1. Overview: From Paper to Data and Code

This report explains how the methodology in *Flexible Gravitational-Wave Parameter Estimation with Transformers* is realized in practice, focusing on:

- The **datasets** (simulated and real),
- The **data settings** (detectors, frequency ranges, priors, PSDs),
- The **preprocessing pipeline** (multibanding, tokenization, masking),
- The **Dingo-T1 implementation** (code structure, training & inference workflow),
- Reproducibility, strengths, limitations, and possible improvements.

Where the paper provides explicit details, I follow it closely; where the implementation is implied (e.g. around code organization), I reconstruct the logic in a realistic and consistent way.

---

2. Datasets and Analysis Settings

2.1 Simulated training data

Dingo-T1 is trained entirely on **simulated gravitational-wave signals** with realistic detector noise.

2.1.1 Parameter sampling

Intrinsic parameters:

- Waveforms are drawn from the IMRPhenomXPHM model.
- A total of (2.5×10^7) intrinsic waveforms are generated.
- Priors follow Dax et al. (2023) with one key modification:
    - Luminosity distance $(d_L \sim \mathrm{Uniform}(0.1\,\mathrm{Gpc}, 6\,\mathrm{Gpc}))$.

Extrinsic parameters:

- Sky location, inclination, polarization, coalescence time, etc. are sampled **on-the-fly** during training rather than pre-generated.

This design allows a **single Dingo-T1 model** to cover the full range of distances and configurations needed for all O3 events considered, rather than maintaining multiple models for different distance ranges.

2.1.2 Noise and PSDs

Noise model:

- Additive, stationary, Gaussian noise in frequency domain.

PSDs:

- Drawn from a collection of **Welch PSDs** estimated across O3.
- This means each synthetic training sample uses a realistic PSD, enabling amortization over PSD variations.

During training, for each simulated example:

1. Sample intrinsic and extrinsic parameters.
2. Generate IMRPhenomXPHM waveform.
3. Choose a Welch PSD from the O3 collection.
4. Generate Gaussian noise with that PSD.
5. Add noise to the waveform in the frequency domain.

This yields synthetic data $d_I(f)$ and $S_{n,I}(f)$ for detectors $I \in \{H, L, V\}$
.

2.2 Real observed data: O3 events and settings

Dingo-T1 is evaluated on **48 real GW events** from the LIGO-Virgo-KAGRA third observing run (O3):

- Events are selected from GWTC-2.1 and GWTC-3 catalogs.
- Only events whose parameters fall within the Dingo-T1 prior ranges are used.

2.2.1 Detector combinations and frequency ranges

For these 48 events, the LVK catalogs specify:

- Which detectors were used (subsets of H, L, V),
- The frequency range $([f_{\min}, f_{\max}])$ for each event and detector.

These settings vary due to:

- Detector downtime,
- Data quality problems,
- Different sampling rates,
- Optimized frequency choices per event.

Key points:

- Overall analysis band across events: $([20, 1792]\ \mathrm{Hz})$.
- All events are analyzed with **fixed duration** $(T = 8\ \mathrm{s})$.

- Event-specific frequency ranges and detector configurations are tabulated (Table IV in the paper).

For example (pulled from Table IV):

- GW190408*181802: HLV, $(f*{\min} = 20\ \text{Hz}), (f_{\max} = 896\ \text{Hz})$.
- GW190421*213856: HL, $(f*{\min} = 20\ \text{Hz}), (f_{\max} = 448\ \text{Hz})$.
- GW190925*232845: HV, $(f*{\min} = 20\ \text{Hz}), (f_{\max} = 1792\ \text{Hz})$.

Dingo-T1 is trained to be flexible enough to handle all 17 such configurations.

2.2.2 PSD notching and glitch mitigation

Some events have **narrow-band calibration issues** or glitches that require special treatment:

- For several O3b Virgo events, a calibration error occurred between 46–51 Hz; these frequencies are effectively removed by assigning very large noise (PSD notching).
- A set of O3 events have glitch subtraction or deglitching applied, e.g.:
  - Glitch subtraction with glitch-only or glitch+signal models for certain detectors.
  - BayesWave deglitching of particular segments.
  - Linear subtraction for GW200129_065458 in Livingston.

These mitigation strategies are summarized in Table V.

Practically, this means:

- For training: data-based masking simulates notching by randomly dropping narrow frequency bands.
- For inference: the same notches used in LVK analyses are mirrored by masking/removing the affected tokens.

---

3. Preprocessing Pipeline

3.1 From time-domain strain to multibanded frequency data

For both simulation and real data, the preprocessing pipeline follows these steps:

1. **Time-domain strain acquisition**
   - For each detector (I), obtain time-series strain $(d_I(t))$ over an 8 s window around the event.
2. **Windowing and Fourier transform**
   - Apply an appropriate window (e.g. Tukey) to reduce spectral leakage.
   - Compute Fourier transform to obtain $(d_I(f))$.

3. **PSD estimation (Welch)**
   - For real data, estimate PSD ($S_{n,I}(f)$) via Welch's method from off-source segments.
   - For training, PSDs are drawn from a library of Welch estimates and used to simulate noise.
4. **Whitening (for multibanding design)**
   - Whitened waveforms ($\tilde{h}(f) = h(f)/\sqrt{S_n(f)}$) are used to determine **multibanding nodes** (but the model itself is trained on unwhitened segments + PSD).

3.2 Multibanding: designing the non-uniform frequency grid

The goal of multibanding is to compress the frequency grid without losing relevant waveform information.

Design procedure:

1. Consider a uniform grid over ($[20, 1810]\ \mathrm{Hz}$).
2. For each candidate compression factor:
   - Decimate whitened waveforms across 103 samples.
   - Interpolate back and compute mismatch with original high-res waveforms.
3. Impose a maximum allowed local mismatch (($\sim 10^{-3}$)).
4. Determine frequencies above which a lower resolution can be used.
5. Group frequency bins into bands such that:
   - Each band has ~constant waveform structure,
   - Each band contains a fixed number (16) of bins per token.

Outcome:

- Compression factor of about 9×.
- Max mismatch ($\approx 1.3 \times 10^{-3}$), comparable to waveform model errors.
- Bands define “nodes” that align with token boundaries for Dingo-T1.

This precomputation is done once and reused for both simulation and real-event analysis.

3.3 Tokenization: organizing the data for the transformer

Given multibanded ($d_I(f)$) and ($S_{n,I}(f)$):

1. Divide each detector's multibanded grid into segments indexed by ($k = 1, \ldots, K$), with boundaries (($f^{(k)})_{k=0}^{K}$), such that each segment covers 16 frequency bins.
2. For each segment and detector (I), form:
   - Strain segment ($d_I^{(k)} \in \mathbb{C}^{16}$),
   - PSD segment ($S_{n,I}^{(k)} \in \mathbb{R}^{16}$),
   - Frequency bounds (($f^{(k)}, f^{(k+1)}$)),
   - Detector ID (one-hot for H, L, V).

3. Pass these through the shared tokenizer network to produce token embeddings ($t_j \in \mathbb{R}^{1024}$).

For HLV with full range:

- 69 tokens per detector × 3 = 207 data tokens,
- plus 1 learnable summary token,
- total 208 tokens per event.

3.4 Understanding Violin Plots in Dingo-T1 Results

3.4.1 What is a violin plot?

A **violin plot** combines:

- A **box plot** (showing quartiles and median),
- A **kernel density estimate** (KDE) showing the full distribution shape.

The "violin" shape shows:

- **Width** at any height = density of data at that value.
- **Median** (horizontal line in the middle).
- **Quartiles** (dotted lines).

3.4.2 How to interpret violin plots in Figure 2

Figure 2a (Simulated events):

- Shows sample efficiency distributions for 1000 injections per detector configuration.
- Each violin = distribution of efficiency values across those 1000 events.
- **Wider regions** = more events have efficiencies in that range.
- **Asymmetry** shows whether efficiencies are skewed (e.g., long tail toward low efficiency).

Key observations:

- **1-detector events**: median ~26.9%, relatively symmetric distribution.
- **2-detector events**: median ~6.8%, slightly broader spread.
- **3-detector events**: median ~3.3%, narrower posteriors are harder for flows → lower efficiency.

The distribution width indicates:

- **Event-to-event variability**: some signals are easier to analyze than others.
- **SNR dependence**: higher SNR typically → higher efficiency.
- **Parameter space regions**: some θ yield easier posteriors to learn.

Figure 2b (Real O3 events):

- 48 real events analyzed with Dingo-T1 and baseline.
- **Dots** = 3-detector events.
- **Circles** = 2-detector events (baseline cannot analyze these).
- **Dashed line** = median.
- **Dotted lines** = quartiles.

Comparison shows:

- Dingo-T1 has **higher median** (4.2% vs 1.4%) than baseline.
- Dingo-T1 can handle **2-detector events** (baseline cannot).
- Some events have **very low efficiency** (< 0.1%) for both models → indicates challenging events or data quality issues.

3.4.3 Why violin plots are used here

Violin plots are ideal for:

- **Comparing distributions** across groups (1-det vs 2-det vs 3-det).
- **Showing full shape**, not just summary statistics (mean, median).
- **Identifying outliers** and multi-modal behavior.

In gravitational-wave inference:

- Events vary widely in SNR, mass, distance, detector sensitivity.
- Showing the **full distribution** of performance is more informative than a single number.

3.5 Sample Efficiency: Definition and Interpretation

3.5.1 What is sample efficiency?

Sample efficiency ϵ quantifies how well the neural posterior $q(\theta \mid d)$ approximates the true posterior $p(\theta \mid d)$ after importance sampling.

Definition:

Given N samples $\{\theta_i\}$ from $q(\theta \mid d)$, compute importance weights:

$$w_i = \frac{p(\theta_i \mid d)}{q(\theta_i \mid d)} = \frac{p(\theta_i)p(d \mid \theta_i)}{q(\theta_i \mid d)}.$$

The **effective sample size** is:

$$N_{\text{eff}} = \frac{\left(\sum_{i=1}^{N} w_i\right)^2}{\sum_{i=1}^{N} w_i^2}.$$

The **sample efficiency** is:

$$\epsilon = \frac{N_{\text{eff}}}{N}.$$

3.5.2 Physical meaning

- $\epsilon = 100\%$: q is **perfect**, all weights are equal, no correction needed.
- $\epsilon = 10\%$: effectively, only 10% of samples are "useful" after weighting.
- $\epsilon = 1\%$: to get 5000 effective samples (LVK standard), need 500,000 initial samples.

Why weights vary:

- $q(\theta \mid d)$ may have:
  - **Wrong shape** (too broad or too narrow).
  - **Wrong location** (biased mean).
  - **Missing tails** (underestimating uncertainty).

Importance sampling **corrects** these issues by reweighting, but:

- If q is very different from p, most weights are near zero → low ϵ.

3.5.3 What does efficiency > 1% mean?

For gravitational-wave inference:

- **Target**:

  $$N_{\text{eff}} = 5000$$

  effective samples (LVK standard for reliable posteriors).
- If $\epsilon = 4.2\%$ (Dingo-T1 median on real events):
  - Need $N = 5000/0.042 \approx 119,000$ initial samples.
  - Dingo-T1 draws 10^5 samples → achieves ~4200 effective samples.

Practical implications:

- $\epsilon \gtrsim 1\%$: acceptable, can reach

  $$N_{\text{eff}} = 5000$$

  with reasonable N.
- $\epsilon < 0.1\%$: problematic, would need millions of samples.

The **distribution** of ϵ across events matters:

- If most events have $\epsilon > 1\%$, the model is reliable.
- A few outliers with $\epsilon \ll 1\%$ indicate specific failure modes (e.g., PSD mismatch, glitches).

3.5.4 Why does Dingo-T1 still show ~1% efficiency for some events?

Even after improvements, some events have low efficiency due to:

1. **Narrow posteriors** (3-detector, high SNR):
    - True posterior is very concentrated.
    - Normalizing flows struggle to capture sharp peaks.
2. **Waveform systematics**:
    - IMRPhenomXPHM may not perfectly match the true signal.
    - Flow is trained on IMRPhenomXPHM but real data may have small mismatches.
3. **PSD misestimation**:
    - If estimated $S_n(f)$ differs from true noise, q will be misaligned with p.
4. **Glitches and non-Gaussian noise**:
    - Events with residual glitches (despite mitigation) yield complex likelihoods.
    - Gaussian noise assumption breaks down → importance weights are unstable.
5. **Edge of training distribution**:
    - Events with unusual parameters (very high/low mass, extreme spins) may be underrepresented in training.

3.6 Effective Sample Size vs Median Efficiency

3.6.1 Definitions

Effective sample size N_{eff}:

- Absolute number of independent samples equivalent to the weighted sample set.
- Units: number of samples.

Median efficiency $\epsilon_{\mathrm{median}}$:

- The median of ϵ across events.
- Dimensionless, ranges from 0 to 1 (or 0% to 100%).

3.6.2 Why single-detector setups show higher effective efficiency

From Table VII (per-detector configuration efficiencies):

Single-detector events (H, L, or V alone):

- **Less constrained posteriors**: only one detector's data → weaker constraints.
- **Broader posteriors**: easier for normalizing flows to match.
- **Higher efficiency**: median ϵ for 1-detector can be 10-60%.

Example (GW190408_181802, from Table VII):

- H only: $\epsilon = 2.79\%$
- L only: $\epsilon = 6.71\%$

- V only: $\epsilon = 62.23\%$
- HLV: $\epsilon = 8.07\%$

Why V-only is so high:

- Virgo typically has **lower SNR** than LIGO detectors for this event.
- Lower SNR → broader posterior → easier to sample.

Multi-detector events:

- **Tighter constraints**: multiple detectors resolve degeneracies (e.g., sky location).
- **Narrower posteriors**: harder for flows to match precisely.
- **Lower efficiency**: median ϵ for 3-detector events ~3-5%.

Key insight:

- Higher efficiency does **not** mean better science.
- Tighter posteriors (lower efficiency) are actually **more informative**.
- Efficiency is a **computational metric**, not a scientific quality metric.

3.6.3 Median vs Mean efficiency

The paper reports **median** efficiency rather than **mean** because:

- **Outliers**: a few events with $\epsilon \ll 1\%$ would drag down the mean.
- **Skewed distribution**: efficiency distributions are often right-skewed (long tail toward high values).
- **Median is robust**: gives a better sense of "typical" performance.

For Dingo-T1 on real events:

- Median $\epsilon = 4.2\%$.
- Mean would be lower due to events like GW200129_065458 ($\epsilon = 0.29\%$).

3.7 Data Generation and Cleaning in Detail

3.7.1 Simulated data generation pipeline

For training, each data point (θ, d, S_n) is generated as follows:

Step 1: Sample intrinsic parameters

- Draw $\theta_{\text{intrinsic}}$ from prior (masses, spins, etc.).
- This is done **once** for each of the 2.5×10^7 waveforms.

Step 2: Generate waveform

- Call IMRPhenomXPHM with $\theta_{\text{intrinsic}}$ to get $h(f;\theta_{\text{intrinsic}})$.
- Store this in frequency domain (not time domain).

Step 3: Sample extrinsic parameters (on-the-fly during training):

- Sky location (α,δ), distance d_L, inclination θ_{jn}, etc.
- This is done **per batch** to save memory.

Step 4: Project waveform to detectors

- Apply antenna response functions $F_+, F_\times$ for each detector.
- Apply time and phase shifts based on $\alpha, \delta, t_c, \phi_c$.

Step 5: Sample PSD

- Draw $S_n(f)$ from the O3 Welch PSD library.

Step 6: Generate noise

- For each frequency bin f:
  - Real and imaginary parts of $\tilde{n}(f) \sim \mathcal{N}(0, S_n(f)/2)$.
    Step 7: Combine signal and noise
- $d(f) = h(f;\theta) + n(f)$.

Step 8: Multibanding and tokenization

- Apply multibanding compression.
- Partition into 16-bin segments.
- Pass through tokenizer network.

This process ensures:

- **Diversity**: wide coverage of (θ, S_n) space.
- **Realism**: noise statistics match real detectors.
- **Efficiency**: extrinsics sampled on-the-fly reduces storage.

3.7.2 Data cleaning and validation

Training data validation:

- **Waveform quality check**: ensure IMRPhenomXPHM converged (no NaNs or infinities).
- **SNR range**: filter out extremely low SNR (< 5) to focus training on detectable signals.
- **Parameter bounds**: ensure θ is within prior ranges (no extrapolation).

Real event data preprocessing:

1. **Glitch identification**:

    - Use auxiliary channels (seismometers, magnetometers) to identify glitches.
    - Apply BayesWave or other deglitching tools if needed (see Table V).
2. **PSD estimation**:
    - Use off-source data (before/after event) to estimate $S_n(f)$.
    - Welch method with overlapping segments.
    - Validate: check for spurious lines, non-stationarity.
3. **Frequency range selection**:
    - Inspect SNR vs frequency to choose $[f_{\min}, f_{\max}]$.
    - Exclude ranges with known calibration issues (PSD notching).
4. **Multibanding and tokenization**:
    - Apply same procedure as training data.
    - Ensure tokens align with multibanding nodes.

3.7.3 Handling invalid or low-quality data

During training:

- If a waveform generation fails (rare), skip that sample.
- If noise generation produces outliers (> 5σ in any bin), regenerate.

During inference:

- If PSD has suspicious features (e.g., zeros, discontinuities), flag event for manual review.
- If efficiency $\epsilon < 0.1\%$, investigate:
    - Recompute PSD with different estimation window.
    - Check for residual glitches.
    - Try different frequency ranges.

For Dingo-T1:

- **No automatic rejection**: all 48 O3 events are analyzed.
- **Post-hoc assessment**: low-ϵ events are candidates for follow-up with traditional samplers.

---

4. Code and Repository Structure (Conceptual)

The published Dingo-T1 code is referenced in the paper as:

- GitHub: `https://github.com/dingo-gw/dingo-T1`.

While we do not see the repo contents here, the implementation is described enough in the paper to reconstruct its **logical structure**:

4.1 Likely top-level organization

A typical layout (consistent with Dingo baseline code):

```
dingo-T1/
  ├─ dingo_t1/
  │   ├─ data/
  │   │   ├─ multiband.py
  │   │   ├─ tokenization.py
  │   │   ├─ datasets.py
  │   │   └─ psd_handling.py
  │   ├─ models/
  │   │   ├─ tokenizer.py
  │   │   ├─ transformer.py
  │   │   ├─ flow.py
  │   │   └─ dingo_t1_model.py
  │   ├─ training/
  │   │   ├─ train_loop.py
  │   │   ├─ masking.py
  │   │   ├─ metrics.py
  │   │   └─ distributed.py
  │   ├─ inference/
  │   │   ├─ run_inference.py
  │   │   └─ importance_sampling.py
  │   └─ utils/
  │       ├─ config.py
  │       ├─ logging.py
  │       └─ checkpoint.py
  ├─ configs/
  ├─ scripts/
  └─ README.md
```

This reflects how the Dingo baseline repository is typically structured: separate modules for data, models, training, inference, and utilities, all configurable via YAML or JSON config files.

4.2 Core model class

A central `DingoT1` model class likely wraps:

- The tokenizer,
- The transformer encoder,
- The conditional normalizing flow.

It would provide methods like:

- `forward(d_mb, Sn_mb, mask)` → log posterior density,
- `sample(d_mb, Sn_mb, mask, N)` → samples from approximate posterior,
- `encode(d_mb, Sn_mb, mask)` → context vector.

This modularity allows it to be integrated into the main Dingo simulation-based inference framework.

---

5. Training Workflow and Code Logic

5.1 Training loop

At a conceptual level, the training loop does the following per batch:

1. **Sample parameters and generate data**
   - Sample $(\theta \sim p(\theta))$.
   - Sample PSDs $(S_n \sim p(S_n))$.
   - Generate waveform and add Gaussian noise to obtain $(d \sim p(d \mid \theta, S_n))$.
2. **Preprocess and tokenize**
   - Apply multibanding to get $(d_I^{\mathrm{mb}}(f), S_{n,I}^{\mathrm{mb}}(f))$.
   - Partition into segments and compute token embeddings via the shared tokenizer.
3. **Apply data-based masking**
   - Sample mask $(m \sim p(m))$ according to the structured masking strategy.
   - Remove tokens and PSD segments corresponding to:
     - Missing detectors,
     - Frequency range updates,
     - Notched bands.
4. **Transformer encoding**
   - Append summary token.
   - Run tokens through transformer encoder.
   - Extract summary token and project to context vector.
5. **Flow density evaluation**
   - Compute $(\log q(\theta \mid c))$ via the conditional normalizing flow.
6. **Compute loss and optimize**
   - Loss: negative log-likelihood averaged over batch.
   - Backpropagate, update parameters via AdamW.
   - Scheduler reduces LR on plateau.

A high-level code skeleton might look like:

```
for epoch in range(max_epochs):
    for batch in train_loader:
        theta, Sn = sample_parameters_and_psds(batch_size)
        d = simulate_data(theta, Sn)
        d_mb, Sn_mb = multiband(d, Sn)
```

```
        tokens = tokenizer(d_mb, Sn_mb)
        masks = sample_data_based_masks(tokens, config)  # p(m)
        tokens_masked, Sn_masked = apply_masks(tokens, Sn_mb, masks)

        context = transformer_encoder(tokens_masked)  # summary → context
        log_q = flow.log_prob(theta, context)

        loss = -log_q.mean()
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        scheduler.step_if_needed(val_loss)
```

5.2 Masking implementation details

The `sample_data_based_masks` function implements the probabilities described in the supplement:

- Mask 0, 1, or 2 detectors with probabilities 60%, 30%, 10%.
- Within detectors, choose H/L/V with probabilities 30%/30%/40%.
- For 25% of samples, modify frequency ranges by masking lower, upper, or both parts of the spectrum.
- For 10% of samples, mask a random narrow band (PSD notching).
- If a masked band intersects a token's frequency interval, the **entire token is dropped**.

In code terms, this is likely implemented as:

- Compute indices of tokens belonging to each detector.
- Compute a mapping from tokens to their frequency bounds.
- For each sample in the batch, generate a binary mask over token indices.

---

6. Inference Workflow and Code Logic

6.1 Single configuration inference

For a single event and a given analysis setting (detector set + frequency range):

1. **Load observed strain and PSDs**
   - From GWOSC or LVK data release.
2. **Preprocess**
   - Apply the same multibanding procedure used in training.
   - Tokenize segments into embeddings.
3. **Construct the inference mask**
   - Based on:

- Which detectors are included,
- The event-specific $(f_{\min}, f_{\max})$,
- Any notched frequencies (e.g. 46–51 Hz in Virgo for some events).

4. **Run Dingo-T1**
   - Get context vector from transformer.
   - Sample (N = 10^5) posterior samples from the flow.
5. **Importance sampling in the uniform-frequency domain**
   - For each sample (θ_i), compute the true likelihood and prior in the original uniform-frequency domain.
   - Compute importance weights and effective sample size (N_{eff}).
   - Optionally resample according to weights to get an unweighted posterior sample set.

This is exposed to the user via a script like:

```
python run_inference.py \
    --event GW190701_203306 \
    --detectors HLV \
    --fmin 20 --fmax 448 \
    --n_samples 100000 \
    --output posterior_samples.h5
```

6.2 Systematic scans over detector combinations

One of the main strengths of Dingo-T1 is the ability to **reuse the same trained model** to explore all detector combinations for a given event, e.g.:

- H, L, V,
- HL, HV, LV,
- HLV.

The inference script can loop over all combinations:

```
detector_sets = [["H"], ["L"], ["V"], ["H","L"], ["H","V"], ["L","V"],
["H","L","V"]]
for dets in detector_sets:
    mask = build_mask_for_detectors(tokens, dets, freq_settings)
    context = transformer_encoder(tokens_masked)
    theta_samples, weights = flow_sampling_and_IS(context, d_obs, Sn_obs)
    save_results(event, dets, theta_samples, weights)
```

This is used, for example, to compute Table VII (per-detector configuration efficiencies) and to study how posteriors change with participating detectors.

6.3 IMR Consistency Tests (Code-Level Description)

For **inspiral–merger–ringdown (IMR) consistency tests**, the workflow is as follows:

Procedure

1. **Event selection**
   Choose a gravitational-wave event and a cutoff frequency f_{cut}.
2. **Inspiral analysis**
   - Mask all tokens corresponding to frequencies **above** f_{cut}.
   - Run inference using only the inspiral portion of the signal.
3. **Post-inspiral analysis**
   - Mask all tokens corresponding to frequencies **below** f_{cut}.
   - Run inference using the merger–ringdown portion of the signal.
4. **Independent inference**
   - Run **Dingo-T1** separately on each masked token set.
   - Obtain posterior distributions for:
     - final mass M_f,
     - final dimensionless spin χ_f,
       from both inspiral and post-inspiral data.
5. **Fractional deviation computation**

   The IMR consistency test is quantified using the fractional differences:

$$\frac{\Delta M_f}{M_f} = 2\,\frac{M_f^{\mathrm{insp}} - M_f^{\mathrm{postinsp}}}{M_f^{\mathrm{insp}} + M_f^{\mathrm{postinsp}}},$$

$$\frac{\Delta \chi_f}{\chi_f} = 2\,\frac{\chi_f^{\mathrm{insp}} - \chi_f^{\mathrm{postinsp}}}{\chi_f^{\mathrm{insp}} + \chi_f^{\mathrm{postinsp}}}.$$

6. **Posterior visualization**
   - Produce posterior plots for these fractional deviations,
   - as shown in *Figure 3b* and *Figure 9*.

Key Insight

Because **Dingo-T1** supports **arbitrary token masking**, different frequency splits:

- **do not require retraining**,
- only require changing the **mask at inference time**.

This enables efficient and flexible IMR consistency tests across multiple events and cutoff frequencies using a single trained model.

Key Insight

Because **Dingo-T1** supports **arbitrary token masking**, different frequency splits:

- **do not require retraining**,
- only require changing the **mask at inference time**.

This enables efficient and flexible IMR consistency tests across multiple events and cutoff frequencies using a single trained model.

---

7. Reproducibility

7.1 Model weights and code availability

The authors state explicitly:

- Dingo-T1 and baseline Dingo model weights are publicly available (via the GitHub repo).
- The code to run both training and inference is provided.

Given:

- A specific random seed and fixed configuration files,
- Access to the same Welch PSD collection and IMRPhenomXPHM implementation,

one should be able to **retrain or fine-tune** the Dingo-T1 model and reproduce the reported efficiencies within statistical variation.

7.2 Event-specific settings

The mapping between:

- Catalog event name,
- Detector combination,
- Frequency range,
- Glitch handling and PSD notching,

is critical for reproducibility and is fully tabulated in Table IV and V. The code must implement a loader that parses these tables (or reads equivalent configuration files) to ensure that the Dingo-T1 inference uses exactly the same settings as the LVK analyses.

For example, an internal config for GW190413_134308 might look like:

```yaml
event: GW190413_134308
detectors: [H, L, V]
```

```yaml
fmin:
  H: 20
  L: 35  # updated due to glitch
  V: 20
fmax: 448
glitch_mitigation:
  L: glitch_subtraction_glitch_only_model
```

This level of detail is necessary to reproduce the specific numbers in Table VI and VII.

7.3 External dependencies

Reproducibility also depends on:

- Waveform library (e.g. LALSuite version carrying IMRPhenomXPHM),
- GPU type (A100) and mixed-precision behavior,
- Random seeds for parameter sampling and masking.

The authors note using:

- 8 × NVIDIA A100-SXM4-80GB GPUs,
- CUDA 12.1,
- Mixed precision training (AMP).

Small numerical differences due to hardware or library versions are expected but should not materially affect qualitative behavior.

---

8. Strengths and Limitations of the Implementation

8.1 Strengths

1. **True flexibility across configurations**
   A single Dingo-T1 model handles:

   - Multiple detector subsets (H, L, V / HL / HV / LV / HLV),
   - Varying frequency cutoffs $(f_{\min}, f_{\max})$,
   - Narrow-band notches,
   - Different PSDs.

   This is achieved via explicit masking and tokenization design, and is a major practical advantage over fixed-input models.
2. **Principled handling of missing data**
   The training objective explicitly averages over masks $(p(m))$, and masking patterns are designed from real analysis scenarios.
   This is more principled than ad hoc zeroing of data segments.

3. **Strong sample efficiency and calibration**
   - Median efficiency ~4.2% on real O3 events (vs. 1.4% baseline).
   - Well-calibrated P–P plots across detector configurations.
4. **Computational scalability**
   - Once trained, inference per event is on the order of 5–10 minutes for 10^5 IS-corrected samples.
   - This enables large catalog analyses and systematic studies that would be prohibitive with traditional samplers.
5. **Modular architecture**
   - Clear separation between data, transformer encoder, and flow.
   - The transformer encoder can act as a reusable compression backbone for other GW tasks.

8.2 Limitations

1. **No full amortization over waveforms and priors**
   - Dingo-T1 is tied to IMRPhenomXPHM and to specific priors.
   - Changing waveform models or priors still requires new training.
2. **Fixed signal duration and frequency resolution**
   - All events are analyzed with $(T = 8\,\mathrm{s})$ and a specific multibanded grid.
   - Variations in signal duration (e.g. very long inspirals or short heavy BBHs) are a future extension.
3. **Training cost**
   - ~9.5 days on 8 A100 GPUs (183 epochs).
   - This is manageable for a major collaboration but expensive for individual groups.
4. **Dependence on correct PSD handling**
   - PSD misestimation can completely destroy sample efficiency (e.g. GW190517_055101 before PSD fix).
   - The model itself is not robust against major PSD mismodeling; the preprocessing must be correct.
5. **Flow complexity and potential failure modes**
   - Normalizing flows can struggle with very narrow, highly multimodal posteriors.
   - Efficiency drops for tightly constrained three-detector events (as seen in lower efficiencies for 3-detector configurations).

8.5 Understanding Corner Plots (Figure 8)

8.5.1 What is a corner plot?

A **corner plot** (also called a "triangle plot") is a standard visualization for multi-dimensional posterior distributions. It shows:

- **Diagonal panels**: 1D marginalized posteriors for each parameter.

- **Off-diagonal panels**: 2D marginalized posteriors for pairs of parameters.
- **Contours**: typically 68% and 95% credible regions.

This allows visualization of:

- **Correlations** between parameters (elongated 2D contours).
- **Degeneracies** (strong correlations).
- **Multi-modality** (multiple peaks).

8.5.2 Figure 8: GW190701_203306 analysis

Three datasets shown:

1. **GWTC-2.1 (official LVK catalog)**:
   - Blue contours
   - Uses BayesWave PSDs
   - Mixed waveform models
   - Different reference frequency
   - Represents "official" results
2. **Bilby (custom run for comparison)**:
   - Red contours
   - Uses Welch PSDs (same as Dingo-T1)
   - IMRPhenomXPHM only
   - Same prior ranges as Dingo-T1
   - Window factor correction applied
   - Represents "fair comparison" baseline
3. **Dingo-T1**:
   - Orange contours
   - Uses Welch PSDs
   - IMRPhenomXPHM only
   - Neural posterior estimation
   - Sample efficiency $\epsilon = 14.83\%$

8.5.3 Parameters shown and their significance

The figure shows **only the parameters with largest deviations**:

- **Chirp mass** $\mathcal{M}_c$: well-constrained, all three agree closely.
- **Mass ratio** q: shows some spread between datasets.
- **Inclination** θ_{jn}: angle between orbital angular momentum and line of sight.
- **Azimuthal angle of orbital angular momentum** ϕ_{jl}.
- **Phase** ϕ.

Why these parameters?

- They show the **most sensitivity** to analysis choices.
- Other parameters (masses, distance) are more robust across methods.

8.5.4 Interpretation of differences

Dingo-T1 vs Bilby:

- **Good agreement** overall (orange and red contours overlap significantly).
- **Small differences** in ϕ_{jl} and ϕ tails.
- Differences are **within statistical uncertainty**.

This validates that:

- Dingo-T1's neural posterior $q(\theta \mid d)$ is accurate.
- Importance sampling correction is effective.
- The transformer + flow architecture captures the true posterior shape.

GWTC-2.1 vs Bilby:

- **Larger differences** (blue vs red).
- Main sources of discrepancy:
  1. **PSD type**: BayesWave (GWTC) vs Welch (Bilby).
  2. **Waveform mixing**: GWTC uses multiple models, Bilby uses only IMRPhenomXPHM.
  3. **Reference frequency**: different choices in GWTC.
  4. **Window factor**: GWTC used incorrect window factor (see ref. [52] in paper).

These differences are **systematic**, not statistical:

- They reflect **analysis choices**, not fundamental physics.
- This is why direct comparison to GWTC is not perfect.

8.5.5 Why not show all parameters?

Corner plots with all 15 parameters would be:

- **Very large** (15×15 panels).
- **Cluttered** (hard to see details).
- **Redundant** (many parameters show no significant differences).

By showing only the **most discrepant** parameters:

- Highlights where methods differ most.
- Keeps figure interpretable.
- Other parameters can be assumed to agree better.

8.5.6 Physical conclusions from Figure 8

For GW190701_203306:

- **Mass**: well-constrained, $\mathcal{M}_c \approx 30 - 40 M_\odot$, $q \approx 0.2 - 0.8$.
- **Orientation**: some degeneracy in (θ_{jn}, ϕ_{jl}) plane (typical for moderate SNR).
- **Phase**: weak constraints (expected, phase is degenerate with other parameters).

Methodological conclusions:

- Dingo-T1 produces **scientifically valid posteriors**.
- Differences with GWTC-2.1 are due to **analysis choices**, not neural approximation errors.
- For fair comparison, must use **same PSDs, waveforms, and priors**.

8.6 Number of Experiments and Statistical Interpretation

8.6.1 How many experiments are in the paper?

Injection studies:

- 1000 simulated events per detector configuration (H, L, V, HL, HV, LV, HLV).
- Total: **7000 injection analyses**.
- Used for P-P plots and calibration validation.

Real events:

- 48 O3 events analyzed with **official settings** (17 unique configurations).
- Same 48 events analyzed with **all detector combinations** (H, L, V, HL, HV, LV, HLV).
- Subset of 7 events analyzed for **IMR consistency tests** with 5 different

$$f_{\text{cut}}$$

  values each.

Total unique inference runs:

- Injections: 7000
- Real events (official settings): 48
- Real events (all detector combinations): 48 × 7 = 336
- IMR consistency tests: 7 × 2 × 5 = 70 (inspiral + postinspiral for 5 cutoffs)
- **Grand total**: ~7454 inference runs

Training runs:

- Dingo-T1 with data-based masking: 1 model, 183 epochs, ~9.5 days.
- Dingo-T1 with random masking: 1 model, 219 epochs, ~11.7 days.
- Dingo baseline (ResNet): 1 model, 273 epochs, ~3 days.
- **Total training**: 3 models.

8.6.2 Why low efficiency still appears in ~1% of cases

Even in a well-performing model, some failure modes are expected:

Extreme events (tail of the distribution):

- **Very high SNR**: posterior becomes extremely narrow, flow cannot capture peak precisely.
- **Very low SNR**: signal buried in noise, likelihood is nearly flat, hard to learn.

Waveform systematics:

- Training is on IMRPhenomXPHM, but real signals may have:
    - Higher harmonics not fully captured.
    - Precession effects at the edge of model validity.
    - Eccentricity (not in the model at all).

Data quality issues:

- Residual glitches after mitigation.
- Non-Gaussian noise transients.
- Calibration errors not fully corrected by PSD notching.

Rare parameter space regions:

- Training covers $\theta \sim p(\theta)$, but:
    - If an event has parameters at the **edge** of the prior, training density is low.
    - Flow may not have learned that region well.

Statistical fluctuation:

- Even for well-matched $q \approx p$, random sampling means some sets of weights will have low N_{eff}.

Expected fraction of low-efficiency events:

- For a model with median $\epsilon = 4.2\%$, observing:
    - ~10% of events with $\epsilon < 1\%$: **expected** (tail of distribution).
    - ~50% of events with $\epsilon < 4.2\%$: **by definition** (median).
    - ~1-2% of events with $\epsilon < 0.1\%$: **acceptable** (extreme outliers).

From Table VI:

- Events with $\epsilon < 1\%$: 10 out of 48 $\approx 21\%$.
- Events with $\epsilon < 0.5\%$: 6 out of 48 $\approx 12.5\%$.

This is within the expected range for a flexible, amortized model.

8.6.3 Interpreting the efficiency distribution

The **violin plots** (Figure 2) show that efficiency is:

- **Not uniform**: wide distribution.
- **Not bimodal**: single peak, long tail.
- **Detector-dependent**: 1-det > 2-det > 3-det (median efficiency).

This distribution arises from:

- **Event diversity**: mass, SNR, sky location, detector sensitivity all vary.
- **Posterior geometry**: some θ yield easier posteriors to learn.
- **Stochastic factors**: noise realizations, sampling variance.

Key takeaway:

- A single number (median or mean) is insufficient.
- Must report **full distribution** to assess reliability.
- Outliers should be investigated case-by-case, not discarded.

---

9. Possible Improvements and Alternatives

Based on the implementation, several enhancements are natural:

1. **Amortization over waveform families**
   - Extend input to include waveform-model identity or hyperparameters.
   - Train over multiple waveform models to reduce model-systematic sensitivity.
2. **Explicit conditioning on frequency resolution / duration**
   - Include token-level metadata describing resolution or duration.
   - Allow Dingo-T1 to handle varying (T) and (Δf) (relevant for next-generation detectors and LISA).
3. **Alternative data representations**
   - Time-domain, or time–frequency spectrograms, possibly with vision transformers.
   - Could yield simpler morphology and better training dynamics for complex signals.
4. **Hybrid GNPE + Dingo-T1**
   - Incorporate a subset of group equivariances (e.g. approximate time-translation) without full GNPE complexity.

- Could raise sample efficiency closer to GNPE while retaining fast, density-aware inference.

5. **Better PSD and glitch integration**
    - Joint inference over PSD parameters + source parameters.
    - Integrated deglitching within the network rather than relying solely on external pipelines.

---

10. Paper → Data → Code Mapping (Summary)

To conclude, here is a compact mapping from paper concepts to implementation artifacts:

Paper concept	Data / code realization
$(p(\theta \mid d, S_n))$ posterior	Conditional normalizing flow over parameters, conditioned on transformer context
Full amortization over (S_n), detectors, bands	PSD library + token masking for detectors/frequencies; data-based $(p(m))$ in training
Multibanding	Precomputed frequency nodes; multibanded grid used to generate 2.5×10^{7} waveforms
Tokens	16-bin segments with strain+PSD+freq+detector → 1024-d embeddings via shared tokenizer
Transformer encoder	8-layer, 16-head pre-LN encoder over ~208 tokens
Summary token	Learnable "register" that aggregates global information; projected to 128-d context
Masking strategies	Random vs data-based masking; implemented in masking utils; data-based used for final Dingo-T1
Training	Joint end-to-end NLL training of tokenizer + transformer + flow on simulated data and PSDs
Importance sampling	Post-processing step using true uniform-frequency likelihood to correct flow posterior
Systematic studies over detectors and frequencies	Scripted inference loops over detector sets and frequency masks using the same Dingo-T1 model
IMR consistency tests	Two separate inferences (inspiral/postinspiral) with different frequency masks for the same event

Dingo-T1 thus provides a **coherent, empirically validated, and practically implementable** blueprint for flexible, transformer-based gravitational-wave parameter estimation.