In the original Transformer (Vaswani et al., 2017), each input token is represented as a 512-dimensional vector. This isn't arbitrary — it's a design choice based on model capacity and performance:

Reasons for 512D:

1. Representation Power:
   Higher dimensions can represent more complex patterns, relationships, and nuances in the data. For language, where tokens have subtle and deep contextual meanings, 512D helps capture these.
2. Multi-Head Attention:
   In the original Transformer, 512D is split across 8 attention heads, each with 64D. So:
   `512 = 8 heads × 64 dimensions/head`.
3. Depth of Encoding:
   Lower-dimensional embeddings like 10D might lose important information or compress too much, leading to poor model performance on complex tasks.
4. Empirical Results:
   Through experimentation, researchers found that models perform better with higher-dimensional embeddings — not just for language, but for images, code, and audio as well.

Then why not just use 10D?

You can, but with limitations:

- It works only for very simple tasks or toy datasets.
- The model will struggle to learn rich patterns.
- Think of it like compressing HD video into a tiny 10px image — you lose too much detail.

□ Example:

If you’re training a mini-Transformer for an educational demo or a small dataset, you might get away with 10D or 32D. But for real-world LLMs like BERT, GPT, etc., high dimensions like 512, 768, 1024, or 2048 are necessary.

TL;DR

You can try using 10D, but 512D is chosen to:

- Boost representational capacity
- Match attention head design
- Preserve semantic richness
- Improve learning in complex tasks

Related posts:

1. Transformer Architecture in LLM

2. Input Embedding in Transformers
3. Positional Encoding in Transformers
4. Multi-Head Attention in Transformers
5. Self Attention in Transformer