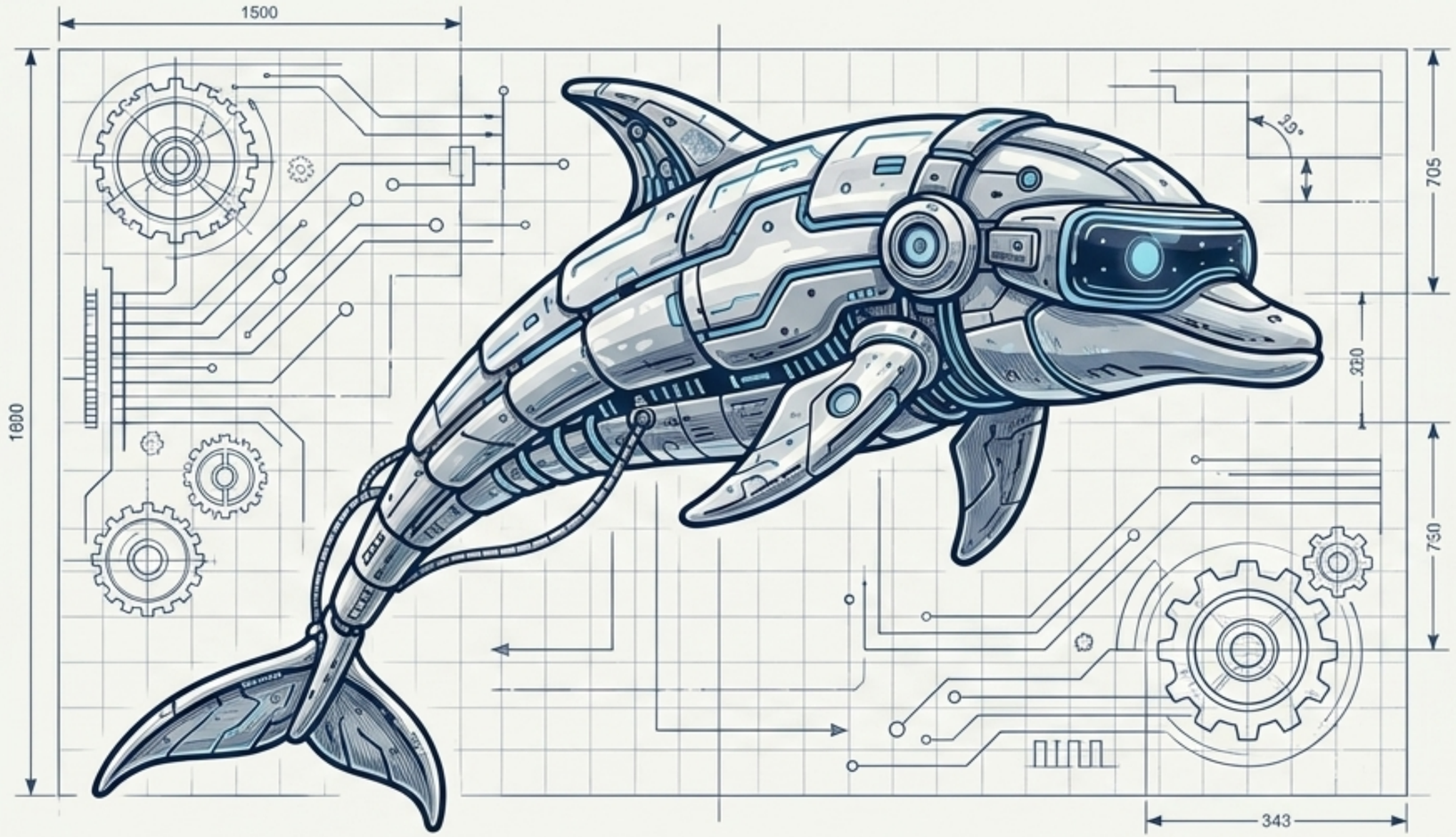
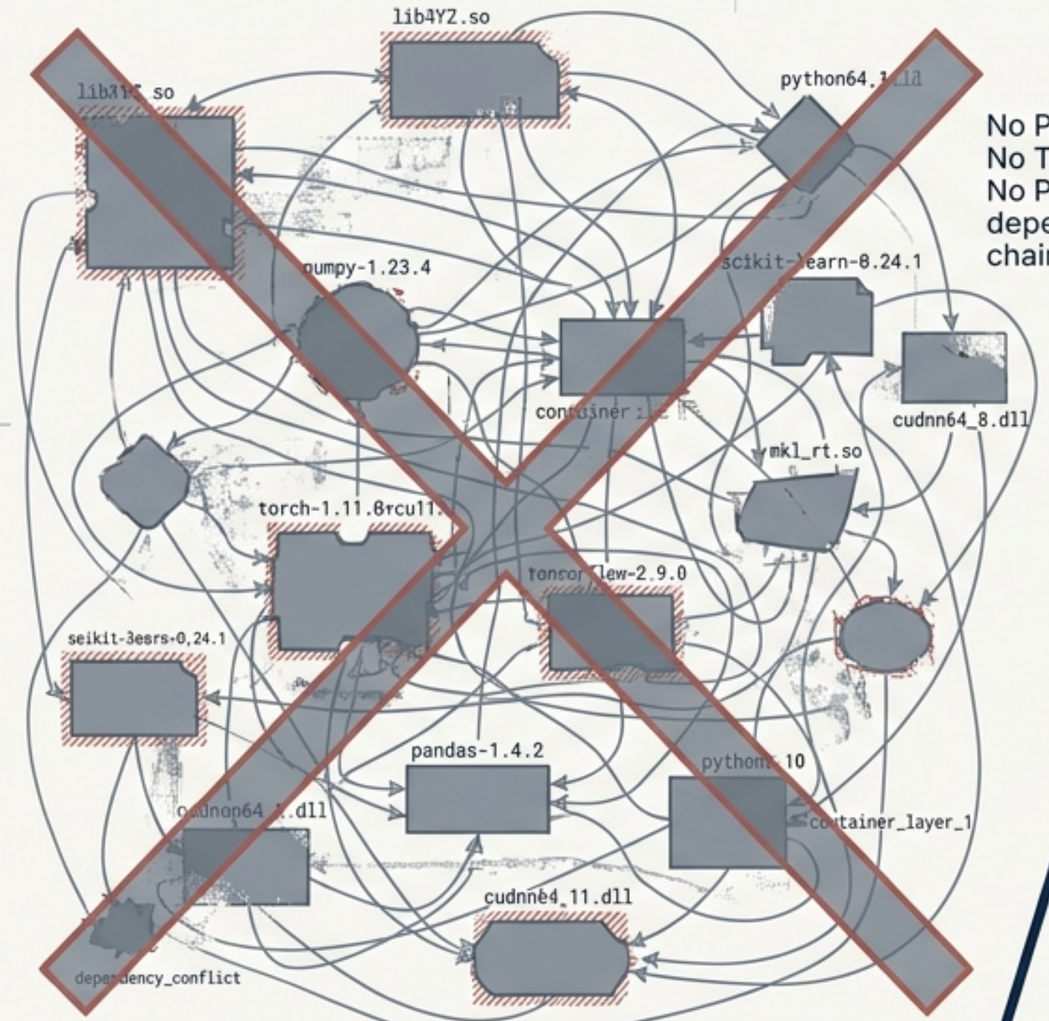


Dolphin: Pure Native Power for Transformer Models

The complete Object Pascal framework for building GPT-style models from scratch.

[Delphi 12+] [CUDA Accelerated] [MIT License]

Bypassing the bloated ML stack for pure, compiled execution.

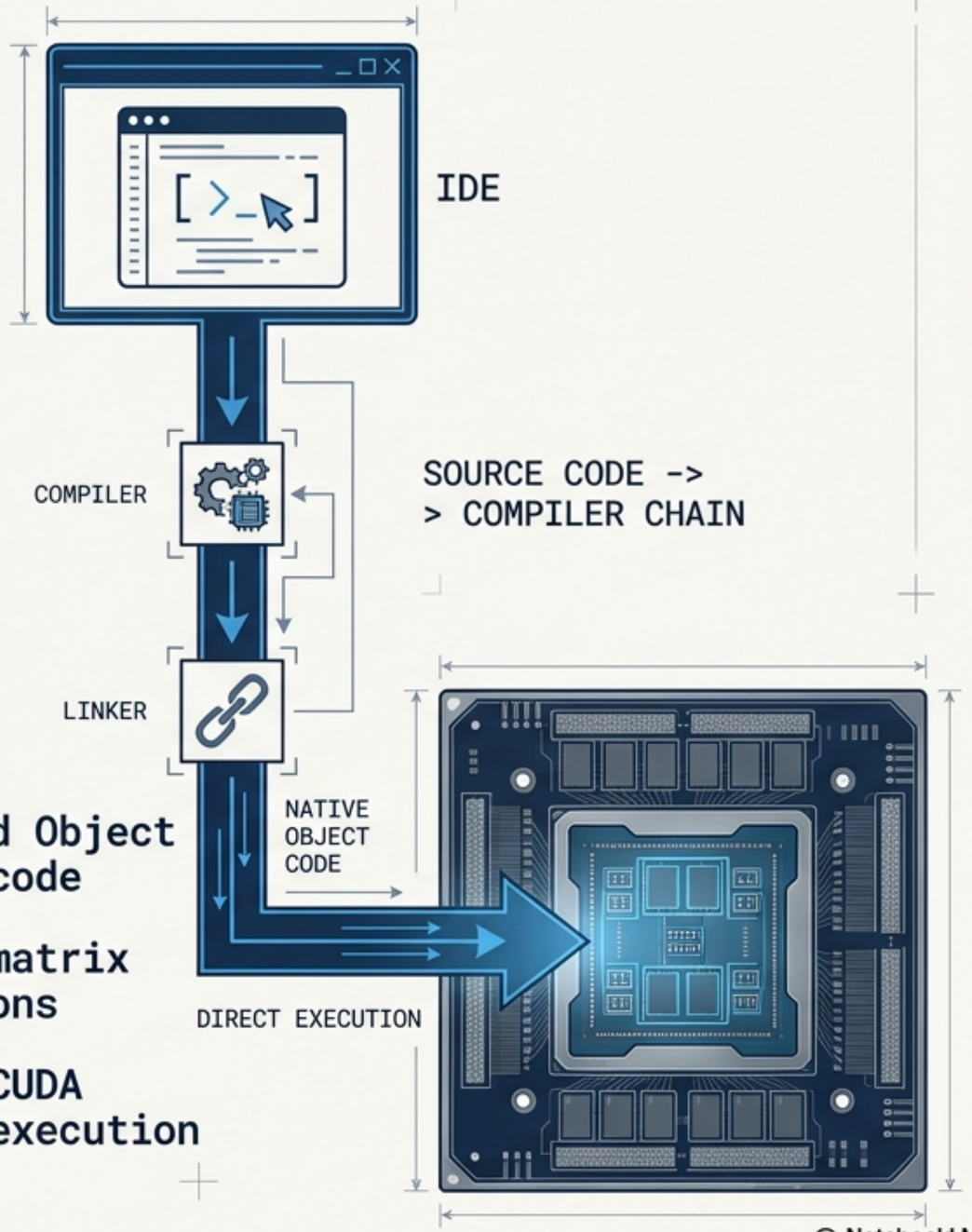


No PyTorch.
No TensorFlow.
No Python dependency chains.

> Compiled Object Pascal code

> Direct matrix operations

> Native CUDA kernel execution



A Complete Native Ecosystem for Model Engineering

GPU Acceleration

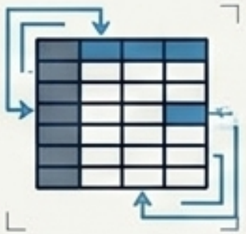
NVIDIA CUDA with hybrid precision (FP64 CPU, FP32 GPU), automatic dispatch, weight caching, and pooled memory management.

Full Training Pipeline

AdamW optimizer, gradient accumulation, micro-batching, early stopping, and GPU auto batch-size detection.

BPE Tokenization

Custom Byte Pair Encoding tokenizer with SQLite-backed vocabulary and Hugging Face compatible export.



TDataFrame

Pandas-like columnar data structure for structured ML — CSV loading, normalization, shuffling, direct GPU batching.

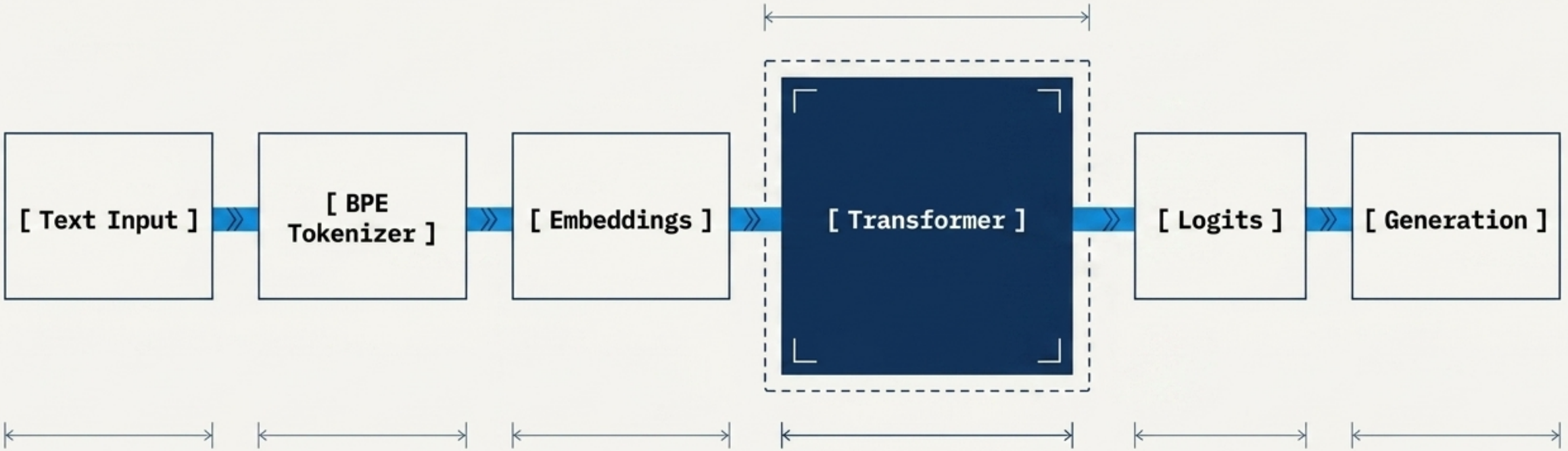
Interoperability

SafeTensors format, Hugging Face compatible exports, and JSON-based configuration.

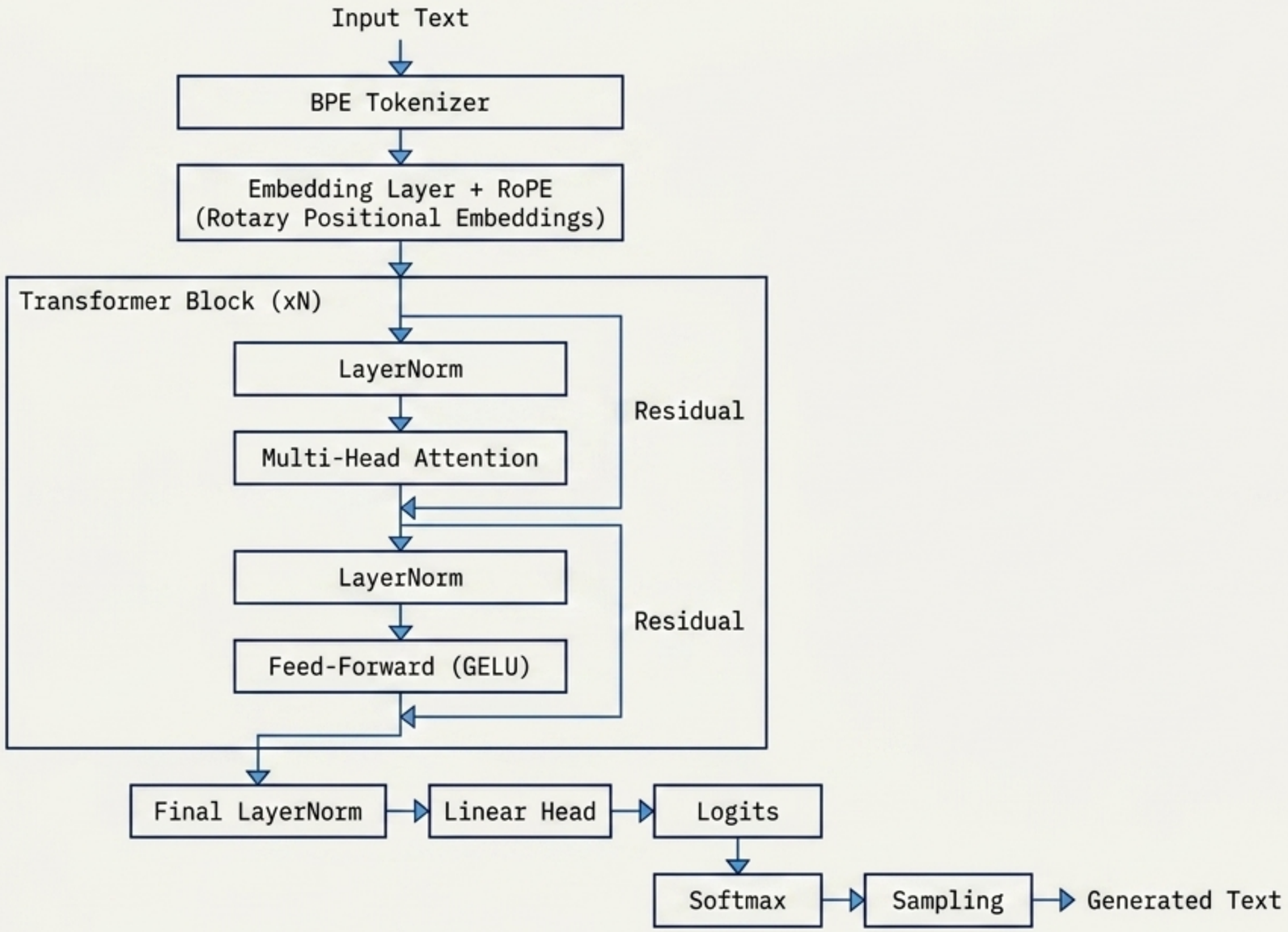
Advanced Generation

Greedy, temperature, top-k, and top-p (nucleus) sampling strategies for text generation.

The Macro Pipeline: End-to-End Native Execution

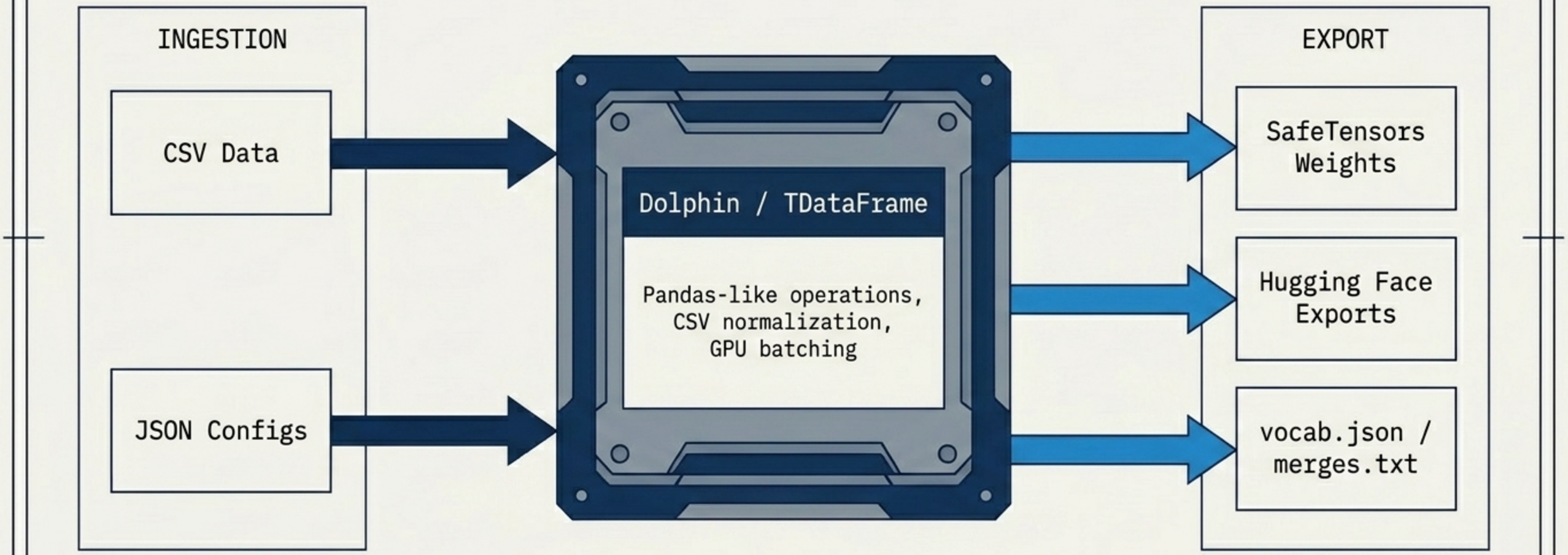


The Micro Architecture: Inside the Pascal Transformer Block



Native Integration with Industry Standard Formats

Ecosystem I/O Diagram



Spin up immediately via CLI, or integrate deeply via API.

CLI QUICK START

1. Open in Delphi:
   Open demos/trainer/pascalgpt.trainer.dpr
   (Delphi 12+, Win64 Release).

2. Prepare Config:
   Use PresetTraining/ or custom config.json.

3. Train:

```
> pascalgpt.trainer.exe -c config.json
dataset.txt
```

PASCAL API

```
// Build & Initialize
Model := TGPTModel.Create(Config);
Model.InitializeWeights;

// Train with GPU
Trainer := TGPTrainer.Create(Model, Dataset,
  Optimizer);
Trainer.UseGPU := True;
Trainer.Train(Epochs);

// Generate Text
Tokens := Model.Generate(PromptTokens, MaxTokens);
Text := Tokenizer.Decode(Tokens);
```

Training Objectives: A mode for every ML use-case.

User Goal	Training Mode	Output Artifacts
End-to-end model creation	FullModel	.safetensors, config.json, vocab
Transfer learning (Start from pretrained)	FineTune	.safetensors, config.json, vocab
Train semantic search (Contrastive learning)	SentenceEmbeddings	.safetensors, config.json, vocab
Extract raw token vectors	EmbeddingsOnly	embeddings.npy, vocab
Build external vocabularies	TokenizerOnly	vocab.json, merges.txt

The Knowledge Map: Navigate the Dolphin Ecosystem

Dolphin Docs

For evaluating the framework

- **Introduction:** Project vision, core capabilities, platform support.
- **How It Works:** End-to-end walkthrough of the full native pipeline.

For integrating and training

- **Trainer Manual:** Command-line usage, build instructions, workflow.
- **API Reference:** Full unit descriptions and configuration options.

For extending the engine

- **Wiki:** Deep technical reference, internals, advanced topics.
- **CUDA Reference:** GPU kernels, memory management, performance tuning.

> Access the repo on GitHub