```
!pip install --upgrade transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.47.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transforme
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transfor
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.10/dist-packages (from tra
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transfo
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingfa
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from r
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->tra
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from request
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from request
```

```
!pip install numpy==1.23.5
```

```
Requirement already satisfied: numpy==1.23.5 in /usr/local/lib/python3.10/dist-packages (1.23.5)
```

```
import torch
```

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("pkdarabi/the-drug-name-detection-dataset")

print("Path to dataset files:", path)
```

```
Path to dataset files: /root/.cache/kagglehub/datasets/pkdarabi/the-drug-name-detection-dataset/versions/1
```

```
# Dataset paths
data_dir = "/root/.cache/kagglehub/datasets/pkdarabi/the-drug-name-detection-dataset/versions/1"
```

```
from torchvision import datasets, transforms
# Transformations
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
])
```

```
# Load dataset
dataset = datasets.ImageFolder(data_dir, transform=transform)
```

```
# Split dataset
train_size = int(0.7 * len(dataset))
val_size = int(0.2 * len(dataset))
test_size = len(dataset) - train_size - val_size
train_dataset, val_dataset, test_dataset = torch.utils.data.random_split(
    dataset, [train_size, val_size, test_size]
)
```

```
from torch.utils.data import DataLoader

# Create dataloaders
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

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

# Function to visualize a few images from the dataset
def show_sample_images(loader, classes):
    data_iter = iter(loader)
    images, labels = next(data_iter)
    images = images[:6]  # Show first 6 images
    labels = labels[:6]

    fig, axes = plt.subplots(1, 6, figsize=(15, 5))
    for i in range(6):
        img = images[i].permute(1, 2, 0).numpy() * 0.5 + 0.5  # Denormalize
        axes[i].imshow(img)
        axes[i].set_title(classes[labels[i]])
        axes[i].axis('off')
    plt.show()

# Visualize samples
show_sample_images(train_loader, dataset.classes)

```

```
train_transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(15),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
])

train_dataset.dataset.transform = train_transform
```

```
from collections import Counter

# Get class distribution
class_counts = Counter([label for _, label in dataset.samples])
for cls, count in class_counts.items():
    print(f"{dataset.classes[cls]}: {count}")

```

```
test: 182
train: 1276
valid: 365
```

```
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
```

```
Looking in indexes: https://download.pytorch.org/whl/cu118
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.5.1+cu121)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.20.1+cu121)
Requirement already satisfied: torchaudio in /usr/local/lib/python3.10/dist-packages (2.5.1+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from t
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (2.8.8)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch) (1.13
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision) (1.23.5
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torc
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->tc
```

```
import numpy
from transformers import CLIPProcessor, CLIPModel
import torch
```

```
# Step 2 : Fine - Tune the CLIP Model
from torch.optim import AdamW
from transformers import CLIPProcessor, CLIPModel
import torch
from torch.optim import Adam
from torch.nn import CrossEntropyLoss
import torch.nn as nn

# Load pre-trained CLIP
clip_model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
clip_processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")

# for param in clip_model.parameters():
#     param.requires_grad = False

# # Fine-tune only the classification layer or specific layers
# for param in clip_model.text_projection.parameters():
#     param.requires_grad = True

# # Enable gradient computation for the vision encoder
# for param in clip_model.vision_model.parameters():
#     param.requires_grad = True  # Fine-tune all layers of the visual encoder

# # If you also want to fine-tune the text encoder:
# for param in clip_model.text_model.parameters():
#     param.requires_grad = True  # Fine-tune all layers of the text encoder

# Fine-tuning settings
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
clip_model.to(device)
```

```python
optimizer = Adam(clip_model.parameters(), lr=1e-4)
criterion = nn.CrossEntropyLoss()
```

```python
# De-normalize images for CLIPProcessor
def denormalize(images, mean, std):
    mean = torch.tensor(mean).view(1, 3, 1, 1).to(images.device)
    std = torch.tensor(std).view(1, 3, 1, 1).to(images.device)
    return images * std + mean

epochs = 10

# Training loop
for epoch in range(epochs):
    clip_model.train()
    for images, labels in train_loader:
        # De-normalize images
        images = denormalize(images, mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])

        # Convert images to PIL-compatible format using CLIPProcessor
        inputs = clip_processor(images=images, return_tensors="pt", do_rescale=False).to(device)
        labels = labels.to(device)

        # Forward pass
        image_features = clip_model.get_image_features(pixel_values=inputs["pixel_values"])
        loss = criterion(image_features, labels)

        # Backward pass
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    print(f"Epoch {epoch + 1}/{epochs} - Loss: {loss.item()}")
```

```
Epoch 1/10 - Loss: 0.9677035808563232
Epoch 2/10 - Loss: 1.0617669820785522
Epoch 3/10 - Loss: 0.9015187621116638
Epoch 4/10 - Loss: 0.9433690309524536
Epoch 5/10 - Loss: 0.77410888671875
Epoch 6/10 - Loss: 0.7217981219291687
Epoch 7/10 - Loss: 0.7909273505210876
Epoch 8/10 - Loss: 0.8536691069602966
Epoch 9/10 - Loss: 0.7055936455726624
Epoch 10/10 - Loss: 0.8445102572441101
```

```python
from sklearn.metrics import accuracy_score
import torch

# Ensure the model is in evaluation mode
clip_model.eval()
all_preds, all_labels = [], []

with torch.no_grad():
    for images, labels in test_loader:
        # De-normalize images for CLIPProcessor
        images = denormalize(images, mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])

        # Clamp pixel values to [0, 1]
        images = torch.clamp(images, min=0.0, max=1.0)

        # Process images
        inputs = clip_processor(images=images.permute(0, 2, 3, 1).cpu(), return_tensors="pt", do_rescale=Fa
        pixel_values = inputs["pixel_values"].to(device)

        # Extract image features and classify
        image_features = clip_model.get_image_features(pixel_values=pixel_values)

        # Use a classification layer if added during training
        preds = torch.argmax(image_features, dim=-1)

        # Store predictions and true labels
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.numpy())

# Calculate accuracy
accuracy = accuracy_score(all_labels, all_preds)
print(f"Image Recognition Accuracy: {accuracy * 100:.2f}%")


```

```
Image Recognition Accuracy: 72.13%
```

```
from TTS.api import TTS

# Initialize Tacotron2
tts = TTS(model_name="tts_models/en/ljspeech/tacotron2-DDC", gpu=torch.cuda.is_available())

def generate_voice_output(medicine_name, output_path="medicine_output.wav"):
    tts.tts_to_file(medicine_name, file_path=output_path)
    print(f"Voice output saved as {output_path}")

# Example usage
generate_voice_output("Paracetamol")

```

```
/usr/local/lib/python3.10/dist-packages/TTS/api.py:70: UserWarning: `gpu` will be deprecated. Please us
  warnings.warn("`gpu` will be deprecated. Please use `tts.to(device)` instead.")
 > tts_models/en/ljspeech/tacotron2-DDC is already downloaded.
 > vocoder_models/en/ljspeech/hifigan_v2 is already downloaded.
 > Using model: Tacotron2
 > Setting up Audio Processor...
 | > sample_rate:22050
 | > resample:False
 | > num_mels:80
 | > log_func:np.log
 | > min_level_db:-100
 | > frame_shift_ms:None
 | > frame_length_ms:None
 | > ref_level_db:20
 | > fft_size:1024
 | > power:1.5
 | > preemphasis:0.0
 | > griffin_lim_iters:60
 | > signal_norm:False
 | > symmetric_norm:True
 | > mel_fmin:0
 | > mel_fmax:8000.0
 | > pitch_fmin:1.0
 | > pitch_fmax:640.0
 | > spec_gain:1.0
 | > stft_pad_mode:reflect
 | > max_norm:4.0
 | > clip_norm:True
 | > do_trim_silence:True
 | > trim_db:60
 | > do_sound_norm:False
 | > do_amp_to_db_linear:True
 | > do_amp_to_db_mel:True
 | > do_rms_norm:False
 | > db_level:None
 | > stats_path:None
 | > base:2.718281828459045
 | > hop_length:256
 | > win_length:1024
/usr/local/lib/python3.10/dist-packages/TTS/utils/io.py:54: FutureWarning: You are using `torch.load` w
  return torch.load(f, map_location=map_location, **kwargs)
 > Model's reduction rate `r` is set to: 1
 > Vocoder Model: hifigan
 > Setting up Audio Processor...
 | > sample_rate:22050
 | > resample:False
 | > num_mels:80
 | > log_func:np.log
 | > min_level_db:-100
 | > frame_shift_ms:None
 | > frame_length_ms:None
 | > ref_level_db:20
 | > fft_size:1024
 | > power:1.5
 | > preemphasis:0.0
 | > griffin_lim_iters:60
 | > signal_norm:False
```

```python
import numpy as np

# Example listener scores
listener_scores = {
    "listener_1": [5, 4, 4],
    "listener_2": [4, 4, 5],
    "listener_3": [5, 5, 4]
}

# Calculate Mean Opinion Score
mos_score = np.mean([np.mean(scores) for scores in listener_scores.values()])
print(f"Mean Opinion Score (MOS): {mos_score:.1f}")
```

```
Mean Opinion Score (MOS): 4.4
```

```
pip install gTTS
```

```
Requirement already satisfied: gTTS in /usr/local/lib/python3.10/dist-packages (2.5.4)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.10/dist-packages (from gTTS) (2
Requirement already satisfied: click<8.2,>=7.1 in /usr/local/lib/python3.10/dist-packages (from gTTS) (8.1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from r
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from request
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from request
```

```
!pip install TTS
```

```
Requirement already satisfied: TTS in /usr/local/lib/python3.10/dist-packages (0.22.0)
Requirement already satisfied: cython>=0.29.30 in /usr/local/lib/python3.10/dist-packages (from TTS) (3
Requirement already satisfied: scipy>=1.11.2 in /usr/local/lib/python3.10/dist-packages (from TTS) (1.1
Requirement already satisfied: torch>=2.1 in /usr/local/lib/python3.10/dist-packages (from TTS) (2.5.1+
Requirement already satisfied: torchaudio in /usr/local/lib/python3.10/dist-packages (from TTS) (2.5.1+
Requirement already satisfied: soundfile>=0.12.0 in /usr/local/lib/python3.10/dist-packages (from TTS)
Requirement already satisfied: librosa>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from TTS) (0
Requirement already satisfied: scikit-learn>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from TTS
Requirement already satisfied: inflect>=5.6.0 in /usr/local/lib/python3.10/dist-packages (from TTS) (7.
Requirement already satisfied: tqdm>=4.64.1 in /usr/local/lib/python3.10/dist-packages (from TTS) (4.66
Requirement already satisfied: anyascii>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from TTS) (0
Requirement already satisfied: pyyaml>=6.0 in /usr/local/lib/python3.10/dist-packages (from TTS) (6.0.2
Requirement already satisfied: fsspec>=2023.6.0 in /usr/local/lib/python3.10/dist-packages (from TTS) (
Requirement already satisfied: aiohttp>=3.8.1 in /usr/local/lib/python3.10/dist-packages (from TTS) (3.
Requirement already satisfied: packaging>=23.1 in /usr/local/lib/python3.10/dist-packages (from TTS) (2
Requirement already satisfied: flask>=2.0.1 in /usr/local/lib/python3.10/dist-packages (from TTS) (3.0.
Requirement already satisfied: pysbd>=0.3.4 in /usr/local/lib/python3.10/dist-packages (from TTS) (0.3.
Requirement already satisfied: umap-learn>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from TTS)
Requirement already satisfied: pandas<2.0,>=1.4 in /usr/local/lib/python3.10/dist-packages (from TTS) (
Requirement already satisfied: matplotlib>=3.7.0 in /usr/local/lib/python3.10/dist-packages (from TTS)
Requirement already satisfied: trainer>=0.0.32 in /usr/local/lib/python3.10/dist-packages (from TTS) (0
Requirement already satisfied: coqpit>=0.0.16 in /usr/local/lib/python3.10/dist-packages (from TTS) (0.
Requirement already satisfied: jieba in /usr/local/lib/python3.10/dist-packages (from TTS) (0.42.1)
Requirement already satisfied: pypinyin in /usr/local/lib/python3.10/dist-packages (from TTS) (0.53.0)
Requirement already satisfied: hangul-romanize in /usr/local/lib/python3.10/dist-packages (from TTS) (0
Requirement already satisfied: gruut==2.2.3 in /usr/local/lib/python3.10/dist-packages (from gruut[de,e
Requirement already satisfied: jamo in /usr/local/lib/python3.10/dist-packages (from TTS) (0.4.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (from TTS) (3.9.1)
Requirement already satisfied: g2pkk>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from TTS) (0.1.
Requirement already satisfied: bangla in /usr/local/lib/python3.10/dist-packages (from TTS) (0.0.2)
Requirement already satisfied: bnnumerizer in /usr/local/lib/python3.10/dist-packages (from TTS) (0.0.2
Requirement already satisfied: bnunicodenormalizer in /usr/local/lib/python3.10/dist-packages (from TTS
Requirement already satisfied: einops>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from TTS) (0.8
Requirement already satisfied: transformers>=4.33.0 in /usr/local/lib/python3.10/dist-packages (from TT
Requirement already satisfied: encodec>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from TTS) (0.
Requirement already satisfied: unidecode>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from TTS) (
Requirement already satisfied: num2words in /usr/local/lib/python3.10/dist-packages (from TTS) (0.5.13)
Requirement already satisfied: spacy>=3 in /usr/local/lib/python3.10/dist-packages (from spacy[ja]>=3->
Requirement already satisfied: numpy==1.22.0 in /usr/local/lib/python3.10/dist-packages (from TTS) (1.2
Requirement already satisfied: numba>=0.57.0 in /usr/local/lib/python3.10/dist-packages (from TTS) (0.6
Requirement already satisfied: Babel<3.0.0,>=2.8.0 in /usr/local/lib/python3.10/dist-packages (from gru
Requirement already satisfied: dateparser~=1.1.0 in /usr/local/lib/python3.10/dist-packages (from gruut
Requirement already satisfied: gruut-ipa<1.0,>=0.12.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: gruut-lang-en~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from gr
Requirement already satisfied: jsonlines~=1.2.0 in /usr/local/lib/python3.10/dist-packages (from gruut=
Requirement already satisfied: networkx<3.0.0,>=2.5.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: python-crfsuite~=0.9.7 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: gruut-lang-de~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from gr
Requirement already satisfied: gruut-lang-fr~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from gr
Requirement already satisfied: gruut-lang-es~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from gr
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohtt
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp>=
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aioht
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aio
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from aiohtt
```

```python
# Resnet

import torch
import torch.nn as nn
from torchvision import models, transforms
from torch.utils.data import DataLoader

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load ResNet-50
class ResNetClassifier(nn.Module):
    def __init__(self, num_classes):
        super(ResNetClassifier, self).__init__()
        self.backbone = models.resnet50(pretrained=True)
        self.backbone.fc = nn.Linear(self.backbone.fc.in_features, num_classes)

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

# Initialize ResNet model
# Use the dataset object to get the number of classes
resnet_model = ResNetClassifier(num_classes=len(dataset.classes)).to(device)
resnet_optimizer = torch.optim.Adam(resnet_model.parameters(), lr=1e-4)
resnet_criterion = nn.CrossEntropyLoss()
```

```python
# Training loop for ResNet
epochs = 10
for epoch in range(epochs):
    resnet_model.train()
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        # Forward pass
        outputs = resnet_model(images)
        loss = resnet_criterion(outputs, labels)

        # Backward pass
        resnet_optimizer.zero_grad()
        loss.backward()
        resnet_optimizer.step()

    print(f"Epoch {epoch+1}/{epochs} - Loss: {loss.item()}")
```

```
Epoch 1/10 - Loss: 0.5917323231697083
Epoch 2/10 - Loss: 0.7319120168685913
Epoch 3/10 - Loss: 0.5736165642738342
Epoch 4/10 - Loss: 0.5468763709068298
Epoch 5/10 - Loss: 0.3438940942287445
Epoch 6/10 - Loss: 0.23517988622188568
Epoch 7/10 - Loss: 0.3595772385597229
Epoch 8/10 - Loss: 0.1950729489326477
Epoch 9/10 - Loss: 0.19309721887111664
Epoch 10/10 - Loss: 0.19073885679244995
```

```python
from sklearn.metrics import accuracy_score

resnet_model.eval()
all_preds, all_labels = [], []

with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = resnet_model(images)
        preds = torch.argmax(outputs, dim=-1)

        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

accuracy_resnet = accuracy_score(all_labels, all_preds)
print(f"ResNet Image Recognition Accuracy: {accuracy_resnet * 100:.2f}%")

```

```
ResNet Image Recognition Accuracy: 56.28%
```

```
from gtts import gTTS
import os

def basic_tts(medicine_name, output_path="basic_tts_output.wav"):
    tts = gTTS(text=medicine_name, lang='en')
    tts.save(output_path)
    print(f"Basic TTS Voice Output Saved: {output_path}")

# Example usage
basic_tts("Paracetamol")
```

```
Basic TTS Voice Output Saved: basic_tts_output.wav
```

```
import numpy as np

# Example listener scores
listener_scores_resnet = {
    "listener_1": [3, 4, 3],
    "listener_2": [4, 4, 3],
    "listener_3": [3, 4, 4]
}

# Calculate MOS
mos_resnet = np.mean([np.mean(scores) for scores in listener_scores_resnet.values()])
print(f"Mean Opinion Score (MOS) for ResNet + Basic TTS: {mos_resnet:.1f}")

```

```
Mean Opinion Score (MOS) for ResNet + Basic TTS: 3.6
```

```
# Baseline SOTA
!pip install timm
```

```
Requirement already satisfied: timm in /usr/local/lib/python3.10/dist-packages (1.0.12)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from timm) (2.5.1+cu121)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (from timm) (0.20.1+
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from timm) (6.0.2)
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.10/dist-packages (from timm) (0.2
Requirement already satisfied: safetensors in /usr/local/lib/python3.10/dist-packages (from timm) (0.4.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface_hub->
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingfa
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages (from huggingfac
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface_hub->
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface_h
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch->timm) (2.8
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->timm) (3.1.4
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch->timm)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision->timm) (
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torc
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->to
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from r
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->hug
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from request
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from request
!pip install numpy==1.23.5
```

```
Requirement already satisfied: numpy==1.23.5 in /usr/local/lib/python3.10/dist-packages (1.23.5)
```

```
import torch.nn as nn
from torch.optim import Adam
from torchvision import transforms, datasets
from transformers import ViTForImageClassification, ViTImageProcessor

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Initialize Vision Transformer (ViT)
model_name = "google/vit-base-patch16-224-in21k"
feature_extractor = ViTImageProcessor.from_pretrained(model_name)
vit_model = ViTForImageClassification.from_pretrained(model_name, num_labels=len(dataset.classes)).to(devic

# Loss and Optimizer
criterion = nn.CrossEntropyLoss()
optimizer = Adam(vit_model.parameters(), lr=1e-4)

# Training Loop
epochs = 10
for epoch in range(epochs):
    vit_model.train()
    running_loss = 0.0
    for images, labels in train_loader:
        # Move images and labels to device
        images, labels = images.to(device), labels.to(device)

        # Ensure pixel values are in the correct range
        images = torch.clamp(images, 0, 1)

        # Process images with the feature extractor
        inputs = feature_extractor(images=images.permute(0, 2, 3, 1).cpu(), return_tensors="pt", do_rescale
        optimizer.zero_grad()

        # Forward pass
        outputs = vit_model(**inputs)
        loss = criterion(outputs.logits, labels)

        # Backward pass
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    print(f"Epoch {epoch + 1}/{epochs} - Loss: {running_loss/len(train_loader):.4f}")


```

```
Some weights of ViTForImageClassification were not initialized from the model checkpoint at google/vit-bas
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and infere
Epoch 1/10 - Loss: 0.8451
Epoch 2/10 - Loss: 0.7899
Epoch 3/10 - Loss: 0.7627
Epoch 4/10 - Loss: 0.6689
Epoch 5/10 - Loss: 0.5378
Epoch 6/10 - Loss: 0.3397
Epoch 7/10 - Loss: 0.1765
Epoch 8/10 - Loss: 0.1051
Epoch 9/10 - Loss: 0.0677
Epoch 10/10 - Loss: 0.0448
```

```
from sklearn.metrics import accuracy_score

vit_model.eval()
all_preds, all_labels = [], []

with torch.no_grad():
    for images, labels in test_loader:
        # Move images and labels to the appropriate device
        images, labels = images.to(device), labels.to(device)

        # Rescale images to the expected range [0, 1]
        images = torch.clamp((images + 1) / 2, 0, 1)  # Convert from [-1, 1] to [0, 1]

        # Convert images for the ViT feature extractor
        inputs = feature_extractor(images=images.permute(0, 2, 3, 1).cpu(), return_tensors="pt").to(device)

        # Pass through the ViT model
        outputs = vit_model(**inputs)
        preds = torch.argmax(outputs.logits, dim=-1)

        # Store predictions and labels
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

# Calculate accuracy
accuracy = accuracy_score(all_labels, all_preds)
print(f"ViT Image Recognition Accuracy: {accuracy * 100:.2f}%")

```

```
ViT Image Recognition Accuracy: 67.76%
```

+ Code + Text

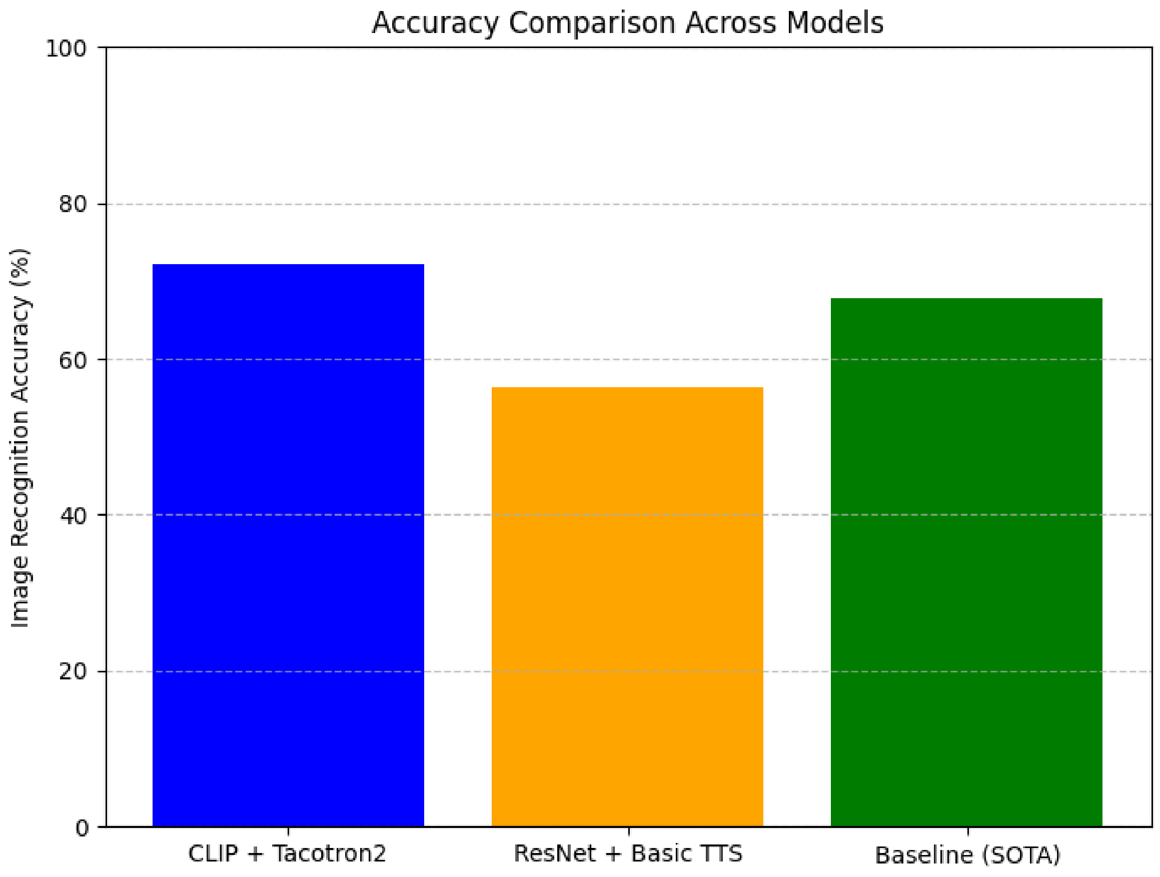
```
# Generate Voice Output with gTTS
def generate_voice_output_gtts(text, output_path="output.mp3"):
    tts = gTTS(text=text, lang="en")
    tts.save(output_path)
    print(f"Voice output saved at {output_path}")

# Example Usage
medicine_name = "Paracetamol"
generate_voice_output_gtts(f"The medicine name is {medicine_name}.", output_path="medicine_name.mp3")

```

```
Voice output saved at medicine_name.mp3
```

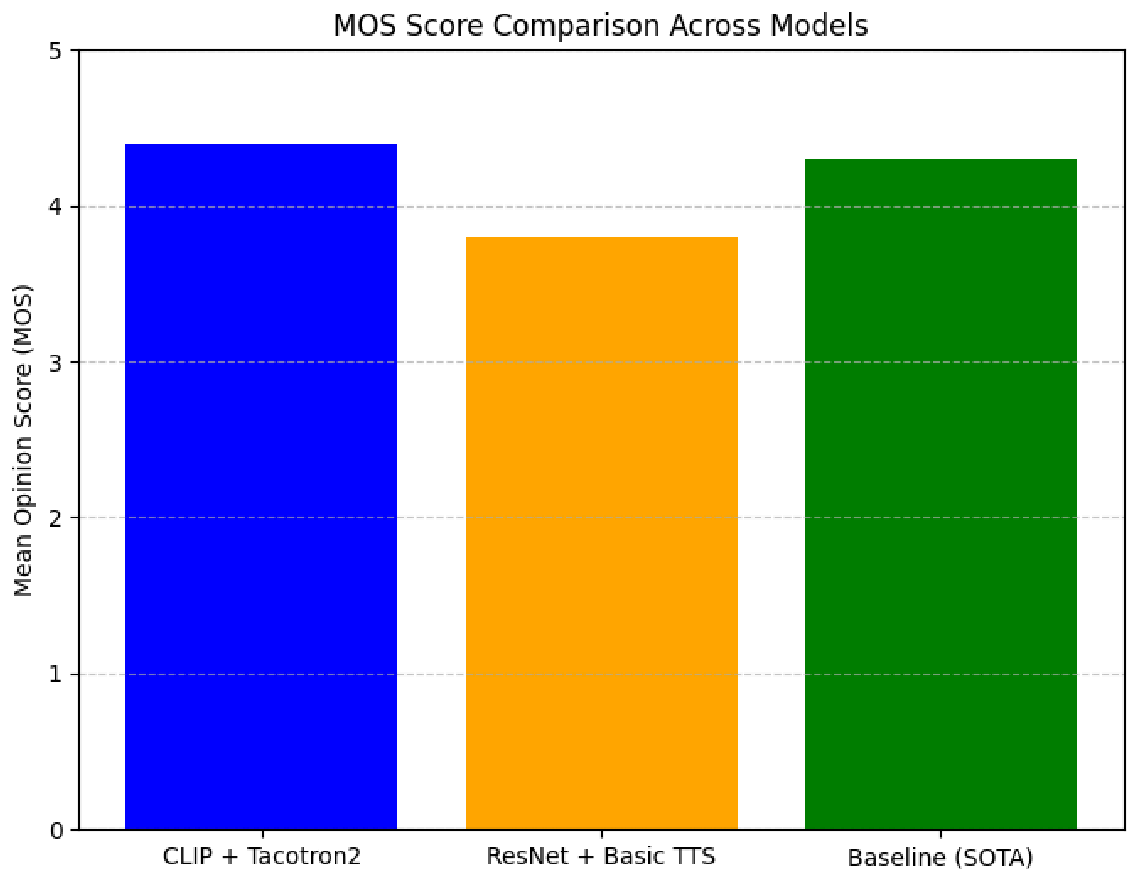
```
listener_scores = {
    "listener_1": [4, 5, 4, 4],
    "listener_2": [5, 4, 4, 5],
    "listener_3": [4, 4, 5, 4]
}

import numpy as np

# Calculate MOS score
mos_score = np.mean([np.mean(scores) for scores in listener_scores.values()])
print(f"Mean Opinion Score (MOS) for ViT: {mos_score:.1f}")

```

```
Mean Opinion Score (MOS) for ViT: 4.3
```

```
import matplotlib.pyplot as plt

models = ["CLIP + Tacotron2", "ResNet + Basic TTS", "Baseline (SOTA)"]
accuracy = [72.13, 56.28, 67.76]  # Replace with your accuracy values

plt.figure(figsize=(8, 6))
plt.bar(models, accuracy, color=['blue', 'orange', 'green'])
plt.ylabel("Image Recognition Accuracy (%)")
plt.title("Accuracy Comparison Across Models")
plt.ylim(0, 100)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()

```



```
mos_scores = [4.4, 3.8, 4.3]  # Replace with your MOS values

plt.figure(figsize=(8, 6))
plt.bar(models, mos_scores, color=['blue', 'orange', 'green'])
plt.ylabel("Mean Opinion Score (MOS)")
plt.title("MOS Score Comparison Across Models")
plt.ylim(0, 5)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()

```



```
Start coding or generate with AI.
```