# Embeddings

Each input is initialised with random embeddings eg 4096 dimension for each neuron - and after passing through all layers these embeddings get new value - so original 4096 dim is maintained without duplication in each layer.

Output of one layer is again an refined embeddings.

So in training process these embeddings are stored temporarily for backpropogation to get gradients and adjust the weights and once weights are set these intermediate layers are discarded.

# Attention mechanism Grant Sanderson class

Embeddings get updated by the word sitting next to and before it to get some meaning, tone , context and so on.

How it’s done mathematically 🡪 Querry and key

Each word asks a query – and there is cor

# Why Good quality training set trains the model accurately to the meaning than large set of irrelevant dataset.

* As the attention mechanism takes the sequence for training and not every word in isolation, so if the input sequence in training is better at teaching the model more knowledge then it makes sense.

# DeepDive in to attention layer math and intuition of the math till it makes sense-

#### ****A) Multi-Head Self-Attention****

1. **Query, Key, and Value Vectors**:
   * Each token embedding is linearly transformed into three vectors:
     + Query Q=XWQQ = XW\_QQ=XWQ​
     + Key K=XWKK = XW\_KK=XWK​
     + Value V=XWVV = XW\_VV=XWV​
   * WQ,WK,WVW\_Q, W\_K, W\_VWQ​,WK​,WV​ are learned weight matrices.
2. **Attention Scores**:
   * Attention scores are computed by taking the dot product of QQQ and KKK, scaled, and passed through a softmax: Attention(Q,K,V)=Softmax(QKTdk)V\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d\_k}}\right)VAttention(Q,K,V)=Softmax(dk​​QKT​)V