 3px rgba(0, 0, 0, 0.06);
    }
    .gr-input:hover, .gr-output:hover {
        border-color: #0984e3;
        box-shadow: 0 6px 10px rgba(0, 0, 0, 0.15);
    }
    footer {
        text-align: center;
        margin-top: 20px;
        font-size: small;
        color: #bdc3c7;
        border-top: 1px solid #34495e;
        padding: 10px 0;
    }
    """

    # Example images
    examples = [
        ["/Users/shaaguns/Desktop/Data Science/3rd sem/DL/Project/flask app/flask_app/Examples/190.png"],
        ["/Users/shaaguns/Desktop/Data Science/3rd sem/DL/Project/flask app/flask_app/Examples/112.png"],
        ["/Users/shaaguns/Desktop/Data Science/3rd sem/DL/Project/flask app/flask_app/Examples/1.png"],
    ]

    description = """
    <div class='description'>
        Discover What's on Your Plate: Upload an Image to Segment Grilled Chicken, Paneer and Eggplant!
        <br><br>
        <b>Legend:</b>
        <ul style="list-style-type: none; text-align: left; padding-left: 0;">
            <li style="color: red;">🟥 Red - Grilled Chicken</li>
            <li style="color: green;">🟩 Green - Paneer</li>
            <li style="color: blue;">🟦 Blue - Eggplant</li>
        </ul>
    </div>
    """

    interface = gr.Interface(
        fn=process_image,
        inputs=gr.Image(type="pil", label="Upload Food Image"),
        outputs=gr.Image(type="pil", label="Segmented Output"),
        examples=examples,
```

```
        title="<div class='title'>Food Image Segmentation</div>",
        description=description,
        css=css,
    )


    # Define Gradio interface
    interface = gr.Interface(
        fn=process_image,
        inputs=gr.Image(type="pil", label="Upload Food Image"),
        outputs=gr.Image(type="pil", label="Segmented Output"),
        examples=examples,
        title="<div class='title'>Food Image Segmentation</div>",
        description= description,
        css=css,
    )

    # Run Gradio app
    if __name__ == "__main__":
        interface.launch(share=True)
```