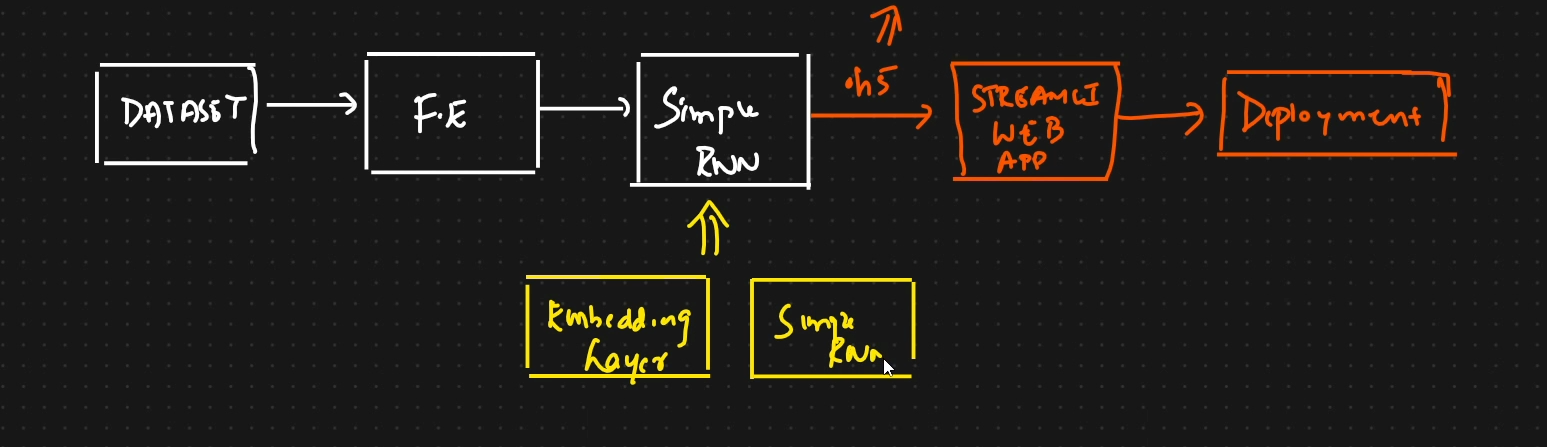
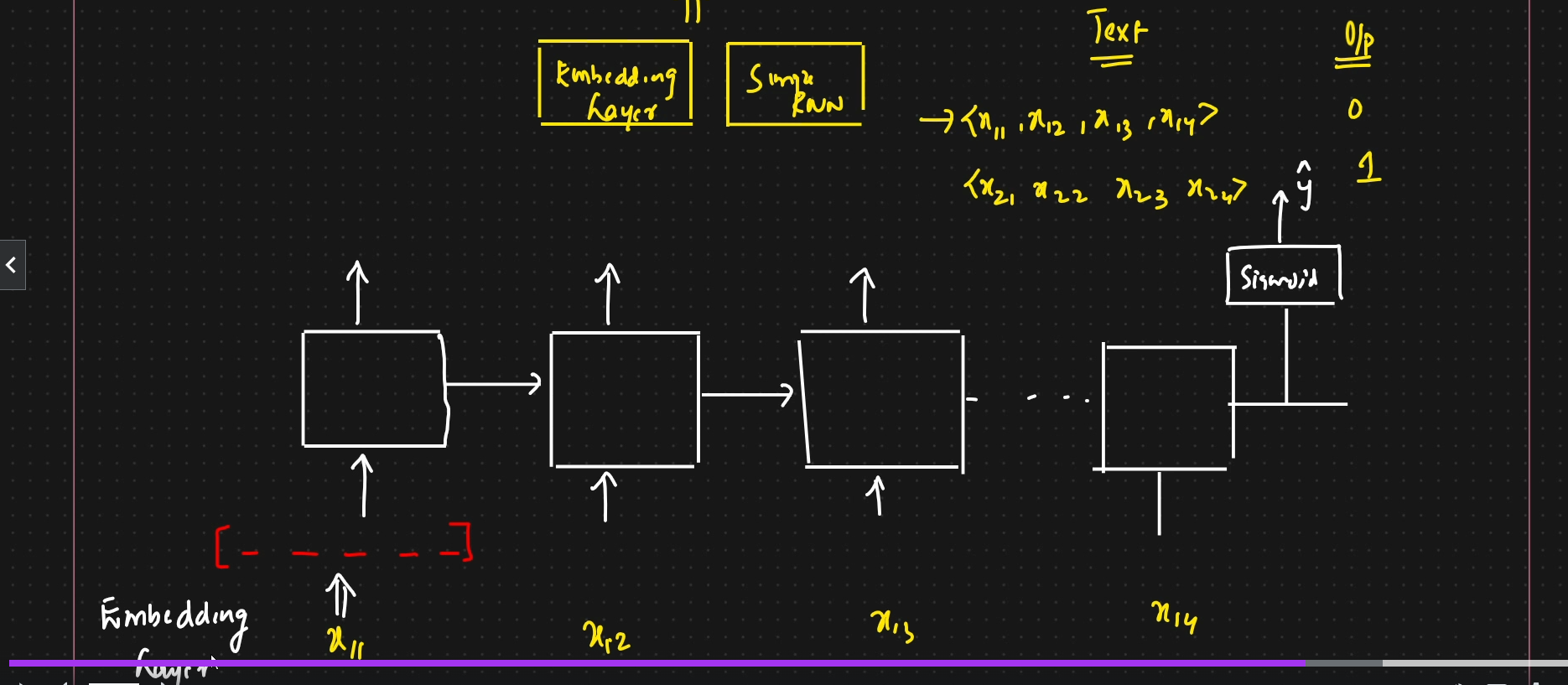
End to End Deep learning project Using simple Rnn

1. Ibdb dataset
2. Sentiment analysis : Positive , Negative : Binary Classification



Words are passed one by one to RNN, and they are before giving to model converted to vectors using embedding layer. It utilizes word2vec , word embeddings to convert it to vectors.



In word embeddings, vocab size**Vocabulary Size (voc\_size):**

* **Not Predefined**: voc\_size=10000 doesn’t mean it has 10,000 words predefined. Instead, it sets an upper limit on the number of distinct words the tokenizer can encode. It’s a way of telling the model, “You can map up to 10,000 different words.

from tensorflow.keras.preprocessing.text import one\_hot

and feature representation size . i.e features\_dimension =300(hidden adjectives that the neural network decides on window size), that is important,

## in one implementation named experiments.ipynb:

We first convert the sentence to one hot encoded , index format

The indexed format is then padded and then converted to vectors using embeddings layer(which utilizes word2vec in the background)

If dim =10, it will generate 10 vectors for each word.

Actual project start

##

1. We load imbd dataset by passing max\_features(vocab\_Size=10000) ,with a train test split.
2. When we see X\_train[0], we see a lot if indexes, which are nothing but a one hot representation of every word for 1 sentences, and all sentences, like we saw earlier
3. Y\_train[0] is 0 or 1, a positive or a negative sentiment
4. Now we decide a max\_len =500 ( this is max sent length) and then do a pre or post padding for each sentence.( in this dataset the words are already one hot encoded)
5. 2 things, max\_length( is len of sentences that could be maximum one) and dim=10 ( is len of output vector of embedding layer.
6. Then create model and train on data X\_train ,Y\_train data
7. Save the model
8. Later create 3 functions ( 1 to preprocess a new text,encode to review( one hot) the pad)
9. Second is decoding the words later post predicting
10. So we have a predict function as well that will predict the encoded stuff