LAB2:PART2

SAANVI VENKATESH KULKARNI,1RVU23CSE391

a)FAST AI

CODE:

```
# Block 1: Install fastai and duckduckgo_search (for image downloading)
!pip install -Uqq fastai ddgs
import sys
!{sys.executable} -m pip install -U ddgs fastai
# Block 2: Import libraries and setup search function
from fastai.vision.all import *
from ddgs import DDGS
import time

def search_images(term, max_images=20):
    print(f"Searching for '{term}'...")
    urls = []

    with DDGS() as ddgs:
        results = ddgs.images(term, max_results=max_images)

        for r in results:
            urls.append(r["image"])
            time.sleep(0.1)  # slow down to avoid rate limit

    return L(urls)

# Block 3: Download images (Forest vs Bird)
searches = 'forest', 'bird'
path = Path('bird_or_not')

for o in searches:
    dest = path/o
    dest.mkdir(exist_ok=True, parents=True)

    urls = search_images(f'{o} photo', max_images=20)
    download_images(dest, urls=urls)
    resize_images(dest, max_size=400)

    time.sleep(5)  # pause between categories (IMPORTANT)



failed = verify_images(get_image_files(path))
failed.map(Path.unlink)
print(f"Removed {len(failed)} corrupted images")
```

```
# Block 4: Train the Model
# "DataBlock" tells fastai how to read the data (Inputs=Images,
Output=Categories)

from fastai.callback.progress import ProgressCallback

# Create DataLoaders
dls = DataBlock(
    blocks=(ImageBlock, CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.2, seed=42),
    get_y=parent_label,
    item_tfms=Resize(192, method='squish')
).dataloaders(path, bs=32)

# Create learner
learn = vision_learner(dls, resnet18, metrics=error_rate)

learn.remove_cbs(ProgressCallback)

# Train without progress bar
learn.fine_tune(5)

# Block 5: Test with a new image
# We'll search for one new bird image to test

from fastdownload import download_url

urls = search_images('bird photo', max_images=1)
download_url(urls[0], 'test_bird.jpg', show_progress=False)

is_bird,_,probs = learn.predict(PILImage.create('test_bird.jpg'))
print(f"This is a: {is_bird}.")
print(f"Probability it's a bird: {probs[0]:.4f}")
```

OUTPUT:

```
Searching for 'bird photo'...
This is a: bird.
Probability it's a bird: 1.0000
```

b)HUGGING FACE

CODE:

```
!pip install -q transformers gradio

from transformers import pipeline

import gradio as gr
classifier =pipeline("sentiment-analysis")
def analyze_text(text):

    result = classifier(text)[0]

    return f"Label:{result['label']}, Score: {round(result['score'],
4)}"
demo = gr.Interface(

    fn=analyze_text,

    inputs="text",

    outputs="text",

    title="My First AI Sentiment App",

    description="Type a sentence below to see if it's Positive or
Negative!"

)
demo.launch(share=True)
```

OUTPUT: https://ebfb71f98fbad270ff.gradio.live