# CHAPTER 4
# ANALYSIS AND DESIGN

## 4.1. Analysis

This chapter provides a detailed explanation and discusses one by one about the methods of solving the problem that have been mentioned in the previous chapter. The main object of this purpose is to compare which neural network models have better accuracy between Convolutional Neural Networks (CNN) and Long Short-Term Memory Neural Networks (LSTM) for text sentiment analysis.

### *4.1.1. Dataset*

In this research the dataset was scrapped from zero with the help of *snscrape* python library. By using *snscrape* it is not necessary to create a Twitter developer account because *snscrape* doesn't need acces token and secret token like Tweepy. The query that was used for scrapping are "(good OR hate) (@TheBatman) until:2022-06-05 since:2022-03-05" which is tweet that contain good or hate and tag @TheBatman with the timeframe of March 5th 2022 which is the day the Batman movie being premiered in Indonesia until June 5th 2022.

| Date | User | Tweet |
|---|---|---|
| 2022-06-04 20:10:43+00:00 | JayN_RTSC | @thesuperadi1 @Clay_Mann_ @TomKingTK @BruceSimcosplay @TheBatman @DCBatman Awes |
| 2022-06-04 03:14:21+00:00 | shababy | How come no one told me @TheBatman was this good! @hbomax |
| 2022-06-04 00:45:03+00:00 | _TreDaGr8test | just rewatched @TheBatman and yeaâ€¦yâ€™all tweaking that ðŸ˜© is good |
| 2022-06-02 23:39:21+00:00 | GarretParsons3 | @TheBatman @warnerbros Best Batman ever! Good job. |
| 2022-06-02 20:16:33+00:00 | yomikeyrocks | @Jamiebower fantastic job in #StrangerThings you'd be sooo good as Joker in @TheBatman |
| 2022-06-02 19:58:19+00:00 | Kevin_Stoe | I hate to say it but I think @TheBatman is better than the Nolan Batman Movies. Glad I waited for the |
| 2022-06-02 03:14:38+00:00 | AngRonilloE | @TheBatman a very good movie. I like it ðŸ¤ðŸ□» https://t.co/xtu9cjLsmy |
| 2022-06-01 19:46:43+00:00 | JeffOrstad | @Strikergotsusp1 @McSports14 @TheBatman @WBHomeEnt Morbius, and I assume you mean th |
| 2022-06-01 19:19:36+00:00 | JeffOrstad | @Strikergotsusp1 @McSports14 @TheBatman @WBHomeEnt Wow you must love this garbage to |
| 2022-06-01 19:16:29+00:00 | JeffOrstad | @TheBatman @WBHomeEnt This movie was â€œmehâ€□ at best. My advice would be not to ignor |
| 2022-05-31 22:15:19+00:00 | justicecoming5 | @blksad_1 @TheBatman was terrible slow and boring, he was not a good #Batman |
| 2022-05-31 16:07:00+00:00 | FilmoscopyPod | .@TheBatman how good is it? Is it overrated? Does it overstay it's welcome? We discuss all this on |
| 2022-05-30 23:58:02+00:00 | lilnicnic87 | I know I'm late to the party, I'm just now seeing it on @HBO But I am AMAZED at how good Robert F |
| 2022-05-30 15:52:41+00:00 | razornivek | @dripashr @LuluDirzo @wbpictures @TheBatman @blackadammovie @theflashmovie @aquamanı |
| 2022-05-30 07:36:00+00:00 | AS_Kirklees | COMPETITION: Thanks to @wbpictures win a @TheBatman DVD. To WIN all you need to do is ans |
| 2022-05-29 18:07:54+00:00 | Vkramsay | @Scott20YL @RealWoven @Raagnarok__ @Moonzlight8 @CatemacDavid @MarvelStudios @The |
| 2022-05-28 21:23:55+00:00 | TrentOrSomethin | Finally watched @thebatman and I don't get why people didn't like it. I thought it was really good. The |
| 2022-05-28 16:59:55+00:00 | bricktop1972 | Just watched @TheBatman so much more than I expected #dark and very unstable #Batman Very g |
| 2022-05-28 12:59:14+00:00 | mjnewman87 | Saw @TheBatman movie last night. Dark but overall very, very good. I look forward to the sequel. |
| 2022-05-25 11:30:16+00:00 | MPhoenix66 | @TheDrkPunisher At Best @TheBatman is average 6-7 outta 10. |
| 2022-05-25 06:50:01+00:00 | keepinwithgrays | @poyserno1 @TheBatman @DCComics Could not agree more. Marvel has gone shit with polished |
| 2022-05-24 21:14:12+00:00 | dwalx96 | Todayâ€™s a good Tuesday ðŸ˜□ @mattreevesLA @TheBatman https://t.co/tujnvYRA7e |
| 2022-05-24 17:59:11+00:00 | kingalingalin | @TheBatman Good reminder for me to go listen to this score again |
| 2022-05-24 16:44:32+00:00 | SugarPlum5674 | @redbox @TheBatman I didn't think it was going to be a good movie but Robert Pattinson killed it!!! |
| 2022-05-24 16:26:02+00:00 | JoeyMagidson | Hereâ€™s my weekly Home Movies column at Awards Radar, highlighted by two very good but diffe |
| 2022-05-24 13:18:02+00:00 | DannHerbolario | @wbpictures @warnerbros @TheBatman I still hate you for this. Forget that it's a digital only comme |
| 2022-05-24 05:57:21+00:00 | Fbc2618Frank | @TheBatman Christian Bale from the new Thor movie. He would make a good Joker. #ChristianBale |

**Figure 4.1** Example of the raw dataset that has been taken using the *snscrape* library

First, the dataset needs to be labeled one by one with the help of reviewers with positive and negative. In this research two reviewers were asked to manually label each of the tweet.

| text | review 1 | review 2 |
|---|---|---|
| @thesuperadi1 @Clay_Mann_ @TomKingTK @BruceSimcosplay @TheBatman @DCBatman Awesome cosplay, you make a good catwoman ðŸ˜ˆâ€¦â¬ | positive | positive |
| How come no one told me @TheBatman was this good! @hbomax | positive | positive |
| just rewatched @TheBatman and yeaâ€¦yâ€™all tweaking that ðŸ’© is good | positive | positive |
| @TheBatman @warnerbros Best Batman ever! Good job. | positive | positive |
| @Jamiebower fantastic job in #StrangerThings you'd be sooo good as Joker in @TheBatman | positive | positive |
| I hate to say it but I think @TheBatman is better than the Nolan Batman Movies. Glad I waited for the 4K Blu-ray release! | positive | positive |
| @TheBatman a very good movie. I like it ðŸ¤ŸðŸ» https://t.co/xtu9cjLsmy | positive | positive |
| @Strikergotsusp1 @McSports14 @TheBatman @WBHomeEnt Morbius, and I assume you mean the new Dr Strange movie, were both not very good. | negative | negative |
| @Strikergotsusp1 @McSports14 @TheBatman @WBHomeEnt Wow you must love this garbage to keep countering everyone. Sorry it wasnâ€™t any good. | negative | negative |
| @TheBatman @WBHomeEnt This movie was â€œmehâ€ at best. My advice would be not to ignore all the negative criticism this movie is getting. Itâ€™s just not that good of a film. | negative | negative |
| @blksad_1 @TheBatman was terrible slow and boring, he was not a good #Batman | negative | negative |
| .@TheBatman how good is it? Is it overrated? Does it overstay it's welcome? We discuss all this on today's episode! #TheBatman #podcasts #RobertPattinson #podcasting #Google #apple #Applepodcasts #comicbooks #Batman #PodcastRecommendations @Apple @Google https://t.co/NzF5XzmJd1 | negative | negative |
| I know I'm late to the party, I'm just now seeing it on @HBO But I am AMAZED at how good Robert Pattinson was as @TheBatman ðŸ˜□ðŸ»ðŸ˜□ðŸ»ðŸ˜□ðŸ» https://t.co/Jw0xToD9mW | positive | positive |
| @dripashr @LuluDirzo @wbpictures @TheBatman @blackadammovie @theflashmovie @aquamanmovie Well Marvel has also some good villains but still Marvel wins over DC and are ownin them with every movie DCEU is just lame | negative | negative |
| COMPETITION: Thanks to @wbpictures win a @TheBatman DVD. To WIN all you need to do is answer this question: 1. Who plays Bruce Wayne in the 2022 The Batman movie? | | |

**Figure 4.2** Show the reviewed and labeled tweet from two reviewers

After the dataset has been reviewed, I need to find the kappa value of the dataset to determine if the dataset that I will use is worth using or not.

**Table 4.1.** Kappa Value Table

| | **Reviewer 2** | | |
|---|---|---|---|
| **Reviewer 1** | ***Positive*** | ***Negative*** | ***Total*** |
| ***Positive*** | 1593 | 87 | 1680 |
| ***Negative*** | 118 | 152 | 270 |
| ***Total*** | 1711 | 239 | 1950 |

From the table above it is shown there are 1950 total of dataset that have been successfully scrapped, from reviewer 1 and reviewer 2 both agree that there are 1593 positive data and there are 87 where reviewer 1 believe the tweets are positive but reviewer 2 believe the tweets are negative. And then there are 118 tweets where reviewer 1 believes the tweets are negative but reviewer 2 said otherwise and lastly reviewer 1 and reviewer 2 agree that there are 152 negative tweets. After that, I need to find the Observed agreement which is a state where both reviewers have the same agreement.

$$O_a = \frac{(P_a + N_a)}{T_o} = \frac{1593 + 152}{1950} = 0{,}894871795$$

$P_a$ is the positive agreement, $N_a$ is the negative agreement, and $T_o$ is the total of the dataset.

After that the agreement of chance are needed, to find the agreement of chance I can use

$$A_c = \left(\frac{P_1}{T_o}\right) * \left(\frac{P_2}{T_o}\right) + \left(\frac{P_3}{T_o}\right) * \left(\frac{P_4}{T_o}\right) = \left(\frac{1711}{1950}\right) * \left(\frac{1680}{1950}\right) + \left(\frac{239}{1950}\right) * \left(\frac{270}{1950}\right) = 0{,}772915$$

And finally, after finding the observed agreement and the agreement of chance I can find the kappa value using

$$K = \frac{O_a - A_c}{1 - A_c} = \frac{(0{,}894871785 - 0{,}772915)}{(1 - 0{,}772915)} = 0{,}537053122$$

From the calculation above, it can be concluded that the dataset is of moderate agreement. And lastly, after finding the kappa value, the non-agreement between 2 reviewers will be deleted.

| text | sentiment |
|---|---|
| @thesuperadi1 @Clay_Mann_ @TomKingTK @BruceSimcosplay @TheBatman @DCBatman Awesome cosplay, you make a good catwoman ðŸ□ˆâ€□â¬› | positive |
| How come no one told me @TheBatman was this good! @hbomax | positive |
| just rewatched @TheBatman and yeaâ€¦yâ€™all tweaking that ðŸ'© is good | positive |
| @TheBatman @warnerbros Best Batman ever! Good job. | positive |
| @Jamiebower fantastic job in #StrangerThings you'd be sooo good as Joker in @TheBatman | positive |
| I hate to say it but I think @TheBatman is better than the Nolan Batman Movies. Glad I waited for the 4K Blu-ray release! | positive |
| @TheBatman a very good movie. I like it ðŸ¤ŸðŸ□» https://t.co/xtu9cjLsmy | positive |
| @Strikergotsusp1 @McSports14 @TheBatman @WBHomeEnt Morbius, and I assume you mean the new Dr Strange movie, were both not very good. | negative |
| @Strikergotsusp1 @McSports14 @TheBatman @WBHomeEnt Wow you must love this garbage to keep countering everyone. Sorry it wasnâ€™t any good. | negative |
| @TheBatman @WBHomeEnt This movie was â€œmehâ€□ at best. My advice would be not to ignore all the negative criticism this movie is getting. Itâ€™s just not that good of a film. | negative |
| @blksad_1 @TheBatman was terrible slow and boring, he was not a good #Batman | negative |
| .@TheBatman how good is it? Is it overrated? Does it overstay it's welcome? We discuss all this on today's episode! #TheBatman #podcasts #RobertPattinson #podcasting #Google #apple #Applepodcasts #comicbooks #Batman #PodcastRecommendations @Apple @Google https://t.co/NzF5XzmJd1 | negative |
| I know I'm late to the party, I'm just now seeing it on @HBO But I am AMAZED at how good Robert Pattinson was as @TheBatman ðŸ'□ðŸ□»ðŸ'□ðŸ□»ðŸ'□ðŸ□» https://t.co/Jw0xToD9mW | positive |
| @dripashr @LuluDirzo @wbpictures @TheBatman @blackadammovie @theflashmovie @aquamanmovie Well Marvel has also some good villains but still Marvel wins over DC and are ownin them with every movie DCEU is just lame | negative |

**Figure 4.3** shows the dataset that were ready to be used.

### *4.1.2. Preprocessing*

After preparing the dataset, the next step is preprocessing the data. Preprocessing the dataset is an important step in any machine learning project, including sentiment analysis using neural networks. Preprocessing helps to ensure that the data is in a format that the model can learn from efficiently and effectively.

There are a few specific reasons why preprocessing is important for sentiment analysis using neural networks:

1. Removing noise and unnecessary information: The raw data may contain a lot of irrelevant or redundant information that could hinder the model's ability to learn effectively. Preprocessing helps to remove this noise and select only the most relevant and important data for the model to learn from.
2. Handling missing or incomplete data: It is not uncommon for real-world datasets to contain missing or incomplete data. Preprocessing can help to identify and handle missing or incomplete data in a way that does not negatively impact the model's performance
3. Normalizing and scaling the data: Neural networks can be sensitive to the scale of the input data. Preprocessing can help to normalize and scale the data to a consistent range, which can improve the model's performance.
4. Encoding categorical data: Categorical data, such as text data, needs to be encoded in a way that the model can understand. Preprocessing can help to encode this data in a way that is suitable for the model.

Overall, preprocessing the dataset is an important step in preparing the data for sentiment analysis using neural networks. It helps to ensure that the data is clean, consistent, and in a format that the model can learn from effectively.

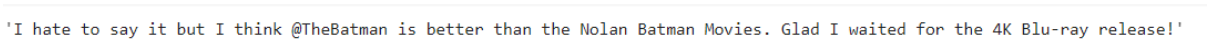


**Figure 4.1** Unprocessed dataset

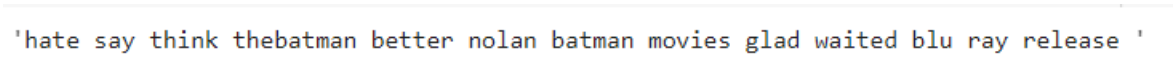


**Figure 4.2** Preprocessed dataset

### *4.1.3. GloVe Word Embedding*

*GloVe* (short for "Global Vectors for Word Representation") is a word embedding method for neural networks that has gained a lot of popularity in recent years. Word embeddings are a way to represent words as numeric vectors, which can then be fed into a neural network as input. The goal of word embeddings is to capture the meaning of words in a way that is more efficient and effective than traditional methods such as one-hot encoding.

The *GloVe* algorithm is trained on a large dataset of co-occurring words, and it learns to represent each word as a vector of real numbers. The resulting word vectors have the property that words that appear in similar contexts are assigned similar vectors. This means that words that are semantically similar will have similar representations in the *GloVe* Space.


**Figure 4.1** Visual illustration about word embedding

There are several benefits in using *GloVe* word embeddings in neural network models:

1. Improve performance: Word embeddings can improve the performance of a neural network by providing a more efficient and effective way to encode the input data.

One-hot encoding, which is a common alternative, is inefficient because it requires a separate dimension for each word in the vocabulary, which can be very large. Word embeddings, on the other hand, can compress the information into a much lower-dimensional space, which can make the model more efficient and easier to train.

2. Handling out-of-vocabulary (OOV) words: *GloVe* word embeddings can handle OOV words (words that are not in the training dataset) more effectively than one-hot encoding. With one-hot encoding, OOV words are typically represented as a vector of all zeros, which does not provide any useful information to the model. With *GloVe* word embeddings, OOV words can be represented by their closest neighbors in the word vector space, which can provide some information to the model even if it has not seen the word before.
3. Transfer learning: *GloVe* word embeddings are pre-trained on a large dataset and can be used as a starting point for another task. Which can save time and resources, and it can also improve the performance of the model.
4. Interpretability: Because *GloVe* word embeddings are numeric vectors, it is possible to perform mathematical operations on them (e.g., addition, subtraction). This can make it easier to understand and interpret the relationships between words. For example, we can find the vector representation of the word "king" and subtract the vector representation of the word "man" to get the vector representation of the word "queen". This kind of interpretability can be useful for debugging and for understanding how the model is making decisions.

Although *GloVe* word embeddings are good, there are also some potential drawbacks when using *GloVe* word embeddings:

1. Lack of fine-tuning: Because *GloVe* word embeddings are pre-trained on a large dataset, they may not be perfectly suited to some specific tasks.
2. Large size: The *GloVe* word embedding model is quite large, with millions of parameters. This can be an issue if working with a small dataset or if limited in terms of computational resources.

Despite these drawbacks, *GloVe* word embeddings are still a powerful tool for neural network models, particularly when working with natural language data. They can improve the performance of the model, handle OOV words effectively, facilitate transfer learning, and provide interpretability.

From the formulation above, it needs to be discussed how neural network models work. As explained below:

Firstly, neural networks are made up of layers of neurons. These neurons are the core processing units of the network.

Neural networks are made up of layers of neurons, these neurons are the core processing units of the network. We have the input layer which receives the input and in this case its three different kinds of shape. While the output layer predicts our final output, and in between them exist the hidden layers which perform most of the computations required by our network.

Each pixel is fed as input to each neuron of the first layer, neurons of one layer are connected to neurons of the next layer through channels. Each of these channels is assigned a numerical value known as weight, the inputs are multiplied to the corresponding weights and their sum is sent as input to the neurons in the hidden layer.

Each of the neurons from the input layer is associated with a numerical value called the "bias" which is then added to the input sum, this value is then passed through a threshold function called the activation function.

$$(x_1 * 0{,}8 + x_3 * 0{,}2) + B_1$$

The result of the activation function will determine if the particular neurons will get activated or not.

An activated neuron transmits data to the neurons of the next layer over the channels, in this manner the data is propagated through the network and it is called forward propagation. In the output layer, the neuron with the highest value fires and determines the output, the value is basically probability.

The neuron is associated with the square who has the highest probability, hence that is the output predicted by the neural network. But, just by looking at the result, neural network has made a wrong prediction. Along with the input, our network also has the output fed to it. The predicted output is compared against the actual output to realize the error in prediction.

Backpropagation is an algorithm that back propagates the errors from output nodes to the input nodes. Backpropagation works by computing the gradient of the loss function with respect to

each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule, this process continues until our weights are assigned such that the neural network models can predict the shapes correctly.

### *4.1.4. Forward Propagation*

As the name suggests, the input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.

In order to generate some output, the input data should be fed in the forward direction only. The data should not flow in reverse direction during output generation or otherwise it would form an infinite cycle and the output could never be generated.

At each neuron in a hidden or output layer, the processing happens in two steps:

1. Preactivation: it is a weighted sum of inputs i.e., the linear transformation of weights to inputs available. Based on this aggregated sum and activation function the neuron makes a decision whether to pass this information further or not.
2. Activation: the calculated weighted sum of inputs is passed to the activation function. An activation function is a mathematical function which adds non linearity to the network. There are four commonly used and popular activation functions such as sigmoid, hyperbolic tangent(tanh), ReLu and Softmax.

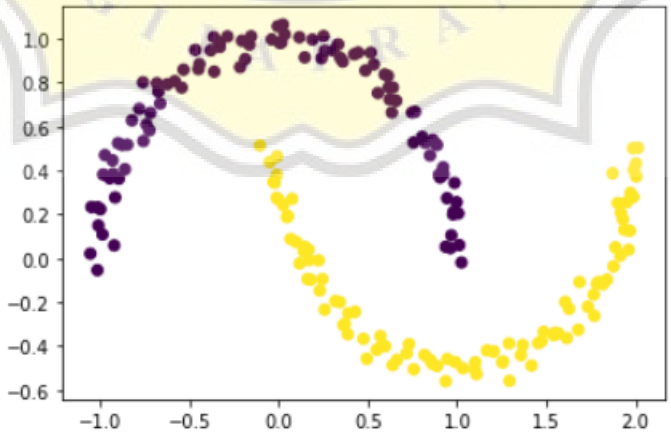

**Figure 4.1** 200 samples are used to generate the data and it has two classes shown in red and green color.

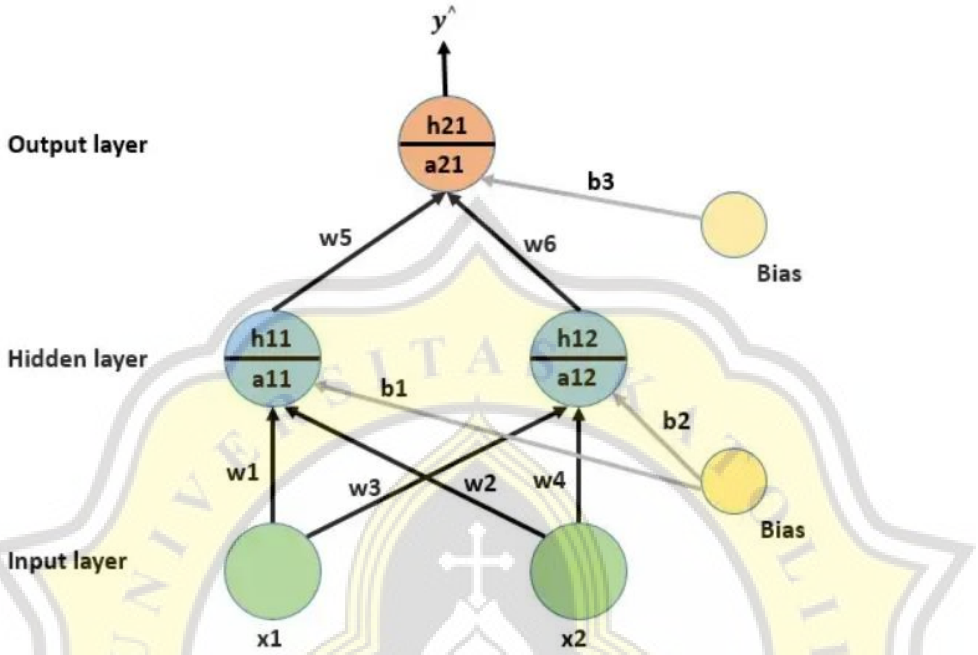


**Figure 4.2** shows the neural network structure for the binary classification problem.

One hidden layer with two neurons, an output layer with a single neuron and sigmoid activation function is used. During forward propagation at each node of hidden and output layer preactivation and activation takes place. For example, at the first node of the hidden layer, a1(preactivation) is calculated first and then h1(activation) is calculated.

And a1 is a weighted sum of inputs. In this example, the weights are randomly generated. Then a11 = w1*x1 + w2*x2 + b1 = 1.76*0.88 + 0.40*(-0.49) + 0 = 1.37 approx. and h11 is the value of activation function applied on a11.

$$h1 = \frac{1}{1 + e^{-a1}} = 0.8\ approx.$$

Similarly

A2 = w3*x1 + w4*x2 + b2 = 0.97*0.88 + 2.24*(-0.49) + 0 = -2.29 approx and

$$h2 = \frac{1}{1 + e^{-a2}} = 0.44\ approx.$$

For any layer after the first hidden layer, the input is output from the previous layer.

A3 = w5*h1 + w6*h2 + b3 = 1.86*0.8 + (-0.97)*0.44 + 0 = 1.1 approx and

$$h3 = \frac{1}{1 + e^{-a3}} = 0.74\ approx.$$

So, there are 74% chances the first observation will belong to class 1. By doing this for all the other observations, predicted output can be calculated.

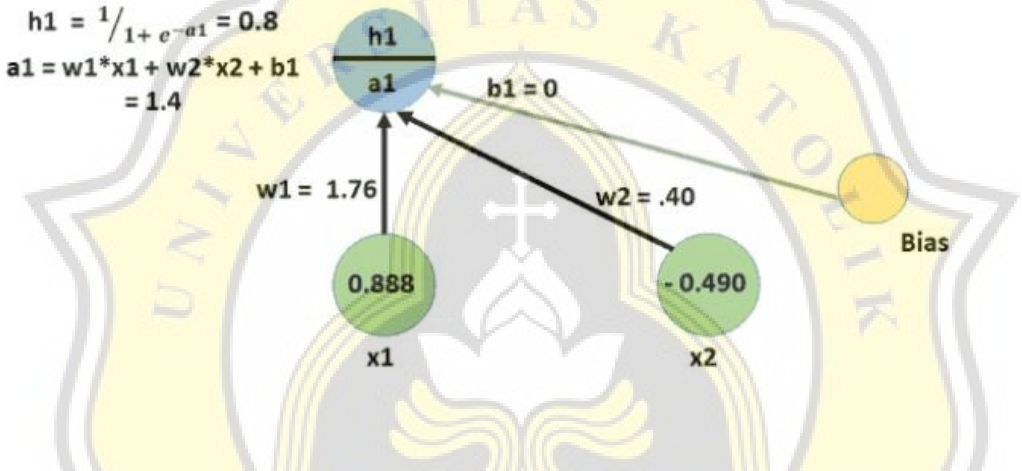


**Figure 4.3** shows the calculation of the first hidden layer

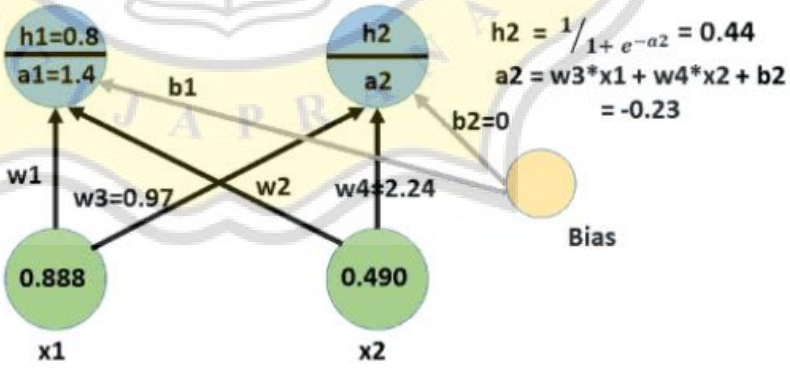


**Figure 4.4** shows the calculation of the second hidden layer

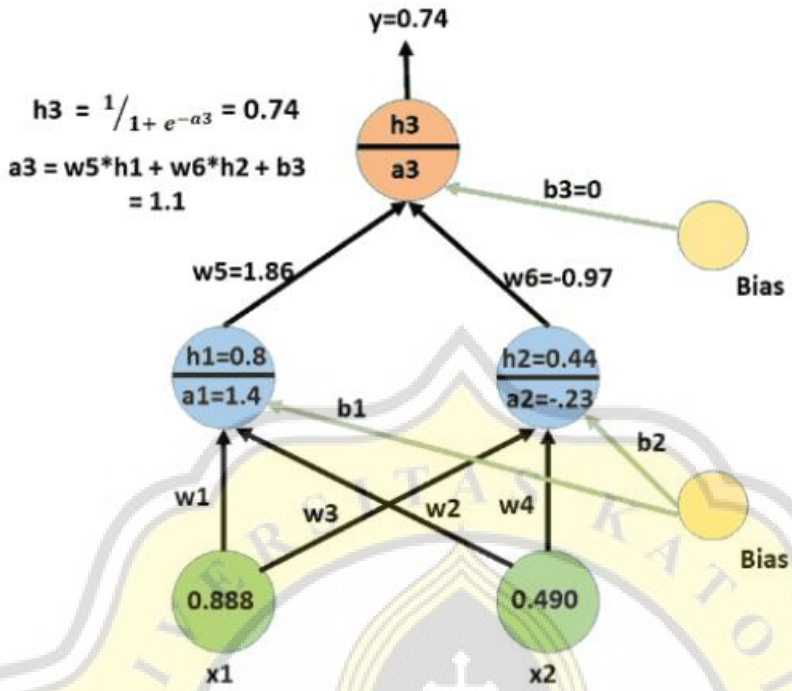


**Figure 4.5** show the calculation of the output layer

### *4.1.5. Backpropagation*

Is a mechanism used to update the weights using gradient descent. It calculates the gradient of the error function with respect to the neural network's weights. The calculation proceeds backwards through the network.

$$* W_x = W_x - a\,(\frac{\partial Error}{\partial W_x})$$

Where $*W_x$ is new weight, $W_x$ is the old weight, $a$ is the learning rate, and finally $\frac{\partial Error}{\partial W_x}$ is a derivative error with respect to the weight.

## 4.2. Convolutional Neural Networks

Convolutional neural networks (CNN) are a type of artificial neural network that are particularly well-suited to processing data that has a grid-like structure, such as an image. However, CNN can also be applied to 1D data, such as text. In this case, the input text is represented as a sequence of words, with each word being a "pixel" in the grid. The 1D CNN "slides" a kernel (a small matrix of weights) across the input text, performing a dot product between the weights in the kernel and the words at each position.

Before the 1D CNN can be applied to the text, it must be converted into a numerical representation that the network can process. This is typically done by tokenizing the text (splitting it into individual words), and then creating a vocabulary of all the unique words in the text. Each word in the vocabulary is then assigned a unique integer index. The input text is then converted into a sequence of these integer indices.

After the convolutional layers have extracted the features from the input text, an activation function is applied to the output of each filter. The purpose of the activation function is to introduce non-linearity into the network, allowing it to learn more complex relationships between the input and output data. Commonly used activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh.

After the activation function has been applied, the output of the convolutional layer is often passed through a pooling layer, which reduces the dimensionality of the output. This is done by taking the maximum, minimum, or average value of a set of adjacent outputs. Pooling helps to reduce the number of parameters in the model, making it more efficient and less prone to overfitting.

After the pooling layers, the output of the 1D CNN is typically passed through one or more fully connected (also known as dense) layers. These layers are responsible for combining the features extracted by the convolutional layers and learning more complex relationships between them. The fully connected layers use weights to combine the inputs from the previous layers and produce an output.

The final layer in the 1D CNN is the output layer, which produces the predicted output for the input text. The output layer can have a variety of different forms, depending on the task being

performed. For example, in a classification task, the output layer might consist of a single neuron with a sigmoid activation function, which outputs a probability value between 0 and 1 indicating the likelihood that the input text belongs to a particular class. In a regression task, the output layer might consist of a single neuron with a linear activation function, which outputs a continuous value.

Here's a more detailed explanation of how 1D CNN work:

1. Preprocessing: Before the 1D CNN can be applied, the input text must be converted into a numerical representation that the network can process. This is typically done by tokenizing the text (splitting it into individual words), and then creating a vocabulary of all the unique words in the text. Each word in the vocabulary is then assigned a unique integer index. The input text is then converted into a sequence of these integer indices. In this research word embedding were added after the preprocessing process/
2. Convolutional layers: The 1D CNN consists of one or more convolutional layers, which are responsible for extracting features from the input text. Each convolutional layer applies a set of filters to the input text, with each filter being responsible for extracting a specific type of feature. For example, one filter might be responsible for detecting the presence of certain words or word combinations, while another filter might be responsible for detecting the presence of certain patterns in the text.
3. Activation functions: After the convolutional layers have extracted the features from the input text, an activation function is applied to the output of each filter. The purpose of the activation function is to introduce non-linearity into the network, allowing it to learn more complex relationships between the input and output data.
4. Pooling layers: After the activation function has been applied, the output of the convolutional layers is often passed through a pooling layer, which reduces the dimensionality of the output. This is done by taking the maximum, minimum, or average value of a set of adjacent outputs. Pooling helps to reduce the number of parameters in the model, making it more efficient and less prone to overfitting.

5. Fully Connected layers: After the pooling layers, the output of the 1D CNN is typically passed through one or more fully connected layers. These layers are responsible for combining the features extracted by the convolutional layers and learning more complex relationships between them.
6. Output layer: The final layer in the 1D CNN is the output layer, which produces the predicted output for the input text. The output layer can have a variety of different forms, depending on the task being performed. In this classification task, the output layer might consist of a single neuron with a sigmoid activation function.

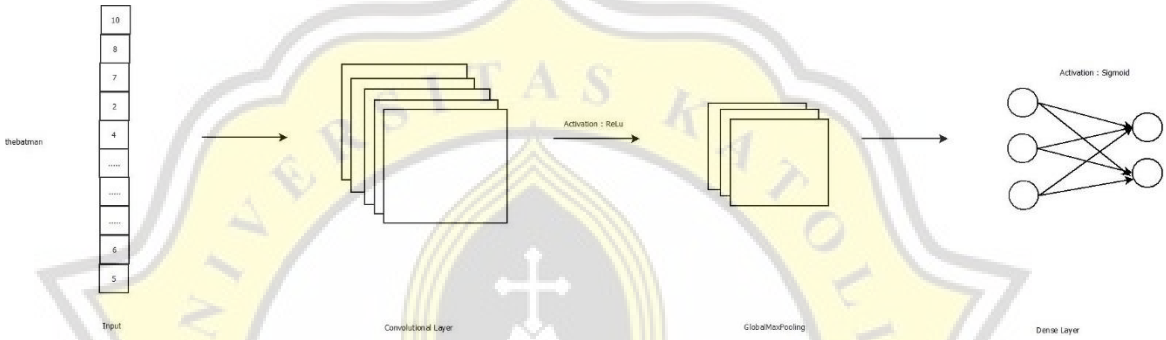


**Figure 4.1** Convolutional 1D architecture

The neural network structure for this research presented the following configuration:

- Input layer: 100 inputs to which correspond the Glove 100 dimension word embeddings
- Convolutional layer
  - ♦ Composed by 128 filters; 1-D kernel with stride = 1; ReLU Activation function
- GlobalMaxPooling layer
  - ♦ GlobalMaxPooling1D is a layer that performs global max pooling on the 1D input. This means that it takes in a 1D input, such as a sequence of words, and applies max pooling across all of the elements in the sequence to produce a single output value. This layer is useful for tasks such as text classification, where the goal is to identify the most important feature in a sequence
- Dense Layer
  - ♦ A dense layer in a convolutional 1D is a fully connected layer. It is called "dense" because all the neurons in the layer are connected to the neurons in the

previous layer. The dense layer takes the output from the previous layers and applies a set of weights to it in order to produce the final output.

## 4.3. LSTM

Long Short-Term Memory (LSTM) is a type of recurrent neural network that is widely used for processing sequential data such as time series, natural language, and speech. LSTM are particularly useful for tasks that require the model to remember and use information from long sequences, because they are able to retain information for longer periods of time than other types of RNN.

One of the key features of LSTM is their ability to control the flow of information through the network using gates. LSTM has three types of gates: input, output, and forget. The input gate determines which information from the current input should be passed to the next layer of the network, the output gate determines which information from the current state should be output, and the forget gate determines which information from the previous state should be discarded.

The use of gates in LSTM allows them to selectively remember and forget information, which is what makes them so effective at processing long sequences. This is in contrast to traditional RNN, which have a more difficult time retaining information over long periods of time because they do not have the ability to selectively forget information.

One of the main benefits of LSTM is their ability to handle long-term dependencies in data. In natural language, for example, a word may have a different meaning depending on the words that come before and after it. LSTM are able to capture these dependencies by retaining information from previous states, which allows them to make more informed predictions/

There are a few potential drawbacks when using LSTM:

1. Computational complexity: LSTM are more complex than traditional RNN, which can make them more difficult to train and deploy in certain situations.
2. Difficulty interpreting results: LSTM are black box models, which means that it can be difficult to understand exactly how they are making decisions. This can make it more difficult to debug and interpret the results.
3. Data preparation: LSTM requires input data to be in a specific format, which can require additional preprocessing.

Despite these drawbacks, LSTM is still a powerful tool for processing sequential data and has been very successful in a variety of tasks. They are particularly well-suited for tasks that require the model to remember and use information from long sequences, and they have proven to be very effective in natural language processing.

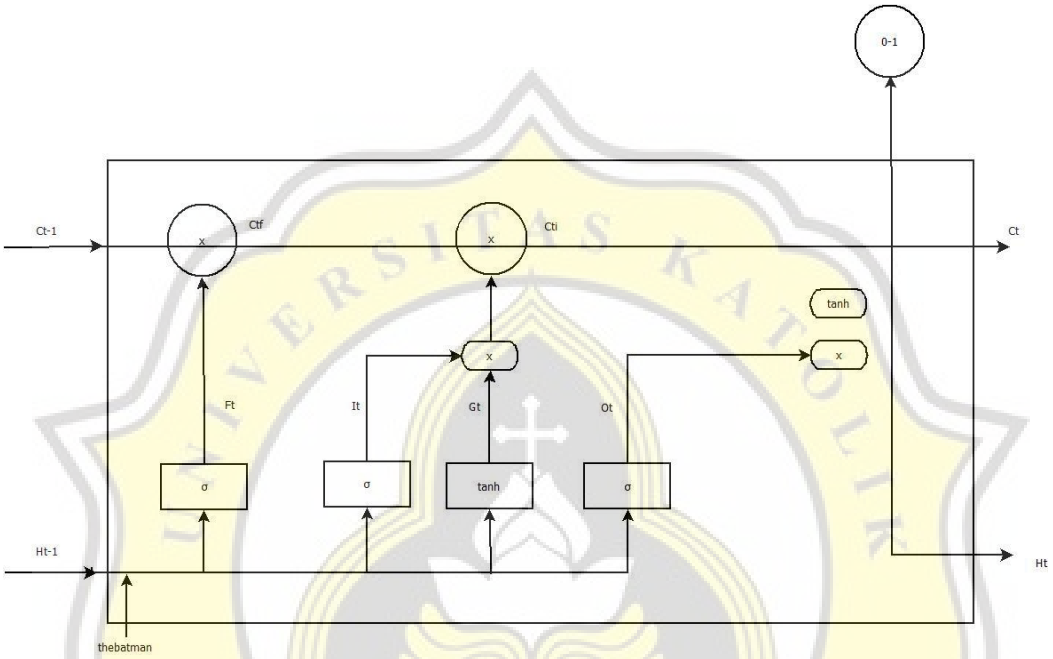


**Figure 4.1** LSTM Architecture

The first process that happening in LSTM cell state are calculating whether the past values is being feed through or not in the forget gate using:

$$F_t = \sigma[(Wf_h + h_{t-1}) + (Wf_x + x_t) + b_f$$

$F_t$ = Forget gate

$Wf_h$ = Weight of forget gate

$h_{t-1}$ = output of the previous state

$Wf_x$ = weight of the input

$X_t$ = input value (thebatman)

$b_f$ = bias of forget gate

After finding the output of the forget gate and if the value is 1 then it went into $C_{tf}$ but if the value is 0 then the value is removed and hence why it is called the forget gate.

$$C_{tf} = C_{t-1} * f_t$$

$C_{tf}$ = Cell state of forget gate

$C_{t-1}$ = previous cell state value

$F_t$ = Forget gate

The next state is to decide what new information that going to be stored in the cell state. This step has two parts. First, a sigmoid layer called the input gate layer decides which values we'll update.

$$i_t = \sigma[(Wi_h + h_{t-1}) + (Wi_x + x_t) + b_i$$

$i_t$ = Input gate sigmoid

$Wi_h$ = Weight of input gate

$h_{t-1}$ = output of the previous state

$Wi_x$ = weight of the input

$X_t$ = input value (thebatman)

$b_i$ = bias of input gate

Next, a tanh layer creates a vector of new candidate values:

$$g_t = tanh[(Wg_h + h_{t-1}) + (Wg_x + x_t) + b_g$$

$g_t$ = Input gate tanh

$Wg_h$ = Weight of input gate

$h_{t-1}$ = output of the previous state

$Wg_x$ = weight of the input

$X_t$ = input value (thebatman)

$b_g$ = bias of tanh input gate

Then, the value of $i_t$ and $g_t$ concatenated using:

$$C_{ti} = i_t * g_t$$

After completing these two steps now the value for cell state can be determined using:

$$C_t = C_{ti} + C_{tf}$$

$C_t$ = Cell state

$C_{ti}$ = Cell state of input gate

$C_{tf}$ = Cell state of forget gate

Finally, the output gate which can be calculated based on the cell state, but will be a filtered version. First, run a sigmoid layer to decide what parts of the cell state are going into the output. Then, put the cell state through tanh and multiply it by the output of the sigmoid gate.

$$o_t = \sigma[(Wo_h + h_{t-1}) + (Wo_x + x_t) + b_o$$

$o_t$ = Output gate

$Wg_h$ = Weight of output gate

$h_{t-1}$ = output of the previous state

$Wg_x$ = weight of the output

$X_t$ = input value (thebatman)

$b_g$ = bias of output gate

$$h_t = \tanh(C_t * o_t)$$

$h_t$ = output

$C_t$ = cell state

$O_t$ = output gate

UNIVERSITAS KATOLIK
SOEGIJAPRANATA

## 4.4. Design

The usage of flowchart aims to know the process of a program that makes it easier to understand the program to be built. Flowchart can be described as:

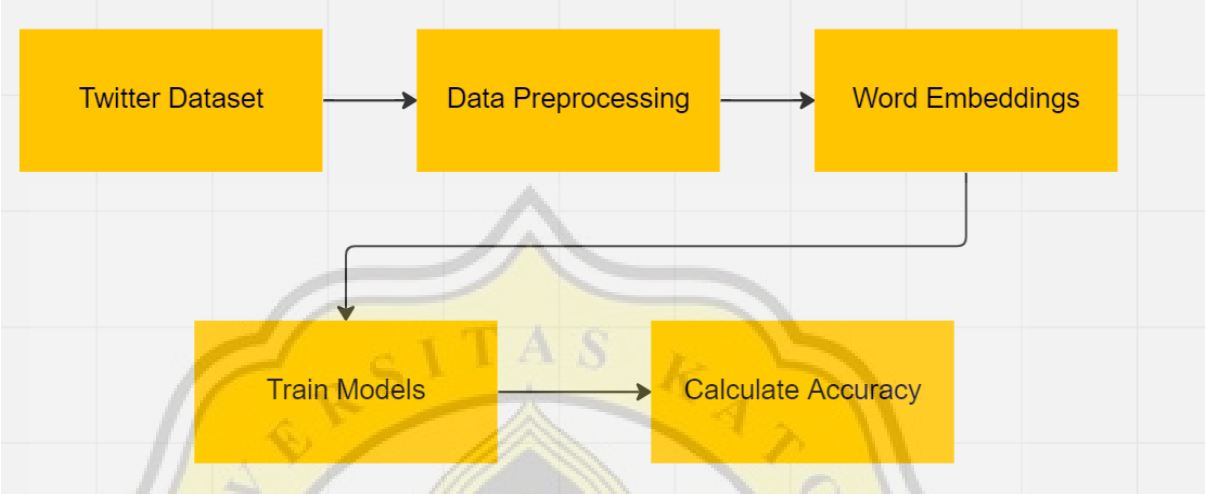


**Figure 4.1** Flowchart of neural network models

Figure 4.11 is used as an overview of the system itself. Starting from preparing the dataset, then the dataset that has been prepared will go into the pre-processing stage to remove any unnecessary character that will affect the accuracy. After that, converting the textual dataset into a numeric form using a feature extraction technique called word embedding and then the dataset will be trained into the neural network models.

In general, neural network models can only recognize numeric form and then the data will be trained with the help of back and forward propagation. The stage in the process is:

1. Prepare Dataset: dataset contains tweets from Twitter.
2. Pre-Processing: prepare and process the initial data so that the data used is cleaned and ready to use.
3. Word Embedding: in this stage, the processed data is still in a textual form. Therefore, *GloVe* or Global Vector word embedding needs to be added to transform the textual form into numerical form.
4. SNN, CNN, and LSTM: after the dataset has been transformed into numerical form, the data is ready and will be trained using CNN and LSTM models.

In this process, the algorithms have a role in training the dataset and determine the accuracy of the models itself.

5. Calculate Accuracy: calculating the accuracy of each model and the results are compared between SNN, CNN and LSTM.