Transfer Learning vs. Fine-Tuning

This notebook shows:

1. **Transfer Learning**
   Loading a pretrained model and using it out-of-the-box (feature extraction).
2. **Fine-Tuning**
   Further training a pretrained model on a labeled dataset for a specific task.

We will use the IMDb sentiment dataset (via `datasets`) and `bert-base-uncased` as our example.

```
# 1. Install dependencies (if needed)
# !pip install transformers datasets evaluate
```

1. Load Model & Tokenizer

What can I help you build?

```
from transformers import BertTokenizer, BertForSequenceClassification

model_name = "bert-base-uncased"

# Tokenizer and model
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name)
# Note: this model is un-fine-tuned—just the pretrained weights.
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: Use
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access pu
  warnings.warn(
```

tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 5.76kB/s]

vocab.txt: 100% 232k/232k [00:00<00:00, 567kB/s]

tokenizer.json: 100% 466k/466k [00:00<00:00, 981kB/s]

config.json: 100% 570/570 [00:00<00:00, 11.0kB/s]

model.safetensors: 100% 440M/440M [00:08<00:00, 86.2MB/s]

```
Some weights of BertForSequenceClassification were not initialized from the mo
You should probably TRAIN this model on a down-stream task to be able to use i
```

2. Prepare Data (IMDb)

```
from datasets import load_dataset

# Load small subset for quick demo
raw_datasets = load_dataset("imdb", split={"train": "train[:1%]", "test": "test[:1%
def tokenize_fn(ex):
    return tokenizer(ex["text"], truncation=True, padding="max_length", max_length=

tokenized = {k: raw_datasets[k].map(tokenize_fn, batched=True) for k in raw_dataset
tokenized["train"] = tokenized["train"].remove_columns(["text"]).with_format("torch
tokenized["test"]  = tokenized["test"].remove_columns(["text"]).with_format("torch"
```

•••

3. Transfer Learning (Feature Extraction)

Here we will **not** train further; we simply run inference on test data with the pretrained model (as a “feature extractor + head”):

- We freeze all parameters so no training happens.
- We just look at the raw predictions.

```
!pip install torch
```

```
•••  Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packa
     Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-pa
```

```
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/p
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-pack
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_6
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.m
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.m
Collecting nvidia-curand-cu12==10.3.5.147 (from torch)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.wh
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.wh
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/l
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/py
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.w
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (3
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 5.3 MB/s eta 0:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.wh
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 25.5 MB/s eta 0:00:
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.wh
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 22.1 MB/s eta 0:00:
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 44.1 MB/s eta 0:0
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (66
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.9 MB/s eta 0:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (21
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 7.1 MB/s eta 0:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.2 MB/s eta 0:00:
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 8.1 MB/s eta 0:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.wh
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.4 MB/s eta 0:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 37.1 MB/s eta 0:00:
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12, nv
  Attempting uninstall: nvidia-nvjitlink-cu12
```

```
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
```

```
import torch

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

model.eval()
inputs = {k: v[:8] for k,v in tokenized["test"].items()}  # small batch
with torch.no_grad():
    logits = model(**inputs).logits

preds = torch.argmax(logits, dim=-1)
print("Raw (un-fine-tuned) predictions:", preds.tolist())
```

4. Fine-Tuning

Now we will fine-tune the same pretrained model on the small IMDb subset:

- Unfreeze all layers.
- Train for 1 epoch for demo purposes.

```
from transformers import Trainer, TrainingArguments

# Re-load a fresh copy of the model (unfrozen)
model_ft = BertForSequenceClassification.from_pretrained(model_name)

training_args = TrainingArguments(
    output_dir="ft_results",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=1,
    logging_steps=10,
    evaluation_strategy="epoch",
    save_strategy="no",
    learning_rate=2e-5,
)

trainer = Trainer(
    model=model_ft,
    args=training_args,
    train_dataset=tokenized["train"],
    eval_dataset=tokenized["test"],
)

trainer.train()
```

5. Evaluate Fine-Tuned Model

```
metrics = trainer.evaluate()
print(f"Fine-tuned model eval accuracy: {metrics['eval_accuracy']:.3f}")
```

6. Summary

- **Transfer Learning (feature extraction)**
  We loaded the pretrained BERT, froze it, and performed inference directly. No task-specific training took place.
- **Fine-Tuning**
  We continued training the pretrained BERT on our small IMDb subset, adjusting all its weights for better sentiment performance.

Fine-tuning is a **specific** form of transfer learning where you adapt the pretrained weights to a downstream task. Transfer learning more broadly can also mean using the pretrained model purely as a fixed feature extractor or initializing only some layers.