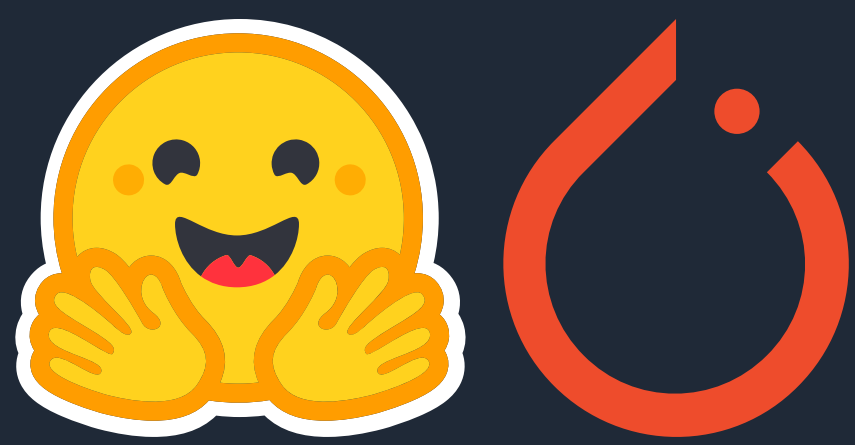
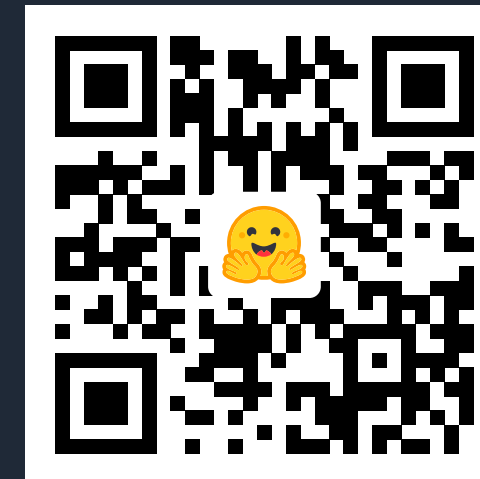




TorchTPU × Hugging Face

Native TPU Acceleration Across the HF Ecosystem

Alvaro Moran Jingya Huang Sayak Paul • Hugging Face



huggingface.co

Same PyTorch code, now on TPU

Until now, TPUs meant leaving your tools: porting PyTorch code to `torch_xla` or rewriting it in JAX. **torch_tpu** is a native PyTorch backend: TPU is just another device.

We made the Hugging Face stack work on it, so the **transformers** and **diffusers** code you already have runs on TPU almost unchanged.

```
from transformers import AutoModelForCausalLM, AutoTokenizer

model_id = "Qwen/Qwen3.5-9B"
tokenizer = AutoTokenizer.from_pretrained(model_id)
- model = AutoModelForCausalLM.from_pretrained(model_id, device_map="cuda")
+ model = AutoModelForCausalLM.from_pretrained(model_id, device_map="tpu")
- inputs = tokenizer("The future of AI is", return_tensors="pt").to("cuda")
+ inputs = tokenizer("The future of AI is", return_tensors="pt").to("tpu")
out = model.generate(**inputs, max_new_tokens=64)
print(tokenizer.decode(out[0], skip_special_tokens=True))
```

1 Transformers on TPU

Transformers now treats TPU as a first-class backend: device detection, tensor-parallel plans, kernels and automatic compilation. Running on TPU also surfaced bugs we fixed upstream, such as StaticCache and Qwen3 TP plans. The model test suite now runs on `torch_tpu`, and results feed a CI dashboard next to NVIDIA and AMD. Every backend gap we hit becomes a standalone repro filed with `torch_tpu`.

34

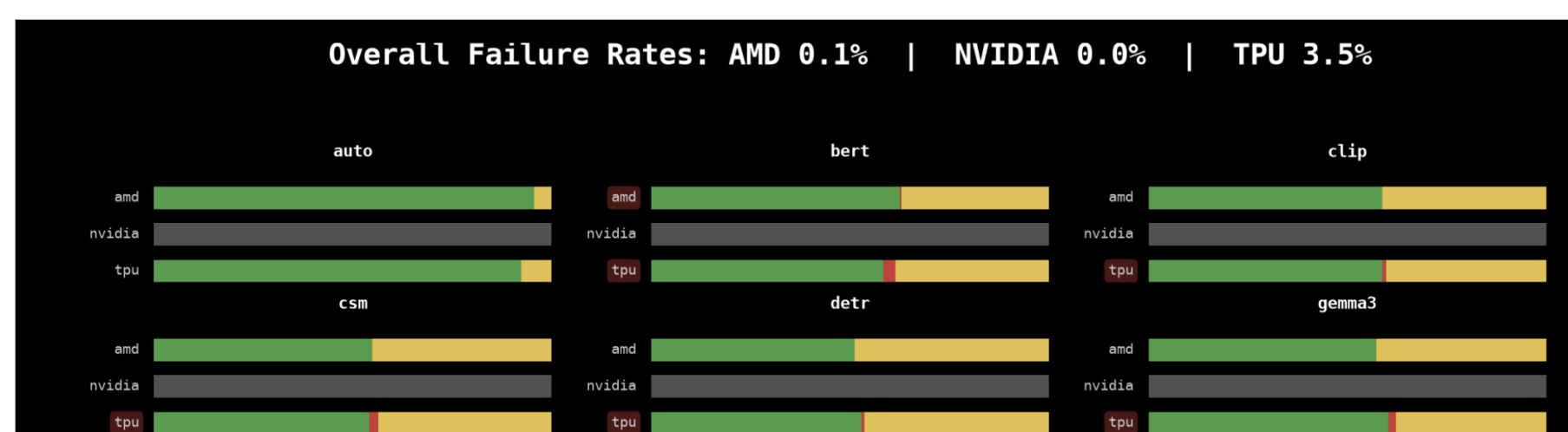
torch_tpu issues reported

16

fixed upstream

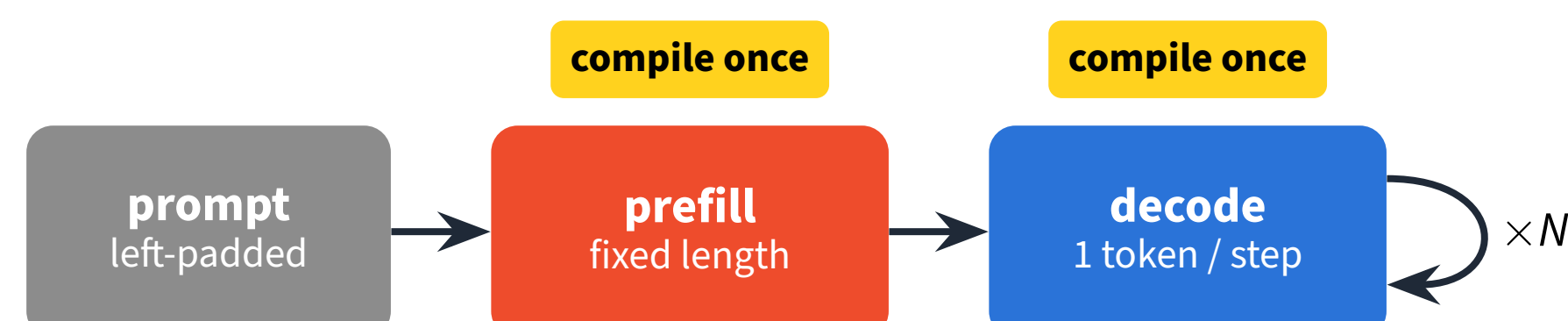
96.5%

transformers tests
passing on TPU



2 Generation tuned for TPU

Every new tensor shape costs a compilation on TPU, so a naive generation loop keeps recompiling. We left-pad prompts to a fixed length and decode into a pre-allocated StaticCache: prefill and decode each **compile once**, and every later step reuses the graph. Compiled graphs persist in a tiered cache (memory, local disk, shared Hugging Face bucket), so later runs and other VMs skip compilation. Tensor and expert parallelism spread large and MoE models across all chips.



200

most popular Hub text-generation
models that fit on a TPU v6e-4

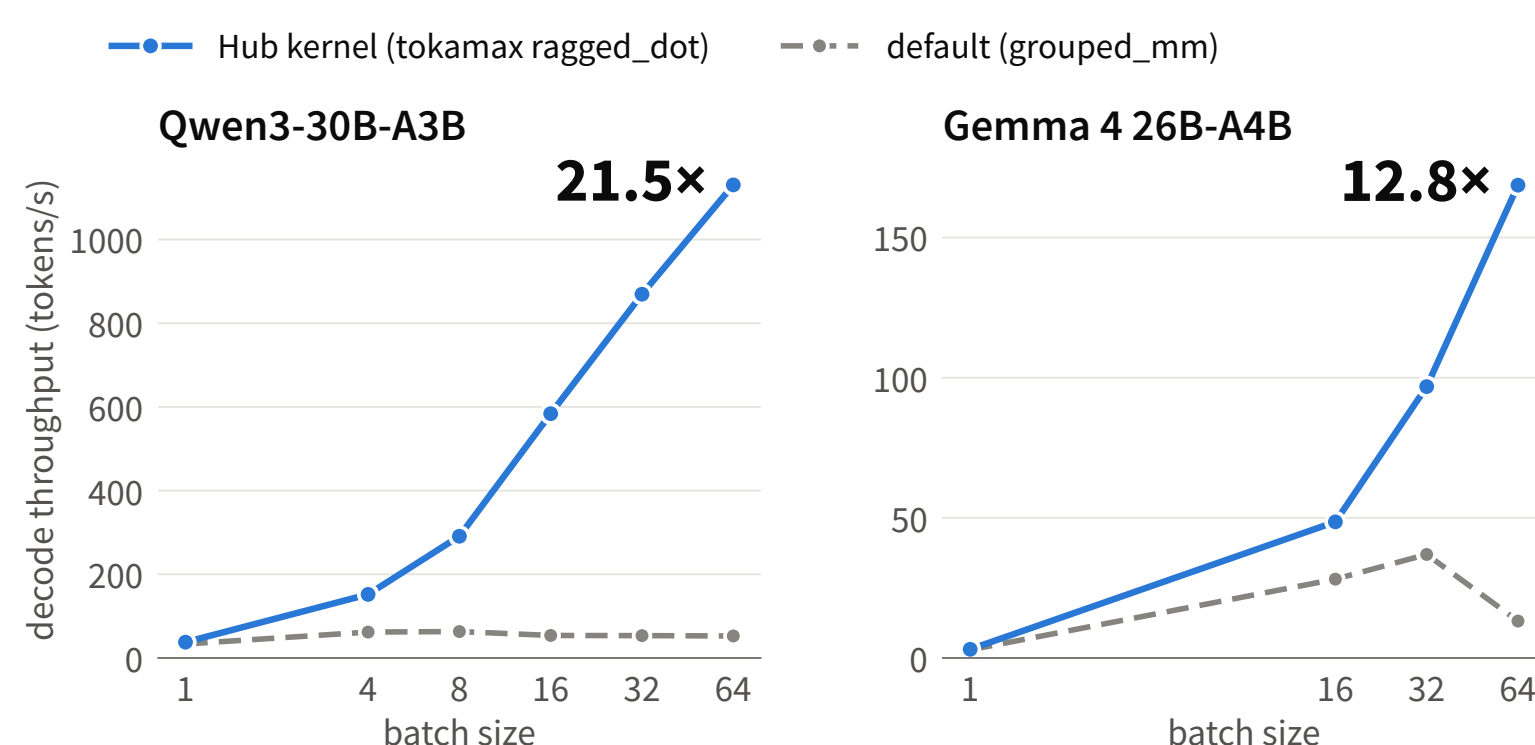
100%

load and generate coherent text on TPU

3 Hub kernels on TPU

The kernels library now builds and loads TPU kernels from the Hub. As a test we published tokamax's ragged-dot MoE kernel as a Hub kernel; transformers picks it up through `experts_implementation`, with no changes to model code. On a TPU v6e-8 at batch 64, decode throughput goes up **21.5×** for Qwen3-30B-A3B and **12.8×** for Gemma 4 26B-A4B, with the same kernel.

```
repo_id = "tengomucho/ragged-dot-tpu"
kernel = get_kernel(repo_id, version=1, trust_remote_code=[repo_id])
ALL_EXPERTS_FUNCTIONS.register(
    "tokamax_ragged_dot", kernel.experts_forward)
model = AutoModelForCausalLM.from_pretrained(
    "Qwen/Qwen3-30B-A3B-Instruct-2507",
    experts_implementation="tokamax_ragged_dot")
```



TPU v6e-8, TP=8 · greedy decode, prefill 256 · mean of timed runs

4 Diffusers on TPU

TPU is a native backend in diffusers: pipelines run on it in eager mode or with `torch.compile`, and the code is the same as on CUDA except for the device. Tensor parallelism splits large image and video pipelines across chips, and five model families support it so far (FLUX.1, FLUX.2, Qwen-Image, Qwen-Image-2.1 and MiniMax-H3). Adding another takes only a `tp_plan`. Every image and video frame on this panel was generated on TPU.

```
from diffusers import DiffusionPipeline
import torch

pipe = DiffusionPipeline.from_pretrained(..., torch_dtype=torch.bfloat16)
- pipe.to("cuda")
+ pipe.to("tpu")

pipe(...)
```

Qwen-Image-2.1

TPU v6e-4, 1024², 40 steps, bf16, batch 1, TP=4

Mode	s / image	ms / step
Eager	8.04	201
torch.compile	3.47 (2.3×)	87



MiniMax-H3

TPU v6e-4, text to video + audio, 1344×768, 124 frames (5.2 s), 49 steps, bf16, batch 1, TP=4

Mode	s / video	s / step
Eager	262.2	5.35
torch.compile	187.6 (1.4×)	3.83

Status & what's next

Working today

- ✓ transformers on `torch_tpu`, with tensor and expert parallelism
- ✓ Model test suite on TPU (dashboard in progress)
- ✓ TPU kernels loaded from the Hub
- ✓ diffusers pipelines with tensor parallelism

Next

- ➔ TPU in the official transformers CI
- ➔ Day-0 TPU support for new models and pipelines
- ➔ More TPU kernels on the Hub
- ➔ Training and fine-tuning with TRL

