**STUDI LITERATUR NEURAL NETWORK**

**Proses Pengimplementasian NN dalam Penelitian**

A picture containing timeline

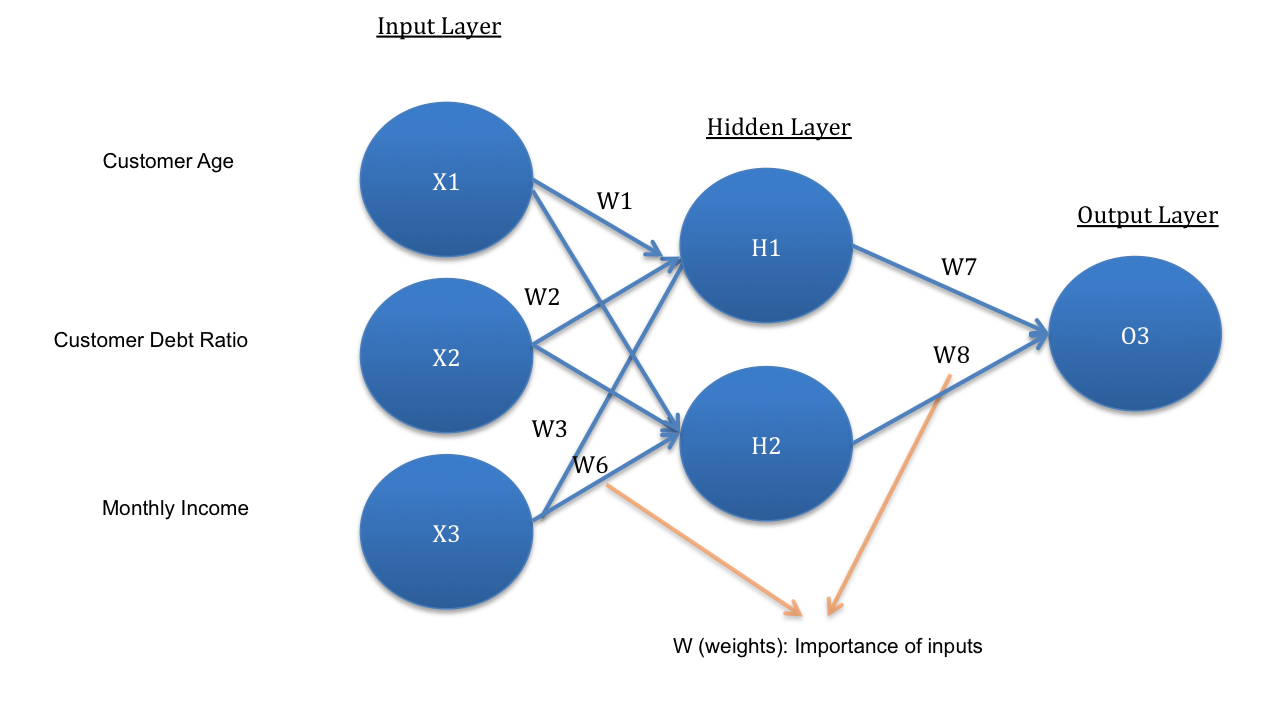
Description automatically generated

(Lucke D., 2020)

**Konsep Dasar Artificial Neural Network (ANN)**

Merangkai serangkaian perhitungan data input dengan hasil output berupa prediksi atau nilai pendekatan terhadap tolak ukur atau tujuan yang ingin dicapai. Misalnya sistem deteksi huruf “a” menggunakan *font* “a” jenis *times new roman* sebagai tolak ukur dalam mendeteksi huruf apapun yang menyerupai bentuk tolok ukur tersebut (contoh: “a”) sebagai huruf a.

**Struktur Dasar ANN**



* Input adalah data yang ingin diolah, yang juga menjadi dasar prediksi sistem
* Output adalah hasil prediksi sistem
* Neuron adalah satuan unit data yang dapat berupa input, hasil perhitungan sementara (pada *hidden layer)*, atau output
* *Layer* adalah kumpulan beberapa neuron dalam satu tahapan proses yang sama
* *Hidden layer* adalah kumpulan beberapa neuron yang bukan input atau output
* Weight adalah konstanta perhitungan (perkalian) yang menentukan hasil prediksi, yang nilainya ditentukan melalui proses *training*

**Jenis-Jenis ANN berdasarkan Arsitektur**

*Feedforward*

Dalam arsitektur *feedforward*, koneksi neuron hanya ada antara neuron pada *input layer* dan *hidden layer* dan antara *hidden layer* dengan output layer. Neuron-neuron dalam satu *layer* tidak saling terkoneksi (Chen et al., 2020, p.4). Pada arsitektur jenis ini, sinyal hanya dapat lewat secara searah dari input ke output tanpa adanya *feedback (loop).*

*Recurrent*

Berbeda dengan *feedforward ANN,* arsitektur *recurrent* menambahkan koneksi antar neuron satu *layer* dan menambahkan *feedback.* Beberapa contoh *neural network* dengan arsitektur *recurrent*, antara lain, RNN, LTSM, NARX, dll. (Chen et al., 2020, p.6).

Diagram

Description automatically generated

(Pekel, et al., 2017)

*Hybrid*

Arsitektur jenis *hybrid* merupakan gabungan dari beberapa arsitektur yang sudah ada (Chen et al., 2020, p.6).

*Jenis Baru*

Contoh-contoh arsitektur jenis baru antara lain, *convolution neural network (CNN), deep belief network (DBN),* dan lain sebagainya (Chen et al., 2020, p.7).

**Perbedaan Arsitektur *Feedforward* NN dengan *Recurrent* NN**

Karena adanya *loop* pada setiap neuron, perhitungan pada RNN dapat dilakukan dengan memperhitungkan dua input, *present memory* dan *recent past memory.* Secara matematis, pada *time stamp* t+1, RNN memperhitungkan input pada *time stamp* t+1 dan t. Hal ini membuat RNN dapat digunakan untuk memodelkan data temporal atau sekuensial seperti data *time series* atau video.

*Feedforward NN* tidak memiliki memori sehingga tidak dapat membedakan data pada *time series* satu dengan yang lain. Hal ini membuat *feedforward NN* sangat sulit untuk memprediksi data berurutan. Sebagai ilustrasi, jika input "neuron" diproses dari huruf ke huruf, saat sistem sudah sampai pada huruf “r”, sistem tersebut sudah melupakan huruf “n”, “e”, dan “u”.

**Jenis-Jenis ANN berdasarkan proses prediksi output *(basis function)***

1. *Backpropagation Neural Networks (BPNNs)*
2. *Radial Basis Function Networks (RFBNs)*
3. *Probabilistic Neural Networks (PNNs)*
4. *Clustered Probabilistic Neural Networks (CPNNs)*

Graphical user interface, text, email

Description automatically generated

(Lee et all., 2018, p.4)

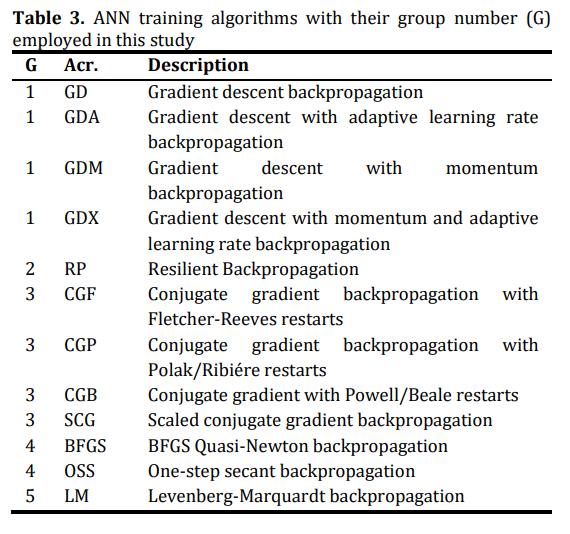
*Backpropagation Neural Network (BPNNs)*

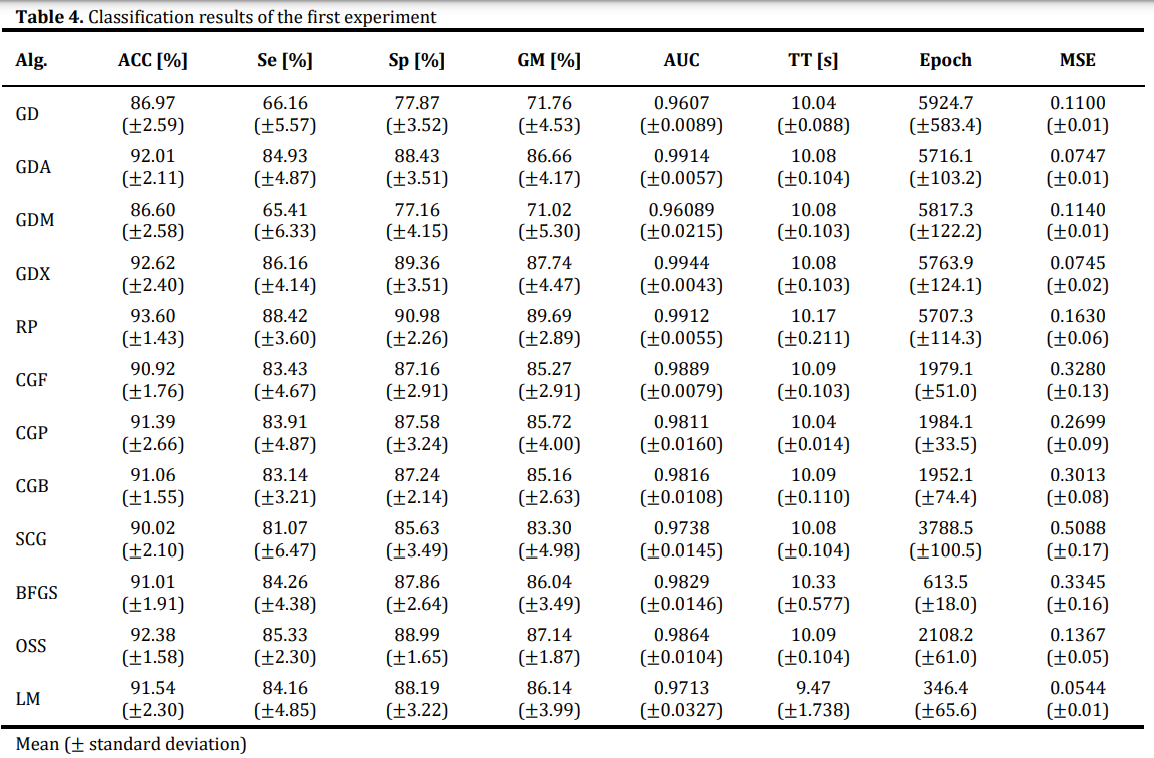
BPNN menggunakan konsep fungsi eror untuk menentukan nilai dari *weights* menggunakan *gradient descent method (GDM)1* dalam proses *training* *machine learning*. BPNN merupakan jenis algoritma ANN yang paling sederhana dan sering diimplementasikan.

**Algoritma-algoritma untuk *training neural network***

CÖMERT Z. (2017) membandingkan performa beberapa *training algorithm* dalam proses klasifikasi sinyal FHR yang terdapat dalam *Cardiotocography (CTG)* dan menganalisis hasil penelitian berdasarkan parameter-parameter seperti, *accuracy (ACC), sensitivity (Se), specificity (Sp),* dan *geometric mean (GM).*

Dapat disimpulkan bahwa performa klasifikasi terbaik didapatkan dengan algoritma Levenberg-Marquardt backpropagation (LM) dan Resilient Backpropagation (RP) dengan nilai GM darif RP and LM secara berturut-turut adalah 89.69% dan 86.14%.

****



**Konsep Dasar Backpropagation dan *Gradient Descent***

Penjelasan intuitif dibalik proses kalkulasi sistem neural network sederhana: <https://www.youtube.com/watch?v=GltT9b31fRY&ab_channel=Simplilearn>

*Ringkasan*

Proses *backpropagation* adalah cara sistem untuk menentukan nilai *weight* yang cocok sesuai dengan tujuan sistem yang diinginkan. Hal ini dilakukan dengan membandingkan nilai output yang diinginkan dengan nilai input sehingga ditemukan nilai *error*. Nilai *error* ini menjadi dasar dilakukannya *adjustment* pada nilai *weight* yang dilakukan beberapa kali hingga nilai *error* mendekati nol.

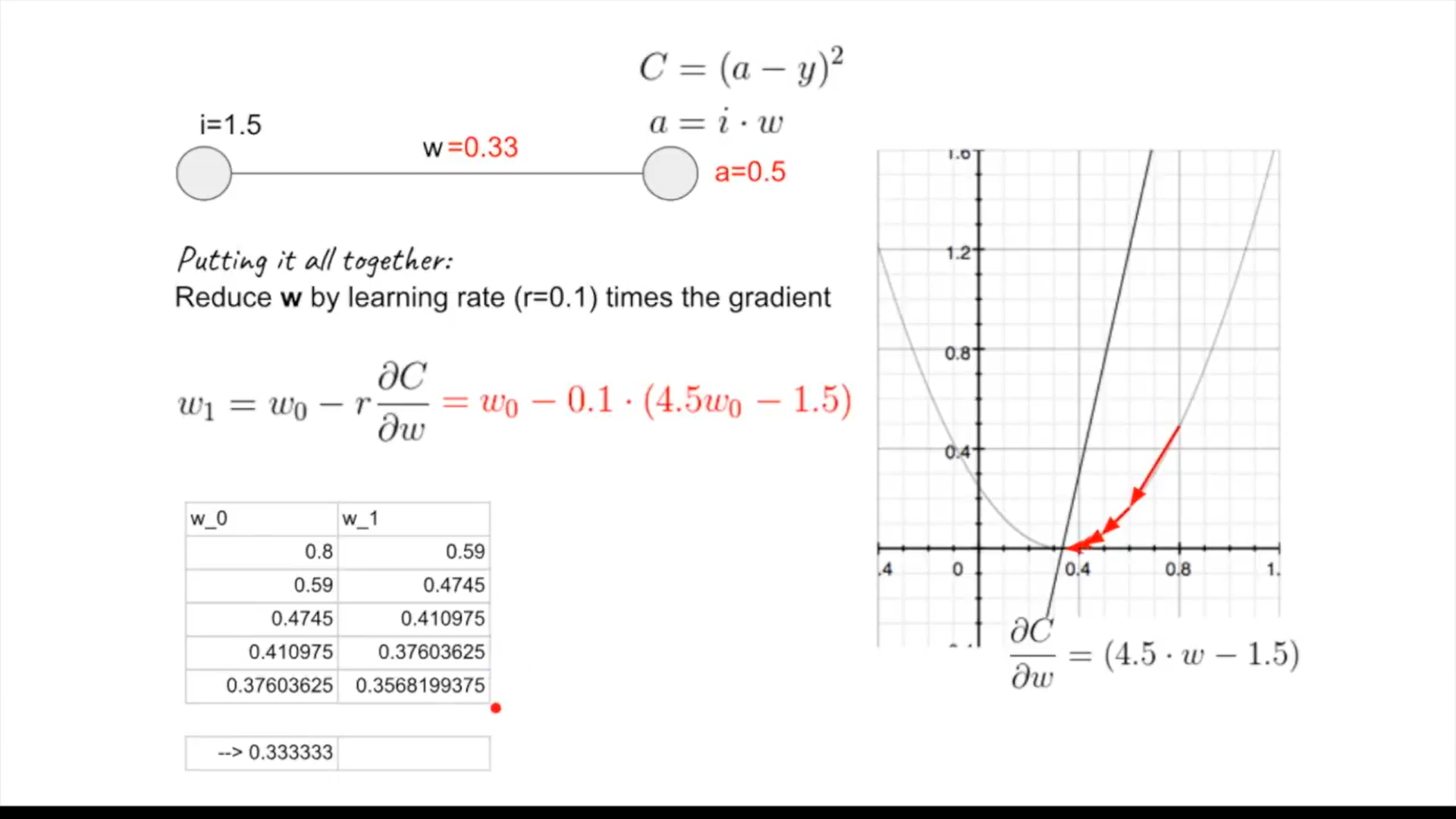
Chart

Description automatically generated

Penjelasan konsep *gradient descent* dengan pendekatan matematis: <https://www.youtube.com/watch?v=8d6jf7s6_Qs&ab_channel=MikaelLaine>

*Ringkasan*

Secara matematis, proses *backpropagation* adalah proses iterasi pengurangan nilai *weight lama* dengan nilai *learning rate* dikalikan dengan nilai *gradient* dari fungsi *error*. Nilai *learning rate* melambangkan besar *adjustment* yang dilakukan setiap kali iterasi, yang nilainya ditetapkan di awal, sedangkan nilai *gradient* melambangkan arah sekaligus seberapa besar perubahan yang perlu dilakukan untuk mencapai nilai *error* yang mendekati nol. Banyaknya layer pada sistem sebanding dengan banyaknya rantai pada penurunan (derivatif) fungsi *error*.



**Masalah dalam RNN standar dan LTSM**

Dua masalah dalam *standard RNN* adalah masalah *exploding gradient* dan *vanishing* *gradient*. *Exploding gradients* berarti nilai gradien pada suatu fungsi eror mendeka

***Vanishing Gradient***

*Vanishing Gradient* terjadi ketika gradien menjadi semakin kecil seiring bergeraknya sistem ke belakang (ke *time step* sebelumnya) selama *backpropagation*.

Ketika pembaruan *weight* sangat kecil, waktu pelatihan akan memakan waktu terlalu lama, dan dalam kasus terburuk, ini dapat sepenuhnya menghentikan pelatihan.

Masalah *vanishing gradient* terjadi saat fungsi aktivasi sigmoid dan tanh antara 0 hingga 0,25 dan 0 hingga 1. Oleh karena itu, nilai *weight* yang diperbarui kecil sehingga nilai *weight* baru sangat mirip dengan nilai bobot lama. Perbedaan yang semakin kecil membuat *gradient* berikutnya menjadi semakin kecil hingga menjadi nol.

Masalah ini dapat dihindari dengan menggunakan fungsi aktivasi ReLU karena gradiennya adalah 0 untuk input negatif dan nol serta 1 untuk input positif.

***Exploding Gradient***

*Exploding gradient* terjadi ketika *gradient* menjadi semakin besar saat dilakukan proses mundur selama *backpropagation*. Situasi ini adalah kebalikan dari masalah *vanishing gradient*.

Masalah ini terjadi karena *weight* bukan karena fungsi aktivasi. Nilai *weight* yang tinggi membuat nilai *gradient* juga tinggi sehingga *weight* baru memiliki nilai *gradient* yang sangat besar. Hal ini mengakibatkan osiliasi di sekitar minimum dan tidak pernah mencapai titik minimum global.

<https://www.numpyninja.com/post/vanishing-and-exploding-gradients-in-neural-networks>

**Long Short-Term Memory (LSTM)**

LSTM adalah ekstensi dari RNN, yaitu dengan memperbesar memori. LSTM digunakan sebagai *building blocks* dari layer-layer RNN. Kemampuan memori besar dari LSTM disebabkan oleh informasi yang tersimpan dalam memori sehingga LSTM dapat membaca (*read),* menulis (*write*), dan menghapus (*delete)* informasi yang tersimpan dalammemori. Memori ini tersimpan dalam *gated cell* yang berfungsi membuat keputusan menyimpan atau menghapus suatu informasi berdasarkan tingkat kepentingan (*importance)* informasi tersebut. Proses *assigning the importance* ini juga dilakukan melalui *weights* yang juga melalui proses *training* dalam algoritma *(it learns over time what information is important and what is not).*

Masalah *vanishing gradient* terselesaikan karena LSTM dapat menjaga besar *gradient* sehingga akurasi *training* menjadi tinggi dan memakan waktu yang lebih sedikit.

LSTM menggunakan gerbang analog dalam bentuk sigmoids sehingga memungkinkan dilakukannya *backpropagation* (karena sifat analognya)

<https://builtin.com/data-science/recurrent-neural-networks-and-lstm>

**Artificial Neural Network yang Telah Digunakan pada Penelitian Serupa**

* Characterization of external short circuit faults in electric vehicle Li-ion battery packs and prediction using artificial neural networks [<https://doi.org/10.1016/j.apenergy.2019.114253>]
  + Secara eksplisit dia pake BPNN:

*In this paper, a back-propagation neural network (BPNN) model is employed to predict the ESC cell current based on the measured voltage information (p.5).*

* + Secara implisit dia pake RNN:

*The current at time instant t is determined by the voltages at time instant t and the previous four-time intervals (p.5).* (Ini adalah karakteristik RNN)

* A Complete Machine Learning Approach for Predicting Lithium-Ion Cell Combustion [<https://doi.org/10.1016/j.tej.2020.106887>]
  + Pake RNN
* Predicting Thermal Runaway in Li-Ion Battery Employing Machine Learning Framework [<https://iopscience.iop.org/article/10.1149/MA2020-012429mtgabs/meta>]
  + Gabungan antara CNN + LTSM (versi lebih bagusnya dari RNN)
* A Neural Network-Based Method for Thermal Fault Detection in Lithium-Ion Batteries [<https://sci-hub.se/10.1109/TIE.2020.2984980>]
  + LSTM (versi 1 level di atas RNN)
* A Fast Charging–Cooling Coupled Scheduling Method for a Liquid Cooling-Based Thermal Management System for Lithium-Ion Batteries [<https://www.engineering.org.cn/en/10.1016/j.eng.2020.06.016>]
  + Feedforward, LM, backpropagation
* Machine Learning and Computer Vision Techniques to Predict Thermal Properties of

Particulate Composites [<https://arxiv.org/ftp/arxiv/papers/2010/2010.01968.pdf>]

* + Convolutional neural network (CNN)
* Li-ion battery temperature estimation based on recurrent neural networks
  + In this work, two types of recurrent neural networks (RNNs), which are long short-term memory-RNN (LSTM-RNN) and gated recurrent unit-RNN (GRU1-RNN)
* Data-model alliance network for the online multi-step thermal warning of energy storage system based on surface temperature diffusion
  + The multi-scale LSTM prediction network has two inputs and two LSTM networks
* Fault prognosis of battery system based on accurate voltage abnormity prognosis using long short-term memory neural networks
  + LSTM
* 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)
  + Wang et al. proposed a power consumption predicting and anomaly detection approach based on LSTM.

Kesimpulan:

* Kebanyakan penelitian prediksi masalah termal pada baterai Li-ion menggunakan arsitektur jenis *recurrent*
* Standard RNN adalah jenis ANN paling sederhana dalam kategori *recurrent* yang juga melibatkan konsep *backpropagation* dengan istilah BPTT
* Penggunaan RNN melahirkan dua masalah, *vanishing gradient* dan *exploding gradient* yang dapat diatasi menggunakan beberapa pendekatan
* Mayoritas menggunakan jenis LSTM, yaitu versi lebih lanjut daripada *standard RNN* yang secara otomatis menyelesaikan dua masalah pada *standard RNN*.
* Algoritma paling sederhana dalam ANN adalah algoritma *gradient descent* dengan menggunakan konsep *error function*. Output yang diinginkan dibandingkan dengan input, nilai eror yang didapat kemudian menjadi dasar dilakukannya *backpropagation* untuk merevisi nilai *weight* dengan cara mengurangi nilai *weight* lama dengan nilai *learning rate* yang dikalikan dengan nilai *gradient* dari fungsi *error* beberapa kali hingga mencapai nilai yang optimal

1The key difference between GRU and LSTM is that GRU's bag has two gates that are reset and update while LSTM has three gates that are input, output, forget. GRU is less complex than LSTM because it has less number of gates. If the dataset is small then GRU is preferred otherwise LSTM for the larger dataset

Apa semua RNN pke BPNN? Ya, tp berubah Namanya jd BPTT, tp BPTT ada gap yg buat dia ga akurat, shg ada yg beralih ke TLMP yg menyelesaikan masalah itu

*Most of the time when implementing a recurrent neural network in the common programming frameworks, backpropagation is automatically taken care of, but you need to understand how it works to troubleshoot problems that may arise during the development process.*

[*https://builtin.com/data-science/recurrent-neural-networks-and-lstm*](https://builtin.com/data-science/recurrent-neural-networks-and-lstm)