1. Installation

```
%%capture
import os, re
if "COLAB_" not in "".join(os.environ.keys()):
    !pip install unsloth
else:
    # Do this only in Colab notebooks! Otherwise use pip install
unsloth
    import torch; v = re.match(r"[0-9\.]{3,}",
str(torch.__version__)).group(0)
    xformers = "xformers==" + ("0.0.32.post2" if v == "2.8.0" else
"0.0.29.post3")
    !pip install --no-deps bitsandbytes accelerate {xformers} peft trl
triton cut_cross_entropy unsloth_zoo
    !pip install sentencepiece protobuf "datasets>=3.4.1,<4.0.0"
"huggingface_hub>=0.34.0" hf_transfer
    !pip install --no-deps unsloth
!pip install transformers==4.55.4
!pip install --no-deps trl==0.22.2
import torch; torch._dynamo.config.recompile_limit = 64;

%%capture
!pip install --no-deps --upgrade timm # Only for Gemma 3N

!pip install unsloth==2025.9.1 unsloth_zoo==2025.9.1
```

###2. Model Loading :

```
from unsloth import FastModel
import torch

fourbit_models = [
    # 4bit dynamic quants for superior accuracy and low memory use
    "unsloth/gemma-3n-E4B-it-unsloth-bnb-4bit",
    "unsloth/gemma-3n-E2B-it-unsloth-bnb-4bit",
    # Pretrained models
    "unsloth/gemma-3n-E4B-unsloth-bnb-4bit",
    "unsloth/gemma-3n-E2B-unsloth-bnb-4bit",

    # Other Gemma 3 quants
    "unsloth/gemma-3-1b-it-unsloth-bnb-4bit",
    "unsloth/gemma-3-4b-it-unsloth-bnb-4bit",
    "unsloth/gemma-3-12b-it-unsloth-bnb-4bit",
    "unsloth/gemma-3-27b-it-unsloth-bnb-4bit",
] # More models at https://huggingface.co/unsloth

model, tokenizer = FastModel.from_pretrained(
    model_name = "unsloth/gemma-3n-E4B-it", # Changed to 1b model
```

```
    dtype = None, # None for auto detection
    max_seq_length = 1024, # Choose any for long context!
    load_in_4bit = True,  # 4 bit quantization to reduce memory
    full_finetuning = False, # [NEW!] We have full finetuning now!
    # token = "hf_...", # use one if using gated models
)
```

```
🦥 Unsloth: Will patch your computer to enable 2x faster free
finetuning.
🦥 Unsloth Zoo will now patch everything to make training faster!
==((====))==  Unsloth 2025.9.1: Fast Gemma3N patching. Transformers:
4.55.4.
   \\   /|    Tesla T4. Num GPUs = 1. Max memory: 14.741 GB. Platform:
Linux.
O^O/ \_/ \    Torch: 2.8.0+cu126. CUDA: 7.5. CUDA Toolkit: 12.6.
Triton: 3.4.0
\        /    Bfloat16 = FALSE. FA [Xformers = 0.0.32.post2. FA2 =
False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which
are red colored!
Unsloth: Gemma3N does not support SDPA - switching to fast eager.
```


```
{"model_id":"5378466ec94c41f3aae8ed0467b5bd99","version_major":2,"vers
ion_minor":0}

{"model_id":"8ff817f914424efcaf8b1b42c1d2c744","version_major":2,"vers
ion_minor":0}

{"model_id":"960c49852257455f9984964ff0427438","version_major":2,"vers
ion_minor":0}

{"model_id":"2b3c1cb9b7c545daa93a4cd2f1f3bc4d","version_major":2,"vers
ion_minor":0}

{"model_id":"027371f7caaf4e6dab25aca50524e751","version_major":2,"vers
ion_minor":0}

{"model_id":"3320e11fa36e4dbb9791ccdcca57d1b0","version_major":2,"vers
ion_minor":0}

{"model_id":"fd2f5e68372941b1888762871e1660ad","version_major":2,"vers
ion_minor":0}

{"model_id":"d9a4ceed2b724bd2adba124d8c688d97","version_major":2,"vers
ion_minor":0}

{"model_id":"73eaee58c85a4fa68b1158019a44b53b","version_major":2,"vers
ion_minor":0}

{"model_id":"17ffd358542d4ed09ba8cb075297649c","version_major":2,"vers
ion_minor":0}
```

{"model_id":"00b9e1651d064ed4bf453c58f2f06d7d","version_major":2,"version_minor":0}

{"model_id":"af23c8675ac640f6925f443a7669cf73","version_major":2,"version_minor":0}

{"model_id":"de8ff2789da745b5ad4fd42f6a5b9f98","version_major":2,"version_minor":0}

{"model_id":"ce2323a30f4f40b6a94726f2dcf3686a","version_major":2,"version_minor":0}

We use Gemma 3N's recommended settings of `temperature = 1.0, top_p = 0.95, top_k = 64`

```
from transformers import TextStreamer
# Helper function for inference
def do_gemma_3n_inference(messages, max_new_tokens = 128):
    _ = model.generate(
        **tokenizer.apply_chat_template(
            messages,
            add_generation_prompt = True, # Must add for generation
            tokenize = True,
            return_dict = True,
            return_tensors = "pt",
        ).to("cuda"),
        max_new_tokens = max_new_tokens,
        temperature = 1.0, top_p = 0.95, top_k = 64,
        streamer = TextStreamer(tokenizer, skip_prompt = True),
    )
```

3. Check Gemma on CKD Data

The prediction is not accurate which means model is not trained on chronic kidney disease data

```
messages = [{
    "role" : "user",
    "content": [

        { "type": "text",  "text" : "Predict whether the patient has ckd or not if The age (years) is 48.0. The blood pressure (mm/Hg) is 80.0. The specific gravity is 1.02. The albumin is 1.0. The sugar is 0.0. The red blood cells is normal. The pus cells is normal. The pus cell clumps is notpresent. The bacteria is notpresent. The blood glucose random (mg/dL) is 121.0. The blood urea (mg/dL) is 36.0. The serum creatinine (mg/dL) is 1.2. The sodium (mEq/L) is 137.52875399361022. The potassium (mEq/L) is 4.62724358974359. The hemoglobin (g/dL) is 15.4. The packed cell volume is 44. The white blood cell count (cells/cumm) is 7800. The red blood cell count
```

```
(millions/cmm) is 5.2. The hypertension is yes. The diabetes mellitus
is yes. The coronary artery disease is no. The appetite is good. The
pedal edema is no. The anemia is no" }
    ]
}]
do_gemma_3n_inference(messages, max_new_tokens = 50)
```

```
Based on the provided data, **it is highly probable that the patient
does NOT have Chronic Kidney Disease (CKD) at this point.** Here's a
breakdown of why:

*   **Serum Creatinine:** A serum creatinine of 1
```

4. Finetuning Gemma 3N!

You can finetune the vision and text parts for now through selection - the audio part can also be finetuned - we're working to make it selectable as well!

We now add LoRA adapters so we only need to update a small amount of parameters!

```
model = FastModel.get_peft_model(
    model,
    finetune_vision_layers     = False, # Turn off for just text!
    finetune_language_layers   = True,  # Should leave on!
    finetune_attention_modules = True,  # Attention good for GRPO
    finetune_mlp_modules       = True,  # Should leave on always!

    r = 8,           # Larger = higher accuracy, but might overfit
    lora_alpha = 8,  # Recommended alpha == r at least
    lora_dropout = 0,
    bias = "none",
    random_state = 3407,
)
```

```
Unsloth: Making `model.base_model.model.model.language_model` require
gradients
```

```
from unsloth.chat_templates import get_chat_template
tokenizer = get_chat_template(
    tokenizer,
    chat_template = "gemma-3",
)

from datasets import load_dataset
dataset = load_dataset("csv",
data_files="/content/ckd_serialized_for_llm.csv")

# By default, this creates a DatasetDict with "train" split
print(dataset)
print(type(dataset))
```

```
DatasetDict({
    train: Dataset({
        features: ['instruction', 'input', 'output'],
        num_rows: 400
    })
})
<class 'datasets.dataset_dict.DatasetDict'>
```

We now use `standardize_data_formats` to try converting datasets to the correct format for finetuning purposes!

```
from unsloth.chat_templates import standardize_data_formats
# Pass the 'train' split of the dataset to the function
dataset['train'] = standardize_data_formats(dataset['train'])

dataset['train'][0]
```

```
{'instruction': 'Based on these features, predict whether the patient has chronic kidney disease or not.',
 'input': 'The age (years) is 48.0. The blood pressure (mm/Hg) is 80.0. The specific gravity is 1.02. The albumin is 1.0. The sugar is 0.0. The red blood cells is normal. The pus cells is normal. The pus cell clumps is notpresent. The bacteria is notpresent. The blood glucose random (mg/dL) is 121.0. The blood urea (mg/dL) is 36.0. The serum creatinine (mg/dL) is 1.2. The sodium (mEq/L) is 137.52875399361022. The potassium (mEq/L) is 4.62724358974359. The hemoglobin (g/dL) is 15.4. The packed cell volume is 44. The white blood cell count (cells/cumm) is 7800. The red blood cell count (millions/cmm) is 5.2. The hypertension is yes. The diabetes mellitus is yes. The coronary artery disease is no. The appetite is good. The pedal edema is no. The anemia is no.',
 'output': 'ckd'}
```

We now have to apply the chat template for `Gemma-3` onto the conversations, and save it to `text`. We remove the `<bos>` token using removeprefix(`'<bos>'`) since we're finetuning. The Processor will add this token before training and the model expects only one.

```
def formatting_prompts_func(examples):
   # Create conversations based on the available columns
   # Assuming instruction and input are user turns, and output is the model's response
   convos = []
   for instruction, user_input, output in zip(examples["instruction"], examples["input"], examples["output"]):
       messages = []
       # Add instruction if present
       if instruction:
           messages.append({"role": "user", "content": instruction})
       # Add input if present (assuming it's part of the user's turn
```

```
or a separate user turn)
        if user_input:
            # Decide how to incorporate input - here assuming it's part
of the user's turn
            if messages:
                messages[-1]["content"] += f"\n{user_input}"
            else:
                messages.append({"role": "user", "content":
user_input})
        # Add model's output
        if output:
            messages.append({"role": "model", "content": output})
        convos.append(messages)

    # Apply the chat template to the created conversations
    # Ensure add_generation_prompt is False for training data
    texts = [tokenizer.apply_chat_template(convo, tokenize = False,
add_generation_prompt = False).removeprefix('<bos>') for convo in
convos]

    return { "text" : texts, }

dataset = dataset.map(formatting_prompts_func, batched = True)
```


```
{"model_id":"8220a852d2cb45d2ab54a1852ec7e02c","version_major":2,"vers
ion_minor":0}
```


Let's see how the chat template did! Notice there is no `<bos>` token as the processor tokenizer will be adding one.

```
dataset['train'][18]
```

```
{'instruction': 'Based on these features, predict whether the patient
has chronic kidney disease or not.',
 'input': 'The age (years) is 60.0. The blood pressure (mm/Hg) is
100.0. The specific gravity is 1.025. The albumin is 0.0. The sugar is
3.0. The red blood cells is normal. The pus cells is normal. The pus
cell clumps is notpresent. The bacteria is notpresent. The blood
glucose random (mg/dL) is 263.0. The blood urea (mg/dL) is 27.0. The
serum creatinine (mg/dL) is 1.3. The sodium (mEq/L) is 135.0. The
potassium (mEq/L) is 4.3. The hemoglobin (g/dL) is 12.7. The packed
cell volume is 37. The white blood cell count (cells/cumm) is 11400.
The red blood cell count (millions/cmm) is 4.3. The hypertension is
yes. The diabetes mellitus is yes. The coronary artery disease is yes.
The appetite is good. The pedal edema is no. The anemia is no.',
 'output': 'ckd',
 'text': '<start_of_turn>user\nBased on these features, predict
whether the patient has chronic kidney disease or not.\nThe age
(years) is 60.0. The blood pressure (mm/Hg) is 100.0. The specific
gravity is 1.025. The albumin is 0.0. The sugar is 3.0. The red blood
```

```
cells is normal. The pus cells is normal. The pus cell clumps is
notpresent. The bacteria is notpresent. The blood glucose random
(mg/dL) is 263.0. The blood urea (mg/dL) is 27.0. The serum creatinine
(mg/dL) is 1.3. The sodium (mEq/L) is 135.0. The potassium (mEq/L) is
4.3. The hemoglobin (g/dL) is 12.7. The packed cell volume is 37. The
white blood cell count (cells/cumm) is 11400. The red blood cell count
(millions/cmm) is 4.3. The hypertension is yes. The diabetes mellitus
is yes. The coronary artery disease is yes. The appetite is good. The
pedal edema is no. The anemia is no.<end_of_turn>\
n<start_of_turn>model\nckd<end_of_turn>\n'}
```

###5. Train the model Now let's train our model. We do 60 steps to speed things up, but you can set `num_train_epochs=1` for a full run, and turn off `max_steps=None`.

```
from trl import SFTTrainer, SFTConfig
trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = dataset['train'], # Specify the 'train' split
    eval_dataset = None, # Can set up evaluation!
    args = SFTConfig(
        dataset_text_field = "text",
        per_device_train_batch_size = 1,
        gradient_accumulation_steps = 4, # Use GA to mimic batch size!
        warmup_steps = 5,
        # num_train_epochs = 1, # Set this for 1 full training run.
        max_steps = 60,
        learning_rate = 2e-4, # Reduce to 2e-5 for long training runs
        logging_steps = 1,
        optim = "adamw_8bit",
        weight_decay = 0.01,
        lr_scheduler_type = "linear",
        seed = 3407,
        report_to = "none", # Use this for WandB etc
    ),
)
```

We also use Unsloth's `train_on_completions` method to only train on the assistant outputs and ignore the loss on the user's inputs. This helps increase accuracy of finetunes!

```
from unsloth.chat_templates import train_on_responses_only
trainer = train_on_responses_only(
    trainer,
    instruction_part = "<start_of_turn>user\n",
    response_part = "<start_of_turn>model\n",
)
```

Let's verify masking the instruction part is done! Let's print the 100th row again. Notice how the sample only has a single `<bos>` as expected!

```
tokenizer.decode(trainer.train_dataset[10]["input_ids"])
```

```
{"type":"string"}
```

Now let's print the masked out example - you should see only the answer is present:

```
tokenizer.decode([tokenizer.pad_token_id if x == -100 else x for x in
trainer.train_dataset[10]["labels"]]).replace(tokenizer.pad_token, "
")
```

```
{"type":"string"}
```

```
# @title Show current memory stats
gpu_stats = torch.cuda.get_device_properties(0)
start_gpu_memory = round(torch.cuda.max_memory_reserved() / 1024 /
1024 / 1024, 3)
max_memory = round(gpu_stats.total_memory / 1024 / 1024 / 1024, 3)
print(f"GPU = {gpu_stats.name}. Max memory = {max_memory} GB.")
print(f"{start_gpu_memory} GB of memory reserved.")

trainer_stats = trainer.train()
```

```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs used = 1
   \\   /|    Num examples = 400 | Num Epochs = 1 | Total steps = 60
O^O/ \_/ \    Batch size per device = 1 | Gradient accumulation steps
= 4
\        /    Data Parallel GPUs = 1 | Total batch size (1 x 4 x 1) =
4
 "-____-"     Trainable parameters = 19,210,240 of 7,869,188,432
(0.24% trained)

<IPython.core.display.HTML object>
```

```
# @title Show final memory and time stats
used_memory = round(torch.cuda.max_memory_reserved() / 1024 / 1024 /
1024, 3)
used_memory_for_lora = round(used_memory - start_gpu_memory, 3)
used_percentage = round(used_memory / max_memory * 100, 3)
lora_percentage = round(used_memory_for_lora / max_memory * 100, 3)
print(f"{trainer_stats.metrics['train_runtime']} seconds used for
training.")
print(
    f"{round(trainer_stats.metrics['train_runtime']/60, 2)} minutes
used for training."
)
print(f"Peak reserved memory = {used_memory} GB.")
print(f"Peak reserved memory for training = {used_memory_for_lora}
```

```
GB.")
print(f"Peak reserved memory % of max memory = {used_percentage} %.")
print(f"Peak reserved memory for training % of max memory =
{lora_percentage} %.")
```

6. Inference

Let's run the model via Unsloth native inference! According to the `Gemma-3` team, the recommended settings for inference are `temperature = 1.0, top_p = 0.95, top_k = 64`

```
from unsloth.chat_templates import get_chat_template
tokenizer = get_chat_template(
    tokenizer,
    chat_template = "gemma-3",
)
messages = [{
    "role": "user",
    "content": [{
        "type" : "text",
        "text" : "Based on the patient description, predict if they
have CKD.Predict whether the patient has ckd or not if The age (years)
is 48.0. The blood pressure (mm/Hg) is 80.0. The specific gravity is
1.02. The albumin is 1.0. The sugar is 0.0. The red blood cells is
normal. The pus cells is normal. The pus cell clumps is notpresent.
The bacteria is notpresent. The blood glucose random (mg/dL) is 121.0.
The blood urea (mg/dL) is 36.0. The serum creatinine (mg/dL) is 1.2.
The sodium (mEq/L) is 137.52875399361022. The potassium (mEq/L) is
4.62724358974359. The hemoglobin (g/dL) is 15.4. The packed cell
volume is 44. The white blood cell count (cells/cumm) is 7800. The red
blood cell count (millions/cmm) is 5.2. The hypertension is yes. The
diabetes mellitus is yes. The coronary artery disease is no. The
appetite is good. The pedal edema is no. The anemia is no",
        "input": "Predict whether the patient has ckd or not if The
age (years) is 48.0. The blood pressure (mm/Hg) is 80.0. The specific
gravity is 1.02. The albumin is 1.0. The sugar is 0.0. The red blood
cells is normal. The pus cells is normal. The pus cell clumps is
notpresent. The bacteria is notpresent. The blood glucose random
(mg/dL) is 121.0. The blood urea (mg/dL) is 36.0. The serum creatinine
(mg/dL) is 1.2. The sodium (mEq/L) is 137.52875399361022. The
potassium (mEq/L) is 4.62724358974359. The hemoglobin (g/dL) is 15.4.
The packed cell volume is 44. The white blood cell count (cells/cumm)
is 7800. The red blood cell count (millions/cmm) is 5.2. The
hypertension is yes. The diabetes mellitus is yes. The coronary artery
disease is no. The appetite is good. The pedal edema is no. The anemia
is no",
    }]
```

```
}]
inputs = tokenizer.apply_chat_template(
    messages,
    add_generation_prompt = True, # Must add for generation
    return_tensors = "pt",
    tokenize = True,
    return_dict = True,
).to("cuda")
outputs = model.generate(
    **inputs,
    max_new_tokens = 128, # Increase for longer outputs!
    # Recommended Gemma-3 settings!
    temperature = 1.0, top_p = 0.95, top_k = 64,
)
tokenizer.batch_decode(outputs)
```

```
["<bos><start_of_turn>user\nBased on the patient description, predict
if they have CKD.Predict whether the patient has ckd or not if The age
(years) is 48.0. The blood pressure (mm/Hg) is 80.0. The specific
gravity is 1.02. The albumin is 1.0. The sugar is 0.0. The red blood
cells is normal. The pus cells is normal. The pus cell clumps is
notpresent. The bacteria is notpresent. The blood glucose random
(mg/dL) is 121.0. The blood urea (mg/dL) is 36.0. The serum creatinine
(mg/dL) is 1.2. The sodium (mEq/L) is 137.52875399361022. The
potassium (mEq/L) is 4.62724358974359. The hemoglobin (g/dL) is 15.4.
The packed cell volume is 44. The white blood cell count (cells/cumm)
is 7800. The red blood cell count (millions/cmm) is 5.2. The
hypertension is yes. The diabetes mellitus is yes. The coronary artery
disease is no. The appetite is good. The pedal edema is no. The anemia
is no<end_of_turn>\n<start_of_turn>model\nBased on the provided
information, **it is highly likely that the patient has Chronic Kidney
Disease (CKD)**. Here's why:\n\n*   **Elevated Serum Creatinine:** A
serum creatinine level of 1.2 mg/dL is above the normal range and
indicates impaired kidney function.\n*   **Hypertension:** The patient
has hypertension (high blood pressure), which is a major risk factor
for kidney disease and can also be a consequence of it.\n*
**Diabetes Mellitus:** The patient has diabetes mellitus, another
major risk factor for kidney disease.\n*   **Blood Pressure:** The
blood pressure is"]
```

You can also use a `TextStreamer` for continuous inference - so you can see the generation token by token, instead of waiting the whole time!

```
messages = [{
    "role": "user",
    "content": [{"type" : "text", "text" : "Predict whether the
patient has ckd or not wnd why just mention critical features if The
age (years) is 48.0. The blood pressure (mm/Hg) is 80.0. The specific
gravity is 1.02. The albumin is 1.0. The sugar is 0.0. The red blood
```

```
cells is normal. The pus cells is normal. The pus cell clumps is
notpresent. The bacteria is notpresent. The blood glucose random
(mg/dL) is 121.0. The blood urea (mg/dL) is 36.0. The serum creatinine
(mg/dL) is 1.2. The sodium (mEq/L) is 137.52875399361022. The
potassium (mEq/L) is 4.62724358974359. The hemoglobin (g/dL) is 15.4.
The packed cell volume is 44. The white blood cell count (cells/cumm)
is 7800. The red blood cell count (millions/cmm) is 5.2. The
hypertension is yes. The diabetes mellitus is yes. The coronary artery
disease is no. The appetite is good. The pedal edema is no. The anemia
is no",}]
}]
inputs = tokenizer.apply_chat_template(
    messages,
    add_generation_prompt = True, # Must add for generation
    return_tensors = "pt",
    tokenize = True,
    return_dict = True,
).to("cuda")

from transformers import TextStreamer
_ = model.generate(
    **inputs,
    max_new_tokens = 256, # Increase for longer outputs!
    # Recommended Gemma-3 settings!
    temperature = 1.0, top_p = 0.95, top_k = 64,
    streamer = TextStreamer(tokenizer, skip_prompt = True),
)
```

Critical Features Indicating CKD:

*   **Hypertension (yes):** High blood pressure is a major risk factor for CKD and can also be a consequence of it.
*   **Diabetes Mellitus (yes):** Diabetes is a leading cause of CKD.
*   **Serum Creatinine (1.2 mg/dL):** This value is slightly elevated, suggesting impaired kidney function. While not drastically high, it's a key indicator.
*   **Blood Pressure (80.0 mm/Hg):** Significantly low blood pressure, especially in the context of other findings, can indicate kidney dysfunction or other serious issues.
*   **Age (48.0 years):** While not definitive on its own, this age group is more susceptible to developing kidney disease.

Reasoning:

The combination of hypertension, diabetes, and an elevated serum creatinine strongly suggests chronic kidney disease (CKD). The low blood pressure is also concerning and warrants further investigation.
<end_of_turn>

7. Saving, loading finetuned models

To save the final model as LoRA adapters, either use Huggingface's `push_to_hub` for an online save or `save_pretrained` for a local save.

[NOTE] This ONLY saves the LoRA adapters, and not the full model. To save to 16bit or GGUF, scroll down!

```
model.save_pretrained("gemma-3n_ckd_finetuned")  # Local saving
tokenizer.save_pretrained("gemma-3n_ckd_finetuned") # Local saving
```

```
['gemma-3n_ckd_finetuned/processor_config.json']
```