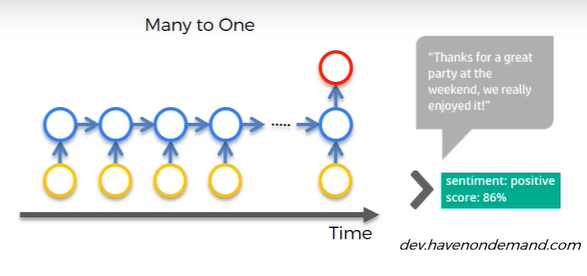
**RESEARCH**

**BREATHING WAVE PATTERN**

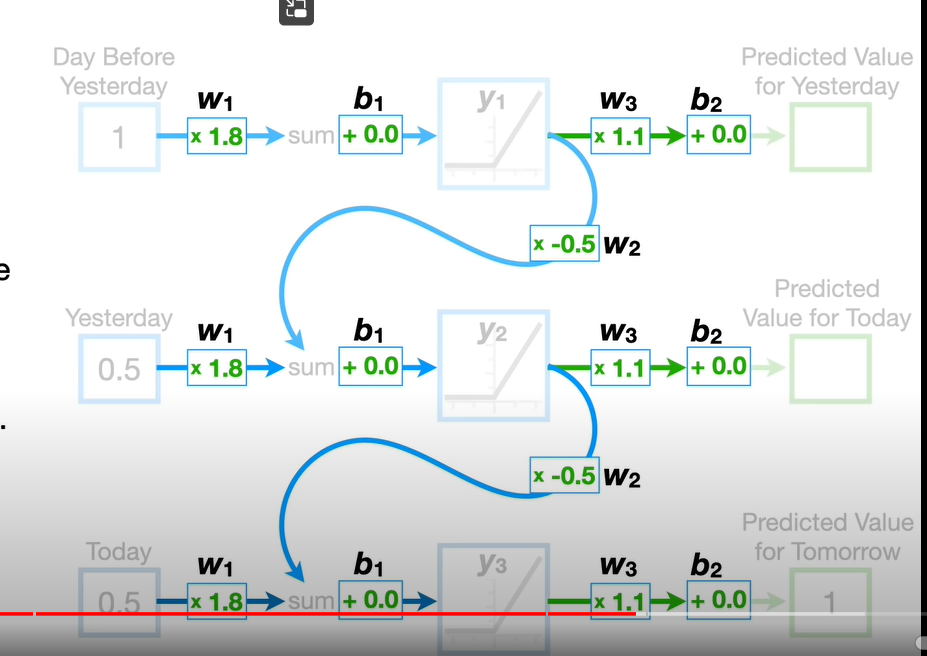
Deep Learning – RNN using LSTM

1. DATASET (~ 1 Paragraf)
   1. Jumlah datanya
   2. Berapa persen yang dipakai buat training dan testing
   3. Proses Metode Standardsasi data
   4. Label Encoder & Hot Encoder
   5. Reshape data
2. RNN

Using the **“Many to One”** RNN Schema to predict only categorical values.



RNN - https://youtu.be/AsNTP8Kwu80



This is the traditional RNN Process, and this is not used often because of the **VANISHING & EXPLODING GRADIENT PROBLEM,** which say if the NUMBER of UNROLL is so many it will creating extreme multiplication for the gradient value, and it will resulting vanished / exploded the model.

**SOLUTION**

1. **Exploding Gradient**
   1. **Truncated Backpropagation**

Stopping updating the weight across all NN, only certain amount NN will be updated this will creating not optimal model.

* 1. **Penalties**

Gradient being penalized / artificial reduced

* 1. **Gradient Clipping**

Limit the gradient

1. **Vanishing Gradient**
   1. **Weight Initialization**

How smart you initialize the value of weight

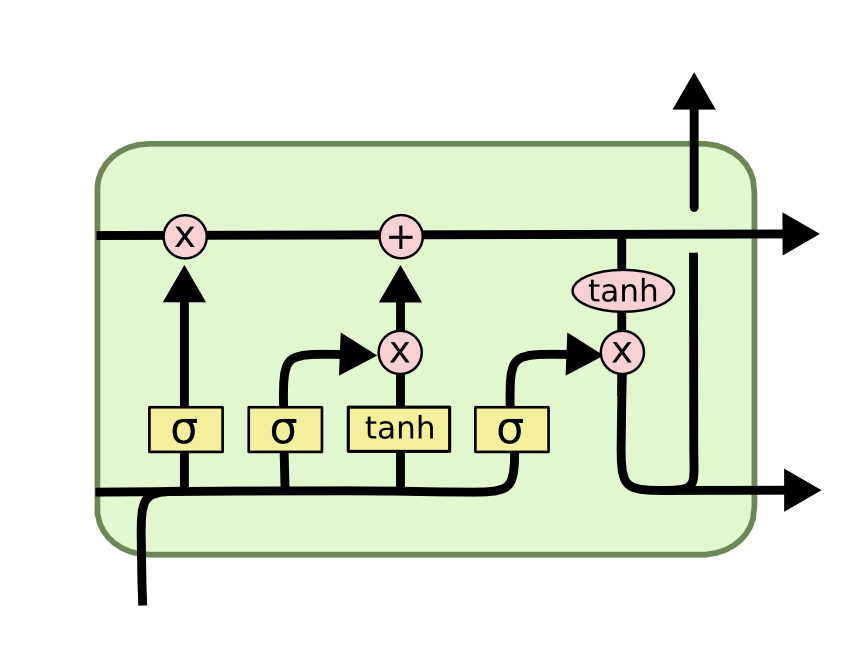
* 1. **Echo State Networks**
  2. **Long Short-term Memory Network (LSTMs)**

**ht**

**LSTM**

**Forgot Valve**

**Adding Valve**



**Output Valve**



**Xt**

**Ct**

**Ct - 1**

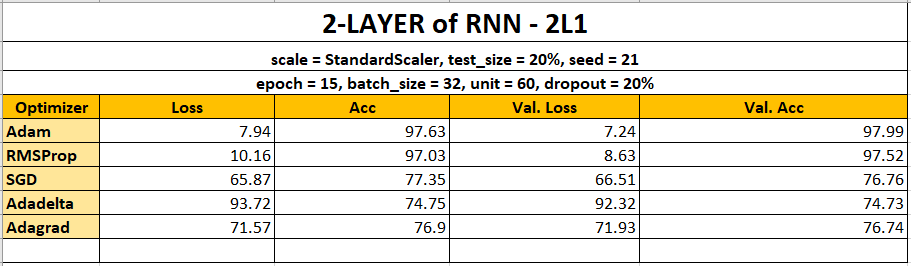
**ht - 1**

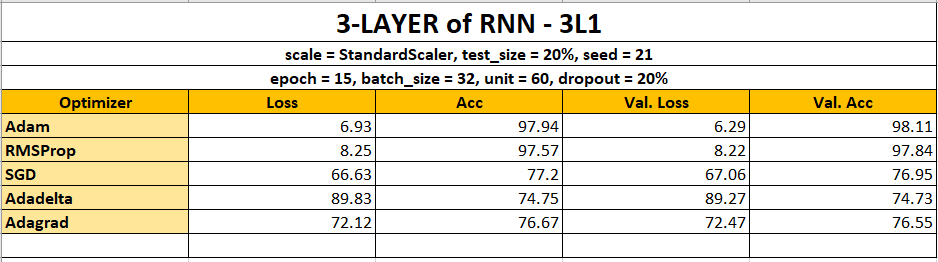
**ht**

**MODEL (WITHOUT MFCC)**

**[Cross Validation] HOLD-OUT**

1. Separate into equal, ***80% training and 20% testing***
2. Accuracy of different Optimizer

****

****



**BEST ACCURACY : 98.11%**

**OPTIMIZER : Adam**

1. **[OPTIONAL]** Augmented Result
   * **UP (val = 0.01)**
     + **Achieve same accuracy with lower epochs**
     + **7 epochs**

loss = 8.21 accuracy = 97.64

val\_loss = 6.91 val\_accuracy = 98.17

* + - **15 epochs**

loss = 4.50 accuracy = 98.74

val\_loss = 3.01 val\_accuracy = 99.05

* + **UP & DOWN (val = 0.01)**
    - **5 epochs**

loss = 7.69 accuracy = 97.85

val\_loss = 5.37 val\_accuracy = 98.40

* + - **9 epochs**

loss = 4.21 accuracy = 98.85

val\_loss = 2.82 val\_accuracy = 99.20

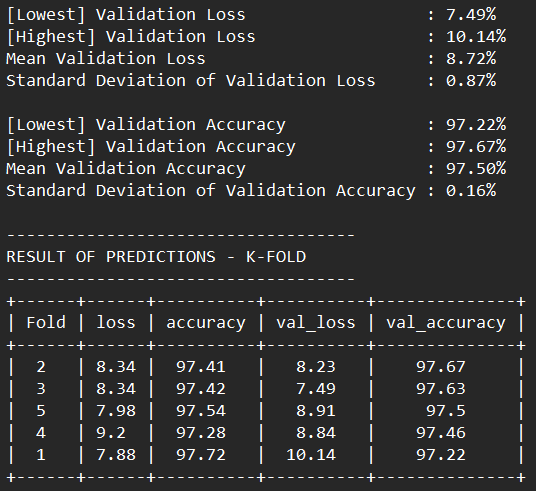
* + - **15 epochs**

loss = 2.17 accuracy = 99.34

val\_loss = 1.22 val\_accuracy = 99.58

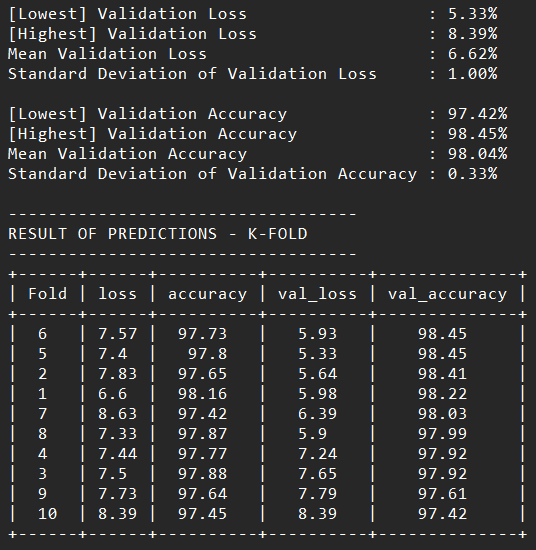
**[Cross Validation] Stratified K-FOLD**

1. **5-FOLD**

****

**NB.** Sorted by ‘val\_accuracy’

1. **10-FOLD**

****

**NB.** Sorted by ‘val\_accuracy’

**BEST FOLD : 10-FOLD**

**CONCLUSION :** Lower ‘val\_loss’ and higher ‘val\_accuracy’

**There is no sign of Overfitting**. Training loss is greater than validation loss

**There is no sign of Underfitting**. Training loss and Validation loss are not about equal, they have bigger gap of value for each.

Souce : [GitHub - karpathy/char-rnn: Multi-layer Recurrent Neural Networks (LSTM, GRU, RNN) for character-level language models in Torch](https://github.com/karpathy/char-rnn#tips-and-tricks)

List ALL the information Parameters – ada di TEAMS

|  |  |  |  |
| --- | --- | --- | --- |
| **MODEL** | **HYPERPARAMETER** | **VALUES** | **FINAL VALUE** |
| **RNN (LSTM)** | epoch  batch\_size  dropout\_rate  init\_mode  init\_recurrent  units | 15, 20  128, 256  0.2, 0.3  ‘glorot\_uniform’, ‘he\_uniform’  ‘glorot\_uniform’, ‘orthogonal’  17, 30, 60 | 20  128  0.3  he\_uniform  orthogonal  17 |

The requirements to use the cuDNN implementation are:

1. activation == tanh
2. recurrent\_activation == sigmoid
3. recurrent\_dropout == 0
4. unroll is False
5. use\_bias is True
6. Inputs, if use masking, are strictly right-padded.
7. Eager execution is enabled in the outermost context.

**[OPTIONAL] INFORMATION**

**NVIDIA RTX 3060 vs Intel i9 11900F (in minutes)**

|  |  |  |  |
| --- | --- | --- | --- |
| **MODEL** | **TASK** | **NVIDIA RTX 3060** | **Intel Core i9 11900F** |
| Whisper Open AI | Transcribe 10 Minute English Video | 1.32 | 9.10 |