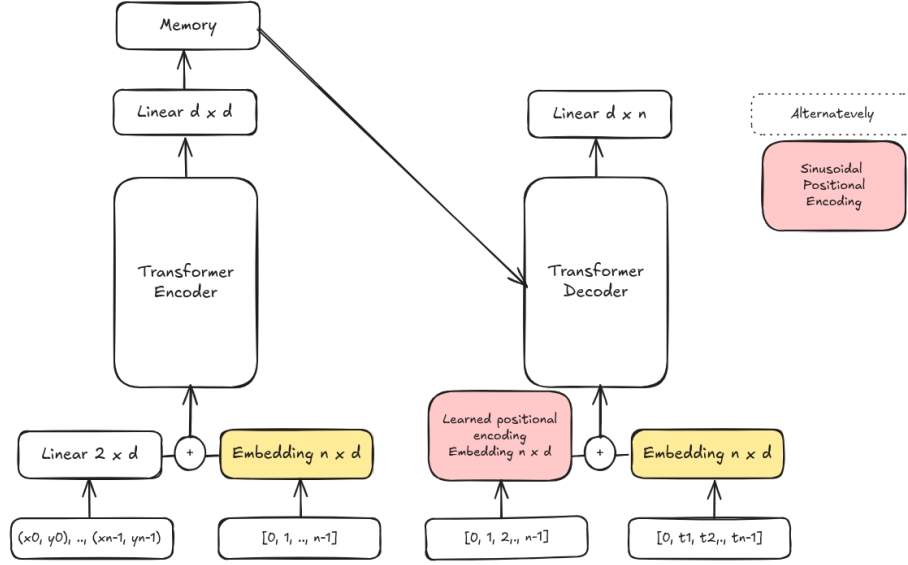


A proposed arcitecture and some training suggestion



This figure illustrates the architecture we adopt to solve the TSP. Overall, the architecture is a standard encoder–decoder Transformer, but with some modifications

- Feature expansion via a linear layer: Each 2D coordinate (x_i, y_i) is first projected to a d -dimensional space through a linear layer. Two features were not enough for learning.
- Shared node embeddings: We use a learnable embedding of size $n \times d$, representing node identities $\{0, 1, \dots, n-1\}$. The same embedding matrix is used in both the encoder and the decoder. **Exactly the same.**
- Optional linear map after the encoder: We include a linear layer on top of the encoder output. This is useful if one wants to use different embedding dimensions in the encoder and decoder. (In our experiments we kept both dimensions equal)
- Positional encoding in the decoder: Since the decoder generates the tour autoregressively, it needs positional information. We allow two alternatives:
 1. Sinusoidal positional encoding, as introduced in [2].

2. Learned positional encoding, implemented via `nn.Embedding`.

In practice, we observe that learned positional encodings train roughly $4\times$ faster.

Additional implementation details:

- Optimizer: We use AdamW, which is known to work better than Stochastic Gradient Descent in Transformer-based architectures [3].
- Weight initialization: We follow the standard initialization used in large language models [1]
- Loss function: We use standard cross-entropy over the predicted next node.

Our hyperparameters follow the notation of [2] and are summarized below:

Hyperparameter	Value	Comments
d_{model}	128	Sufficient as found empirically
d_{ff}	$4d_{\text{model}}$	As suggested in [2]
h Encoder	8	Standard choice
h Decoder	8	Standard choice
N_{enc}	4	Empirically effective
N_{dec}	4	Empirically effective
Learning rate	10^{-4}	You may add a warm-up or scheduler

References

- [1] ApX Machine Learning. *Transformer Weight Initialization*. 2025. URL: <https://apxml.com/courses/foundations-transformers-architecture/chapter-7-implementation-details-optimization/transformer-weight-initialization> (visited on 11/07/2025).
- [2] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [3] Yushun Zhang et al. “Why transformers need adam: A hessian perspective”. In: *Advances in neural information processing systems* 37 (2024), pp. 131786–131823.