

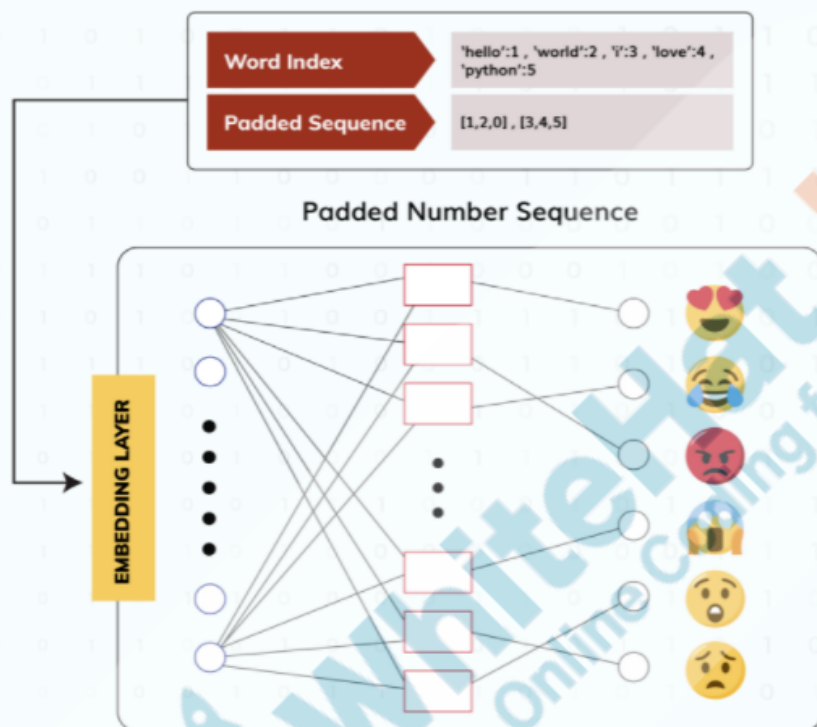


Topic	DEFINE MODEL FOR SENTIMENT ANALYSIS	
Class Description	Analyzing the sentiment related to text by defining a machine learning model.	
Class	PRO C115	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> • Introduction to the CNN model for text analysis. • Introduction to Embedding, Conv1D and LSTM layer. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone • Student Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen 	
Class structure	Warm-Up Teacher-Led Activity 1 Student-Led Activity 1 Wrap-Up	10 mins 10 mins 20 mins 05 mins
Credit & Permissions:	Tensorflow by Google Brain team Keras by Google Engineer Francois Chollet Numpy by The NumPy community Kaggle: Emotion Detection from Text	
WARM-UP SESSION - 10 mins		
<div>  </div> Teacher Starts Slideshow		

Slide 1 to 5 Refer to speaker notes and follow the instructions on each slide.	
Teacher Action	Student Action
<p>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</p> <p>Can you tell me what we learned in the previous class?</p> <p><i>Note: Encourage the student to give answers and be more involved in the discussion.</i></p> <p>Amazing!</p> <p>Till now we have preprocessed the text data to visualize it using Pandas.</p> <p>We also learned how we can convert text into a number sequence.</p>	<p>ESR: Hi, thanks! Yes, I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
WARM-UP QUIZ Click on In-Class Quiz	
<div> <div>Continue WARM-UP Session</div> <div>Slide 6 to 12</div>  </div>	
<p>Now that we have our text in the form of numbers, we'll now define a CNN model for sentiment prediction.</p> <p>As text and images are different, we need different layers for text feature extraction and classification.</p> <p>Today we are going to learn about new layers of the CNN model. i.e Embedding layer, Conv1D and LSTM layer.</p>	<p>ESR: Yes</p>

Are you excited?

Defining Model For Sentiment Analysis



Teacher Ends Slideshow

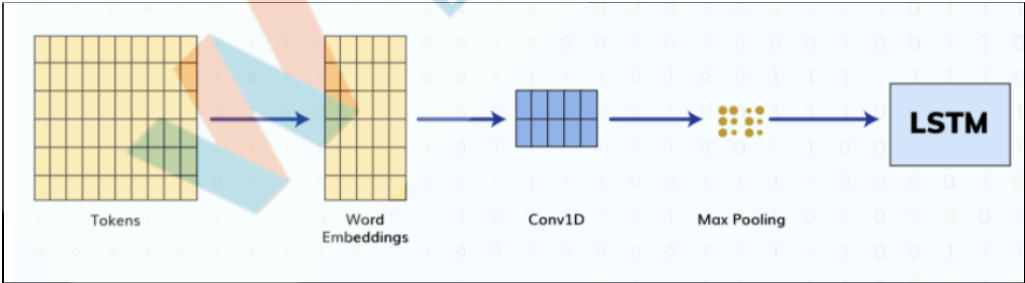


TEACHER-LED ACTIVITY - 10 mins

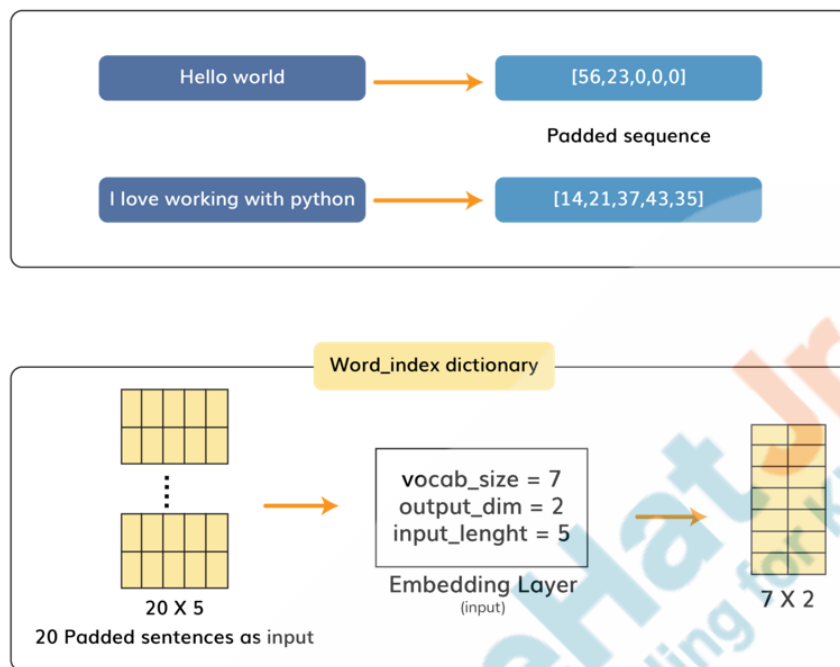
Teacher Initiates Screen Share

ACTIVITY

- Introduction to embedding layer, Conv1D layer and LSTM layer.
- Define the CNN model for text analysis.

Teacher Action	Student Action
<p>Note: Open Teacher activity 2 for the boilerplate code and run all the cells.</p> <p>We have already converted our text dataset into number sequences. These padded sequences will be used to train the CNN model.</p> <p>The next step would be to convert these lists of padded sequences and labels into Numpy arrays to give it to the Embedding layer as shown below:</p>	
<pre>import numpy as np training_padded = np.array(training_padded) training_labels = np.array(training_labels)</pre>	
<p>From tensorflow.keras import the Sequential class to define a Sequential CNN model. Also, to extract features and for classification, different layers are imported such as Embedding, LSTM, Dense, Conv1D, Dropout, Maxpooling1D.</p>	
	
<p>What were the two important steps to pre-process images?</p> <p>Great!</p> <p>To extract the features from text data we will be using the following layers:</p>	<p>ESR: Extract features and classification</p>

<ol style="list-style-type: none"> 1. Embedding layer: The embedding layer is the very first layer used for creating ML models for NLP. It requires that the input data be tokenized. It converts each word into a fixed size array depending upon different features. 2. Conv1D: The Conv1D layer is used to extract features from the text data. A kernel is used to run over word embeddings to give a Feature array. 3. Max Pooling layer: The Max Pooling layer is used to extract the most prominent feature from the feature array. 	
<p>The Embedding layer uses the Word Embedding approach for representing words using floating-point numbers depending upon different features; it gives an array for feature learning.</p> <p>Do you remember how the word_index dictionary is created?</p> <p>Great!</p> <p>So now using these values we had created padded sequences. In these sequences, the words are identified by unique numbers.</p>	<p>ESR: Yes. Each word was assigned with a numerical value</p>



The **Embedding layer** is defined as the first hidden layer of a neural network. It must specify three arguments:

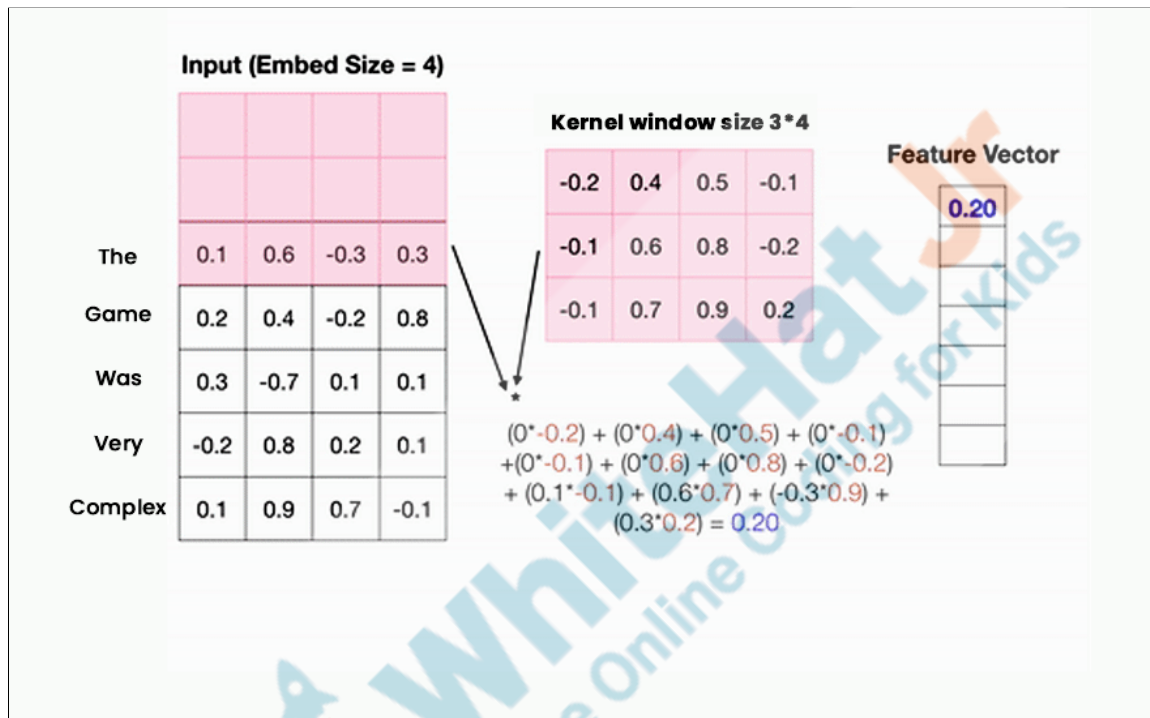
Teacher activity 3: Embedding layer

- **vocab_size:** It is the number of unique words in training data. For example, we have 7 unique words in 2 sentences, then the size of the vocabulary would be 7 unique words.
- **output_dim:** is the size of the output array in which words will be embedded.
- **input_length:** It is the length of input sequence. It is the same as `max_len` used in padding the sequences. Here the `input_length` would be 5.

If we have 20 sentences with 7 unique words where the padding is done with `max_length` of 5, this gives us a reduced array of `7 x 2`.

<p>Dropout may be implemented on any or all hidden layers in the network as well as the visible or input layer. It is used to avoid errors while training the model.</p>	
<pre>from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Embedding, LSTM, Dense from tensorflow.keras.layers import Conv1D, Dropout, MaxPooling1D model = tf.keras.Sequential([Embedding(vocab_size, embedding_dim, input_length=max_length), Dropout(0.2),</pre>	
<p>Conv1D layer: In the Conv1D layer, the convolution kernel only moves along a single axis, thus it is applied to linear data like text. A convolution kernel slides over an embedding array to give a feature array.</p> <p>In case of text, a convolutional kernel is applied at embeddings for multiple words. The kernel will no longer be square, instead, it will be a rectangle. Here we are considering four features so the kernel used has 3X4 size.</p> <p>Teacher activity 4: Convolution 1D layer</p>	
<p>Convolution over Word Sequences:</p> <p>To process an entire sequence of words, these kernels will slide down a list of word embeddings. This is called a 1D convolution because the kernel is moving in only one dimension.</p> <p>A single kernel will move one by one down a list of input embeddings. The resultant output will be a feature array that contains about as many values as there were input embeddings.</p> <p>When the kernel is run over the array received from the embedding layer, we get a single output value as shown below.</p>	

Note: Refer to additional activities of C112 for matrix multiplication.
The image below is for Teacher reference. It shows how a kernel works in case of Conv1D layer to give feature array.



Do you remember the parameters we needed for the Conv2D layer?

Convolution 1D layer too must have these arguments:

1. **filters:** Defines the number of filters required to perform the convolution operation.
2. **kernel_size:** Defines the size of the kernel window.
3. **activation:** Defines the activation function for the Conv1D layer.

ESR: filters, kernel size and activation function.


```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.layers import Conv1D, Dropout, MaxPooling1D

model = tf.keras.Sequential([
    Embedding(vocab_size, embedding_dim, input_length=max_length),
    Dropout(0.2),
    Conv1D(filters = 256, kernel_size = 3, activation = "relu"),
```

Max Pooling layer:

A convolutional operation produces a **feature array** that can represent local features in sequences of word embeddings.

Identify the significant words from the following sentence:

‘ It was a tough fight, I managed to win’.

ESR: tough, win

Yes!

These words may appear anywhere in the sentence, it signifies fight and victory. These are considered high-level features that are extracted by the **Max Pooling** layer.

Max Pooling layer retains only maximum value in the feature array which is the most useful, local feature. In the above feature array, three maximum values are considered as shown for pool size 3.

Feature Array

0.20
0.53
-0.66
0.35
1.57
1.76
0.72

Max pooling
(pooling size = 3)

1.57
1.76
0.72

Pooling layers **subsample** their input. Thus, the output after the max-pooling layer would be a **feature map** containing the **most prominent features** of the previous feature map.

Max pooling layer takes parameter **pool_size** to select the prominent features.

[Teacher Activity 5: Max Pooling layer](#)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.layers import Conv1D, Dropout, MaxPooling1D

model = tf.keras.Sequential([
    Embedding(vocab_size, embedding_dim, input_length=max_length),
    Dropout(0.2),

    Conv1D(filters = 256, kernel_size = 3, activation = "relu"),
    MaxPooling1D(pool_size = 3),

    Conv1D(filters = 128, kernel_size = 3, activation = "relu"),
    MaxPooling1D(pool_size = 3),
```

Long Short Term Memory (LSTM):

Long Short Term Memory Network is another type of Neural Network that allows relevant information to persist.

Let me check how much you can remember things. I am going to tell you a story in a few sentences. I will be asking you questions related to the story.

There were two friends who were walking across a desert. They kept on walking until they found an oasis, where they decided to take a bath. One of them got stuck in the mire and started drowning, but his friend saved him. After he recovered from the near drowning, he wrote on a stone “**Today my best friend saved my life**”.

Tell me some of the words I used in the story?

ESR: Varied

Great!

So you were able to recollect the words which were related or relevant to the story.

Similarly, LSTM remembers the previous information and uses it for processing the current input.

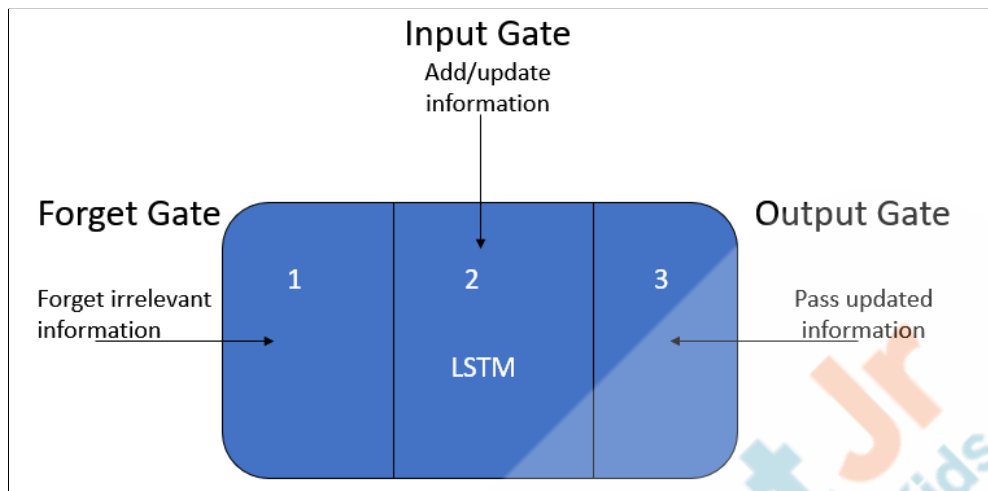
The LSTM consists of three parts, as shown in the image below and each part performs an individual function.

The **first part** or **input gate** chooses whether the information coming from the previous input is to be remembered. If it is irrelevant, it can be forgotten.

The **second part** or **output gate** tries to learn new information from the input to this cell.

At last, in the **third part** or **forget gate** passes the updated information from the current timestamp to the next input.

A common LSTM unit also called a **cell** contains an **input gate**, an **output gate** and a **forget gate**.



The **cell** remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

The input propagates forwards and backward through the LSTM layer and then concatenates with the outputs.

This makes LSTM an efficient network as irrelevant data is discarded and only keeps information relevant for training the model. We then fit it into a Dense neural network to do classification.

[Teacher activity 6: Reference: LSTM layer](#)

Define LSTM with 128 output units that return 128 features depending on the three timestamps by three gates of cells.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.layers import Conv1D, Dropout, MaxPooling1D

model = tf.keras.Sequential([
    Embedding(vocab_size, embedding_dim, input_length=max_length),
    Dropout(0.2),

    Conv1D(filters = 256, kernel_size = 3, activation = "relu"),
    MaxPooling1D(pool_size = 3),

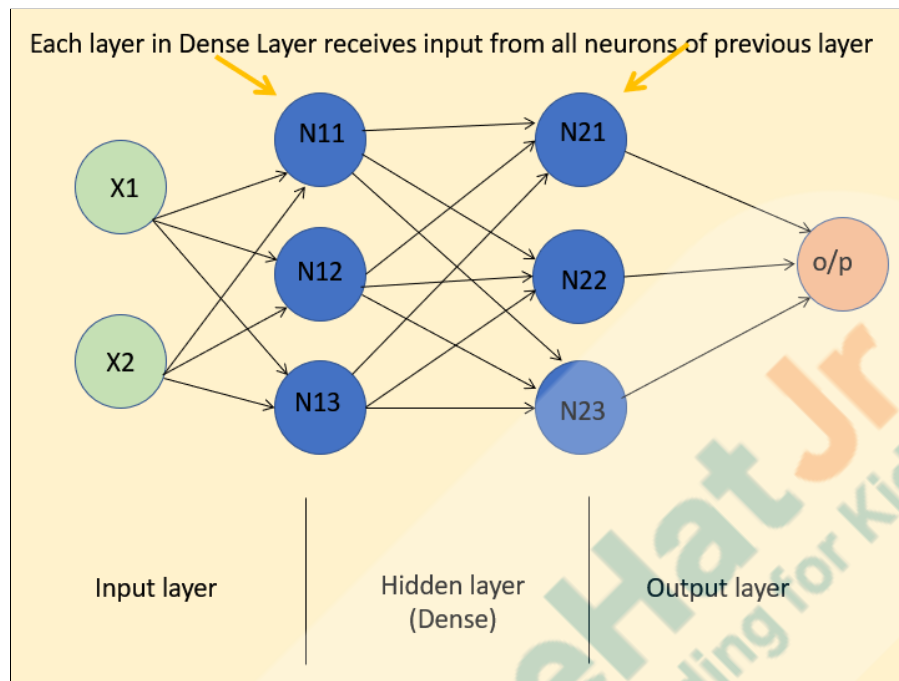
    Conv1D(filters = 128, kernel_size = 3, activation = "relu"),
    MaxPooling1D(pool_size = 3),

    LSTM(128),
```

Dense layer:

Now, in the dense layer, each neuron is connected with the neuron in its preceding layer.

This layer is used at the final stage of CNN after LSTM to perform classification.



So, our first Dense layer would have **128** neurons and a **relu** activation function.

We have a Dropout layer to avoid errors.

Next, we are defining two more dense layers with **relu** and **softmax** activations.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.layers import Conv1D, Dropout, MaxPooling1D

model = tf.keras.Sequential([
    Embedding(vocab_size, embedding_dim, input_length=max_length),
    Dropout(0.2),

    Conv1D(filters = 256, kernel_size = 3, activation = "relu"),
    MaxPooling1D(pool_size = 3),

    Conv1D(filters = 128, kernel_size = 3, activation = "relu"),
    MaxPooling1D(pool_size = 3),

    LSTM(128),

    Dense(128, activation = "relu"),
    Dropout(0.2),
    Dense(64, activation = "relu"),
    Dense(6, activation = "softmax")])
```

Now that the model is created we need to compile, fit and save the model.

That was interesting!

Are you excited to define and test the model?

ESR: Yes

Teacher Stops Screen Share

So now it's your turn.

Please share your screen with me.

Teacher Starts Slideshow




Slide 13 to 14

Refer to speaker notes and follow the instructions on each slide.

We have one more class challenge for you.

Remember in the previous class we trained the model in

<p>teachable machines and then we had downloaded and tested the model locally?</p> <p>After training your model on Google Colab we will test it on our system.</p> <p>Let's try. I will guide you through it.</p>	
<div> <div>Teacher Ends Slideshow</div>  </div>	
STUDENT-LED ACTIVITY - 20 mins	
<ul style="list-style-type: none"> • Ask the student to press the ESC key to come back to the panel. • Guide the student to start Screen Share. • The teacher gets into Full Screen. 	
Student Initiates Screen Share	
<p style="text-align: center;"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Create a CNN model for text sentiment analysis. • Compile, fit and save the model. • Predict the sentiment from any random text. • Download the .h5 file and run it locally in your system to check the performance of your model. 	
Teacher Action	Student Action
<p><i>Guide the student to open the boilerplate code Student Activity 1 and run all the code.</i></p> <p>Note: The student will perform the same activities as done by the teacher.</p> <p>Note: Please refer to Teacher Activity 2 to follow through and guide the student to do the activities.</p>	
<p>Guide the student to perform the activities as mentioned below:</p>	

1. Define the model by importing layers from **tensorflow.keras**.
2. Define Embedding layer, Conv1d, LSTM and Dense layers.
3. After defining layers, compile the model as performed in class 113.

Do you remember the parameters taken by the **compile()** function?

ESR: loss, optimizer and metrics

4. Similarly, compile the model using **loss**, **optimizer** and **metrics** parameters first.

```
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 16)	160000
dropout (Dropout)	(None, 100, 16)	0
conv1d (Conv1D)	(None, 98, 256)	12544
max_pooling1d (MaxPooling1D)	(None, 32, 256)	0
conv1d_1 (Conv1D)	(None, 30, 128)	98432
max_pooling1d_1 (MaxPooling1D)	(None, 10, 128)	0
lstm (LSTM)	(None, 128)	131584
dense (Dense)	(None, 128)	16512
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dense_2 (Dense)	(None, 6)	390

=====
 Total params: 427,718
 Trainable params: 427,718
 Non-trainable params: 0
 =====

<p>Thus the model is being compiled we got the total parameters and the trainable parameters as output from these layers.</p> <p>Output data from CNN is also a 3D array consisting of batch_size, height, width.</p> <p>So here each layer returns an array. As the output is not received in batches, batch_size='None'.</p> <p>Also, each layer gives parameters that are used to train the model for prediction.</p> <p>5. After compiling we have to train the model. Can you suggest what will be the next step?</p> <p>Great!!!</p> <p>6. Define the number of epochs to 30 and then train the model using the fit() function.</p>	<p>ESR: Using the fit() function</p>
<pre>num_epochs = 30 history = model.fit(training_padded, training_labels, epochs=num_epochs, verbose=2)</pre>	
<pre>Epoch 1/30 500/500 - 26s - loss: 1.5039 - accuracy: 0.3449 - 26s/epoch - 52ms/step Epoch 2/30 500/500 - 21s - loss: 0.8811 - accuracy: 0.6001 - 21s/epoch - 43ms/step Epoch 3/30 500/500 - 21s - loss: 0.5955 - accuracy: 0.7616 - 21s/epoch - 43ms/step Epoch 4/30 500/500 - 22s - loss: 0.4234 - accuracy: 0.8598 - 22s/epoch - 44ms/step Epoch 5/30</pre>	
<p>Save the model using the save() function.</p>	
<pre>model.save("Text_Emotion.h5")</pre>	
<p>Let's test the model.</p> <p>1. Create a list of sentences and write some random sentences for testing.</p>	

2. Once this is done we need to tokenize these sentences by using the **text to sequence()** method. Perform padding using the **pad_sequences()** method.
3. Test the model using **predict()** method. Find the result using the **argmax()** method. The **argmax()** method will give the label of the class with the highest probability in the result.

```

sentence = ["I am happy to meet my friends. We are planning to go a party.",
            "I had a bad day at school. i got hurt while playing football"]

sequences = tokenizer.texts_to_sequences(sentence)

padded = pad_sequences(sequences, maxlen=max_length,
                       padding=padding_type, truncating=trunc_type)

result=model.predict(padded)

predict_class = np.argmax(result, axis=1)
predict_class

# {"anger": 0, "fear": 1, "joy": 2, "love": 3, "sadness": 4, "surprise": 5}

array([2, 4])
  
```

It is returning the class labels for respective sentences. As you can see the prediction of your model is almost right. Let's test the model locally on your device:

1. As we saved the model, we have a .h5 file saved in our **colab directory**.
2. **Download** the file and **save** it in a folder.
3. Open [Student Activity 2](#) and download the boilerplate code. Paste it in the same folder.
4. Now open the boilerplate code in VS code.

```

Text_Sentiment_Boilerplate.py > ...
1  import tensorflow.keras
2  import numpy as np
3  from tensorflow.keras.preprocessing.text import Tokenizer
4  from tensorflow.keras.preprocessing.sequence import pad_sequences
5
6  sentence = ["I am happy to meet my friends. We are planning to go to a party.",
7             "I had a bad day at school. i got hurt while playing football"]
8
9  # Tokenization
10
11 # Create a word_index dictionary
12
13 # Padding the sequence
14
15 # Define the model using .h5 file
16
17 # Test the model
18
19 # Print the result

```

5. Here we have to predict the sentiments of the given sentences written in the list called a **sentence**.

Note: The student can write any sentence.

6. Perform tokenization and padding on these sentences.

```

# Tokenization
tokenizer = Tokenizer(num_words=10000, oov_token='<OOV>')
tokenizer.fit_on_texts(sentence)

# Create a word_index dictionary
word_index = tokenizer.word_index
sequence = tokenizer.texts_to_sequences(sentence)
print(sequence[0:2])

# Padding the sequence
padded = pad_sequences(sequence, maxlen=100,
                        padding='post', truncating='post')
print(padded[0:2])

```

7. Use a **.h5** file to load the model and use the **predict()** method to predict the sentiment of the text.

```
# Define the model using .h5 file
model=tensorflow.keras.models.load_model('Text_Emotion.h5')

# Test the model
result=model.predict(padded)
print(result)

# Print the result
predict_class = np.argmax(result, axis=1)
print(predict_class)

# {"anger": 0, "fear": 1, "joy": 2, "love": 3, "sadness": 4, "surprise": 5}
```

8. Go to the command prompt and traverse the working folder as we did in class 110.
9. Create a virtual environment (**Windows/Mac**) using **python -m venv <name_of_the_environment>** for testing the model.

10. Activate the virtual environment using the following command:

```
<name_of_the_environment>\Scripts\activate
```

*Run the command to create a virtual environment with the name **"Text_Sentiment_Analysis"**. The environment is user-defined. We can keep the name as we want relevant to our project.*

```
C:\whitehat jr\PRO C115\Text_Sentiment_Analysis>Scripts\activate
```

11. Now, run the desired python file to check the output. You can run the code stepwise and check the output one by one.

```
(Text_Sentiment_Analysis) C:\whitehat jr\PRO C115>python Text_Sentiment.py
```

Tokenized and padded sequences:

```
[[2, 5, 6, 3, 7, 8, 9, 10, 11, 12, 3, 13, 4, 14], [2, 15, 4, 16, 17, 18, 19, 2, 20, 21, 22, 23, 24]]
[[ 2  5  6  3  7  8  9 10 11 12  3 13  4 14  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0]
 [ 2 15  4 16 17 18 19  2 20 21 22 23 24  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0]]
```

Result and prediction of the sentiment using labels:

```
[[1.26573211e-03 1.21657131e-03 2.26438437e-02 6.40540496e-02
 1.04617124e-04 9.10715103e-01]
 [4.97628838e-01 7.16012642e-02 3.41708571e-01 1.01922369e-02
 7.38635287e-02 5.00548165e-03]]
[5 0]
```

Analyzing output:

As you can see the first array has six parameters since we have **six classes**.

Do you remember we had used the **argmax()** method?

Yes!! we had used the **argmax()** method for finding the maximum probability related to a particular class?



Here, you can see that after using the **argmax()** method we got output as [5 0].




5 corresponds to **surprise** and 0 corresponds to **anger**.

Since the ML model cannot be 100% correct we may not get the expected predictions always.

Great work!! This way we can create a Machine Learning model to predict the sentiments.

ESR: Yes, we had used it to find the class related to testing data

You can stop sharing your screen now.	
Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 05 mins	
<div> <div>Teacher Starts Slideshow</div> <div>Slide 15 to 19</div> <div></div> </div>	
<p>Activity details:</p> <p>Following are the WRAP-UP session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Revise the current class activities. • Discuss the quizzes. 	
<div> <div>WRAP-UP QUIZ</div> <div>Click on In-Class Quiz</div> </div>	
<div> <div>Continue WRAP-UP Session</div> <div>Slide 20 to 25</div> <div></div> </div>	
<p>Activity Details</p> <p>Following are the session deliverables:</p> <ul style="list-style-type: none"> • Explain the facts and trivia • The next class challenge is to find how python is used for creating websites • Project for the day • Additional Activity (Optional) 	
<p>FEEDBACK</p> <ul style="list-style-type: none"> • Appreciate and compliment the student for trying to learn a difficult concept. • Get to know how they are feeling after the session. • Review and check their understanding. 	

Teacher Action	Student Action
<p>You get “hats-off” for your excellent work!</p>	<p><i>Make sure you have given at least 2 hats-off during the class for:</i></p> <div> <div>Creatively Solved Activities  +10</div> <div>Great Question  +10</div> <div>Strong Concentration  +10</div> </div>
<p>PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections</p>	
<p>Teacher Clicks</p>	<p>✕ End Class</p>

ACTIVITY LINKS		
Activity Name	Description	Links
Teacher Activity 1	Previous Class Code	https://colab.research.google.com/drive/1-XnOC4oGfKqDT6HGdA1guosGMIO6QWkT?usp=sharing
Teacher Activity 2	Boilerplate Code	https://colab.research.google.com/drive/1sFb2RvAZJIMpdyKXkEK8MazW9HyG0gqx?usp=sharing
Teacher Activity 3	Embedding layer	https://www.tensorflow.org/api_docs/python/tf/keras/layers/Embedding
Teacher Activity 4	Convolution 1D layer	https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv1D
Teacher activity 5	Max Pooling layer	https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool1D
Teacher Activity 6	LSTM layer	https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
Teacher Reference 1	Reference Code	https://colab.research.google.com/drive/1-kuwdZlqnczX1aSsFICzAt1BC2g7P6H2?usp=sharing
Teacher Reference 2	Reference Code (VS Code)	https://github.com/procodingclass/PRO-C115-Reference-Code
Teacher Reference 3	Project	https://s3-whjr-curriculum-uploads.whjr.online/a2f7b916-9e9b-4476-8470-90a4bde2627d.pdf
Teacher Reference 4	Project Solution	https://colab.research.google.com/drive/1T3d8LQizWHHzP9wP_RkgvTyWyIDjFtCt?usp=sharing
Teacher Reference 5	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/7e0118ac-aba0-43e4-a957-e628facd8167.html
Teacher Reference 6	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/7e0118ac-aba0-43e4-a957-e628facd8167.html

		whjr.online/c57b9df6-fb4f-4a2e-a9d7-abf218c1277f.pdf
Student Activity 1	Boilerplate Code	https://colab.research.google.com/drive/1sFb2RvAZJIMpdyKXkEK8MazW9HyG0gqx?usp=sharing
Student Activity 2	Boilerplate Code(VS Code)	https://github.com/procodingclass/PRO-C13-Student-Boilerplate-VS-code-

