**BAB II**

**TINJAUAN PUSTAKA**

# **Penelitian Terdahulu**

Pada tahun 2018, Fittria Shofrotun Ni’mah, T. sutojo, dan De Rosal Ignatius Moses Setiadi membuat sebuah jurnal tentang Identifikasi Tumbuhan Obat Herbal Berdasarkan Citra Daun Menggunakan Algoritma Gray Level Co-occurrence Matrix dan K-Nearest Neighbor. Dalam penelitian tersebut masih memiliki kekurangan yaitu ekstraksi fitur dengan GLCM dan klasifikasi KNN dalam penelitian tersebut masih menghasilkan akurasi yang masih lebih kecil.

Erni Yusniar dan Aziz Kustiyo pernah membuat sebuah jurnal tentang Identifikasi Daun *Shorea* menggunakan KNN dengan Ekstraksi Fitur 2DPCA. Penelitian tersebut masih memiliki kekurangan. Yaitu, jumlah data yang digunakan untuk penelitian kurang bervariasi, dan akurasi rata-rata akurasi terbaik sebesar 75%.

Ditahun 2018, Jimmy Pujoseno membuat sebuah tugas akhir tentang Implementasi *Deep Learning* menggunakan *Convolutional Neural Network* untuk Klasifikasi Alat Tulis. Adapun kekurangan dari tugas akhir tersebut adalah programnya menggunakan pemrograman console tanpa GUI.

# **Tanaman Toga**

Kegiatan menanami pekarangan dengan tananam obat dikenal dengan nama toga. Program yang dahulu dinamai apotek hidup ini tengah digunakan oleh pemerintah indonesia. Istilah toga lebih mengacu kepada penataan pekarangan. Jadi tidak berarti tanaman yang hanya tanaman hias yang berkhasiat obat. Tanaman obat yang tergolong rempah-rempah atau bumbu dapur, tananam pagar, tanaman buah, tanaman sayur, atau bahkan tananam liar pun dapat ditata di pekarangan sebagai toga. Selain sebagai bahan obat bagi anggota keluarga yang sakit, tanaman tersebut dapat dimanfaatkan untuk aneka keperluan sesuai dengan kegunaan lainnya.

Toga adalah singkatan dari tanaman obat keluarga. Tanaman obat keluarga pada hakekatnya sebidang tanah baik di halaman rumah, kebun ataupun ladang yang digunakan untuk membudidayakan tanaman yang berkhasiat sebagai obat dalam rangka memenuhi keperluan keluarga akan obat-obatan. Kebun tanaman obat atau bahan obat dan selanjutnya dapat disalurkan kepada masyarakat, khususnya obat yang berasal dari tumbuh-tumbuhan.

Tanaman toga ini dapat dimanfaatkan sebagai bahan ramuan tradisional. Dilihat dari aspek pemanfaatannya tanaman obat keluarga di bagi menjadi beberapa jenis,di antaranya sebagai berikut :

1. Jenis tanaman obat keluarga yang dimanfaatkan daunya. Contohnya, Seledri, manfaatnya untuk menyembuhkan tekanan darah tinggi.
2. Jenis tanaman obat keluarga yang dimanfaatkan batangnya. Kayu manis dimanfaatkan untuk mengobati penyakit batuk, sesak napas, nyeri lambung, perut kembung, diare, rematik, dan menghangatkan lambung.
3. Jenis tanaman obat keluarga yang dimanfaatkan akarnya.
4. Jenis tanaman obat keluarga yang dimanfaatkan umbinya.

# **Morfologi Daun**

**(beri kata2, kenapa ada angka 1 dan 2)**

1. Bentuk umum

Bentuk daun sangat bervariasi. Sepintas dapat diamati bahwa bentuk daun ada yang bulat, bulat telur, panjang, seperti pita dan ada juga yang berbentuk segitiga, runcing, seperti tombak, jantung, ginjal, dan lain- lain. Bagian tepinya ada yang rata dan ada yang berlekuk.

1. Variasi

Jika perhatikan lebih saksama bagian dari helai daun, ternyata variasi dapat dijumpai pada bagian ujung daun, pangkal daun, susunan tulang daun, tepi daun, dan daging daun.

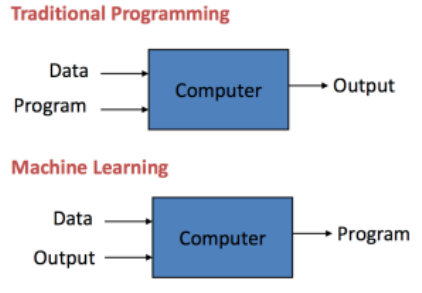
# **Citra Digital**

Citra digital adalah suatu citra elektronik yang diambil dari dokumen, seperti foto, buku, maupun sebuah video. Proses perubahan citra analog menjadi citra digital dinamakan dengan digitasi. Digitasi adalah proses mengubah gambar, *teks*, atau suara dari benda yang dapat dilihat ke dalam data elektronik dan dapat disimpan serta diproses untuk keperluan lainnya.

Citra digital pada komputer dipetakan menjadi bentuk grid atau elemen piksel berbentuk matriks 2 dimensi. Setiap piksel-piksel tersebut memiliki angka yang merepresentasikan channel warna. Angka pada setiap piksel disimpan secara berurutan oleh komputer dan sering dikurangi untuk keperluan kompresi maupun pengolahan tertentu.

# ***Machine Learning***

*Machine Learning* adalah serangkaian teknik yang dapat membantu dalam menangani dan memprediksi data yang sangat besar dengan cara merepresentasikan data-data tersebut dengan algoritma pembelajaran. *Machine Learning* dapat membuat komputer memprogram diri mereka sendiri. Jika pemrograman adalah pekerjaan untuk membuat otomasi, maka *Machine Learning* mengotomatisasi proses otomasi. Pada dasarnya *Machine* *Learning* membiarkan data melakukan pekerjaan. gambaran umum *Machine Learning* dibandingkan dengan pemrograman secara tradisional ditunjukkan pada gambar 2.1.



Gambar 2.1. Perbandingan Pemrograman

Pada gambar 2.1 dapat dilihat bahwa pemrograman secara tradisional data dan program dijalankan di komputer untuk menghasilkan *output*. Sedangkan pada *Machine Learning* data dan *output* dijalankan di komputer untuk membuat sebuah program. Ada banyak algoritma *Machine* *Learning* yang dikembangkan setiap tahunnya. Setiap algoritma pembelajaran mesin memiliki tiga komponen penting, antara lain:

1. Representasi: bagaimana merepresentasikan pengetahuan. Contohnya termasuk *Decision tree, Neural Network, Support Vector Machine* dan lain-lain.
2. Evaluasi: cara mengevaluasi prediksi dan hipotesis. Contohnya meliputi *Mean Squared Error, Cost function* dan lain-lain.
3. Optimasi: cara program dari model dihasilkan dan proses pencarian parameter terbaik.

Misalnya *Convex Optimization* dan *Gradient Descent* Secara umum algoritma pembelajaran ada tiga jenis cara untuk melakukan pembelajaran pada *Machine Learning*, yakni:

* 1. *Supervised Learning*: Data pembelajaran mencakup keluaran yang sudah ditentukan.
  2. *Unsupervised Learning*: Data pembelajaran tidak mencakup keluaran yang ditentukan.
  3. *Reinforcement Learning*: Pemberian hadiah dari setiap serangkaian tindakan yang dilakukan.

Ada 5 langkah dasar yang digunakan untuk melakukan tugas *Machine Learning* dan tugas ini sangatlah penting dalam mempersiapkan solusi untuk segala bentuk permasalahan dalam *Machine Learning* maupun *Deep Learning*:

1. Mengumpulkan data: Dibentuk dari beberapa file yang berisi deretan data yang dapat dipelajari dan stelah itu dipisah antara fitur masukkan dan keluaran
2. Mempersiapkan data: Penentuan kualitas data dan *preprocess* data sehingga hasil yang didapat juga baik.
3. Melatih sebuah model: Langkah ini melibatkan pemilihan algoritma dan representasi data yang tepat dalam bentuk model.
4. Mengevaluasi model: Untuk menguji keakuratan berdasarkan bagian *test set* pada *dataset*.
5. Meningkatkan kinerja: Langkah ini melibatkan pemilihan *hyperparameter* untuk meningkatkan efisiensi dan biasanya menggunakan *cross-validation.*

# ***Deep Learning***

*Deep Learning* merupakan salah satu bidang dari *Machine Learning* yang memanfaatkan jaringan syaraf tiruan untuk implementasi permasalahan dengan *dataset* yang besar. Teknik *Deep Learning* memberikan arsitektur yang sangat kuat untuk *Supervised Learning*. Dengan menambahkan lebih banyak lapisan maka model pembelajaran tersebut bisa mewakili data citra berlabel dengan lebih baik.

Pada *Machine Learning* terdapat teknik untuk menggunakan ekstraksi fitur dari data pelatihan dan algoritma pembelajaran khusus untuk mengklasifikasi citra maupun untuk mengenali suara. Namun, metode ini masih memiliki beberapa kekurangan baik dalam hal kecepatan dan akurasi. Aplikasi konsep jaringan syaraf tiruan yang dalam (banyak lapisan) dapat ditangguhkan pada algoritma *Machine Learning* yang sudah ada sehingga komputer sekarang bisa belajar dengan kecepatan, akurasi, dan skala yang besar. Prinsip ini terus berkembang hingga *Deep Learning* semakin sering digunakan pada komunitas riset dan industri untuk membantu memecahkan banyak masalah data besar seperti *Computer vision, Speech recognition, dan Natural Language Processing*.

Dalam *Deep Learning*, metode CNN atau *Convolutional Neural Network* sangatlah bagus dalam menemukan fitur yang baik pada citra ke lapisan berikutnya untuk membentuk hipotesis nonlinier yang dapat meningkatkan kompleksitasan sebuah model. Model yang kompleks tentunya akan membutuhkan waktu pelatihan yang lama sehingga di dunia *Deep Learning* pengunaan GPU sudah sangatlah umum(Danukusumo, 2017).

## ***Feature* *learning***

Berbeda dengan *feature engineering* yang digunakan dalam *machine learning* pada umumnya, *feature learning* adalah metode dimana proses *feature extraction* dilakukan secara otomatis dan adaptif oleh model. *Feature learning* muncul karena *feature engineering* sangatlah terbatas dalam artian tiap kasus data yang berbeda memerlukan *feature extraction* yang berbeda. Hal tersebut menjadikan metode *feature engineering* tidak memiliki kemampuan generalisasi pada keragaman jenis data seperti yang dibutuhkan dalam kasus klasifikasi objek pada citra.

Metode *supervised feature learning* yang banyak berkembang adalah *deep learning feature learning* dimana menggunakan *deep network* yang melakukan proses ekstraksi fitur dan klasifikasi menggunakan learning. Model *deep learning* yang umum digunakan pada pengolahan citra adalah *Convolutional Neural Network* (CNN).

## ***Convolution* *Neural* *Network***

*Convolution Neural Network,* biasanya disingkat dengan CNN atau ConvNet, adalah salah satu kelas *deep feed-forward artificial neural network* yang banyak diaplikasikan pada analisis citra (Atom et al. 2018). CNN terinspirasi oleh proses-proses biologi di mana pola konektivitas antar neuron menyerupai organisasi *visual cortex pada binatang.*

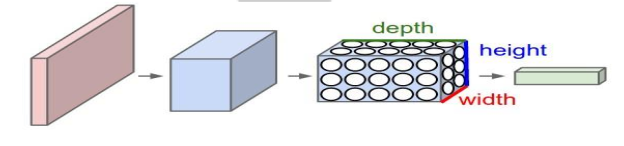
*Convolutional Neural Network (CNN) adalah pengembangan dari Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena ke dalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik. Pada tahun 2012, Alex Krizhevsky dengan penerapan CNN miliknya berhasil menjuarai kompetisi *ImageNet Large Scale Visual Recognition Challenge 2012.* Prestasi tersebut menjadi momen pembuktian bahwa metode *Deep Learning*, khususnya CNN. Metode CNN terbukti berhasil mengungguli metode M*achine Learning* lainnya seperti SVM pada kasus klasifikasi objek pada citra. (Suartika ,I Wayan, Arya Yudhi Wijaya, dan Rully Soelaiman).

### ***Konsep Convolutional Neural* *Network***

Cara kerja CNN memiliki kesamaan pada MLP, namun dalam CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi. Nama konvolusi sendiri merupakan operasi aljabar linear yang mengkalikan matriks dari filter pada citra yang akan diproses. Proses ini disebut dengan lapisan konvolusi dan merupakan salah satu jenis dari banyak lapisan yang bisa dimiliki dalam satu jaringan. Meskipun begitu lapisan konvolusi merupakan lapisan utama yang paling penting untuk digunakan. Jenis lapisan lain yang biasa digunakan adalah *Pooling Layer*, yakni lapisan yang digunakan untuk mengambil nilai maksimal atau nilai rata-rata dari bagain-bagian piksel pada citra.

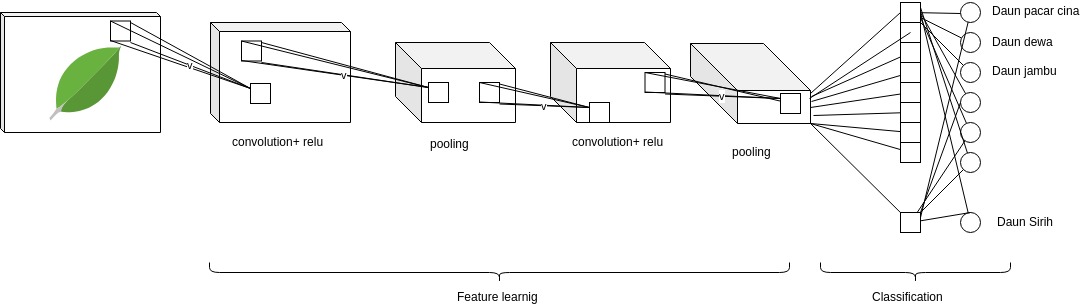
### ***Arsitektur Convolution Neural* *Network***

Dapat dilihat gambaran umum arsitektur *Convolution Network*: pada Gambar 2.2 dimana setiap lapisannya input yang dimasukkan memiliki volume yang berbeda dan diwakili dengan kedalaman, tinggi dan lebar.



Gambar 2.2 Contoh Jaringan CNN

Setiap besaran yang didapat tergantung dari hasil filtrasi dari lapisan sebelumnya dan juga banyaknya filter yang digunakan. Model jaringan seperti ini sudah terbukti sangat ampuh dalam menangani permasalah klasifikasi citra. Arsitektur dari CNN dibagi menjadi 2 bagian besar, *Feature Learning Layer* dan *Classification* (MLP), seperti pada Gambar 2.3.

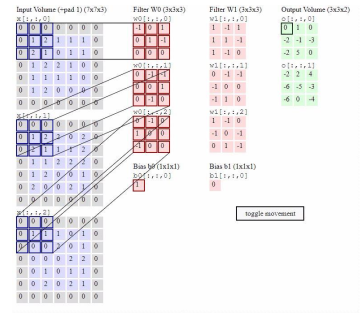


Gambar 2.3 Arsitektur CNN

Gambar 2.3 menjelaskan komponen-komponen yang terdapat pada *feature* *learnig* dan *classification*, konvolusi dan polling termasuk dalam feature learning , untuk mendapatkan *feature* bisa dilakukan berulang-ulang sesuai model yang diinginkan, pada gambar 2.3 terdapat 2 kali konvolusi dan 2 kali *polling*, sedangkan pada c*lassification* sama dengan *multi layer perceptron* terdapat *input layer, hidden layer* dan *output layer*

### ***Convolution* *Layer***

*Convolution Layer* melakukan operasi konvolusi pada *output* dari layer sebelumnya. Layer tersebut adalah proses utama yang mendasari sebuah CNN. Konvolusi adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Dalam pengolahan citra, konvolusi berati mengaplikasikan sebuah kernel pada citra di semua *offset* secara keseluruhan adalah citra yang akan dikonvolusi. Kernel bergerak dari sudut kiri atas ke kanan bawah. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input.



(gambar e)

*Convolutional layer* terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (*pixels*). Sebagai contoh, layer pertama pada *feature extraction* layer biasanya adalah *convolution layer* dengan ukuran 5x5x3. Panjang 5 *pixels*, tinggi 5 *pixels* dan tebal atau jumlah 3 buah sesuai dengan *channel* dari *image* yaitu *red, green, blue*, ketiga filter ini akan digeser keseluruh bagian dari gambar. Setiap pergeseran akan dilakukan operasi “dot” antara *input* dan nilai dari *filter* tersebut sehingga menghasilkan sebuah *output* atau biasa disebut sebagai *activation map* atau feature *map*, seperti pada gambar 2.4.

Gambar 2.4 konvolusi CNN

*Stride* adalah parameter yang menentukan berapa jumlah pergeseran filter. Jika nilai *stride* adalah 1, maka *conv filter* akan bergeser sebanyak 1 *pixels* secara *horizontal* lalu *vertical*. Pada gambar 2.4 stride yang digunakan adalah 2. Semakin kecil stride maka akan semakin detail informasi yang kita dapatkan dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan *stride* yang besar. Namun perlu diperhatikan bahwa dengan menggunakan *stride* yang kecil kita tidak selalu akan mendapatkan performa yang bagus.

*Padding* atau *Zero* Padding adalah parameter yang menentukan jumlah *pixels* (berisi nilai 0) yang akan ditambahkan di setiap sisi dari input. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi *output* dari *conv layer (Feature Map)*. Tujuan dari penggunaan *padding* adalah dimensi *output* dari *conv* *layer* selalu lebih kecil dari inputnya (kecuali penggunaan 1x1 *filter* dengan *stride* 1). *Padding* juga mampu meningkatkan performa dari model, karena *conv* *filter* akan fokus pada informasi yang sebenarnya yaitu yang berada diantara *zero* *padding* tersebut. Pada ilustrasi diatas, dimensi dari input sebenarnya adalah 5x5, jika dilakukan *convolution* dengan *filter* 3x3 dan *stride* sebesar 2, maka akan didapatkan *feature* *map* dengan ukuran 2x2. Namun jika kita tambahkan *zero* *padding* sebanyak 1, maka *feature* *map* yang dihasilkan berukuran 3x3 (lebih banyak informasi yang dihasilkan).

### ***Convolution* *Operation***

Operasi konvolusi adalah operasi pada dua fungsi argumen bernilai nyata (Goodfellow, Bengio, & Courville, 2016). Operasi ini menerapkan fungsi *output* sebagai *Feature* *Map* dari input citra. Input dan *output* ini dapat dilihat sebagai dua argumen bernilai riil. Secara formal operasi konvolusi dapat ditulis dengan persamaan (2.1).

|  |  |
| --- | --- |
| *s(t) = (x\*w)(t)* | (2.1) |

Fungsi s(t) memberikan *output* tunggal berupa *Feature* *Map*, argumen pertama adalah input yang merupakan x dan argumen kedua w sebagai kernel atau filter. Jika kita melihat input sebagai citra dua dimensi, maka kita bisa mengasumsikan t sebagai *pixel* dan menggantinya dengan i dan j. Oleh karena itu, operasi untuk konvolusi ke input dengan lebih dari satu dimensi dapat ditulis dengan persamaan (2.2).

|  |  |
| --- | --- |
| *S(i,j) = (K\*I) (i,j) = ∑ ∑ I (i – m, j-n) K (m,n)* | (2.2) |

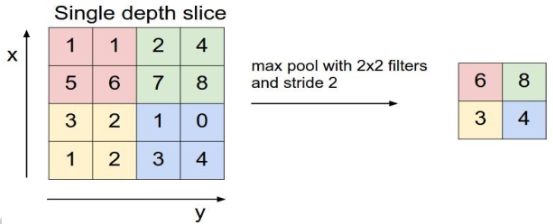
Persamaan di atas adalah perhitungan dasar dalam operasi konvolusi dimana i dan j adalah piksel dari citra. Perhitungannya bersifat komutatif dan muncul saat K sebagai kernel, I sebagai input dan kernel yang dapat dibalik relatif terhadap input. Sebagai alternatif, operasi konvolusi dapat dilihat sebagai perkalian matriks antara citra masukan dan kernel dimana keluarannya dapat dihitung dengan *dot* *product*(Rismiyati, 2016). Kita juga dapat menentukan volume *output* dari masing-masing lapisan dengan *hyperparameters*. *Hyperparameter* yang digunakan pada persamaan dibawah ini digunakan untuk menghitung berapa banyak *neuron* aktivasi dalam sekali *output*. Untuk menghitung dimensi dari *feature* *map* kita bisa gunakan persamaan (2.3)

|  |  |
| --- | --- |
| *output* = (W−N+2P)/S +1  W = Panjang/Tinggi Input  N = Panjang/Tinggi Filter  P = *Zero Padding*  S = *Stride* | (2.3) |

### ***Maxpooling* *Operation***

*Pooling Layer* adalah lapisan yang menggunakan fungsi dengan *Feature Map* sebagai masukan dan mengolahnya dengan berbagai macam operasi statistik berdasarkan nilai piksel terdekat. Pada model CNN, lapisan *Pooling* biasanya disisipkan secara teratur setelah beberapa lapisan konvolusi. Lapisan Pooling yang dimasukkan diantara lapisan konvolusi secara berturut-turut dalam arsitektur model CNN dapat secara progresif mengurangi ukuran volume *output* pada *Feature Map*, sehingga mengurangi jumlah parameter dan perhitungan di jaringan, dan untuk mengendalikan *Overfitting*.

Lapisan *Pooling* bekerja di setiap tumpukan *Feature Map* dan mengurangi ukurannya. Bentuk lapisan *Pooling* yang paling umum adalah dengan menggunakan *filter* berukuran 2x2 yang diaplikasikan dengan langkah sebanyak 2 dan kemudian beroperasi pada setiap irisan dari input. Bentuk seperti ini akan mengurangi *Feature Map* hingga 75% dari ukuran aslinya.



Gambar 2.5 Contoh Operasi Max Pooling

Lapisan *Pooling* akan beroperasi pada setiap irisan kedalaman *volume* *input* secara bergantian. Pada gambar 2.5, lapisan *pooling* menggunakan salah satu operasi maksimal yang merupakan operasi yang paling umum. Gambar 2.5 menunjukkan operasi dengan langkah 2 dan ukuran *filter* 2x2. Dari ukuran input 4x4, pada masing-masing 4 angka pada input operasi mengambil nilai maksimalnya dan membuat ukuran *output* baru menjadi 2x2.

### ***Fully* *Connection***

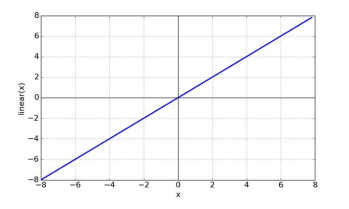
Lapisan *Fully-Connected* adalah lapisan di mana semua *neuron* aktivasi dari lapisan sebelumnya terhubung semua dengan *neuron* di lapisan selanjutnya seperti halnya jaringan saraf tiruan biasa. Setiap aktivasi dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua *neuron* di lapisan *Fully-Connected*. Lapisan *Fully-Connected* biasanya digunakan pada metode *Multi- Lapisan Perceptron* dan bertujuan untuk mengolah data sehingga bisa diklasifikasikan.

Perbedaan antara lapisan *Fully-Connected* dan lapisan konvolusi biasa adalah *neuron* di lapisan konvolusi terhubung hanya ke daerah tertentu pada input, sementara lapisan *Fully-Connected* memiliki *neuron* yang secara keseluruhan terhubung. Namun, kedua lapisan tersebut masih mengoperasikan produk dot, sehingga fungsinya tidak begitu berbeda (Danukusumo, 2017).

### ***Activation* *Function***

Sesuai dengan namanya, *activation function* befungsi untuk menentukan apakah neuron tersebut harus “aktif” atau tidak berdasarkan dari *weighted* *sum* dari input. Secara umum terdapat 2 jenis *activation* *function*, *Linear* dan *Non*-*Linear* *Activation function.*

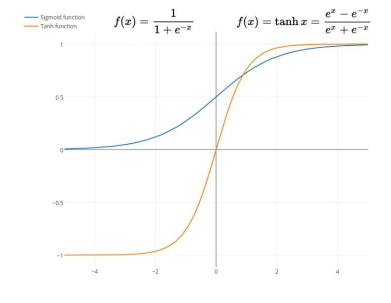
a. Linear Function



Gambar 2.6 Linear Function

Gambar 2.6 menunjukkan grafis dari fungsi linier, bisa dikatakan secara “*default*” *activation function* dari sebuah neuron adalah Linear. Jika sebuah *neuron* menggunakan *linear* *function*, maka keluaran dari *neuron* tersebut adalah *weighted* *sum* dari input + bias.

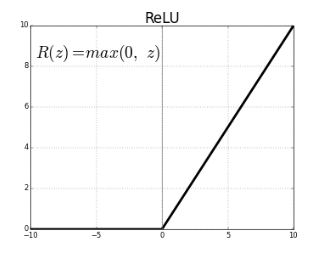
b. Sigmoid and Tanh Function (Non-Linear)



Gambar 2.7 Sigmoid and Tanh Function (Non-Linear)

Gambar 2.7 menunjukkan grafis dari fungsi sigmoid dan tanh. Sigmoid *function* mempunyai rentang antara 0 hingga 1 rentang seperti pada Gambar 2.7 kurva warna biru sedangkan dari Tanh adalah -1 hingga 1 seperti pada gambar 2.7 kurva warna merah. Kedua fungsi ini biasanya digunakan untuk klasifikasi 2 class atau kelompok data.

c. ReLU (Non-Linear)



Gambar 2.8 ReLU (Non-Linear)

Aktivasi ReLU (*Rectified Linear Unit*) merupakan lapisan aktivasi pada model CNN yang mengaplikasikan fungsi f(x)=max(0,x) yang berarti fungsi ini melakukan *Thresholding* dengan nilai nol terhadap nilai piksel pada input citra. Aktivasi ini membuat seluruh nilai piksel yang bernilai kurang dari nol pada suatu citra akan dijadikan 0. Gambar 2.8 menunjukkan grafis dari fungsi ReLU.

### ***Softmax***

*Softmax Classifier* merupakan bentuk lain dari algoritma *Logistic* *Regression* yang dapat kita gunakan untuk mengklasifikasi lebih dari dua kelas. Standar klasifikasi yang biasa dilakukan oleh algoritma *Logistic* *Regression* adalah tugas untuk klasifikasi kelas biner. Pada *Softmax* terdapat bentuk persamaan (2.4)

|  |  |
| --- | --- |
|  | (2.4) |

Notasi f j menunjukkan hasil fungsi untuk setiap elemen ke-j pada vektor keluaran kelas. Argumen z adalah hipotesis yang diberikan oleh model pelatihan agar dapat diklasifikasi oleh fungsi Softmax. Softmax juga memberikan hasil yang lebih intuitif dan juga memiliki interpretasi probabilistik yang lebih baik dibanding algoritma klasifikasi lainya. Softmax memungkinkan kita untuk menghitung probabilitas untuk semua label. Dari label yang ada akan diambil sebuah vektor nilai bernilai riil dan merubahnya menjadi vektor dengan nilai antara nol dan satu yang bila semua dijumlah akan bernilai satu.

### ***Learning* *Rate***

Penggunaan parameter *learning rate* memiliki pengaruh penting terhadap waktu yang dibutuhkan untuk tercapainya target yang diinginkan. Secara perlahan akan mengoptimalkan nilai perubahan bobot dan menghasilkan error yang lebih kecil (Fajri, 2011). Variabel *learning rate* menyatakan suatu konstanta yang bernilai antara 0.1-0.9. Nilai tersebut menunjukkan kecepatan belajar dari jaringannya. Jika nilai learning rate yang digunakan terlalu kecil maka terlalu banyak *epoch* yang dibutuhkan untuk mencapai nilai target yang diinginkan, sehingga menyebabkan proses training membutuhkan waktu yang lama.

Semakin besar nilai *learning rate* yang digunakan maka proses pelatihan jaringan akan semakin cepat, namun jika terlalu besar justru akan mengakibatkan jaringan menjadi tidak stabil dan menyebabkan nilai error berulang bolak-balik diantara nilai tertentu, sehingga mencegah error mencapai target yang diharapkan. Oleh karena itu pemilihan nilai variable *learning rate* harus seoptimal mungkin agar didapatkan proses *training* yang cepat (Hermawan, 2006).

### ***Cross Entropy Loss* *Function***

*Loss Function* atau *Cost Function* merupakan fungsi yang menggambarkan kerugian yang terkait dengan semua kemungkinan yang dihasilkan oleh model. *Loss Function* bekerja ketika model pembelajaran memberikan kesalahan yang harus diperhatikan. *Loss Function* yang baik adalah fungsi yang menghasilkan error yang diharapkan paling rendah. Ketika suatu model memiliki kelas yang cukup banyak, perlu adanya cara untuk mengukur perbedaan antara probabilitas hasil hipotesis dan probabilitas kebenaran yang asli, dan selama pelatihan banyak algoritma yang dapat menyesuaikan parameter sehingga perbedaan ini diminimalkan. *Crossentropy* adalah pilihan yang masuk akal. Gambaran umum algoritma ini adalah meminimalkan kemungkinan log negatif dari dataset, yang merupakan ukuran langsung dari performa prediksi model.

### ***Forward* *Propagation***

Proses *forward propagation* pada jaringan CNN dilakukan untuk meneruskan nilai pada lapisan masukan hingga pada lapisan keluaran. Nilai ini diteruskan melalui lapisan konvolusi, subsampling dan lapisan *fully connected* sesuai dengan urutan lapisan tersebut ditempatkan pada jaringan yang digunakan. Maka dari itu perlu dilakukan perancangan bentuk struktur CNN yang akan digunakan terlebih dahulu. Urutan proses runut maju pada CNN dapat diringkas sebagai berikut :

* 1. Inisialisasi nilai awal pada filter pada lapisan konvolusi dan bobot pada lapisan *fully connected* dengan nilai acak, dan bias dengan nilai awal 0.
  2. Melakukan proses konvolusi gambar masukan sesuai dengan filter pada lapisan konvolusi yang dioperasikan pada gambar masukan (I). Tanda(\*) menotasikan proses konvolusi, dan σ(x) menotasikan fungsi aktivasi, seperti pada persamaan (2.5).

|  |  |
| --- | --- |
|  | (2.5) |

* 1. *Feature maps* yang didapatkan akan dikurangi ukurannya untuk mengurangi kompleksitas perhitungan pada lapisan selanjutnya. Proses ini dilakukan pada lapisan *subsampling*. Proses subsampling dengan menggunakan *max pooling*, atau meloloskan nilai tertinggi dari *feature* *maps* yang ada dalam sebuah jendela *subsampling*.
  2. Hasil dari lapisan *subsampling* merupakan *feature* *maps* yang telah direduksi ukurannya, jika pada struktur lapisan CNN yang digunakan terdapat lapisan konvolusi setelah lapisan *subsampling*, maka tahapan selanjutnya adalah sama dengan tahap 1-3, jika tidak maka lanjutkan ke tahap 5.
  3. *Feature* *maps* yang didapat dari lapisan subsampling terakhir merupakan *feature* *maps* yang akan digunakan pada lapisan *fully* *connected* sebagai fitur untuk melakukan klasifikasi. *Feature* *maps* yang berupa matriks akan diuraikan menjadi vector yang. Proses ini disebut *vectorization* and *concatenation* (Zhang, 2016. Fitur yang masuk ke dalam lapisan fully connected (f) merupakan hasil proses vektorisasi (F(x)) dari hasil subsampling pada lapisan sebelumnya (), proses ini menggabungkan seluruh n buah *feature* *maps,* seperti yang ditujukkan pada persamaan(2.6)

|  |  |
| --- | --- |
|  | (2.6) |

* 1. Selanjutnya adalah proses perhitungan prediksi target dari fitur yang masuk ke dalam lapisan *fully connected*. Nilai prediksi kelas (y(i)̂) ini dilakukan dengan melakukan perhitungan menggunakan persamaan(2.7). Perhitungan pada persamaan ini menggunakan fitur dari lapisan subsampling sebelumnya (f(j)) yang dikalikan dengan bobot yang terkoneksi (W(i,j)) dan ditambahkan dengan bias (b(i)).

|  |  |
| --- | --- |
|  | (2.7) |

* 1. Untuk mengetahui seberapa baik proses pembelajaran telah dilakukan, maka nilai *Loss* dihitung.

### ***Backward* *Propagation***

Proses untuk Proses untuk memperbaharui nilai filter dan bobot pada jaringan adalah proes propagasi balik. Perhitungan perubahan nilai bobot dihitung dimulai dari lapisan *fully* *connected*. Pada lapisan ini perubahan bobot dicari dengan mencari *derivatif* *loss* *function* terhadap bobot (Zhang, 2016). Perhitungan perubahan (ΔW(i,j)) yang terhubung dengan node penghasil nilai fitur f(j) berdasarkan selisih prediksi kelas dari data ke i (ŷ(i)) dengan target aktual dari data ke i (y(i)) pada lapisan fully connected dapat dilihat pada persamaan (2.8).

|  |  |
| --- | --- |
|  | (2.8) |

Perubahan bias (Δb(i,j)) juga dapat dilakukan dengan mencari derivatif loss function terhadap bias. Perubahan bias dapat dihitung menggunakan persamaan(2.9).

|  |  |
| --- | --- |
|  | (2.9) |

Selanjutnya adalah menghitung perubahan nilai filter pada lapisan konvolusi, perubahan ini diasarkan atas galat pada lapisan subsampling. Sehingga, sebelum menghitung perubahan bobot pada lapisan konvolusi, perlu dilakukan upsampling dari galat, karena setelah melakukan konvolusi feature maps melewati lapisansubsampling dan proses vektorisasi. Perhitungan perubahan feature maps (Δf) dilakukan dengan persamaan (2.10)

|  |  |
| --- | --- |
|  | (2.10) |

Setelah didapat perubahan dari *feature* *maps* yang masih berbentuk vector panjang, maka dilakukan proses untuk membalikkan vector ini ke bentuk matriks 2 dimensi. Perubahan ini dapat dinotasikan pada persamaan (2.11)

|  |  |
| --- | --- |
|  | (2.11) |

Proses upsampling adalah merubah matriks {} yang merupakan matriks hasil subsampling kembali ke ukuran awal sebelum dilakukan proses *subsampling*. Hal ini dilakukan dengan meneruskan nilai matriks {} kepada koordinat dari *feature* *maps* yang diloloskan nilainya pada proses subsampling (berkontribusi). Sedangkan untuk koordinat yang tidak diloloskan nilainya pada proses subsampling dapat diberi nilai 0. Penerusan nilai ini dinotasikan pada persamaan (2.12).

|  |  |
| --- | --- |
|  | (2.12) |

Setelah proses s*upsampling*, maka (i,j) dapat digunakan untuk menghitung perubahan nilai pada filter konvolusi di lapisan sebelumnya. Pencarian perubahan nilai filter (,p) dilakukan dengan melakukan konvolusi gambar masukan (I) dengan menggunakan (,σ). Proses pencarian nilai perubahan nilai filter konvolusi dapat dinotasikan pada persamaan (2.13)

|  |  |
| --- | --- |
|  | (2.13) |
|  |  |

Pada lapisan konvolusi juga terdapat bias, nilai bias juga diperbaharui untuk mendukung proses pembelajaran. Perhitungan perubahan nilai bias (,p) dilakukan hampir sama dengan perhitungan perubahan nilai filter konvolusi, namun tidak melibatkan nilai masukan. Sehingga perubahan nilai bias sama dengan jumlah seluruh (,σ) setelah *upsampling* seperti yang telah dinotasikan pada persamaan (2.14)

|  |  |
| --- | --- |
|  | (2.14) |

Setelah menghitung perubahan pada tiap-tiap lapisan, maka proses emperbaharui nilai filter, bias pada lapisan konvolusi, bobot pada lapisan *fully connected*, serta bias yang lama dapat dilakukan sebagaimana dijabarkan pada persamaan (2.15)(2.16)(2.17)(2.18)

|  |  |
| --- | --- |
|  | (2.15) |
|  | (2.16) |
| W = W - ɑ **.** ∆W | (2.17) |
| b = b - ɑ **.** ∆b | (2.18) |

Proses ini dilakukan hingga kondisi terhenti ditemukan, kondisi terhenti ini bisa saja berupa *epoch* maksimum yang tercapai atau nilai loss yang berada dibawah batasan yang ditetapkan. Proses perubahan nilai bobot, bias dan filter dilakukan setiap satu data masuk ke dalam jaringan

### ***Stochastic Gradien* *Descent***

*Gradient Descent* adalah salah satu algoritma yang paling populer untuk melakukan optimasi pada model jaringan syaraf tiruan dan algoritma ini adalah cara yang paling sering dipakai dalam berbagai macam model pembelajaran. Pada dasarnya ketika akan melatih sebuah model, kita membutuhkan sebuah *Loss Function* yang dapat memungkinkan kita untuk mengukur kualitas dari setiap bobot atau parameter tertentu. Tujuan pengoptimalan adalah untuk menemukan parameter yang dapat meminimalkan *Loss Function* (Ruder, 2017). *Gradient Descent* bekerja dengan cara meminimalkan fungsi J(θ) yang memiliki parameter θ dengan memperbarui parameter ke suatu arah menurun. *Gradient Descent* memiliki *Learning Rate* (η) yang digunakan untuk menentukan langkah-langkah yang kita ambil untuk mencapai titik minimum. Hal ini bisa digambarkan dimana suatu objek akan menuruni sebuah bukit dengan langkah tersebut hingga mencapai pada lembah (titik minimum).

SGD sering melakukan update dengan varians tinggi yang menyebabkan fungsi objektif meningkat secara tidak beraturan. Di satu sisi hal ini dapat membuat *Loss Function* melompat ke titik minimal yang baru dan berpotensi melompat ke minimum yang tidak pasti. Namun, hal ini dapat dicegah dengan cara mengurangi learning rate, dan SGD akan menuruni *Loss Function* ke titik minimum dengan optimal.

# **Android**

Aplikasi Android ditulis dengan menggunakan bahasa pemrograman Java. Kode Java dikompilasi beserta dengan data dan file sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh tool apt ke dalam paket Android, dan hasilnya berupa file archive berekstensi .apk. File inilah yang di distribusikan atau diunduh pengguna dan menginstalnya ke dalam perangkat mobile. Semua kode yang ada di dalam sebuah file berekstensi .apk dianggap sebagai sebuah aplikasi. Dalam banyak hal, setiap aplikasi Android hidup di dalam dunianya sendiri:

1. Secara default, setiap aplikasi berjalan pada proses Linuxnya masing-masing. Android memulai prosesnya ketika kode yang ada di dalam aplikasi akan dieksekusi, dan akan mengakhirinya ketika proses itu tidak lagi diperlukan(*system resources* akan digunakan oleh aplikasi lain).
2. Setiap proses mempunyai *virtual machine*(VM) masing-masing, sehingga kode aplikasi akan dipisahkan dari aplikasi yang lainnya.
3. Secara default, setiap aplikasi diberikan sebuah user ID Linux yang unik. Aturan penggunaan telah ditentukan sedemikian rupa sehingga hanya pengguna yang berhak saja yang dapat mengakses dan menggunakan aplikasi tersebut walaupun ada beberapa cara untuk mengekspor sebuah aplikasi ke dalam aplikasi lainnya.

User ID yang sama juga dapat digunakan oleh dua buah aplikasi sehingga aplikasi tersebut dapat melihat file yang ada di kelolah oleh kedua aplikasi tersebut. User ID dapat mengatur dua aplikasi, sehingga kedua aplikasi tersebut dapat saling berbagi file dan *virtual machine* untuk menghemat *system resources.*

# ***Javascript Objet Notation(JSON)***

JSON singkatan dari JavaScript Object Notation adalah suatu format ringkas pertukaran data komputer. Formatnya berbasis teks dan terbaca-manusia serta digunakan untuk merepresentasikan struktur data sederhana dan larik asosiatif (disebut objek). Format JSON sering digunakan untuk mentransmisikan data terstruktur melalui suatu koneksi jaringan pada suatu proses yang disebut serialisasi. Aplikasi utamanya adalah pada pemrograman aplikasi web AJAX dengan berperan sebagai alternatif terhadap penggunaan tradisional format XML.