n → number of filters

m → number of words in a Sentence

k ⇒ Kernel size of filters

emb_dim → embedding dimension

How does ConvlD work with Word Embedding.?

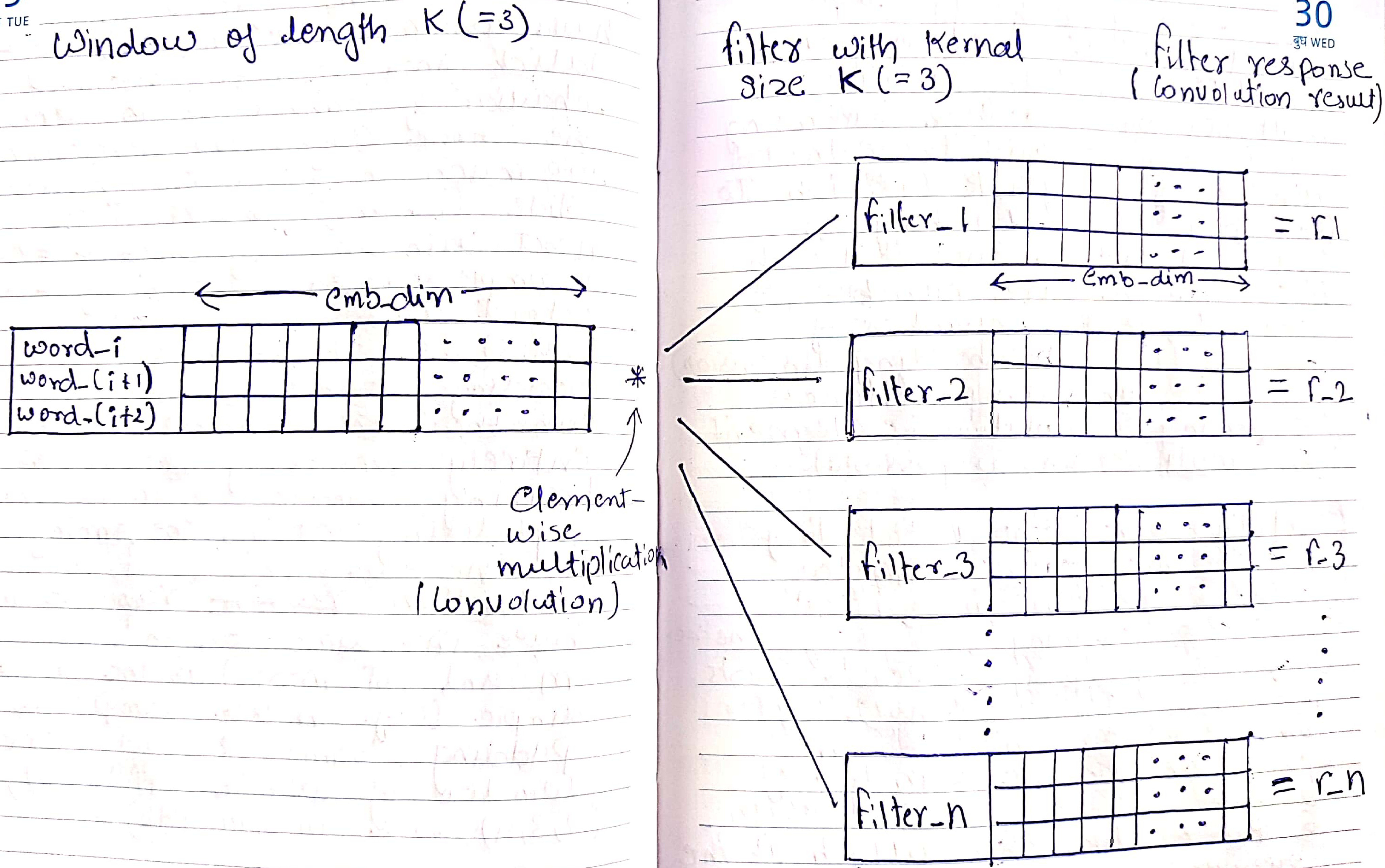
Suppose we have a Sentence Consisting of 'm' words where each word has been represented using word embedding:

Sample Sentence	Feature (i.e. Word embedding)									
word-1								. . . .		
word-2								. . . .		
word-3								. . . .		
word-4								. . . .		
word-5								. . . .		
word-6								. . . .		
.	.	.	.	.	.	.	.	. . . .	.	.
.	.	.	.	.	.	.	.	. . . .	.	.
.	.	.	.	.	.	.	.	. . . .	.	.
.	.	.	.	.	.	.	.	. . . .	.	.
.	.	.	.	.	.	.	.	. . . .	.	.
.	.	.	.	.	.	.	.	. . . .	.	.
word_m-1								. . . .		
word_m								. . . .		

Now we would like to apply 1D convolution layer consisting of 'n' different filters with Kernal size of 'K' on this data. To do so, sliding windows of length 'K' are extracted from the data and then each filter is applied on each of those extracted windows.

Here is an illustration of what happens.

Here, I have assumed K=3 and removed the bais parameter of ~~of e~~ each filter for simplicity.


Window of length K (=3)
filter with Kernal size K (=3)
filter response (Convolution result)
emb_dim
word_i
word_(i+1)
word_(i+2)
*
Element-wise multiplication (Convolution)
filter_1
= r_1
emb_dim
filter_2
= r_2
filter_3
= r_3
filter_n
= r_n

As we can see in the figure above, the response of each filter is equivalent to the result of its convolution (i.e element-wise multiplication and then summing all the results) with the extracted window of length K (i.e ith to (i+K-1)-th words in the given sentence). further, note that each filter has the same number of channels as the number of feature (i.e word-embedding dimension) of the training sample (hence performing convolution, i.e element-wise multiplication, is possible).

Essentially, each filter is detecting the presence of a particular feature of pattern in a LOCAL window of training data (e.g whether a couple of specific words exists in this window or not). After all the filters have been applied on all the window of length 'K' we would have an output of like this which is the result of convolution.

26
शनि SAT

Filter response	Filter-1	Filter-2	. . . .	Filter-n
window_1			. . . .	
window_2			. . . .	
window-3			. . . .	
window_4			. . . .	
window_5			. . . .	
window-6			. . . .	
window_7			. . . .	
window-8			. . . .	
.	.	.	. . . .	.
.	.	.	. . . .	.
.	.	.	. . . .	.
.	.	.	. . . .	.
.	.	.	. . . .	.
.	.	.	. . . .	.
window_(m-k+1)			. . . .	

27
रवि SUN

As we can see, there are $m-k+1$ windows in the figure since we have assumed that the padding = 'Valid' and Stride=1 (default behavior of Conv1D layer in Keras). The Stride argument determines how much the window should slide (i.e shift) to extract the next window (e.g in our ~~example~~ example above, a stride of 2 would extract windows of words: (1,2,3), (3,4,5), (5,6,7)... instead). The padding argument determines whether the window should entirely consists of the words in training sample of there should be padding at the beginning and at the end: this way, the convolution ~~responce~~ response may have the same length (i.e. m and not $m-k+1$) as the training sample (e.g. in our example above, padding = 'same' would extract window of words: (PAD, 1, 2), (2, 3, 4), ..., ($m-2$, $m-1$, PAD)).

number of parameters in the 1-D convolution layer is equal to :

num_filters * (kernal_size * n_features)
+ one_bias_per_filter

= n * (K * emb_dim) + n

for example, if, n=32, m=20, k=3,
emb_dim = 100

= n * (K * emb_dim) + n

= 32 * (3 * 100) + 32

= 9632