tesss

November 6, 2024

1 Fine-tune BLIP using Hugging Face transformers, datasets, peft and bitsandbytes

Let's leverage recent advances from Parameter Efficient Fine-Tuning methods to fine-tune a large image to text model! We will show through this tutorial that it is possible to fine-tune a 3B scale model (~6GB in half-precision)

Here we will use a dummy dataset of football players that is uploaded on the Hub. The images have been manually selected together with the captions. Check the documentation on how to create and upload your own image-text dataset.

1.1 Set-up environment

[ ]:
```
!pip install Pillow
```

[1]:
```
!pip install -q git+https://github.com/huggingface/peft.git transformers␣
↪bitsandbytes datasets
```

1.2 Load the image captioning dataset

Let's load the image captioning dataset, you just need few lines of code for that.

[1]:
```
from datasets import load_dataset

dataset = load_dataset("Pranavkpba2000/skin_cancer_small_dataset",␣
↪split="train")
```

```
/home/admincit/DoRA-LoRA/.conda/lib/python3.11/site-packages/tqdm/auto.py:21:
TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

Let's retrieve the caption of the first example:

[7]:
```
new_dataset[0]["label"]
```

[7]: 0

And the corresponding image

[6]:

```python
new_dataset[0]["image"]
```

[6]:

[5]:

```python
# Assuming dataset is a list of dictionaries like:
# dataset = [{"image": image_data, "label": 0}, {"image": image_data, "label":␣
↪1}, ...]

from collections import defaultdict

def extract_images_per_class(dataset, num_images_per_class=4):
    """
    Extracts a specified number of images from each class and creates a new␣
↪dataset.

    Args:
    - dataset: Original dataset containing images and labels.
    - num_images_per_class: The number of images to extract from each class.

    Returns:
    - new_dataset: A new dataset with the extracted images and labels.
    """
    # Dictionary to store images by class (labels 0-7)
    class_to_images = defaultdict(list)

    # Group images by their labels
    for item in dataset:
        class_to_images[item["label"]].append(item)
```

```python
    # Create a new dataset with only 4 images per class (0 to 7)
    new_dataset = []
    for label in range(8):  # There are 8 labels (0 to 7)
        # Get the images for the current label, and ensure we take at most 4
        images_of_class = class_to_images[label][:num_images_per_class]

        # Add these images to the new dataset
        new_dataset.extend(images_of_class)

    return new_dataset

# Example usage:
new_dataset = extract_images_per_class(dataset)

# Print the new dataset
for item in new_dataset:
    print(f"Label: {item['label']}, Image: {item['image']}")
```

```
Label: 0, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FAD0BBC3450>
Label: 0, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FAD0BBC3850>
Label: 0, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FAD0BBC3E90>
Label: 0, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FAD0BBC2550>
Label: 1, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACEEDB50D0>
Label: 1, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACEEDB5690>
Label: 1, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACEEDB5C10>
Label: 1, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACEEDB6190>
Label: 2, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACDD151F50>
Label: 2, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACDD152510>
Label: 2, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACDD152AD0>
Label: 2, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACDD153010>
Label: 3, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACCB512250>
Label: 3, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACCB512810>
Label: 3, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
```

```
at 0x7FACCB512D90>
Label: 3, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACCB513350>
Label: 4, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACC3741D90>
Label: 4, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACC3742350>
Label: 4, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACC37428D0>
Label: 4, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACC3742E50>
Label: 5, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACB65FE050>
Label: 5, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACB65FE610>
Label: 5, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACB65FEB90>
Label: 5, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACB65FF110>
Label: 6, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACA69C2350>
Label: 6, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACA69C2910>
Label: 6, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACA69C2E90>
Label: 6, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FACA69C3410>
Label: 7, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FAC96D865D0>
Label: 7, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FAC96D86B50>
Label: 7, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FAC96D870D0>
Label: 7, Image: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=224x224
at 0x7FAC96D87650>
```

1.3 Create PyTorch Dataset

Let's define below the dataset as well as the data collator!

[13]:
```python
from torch.utils.data import Dataset,DataLoader
import torch

class ImageCaptioningDataset(Dataset):
    def __init__(self, dataset, processor):
        self.dataset = dataset
        self.processor = processor
```

```python
    def __len__(self):
        return len(self.dataset)

    def __getitem__(self, idx):
        item = self.dataset[idx]

        # Process the image data
        encoding = self.processor(images=item["image"], padding="max_length", return_tensors="pt")

        # Remove batch dimension
        encoding = {k: v.squeeze() for k, v in encoding.items()}

        # Map labels to prompts
        label_to_prompt = {
            0: "The detected disease is Actinic Keratosis (AK), a precancerous condition characterized by scaly, crusty patches of skin. It can potentially develop into skin cancer if not treated.",
            1: "The detected disease is Basal Cell Carcinoma (BCC), a common form of skin cancer that typically appears as a pearly or waxy bump. It is slow-growing and usually doesn't spread.",
            2: "The detected disease is Benign Keratosis (BKL), a non-cancerous skin growth that often appears as a wart or mole. These are usually harmless but may need monitoring.",
            3: "The detected disease is Melanoma (MEL), a serious and aggressive form of skin cancer that often presents as a mole with irregular edges or different colors. It requires immediate attention.",
            4: "The detected disease is Nevus (NV), commonly known as a mole, which can vary in color and size. Most are harmless, but changes in size, shape, or color may require evaluation.",
            5: "The detected disease is Squamous Cell Carcinoma (SCC), a type of skin cancer that often appears as a firm, red nodule or scaly patch. It can spread if not treated early.",
            6: "The detected disease is Dermatofibroma (DF), a benign growth on the skin that is typically brown or tan. These are harmless and generally do not require treatment.",
            7: "The detected disease is Vascular Lesions (VASC), abnormal blood vessel growths that can appear as red or purple spots on the skin. They are generally harmless but may require treatment for cosmetic reasons."
        }

        # Assign the appropriate prompt based on the label
        encoding["text"] = str(label_to_prompt.get(item["label"], "Unknown condition. Please provide more details."))

        return encoding
```

```python
def collate_fn(batch):
    # Pad the input_ids and attention_mask
    processed_batch = {}
    for key in batch[0].keys():
        if key != "text":
            processed_batch[key] = torch.stack([example[key] for example in␣
 ↪batch])
        else:
            text_inputs = processor.tokenizer(
                [example["text"] for example in batch], padding=True,␣
 ↪return_tensors="pt"
            )
            processed_batch["input_ids"] = text_inputs["input_ids"]
            processed_batch["attention_mask"] = text_inputs["attention_mask"]
    return processed_batch
```

1.4 Load model and processor

[9]:
```python
from transformers import AutoProcessor, Blip2ForConditionalGeneration

processor = AutoProcessor.from_pretrained("Salesforce/blip2-opt-2.7b")
model = Blip2ForConditionalGeneration.from_pretrained("ybelkada/blip2-opt-2.
 ↪7b-fp16-sharded", device_map="auto", load_in_8bit=True)
```

```
The `load_in_4bit` and `load_in_8bit` arguments are deprecated and will be
removed in the future versions. Please, pass a `BitsAndBytesConfig` object in
`quantization_config` argument instead.
Loading checkpoint shards: 100%|      | 8/8 [00:03<00:00,  2.11it/s]
```

Next we define our LoraConfig object. We explicitly tell

[10]:
```python
from peft import LoraConfig, get_peft_model

# Let's define the LoraConfig
config = LoraConfig(
    r=16,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none",
    target_modules=["q_proj", "k_proj"],
    use_dora=True
)

model = get_peft_model(model, config)
model.print_trainable_parameters()
```

```
trainable params: 5,406,720 || all params: 3,750,086,656 || trainable%: 0.1442
```

Now that we have loaded the processor, let's load the dataset and the dataloader:

[14]:
```
train_dataset = ImageCaptioningDataset(new_dataset, processor)
train_dataloader = DataLoader(train_dataset, shuffle=True, batch_size=3,␣
↪collate_fn=collate_fn)
```

1.5 Train the model

Let's train the model! Run the simply the cell below for training the model

[15]:
```
import torch

optimizer = torch.optim.Adam(model.parameters(), lr=5e-4)

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

model.train()

for epoch in range(50):
  print("Epoch:", epoch)
  for idx, batch in enumerate(train_dataloader):
    input_ids = batch.pop("input_ids").to(device)
    pixel_values = batch.pop("pixel_values").to(device, torch.float16)

    outputs = model(input_ids=input_ids,
                    pixel_values=pixel_values,
                    labels=input_ids)

    loss = outputs.loss

    print("Loss:", loss.item())

    loss.backward()

    optimizer.step()
    optimizer.zero_grad()
```

```
Epoch: 0

Expanding inputs for image tokens in BLIP-2 should be done in processing. Please
follow instruction here (https://gist.github.com/zucchini-
nlp/e9f20b054fa322f84ac9311d9ab67042) to update your BLIP-2 model. Using
processors without these attributes in the config is deprecated and will throw
an error in v4.47.

Loss: 3.650390625
Loss: 3.77734375
Loss: 3.263671875
Loss: 3.1171875
```

```
Loss: 2.96484375
Loss: 2.388671875
Loss: 2.591796875
Loss: 2.408203125
Loss: 1.85546875
Loss: 2.419921875
Loss: 1.7138671875
Epoch: 1
Loss: 1.8408203125
Loss: 1.94140625
Loss: 1.8125
Loss: 1.4033203125
Loss: 1.716796875
Loss: 1.6279296875
Loss: 1.380859375
Loss: 1.392578125
Loss: 1.34375
Loss: 1.357421875
Loss: 1.12109375
Epoch: 2
Loss: 1.01953125
Loss: 1.05859375
Loss: 1.029296875
Loss: 0.96240234375
Loss: 0.88818359375
Loss: 1.013671875
Loss: 0.8525390625
Loss: 0.9345703125
Loss: 0.82568359375
Loss: 0.79052734375
Loss: 0.8017578125
Epoch: 3
Loss: 0.7568359375
Loss: 0.6083984375
Loss: 0.71044921875
Loss: 0.499267578125
Loss: 0.6962890625
Loss: 0.61279296875
Loss: 0.61181640625
Loss: 0.76806640625
Loss: 0.583984375
Loss: 0.6669921875
Loss: 0.464111328125
Epoch: 4
Loss: 0.4013671875
Loss: 0.48583984375
Loss: 0.52490234375
Loss: 0.434326171875
```

```
Loss: 0.429443359375
Loss: 0.416259765625
Loss: 0.331298828125
Loss: 0.5009765625
Loss: 0.442626953125
Loss: 0.63427734375
Loss: 0.254638671875
Epoch: 5
Loss: 0.468017578125
Loss: 0.34130859375
Loss: 0.322021484375
Loss: 0.423583984375
Loss: 0.28173828125
Loss: 0.417236328125
Loss: 0.2283935546875
Loss: 0.377197265625
Loss: 0.318359375
Loss: 0.434326171875
Loss: 0.249267578125
Epoch: 6
Loss: 0.294677734375
Loss: 0.283203125
Loss: 0.2261962890625
Loss: 0.232177734375
Loss: 0.18603515625
Loss: 0.321533203125
Loss: 0.3603515625
Loss: 0.2088623046875
Loss: 0.2242431640625
Loss: 0.365966796875
Loss: 0.21142578125
Epoch: 7
Loss: 0.31005859375
Loss: 0.1868896484375
Loss: 0.37255859375
Loss: 0.1507568359375
Loss: 0.2403564453125
Loss: 0.247314453125
Loss: 0.172607421875
Loss: 0.1883544921875
Loss: 0.249267578125
Loss: 0.1748046875
Loss: 0.085693359375
Epoch: 8
Loss: 0.216552734375
Loss: 0.27490234375
Loss: 0.2498779296875
Loss: 0.2015380859375
```

```
Loss: 0.267333984375
Loss: 0.2301025390625
Loss: 0.10943603515625
Loss: 0.2200927734375
Loss: 0.1685791015625
Loss: 0.0625
Loss: 0.263916015625
Epoch: 9
Loss: 0.276611328125
Loss: 0.1695556640625
Loss: 0.1646728515625
Loss: 0.2001953125
Loss: 0.1322021484375
Loss: 0.16796875
Loss: 0.135986328125
Loss: 0.167236328125
Loss: 0.2154541015625
Loss: 0.1341552734375
Loss: 0.0948486328125
Epoch: 10
Loss: 0.1710205078125
Loss: 0.1771240234375
Loss: 0.0877685546875
Loss: 0.127197265625
Loss: 0.127197265625
Loss: 0.1929931640625
Loss: 0.162353515625
Loss: 0.1334228515625
Loss: 0.104248046875
Loss: 0.1922607421875
Loss: 0.09332275390625
Epoch: 11
Loss: 0.1712646484375
Loss: 0.07861328125
Loss: 0.065673828125
Loss: 0.10284423828125
Loss: 0.163330078125
Loss: 0.10614013671875
Loss: 0.1224365234375
Loss: 0.088134765625
Loss: 0.09442138671875
Loss: 0.1611328125
Loss: 0.108642578125
Epoch: 12
Loss: 0.11749267578125
Loss: 0.135986328125
Loss: 0.083740234375
Loss: 0.129638671875
```

```
Loss: 0.08367919921875
Loss: 0.15283203125
Loss: 0.1669921875
Loss: 0.0921630859375
Loss: 0.15625
Loss: 0.1522216796875
Loss: 0.047821044921875
Epoch: 13
Loss: 0.11138916015625
Loss: 0.0794677734375
Loss: 0.124755859375
Loss: 0.07879638671875
Loss: 0.08203125
Loss: 0.07830810546875
Loss: 0.12548828125
Loss: 0.12005615234375
Loss: 0.12408447265625
Loss: 0.10015869140625
Loss: 0.037109375
Epoch: 14
Loss: 0.10284423828125
Loss: 0.11944580078125
Loss: 0.066162109375
Loss: 0.04144287109375
Loss: 0.093505859375
Loss: 0.04815673828125
Loss: 0.04376220703125
Loss: 0.06890869140625
Loss: 0.104736328125
Loss: 0.11846923828125
Loss: 0.0865478515625
Epoch: 15
Loss: 0.0953369140625
Loss: 0.077392578125
Loss: 0.05035400390625
Loss: 0.1038818359375
Loss: 0.09600830078125
Loss: 0.043853759765625
Loss: 0.09912109375
Loss: 0.08575439453125
Loss: 0.090576171875
Loss: 0.1419677734375
Loss: 0.053985595703125
Epoch: 16
Loss: 0.0855712890625
Loss: 0.043182373046875
Loss: 0.0545654296875
Loss: 0.06170654296875
```

```
Loss: 0.0731201171875
Loss: 0.057220458984375
Loss: 0.07366943359375
Loss: 0.0919189453125
Loss: 0.07794189453125
Loss: 0.0716552734375
Loss: 0.0176239013671875
Epoch: 17
Loss: 0.020111083984375
Loss: 0.0728759765625
Loss: 0.0302886962890625
Loss: 0.086181640625
Loss: 0.0911865234375
Loss: 0.07891845703125
Loss: 0.0322265625
Loss: 0.051727294921875
Loss: 0.06097412109375
Loss: 0.0228118896484375
Loss: 0.026336669921875
Epoch: 18
Loss: 0.039031982421875
Loss: 0.0277252197265625
Loss: 0.0792236328125
Loss: 0.06866455078125
Loss: 0.0457763671875
Loss: 0.06121826171875
Loss: 0.064208984375
Loss: 0.055450439453125
Loss: 0.06097412109375
Loss: 0.035430908203125
Loss: 0.014556884765625
Epoch: 19
Loss: 0.0687255859375
Loss: 0.06414794921875
Loss: 0.051239013671875
Loss: 0.015838623046875
Loss: 0.06365966796875
Loss: 0.05023193359375
Loss: 0.062744140625
Loss: 0.042236328125
Loss: 0.06768798828125
Loss: 0.054595947265625
Loss: 0.03533935546875
Epoch: 20
Loss: 0.043060302734375
Loss: 0.01311492919921875
Loss: 0.032257080078125
Loss: 0.058380126953125
```

```
Loss: 0.05206298828125
Loss: 0.08868408203125
Loss: 0.0537109375
Loss: 0.04583740234375
Loss: 0.011566162109375
Loss: 0.047088623046875
Loss: 0.056976318359375
Epoch: 21
Loss: 0.03021240234375
Loss: 0.0222320556640625
Loss: 0.04888916015625
Loss: 0.04742431640625
Loss: 0.066162109375
Loss: 0.039520263671875
Loss: 0.0250396728515625
Loss: 0.039398193359375
Loss: 0.0284576416015625
Loss: 0.04693603515625
Loss: 0.0345458984375
Epoch: 22
Loss: 0.0399169921875
Loss: 0.050811767578125
Loss: 0.043975830078125
Loss: 0.0234832763671875
Loss: 0.032073974609375
Loss: 0.037078857421875
Loss: 0.048797607421875
Loss: 0.042236328125
Loss: 0.034088134765625
Loss: 0.056427001953125
Loss: 0.00933074951171875
Epoch: 23
Loss: 0.061126708984375
Loss: 0.036224365234375
Loss: 0.0623779296875
Loss: 0.060089111328125
Loss: 0.04315185546875
Loss: 0.048004150390625
Loss: 0.043365478515625
Loss: 0.032928466796875
Loss: 0.03076171875
Loss: 0.00714111328125
Loss: 0.004970550537109375
Epoch: 24
Loss: 0.03558349609375
Loss: 0.02239990234375
Loss: 0.049713134765625
Loss: 0.05462646484375
```

```
Loss: 0.0390625
Loss: 0.0280609130859375
Loss: 0.033905029296875
Loss: 0.030853271484375
Loss: 0.037811279296875
Loss: 0.031829833984375
Loss: 0.0298004150390625
Epoch: 25
Loss: 0.0286102294921875
Loss: 0.0269927978515625
Loss: 0.022064208984375
Loss: 0.03302001953125
Loss: 0.010955810546875
Loss: 0.02923583984375
Loss: 0.0399169921875
Loss: 0.021820068359375
Loss: 0.05255126953125
Loss: 0.035552978515625
Loss: 0.0240325927734375
Epoch: 26
Loss: 0.038787841796875
Loss: 0.01160430908203125
Loss: 0.04412841796875
Loss: 0.037841796875
Loss: 0.03302001953125
Loss: 0.036376953125
Loss: 0.049835205078125
Loss: 0.0258331298828125
Loss: 0.039031982421875
Loss: 0.02471923828125
Loss: 0.032501220703125
Epoch: 27
Loss: 0.0234527587890625
Loss: 0.0322265625
Loss: 0.03314208984375
Loss: 0.0284423828125
Loss: 0.04656982421875
Loss: 0.04766845703125
Loss: 0.0261688232421875
Loss: 0.0400390625
Loss: 0.04083251953125
Loss: 0.0341796875
Loss: 0.03521728515625
Epoch: 28
Loss: 0.036285400390625
Loss: 0.037567138671875
Loss: 0.0270843505859375
Loss: 0.047149658203125
```

```
Loss: 0.0297698974609375
Loss: 0.0372314453125
Loss: 0.0430908203125
Loss: 0.03277587890625
Loss: 0.0328369140625
Loss: 0.02239990234375
Loss: 0.02423095703125
Epoch: 29
Loss: 0.01378631591796875
Loss: 0.0186767578125
Loss: 0.064208984375
Loss: 0.04095458984375
Loss: 0.028564453125
Loss: 0.047088623046875
Loss: 0.048675537109375
Loss: 0.0176544189453125
Loss: 0.0280303955078125
Loss: 0.0186309814453125
Loss: 0.00667572021484375
Epoch: 30
Loss: 0.06317138671875
Loss: 0.0477294921875
Loss: 0.025177001953125
Loss: 0.10858154296875
Loss: 0.046417236328125
Loss: 0.0416259765625
Loss: 0.056671142578125
Loss: 0.05096435546875
Loss: 0.058349609375
Loss: 0.0555419921875
Loss: 0.05364990234375
Epoch: 31
Loss: 0.028076171875
Loss: 0.060089111328125
Loss: 0.04974365234375
Loss: 0.06939697265625
Loss: 0.0548095703125
Loss: 0.04608154296875
Loss: 0.01837158203125
Loss: 0.06866455078125
Loss: 0.05401611328125
Loss: 0.040985107421875
Loss: 0.03564453125
Epoch: 32
Loss: 0.047027587890625
Loss: 0.030181884765625
Loss: 0.046844482421875
Loss: 0.052764892578125
```

```
Loss: 0.0458984375
Loss: 0.033538818359375
Loss: 0.032501220703125
Loss: 0.029998779296875
Loss: 0.03997802734375
Loss: 0.0267181396484375
Loss: 0.0361328125
Epoch: 33
Loss: 0.023406982421875
Loss: 0.03131103515625
Loss: 0.044708251953125
Loss: 0.03936767578125
Loss: 0.03558349609375
Loss: 0.042633056640625
Loss: 0.036376953125
Loss: 0.0193023681640625
Loss: 0.0262298583984375
Loss: 0.03955078125
Loss: 0.019195556640625
Epoch: 34
Loss: 0.022552490234375
Loss: 0.03045654296875
Loss: 0.01165008544921875
Loss: 0.028839111328125
Loss: 0.03973388671875
Loss: 0.028839111328125
Loss: 0.0292816162109375
Loss: 0.031829833984375
Loss: 0.017578125
Loss: 0.0235443115234375
Loss: 0.0185699462890625
Epoch: 35
Loss: 0.021728515625
Loss: 0.0213470458984375
Loss: 0.03973388671875
Loss: 0.0186614990234375
Loss: 0.01776123046875
Loss: 0.0230865478515625
Loss: 0.020599365234375
Loss: 0.0220794677734375
Loss: 0.026123046875
Loss: 0.04229736328125
Loss: 0.0122528076171875
Epoch: 36
Loss: 0.0178375244140625
Loss: 0.01250457763671875
Loss: 0.0279083251953125
Loss: 0.016326904296875
```

```
Loss: 0.0211639404296875
Loss: 0.0209503173828125
Loss: 0.026580810546875
Loss: 0.031829833984375
Loss: 0.0283203125
Loss: 0.0277862548828125
Loss: 0.028472900390625
Epoch: 37
Loss: 0.031494140625
Loss: 0.0222320556640625
Loss: 0.0195159912109375
Loss: 0.023345947265625
Loss: 0.0305023193359375
Loss: 0.0428466796875
Loss: 0.01885986328125
Loss: 0.01277923583984375
Loss: 0.0194549560546875
Loss: 0.018096923828125
Loss: 0.0308685302734375
Epoch: 38
Loss: 0.027801513671875
Loss: 0.034210205078125
Loss: 0.02410888671875
Loss: 0.01190185546875
Loss: 0.017730712890625
Loss: 0.02386474609375
Loss: 0.0223846435546875
Loss: 0.02935791015625
Loss: 0.021270751953125
Loss: 0.014617919921875
Loss: 0.009246826171875
Epoch: 39
Loss: 0.016021728515625
Loss: 0.0095672607421875
Loss: 0.019775390625
Loss: 0.012939453125
Loss: 0.032379150390625
Loss: 0.0266571044921875
Loss: 0.012237548828125
Loss: 0.020538330078125
Loss: 0.023101806640625
Loss: 0.0183563232421875
Loss: 0.02496337890625
Epoch: 40
Loss: 0.0214080810546875
Loss: 0.00978851318359375
Loss: 0.0177764892578125
Loss: 0.019256591796875
```

```
Loss: 0.0164794921875
Loss: 0.014404296875
Loss: 0.01546478271484375
Loss: 0.0149688720703125
Loss: 0.015411376953125
Loss: 0.0167083740234375
Loss: 0.01406097412109375
Epoch: 41
Loss: 0.01120758056640625
Loss: 0.01555633544921875
Loss: 0.0167694091796875
Loss: 0.0189971923828125
Loss: 0.0188140869140625
Loss: 0.01485443115234375
Loss: 0.0255279541015625
Loss: 0.01010894775390625
Loss: 0.016815185546875
Loss: 0.0240631103515625
Loss: 0.011383056640625
Epoch: 42
Loss: 0.00954437255859375
Loss: 0.01165008544921875
Loss: 0.0217742919921875
Loss: 0.0195770263671875
Loss: 0.01313018798828125
Loss: 0.01529693603515625
Loss: 0.0217742919921875
Loss: 0.007671356201171875
Loss: 0.0175933837890625
Loss: 0.01103973388671875
Loss: 0.0104217529296875
Epoch: 43
Loss: 0.0107269287109375
Loss: 0.01074981689453125
Loss: 0.00830078125
Loss: 0.0162200927734375
Loss: 0.0111846923828125
Loss: 0.024017333984375
Loss: 0.0172271728515625
Loss: 0.01366424560546875
Loss: 0.0192718505859375
Loss: 0.01418304443359375
Loss: 0.01183319091796875
Epoch: 44
Loss: 0.009674072265625
Loss: 0.0233917236328125
Loss: 0.006427764892578125
Loss: 0.0214080810546875
```

```
Loss: 0.01214599609375
Loss: 0.009185791015625
Loss: 0.01424407958984375
Loss: 0.0108489990234375
Loss: 0.00537109375
Loss: 0.01328277587890625
Loss: 0.0115814208984375
Epoch: 45
Loss: 0.01459503173828125
Loss: 0.007694244384765625
Loss: 0.0139923095703125
Loss: 0.006671905517578125
Loss: 0.0121917724609375
Loss: 0.0104827880859375
Loss: 0.0126800537109375
Loss: 0.01329803466796875
Loss: 0.016815185546875
Loss: 0.0166015625
Loss: 0.00612640380859375
Epoch: 46
Loss: 0.01532745361328125
Loss: 0.01168060302734375
Loss: 0.00859832763671875
Loss: 0.0157318115234375
Loss: 0.0072174072265625
Loss: 0.014190673828125
Loss: 0.009185791015625
Loss: 0.00794219970703125
Loss: 0.0124053955078125
Loss: 0.0126953125
Loss: 0.01367950439453125
Epoch: 47
Loss: 0.00861358642578125
Loss: 0.01355743408203125
Loss: 0.01629638671875
Loss: 0.01947021484375
Loss: 0.01047515869140625
Loss: 0.0183868408203125
Loss: 0.001873016357421875
Loss: 0.01343536376953125
Loss: 0.0141448974609375
Loss: 0.0140228271484375
Loss: 0.01436614990234375
Epoch: 48
Loss: 0.007282257080078125
Loss: 0.0187225341796875
Loss: 0.01004791259765625
Loss: 0.015625
```

```
Loss: 0.0111846923828125
Loss: 0.01235198974609375
Loss: 0.01299285888671875
Loss: 0.01039886474609375
Loss: 0.01416778564453125
Loss: 0.00791168212890625
Loss: 0.0072784423828125
Epoch: 49
Loss: 0.006855010986328125
Loss: 0.0127410888671875
Loss: 0.0170135498046875
Loss: 0.007205963134765625
Loss: 0.0081787109375
Loss: 0.01494598388671875
Loss: 0.0098724365234375
Loss: 0.0155487060546875
Loss: 0.005764007568359375
Loss: 0.01068115234375
Loss: 0.005847930908203125
```

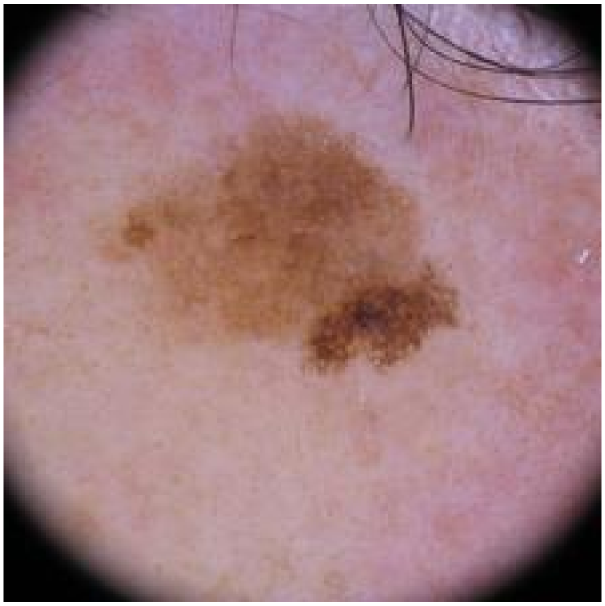
1.6 Inference

Let's check the results on our train dataset

[18]:
```
# load image
example = dataset[500]
image = example["image"]

image
```

[18]:



[19]:
```
example['label']
```

[19]: 0

[20]:
```
# prepare image for the model
inputs = processor(images=image, return_tensors="pt").to(device, torch.float16)
pixel_values = inputs.pixel_values

generated_ids = model.generate(pixel_values=pixel_values, max_length=25)
generated_caption = processor.batch_decode(generated_ids,␣
↪skip_special_tokens=True)[0]
print(generated_caption)
```

```
The detected disease is Nevus (NV), commonly known as a mole, which can vary in
color and size.
```

1.7 Push to Hub

[ ]:
```
from huggingface_hub import notebook_login

notebook_login()
```

[ ]:
```
model.push_to_hub("ybelkada/blip2-opt-2.7b-football-captions-adapters")
```

1.8 Load from the Hub

Once trained you can push the model and processor on the Hub to use them later. Meanwhile you can play with the model that we have fine-tuned! Please restart the runtime to run the cell below!

[ ]:
```
from transformers import Blip2ForConditionalGeneration, AutoProcessor
from peft import PeftModel, PeftConfig

peft_model_id = "ybelkada/blip2-opt-2.7b-football-captions-adapters"
config = PeftConfig.from_pretrained(peft_model_id)

model = Blip2ForConditionalGeneration.from_pretrained(config.
↪base_model_name_or_path, load_in_8bit=True, device_map="auto")
model = PeftModel.from_pretrained(model, peft_model_id)

processor = AutoProcessor.from_pretrained("Salesforce/blip2-opt-2.7b")
```

Let's check the results on our train dataset!

[22]:
```
!pip install matplotlib
```

```
huggingface/tokenizers: The current process just got forked, after parallelism
has already been used. Disabling parallelism to avoid deadlocks…
```

```
To disable this warning, you can either:
	- Avoid using `tokenizers` before the fork if possible
	- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true |
false)

Collecting matplotlib
  Downloading matplotlib-3.9.2-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.0-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.54.1-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (163 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.7-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.3 kB)
Requirement already satisfied: numpy>=1.23 in ./.conda/lib/python3.11/site-
packages (from matplotlib) (2.1.3)
Requirement already satisfied: packaging>=20.0 in ./.conda/lib/python3.11/site-
packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in ./.conda/lib/python3.11/site-
packages (from matplotlib) (11.0.0)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.2.0-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in
./.conda/lib/python3.11/site-packages (from matplotlib) (2.9.0)
Requirement already satisfied: six>=1.5 in ./.conda/lib/python3.11/site-packages
(from python-dateutil>=2.7->matplotlib) (1.16.0)
Downloading
matplotlib-3.9.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.3
MB)
                         8.3/8.3 MB
68.8 MB/s eta 0:00:00
Downloading
contourpy-1.3.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (323
kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading
fonttools-4.54.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.9
MB)
                         4.9/4.9 MB
91.3 MB/s eta 0:00:00
Downloading
kiwisolver-1.4.7-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.4
MB)
```

```
                                   1.4/1.4 MB
54.7 MB/s eta 0:00:00
Downloading pyparsing-3.2.0-py3-none-any.whl (106 kB)
Installing collected packages: pyparsing, kiwisolver, fonttools, cycler,
contourpy, matplotlib
Successfully installed contourpy-1.3.0 cycler-0.12.1 fonttools-4.54.1
kiwisolver-1.4.7 matplotlib-3.9.2 pyparsing-3.2.0
```

[25]:

```python
import torch
from matplotlib import pyplot as plt

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

fig = plt.figure(figsize=(18, 14))

# prepare image for the model
for i, example in enumerate(new_dataset[:1]):
  image = example["image"]
  inputs = processor(images=image, return_tensors="pt").to(device, torch.
 ↪float16)
  pixel_values = inputs.pixel_values

  generated_ids = model.generate(pixel_values=pixel_values, max_length=25)
  generated_caption = processor.batch_decode(generated_ids,␣
 ↪skip_special_tokens=True)[0]
  fig.add_subplot(2, 3, i+1)
  plt.imshow(image)
  plt.axis("off")
  plt.title(f"Generated caption: {generated_caption}")
```

Generated caption: The detected disease is Actinic Keratosis (AK), a precancerous condition characterized by scaly, crust