## horizontal line

Self Attention

09.06.2025

# Transformer

When Attention mechanism is introduced the main problem with them is the ingoing sequential order of input. If we have 30,000 words it would take a very long time to process and also the issue of Transfer learning i.e. models cannot be trained like that they can be fine tuned for future instead they need to be trained from scratch always.

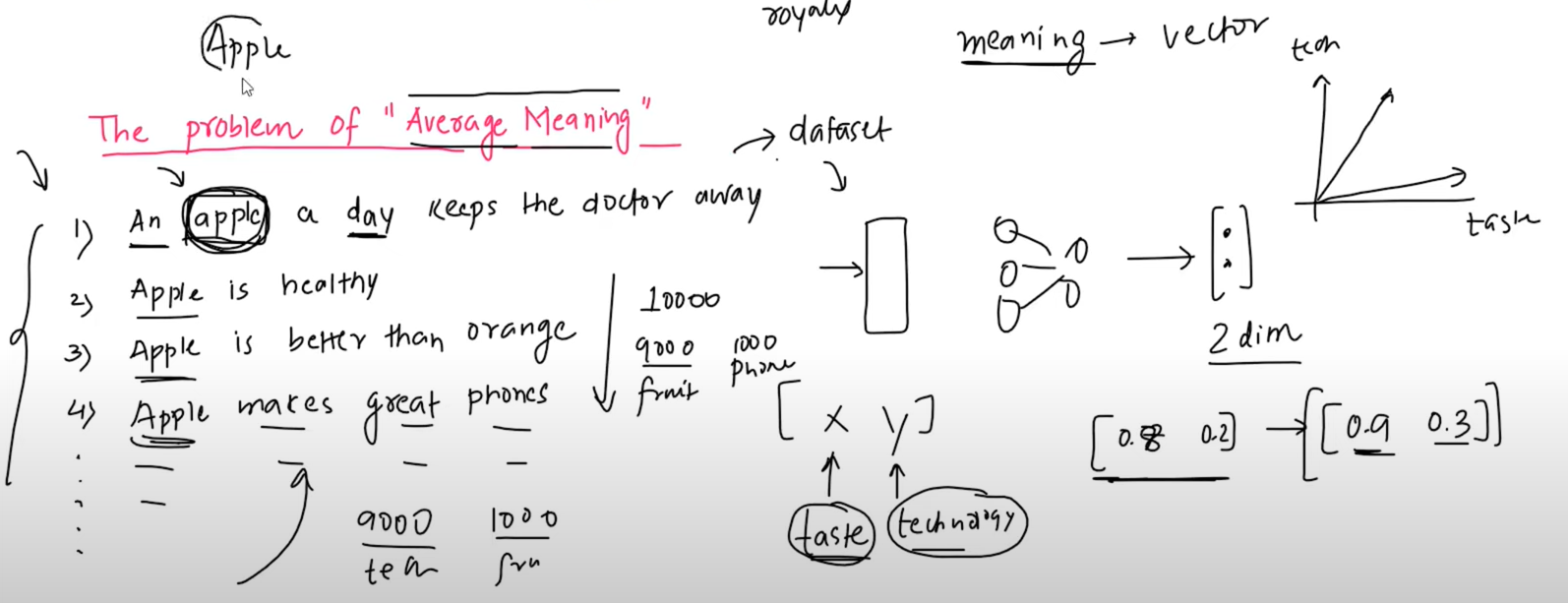
So there would be a mechanism of parallel processing.

# Self Attention

The first thing we do is to convert words to numbers i.e. vectorization.

Embeddings are useful to do that because they contain context of words e.g King [0.9 1 0.8 ] it can be like [power human royal] while any other word may have different embeddings like cricketer.

## The problem with Embeddings



The main problem with embeddings is that they contain static meanings that do not change according to data . So we want contextual dynamic embeddings that contain context of data. Therefore the mechanism Self Attention is introduced i.e. a way of converting static embeddings to dynamic contextual embeddings.

# How does Self Attention work ??

Now we want to know what’s the calculation inside this self attention.

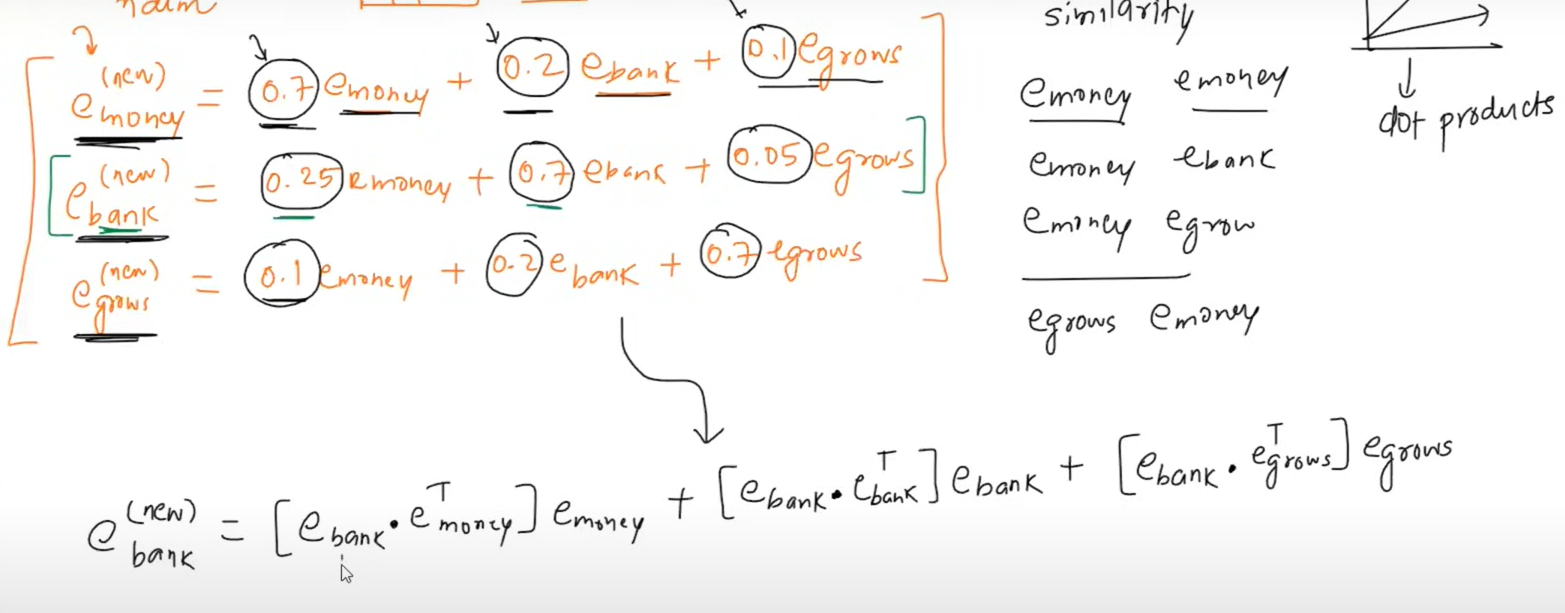
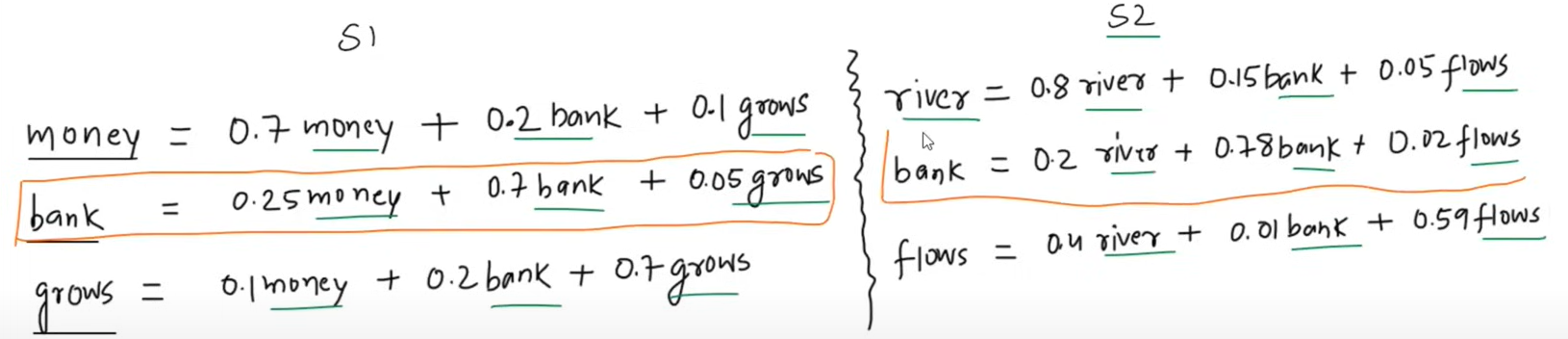
E.g. Money Bank Grows River Bank Flows

Now the static embedding takes “Bank” as the same vector but this indicates 2 separate meanings: money/financial and a place in the 2nd one.

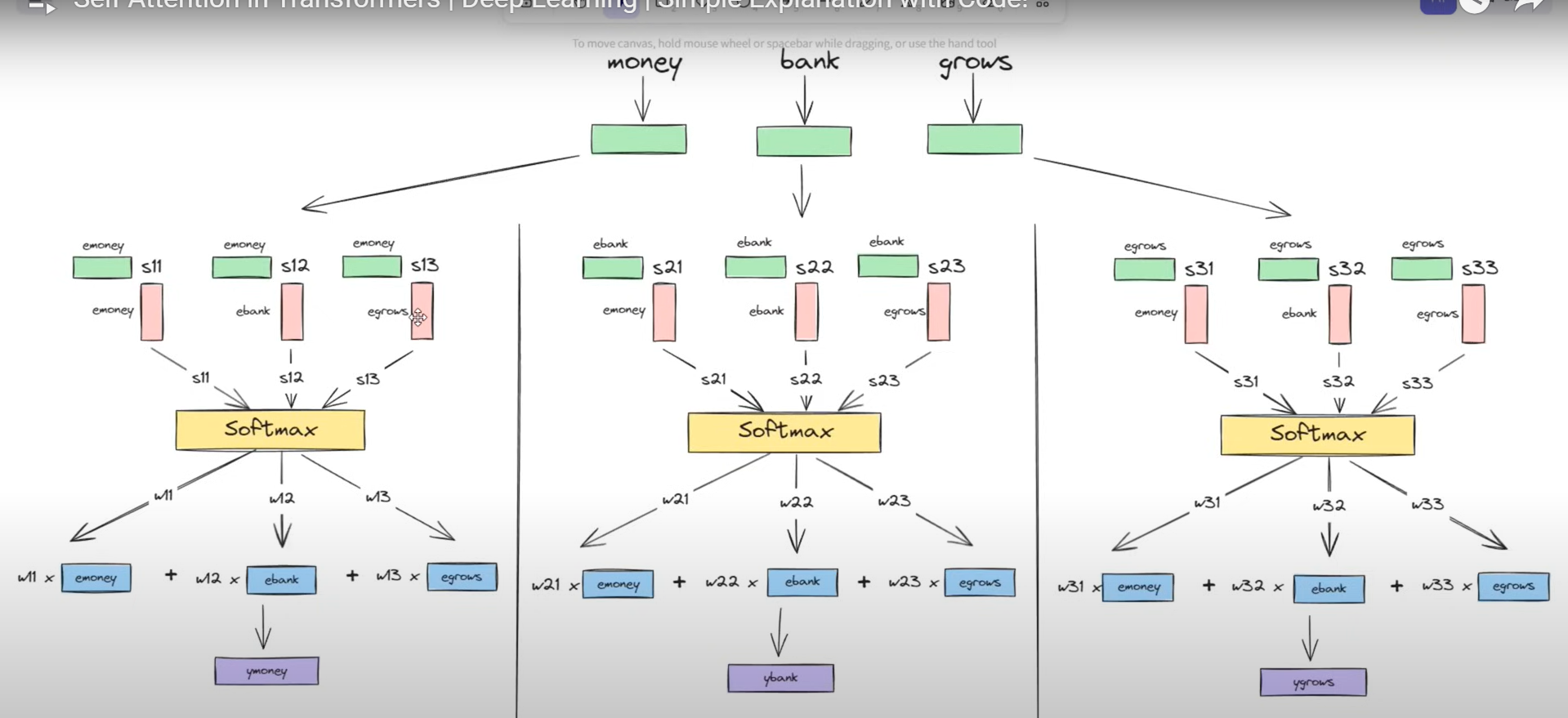
Now, bank = 0.3money + 0.7 bank + 0.1 grows

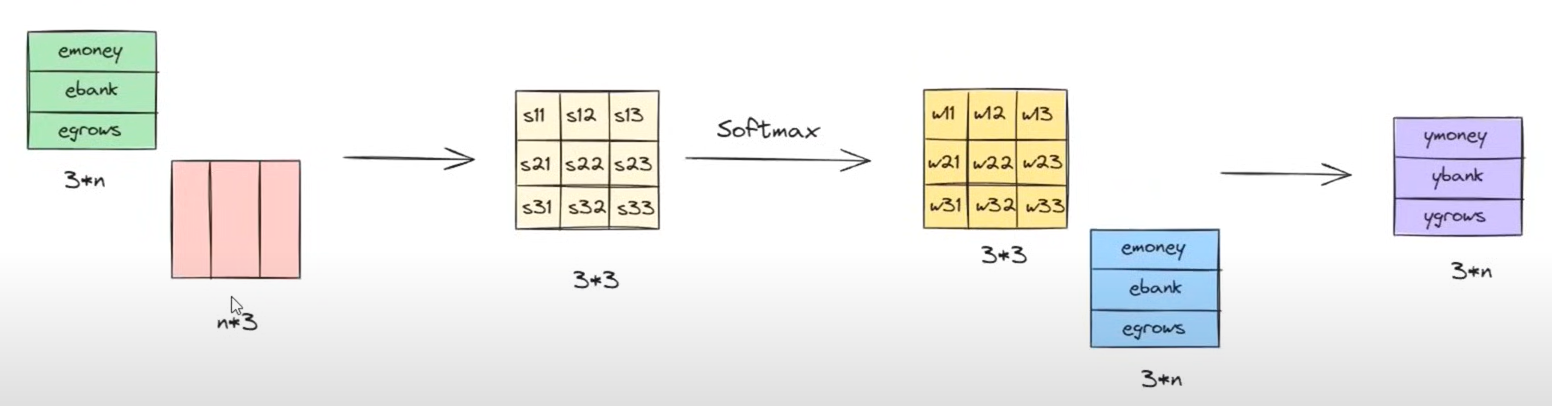
Similarly in 2nd sentence bank = 0.5 river + 0.4 bank + 0.1 flows

These numbers are basically similarity which means what is the similarity between money and bank = 0.3 . This is simply calculated by a dot product between 2 vectors.



We will convert these compositions into corresponding embeddings. Now we will generate a new dynamic embedding that has meaning regarding the context of data.

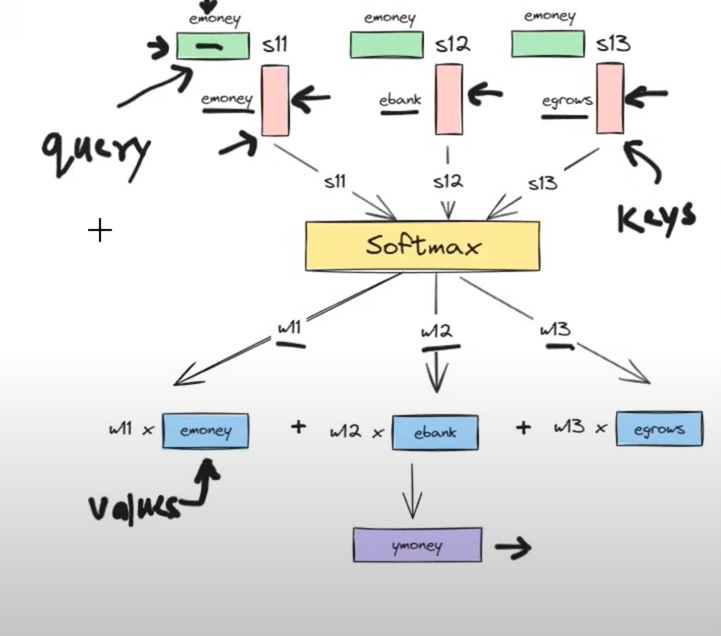




## Points to consider 👍

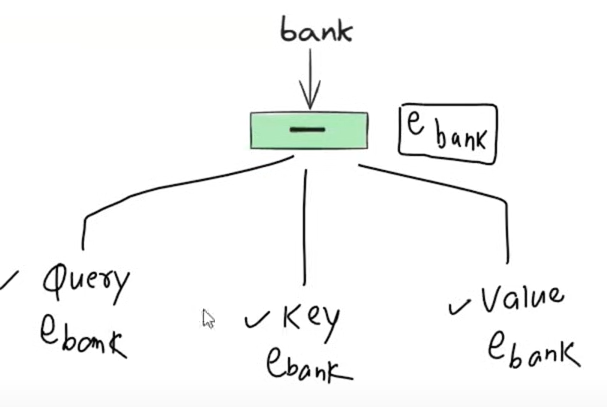
1. This is a parallel operation
2. There are no parameters involved, just a dot product and a softmax.

Now the problem arises is that since there are no parameters involved , there is no training of embeddings from data. This causes them to capture general meaning but not the contextual meaning of data.



To introduce learning , we identify query , keys and value vectors as the basis of a python dictionary.

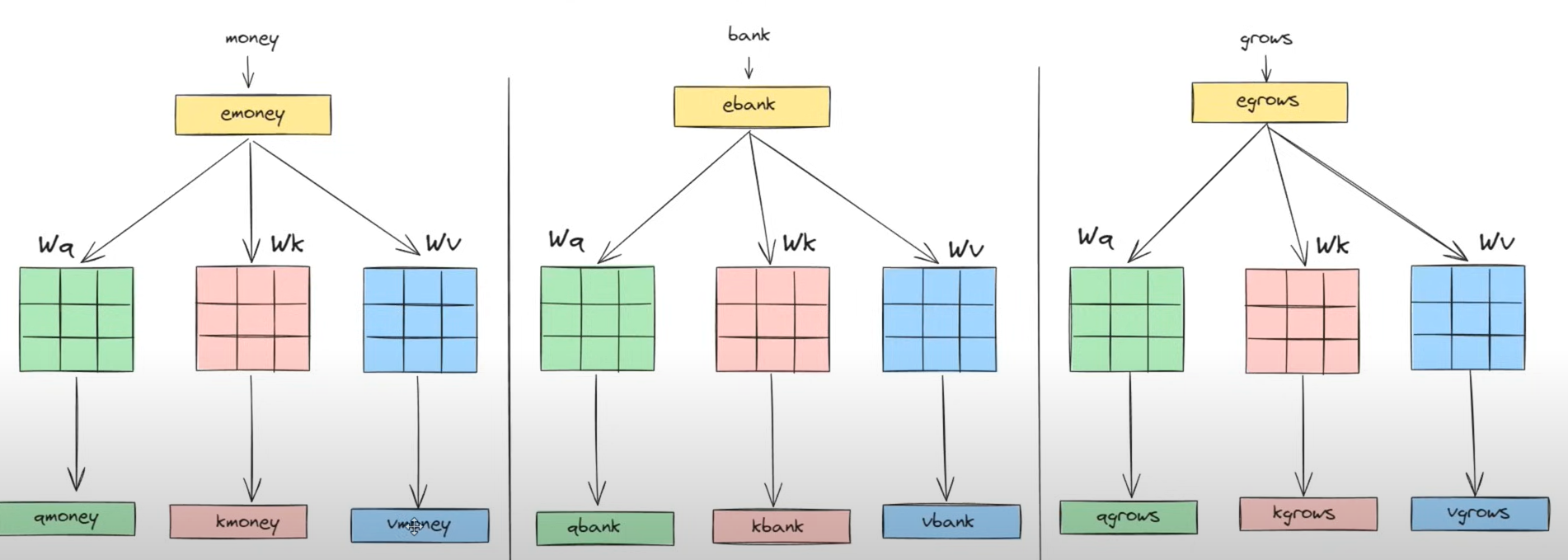
* Emoney as a query - what similarity do we have ?
* Emoney as key - telling similarity
* EMoney as value - giving value that multiplied with w11.



So we get every vector composed of 3 vectors because every time we need a different aspect of it, not all the time the whole vector.

## How to build these vectors (Query , Key , Value) based on Embedding Vector ??

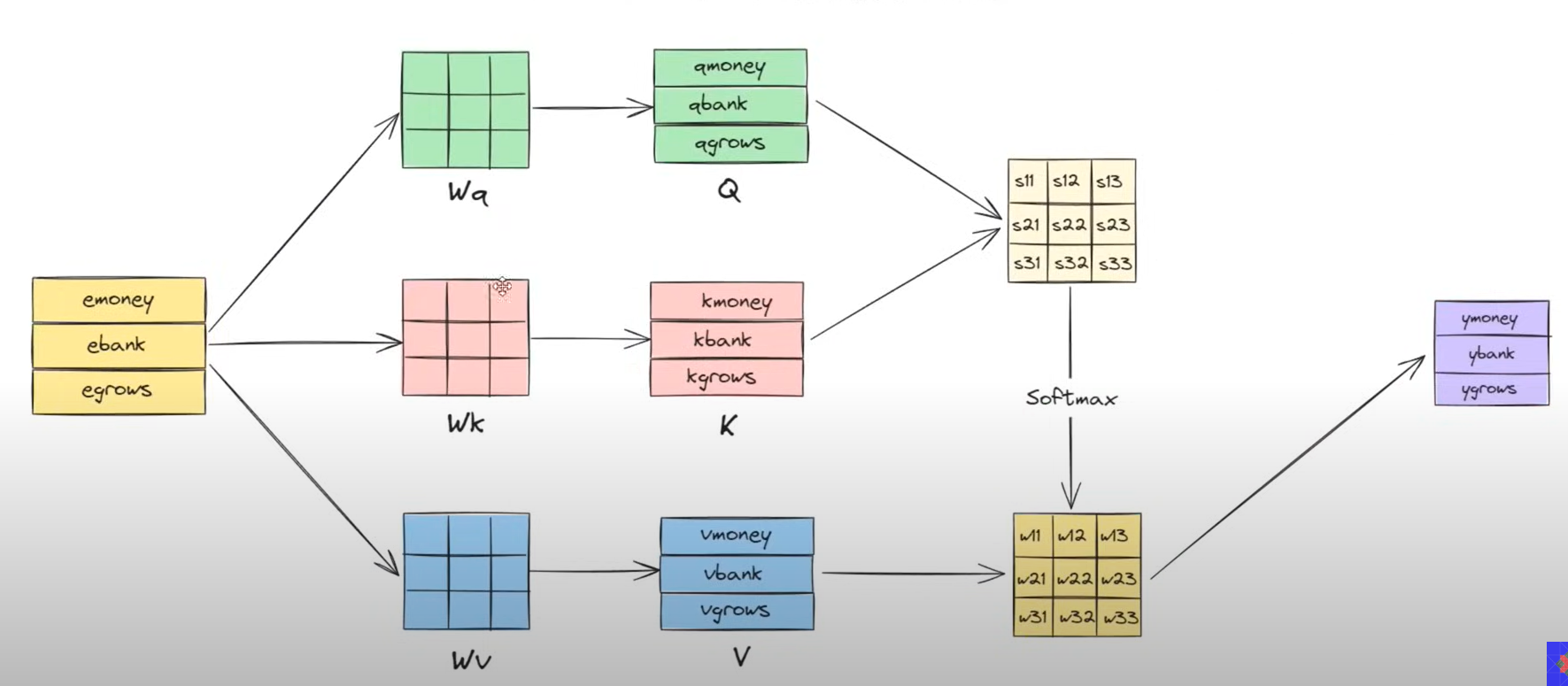
We use linear transformation to make 3 different vectors from a vector.



We take a matrix with random weights to produce it with embedding of the vector.

The query matrix of all vectors are the same , similarly the key and value matrix of all vectors.

During backpropagation one translation occurs and loss calculated these weights are updated using gradients.



Attention (Q,K,V) = Softmax( ) V

# Scaled Dot product Attention

Attention (Q,K,V) = Softmax( / ()) V

= dimension of the key vector

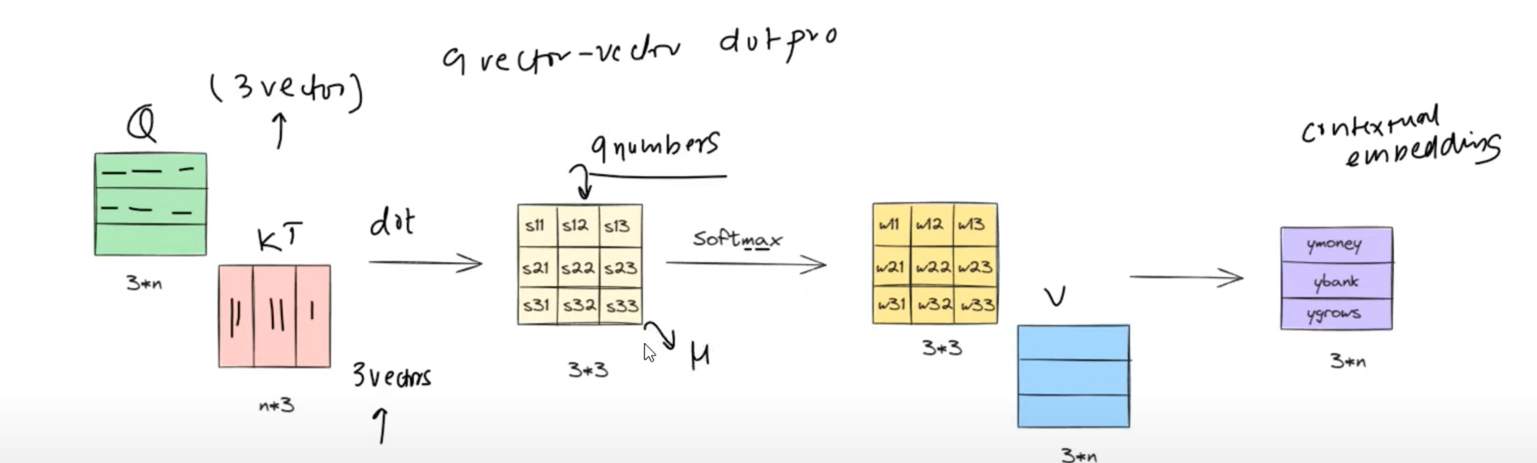
Let's suppose we have Emoney from dim-3. And matrices Wa , Wb , Wc of (3X3). Therefore Qmoney = Qkey = Qvalue = dim-3 (Although they can vary)

Nature of Dot Product :

High dimension dot product —--- High variance

Low dimension dot product —---- Low variance

of 3-dim vectors >> of 2-dim vectors



Vectors of any dimension have 9 vector-vector dot products.

What variance causes is it gives variety , high variance means more variety i.e one no. is very high other can be very low. What softmax does is it enlarges the probability of large no. and reduces down other small ones very low.

This causes a Vanishing Gradient Problem with smaller ones and we lose what context in them. To reduce down variance we need to divide it with a scaling factor.

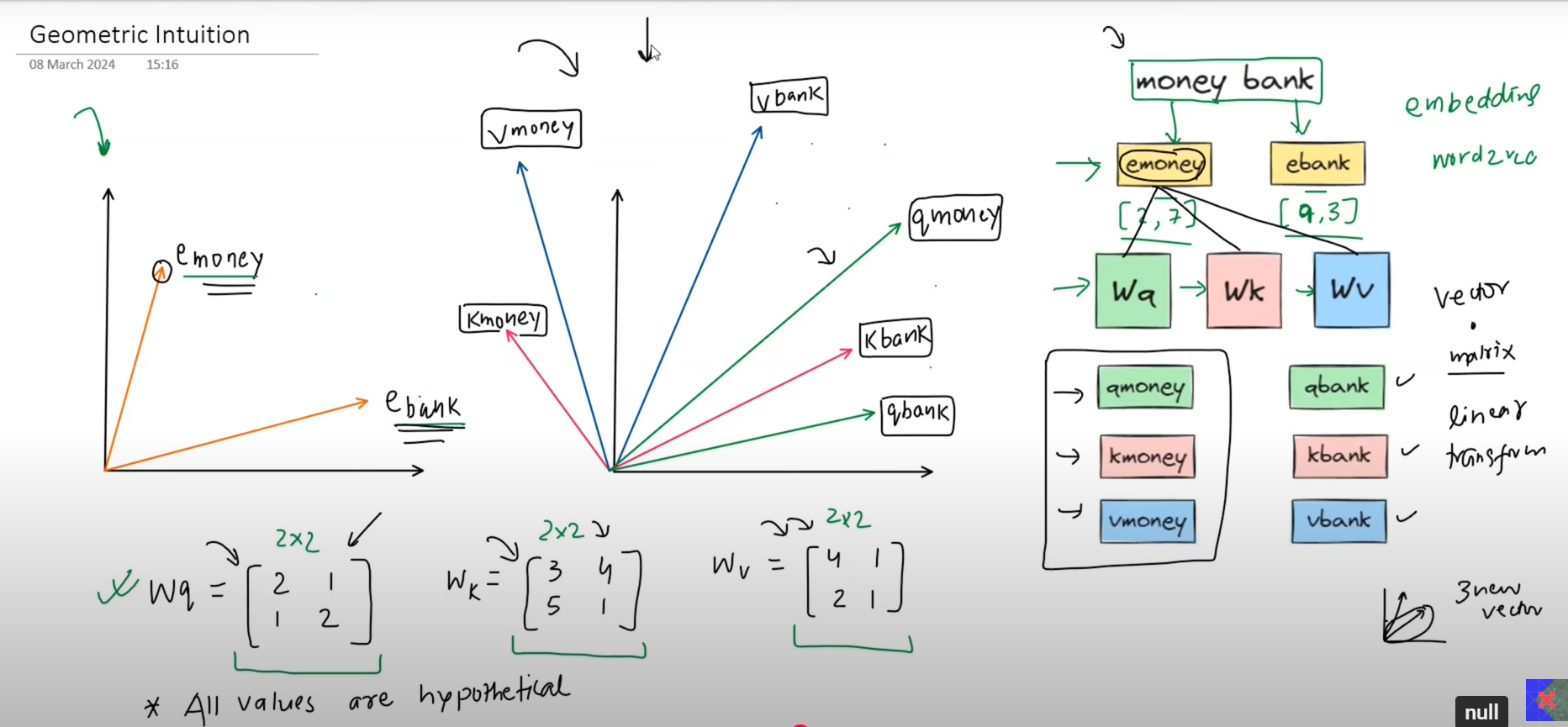
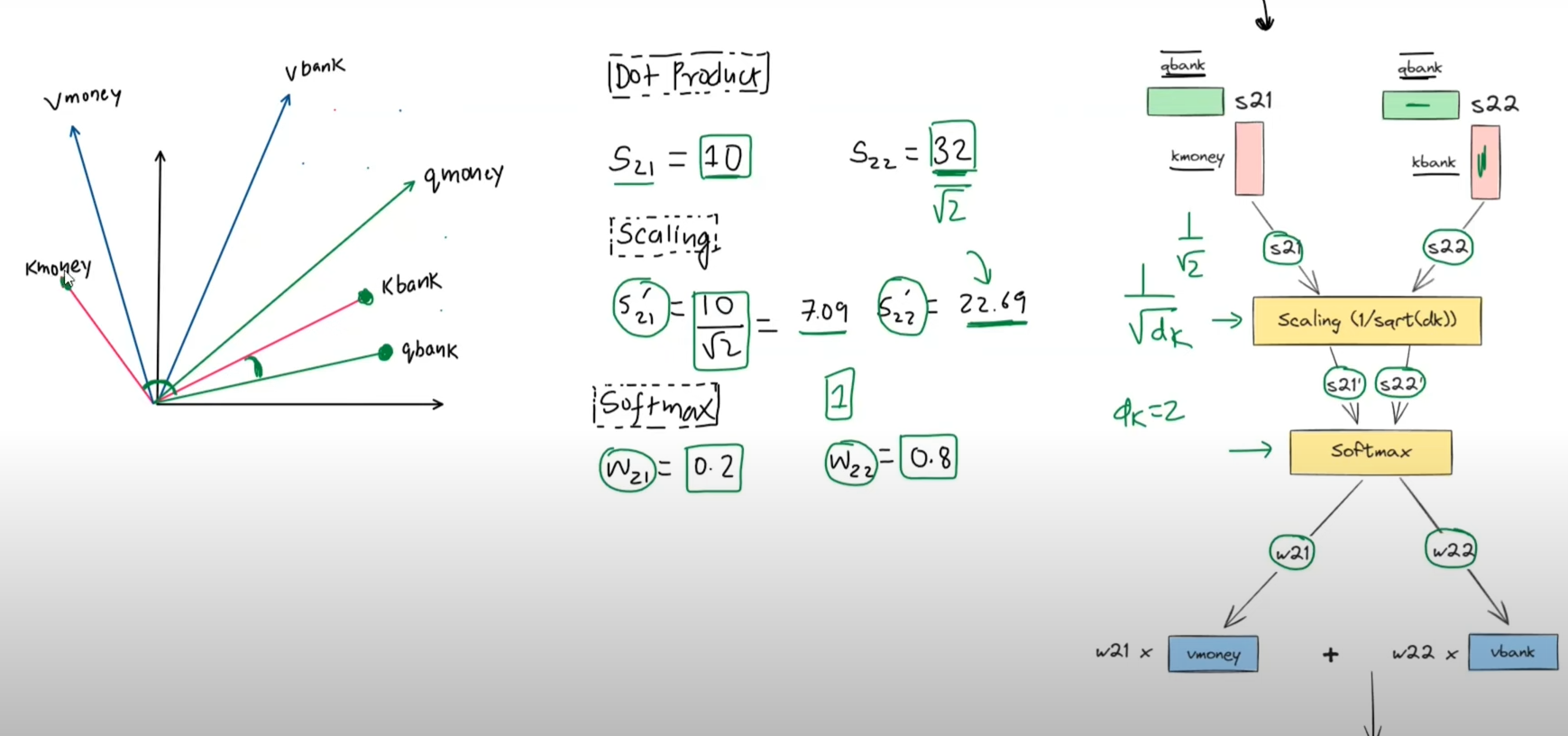
of 1-dim vectors = Var(X) Property :

of 2-dim vectors = Var(Y) = 2Var(X) X —-- Var(X)

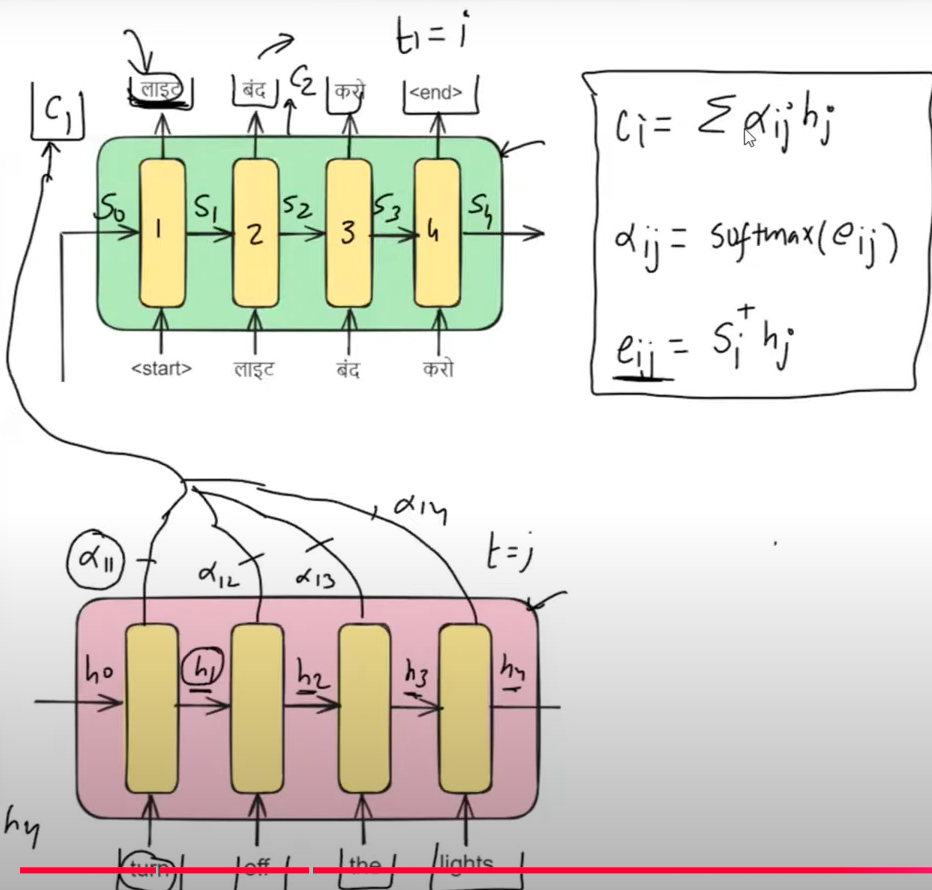
Therefore , Y/ = 2Var(X) = Var(x) CX —-- Var(X)

Thus dividing by we get the same variance.

# Geometric Intuition of Self Attention



# Self Attention Vs Luang Attention

Luang Attention have 3 equations 

Similarly we have Self Attention :

Y\_turn = W11 + W12 + W13 + W14

Y\_turn =

Wij =

Sij = Here Si is query , hj is Key and also value.

Self attention is called so because it is also an “Attention mechanism” following the same 3 equations.

It is called “Self” because it is an attention between the same sequence i.e. intra sequence attention mechanism while Luang attention is inter sequence attention mechanism having attention between 2 different sequences like english and hindi.