Differentiable (binned) Likelihoods with JAX

Peter Fackeldey, Benjamin Fischer, **Felix Zinn**

PyHEP.dev 2026 – Nikhef, Amsterdam

Overview

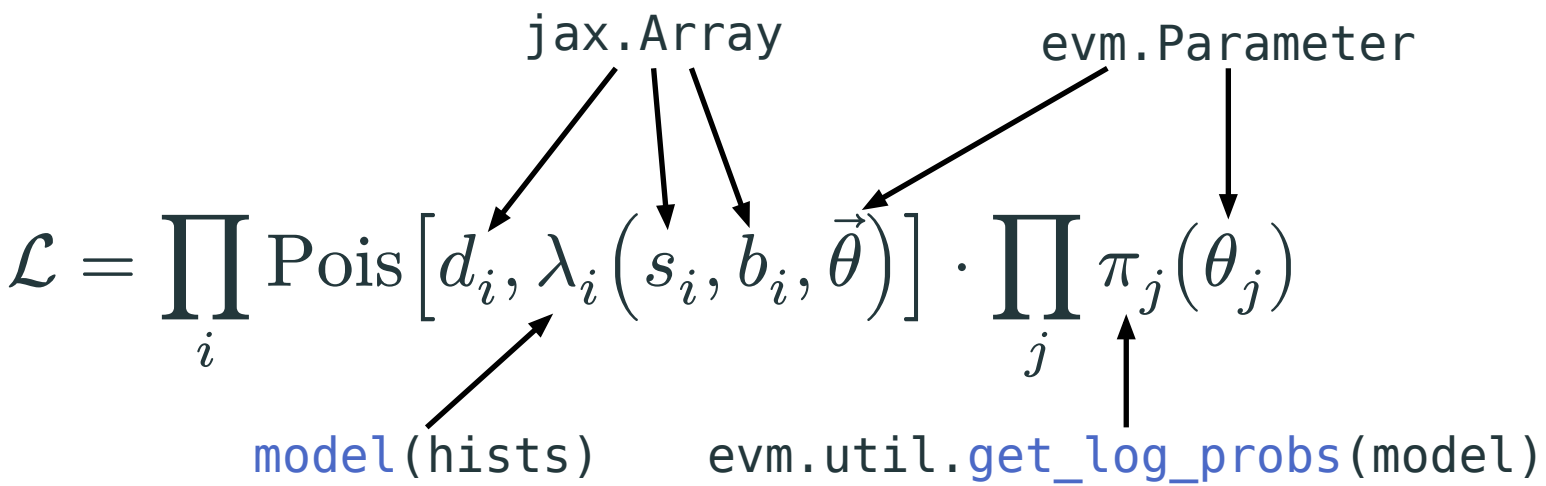
What is evermore

- provides tools for statistical model & likelihood building
- based on JAX and Flax
- integrates nicely with JAX ecosystem (e.g. Optax, Optimistix, Orbax, ...)

Key Concept

- everything in evermore is a JAX PyTree
  → compatible with JAX composable transformations
  - `jax.grad`: gradients through likelihood fits (neos)
  - `jax.jit`, `jax.vmap`: GPU acceleration, vectorized fitting

Likelihood

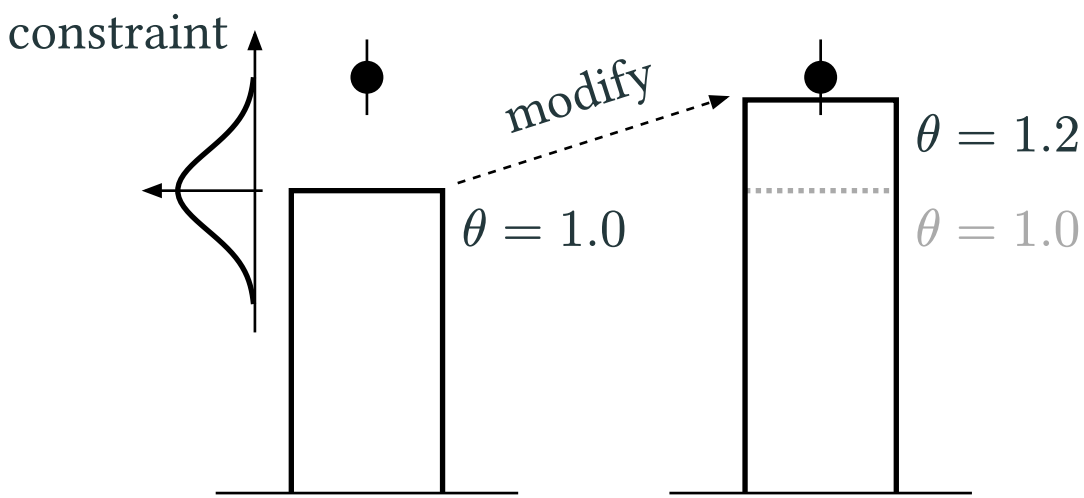

jax.Array
evm.Parameter
$\mathcal{L} = \prod_i \text{Pois}\left[d_i, \lambda_i\left(s_i, b_i, \vec{\theta}\right)\right] \cdot \prod_j \pi_j\left(\theta_j\right)$
model(hists)
evm.util.get_log_probs(model)

Building Blocks

Parameters

Effects

Modifiers



- bin content is *modified* by a *parameter* through an *effect*
- parameter may have a *constraint* associated with it

Models as PyTrees

→ represent statistical models as PyTrees (using `flax.nnx.Module`)

Python

```python
class Model(nnx.Module):
    def __init__(self):
        self.mu = evm.Parameter(name="mu", value=1.0)
        self.lumi = evm.NormalParameter(name="lumi", value=0.0)

    def __call__(self, hists: dict[str, Array]) -> Array:
        mu_mod = self.mu.scale()
        lumi_mod = self.lumi.scale_log_asymmetric(up=1.1, down=0.9)
        sig_mod = evm.modifier.Compose(mu_mod, lumi_mod)
        return sig_mod(hists["signal"]) + lumi_mod(hists["bkg"])

model = Model()
hists = {"signal": jnp.array([3]), "bkg": jnp.array([10])}
print(f"Expectation: {model(hists)}")  # -> Expectation: [13.]
```

Likelihood Definition

$$\log \mathcal{L} = \sum_{\text{bins}} \text{Poisson}\left(d_i, \lambda_i\left(s_i, b_i, \vec{\theta}\right)\right) + \sum_j \pi_j\left(\theta_j\right)$$

```python
def loss(dynamic, static, graphdef, hists, observation):
    model = nnx.merge(graphdef, dynamic, static)
    expectation = model(hists)

    # Poisson NLL
    log_likelihood = evm.pdf.PoissonContinuous(lamb=expectation).log_prob(observation).sum()

    # parameter contraints
    constraints = evm.loss.get_lob_probs(model)
    log_likelihood += evm.util.sum_over_leaves(constraints)

    return -jnp.sum(log_likelihood)
```

Putting it together (Maximum Likelihood Fit)

Python

```python
# model and loss defined as in previous slides
graphdef, dynamic, static = nnx.split(model, evm.filter.is_dynamic_parameter, ...)

tx = optax.adam(learning_rate=0.01)
optimizer = nnx.Optimizer(dynamic, tx, wrt=evm.filter.is_dynamic_parameter)

@nnx.jit
def make_step(optimizer, dynamic):
    grads = nnx.grad(loss)(dynamic, static, graphdef, hists, observation)
    optimizer.update(dynamic, grads)
    return dynamic

for step in range(steps):
    dynamic = make_step(optimizer, dynamic)
fitted_model = nnx.merge(graphdef, dynamic, static)
```

see everwillow contribution for more statistical methods

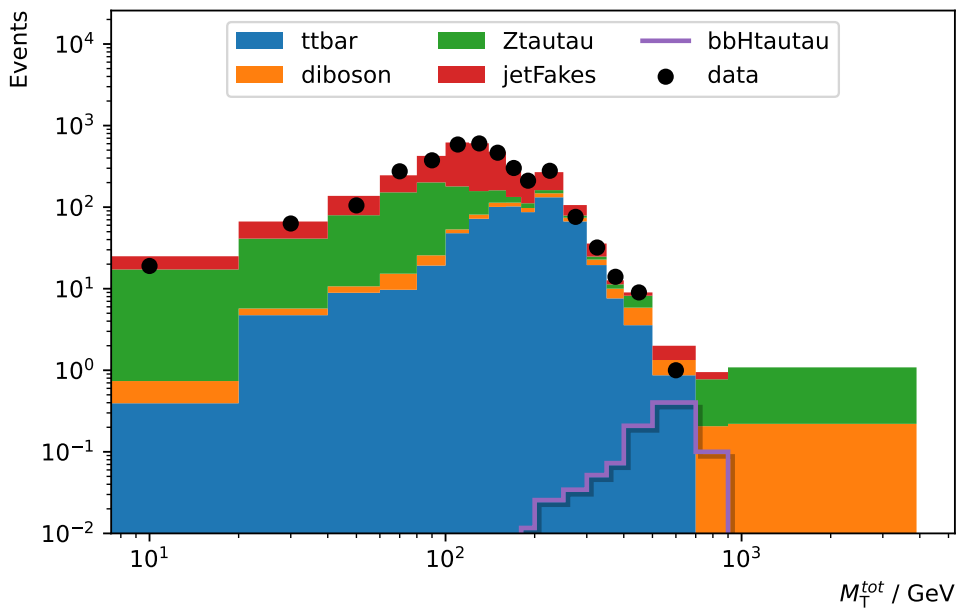
Likelihood Scan

- define fit function for fixed value of Parameter of Interest (μ)
- vectorize over POI values using `vmap`

```python
def fixed_mu_fit(mu): ...

mu_values = jnp.linspace(0, 10, 101)
# vectorize over mu_values
likelihood_scan = nnx.vmap(fixed_mu_fit, in_axes=(0,))(mu_values)
```

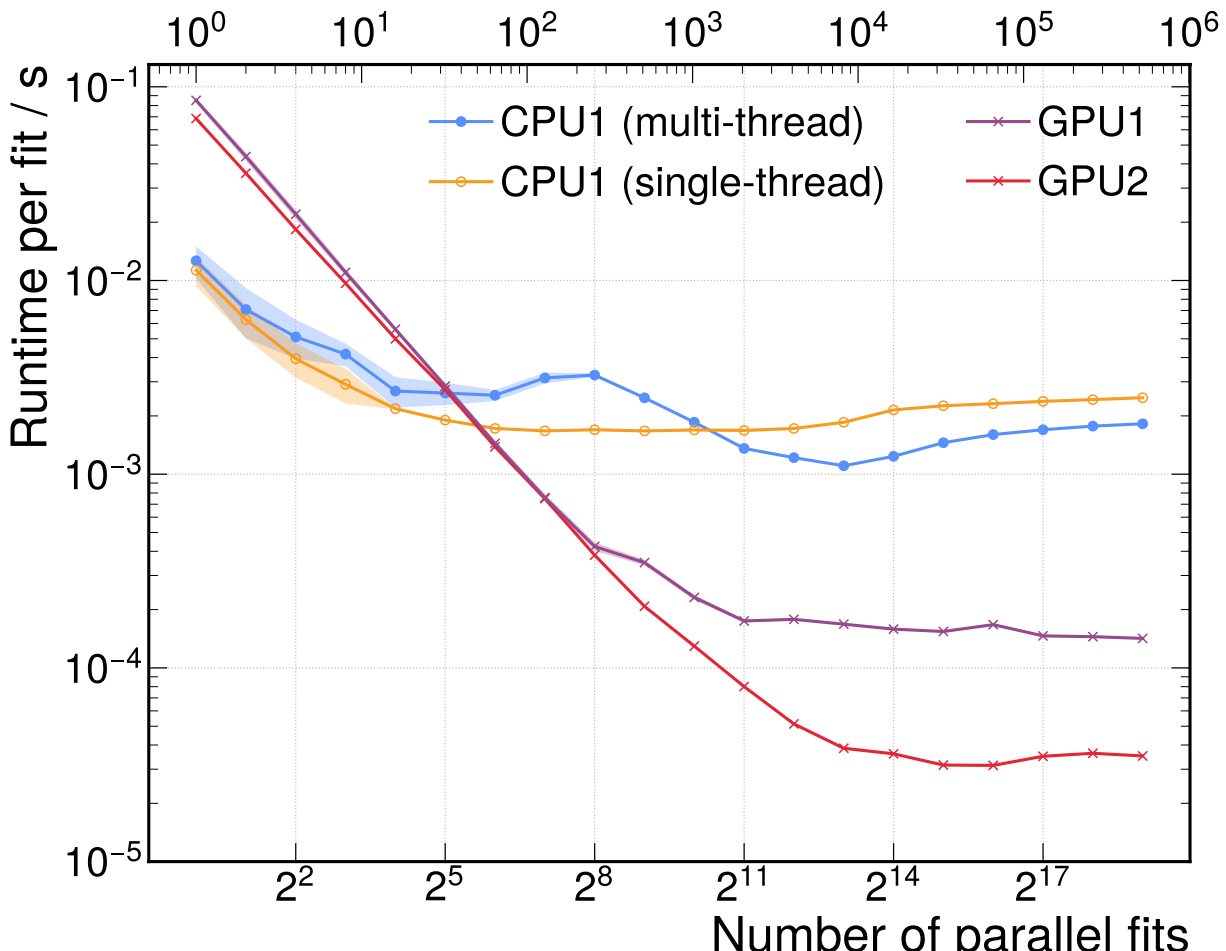
Performance



- simplified toy analysis taken from CMS Combine tutorial
- 1 signal process, 4 background processes
- 10 lnN uncertainties, 5 shape uncertainties, Monte-Carlo uncertainties (Barlow-Beeston)
- signal strength modifier r for likelihood profile
- fixed number of optimization steps (1000)

Performance

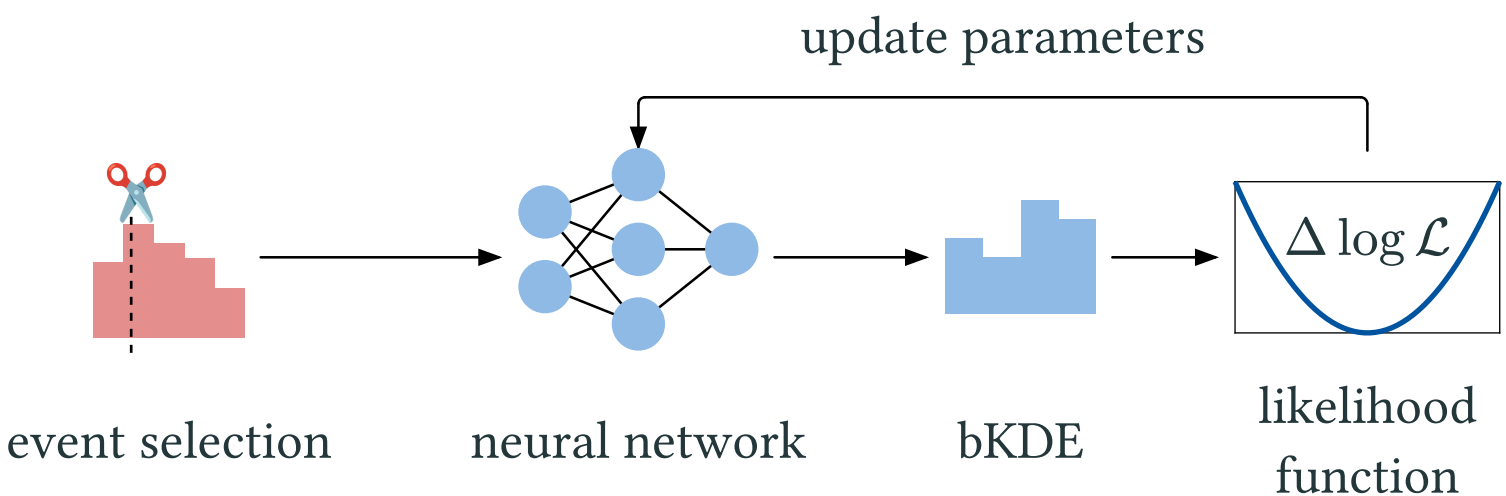
Model 1	Model 2
Ryzen 5950X	Epyc 7352
GTX 1080	RTX 6000 Ada
147 GFlops	1534 GFlops



single-thread using environment variable `NPROC=1`

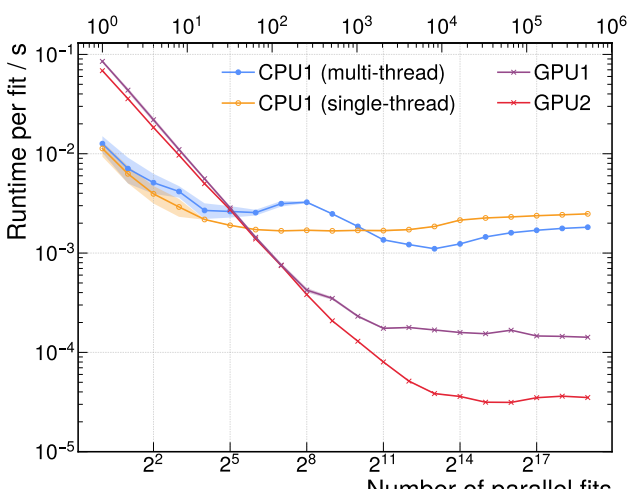
Gradients

- `jax.grad` allows to compute gradients through likelihood fits
- end-to-end differentiable workflow possible
- applied in CMS Run3 analysis (see my CHEP 2026 contribution, WIP)



Summary

- statistical models and likelihood functions with JAX
- key goals:
  - performance
  - differentiability
  - neural-network-like model api
  - seamless integration into JAX ecosystem



evermore

evermore

docs passing | CI passing | pypi v0.4.1 | python 3.11 | 3.12 | 3.13 | conda-forge v0.3.5 | license BSD 3-Clause

Differentiable (binned) likelihoods in JAX.

Installation

```
python -m pip install evermore
```

From source:

```
git clone https://github.com/pfackeldey/evermore
cd evermore
python -m pip install .
```

- GitHub
- Documentation