1. Physical Problem: Gravitational-Wave Parameter Estimation

1.1 What are gravitational waves?

Gravitational waves (GWs) are ripples in spacetime produced by accelerated masses, especially compact objects like black holes and neutron stars.
When two compact objects orbit each other and eventually merge, their changing quadrupole moment generates waves that propagate at the speed of light.

We can think of spacetime as a flexible medium. A binary system is like two heavy balls moving on a sheet; their orbital motion generates waves that propagate outward.

The detectors (LIGO Hanford H, LIGO Livingston L, Virgo V, KAGRA) are kilometer-scale Michelson interferometers.
They measure **strain**

$$h(t)$$

, the fractional change in arm length:

$$h(t) = \frac{\Delta L(t)}{L}.$$

The data actually recorded is:

$$d(t) = h(t) + n(t),$$

where

$$n(t)$$

is noise from seismic, thermal, quantum, and instrumental sources.

1.2 Compact binaries and waveform phases

The paper focuses on **compact binary coalescences** (CBC):

- Binary black holes (BBH)
- Binary neutron stars (BNS)
- Possibly neutron-star–black-hole binaries

The coalescence(Merging) has three conceptual phases:

1. **Inspiral** (low frequency, long duration)
   - Two objects orbit each other and slowly spiral in due to gravitational radiation reaction.
   - Frequency and amplitude of GWs **chirp** (increase with time).

- This phase is relatively well described by post-Newtonian theory.

2. **Merger** (peak luminosity)
   - Horizons touch and merge.
   - Fully nonlinear strong-field regime of GR.
   - Short but very loud in GWs.
3. **Ringdown**
   - The remnant black hole is perturbed.
   - It rings down via quasi-normal modes.
   - Exponentially decaying sinusoids characterized by final mass (M_f) and spin (χ_f).

The observed waveform $(h(t;\theta))$depends on source parameters (θ).
What is source parameters?

1.2.1 Parameterization of the source

The parameter vector (θ) typically includes:

- **Intrinsic parameters** (internal to the binary):
  - Individual masses (m_1, m_2) or chirp mass $(\mathcal{M}_c)$ and mass ratio $(q = m_2/m_1)$.
  - Spin magnitudes (a_1, a_2) and spin orientation angles.
  - Possibly eccentricity (not considered here).
- **Extrinsic parameters**:
  - Luminosity distance (d_L).
  - Sky position: right ascension (α), declination (δ).
  - Orientation: inclination angle (θ_{jn})(angle between total angular momentum and line of sight), polarization angle (ψ).
  - Coalescence time (t_c) and phase (ϕ_c).

The paper's model infers all parameters except the phase, which is reconstructed analytically in post-processing.

1.3 What does "parameter estimation" mean?

Given detector data (d) (time/frequency series from H/L/V), we want to infer a **posterior distribution** over (θ), i.e.:

$$p(\theta \mid d, S_n),$$

where (S_n) is the noise power spectral density (PSD) describing frequency-dependent noise levels.

This posterior tells us:

- What mass and spin the black holes likely had.
- Where they were located and how far away they are.

- How consistent the event is with General Relativity (GR).

This is critical for:

- Astrophysical population studies.
- Cosmological measurements (e.g. Hubble constant).
- Fundamental tests of GR (e.g. inspiral–merger–ringdown consistency).

---

2. Bayesian Inference & GW Likelihood (Mathematics)

2.1 Bayes' theorem in GW inference

Bayes' rule gives:

$$p(\theta \mid d) = \frac{p(\theta)\, p(d \mid \theta)}{p(d)}.$$

- $(p(\theta))$: prior over parameters (physical assumptions before data).
- $(p(d \mid \theta))$: likelihood (how likely the data is if parameters are (θ)).
- $(p(d))$: evidence (normalization, often not needed for parameter estimation).
- $(p(\theta \mid d))$: posterior.

In the more explicit, PSD-conditioned form used in the supplement:

$$p(\theta \mid d) = \frac{p(\theta)\, p(d \mid \theta)}{p(d)}, \quad p(d \mid \theta, S_n) \propto \exp\left[-\frac{1}{2}\langle d - h(\theta) \mid d - h(\theta)\rangle_{S_n}\right]$$

2.2 Assumptions about detector noise

The standard GW likelihood assumes:

- Noise is **additive**: $(d = h(\theta) + n)$.
- Noise is **stationary**: statistical properties do not change over time.
- Noise is **Gaussian**, fully characterized by its PSD $(S_n(f))$.

In the **frequency domain**, this means:

- Real and imaginary parts of $(\tilde{n}(f))$ are Gaussian with variance proportional to $(S_n(f))$.
- Different frequencies are approximately independent.

The likelihood for a given detector configuration is then Gaussian in the data-vector space, with covariance given by $(S_n(f))$.

2.3 Noise-weighted inner product

The core object in GW statistics is the **noise-weighted inner product**:

$$\langle a \mid b \rangle_{S_n} = 4 \sum_I \Re \int_{f_{\min,I}}^{f_{\max,I}} \frac{a_I^*(f)\, b_I(f)}{S_{n,I}(f)}\, df$$

Where:

- ($I \in \{H, L, V\}$) indexes detectors.
- ($a_I(f)$), ($b_I(f)$): complex frequency-domain series for detector (I).
- ($S_{n,I}(f)$): PSD of detector (I).
- ($f_{\min,I}, f_{\max,I}$): analysis frequency range for detector (I).
- ($\Re$): real part.

Physical meaning:

- Frequencies where the detector is more sensitive (smaller ($S_n(f)$)) contribute more.
- This inner product is analogous to a weighted Euclidean inner product where the weight is (S_n^{-1}).

2.4 Likelihood in terms of inner product

Under Gaussian noise, the likelihood is:

$$p(d \mid \theta, S_n) \propto \exp\left(-\frac{1}{2}\langle d - h(\theta) \mid d - h(\theta)\rangle_{S_n}\right).$$

Expanding the inner product:

$$\langle d - h \mid d - h \rangle = \langle d \mid d \rangle - 2\Re\langle d \mid h \rangle + \langle h \mid h \rangle.$$

The term ($\langle d \mid d \rangle$) does not depend on (θ) and only contributes to the normalization. For sampling, the relevant part is:

$$\log p(d \mid \theta, S_n) = -\frac{1}{2}\langle h(\theta) \mid h(\theta)\rangle_{S_n} + \Re\langle d \mid h(\theta)\rangle_{S_n} + \text{const.}$$

This is mathematically equivalent to a **chi-squared** measure of mismatch between the data and the template waveform.

2.5 Full amortization and analysis settings

The posterior formally depends on more than just (d): it depends on all **analysis choices**:

- Prior ($p(\theta)$).
- Waveform model ($h(\theta)$) (here IMRPhenomXPHM).
- PSDs ($S_{n,I}(f)$).
- Set of detectors used.
- Frequency ranges ($[f_{\min,I}, f_{\max,I}]$).
- Frequency resolution (Δf).

So really:

$$p(\theta \mid d;\, S_n, \text{detectors}, f_{\min}, f_{\max}, \text{prior}, \text{model}).$$

Full amortization means:
one neural model ($q(\theta \mid d, S_n)$) that can handle **all relevant combinations** of these settings **without retraining**.
Dingo-T1 amortizes over:

- PSD variations.
- Detector subsets (H, L, V combinations).
- Frequency ranges (changing $(f_{\min}, f_{\max})$).
- Local frequency cuts (PSD notching).

It does **not yet** amortize over:

- Priors,
- Waveform models,
- Frequency resolution (signal duration).
  These are listed as future work.

2.6 Understanding the Power Spectral Density (PSD)

2.6.1 What is the PSD physically?

The **Power Spectral Density** $S_n(f)$ describes how the **power** of the noise is distributed across frequencies. It is a statistical characterization of the noise, **not the noise itself**.

For stationary Gaussian noise $n(t)$, the frequency-domain noise $\tilde{n}(f)$ has:

$$\mathbb{E}[\tilde{n}(f)\tilde{n}^*(f')] = \frac{1}{2} S_n(|f|)\delta(f - f').$$

This means:

- The real and imaginary parts of $\tilde{n}(f)$ are independent Gaussian random variables.
- Their variance at frequency f is proportional to $S_n(f)$.

Physical meaning:

- Where $S_n(f)$ is **large**: the detector is **noisy** at that frequency (low sensitivity).
- Where $S_n(f)$ is **small**: the detector is **quiet** (high sensitivity).

2.6.2 Why does PSD have units $1/\sqrt{\mathrm{Hz}}$?

The strain $h(t)$ is dimensionless (fractional length change). In frequency domain:

- $\tilde{h}(f)$ has units $[1/\mathrm{Hz}]$ (due to Fourier transform).

- The **power** (variance) of noise at frequency f is $S_n(f) \cdot \Delta f$.
  For this to be dimensionless (variance of a dimensionless quantity), we need:

$$S_n(f) \cdot \Delta f \sim [\text{dimensionless}]$$

Thus:

$$S_n(f) \sim [1/\text{Hz}].$$

However, by convention, we often write $S_n(f)$ as a **strain spectral density**, with units:

$$\text{strain}/\sqrt{\text{Hz}}.$$

This is equivalent to

$$\sqrt{S_n(f)}$$

having units

$$[1/\sqrt{\text{Hz}}]$$

, which is the **amplitude spectral density** (ASD).

2.6.3 How noise enters the likelihood

The Gaussian likelihood depends on the noise-weighted inner product:

$$\langle a \mid b \rangle_{S_n} = 4 \sum_I \Re \int_{f_{\min,I}}^{f_{\max,I}} \frac{a_I^*(f) b_I(f)}{S_{n,I}(f)} df.$$

Here:

- $1/S_n(f)$ acts as a **frequency-dependent weight**.
- Frequencies where the detector is more sensitive (small S_n) contribute **more** to the likelihood.
- Frequencies where noise is large contribute **less**.

This weighting is **optimal** under the Gaussian noise assumption: it maximizes the signal-to-noise ratio.

2.6.4 PSD estimation vs true noise

Key distinction:

- $S_n(f)$ is an **estimate** of the noise power, typically computed via Welch's method from off-source data.
- The **actual noise** $n(t)$ in the data is a random realization.

For Dingo-T1:

- During **training**: PSDs are drawn from a library of Welch estimates across O3, and noise is **simulated** with those PSDs.
- During **inference**: the PSD is estimated from the specific event's data and **conditioned on** by including S_n segments in the tokens.

This conditioning is crucial: the same signal with different PSDs would have different posteriors.

3. Waveform Model and Data Generation

3.1 IMRPhenomXPHM

Dingo-T1 is trained on waveforms generated using **IMRPhenomXPHM**, a phenomenological model for precessing binary black holes with higher harmonics.

- **IMR**: inspiral–merger–ringdown.
- **Phenom**: fits to numerical relativity and post-Newtonian results.
- **X**: improved and extended model.
- **P**: includes precession effects (spins misaligned with orbital angular momentum).
- **HM**: includes higher multipoles beyond the dominant $(\ell = 2, m = \pm 2)$.

The model outputs a complex frequency series $(h(f; \theta))$ for each detector, after applying:

- antenna response functions $(F_+, F_\times)$,
- time and phase shifts,
- projection depending on sky location and polarization.

3.2 Priors

The prior ranges are mostly as in Dax et al. (2023), with one key difference:

- Luminosity distance prior: $(d_L \in [0.1, 6]\ \mathrm{Gpc})$, uniform.

This is chosen so that **a single Dingo-T1 model** covers the full range of catalog events, instead of training separate models for different distance ranges.

The model infers all parameters except the phase, which is reconstructed later analytically.

3.3 Simulated training data

Training data are generated as follows:

1. Sample intrinsic parameters from the chosen prior.
2. Generate waveform $h(\theta)$ using IMRPhenomXPHM.
3. Sample extrinsic parameters (sky position, distance, orientation, time of coalescence) on-the-fly.
4. Generate noise:

    - PSDs from a collection of Welch PSD estimates from actual O3 data.
    - Add stationary Gaussian noise with that PSD to the signal.
5. Transform to frequency domain if needed, and apply multibanding (see next section).

Total:
2.5×10^7 intrinsic waveforms (plus extrinsics generated ad hoc during training).

---

4. Data Representation: Frequency Domain & Multibanding

4.1 Why frequency domain?

The likelihood involves integrals over frequency weighted by $(1/S_n(f))$.
Frequency-domain representations:

- Make convolution-like operations (e.g. filtering) more efficient.
- Align naturally with how the likelihood is defined.
- Are standard in GW parameter estimation.

Each detector (I) yields:

- $(d_I(f))$: data strain in frequency domain.
- $(S_{n,I}(f))$: PSD.

4.1a Understanding Frequency Limits and Cutoffs

Physical meaning of f_{min} and f_{max}

The frequency limits define the **analysis band**:

- f_{min}

  : lowest frequency included in the analysis.
- f_{max}

  : highest frequency included.

These are **not** simple high-pass or low-pass filters in the traditional sense. Instead:

Lower frequency cutoff f_{min}:

- Typically set by **detector sensitivity**: below ~20 Hz, seismic noise dominates.
- Also limited by **computational cost**: lower frequencies require longer time segments.
- Sometimes **raised** for specific events to avoid non-Gaussian noise artifacts (glitches, calibration issues).

Upper frequency cutoff $f_{\max}$:

- Determined by **signal morphology**: heavier systems merge at lower frequencies.
- Limited by **sampling rate**: Nyquist frequency is $f_s/2$.
- For a binary with total mass M, most power is radiated below a few

$$\times(c^3)/(GM) \sim \mathrm{kHz} \cdot (10M_\odot/M)$$

.

Relation to detector sensitivity

The noise PSD $S_n(f)$ varies with frequency:

- At very low frequencies (< 10 Hz): seismic noise, large S_n.
- Around 100-300 Hz: detector sensitivity is **best**, smallest S_n.
- At high frequencies (> 1 kHz): shot noise dominates, S_n increases.

The choice of

$$[f_{\min}, f_{\max}]$$

should ideally:

- Include frequencies where S_n is small (good sensitivity).
- Exclude frequencies where signal is negligible or noise is non-Gaussian.

Impact on inference accuracy

Narrower frequency range:

- **Fewer constraints** on parameters $\rightarrow$ **broader posteriors**.
- Particularly affects parameters that are degenerate over short frequency ranges (e.g., mass and distance).

Wider frequency range:

- **More information** $\rightarrow$ **tighter posteriors**.
- But only if the data in the extra range is clean and well-modeled.

For Dingo-T1:

- The model must learn how posteriors **scale** with available frequency content.
- Training with variable

$$f_{\min}, f_{\max}$$

(via masking) teaches the model these dependencies.

4.2 Frequency range and duration

- Typical global minimal frequency ($f_{\min} = 20\,\mathrm{Hz}$).
- Maximum frequency depends on sampling rate, up to ($\sim 1792\,\mathrm{Hz}$).
- Fixed signal duration ($T = 8\,\mathrm{s}$).

Resolution is ($\Delta f = 1/T = 0.125\,\mathrm{Hz}$), so raw frequency series would have thousands of bins.

However, analysis frequency ranges vary per event and per detector (see Table IV in the paper).

4.3 Multibanding: non-uniform frequency grid

Problem: A uniform frequency grid is high-dimensional and redundant where the signal is smooth.

Solution: **Multibanding** (non-uniform frequency resolution).

What is multibanding physically?

Multibanding exploits the **multi-scale structure** of GW signals:

- At **low frequencies** (early inspiral): waveform evolves slowly, many cycles per frequency bin.
- At **high frequencies** (late inspiral, merger): waveform evolves rapidly, but amplitude decreases.

A uniform frequency grid wastes resolution where the waveform is smooth and may undersample where it changes rapidly.

Multibanding solution:

- Use **fine resolution** at low frequencies (where signal is strong and slow).
- Use **coarse resolution** at high frequencies (where signal is weak and fast, or smooth).

This is analogous to:

- **Wavelet transforms** in signal processing (multi-resolution analysis).
- **Adaptive mesh refinement** in numerical simulations.

Computational role

- Reduces dimensionality by ~9× without losing waveform information.
- Speeds up training (fewer parameters to process).
- Maintains mismatch below

  $$1.3 \times 10^{-3}$$

  (comparable to waveform modeling errors).

Purpose in this project

- Multibanding is applied **before tokenization**.
- Each token contains 16 bins from the **multibanded** grid.
- This ensures tokens have manageable size while capturing essential waveform features.

Detailed procedure

Important

Procedure (from supplemental material):

1. Consider a range $[20, 1810]\,\mathrm{Hz}$.
2. For candidate compression factors, take whitened waveforms and **decimate** frequency bins.
3. Compare decimated waveform to high-resolution version, compute local mismatch.
4. Impose a maximum allowed local deviation to decide where resolution can be reduced.
5. Group consecutive bins into segments (tokens) of fixed size (16 bins).
6. Define "nodes" (band boundaries) so that each band is compatible with the allowed compression.

Outcome:

- A compressed multibanded grid with a factor ~ 9 reduction in dimension.
- Maximum mismatch between decimated and interpolated waveforms $\leq 1.3 \times 10^{-3}$, comparable to waveform modeling errors of IMRPhenomXPHM itself.
- This multibanded grid is used to generate the training dataset.

4.4 Tokenization of frequency data

Once we have the multibanded frequency grid, we define **tokens**.

For each detector $(I \in \{H, L, V\})$:

1. Partition the multibanded frequency grid into (K) equal-length segments in the **index** space, with boundaries $((f^{(k)})_{k=0}^{K+1})$.
2. Each segment $([f^{(k)}, f^{(k+1)}])$ contains exactly 16 frequency bins.
3. Extract:
   - the strain segment $(d_I^{(k)} \in \mathbb{C}^{16})$,
   - the PSD segment $(S_{n,I}^{(k)} \in \mathbb{R}^{16})$.

This defines (3K) segments overall. Across the full frequency range, the paper has (K = 69) per detector, hence:

- (69) tokens per detector,
- $(3 \times 69 = 207)$ tokens for HLV.

Each segment, together with metadata, forms one **token**.

5. Tokenizer Network and Token Embeddings

5.1 Information contained in a token

Each token encodes:

$$\boxed{t\left(d_I^{(k)}, S_{n,I}^{(k)}, f^{(k)}, f^{(k+1)}, I\right),}$$

where:

- $(d_I^{(k)})$: 16-bin strain segment (real + imaginary parts).
- $(S_{n,I}^{(k)})$: 16-bin PSD segment.
- $(f^{(k)}, f^{(k+1)})$: frequency bounds of the segment.
- (I): detector identity (H, L, or V) as a one-hot vector.

Just giving the raw 16 complex values / 16 PSD values is not enough; the network must know:

- which detector they come from,
- what frequency interval they represent.

5.1a Why Tokenization is Essential for Transformers

Tokenization for non-language data

In natural language processing (NLP), tokenization means breaking text into discrete units (words, subwords, characters). For transformers:

- Each token is mapped to an embedding vector.
- The transformer processes **sequences of these embeddings**.

For gravitational-wave data:

- Raw data is a **continuous frequency series** with thousands of bins.
- We cannot feed the entire series as one token (too high-dimensional, loses locality).
- We cannot feed each frequency bin as a token (sequence too long, loses local correlations).

Solution: Partition the frequency series into **segments** (tokens) of manageable size (16 bins).

Why this representation works for GW signals

Gravitational-wave signals have **multi-scale structure**:

- **Local structure**: within a narrow frequency band, the waveform is relatively smooth.
- **Global structure**: across the full band, the chirping pattern encodes mass, spin,

distance.

Tokenization captures both:

- **Each token** encodes local frequency content (16 bins).
- **Self-attention** across tokens captures global correlations.

This is analogous to:

- **Computer vision**: patch embeddings in Vision Transformers.
- **Audio processing**: frame-level features in speech transformers.

Tokenization vs direct convolution

An alternative would be to use **convolutional neural networks (CNNs)**:

- CNNs have **fixed receptive fields** and locality bias.
- They process data with a hierarchical, local-to-global architecture.

Transformers with tokenization differ:

- **Global receptive field** from the first layer (via self-attention).
- **Permutation equivariance** (order matters only via positional encodings or conditioning).
- **Flexibility** to handle variable-length sequences by masking.

For Dingo-T1, the key advantage is:

- CNNs would require fixed detector and frequency structure.
- Transformers can **drop tokens** (masking) without architectural changes.

5.2 Why not simple positional encodings?

Standard language transformers use **positional encodings** (`sinusoidal` or learned) added to token embeddings to encode sequence order.
Here, the “position” is:

- continuous (frequency),
- non-uniform (multibanded),
- plus discrete detector identity.

Using standard positional encodings would require:

- separate encodings for $(f^{(k)})$,
- for $(f^{(k+1)})$,
- and for detector ID.

The authors found this unwieldy and less natural than direct conditioning.

5.3 Tokenizer architecture

Each 16-bin segment is mapped to a **1024-dimensional** embedding:

- Input: concatenation of:
  - real and imaginary parts of $(d_I^{(k)})$,
  - PSD values $(S_{n,I}^{(k)})$,
  - scalar $(f^{(k)}, f^{(k+1)})$,
  - one-hot detector identity.
- Network:
  - Fully-connected layers,
  - A 512-dimensional residual block,
  - Conditional inputs (frequency range and detector ID) injected via a **gated linear unit (GLU)**.

The tokenizer is **shared** across detectors:

- Same weights for H, L, V,
- But detector information is encoded via the input one-hot vector.

Output:

- Token embedding $(\in \mathbb{R}^{1024})$.

5.4 Summary token

In addition to the $(3K = 207)$ data tokens, a **learnable summary token** is appended :

- A parameter vector initialized randomly and learned during training.
- Plays the role of a “register” or [CLS] token: a dedicated location where global information can be aggregated via self-attention.

So, the full input to the transformer encoder is:

- $(n = 207 + 1 = 208)$ tokens, each in $(\mathbb{R}^{1024})$.
- Represented as a matrix $(X \in \mathbb{R}^{n \times d_{\text{model}}})$, with $(d_{\text{model}} = 1024)$.

---

6. Transformer Encoder: Mathematics and Intuition

6.1 High-level intuition

A transformer encoder layer processes the token sequence through:

1. Multi-head self-attention.

2. Feed-forward networks (FFN).
3. Residual connections and Layer Normalization (pre-LN).

Self-attention allows each token to “look at” all the others and aggregate information based on learned relevance scores. The summary token, in particular, attends to all data tokens and aggregates a compressed global representation.

6.2 Self-attention mathematically

Given input token embeddings $(X \in \mathbb{R}^{n\times d_{\text{model}}})$, we compute:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V,$$

with:

- $(W^Q, W^K, W^V \in \mathbb{R}^{d_{\text{model}}\times d_k})$
- $(Q, K, V \in \mathbb{R}^{n\times d_k})$.

For each pair of tokens (i, j):

- The raw attention score (before softmax) is:

$$s_{ij} = \frac{(QK^\top)_{ij}}{\sqrt{d_k}} + M_{ij},$$

  where (M_{ij}) is a mask term (0 or $(-\infty)$).
- We then apply softmax row-wise:

$$\alpha_{ij} = \frac{\exp(s_{ij})}{\sum_{j'} \exp(s_{ij'})},$$

  so that each row (i) yields attention weights across all tokens (j).
- The output of attention is:

$$\text{Attention}(Q, K, V) = \alpha V,$$

  giving a new sequence of token representations.

6.3 Mask (M) and missing data

The mask (M) is crucial:

- $(M_{ij} = 0)$ if token (j) is allowed to be attended to from token (i).
- $(M_{ij} = -\infty)$ if token (j) should be ignored (e.g. masked out segment or missing detector).

Because `softmax(-∞) = 0`, these positions effectively disappear from the attention distribution.

Thus:

- If a frequency band is missing (due to masking) or a detector is not present, its tokens are excluded.
- The transformer learns to compute a posterior from the remaining tokens only.

6.4 Multi-head attention

Instead of a single attention operation, we use **multi-head attention** with $(h = 16) $ heads:

Each head (k) has its own (W_k^Q, W_k^K, W_k^V), and computes:

$$\text{head}_k = \text{Attention}(Q_k, K_k, V_k).$$

The outputs are concatenated along the feature dimension and projected back to dimension (d_{model}):

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O.$$

This allows the model to attend to different aspects or subspaces of the token embeddings simultaneously (e.g. different frequency bands, detectors, or signal vs noise patterns).

6.5 Pre-Layer-Norm Transformer blocks

Each transformer block uses **pre-layer normalization**:

For each block:

1. Self-attention sublayer:

$$Y = X + \text{MultiHead}(\text{LN}(X)).$$

2. Feed-forward network (FFN) sublayer:

$$Z = Y + \text{FFN}(\text{LN}(Y)).$$

Where:

- LN is LayerNorm, applied across feature dimension.
- FFN is a two-layer MLP applied independently to each token:

$$\text{FFN}(x) = W_2\,\sigma(W_1 x + b_1) + b_2$$

  with hidden size 2048 and nonlinearity (\sigma).

Pre-LN improves training stability and allows higher learning rates without warm-up.

Dingo-T1 uses:

- $(N = 8)$ transformer layers,
- $(h = 16)$ heads,
- $(d_{\text{model}} = 1024)$,

- FFN hidden size of 2048.

6.6 Summary token as global representation

After 8 layers, we extract the final **summary token** from the sequence.
Call it ($s \in \mathbb{R}^{d_{\text{model}}}$) (the row of ($X$) corresponding to the summary token).

This vector has aggregated information from all unmasked tokens through self-attention. It is then linearly projected to a **128-dimensional context vector**:

$$c = W_c s + b_c, \quad c \in \mathbb{R}^{128}.$$

The authors observed that varying context size between 56 and 256 dimensions does not significantly affect training loss; 128 is sufficient.

6a. Amortization in Simulation-Based Inference

6a.1 What is amortized inference?

Traditional Bayesian inference:

- For each new observation d, run a sampler (MCMC, nested sampling) to compute $p(\theta \mid d)$.
- This requires many evaluations of the likelihood $p(d \mid \theta)$.
- Computational cost: **hours to days per event**.

Amortized inference:

- Train a neural network $q(\theta \mid d)$ on **simulated data**

$$(\theta, d) \sim p(\theta)p(d \mid \theta)$$

- After training, inference on a new observation d_{new} is a **single forward pass**.
- Computational cost: **seconds to minutes per event**.

The term "amortized" comes from economics:

- Upfront investment (training cost) is **amortized** over many uses (inference on many events).

6a.2 Amortization over data analysis settings

Standard amortized inference (e.g., in Dax et al. 2023) amortizes over:

- θ: parameters.
- d: data realizations.

But it does **not** amortize over:

- Different detector combinations.

- Different frequency ranges.
- Different PSDs (unless explicitly conditioned).

Dingo-T1's innovation:

- Amortizes over **analysis settings** by conditioning on S_n and training with **masks** m.
- The loss becomes:

$$\mathcal{L} = \mathbb{E}_{p(\theta)p(S_n)p(d|\theta,S_n)p(m)} \left[-\log q(\theta \mid m(d), m(S_n)) \right].$$

This means:

- The network learns

  $$q(\theta \mid d, S_n, \text{detectors}, f_{\text{min}}, f_{\text{max}})$$

  for **all configurations** in the training distribution $p(m)$.

6a.3 Why this is useful

Scientific flexibility:

- Analyze the same event under different assumptions (e.g., systematic studies over detectors).
- Perform robustness checks by varying frequency cutoffs.

Operational efficiency:

- Handle detector downtime without retraining.
- Adapt to calibration issues (PSD notching) on-the-fly.

Future scalability:

- As observing runs grow, retraining for each configuration becomes impractical.
- Amortized models can be updated incrementally (fine-tuning) rather than from scratch.

---

7. Conditional Normalizing Flow: Modeling the Posterior

7.1 Why normalizing flows?

We need a model $q(\theta \mid d, S_n)$ that:

- Outputs **samples** from a high-dimensional posterior over GW parameters.
- Also provides **densities** $q(\theta \mid d, S_n)$ for importance sampling.

Normalizing flows provide:

- An invertible mapping $f(\cdot;c)$ from a simple base distribution (e.g. standard Gaussian) to a complex distribution over θ.
- Exact computation of densities using change of variables.

Given:

- Base $z \sim p_z(z)$ (e.g. $\mathcal{N}(0, I)$).
- Flow $\theta = f(z;c)$.

The density is:

$$q(\theta \mid c) = p_z(z) \left| \det \frac{\partial f^{-1}(\theta;c)}{\partial \theta} \right|.$$

In practice, we compose multiple simple invertible layers to build a complex flow.

7.2 Rational-quadratic spline coupling transforms

Dingo-T1 uses a **conditional flow** based on rational-quadratic spline coupling layers as in Durkan et al. (Neural Spline Flows).

Key idea:

- Partition the parameter vector θ into two parts (θ_A, θ_B)
- Keep θ_A unchanged.
- Transform θ_B via a spline whose parameters depend on θ_A and the context c.

A coupling transform:

$$\theta'_A = \theta_A,$$

$$\theta'_B = g(\theta_B; \text{spline parameters}(\theta_A, c)).$$

The spline is piecewise rational-quadratic, ensuring:

- Invertibility,
- Continuity,
- Differentiability,
- Efficient computation of log-Jacobian.

By stacking many such layers with alternating partitions, we obtain a flexible, expressive flow.

The Dingo-T1 flow architecture is **identical** to that used in Dax et al. (2023), except that:

- Here it is conditioned on the transformer context vector c,
- And GNPE symmetries are not used.

7.3 Conditional posterior

The final density estimator is:

$$q(\theta \mid d, S_n) = \hat{q}_\phi(\theta \mid c),$$

where c is the 128-dim context from the transformer, and ϕ are flow parameters.

We can then draw samples:

1. Sample

$$z \sim \mathcal{N}(0, I)$$

2. Apply the conditional flow:

$$\theta = f(z; c)$$

   Or evaluate density

$$q(\theta \mid d, S_n)$$

   by inverting the flow and computing Jacobians.

---

8. Training Objective and Masking Strategy

8.1 Training objective

The model (tokenizer + transformer + flow, parameters $(\{\varphi, \phi\})$) is trained end-to-end to **maximize the likelihood** of true parameters under (q), i.e. minimize negative log-likelihood:

$$\mathcal{L} = \mathbb{E}_{p(\theta)p(S_n)p(d|\theta,S_n)p(m)} \left[-\log q(\theta \mid m(d), m(S_n))\right].$$

Here:

- $(p(\theta))$: training prior over parameters.
- $(p(S_n))$: distribution over PSDs (from O3).
- $(p(d \mid \theta, S_n))$: data-generating process (waveform + Gaussian noise).
- $(p(m))$: distribution over **token masks** (m).
- $(m(d), m(S_n))$: masked data and PSD (tokens removed).

In practice:

- For each training batch:
  - Sample (θ).
  - Generate (d).
  - Sample mask (m).
  - Apply mask to tokens by dropping them before the transformer.
  - Compute $(-\log q(\theta \mid m(d), m(S_n)))$.

    - Backpropagate gradients through the entire network.

8.2 Random masking vs data-based masking

Two masking strategies were compared:

(a) Random masking

- 40% of training samples are partially masked.
- For each masked sample:
    - Sample number of masked tokens ($n_{\text{masked}} \sim U[0, 181]$).
    - Randomly select that many tokens to drop.
- No structure: detectors and frequency bands are masked arbitrarily.

This trains robustness to missing tokens but does **not** reflect real analysis patterns.

(b) Data-based masking (used in Dingo-T1)

This strategy encodes realistic variations of analysis settings:

1. **Missing detectors**:
    - With 60% probability: mask none.
    - 30%: mask one detector.
    - 10%: mask two detectors.
    - Probability of masking each detector (H/L/V) is 30%/30%/40%, reflecting Virgo's larger downtime.
2. **Frequency range changes** (25% of training samples):
    - For those samples:
        - Mask lower frequencies in 10% of them (increasing ($f_{\min}$)).
        - Mask upper frequencies in 70% (decreasing ($f_{\max}$)).
        - Mask both ends in 20%.
    - New cutoffs drawn from:
        - ($f_{\text{min,new}} \sim U[20, 180]\,\text{Hz}$),
        - ($f_{\text{max,new}} \sim U[80, 1810]\,\text{Hz}$).
    - In 70% of these cases, the same cut is applied to all detectors (global frequency change).
    - If a frequency boundary falls inside a token, the **whole token** is removed.
3. **PSD notching** (10% of training samples):
    - Emulate narrow-band calibration issues (e.g. Virgo 46–51 Hz in O3b).
    - Draw notch width ($\Delta f_{\text{mask}} \sim U[0, 10]\,\text{Hz}$).
    - Lower bound uniform over frequency range.
    - Remove all tokens overlapping with notch band.

Average masking rates:

- Random masking: 18.0% of tokens masked.
- Data-based masking: 25.5% masked.

Empirically, data-based masking leads to **better performance** on real events, especially for 1- and 2-detector configurations.

---

9. Training Details

9.1 Optimizer and schedule

- Optimizer: **AdamW** with $(\beta_1 = 0.8), (\beta_2 = 0.99)$.
- Initial learning rate: 0.001.
- Weight decay: 0.005.
- Scheduler: ReduceLROnPlateau:
  - Halve the learning rate when validation loss plateaus for 10 epochs.
- Gradient accumulation used for Dingo-T1 due to memory limits.

9.2 Hardware and training time

- Hardware: 8 × NVIDIA A100-SXM4-80GB GPUs (CUDA 12.1).
- Distributed data-parallel training + mixed precision (AMP).
- Batch size: effective 16,384 samples per update.
- Training times (early stopped):

Model	Epochs	Training time
Dingo Baseline	273	3 days 1 hour
Dingo-T1 (data-based)	183	9 days 9 hours
Dingo-T1 (random)	219	11 days 17 hours

Given the scale, hyperparameter optimization was limited to small scans over LR and $((\beta_1, \beta_2))$.

9.3 Model sizes

Method	Embedding params	Flow params	Total params
Dingo baseline	22M	92M	114M
Dingo-T1	68M	92M	160M

The transformer-based encoder is significantly larger and more expressive than the residual network encoder of the baseline.

---

10. Inference and Importance Sampling

10.1 Inference workflow

For a real event:

1. Load observed strain $d_I(t)$ and estimate PSDs $S_{n,I}(f)$.
2. Apply multibanding to get $d_I^{\mathrm{mb}}, S_{n,I}^{\mathrm{mb}}$.
3. Tokenize into embeddings.
4. Construct mask based on:
   - Which detectors are present.
   - Event-specific $f_{\min}, f_{\max}$.
   - Any notched frequencies.
5. Run transformer encoder $\rightarrow$ context c
6. Sample 10^5 posterior samples from flow:

$$\theta_i \sim q(\cdot \mid c)$$

7. Optionally apply importance sampling (see below).

Runtime:

- Neural sampling: < 5 seconds.
- Phase reconstruction + IS: 5–10 minutes (64 CPU cores).
- Total: **< 10 minutes per event**.

10.2 Importance sampling (IS)

The neural posterior

$$q(\theta \mid d, S_n)$$

is trained on **multibanded** data and may have small approximation errors.

To correct for these:

- Compute importance weights:

$$w_i = \frac{p(\theta_i \mid d, S_n)}{q(\theta_i \mid d, S_n)} \propto \frac{p(\theta_i)p(d \mid \theta_i, S_n)}{q(\theta_i \mid d, S_n)},$$

  where

$$p(d \mid \theta, S_n)$$

is evaluated on the **uniform-frequency** grid (original data resolution).

- The **effective sample size** is:

$$\boxed{N_{\text{eff}} = \frac{(\sum_i w_i)^2}{\sum_i w_i^2}.}$$

- **Sample efficiency**:

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

  where

$$N = 10^5$$

  is the number of unweighted samples.

High efficiency (e.g., $\epsilon > 5\%$) means the neural posterior is close to the true posterior.

10.3 Why IS is necessary

Importance sampling serves two purposes:

1. **Corrects for multibanding approximation**:
    - The model is trained on compressed data.
    - IS reweights to the exact likelihood on full-resolution data.
2. **Corrects for flow approximation errors**:
    - Normalizing flows are powerful but not perfect.
    - IS ensures the final samples are drawn from the true posterior (up to sampling noise).

10.4 Comparison with GNPE

Group-equivariant NPE (GNPE):

- Encodes symmetries (e.g. time-translation), aligning merger times across detectors.
- Gives better sample efficiencies (and thus more accurate initial posteriors).
- But:
    - Requires an iterative Gibbs sampling scheme at inference time.
    - Breaks direct access to the posterior density; requires an additional unconditional flow to reconstruct density for importance sampling.

GNPE thus leads to:

- More complex and slower inference ($1 hour for 10^5 IS - corrected samples$).
- Less flexible experimentation (harder to quickly vary analysis settings).

Dingo-T1:

- Lower efficiency than GNPE but much higher than baseline NPE.
- Direct density access, simpler and faster inference $(\ 5\text{–}10 minutes)$.
- Much greater flexibility in detector/frequency configurations.

---

11. Results: Sample Efficiencies and Calibration

11.1 Simulated events

On 1000 simulated signals per detector configuration, over the full frequency range:

- Dingo-T1 (data-based masking):
  - 1-detector: median efficiency 26.9%.
  - 2-detector: 6.8%.
  - 3-detector (HLV): 3.3%.
- Dingo-T1 (random masking):
  - 1-detector: 15.7%.
  - 2-detector: 4.9%.
  - 3-detector: 3.4%.

This shows:

- **Data-based masking** improves 1- and 2-detector performance.
- Efficiency decreases with more detectors because posteriors are narrower and harder to approximate.

11.2 Real O3 events (48 events, 17 configurations)

Using official catalog data-analysis settings for GWTC-2.1 and GWTC-3:

- Dingo-T1 (data-based masking) across 48 events:
  - Median efficiency: 4.2%.
  - 2-detector events: 4.5% median.
  - 3-detector events: 4.2% median.
- Dingo-T1 (random masking):
  - Overall median: 2.5%.
  - 2-detector events: 1.2%.
  - 3-detector events: 2.7%.
- Dingo baseline (fixed HLV configuration, 30 events):
  - Median efficiency: 1.4%.

So Dingo-T1 improves median efficiency roughly **3× over the baseline** and about **2× over random masking**.

11.3 Calibration: P–P plots

P–P plots compare the empirical distribution of posterior percentile ranks of true parameters against the uniform distribution.

- For each parameter and configuration, we:
    1. Simulate many injections with known true (θ).
    2. Compute the posterior.
    3. Record the percentile of the true parameter within the marginal posterior.
    4. Plot CDF of these percentiles.
- If the model is perfectly calibrated, the CDF is a diagonal line (uniform distribution).
- The authors show P–P plots for all detector configurations and report that deviations from uniformity are small and consistent with statistical fluctuations (Kolmogorov–Smirnov tests at significance level $\$(\alpha = 0.05)$).

Thus, Dingo-T1 is **well calibrated** on simulated data.

11.4 Comparison with Bilby and LVK catalogs

Direct comparison between Dingo-T1 and LVK catalog posteriors is nontrivial because of differences in:

- PSD choices (BayesWave vs Welch),
- Waveform models and reference frequencies,
- Priors and mixing of multiple waveform runs in LVK catalogs.

To allow a fair comparison, the authors:

1. Run Bilby with the **same PSDs, waveform, priors, and window factor corrections** as Dingo-T1 for one event (GW190701_203306).
2. Compare Bilby and Dingo-T1 posteriors: they show good agreement, with some differences as expected from modeling approximations.
3. Overplot GWTC-2.1 posterior for illustration; it differs more due to different analysis settings.

---

12. Applications Enabled by Flexibility

12.1 Systematic studies over detector combinations

Because the same Dingo-T1 model can ingest **any subset of detectors** and frequency ranges, the authors analyze each event for all possible detector combinations:

- H, L, V individually.
- HL, HV, LV pairs.

- HLV triple.

This provides:

- Insights into which detectors drive constraints.
- Diagnostic information about poor performance cases (e.g. glitch contamination, PSD misestimation).

Example:

- GW200129_065458 shows low efficiency whenever Livingston is included, consistent with glitch subtraction issues in the official analysis.
- GW190517_055101 initially had 0% efficiency for Hanford-involving configurations; the culprit was a bad PSD estimate around 500 Hz. Correcting the PSD or masking that region increased efficiency to ~2.17%.

Such comprehensive studies were previously **computationally prohibitive** with standard samplers.

12.2 Inspiral–merger–ringdown (IMR) consistency tests

Dingo-T1 is used to perform GR tests by comparing final mass and spin inferred from the inspiral vs postinspiral parts.

Definitions:

- Inspiral data: $([f_{\min}, f_{\mathrm{cut}}])$.
- Postinspiral data: $([f_{\mathrm{cut}}, f_{\max}])$.
- For each, infer posterior on $((M_f, \chi_f))$.

Then compute **fractional deviations**:

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

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

GR predicts that both should be consistent with 0 for quasi-circular BBH mergers.

The authors:

- Analyze 7 events with specific inspiral/postinspiral ranges (Table III).
- For GW190630*185205, repeat with different $(f*{\text{cut}} \in {120, 130, 135, 140, 150},\mathrm{Hz})$ to test robustness.

Results:

- Posteriors on $((\Delta M_f/M_f, \Delta\chi_f/\chi_f))$ are consistent with (0,0).
- Variation with (f_{cut}) is modest and consistent with expected statistical scatter.

This demonstrates:

- Dingo-T1 can reconfigure frequency cuts **at inference time**.
- IMR consistency tests that used to be expensive can be done quickly and systematically.

---

13. Limitations and Future Directions

The authors highlight several directions:

1. **Signal duration and frequency resolution**
   - Currently, signal duration is fixed (8 s) and frequency resolution is fixed.
   - Next step: condition tokens on information about frequency resolution / duration, to amortize over variable lengths.
2. **Alternative data representations**
   - Time-domain or time–frequency representations where signal morphology may be simpler.
   - For time–frequency images, vision transformers (ViT) could be used.
3. **Waveform models and priors**
   - Dingo-T1 is trained on IMRPhenomXPHM only.
   - Future: amortize over multiple waveform families, uncertainties in waveform modeling, or over priors.
4. **LISA and data gaps**
   - For LISA (space-based observatory), long data gaps are expected.
   - Missing-token approach is naturally suited to handle these gaps.
5. **General-purpose GW backbone**
   - The Dingo-T1 transformer encoder (without the flow) could act as a **compression backbone** for multiple GW tasks: detection, classification, glitch analysis, etc.
   - This is analogous to using pre-trained transformers in NLP or vision.

---

14. Conceptual Summary

- The paper targets a real bottleneck in GW science: **flexible, scalable parameter estimation** that adapts to changing detectors and frequency ranges without retraining a new model for each configuration.
- The core solution is architectural and training-centric:
  - Represent multibanded GW data as sequences of **tokens** carrying local frequency

    segments and detector identity.
    - Use a **transformer encoder** with masked self-attention to aggregate information across detectors/frequencies and **learn to handle missing data**.
    - Condition a **normalizing flow** on the transformer summary to model the posterior over physical parameters.
- The model is trained with **data-based masking** to reflect realistic analysis configurations (missing detectors, different frequency cutoffs, PSD notches). This is essential for good performance on real data.
- With importance sampling, Dingo-T1 provides:
    - Good sample efficiency (~4.2% median on 48 O3 events).
    - Fast analysis (~5–10 minutes per event for 10^5 posterior samples with correction).
    - Enough accuracy and calibration to replace much slower traditional samplers in many use cases.
- The approach moves GW inference closer to **foundation models**:
  one pre-trained transformer-based encoder can be reused across tasks and configurations, with domain-specific heads (e.g. normalizing flows) on top.
