BAB III
LANDASAN TEORI

3.1. 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 kita lihat ke dalam data elektronik dan dapat kita simpan 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.

3.2. Klasifikasi Citra

Klasifikasi citra merupakan sebuah pekerjaan untuk memasukkan sebuah citra dan menetapkannya ke sebuah kategori. Ini adalah salah satu permasalahan dalam *Computer Vision* yang dapat disederhanakan dan memiliki berbagai macam aplikasinya. Salah satu aplikasi dalam klasifikasi citra adalah pengklasifikasian nama tempat pada suatu citra.

Dalam klasifikasi yang terawasi, setiap masukkan citra pada *training set* diberi label. Saat klasifikasi, label tersebut akan menjadi perbandingan dengan hasil hipotesis yang diberikan oleh model pembelajaran dan akan menghasilkan nilai error. Klasifikasi yang terawasi bisa sangat efektif dan akurat dalam mengklasifikasikan citra tempat maupun objek lainnya. Banyak metode dan algoritma yang dapat mendukung proses klasifikasi yang terawasi terutama dengan teknik *Deep Learning.*

3.3. Candi-candi Indonesia

Candi-candi Indonesia merupakan bangunan peninggalan kerajaan Hindu dan Budha pada masa lalu. Candi-candi tersebut biasanya menjadi tempat diadakannya berbagai macam upacara dan kegiatan religi lainnya. Candi juga identik dengan tempat yang diibaratkan sebagai tempat para dewa, sehingga banyak dari arsitekturnya memiliki nilai seni yang biasanya disampaikan lewat relief dan bentuk bangunannya.

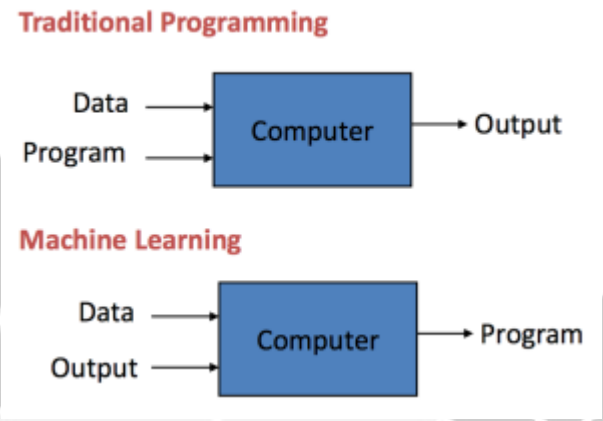
Candi-candi ini tersebar hampir di seluruh Indonesia, masing-masing memiliki keunikannya berdasarkan tempat dan pembuatnya. Keunikan ini juga dilihat dari megahnya candi tersebut dibuat seperti Candi Borobudur dan Candi Prambanan. Hal ini membuat candi-candi ini merupakan nilai sejarah yang tidak tergantikan dan menunjukkan betapa hebatnya peradaban kerajaan kuno di Indonesia.

Gambar 3.1. Gambar Candi Prambanan dan Candi Borobudur

Sumber: flickr.com

3.4. *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. Berikut gambaran umum *Machine Learning* dibandingkan dengan pemrograman secara tradisional.



Gambar 3.2. Perbandingan Pemrograman Tradisional Dengan *Machine Learning*

Dari gambar diatas 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:

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

Selain dari algoritma pembelajaran ada empat jenis cara pembelajaran pada *Machine Learning*, yakni:

- *Supervised Learning*: Data pembelajaran mencakup keluaran yang sudah ditentukan.
- *Unsupervised Learning*: Data pembelajaran tidak mencakup keluaran yang ditentukan.

- *Semi-supervised Learning*: Data pembelajaran mencakup beberapa keluaran yang ditentukan.
- *Reinforcement Learning*: Pemberian hadiah dari setiap serangkaian tindakan yang dilakukan.

Supervised Learning adalah cara pembelajaran yang banyak dipelajari dan memiliki banyak algoritma. Cara pembelajaran ini sangatlah efektif untuk permasalahan klasifikasi dan regresi.

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.*

3.5. *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*.

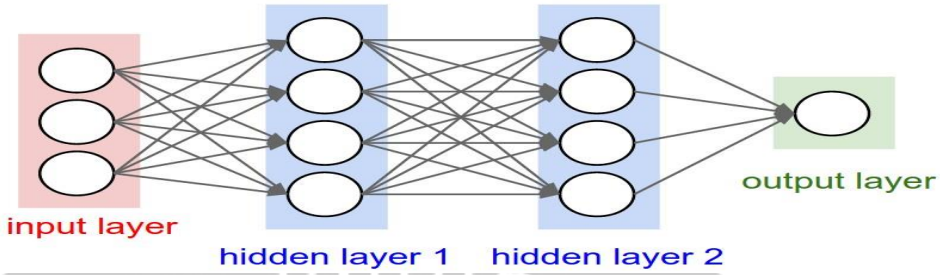
Feature Engineering adalah salah satu fitur utama dari *Deep Learning* untuk mengekstrak pola yang berguna dari data yang akan memudahkan model untuk membedakan kelas. *Feature Engineering* juga merupakan teknik yang paling penting untuk mencapai hasil yang baik pada tugas prediksi. Namun, sulit untuk dipelajari dan dikuasai karena kumpulan data dan jenis data yang berbeda memerlukan pendekatan teknik yang berbeda juga. Algoritma yang digunakan pada *Feature Engineering* dapat menemukan pola umum yang penting untuk membedakan antara kelas

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 kekompleksitasan sebuah model. Model yang kompleks tentunya akan membutuhkan waktu pelatihan yang lama sehingga di dunia *Deep Learning* pengunaan GPU sudah sangatlah umum.

.

3.6. Deep Feedforward Networks

Bisa dikatakan bahwa *Deep Feedforward Networks* merupakan ekstensi dari model standar jaringan syaraf tiruan yang terinspirasi dari cara bekerja neuron pada otak manusia. Mengapa disebut juga dengan *Feedforward* karena cara kerjanya yakni dimana sebuah input *x* mengalir maju dengan mediasi komputasi atau fungsi *f* untuk menghasilkan output *y*. Berikut adalah gambaran dasar arsitektur jaringan syaraf.



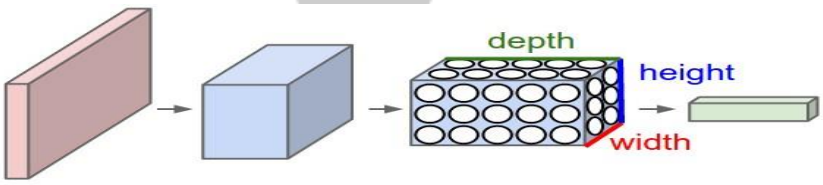
Gambar 3.3. Contoh Jaringan Deep Feedforward

Sumber: cs231n.github.io

Seperti yang sudah dibahas sebelumnya, lebih banyak hidden lapisan yang digunakan maka bisa dibilang jaringan tersebut menjadi lebih dalam. Arsitektur ini jugalah yang nantinya menjadi dasar dari *Convolution Neural Net*.

3.7. *Convolutional Neural Network*

Merupakan jaringan syaraf yang dikhususkan untuk memproses data yang memiliki struktur grid. Sebagai contoh dasarnya adalah berupa citra dua 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. Berikut gambaran umum arsitektur *Convolution Net*:



Gambar 3.4. Contoh Jaringan CNN

Sumber: cs231n.github.io

Dapat dilihat pada gambar diatas dimana setiap lapisannya input yang dimasukkan memiliki volume yang berbeda dan diwakili dengan kedalaman, tinggi dan lebar. 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.

3.8. Operasi Konvolusi

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 rumus berikut.

$$s(t) = (x * w)(t) \tag{3.1}$$

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 sebagai berikut

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.2}$$

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*.

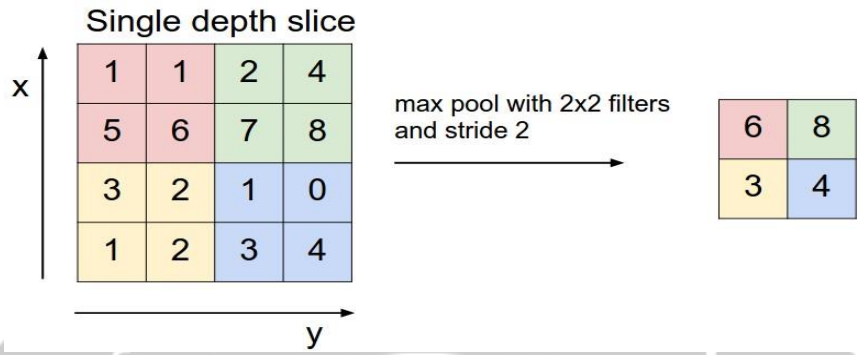
Kita juga dapat menentukan volume output dari masing-masing lapisan dengan *hyperparameters*. *Hyperparameter* yang digunakan pada persamaan dibawah ini digunakan untuk menghitung berapa banyak neuron aktivasi dalam sekali output.

$$(W - F + 2P)/S + 1 \tag{3.3}$$

Dari persamaan diatas kita dapat menghitung ukuran spasial dari volume output dimana *hyperparameter* yang dipakai adalah ukuran volume (**W**), filter (**F**), *Stride* yang diterapkan (**S**) dan jumlah padding nol yang digunakan (**P**). *Stride* adalah nilai yang digunakan untuk menggeser filter melalui input citra dan *Zero Padding* adalah nilai untuk menempatkan angka nol di sekitar border citra.

3.9. *Pooling Layer*

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*. Hal terpenting dalam pembuatan model CNN adalah dengan memilih banyak jenis lapisan *Pooling* dan hal ini bisa menguntungkan kinerja model (Lee, Gallagher, & Tu, 2015). 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 3.5. Contoh Operasi Max Pooling

Sumber: cs231n.github.io

Lapisan *Pooling* akan beroperasi pada setiap irisan kedalaman volume input secara bergantian. Pada gambar di atas, lapisan *pooling* menggunakan salah satu operasi maksimal yang merupakan operasi yang paling umum. Gambar 3.4. 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.

3.10. Aktivasi ReLU

Atktivasi 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.

3.11. *Fully-Connected Layer*

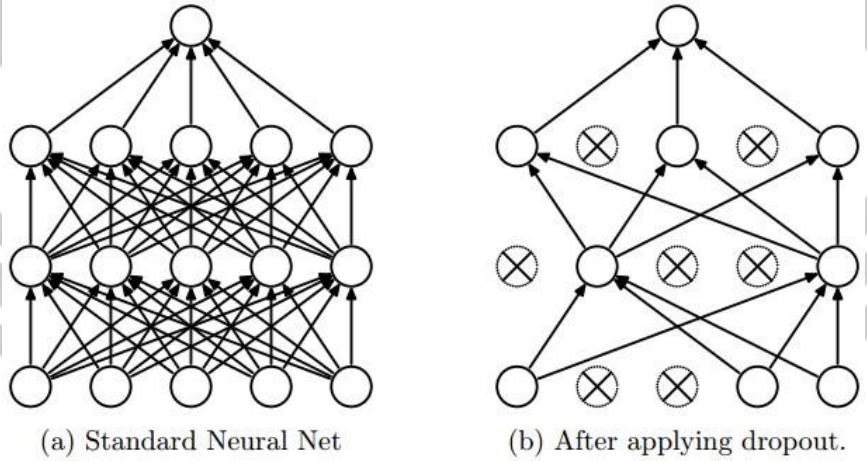
Lapisan *Fully-Connected* adalah lapisan di mana semua neuron aktivasi dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya seperti halnya jaringan saraf tiruan biasa. Setiap aktivasi dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua 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.

3.12. *Dropout Regularization*

Dropout adalah teknik regularisasi jaringan syaraf dimana beberapa neuron akan dipilih secara acak dan tidak dipakai selama pelatihan. Neuron-neuron ini dapat dibilang dibuang secara acak. Hal ini berarti bahwa kontribusi neuron yang dibuang akan diberhentikan sementara jaringan dan bobot baru juga tidak diterapkan pada neuron pada saat melakukan *backpropagation*.



Gambar 3.6. Jaringan Syaraf Sebelum dan Sesudah Mengaplikasikan Dropout

Sumber: cs231n.github.io

Pada gambar 3.5. jaringan syaraf (a) merupakan jarinagn syaraf biasa dengan 2 lapisan tersembunyi. Sedangkan pada bagian (b) jaringan syaraf sudah diaplikasikan teknik regularisasi *dropout* dimana ada beberapa neuron aktivasi yang tidak dipakai

lagi. Teknik ini sangat mudah diimplementasikan pada model CNN dan akan berdampak pada performa model dalam melatih serta mengurangi *overfitting* (Srivastava, Hinton, & Krizhevsky, 2014).

3.13. *Softmax Classifier*

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* bentuk persamaan yang muncul adalah sebagai berikut.

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{3.4}$$

Notasi fj 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.

3.14. *Crossentropy 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.

3.15. *Stochastic Gradient 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).

Stochastic Gradient Descent (SGD) merupakan metode *Gradient Descent* yang melakukan update parameter untuk setiap data pelatihan x(i) serta label y(i) dan memiliki persamaan dasar sebagai berikut.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{3.5}$$

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.

3.16. *Max Norm Constraint*

Merupakan salah satu bentuk lain dari regularisasi yang berfungsi untuk merubah bobot pada vektor untuk setiap neuron dan digunakan untuk optimisasi gradien. Saat pelatihan regularisasi ini melakukan update parameter pada optimasi seperti biasa, dan kemudian menerapkan suatu batasan yang dapat merubah vektor bobot. Penggunaan regularisasi ini dapat menunjukkan perbaikan hasil dari model yang sudah dilatih.

Regularisasi ini juga merupakan cara yang lebih baik untuk membatasi kompleksitas model dengan secara eksplisit merubah parameter setiap iterasi jika nilai parameter melebihi ambang batas kewajaran.