My problems. Phase 1

- Page 3: PSD used in the supplement.
- The noise is assumed to be **additive**, meaning it is independent of the signal. All noise contributions can therefore be linearly summed.

Noise Type	Mathematical Model	Characteristics
Additive	$d = h(\theta) + n$	Noise is independent of the signal amplitude. Typically assumed Gaussian: $n \sim \mathcal{N}(0, \sigma^2)$.
Multiplicative	$d = h(\theta)\,(1 + n)$	Noise depends on the signal amplitude (e.g., optical systems or calibration uncertainty).

- The real and imaginary parts of $n(f)$ are Gaussian distributed, with variance proportional to the noise power spectral density $S_n(f)$.
- Different frequencies are approximately independent.???

- In the text we have: This is mathematically equivalent to a chi-squared measure of mismatch between the data and the template waveform. this means:
- **It means that the mismatch between the observed data and a theoretical waveform template is quantified by a χ^2 statistic: the smaller the χ^2, the better the template matches the data. This is mathematically identical to the standard chi-squared measure used in statistics to compare data with a model when the noise is Gaussian.**
- Posterior depends on all analysis choices.

- **Amortization** (in machine learning / simulation-based inference) refers to using a single trained neural network to **approximate posterior distributions** $(q(\theta \mid d, S_n))$ for **any new data** (d) and **experimental settings** (S_n) — **without retraining**. This enables fast, scalable Bayesian inference by "amortizing" the computational cost of inference across many possible inputs.

In page 5: The noise is modeled as a zero-mean, stationary Gaussian process in the frequency domain:

$$\mathbb{E}[\,n(f)\,n^*(f')\,] = S_n(|f|)\,\delta(f - f')$$

Parameters:

- $n(f)$: Fourier transform of the noise at frequency f
- $n^*(f')$: Complex conjugate at frequency f'
- $\mathbb{E}[\cdot]$: Ensemble (statistical) expectation
- $S_n(|f|)$: Noise power spectral density (PSD) — variance per unit frequency
- $\delta(f - f')$: Dirac delta function — indicates noise is uncorrelated between different frequencies

---

In page 6:
In gravitational-wave data analysis, we describe noise using two related quantities:

- **Power Spectral Density (PSD)**: denoted $S_n(f)$, it tells us how much *noise power* exists at each frequency. Its units are **strain²/Hz**.
- **Amplitude Spectral Density (ASD)**: this is just the square root of the PSD, written as $\sqrt{S_n(f)}$. It has more intuitive units: **strain/√Hz**, and is often what's plotted in sensitivity curves (like for LIGO).

So, when people say "the noise is 10^{-23} strain/√Hz," they're referring to the **ASD**, not the PSD.

We usually plot the **Amplitude Spectral Density (ASD)** — $\sqrt{S_n(f)}$ — instead of the PSD because:

- **Human intuition**: Strain (a dimensionless measure of spacetime distortion) is what we actually care about. ASD shows noise *in the same units as the signal* (strain/√Hz), so you can directly compare a gravitational-wave signal's amplitude to the noise curve.
- **Visual clarity**: Gravitational-wave signals and detector sensitivity span many orders of magnitude. The square root compresses the scale, making features easier to see.
- **Convention**: Detectors like LIGO publish their sensitivity as ASD curves, so the community uses it for consistency—e.g., if a signal's strain amplitude is *above* the ASD curve, it's likely detectable.

For simulating the noise for training the Data: in page 8 we have Generating Noise:

What This Means (Step by Step):

You're describing a standard method to **simulate realistic detector noise** for gravitational-wave data analysis (e.g., for LIGO/Virgo O3 observing run). Here's what each part means:

1. "PSDs from a collection of Welch PSD estimates from actual O3 data"

Goal: Get a realistic model of the detector's noise spectrum during the O3 observing run.

How?

- Take many short segments of real O3 strain data recorded during times when no gravitational-wave signal was present (i.e., "quiet" background data).
- For each segment, compute its power spectral density (PSD) using Welch's method—a standard technique that averages periodograms over windowed segments to produce a low-variance estimate.
- Combine these individual PSD estimates (typically by taking the median or average across segments) to obtain a smooth, representative noise PSD, denoted $S_n(f)$, that characterizes the typical noise level of the detector during O3.

Why Welch? It provides a stable and robust estimate of the noise spectrum even when the underlying data has mild non-stationarities.

2. "Add stationary Gaussian noise with that PSD to the signal"

Goal: Create a synthetic data stream that mimics real detector output: `data = signal + realistic noise`.

How?

- Generate stationary Gaussian noise in the frequency domain whose statistical properties match the measured $S_n(f)$:
  - For each frequency bin f, draw a complex random number $n(f)$ such that:
    - $\mathbb{E}[n(f)] = 0$
    - $\mathbb{E}[n(f)n^*(f')] = S_n(f)\,\delta(f - f')$
  - Enforce Hermitian symmetry ($n(-f) = n^*(f)$) so that the inverse Fourier transform yields a real-valued time series.
- Apply an inverse Fourier transform to obtain time-domain noise $n(t)$.
- Add this noise to your simulated gravitational-wave signal $h(t)$:

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

Note: Real detector noise is not perfectly stationary or Gaussian, but this approximation is widely used because it captures the dominant statistical behavior and enables tractable analysis.

Why Do This?

- To test data analysis pipelines (e.g., matched filtering, Bayesian inference) under realistic noise conditions.
- To generate training or validation datasets for machine learning models intended for use on real LIGO/Virgo data.

- To ensure that statistical forecasts (e.g., Fisher matrix predictions) reflect the actual sensitivity of the instrument during the O3 observing run.

This approach combines idealized theoretical waveforms with empirically measured noise characteristics, making simulations more faithful to real observations.

In page 9 for sampling rate:

What is the Nyquist Frequency ($f_s/2$)?

The **Nyquist frequency** is a fundamental concept in digital signal processing. It comes from the **Nyquist–Shannon sampling theorem**, which states:

> To perfectly reconstruct a continuous signal from its discrete samples, the signal must contain **no frequency components higher than half the sampling rate**.

Definitions

- f_s: **Sampling rate** (or sampling frequency), measured in Hz (samples per second).
  Example: LIGO data is typically sampled at $f_s = 4096$ Hz → 4096 data points per second.
- **Nyquist frequency**: $f_{\text{Nyquist}} = \frac{f_s}{2}$
  This is the **maximum frequency** that can be reliably represented in a discretely sampled signal.

Why $f_s/2$?

When you sample a continuous waveform at regular intervals $\Delta t = 1/f_s$, any oscillation faster than once every **two samples** becomes ambiguous.

For example:

- If $f_s = 1000$ Hz → $\Delta t = 1$ ms.
- A sine wave at 600 Hz would complete more than half a cycle between samples.
- But due to aliasing, it would appear indistinguishable from a lower-frequency wave (e.g., 400 Hz).

This phenomenon is called **aliasing**: high-frequency signals "fold back" and masquerade as lower frequencies.

To avoid this, real instruments apply an **anti-aliasing filter** before sampling, removing all frequency content above $f_s/2$.

Connection to Gravitational-Wave Analysis

In your context:

- The **upper frequency cutoff** $f_{\max}$ of your analysis cannot exceed the Nyquist frequency.
- Even if a gravitational-wave signal (e.g., from a light binary) contains power above $f_s/2$, **you cannot observe it** in the sampled data — it's either removed by the anti-aliasing filter or corrupted by aliasing.
- Therefore, $f_{\max} \leq f_s/2$ is a hard limit imposed by the data acquisition system.

Example:
LIGO O3 data uses $f_s = 4096$ Hz → $f_{\text{Nyquist}} = 2048$ Hz.
So no analysis can meaningfully include frequencies above 2048 Hz, regardless of the astrophysical signal.

Summary

- $f_s/2$ is not arbitrary — it's the **theoretical upper limit** for faithful frequency representation in discrete data.
- It arises from the mathematics of sampling and is enforced in all digital instruments.
- In gravitational-wave astronomy, your choice of $f_{\max}$ must respect this limit, even if the signal morphology suggests higher frequencies.

In page 10: for the fixed signal Duration part:

What Does “Fixed Signal Duration (T = 8 s)” Mean?

In gravitational-wave data analysis (e.g., for LIGO), we often **extract a short segment of data** around the time a signal is expected (or injected).
Here, **T = 8 seconds** means:

> The analyzed waveform (or data chunk) is exactly **8 seconds long** in the time domain.

This is a common choice because:

- Compact binary coalescences (like black hole mergers) spend only a few seconds in LIGO's sensitive band (20–2000 Hz).
- It balances computational cost and physical relevance.

How Is Frequency Resolution (Δf) Determined?

When you take a **Fourier transform** of a time series of duration **T**, the resulting frequency series is **discrete**, with spacing:

$$\Delta f = \frac{1}{T}$$

- This is a fundamental property of the Fourier transform: longer signals → finer frequency resolution.

- For T = 8 s:
  $\Delta f = \frac{1}{8} = 0.125$ Hz

So the frequency bins are: 0 Hz, 0.125 Hz, 0.25 Hz, 0.375 Hz, ..., up to the Nyquist frequency.

How Many Frequency Bins Are There?

The number of bins between a minimum frequency f_{min} and maximum frequency f_{max} is approximately:

$$N_{\text{bins}} \approx \frac{f_{\text{max}} - f_{\text{min}}}{\Delta f}$$

Given:

- $f_{\text{min}} = 20$ Hz
- $f_{\text{max}} \approx 1792$ Hz
- $\Delta f = 0.125$ Hz

Then:

$$N_{\text{bins}} \approx \frac{1792 - 20}{0.125} = \frac{1772}{0.125} \approx 14{,}176$$

→ **Over 14,000 frequency bins!**

This is what “raw frequency series would have thousands of bins” means.

Why Does This Matter?

- **Computational cost**: Evaluating likelihoods or filters over 14k+ bins is expensive.
- **Redundancy**: Many adjacent bins carry highly correlated information.
- **Practical workaround**: Analysts often **downsample** or **bin** the frequency series (e.g., using coarse grids or interpolation) to reduce dimensionality while preserving accuracy.

Summary

- **T = 8 s**: The signal (or data segment) lasts 8 seconds in time.
- $\Delta f = 1/T = 0.125$ **Hz**: The spacing between frequency bins in the Fourier domain.
- $f_{\text{min}} = 20$ **Hz,** $f_{\text{max}} \approx 1792$ **Hz**: The physical and instrumental limits of the analysis band.
- **Thousands of bins**: Result from high frequency resolution (small Δf) over a wide band — a direct consequence of using an 8-second time window.

In page 10-4.3-Problem:
The statement means:

> On a **uniform frequency grid** (e.g., bins every $\Delta f = 0.125$ Hz from 20 Hz to 1792 Hz), the resulting data vector is **high-dimensional** (thousands of bins) and **redundant** in regions where the gravitational-wave signal varies slowly (i.e., is smooth).

Why?

1. **High-dimensional**:
   With fine resolution ($\Delta f = 1/T = 0.125$ Hz for $T = 8$ s) over a wide band, you get $\sim$ 14,000 frequency bins. This leads to large vectors and expensive computations.
2. **Redundant where the signal is smooth**:
   - In frequency regions where the signal $h(f)$ changes **gradually** (e.g., during the inspiral phase of a binary), adjacent bins contain nearly identical information.
   - For example, if $h(f)$ is almost constant over 10 consecutive bins, storing all 10 values is unnecessary — one or two would suffice to capture the shape.
   - This redundancy offers no significant gain in accuracy but increases memory and computational cost.

Practical Implication:

Instead of using all bins, analysts often:

- Use **non-uniform (adaptive) grids** that place more points where the signal changes rapidly (e.g., near merger) and fewer where it's smooth.
- Apply **frequency binning**, interpolation, or compression techniques to reduce dimensionality while preserving signal fidelity.

Thus, a uniform grid is **inefficient** for smooth signals: it wastes resources on highly correlated, low-information samples.

Each token contains 16 bins from the multibanded grid.

In Tokenization we have 3K segment. why?
The factor **3K** comes from the fact that there are **three detectors**, and **each detector contributes K segments**.

Breakdown:

- The set of detectors is:
  $I \in \{\mathrm{H}, \mathrm{L}, \mathrm{V}\}$
  → H = LIGO Hanford, L = LIGO Livingston, V = Virgo
  → **3 detectors total**

- For **each detector**, the multibanded frequency grid is split into **K segments** (here, $K = 69$).
  Each segment corresponds to a contiguous block of 16 frequency bins.
- Therefore:
  - Detector H → K segments
  - Detector L → K segments
  - Detector V → K segments
- Total number of segments across all detectors:
  $K + K + K = 3K$

So, **3K = 3 × 69 = 207 segments in total**.

Each of these 207 segments contains:

- A complex strain vector $d_I^{(k)} \in \mathbb{C}^{16}$
- A real PSD vector $S_{n,I}^{(k)} \in \mathbb{R}^{16}$

Thus, **3K** simply reflects the combination of **K frequency segments per detector × 3 detectors**.

Self attention in page 13:

What Does "Self-Attention Across Tokens" Mean Here?

In this context, **tokens** are small segments (or "patches") of the gravitational-wave signal in the frequency domain — each token contains 16 consecutive frequency bins of the strain data (and possibly the noise PSD).

Now, **self-attention** is a mechanism from transformer neural networks that allows the model to **dynamically weigh how much each token should "pay attention to" every other token**, based on their content.

How It Works for GW Signals:

1. **Local information (within a token):**
   Each token captures fine-grained, local structure — e.g., the smooth phase evolution of the waveform over a narrow frequency band (like 50–52 Hz). This is encoded in the 16 complex strain values.
2. **Global information (across tokens):**
   The full chirp signal spans many tokens — from low frequencies (early inspiral) to high frequencies (merger). The *relationship* between tokens carries physical meaning:
   - The rate at which frequency increases (chirp) depends on the binary's **chirp mass**.
   - Amplitude evolution encodes **distance** and **inclination**.
   - Modulations encode **spins** or **precession**.

3. **Self-attention connects them:**
   The self-attention layer computes correlations between all pairs of tokens. For example:

   - A token at 30 Hz might "attend to" a token at 200 Hz because their phase relationship matches what's expected for a binary with a certain mass.
   - The model learns to recognize consistent global patterns (e.g., a specific chirp trajectory) by linking local pieces.

   Mathematically, for tokens $x_1, x_2, \ldots, x_K$, self-attention computes:

   $$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$$

   where Q, K, V are projections of the tokens. This creates a **weighted sum of all tokens** for each output, with weights learned from data.

Why This Is Powerful for GWs:

- Unlike fixed methods (e.g., matched filtering with templates), self-attention **adapts** to the signal's structure.
- It can capture **long-range dependencies**: the low-frequency inspiral and high-frequency merger are physically linked, and self-attention can model that link even if they're far apart in frequency.
- It enables **end-to-end learning** of both local waveform features and global parameter encoding — without hand-crafted features.

In short:

> **Tokens** = local building blocks;
> **Self-attention** = learnable glue that assembles them into a coherent global picture of the source physics.

Page 18: amortization and dingo

What Does "Amortization Over Data Analysis Settings" Mean?

In standard **amortized inference** (e.g., Dax et al. 2023), a neural network is trained to approximate the posterior distribution $q(\theta \mid d)$ — that is, it learns to infer physical parameters θ (like masses or spins) from gravitational-wave data d.

However, this standard approach assumes **fixed analysis conditions**, such as:

- A fixed set of detectors (e.g., always H+L),
- A fixed frequency range (e.g., always 20–1024 Hz),
- A fixed noise spectrum (or one that's not explicitly varied during training).

As a result, if you change any of these settings — say, switch from two detectors to three, or analyze a different frequency band — you'd need to **retrain the network**.

Dingo-T1's Innovation

Dingo-T1 generalizes amortized inference by **also amortizing over analysis settings**. It does this by:

1. **Conditioning on the noise PSD** $S_n(f)$ — so the network sees how noisy each frequency bin is.
2. **Using binary masks** m to indicate which parts of the data are active:
   - $m(d)$ zeros out frequencies or detectors not used in a given analysis.
   - $m(S_n)$ applies the same masking to the PSD.
   - The mask encodes choices like:
     • Which detectors are included (H, L, V, or any subset),
     • What $f_{\min}$ and $f_{\max}$ are (by masking bins outside the range).
3. **Training with a loss that averages over all these configurations**:

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

   This means the network is exposed during training to **many combinations** of:
   - signals (θ),
   - noise realizations (S_n),
   - data (d),
   - analysis settings (via m).

What the Network Learns

After training, the model can infer:

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

for any configuration that was represented in the training distribution $p(m)$ — **without retraining**.

For example:

- You can give it data from only Hanford and Livingston → it works.
- You can restrict the analysis to 30–500 Hz → it adapts.
- You can use a PSD from a different observing run → it still works (as long as similar PSDs were seen during training).

Why This Matters

- **Flexibility**: One model supports many analysis scenarios.
- **Efficiency**: No need to train separate networks for every detector combination or frequency band.

- **Realism**: Matches how real analyses are done — settings change based on data quality, source type, or scientific goal.

In short:

> Dingo-T1 doesn't just learn "what θ produced this signal?" — it also learns "how to interpret the signal **given how I'm choosing to look at it**."

More about Normalizing Flow:

What Is a Normalizing Flow?

A **normalizing flow** is a type of deep generative model that learns to represent a complex probability distribution (like a posterior $q(\theta \mid d, S_n)$) by transforming a simple base distribution (like a standard Gaussian) through a series of invertible, differentiable functions.

Basic Idea:

- Start with a simple random variable $z \sim p_z(z)$ (e.g., $z \in \mathbb{R}^D$, $p_z = \mathcal{N}(0, I)$).
- Apply a sequence of invertible transformations:
  $\theta = f_K \circ f_{K-1} \circ \cdots \circ f_1(z)$
- The resulting variable θ follows a more complex distribution $q(\theta)$.
- Because each f_i is invertible and differentiable, we can compute the exact probability density $q(\theta)$ using the **change-of-variables formula**:

$$q(\theta) = p_z(z) \cdot \left| \det \frac{\partial f^{-1}}{\partial \theta} \right| = p_z(z) \cdot \prod_{i=1}^{K} \left| \det \frac{\partial f_i^{-1}}{\partial x_i} \right|$$

This gives both:

1. **Sampling**: draw $z \sim p_z$, then compute $\theta = f(z)$ → you get samples from $q(\theta)$.
2. **Density evaluation**: for any θ, compute how probable it is under q — i.e., evaluate $q(\theta)$ numerically.

What Does $q(\theta \mid d, S_n)$ Mean?

- $q(\theta \mid d, S_n)$ is an **approximation to the true Bayesian posterior** $p(\theta \mid d)$.
- It represents **our updated belief** about the physical parameters θ (e.g., masses, spins, distance) after observing data d, given knowledge of the noise characteristics S_n.
- The function q is a **probability density function (PDF)** over θ:
  - Higher $q(\theta \mid d, S_n)$ → more plausible θ given the data.
  - Lower $q(\theta \mid d, S_n)$ → less plausible.

In gravitational-wave inference, this posterior is often **multi-modal**, **correlated**, and **high-dimensional** (8+ parameters), so we need a flexible model like a normalizing flow to capture its shape.

Why Use Normalizing Flows for GW Posterior Modeling?

Because they provide **both**:

1. **Fast sampling**: generate thousands of posterior samples in milliseconds (useful for visualization or downstream tasks).
2. **Exact density evaluation**: compute $q(\theta \mid d, S_n)$ for any θ, which enables:
   - **Importance sampling** (e.g., to reweight samples under a different prior),
   - **Bayes factor estimation**,
   - **Training via maximum likelihood** (since you can compute $\log q(\theta \mid d, S_n)$).

Other methods like MCMC give samples but not densities; GANs give samples but no reliable densities. Normalizing flows give **both** — making them ideal for amortized Bayesian inference in GW astronomy.

Summary

- **Normalizing flow**: a neural network that transforms simple noise into complex distributions, while allowing exact density computation.
- $q(\theta \mid d, S_n)$: a learned approximation of the posterior — a full probability density over source parameters.
- **Key advantage**: you can both *sample* from it and *evaluate* how probable any parameter set is — essential for rigorous statistical inference.

What Is Importance Sampling (IS) and Why Is It Used Here?

Importance sampling (IS) is a statistical technique to **correct biases** in an approximate probability distribution by reweighting samples drawn from it.

In this context:

- You have a **neural posterior** $q(\theta \mid d, S_n)$ — a fast but slightly imperfect approximation of the true Bayesian posterior $p(\theta \mid d, S_n)$.
- You want to obtain **unbiased estimates** (e.g., credible intervals, means) as if you had sampled from the true posterior.

IS lets you do that **without running expensive MCMC**.

How It Works

1. **Draw samples from the neural posterior**:
   $\theta_i \sim q(\theta \mid d, S_n)$ for $i = 1, \ldots, N$
   (e.g., $N = 10^5$ samples — fast because the flow is invertible).
2. **Compute importance weights**:

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

   - $p(\theta_i)$: prior probability of θ_i
   - $p(d \mid \theta_i, S_n)$: **exact likelihood**, computed on the **full-resolution uniform frequency grid** (not the multibanded tokenized version used during training)
   - $q(\theta_i \mid d, S_n)$: density given by the neural flow (which was trained on the compressed/tokenized data)

   → The weight w_i measures how much more (or less) probable θ_i is under the **true posterior** compared to the **approximate posterior**.
3. **Use weighted samples**:
   Any expectation under the true posterior can be approximated as:

$$\mathbb{E}_{p(\theta|d)}[f(\theta)] \approx \frac{\sum_{i=1}^{N} w_i f(\theta_i)}{\sum_{i=1}^{N} w_i}$$

How Good Is the Approximation? → Effective Sample Size (ESS)

Not all weighted samples are equally useful. If a few samples have huge weights and the rest have near-zero weights, the estimate is unreliable.

The **effective sample size** quantifies this:

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

- $N_{\text{eff}} = N$: perfect match between q and p (all weights equal).
- $N_{\text{eff}} \ll N$: poor approximation; only a few samples matter.

Then, **sampling efficiency** is defined as:

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

- Example: if $N = 10^5$ and $\epsilon = 5\%$, then $N_{\text{eff}} = 5{,}000$ — enough for robust inference.
- If $\epsilon < 1\%$, the neural posterior is too inaccurate, and IS becomes unstable.

Why Evaluate the Likelihood on the Uniform Grid?

The neural network was trained on a **compressed representation** (multibanded tokens), which discards some high-resolution information.

But the **true likelihood** $p(d \mid \theta, S_n)$ must be computed using the **original, full-resolution data** (uniform frequency grid with $\Delta f = 0.125$ Hz) to avoid bias.

So IS acts as a **refinement step**:

> "Fast inference with tokens → correct small errors using exact physics."

Summary

- **Goal**: Fix small inaccuracies in the neural posterior q.
- **Method**: Importance sampling — reweight samples using the exact posterior density.
- **Key requirement**: Ability to evaluate both $q(\theta \mid d, S_n)$ (from the flow) and $p(d \mid \theta, S_n)$ (from physical waveform models).
- **Diagnostic**: Sampling efficiency $\epsilon = N_{\text{eff}}/N$. High ϵ → q is a good approximation.
- **Outcome**: Unbiased, high-fidelity posterior estimates at low computational cost.

More about masking:

What Is the Sample Mask (m)?

In this context, the **sample mask** m is a **binary vector** (or tensor) that indicates **which parts of the input data are valid or active** for a given analysis configuration. It is used to make a single neural network handle **many different analysis settings** (e.g., different detectors, frequency ranges) without retraining.

How Is It Constructed?

- The full multibanded frequency grid is divided into K segments (tokens), e.g., $K = 69$ per detector.
- For a specific analysis, you might:
  - Use only **some detectors** (e.g., Hanford + Livingston, but not Virgo),
  - Analyze only a **subset of frequencies** (e.g., 30–500 Hz, not the full 20–1792 Hz).

The mask m encodes these choices:

- Each token gets a value:
  $m_k = 1$ if token k is **included** in the current analysis,
  $m_k = 0$ if it is **excluded** (e.g., outside $[f_{\min}, f_{\max}]$ or from an unused detector).

So m has length $3K$ (for 3 detectors × K tokens), with 1s and 0s marking active/inactive tokens.

How Is the Mask Applied?

> "Apply mask to tokens by dropping them before the transformer."

This means:

- Before feeding the tokens into the transformer model, **all tokens where $m_k = 0$ are removed** (i.e., “dropped”).
- Only the **unmasked tokens** ($m_k = 1$) are passed to the network.

This is different from setting masked values to zero — here, the **sequence length itself changes**, and the transformer only processes the relevant subset.

Example:

- Full token sequence: [H_1, H_2, ..., H_{69}, L_1, ..., L_{69}, V_1, ..., V_{69}] → 207 tokens.
- If you analyze only Hanford and Livingston in 50–400 Hz, maybe only tokens H_{10}–H_{40} and L_{10}–L_{40} are kept.
- Mask sets all others to 0, and those tokens are **discarded**.
- Transformer receives only ~60 tokens instead of 207.

Why Use This Approach?

1. **Flexibility**: One model supports arbitrary detector combinations and frequency bands.
2. **Efficiency**: The transformer doesn’t waste computation on irrelevant or missing data.
3. **Correct conditioning**: The network learns that “not seeing Virgo data” is different from “seeing Virgo data with zero signal.”
4. **Training realism**: During training, random masks $m \sim p(m)$ simulate all possible analysis scenarios, so the model generalizes.

Summary

- **Mask m**: A binary indicator of which tokens (frequency segments from which detectors) are used in a given analysis.
- **“Dropping” masked tokens**: Physically removing inactive tokens from the input sequence before the transformer.
- **Purpose**: Enables a single neural posterior $q(\theta \mid d, S_n, m)$ to work across diverse observational setups — the core idea behind amortization over analysis settings.

My problems, Phase 2

Backpropagation (backprop) is the algorithm used to **train neural networks** by efficiently computing how much each parameter (weight) contributed to the error, so it can be updated to reduce that error.

In simple terms:

- You run the network forward → get a prediction.
- Compare prediction to true answer → compute loss (error).
- Then **backpropagate**: send the error backward through the network, layer by layer, using the chain rule of calculus.
- This gives the **gradient** (slope) of the loss with respect to every weight.
- Finally, weights are updated (e.g., via gradient descent) to make the network more accurate next time.

Without backprop, training deep networks would be extremely slow or impossible.

In page 5: By training with masks
that simulate missing detectors and frequency bands, the transformer learns to marginalize over missing information.

What Does "Marginalize Over Missing Information" Mean?

In Bayesian inference, **marginalization** means accounting for uncertainty by *averaging over unknown or missing variables* instead of ignoring them.

When the transformer is trained with **random masks** that:

- Remove data from certain detectors (e.g., no Virgo),
- Exclude parts of the frequency band (e.g., mask out 500–1000 Hz),

it learns to **infer parameters correctly even when some data is absent** — not by guessing the missing data, but by **integrating over all plausible possibilities** consistent with what *is* observed.

For example:
If only Hanford and Livingston data are provided, the model doesn't assume Virgo's signal was zero. Instead, it produces a posterior that reflects uncertainty due to Virgo's absence — just as a proper Bayesian analysis would.

Thus, **training with masks teaches the network to behave like a robust Bayesian estimator that automatically marginalizes over unobserved data channels**.

Important fact about Dingo T1:

DingoT1 use transformer as an encoder -> Token -> Context -> Estimation
DingoT1 use Normalizing Flow as and Decoder -> Context -> Posterior distribution -> generation

---

About Self Attention:

- Q, K, V all comes from the same input X
- Each token attends to all other tokens (including itself).

So in Dingo T1: Self-attention allows each segment to "ask": "Which other segments (frequencies,
detectors) are most relevant for understanding this signal?"

Consider two tokens:

- Token A: Low-frequency segment (20-40 Hz) from Hanford.
- Token B: High-frequency segment (200-300 Hz) from Livingston.
  If the true source is a chirping binary:
- Token A's query might "look for" corresponding high-frequency content that matches the chirp evolution.
- The attention score α AB will be high if Token B's key indicates it contains the expected merger-frequency content.
- Token A's output aggregates information from Token B (and others) weighted by relevance.

But when we talk about **Multi-Head Attention** the number of inputs is not only one (X) but multiple inputs (h)
Then all (h) concat with each other:

What Does "Concat" Mean Here?

In the Multi-Head Attention mechanism, **"Concat"** means **joining the outputs of all attention heads side by side along the feature (embedding) dimension**.

Step-by-step:

1. Suppose you have:
   - `h` attention heads,
   - Each head produces an output matrix of shape `(sequence_length, d_v)`, where `d_v` is the feature size per head.
2. After computing all heads:
   - `head¹ ∈ ℝ^(L × d_v)`
   - `head² ∈ ℝ^(L × d_v)`
   - …
   - `headʰ ∈ ℝ^(L × d_v)`
3. **Concatenation** stacks them horizontally (along the feature axis):

$$\text{Concat}(\text{head}^1, \ldots, \text{head}^h) \in \mathbb{R}^{L\times(h\cdot d_v)}$$

4. This big concatenated matrix is then multiplied by a **learnable projection matrix** W^O (of shape $(h \cdot d_v) \times d_{\text{model}}$) to map it back to the model's hidden dimension:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}^1, \ldots, \text{head}^h)\, W^O$$

Why Concatenate?

- Each head learns to attend to different parts of the input (e.g., different frequency correlations in GW data).
- Concatenation **preserves all these diverse representations** before fusing them via the final linear projection.
- This allows the model to combine complementary information from multiple "views" of the data.

> In short: **Concat = glue all head outputs together into one wide vector per token, then project to standard size.**

So the result?

Each head can specialize in different frequency ranges, detectors, or aspects of the signal/noise structure.

- Dingo T1 uses 16 16 parallel attention mechanisms within each transformer layer. so the world multi doesn't refer to multi detectors.
- Multi-Head Capture both local (within-detector) and global (cross-detector) patterns.