Fine-tune a small pre-trained language model (e.g., T5-small, DistilBERT) for a specific text classification or summarization task.

```
!pip install -q transformers datasets sentencepiece accelerate
import os, torch
os.environ["WANDB_DISABLED"]="true"
device=torch.device("cuda")

from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
from transformers import DataCollatorForSeq2Seq, TrainingArguments, Trainer

tokenizer=AutoTokenizer.from_pretrained("t5-small")
model=AutoModelForSeq2SeqLM.from_pretrained("t5-small").to(device)

ds=load_dataset("imdb")
ds=ds["train"].select(range(3000)).train_test_split(0.2)

def pre_classify(b):
    x=["classify sentiment: "+t for t in b["text"]]
    y=["positive" if l==1 else "negative" for l in b["label"]]
    t=tokenizer(x, truncation=True)
    l=tokenizer(y, truncation=True)
    t["labels"]=l["input_ids"]
    return t

cls=ds.map(pre_classify, batched=True, remove_columns=["text","label"])
collator=DataCollatorForSeq2Seq(tokenizer, model=model)

args1=TrainingArguments(
    "t5-classification",
    per_device_train_batch_size=4,
    num_train_epochs=1,
    learning_rate=3e-4,
    logging_steps=50,
    fp16=True
```

```
)

trainer1=Trainer(model=model, args=args1, train_dataset=cls["train"], eval_dataset=cls["test"], data_collato
trainer1.train()

def classify(x):
    i=tokenizer("classify sentiment: "+x, return_tensors="pt").to(device)
    o=model.generate(**i, max_new_tokens=5)
    return tokenizer.decode(o[0], skip_special_tokens=True)

ds2=load_dataset("cnn_dailymail","3.0.0")
ds2=ds2["train"].select(range(2000)).train_test_split(0.1)

def pre_summarize(b):
    x=["summarize: "+a for a in b["article"]]
    y=b["highlights"]
    t=tokenizer(x, truncation=True)
    l=tokenizer(y, truncation=True)
    t["labels"]=l["input_ids"]
    return t

sum_ds=ds2.map(pre_summarize, batched=True, remove_columns=["article","highlights"])

args2=TrainingArguments(
    "t5-summarization",
    per_device_train_batch_size=2,
    num_train_epochs=1,
    learning_rate=3e-4,
    logging_steps=50,
    fp16=True
)

trainer2=Trainer(model=model, args=args2, train_dataset=sum_ds["train"], eval_dataset=sum_ds["test"], data_c
trainer2.train()

def summarize(x):
    i=tokenizer("summarize: "+x, return_tensors="pt").to(device)
```

```
    o=model.generate(**i, max_new_tokens=60)
    return tokenizer.decode(o[0], skip_special_tokens=True)
```

```
print("Classification:", classify("The movie was fantastic!"))
print("Summarization:", summarize("The Eiffel Tower is a world-famous landmark visited by millions of touris

```

```
Classification: negative
Summarization: Eiffel Tower is a world-famous landmark visited every year.
```

Step	Training Loss
50	2.262200
100	0.000000
150	0.000000