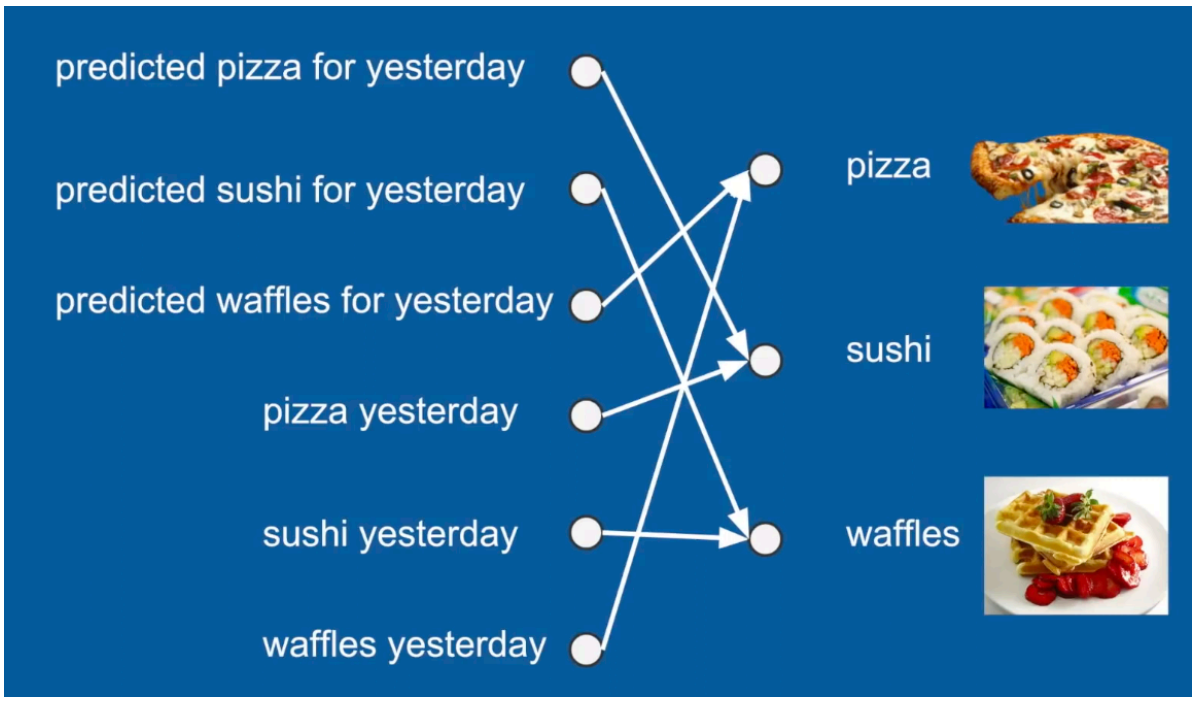
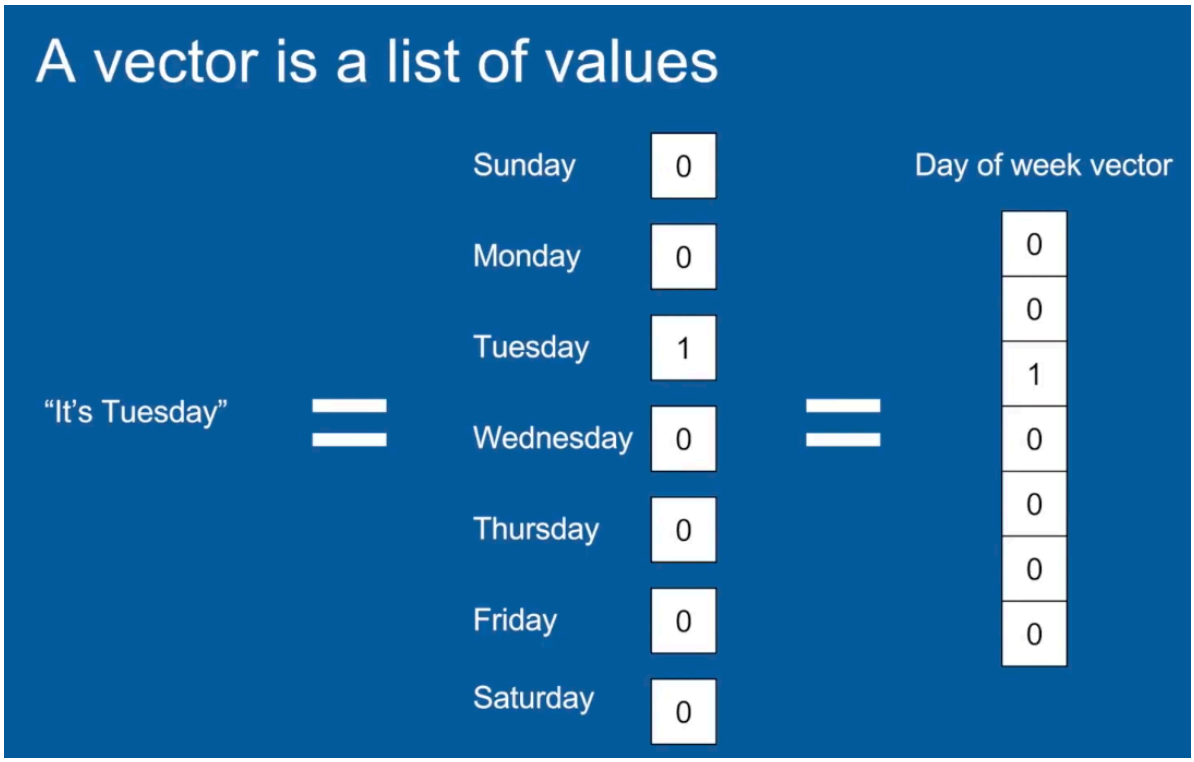
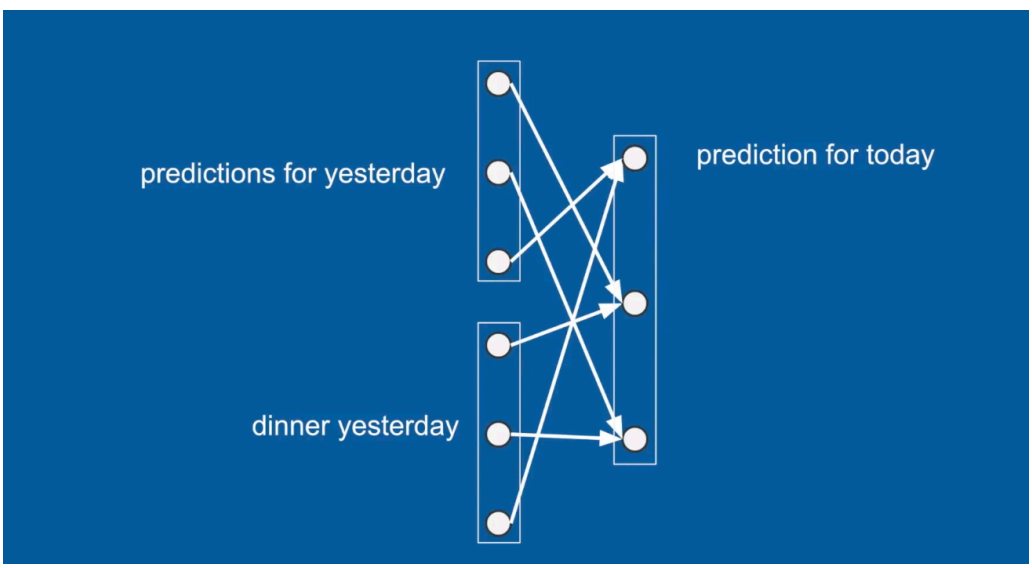
- ***introduction***
  - -> identifying pictures
  - -> sequence to sequence translation
    - -> speech to text <- convolutional neural networks
    - -> translating languages <- recurrent neural networks
      - -> long short term memory
- ***predicting what is for dinner example***
  - -> e.g with menu options
  - -> there are three menu options
  - -> depending on factors -> for example if you are in a meeting late at work - then what you eat for dinner will change probability wise
    - -> it depends on the day of the week, the month of the year
    - -> all of these inputs determine the outcome in a "voting process"
    - -> you can train the model on the history of what you had for dinner
  - -> you can change the neural network based off of how good its predictions are
    - -> in this example the voting process depends on what the previous dinner dish was
    - -> using the previous predictions

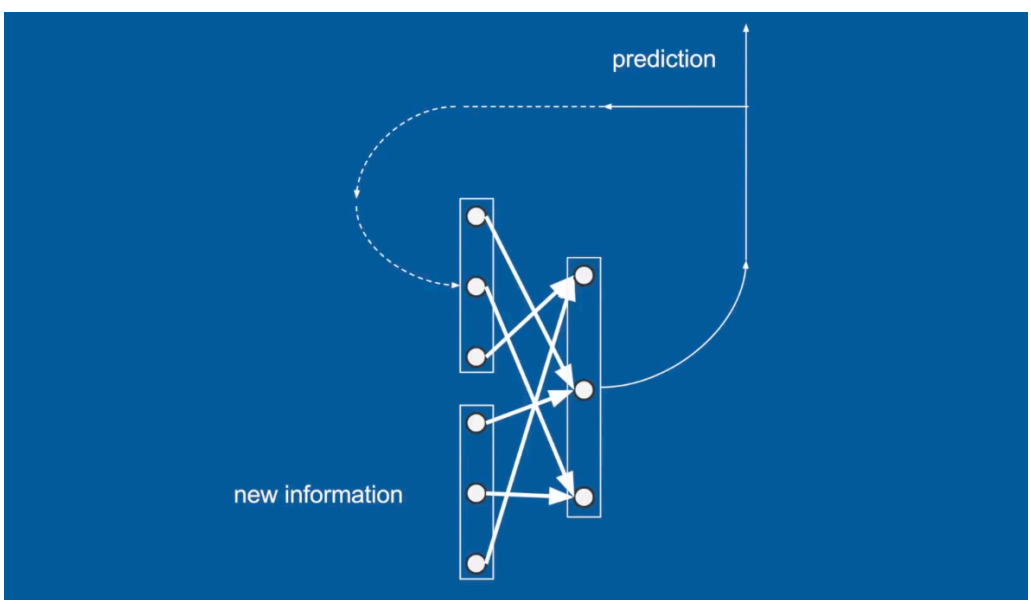


- ***vectors***
  - -> a list of numbers
  - -> for example a vector which contains the different temperatures for days of a week
  - -> this is native to a computer
  - -> you can concert words into numbers -> and then store those numbers in a vector

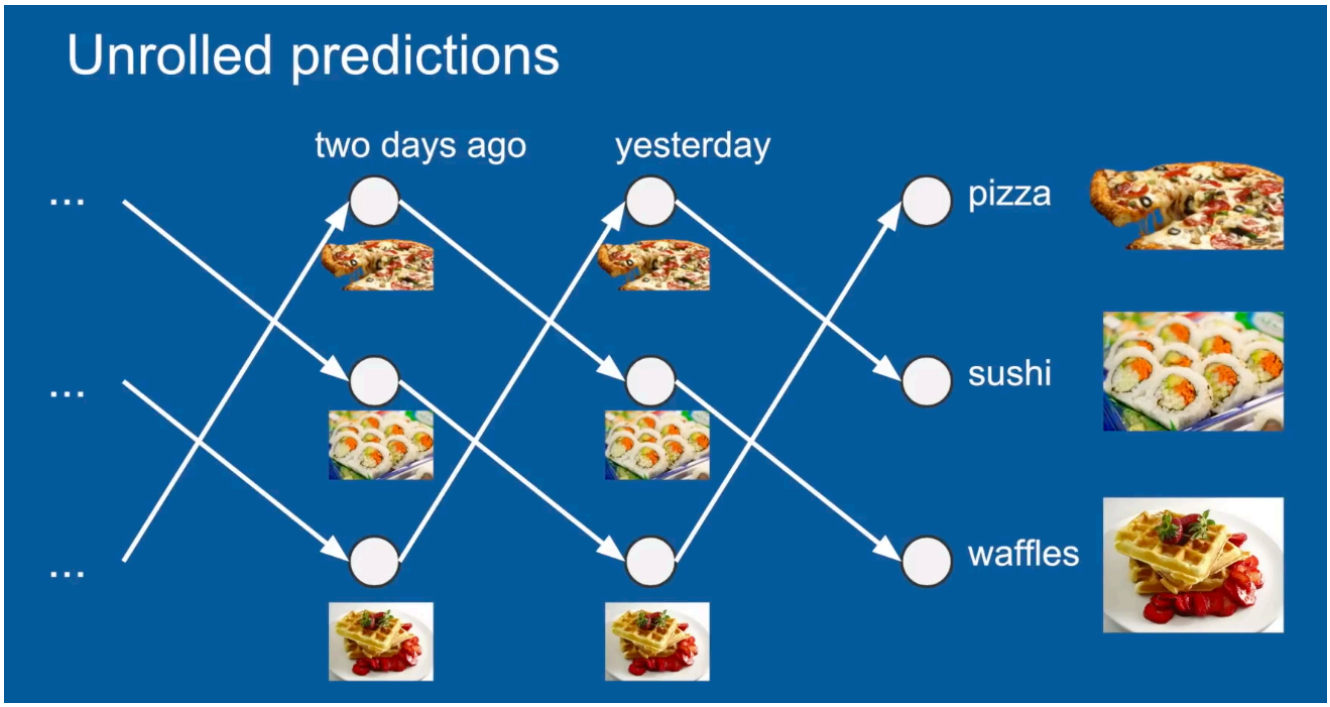


- ***making a prediction vector for the dinner example***
  - -> the inputs and outputs can be grouped into vectors
  - -> if the output of one day is used for the inout into the next

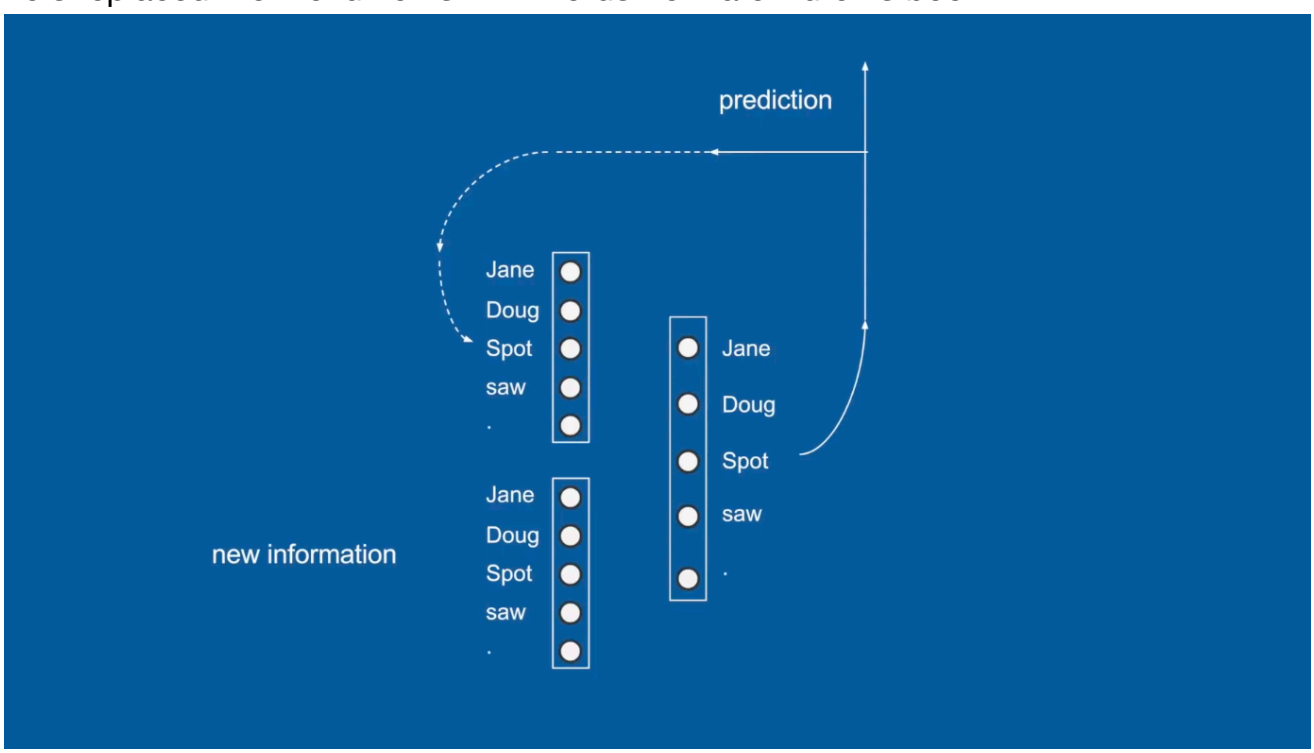




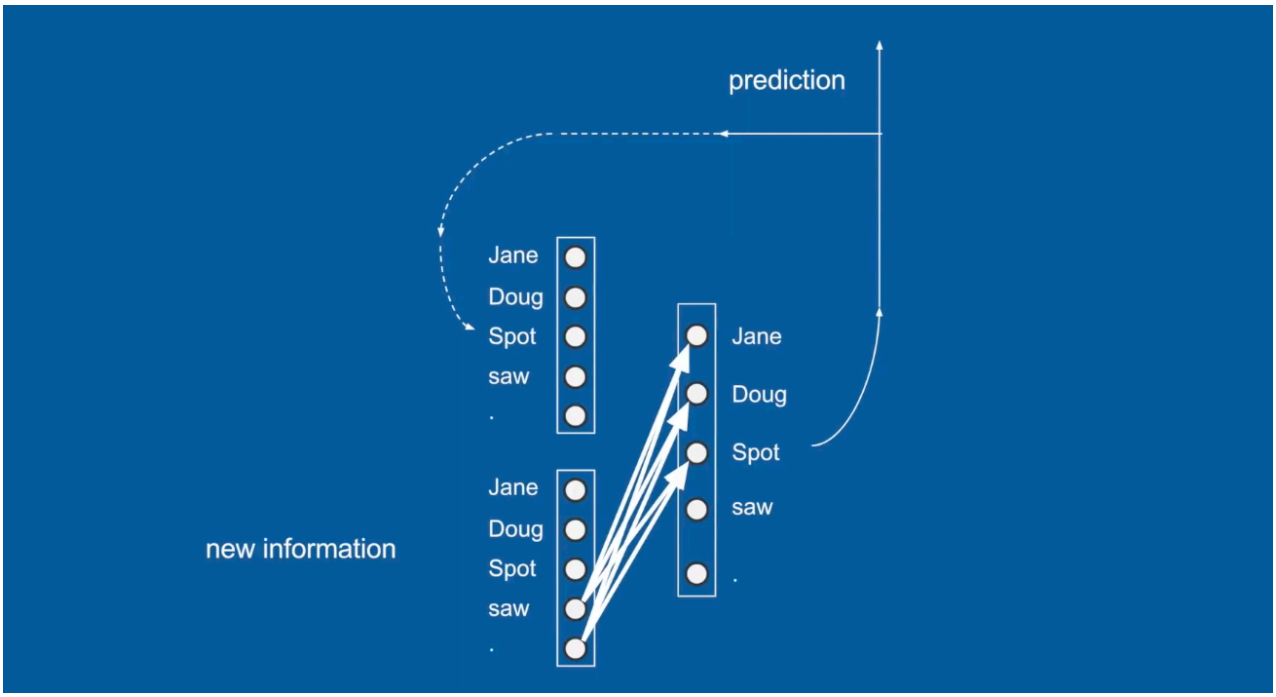
- ***-> unrolled predictions***
  - -> you can use the current vector and go back in time -> if the input of one vector depends on the output of another
  - -> this is an example of a recurrent neural network



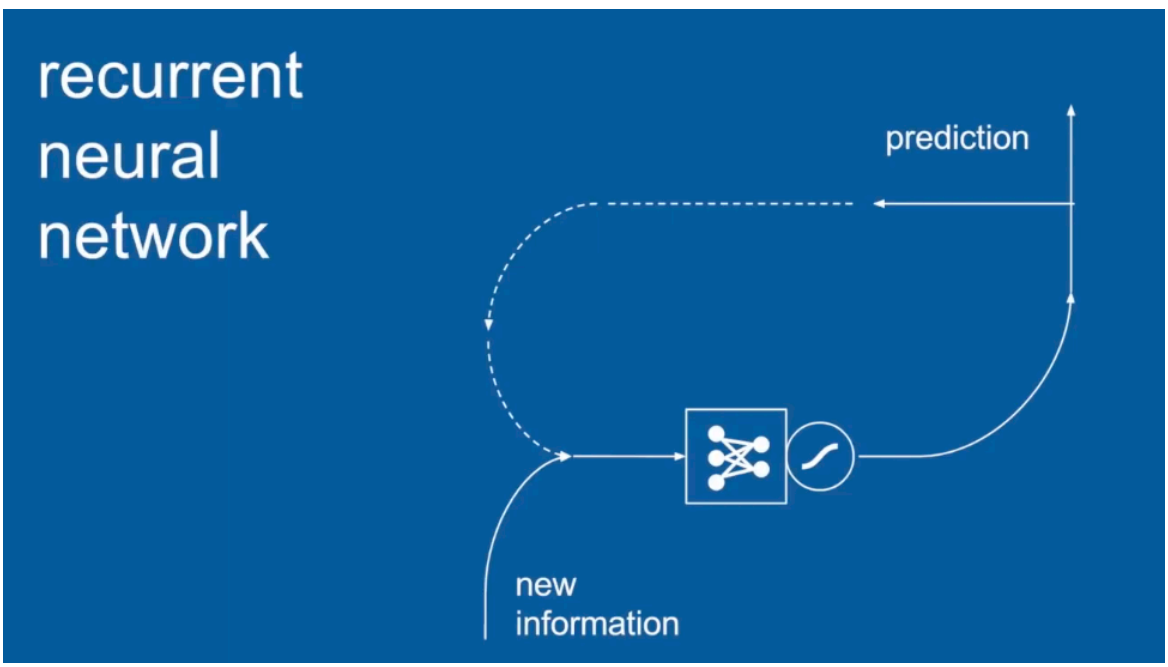
- ***-> writing a children's book to show the holes in the recurrent neural network approach***
  - -> he has a group of words and a dictionary of them
  - -> the network is supposed to put them into the right order
  - -> he's replaced the menu items with words from a children's book



  - ***-> after training the network -> we see certain patterns***
    - -> for example depending on the word we're on there is a certain probability that the next word will be x

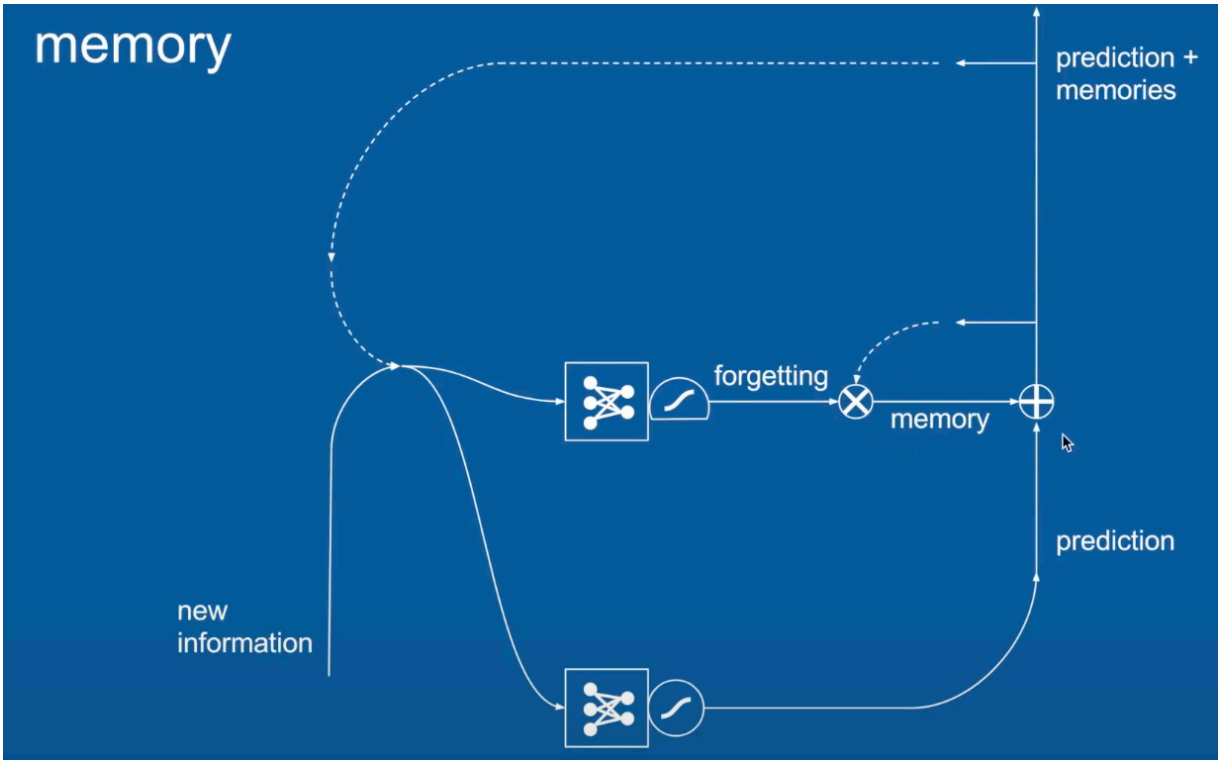


- ***-> recurrent neural networks***
  - -> this is also using a sigmoid squashing function
  - -> all of the votes coming out are put into a squashing function
  - -> it's ranging between ±1
  - -> there is a loop -> the outputs of one iteration are inputted into the next
    - -> using the sigmoid function stops the outputs from blowing up by normalising them
    - -> this is an example of a negative feedback loop

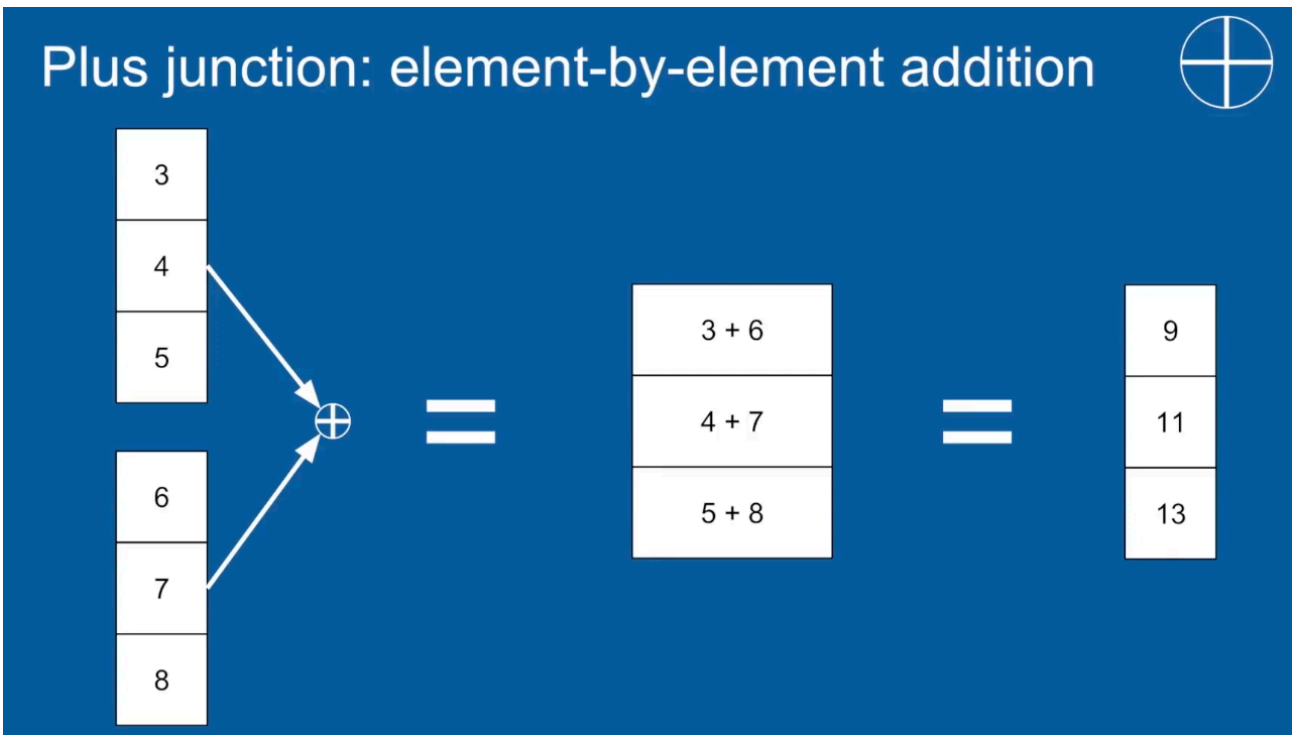


- ***-> each of the predictions is one of the words it predicts -> you're on one word and it's predicting the next - this is one of those loops***
  - -> so it's memory is short-term
  - -> if it's predicted a word then 10 words later it will forget the words which it originally predicted - because it's doing one of them at a time
  - -> this is about how we can remember the predictions it made further back to avoid making mistakes
    - -> adding memory to the recurrent neural networks
    - -> we are adding memory into the loop to remember the value of the weights for

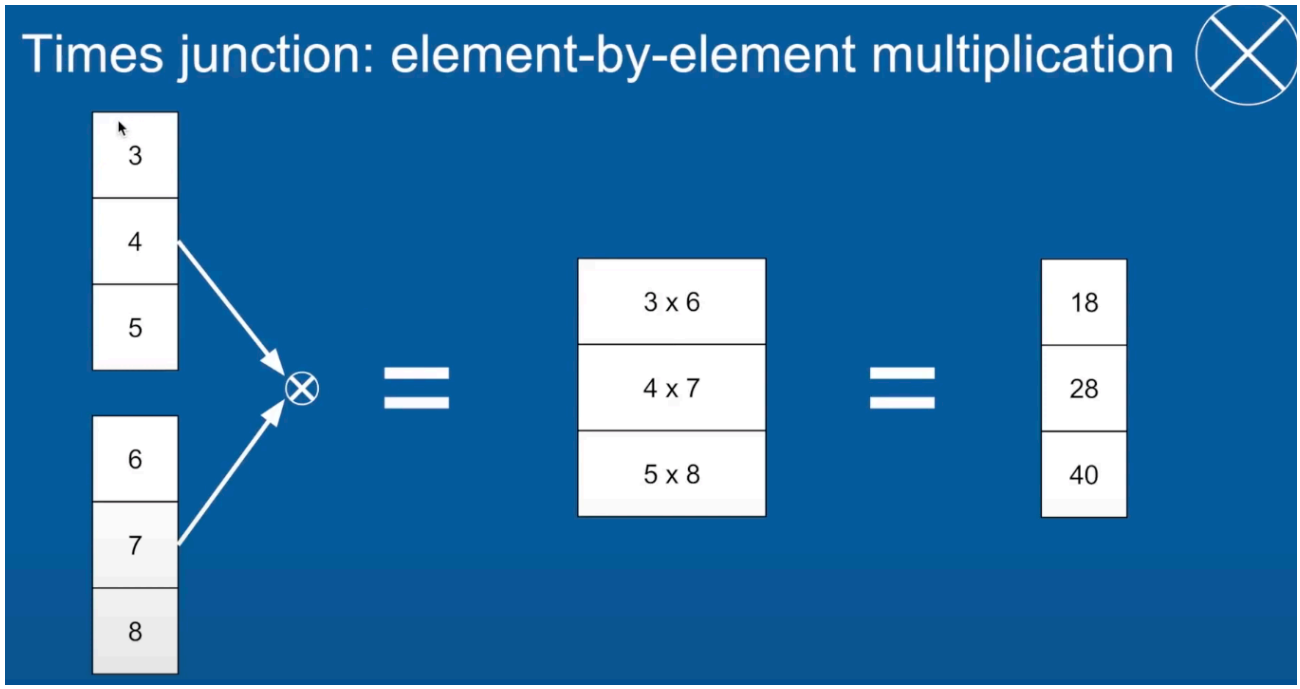
each iteration of the loop (each time a new word is predicted)
    - -> to add memory in - we need another squashing function



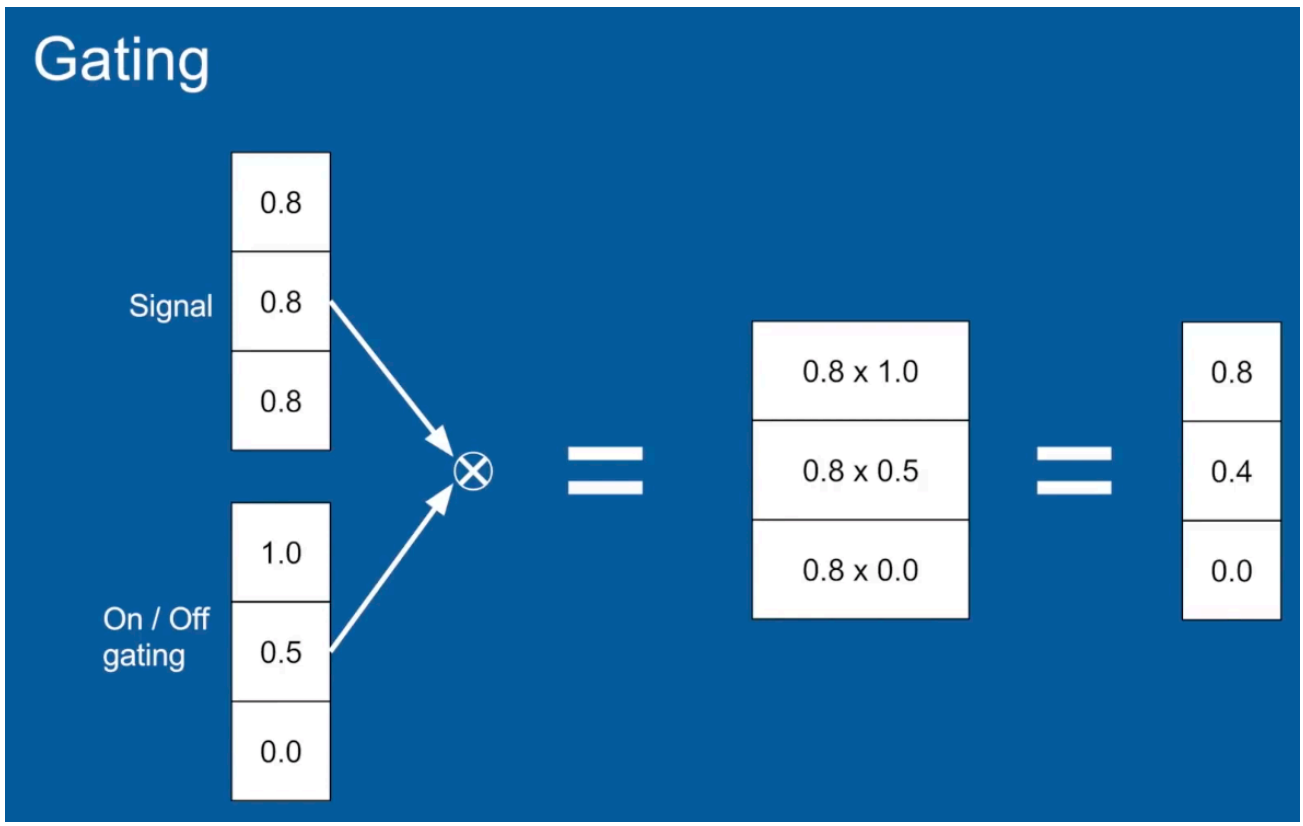
- ***-> the cross in a circle is a gate***



    - -> you're adding the two vectors together
- ***-> the next one is a dot product***
    - -> you're multiplying the two vectors together element by element
    - -> these are junctions <- calculations which are performed at a node in the graph

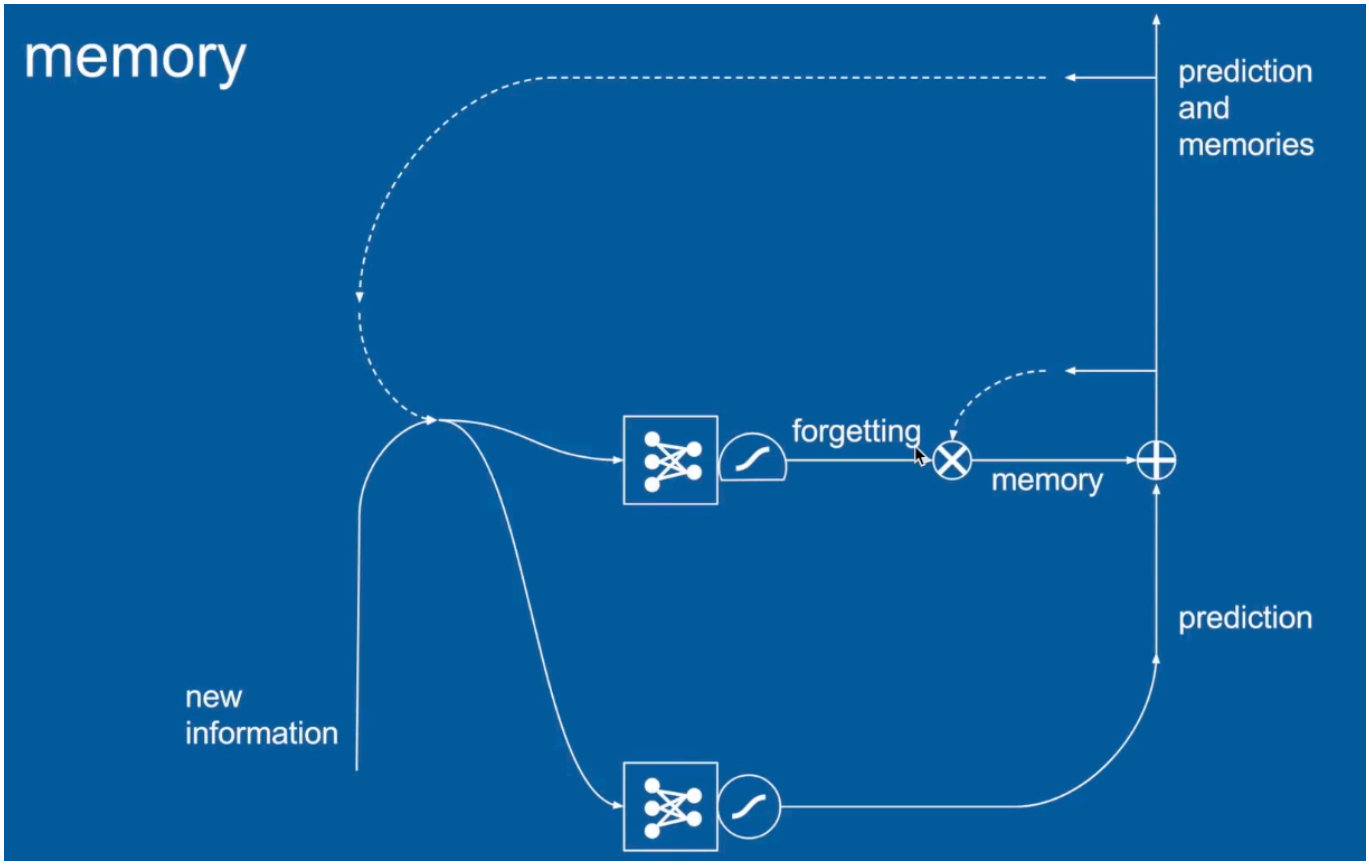


- ***-> gating***

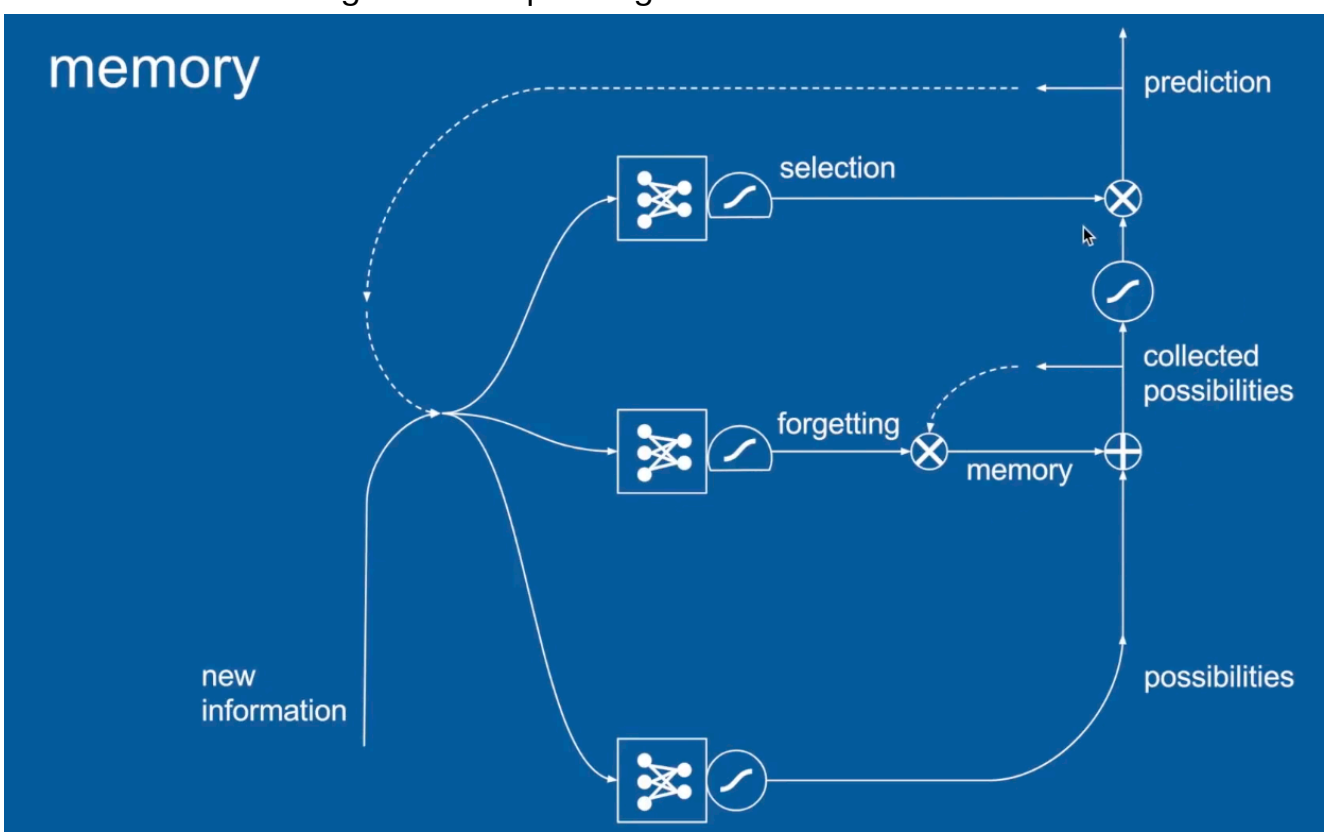


- -> he's using the example of a tap -> either it's open or it's not
- -> either the signal is passing through it or not
- -> what determines the magnitude of the element is the magnitude of the 'gating'
- -> the magnitude is between 0 and 1
- -> so do this we introduce a logistic function <- it's another sigmoid function
    - -> it's like a tanh function which has been shifted upwards

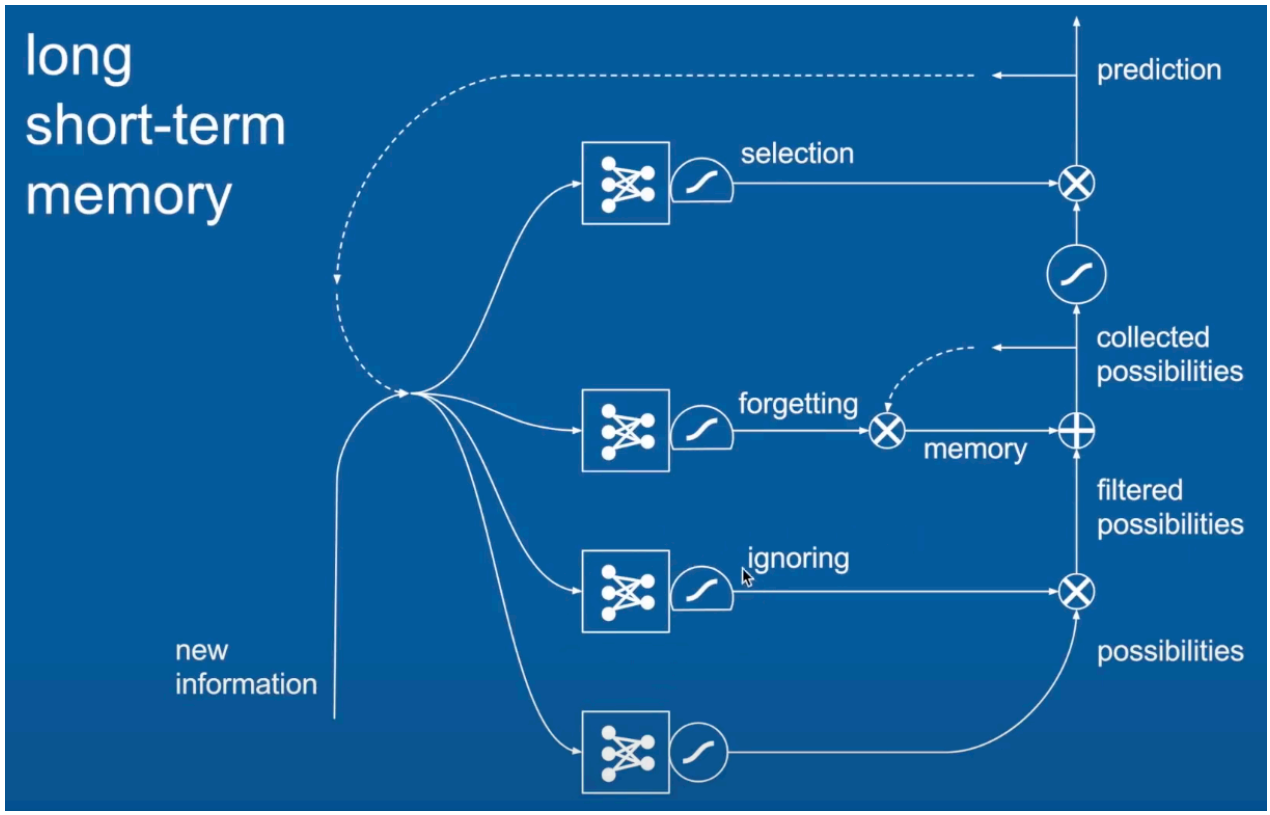
- ***-> a copy of the predictions is saved in the memory in the graph***



  - -> the memory is storing a copy of the predictions for the next iteration through the network
  - -> there is another gate there which forgets some of those points
  - -> so we have the predictions plus the memories which we've accumulated
- ***-> adding a selection filter into the model***
  - -> this votes on what each of the gates should be
  - -> we are also introducing another squashing function

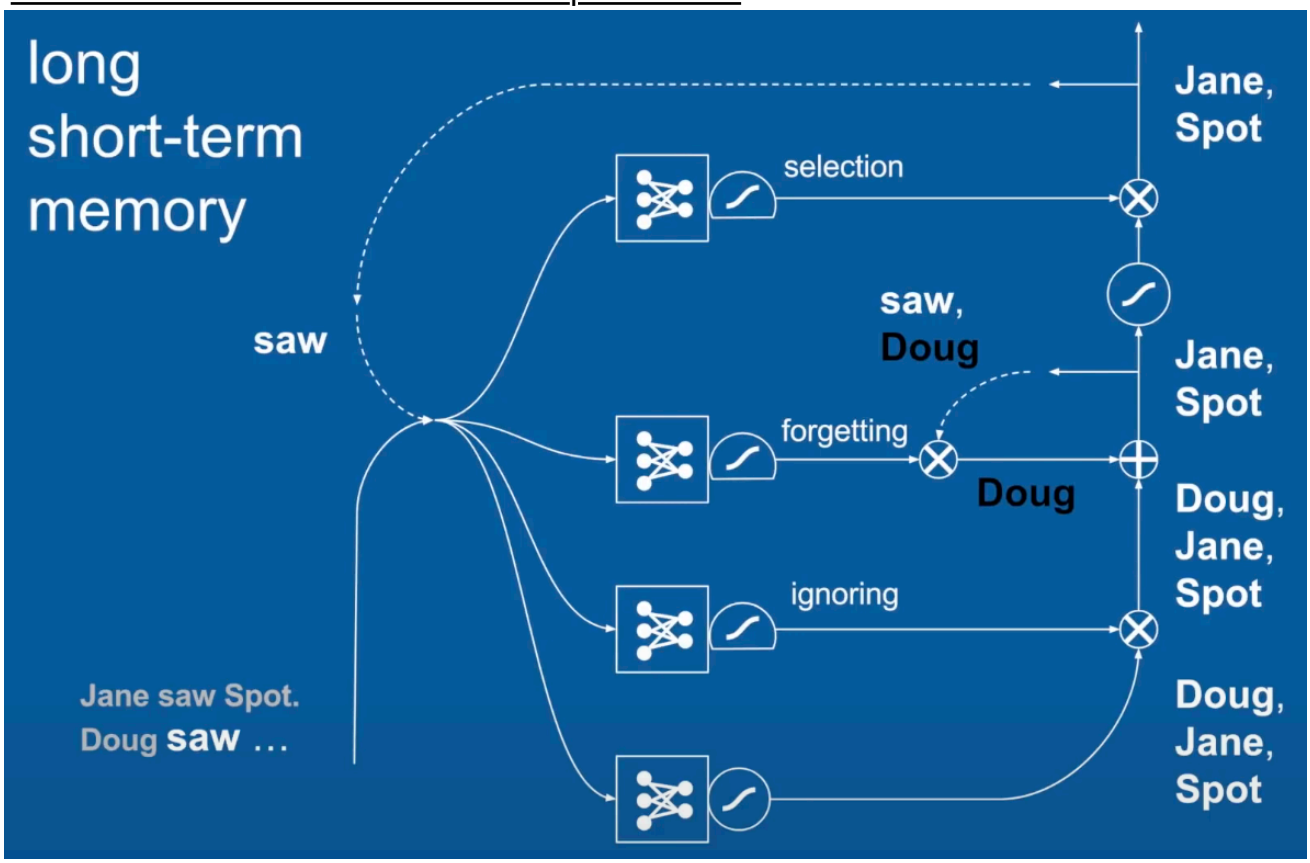


- -> after an iteration, we store some of the predictions the model makes
- -> there is a gate which chooses whether to get rid of those predictions or not <- it's "voting" on them, and learning which predictions to let go
- -> those predictions are then fead into the next iteration for the model
- -> and since there is long term memory, it's storing the weights which the model had at each stage (while generating one word at a time)
  - -> it's an attention mechanism <- getting rid of the predictions which we don't want

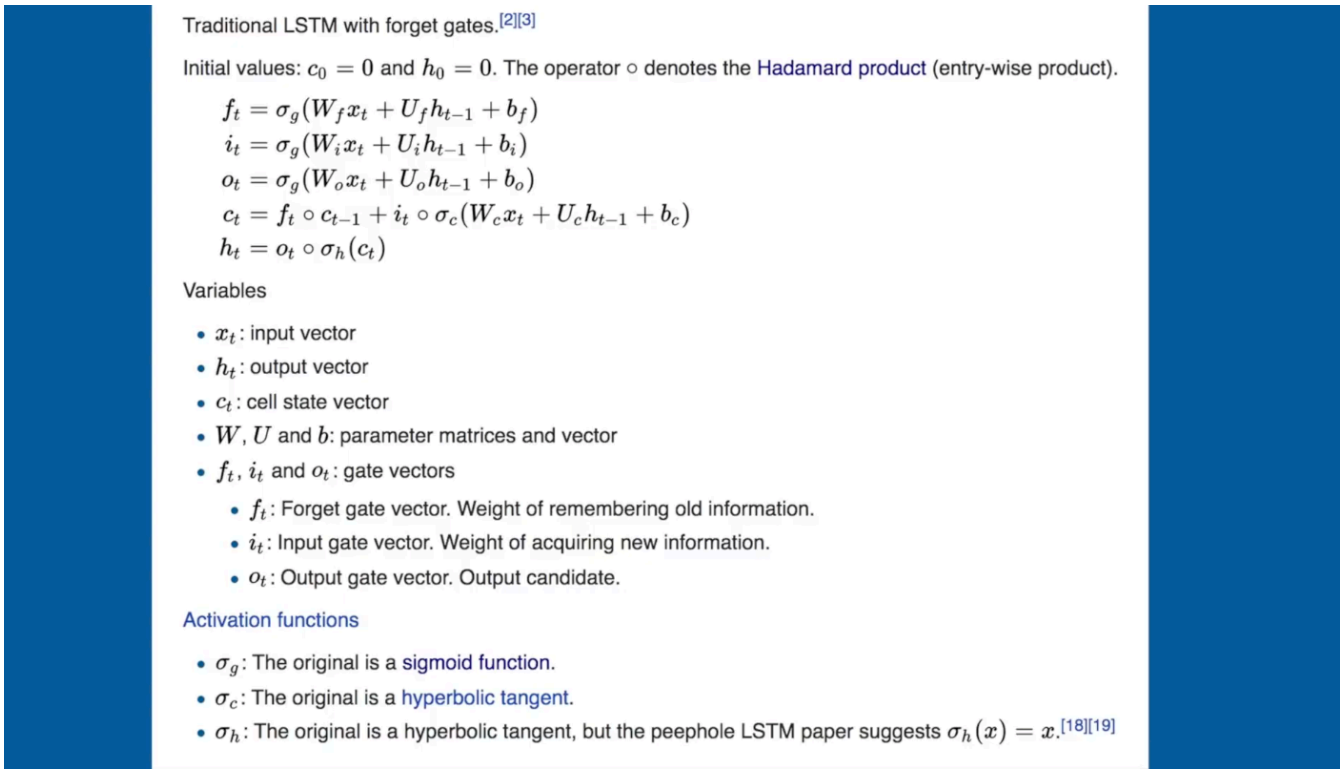


- ***an example of long term short term memory***
  - -> this is an example which writes a children's book
  - -> assuming the model has been trained on previous children's books
  - -> in this example we've entered the most recent word in the story into the model
    - -> it's predicting them one at a time - because the meaning matters
    - -> the same word can mean multiple things
  - -> the input word is being passed into all four of the neural networks
    - -> selection
    - -> forgetting
    - -> ignoring
  - -> this example is simple (and so neglects attention)
  - ***-> they are passing one word (Doug) through the network***
    - -> this word was output from a previous iteration through the network
    - ***-> those vectors are then passed through the four parts of the circuit***
      - ***-> predictions***
        - -> this is the selection part of the network
        - -> it's predicting the next words for in the sentence
        - -> some words are having positive predictions and some of them are having negative ones
      - ***-> he neglects attention and forgetting***
        - -> assuming there is no memory in the network
      - -> then the selection
        - -> when the most recent word is a name
        - -> it's sending out the word saw as the next prediction

- ***-> then the most recent word is now 'saw'***
  - -> this is again passed through the recurrent neural network
  - -> this is in the similar way to the previous word
  - -> the previous set of possibilities were remembered
    - -> these are passed to the forgetting gate
    - ***-> based off of the training of the model -> it is choosing which of these predictions to remember and which to forget***
      - -> (for example it's just remembering the predictions for the names)
      - -> it's running vector operations in order to do this (similar to the ones above)
      - -> the selection gate
      - -> it's passing through the predictions for the names
  - -> in other words long short term memory can choose which predictions to remember and use these in the next predictions



- ***-> uses of long short term memory***
  - -> translating languages <- this is a phrase to phrase process
    - -> long term short term memories can find the different grammar structures in those phrases
  - ***-> translating speech to text***
    - -> to predict which word is being spoken
    - -> using the history of the words to better predict what is coming next
  - -> for information which is related to time <- audio, video
  - ***-> robotics***
    - -> an agent taking in information from a set of sensors
    - -> making a decision and carrying out an action
    - -> they can influence which action is taken noq
- ***-> maths for long term short term memory***
  - -> the models for lstm on diagrams are mathematical and represent these probabilities



Traditional LSTM with forget gates.[2][3]

Initial values: $c_0 = 0$ and $h_0 = 0$. The operator $\circ$ denotes the Hadamard product (entry-wise product).

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

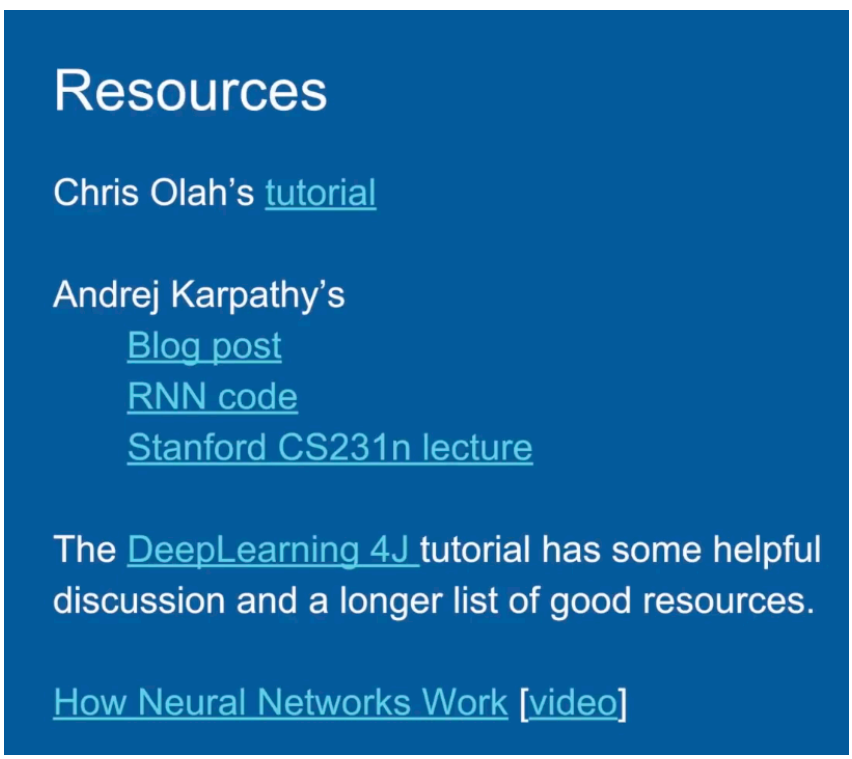
Variables

- x_t: input vector
- h_t: output vector
- c_t: cell state vector
- W, U and b: parameter matrices and vector
- f_t, i_t and o_t: gate vectors
  - f_t: Forget gate vector. Weight of remembering old information.
  - i_t: Input gate vector. Weight of acquiring new information.
  - o_t: Output gate vector. Output candidate.

Activation functions

- σ_g: The original is a sigmoid function.
- σ_c: The original is a hyperbolic tangent.
- σ_h: The original is a hyperbolic tangent, but the peephole LSTM paper suggests $\sigma_h(x) = x$.[18][19]

- ***-> further resources for lstms***
  - -> the wikipedia page
  - -> tutorials and discussions
  - -> Karpathy's blog post for examples of what lstms can do
  - -> the video on how neural networks do
  - -> this entire video was for recurrent neural networks



  - -> the main neural network components which make up a long term short term memory networks are -> prediction, ignoring, forgetting, and selection