BAB II TEORI DASAR

2.1 Fasies Seismik

Fasies merupakan kumpulan lapisan yang perlihatkan perbedaan karakteristik litologi, sedimentologi dan geometri tertentu dengan batuan sekitarnya (Walker, 1973). Dalam ilmu geologi, kata fasies mengacu pada suatu cara untuk mengklasifikasikan suatu batuan berdasarkan sifat tertentu seperti komposisi, karakteristik fisik ataupun atribut lainnya yang memungkinkan untuk digunakan sebagai media pengklasifikasi. Selain itu, fasies digunakan untuk menjelaskan batuan sedimen yang dapat diamati dan diintepretasikan dengan lingkungan pengendapan (Miall, 1985).

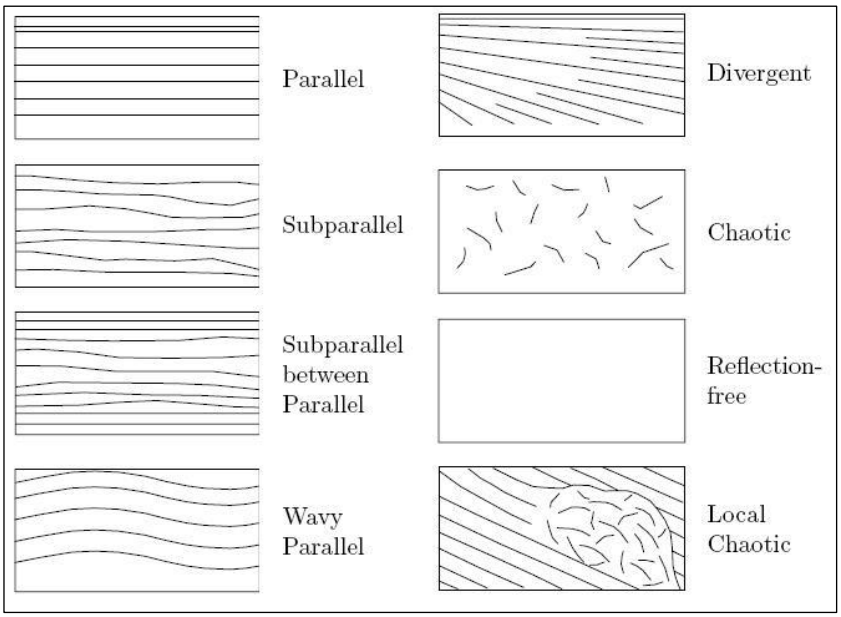
Gelombang seismik yang menjalar dan terefleksi kembali kepermukaan memberikan gambaran bentuk luaran dan tekstur dalam dari benda geologi yang ada dibawah permukaan. Untuk menganalisa bentuk dan tekstur geologi melalui rekaman seismik, perlu dilakukan analisis fasies seismik terlebih dahulu (Alfatih, 2017).

Analisa fasies seismik sangat membantu dalam menelusuri batas-batas sikuen. Sifat dari data seismik yang menampilkan kondisi bawah permukaan apa adanya sangat membantu dalam mengidentifikasi suatu fasies (Frisdio, 2017). Umumnya, gambaran data seismik perlihatkan struktur yang berkembang dan kondisi bawah permukaan sehingga dapat ditentukan jenis perangkap dan batas-batasnya. Jenis dari fasies seismik sangat berperan dalam penentuan lingkungan pengendapan (Sukmono, 1999).

Saat melakukan analisa data seismik, sering dijumpai berbagai macam bentuk reflektor seismik akibat dari kondisi geologi tertentu. Berbagai macam bentuk reflektor ini dapat menggambarkan karakteristik pengendapan suatu batuan. Satu set fasies seismik merupakan satu set seismik dalam 3D yang tersusun berdasarkan kumpulan reflektor yang memiliki parameter berbeda dengan set

fasies disekitarnya (Mitchum, 1977). Dalam intepretasi fasies seismik, pola dari reflektor seismik dibedakan menjadi beberapa macam, yaitu :

1. *Parallel* : pola pengendapan sedimen yang cenderung seragam, biasanya terdapat di *shelf* dengan s*ubsiden* seragam pada suatu *basin plain* yang cukup stabil.
2. *Subparallel* : pola pengendapan akibat situasi yang tidak tenang akibat terganggu oleh arus laut, sistem pengendapan cenderung cepat dan terdapat zona pengisian.
3. *Subparallel between parallel* : pola pengendapan yang mungkin terbentuk pada lokasi *fluvial plain*, memiliki butir endapan sedang dengan kondisi tektonik yang cukup stabil.
4. *Wavy parallel* : pola pengendapan yang terbentuk dari lipatan kompresi pada lapisan *parallel* dibagian atas *diapir* / *sheet drape* berendapan butir halus.
5. *Divergent* : pola pengendapan yang terjadi diakibatkan oleh proses sedimentasi yang permukaannya membentuk miring progresif.
6. *Chaotic* : pola pengendapan akibat energi yang bekerja cukup tinggi atau terdapat deformasi setelah terjadi proses sedimentasi.
7. *Reflector free* : pola pengendapan yang disebabkan oleh adanya *interior reef* tunggal, kubah garam ataupun batuan beku.
8. *Local chaotic* : pola pengendapan yang dihasilkan oleh gempa bumi atau gravitasi yang tidak stabil di laut dalam, pengendapan terjadi cenderung cepat.



Gambar 2.1. Pola reflektor seismik (Mitchum, 1977).

Dalam menentukan fasies seismik, terdapat parameter - parameter yang perlu dianalisis untuk mendapatkan hasil identifikasi yang lebih akurat. Beberapa atribut seismik secara tak langsung menggambarkan kondisi lingkungan pengendapan suatu formasi batuan. Atribut - atribut seismik yang dimaksud antara lain yaitu :

1. Amplitudo

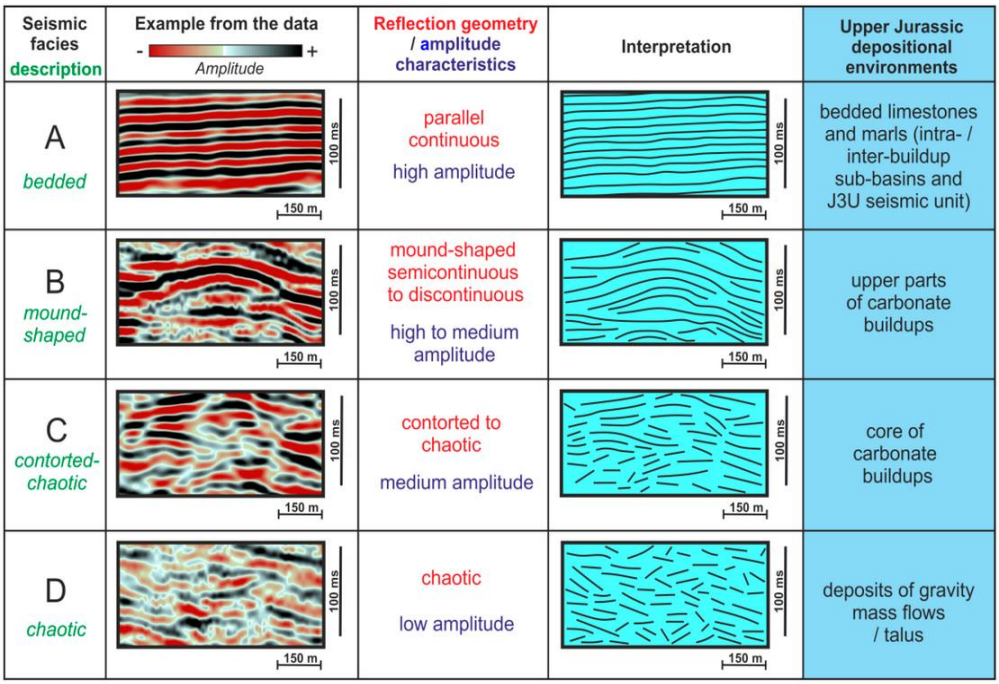
   Atribut ini sangat berhubungan dengan perbedaan sifat antar dua batuan yang didasari oleh perbedaan *impedance*. Nilai besar amplitudo memberikan informasi mengenai besar energi pembentukan saat proses pengendapan. Amplitudo yang tinggi dengan kontinuitas rendah biasa dikaitkan dengan energi rendah yang mengendap pada perairan tenang. Amplitudo yang tinggi dengan kontinuitas tinggi biasa dikaitkan dengan pengendapan yang mengalami erosi, sedangkan amplitudo rendah biasa dikaitkan dengan *deep water shales*.

2. Frekuensi

   Atribut ini umumnya dikaitkan dengan kehadiran kontak fluida dan *bed thickness*.

3. *Interval Velocity*

   Atribut ini umum digunakan sebagai data pendukung untuk intepretasi litologi.



Gambar 2.2. Diagram variasi atribut dalam menentukan fasies seismik (Anonim, 2014).

2.2 *Machine Learning*

Machine learning merupakan kemampuan dari komputer untuk dapat melakukan pembelajaran tanpa menjelaskan secara eksplisit ke komputer (Samuel, 1959). Dengan kata lain, *machine learning* memberikan kemampuan kepada komputer untuk melakukan aktivitas belajar menyelesaikan masalah secara mandiri. *Machine learning* melakukan interferensi terhadap data dengan membuat model matematis dan menghasilkan pola - pola data dengan memanfaatkan statistika dan aljabar linear (Putra J. G., 2020).

Dalam *machine learning*, terdapat tiga skenario pembelajaran didalamnya, yakni *unsupervised learning*, *supervised learning* serta *reinforcement learning*.

1. *Unsupervised learning* sendiri merupakan algoritma yang belajar berdasarkan masukan data pembelajaran yang tak berlabel dan setelahnya

mencoba membagi kelompok data berdasarkan kesamaan ciri yang ditemukan (Nurhikmat, 2018).

2. *Supervised learning* berupa algoritma yang mencoba belajar berdasarkan dari kumpulan contoh pasang *input* dan *output* yang diberikan. Algoritma ini mencoba mengikuti contoh yang diberikan dan menghasilkan prediksi yang dapat menyelesaikan masalah *input* baru menjadi *output* yang cukup tepat (Russel, 1995).
3. *Reinforcement learning* merupakan algoritma yang belajar dari masukan yang dicampur (berlabel dan tak berlabel) untuk mengumpulkan informasi pembelajaran secara aktif dengan interaksi untuk tiap aksi dari pembelajaran.

Masa kini, telah banyak dilakukan penelitian dengan konsep pendekatan *machine learning*. Salah satunya dibidang data *mining*, *computer vision* dan *robotic* (Chan, 2018). Dalam melakukan proses deteksi serta pengenalan untuk mendukung pengolahan data yang besar dan komputasi tingkat tinggi, dapat dilakukan pengolahan dengan *machine learning* teknik *deep learning*.

2.2.1 *Deep Learning*

Deep learning diimplementasikan pada *problem dataset* yang cukup besar. *Deep learning* adalah metode *machine learning* yang memiliki banyak *layer* dan dimanfaatkan untuk pengolahan dalam melakukan klasifikasi, pengenalan pola, ekstraksi fitur (Deng, 2014). Singkatnya, *deep learning* merupakan bagian *machine learning* dengan pengembangan *layer* yang lebih kompleks, terutama pada *hidden layer* (Suryahadinata, 2020).

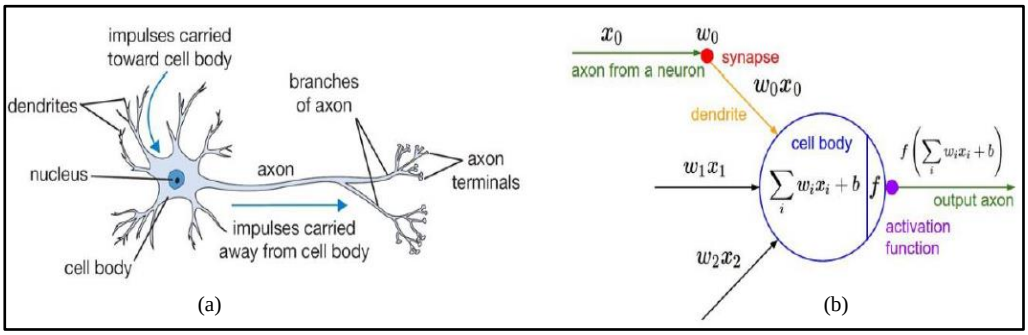
Deep learning menyajikan struktur seperti *supervised learning*. Dengan menambah banyak *layer*, model *deep learning* yang dibuat dapat mewakili data *image* berlabel secara baik (Rena, 2019). Beberapa tahun terakhir, teknik *deep learning* sering dipakai dalam pembuatan model penelitian dan terbukti mendapatkan gambaran model yang efisien. Keunggulan teknik *deep learning* dibandingkan dengan *machine learning* sebelumnya yaitu adanya

ekstraksi fitur otomatis dengan mengutamakan *labelling* pada *dataset* dalam proses di jaringan (Nurhikmat, 2018).

Dalam *deep learning* terdapat metode *Convolutional Neural Network* (CNN) yang mana metode ini efisien dalam menemukan fitur yang baik pada *image*, membuat hipotesis *nonlinear* dan meningkatkan kekompleksitasan model yang dibangun. Teknik ini berproses komputasi kuat dan proses pelatihan modelnya lebih dalam (Rena, 2019). Beberapa contoh dari penerapan d*eep learning* dalam memecahkan masalah yang lebih kompleks yaitu pengenalan sidik jari, *computer vision*, *speech recognition* dan dalam ilmu geofisika adalah pengklasifikasi fasies (Suryahadinata, 2020).

2.2.2 *Neural Network*

Neural Network merupakan gambaran duplikasi dari otak manusia yang mempraktekan proses kerja pada otak manusianya (Kusumadewi, 2004). Metode ini terinspirasi dari sel - sel neuron pada jaringan manusia yang bekerja secara terintegrasi oleh jutaan hingga milyaran neuron pada otak manusia. Neuron – neuron tersebut saling mengirimkan impuls saraf hingga tiap neuron akan saling terhubung satu sama lain.

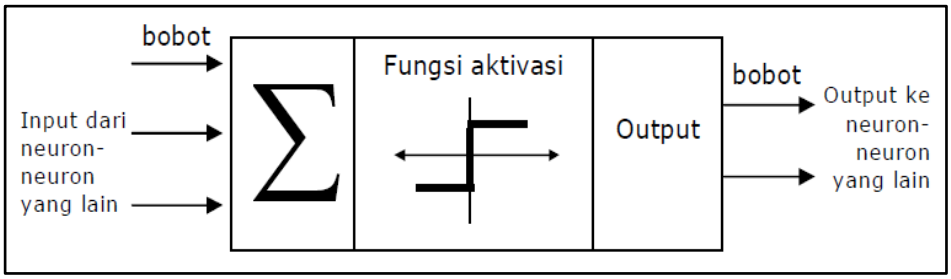


Gambar 2.3. Sistem neuron pada otak manusia (a) dan bentuknya secara matematika (b) (Sena S. , 2017).

Sistem integrasi neuron pada otak manusia diimplementasikan untuk membuat *Artificial Intellegent* (AI). *Neural network* didesain bukan untuk menghasilkan *output* sesuai dengan keinginan langsung, namun *output* yang dihasilkan berupa hasil yang salah kemudian mendekati hasil yang diharapkan sehingga akhirnya akan memperoleh hasil yang maksimal

(Suryahadinata, 2020). Perubahan *output* tersebut didasari oleh pengalaman yang diberikan dalam bentuk data yang biasa disebut data *training*.

Neural network memiliki bentuk yang berbeda-beda. Namun semua komponen pada *neural network* yang dimiliki cenderung sama. Layaknya jaringan syaraf otak manusia, *neural network*pun terdiri dari unit - unit neuron yang saling terhubung yang direpresentasikan pada gambar 2.4 (Nurhikmat, 2018). Secara umum, *neural network* terdiri atas tiga bagian utama, yakni *input* sebagai memasukkan data *training*, lalu data yang masuk kedalam *input layer* akan diberi pembobotan yang kemudian datanya dimasukkan kedalam *hidden layer*. Pada *hidden layer* umumnya data *input* yang telah diberi bobot akan dilakukan aktivasi dengan persamaan tertentu. Aktivasi diberlalukan agar dapat dibandingkan dengan ambang nilai tertentu. Setelah data *input* diberi bobot dan diaktivasi, lalu akan menghasilkan *output* data baru (Suryahadinata, 2020).



Gambar 2.4. Satu unit neuron pada jaringan saraf tiruan (Suryahadinata, 2020).

2.2.3 *Training, Validation* dan *Testing set*

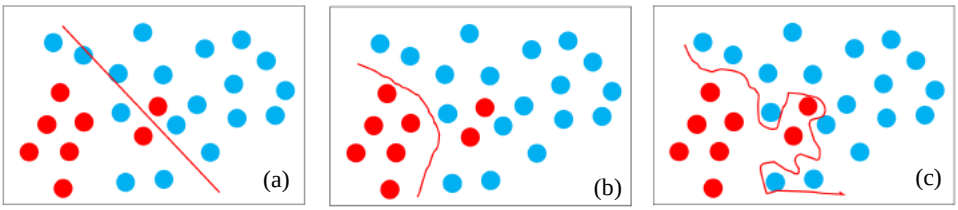
Dalam membangun dan mengevaluasi suatu model, dibutuhkan kumpulan data (sampel dan statistik) yang biasa disebut dengan *dataset.* Dalam *machine learning*, *dataset* dibagi menjadi tiga jenis, yaitu :

1. *Training set,* yakni kumpulan data yang dipakai dalam melatih dan membangun model;
2. *Validation set,* yakni kumpulan data yang digunakan untuk mengoptimalisasi model; dan
3. *Testing set,* yakni kumpulan data yang dipakai untuk menguji model.

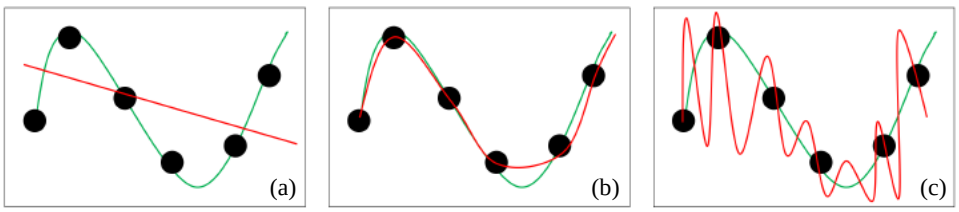
Umumnya, rasio pembagian *dataset* adalah 80% *training set*, 10% *validation set* dan 10% *testing set* (Putra J. G., 2020). Namun dalam beberapa kasus tertentu tidak adanya *validation set*. Untuk mengatasi ketidaktersediaan *validation set* maka perlu dilakukan evaluasi model dengan metode *k-cross validation.*

2.2.4 *Overfitting* dan *Underfitting*

Overfitting merupakan keadaan dimana ketika model yang dibuat mendapatkan nilai baik pada kinerjanya untuk *training set* namun buruk untuk *unseen dataset* (data yang tidak terlihat) yang terjadi akibat terlalu fleksibelnya model yang dihasilkan dengan kemampuan yang cukup tinggi dalam mengestimasi banyak fungsi dan terlalu cocok terhadap *training set* sehingga hasil prediksi seolah sempurna. Sedangkan *underfitting* merupakan keadaan dimana ketika kinerja model bernilai buruk pada keseluruhan data set akibat model yang dihasilkan sangat tidak fleksibel (Putra J. G., 2020). Pada gambar 2.5 diperlihatkan gambaran kasus *overfitting* (gambar 2.5.a), kasus *underfitting* (2.5.b) dan persebaran klasifikasi yang sesuai (gambar 2.5.c). Sedangkan pada gambar 2.6 diperlihatkan gambaran kasus *overfitting* (gambar 2.6.a), kasus *underfitting* (2.5.b) dan regresi yang sesuai (gambar 2.6.c).



Gambar 2.5. *Overfitting* (a) vs *Underfitting* (b) dan klasifikasi yang sesuai (c).

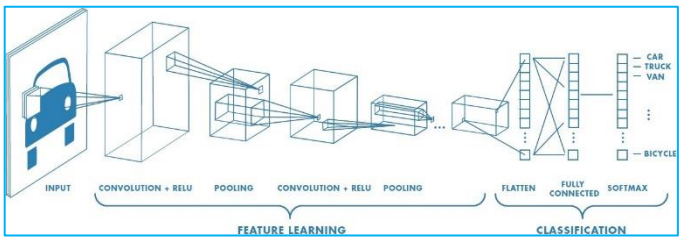


Gambar 2.6. *Overfitting* (a) vs *Underfitting* (b) pada regresi yang sesuai (c)

2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan ubahan dari *multilayer perception* (MLP) yang dibangun guna mengolah data dalam bentuk bentuk *image* (Nurfita, 2018). CNN sendiri masuk kedalam *Deep Neural Network* (DNN) dikarenakan memiliki jaringan yang dalam serta berlapis - lapis. CNN hampir sama dengan MLP dalam sudut cara kerjanya, namun dalam CNN tiap neuron ditampilkan dalam bentuk dua dimensi sedangkan MLP sendiri *neuron* hanya ditampilkan dalam satu dimensi (Putra I. W., 2020). Data pada CNN memiliki tampilan jaringan data dua dimensi dengan operasi linear menggunakan operasi konvolusi. Penerapan CNN dalam klasifikasi data dan *image processing* mulai dikenal sejak berlangsungnya perlombaan *ImageNet Large-Scale Visual Recognition Challange* tahun 2010. Lalu dikenal pula arsitektur - arsitektur CNN seperti UNet, Alexnet, GoogLeNet dan ResNet.

CNN memiliki arsitektur yang bekerja dengan cara mempelajari dan tersusun atas beberapa langkah. Pada masukan dan keluaran tiap langkah terdapat *array - array* (*feature map*). Tiap langkah memiliki lapisan konvolusi, lapisan fungsi aktivasi dan lapisan *pooling*. Konvolusi dilakukan menggunakan jaringan kernel berukuran tertentu dengan banyaknya kernel terpakai bergantung banyaknya fitur yang akan dihasilkan. Fungsi aktivasi pada proses konvolusi biasanya menggunakan fungsi ReLU (*Rectifier Linear Unit*) dengan *output* dari proses fungsi aktivasi ini kemudian masuk ke proses *pooling*. Proses *pooling* ini dilakukan dengan melakukan perulangan hingga mendapatkan gambaran fitur terbaik kelas *output* (Nurhikmat, 2018). Adapun gambaran arsitektur CNN dipresentasikan pada gambar 2.7.



Gambar 2.7. Arsitektur *Convolutional Neural Network* (Nurhikmat, 2018).

2.3.1 *Convolutional Layer*

Convolutional layer merupakan tahapan dalam arsitektur CNN. Didalamnya dilakukan operasi konvolusi pada hasil keluaran dari lapisan sebelumnya. Konvolusi merupakan kegiatan dimana dilakukan penerapan dari sebuah fungsi kepada hasil dari fungsi lainnya yang dilakukan berulang kali (Nurhikmat, 2018). Operasi konvolusi memberikan fungsi keluaran sebagai *feature map* dari masukan berupa citra. *Input* dan *output* ini diasumsikan bernilai riil. Persamaaan konvolusi yaitu :

$$s(t) = x(t) * t = \sum_{\alpha=-\infty}^{\infty} x(\alpha) * w\,(t-a) \quad (3.1)$$

Dengan *s(t)* = fungsi konvolusi

X = masukan

W = bobot

Fungsi konvolusi menghasilkan keluaran tunggal (*feature map*) dengan argumen awal yakni masukan (x) dan argumen kedua berupa filter atau *kernel* (w). Apabila masukan berupa citra dalam bentuk dua dimensi, dapat diasumsikan t merupakan *pixel* serta menggantikannya dengan i dan j. Persamaan konvolusi ke masukan menjadi :

$$s\,(i,j) = (K * I)(ij) = \sum_{\infty}\sum_{\infty} I(i-m, j-n)K(m,n) \quad (3.2)$$

$$s\,(i,j) = (K * I)(ij) = \sum_{\infty}\sum_{\infty} I(i+m, j+n)K(m,n) \quad (3.3)$$

Persamaan tersebut komulatif dan muncul dengan K bernilai *kernel*, lalu *I* merupakan masukan dengan *kernel* yang bisa berubah tempat menyesuaikan masukan. Kondisi lain operasi konvolusi yakni sebagai perkalian matriks antar *input* berupa citra dengan *kernel*nya dimana hasil keluaran didapat dengan melakukan operasi *dot product*. Volume keluaran dihasilkan dari semua lapisan ber*hyperparameters*. *Hyperparameter* dipakai untuk mencari jumlah neuron aktivasi selama sekali proses. Adapun persamaan dari *hyperparameter* yakni:

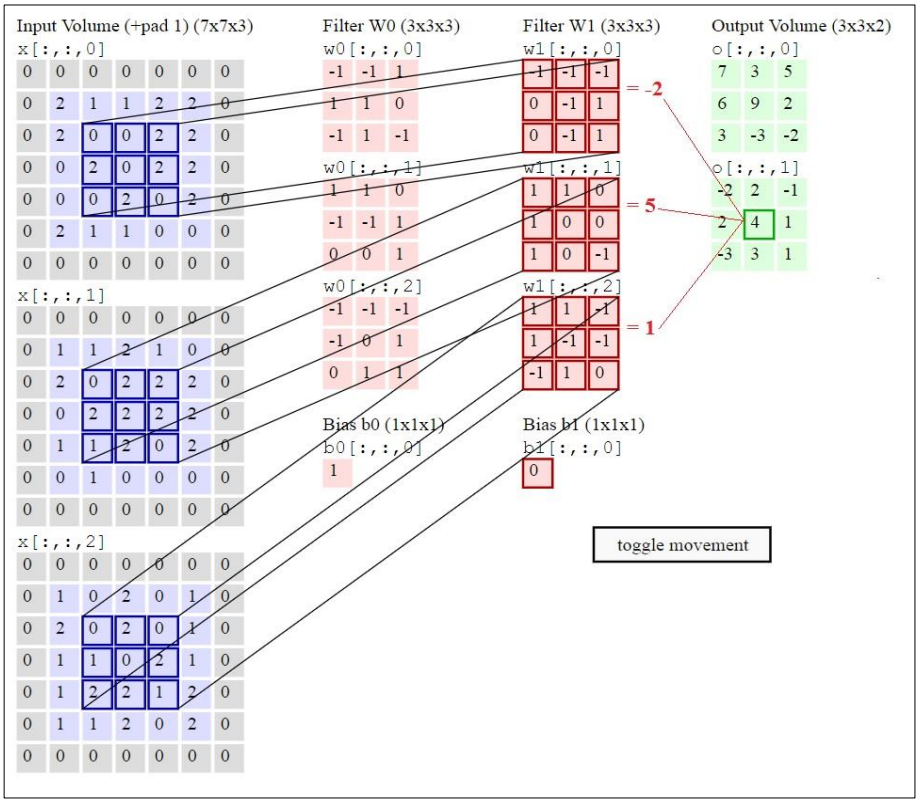
$$(W - F + 2P)/(S + 1) \quad (3.4)$$

Dengan *W* = Ukuran volume gambar

F = Ukuran filter

P = Nilai *padding* yang digunakan

S = Ukuran pergeseran (*stride*)



Gambar 2.8. *Convolutional layer* (Sena S. , 2017).

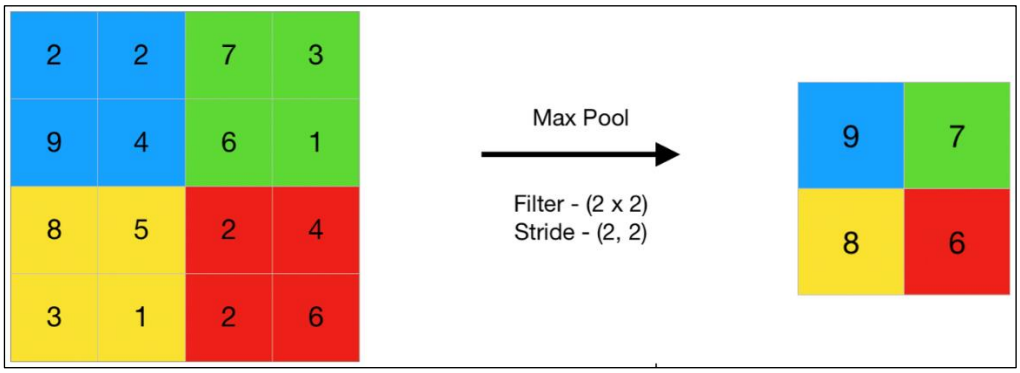
Terlihat pada gambar 2.8, *Convolutional layer* disusun dari beberapa neuron sedemikian rupa sehingga membuat bentuk filter berukuran tinggi dan panjang (*pixels*). Filter ini kemudian digeser kesemua bagian citra dengan nilai tertentu (*stride*). Tiap bergeser, terjadi operasi "*dot*" antara masukan dengan nilai filter dan mendapatkan hasil berupa *activation map* atau *feature map*. Ukuran spasial dari volume *output* dapat dihitung dari nilai *hyperparameter padding* yang digunakan bernilai nol. Nilai nol *padding* ini akan menghasilkan angka nol disekitar *border* citra.

2.3.2 *Pooling Layer*

Pooling adalah kegiatan mengurangi ukuran matriks dengan operasi *pooling*. Pada dasarnya, *pooling layer* terbentuk dari sebuah filter berukuran dan langkah yang telah ditentukan serta berganti – ganti tergeser pada seluruh area *feature map*. *Pooling layer* yang masuk pada *layer* konvolusi secara

berkesinambungan pada arsitektur CNN mampu menghasilkan ukuran volume *output* pada *feature map* yang berkurang hingga menjadikan jumlah parameter dan hitungan jaringan berkurang dalam mengendalikan *overfitting*. Hasil proses *pooling* adalah matriks berdimensi lebih kecil dari *image* awalnya.

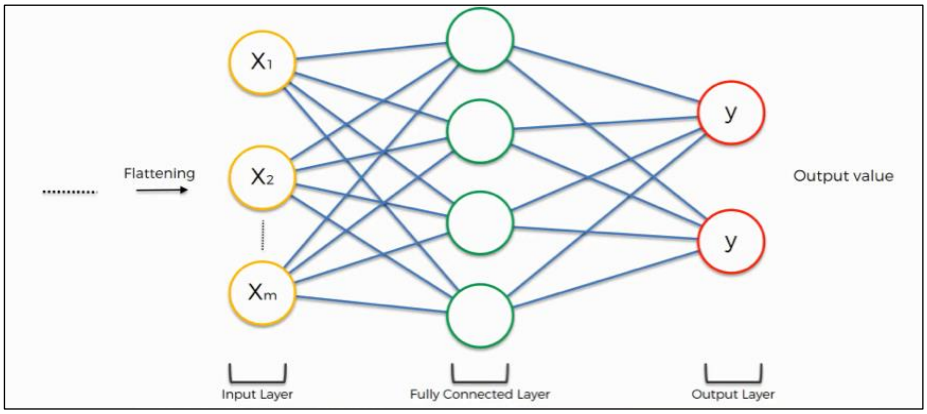
Terdapat dua jenis *pooling* yang sering digunakan dalam *pooling layer*, yaitu *maximal pooling* dan *average pooling*. Pada *average pooling* yang digunakan merupakan nilai rata-rata, sedangkan yang digunakan *maximal pooling* berupa nilai tertinggi. Pada gambar 2.9, terlihat proses kerja dari *maximal pooling* yang mana dari suatu stride diambil satu nilai tertinggi untuk dijadikan *output*-nya.



Gambar 2.9. *Max pooling layer* (savyakhosla, 2019).

2.3.3 *Fully-connected layer*

Fully-connected layer adalah lapisan yang menghubungi semua neuron aktivasi dari lapisan sebelumnya dengan semua neuron dilapisan selanjutnya. Perbedaan lapisan ini dengan *layer* konvolusi yaitu pada lapisan konvolusi neuron hanya terhubung ke suatu daerah pada *inputnya*, sedangkan neuron pada *fully-connected layer* akan terhubung menyeluruh. Representasi dari *fully-connected layer* diperlihatkan pada gambar 2.10.



Gambar 2.10. *Fully-connected Layer* (Team, 2018).

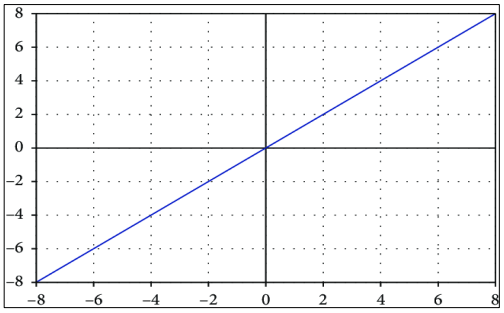
Setelah melalui lapisan – lapisan sebelumnya, citra masih berupa *multidimensional array*. Agar dapat digunakan sebagai *input*, maka perlu dilakukan proses *flatten* dari *multidimensional array* menjadi bentuk vektor. Gambaran dari *fully-connected layer* memiliki bentuk yang sama dengan neuron network biasa. Semua *input* pada *fully-connected layer* berupa tiap *pixels* hasil *convolutional layer* dan *pooling layer* yang telah alami proses *flatten* kedalam bentuk vektor (Suryahadinata, 2020).

2.3.4 Fungsi Aktivasi

Fungsi aktivasi adalah nilai yang mempresentasikan tingkat aktivitas internal dalam jaringan (Mubarok, 2019). Fungsi aktivasi digunakan dalam menentukan apakah sebuah neuron harus aktif atau tidak yang didasari oleh nilai penjumlahan setelah diberi pembobotan (Suryahadinata, 2020). Berdasarkan artikel yang ditulis oleh Samuel Sena (Sena S. , 2017), terdapat fungsi aktivasi yang biasa dipakai pada jaringan *neural network,* diantaranya yaitu:

1. Aktivasi linear

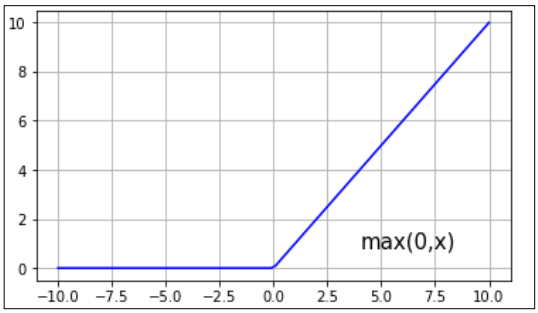
   Fungsi aktivasi linear atau yang biasa disebut dengan aktivasi standar (*default*) merupakan operasi yang menggunakan *linear function* dan memiliki nilai *output* dari neuron berupa nilai *input* + bias.



Gambar 2.11. Fungsi aktivasi linear.

2. Fungsi aktivasi ReLu (*Rectified Linear Unit*)

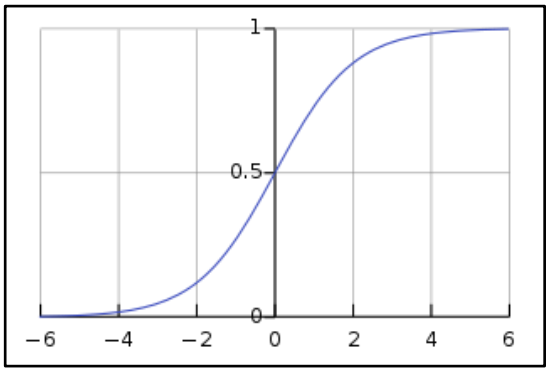
   Fungsi aktivasi ReLu adalah operasi yang dapat mengubah nilai negatif menjadi ambang nol dan nilai positif berambang nilai itu sendiri.



Gambar 2.12. Fungsi aktivasi ReLu.

3. Fungsi aktivasi *sigmoid*

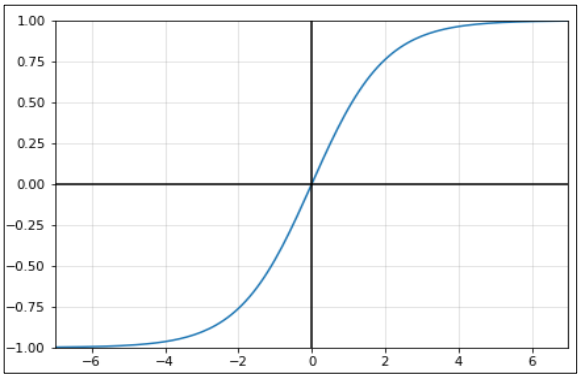
   Fungsi aktivasi *sigmoid* adalah operasi yang merubah nilai masukan menjadi ambang 0 hingga 2. *Output* yang dihasilkan dari fungsi *sigmoid* kurang lebih sama apabila *input* dinormalisasi terlebih dahulu.



Gambar 2.13. Fungsi aktivasi *sigmoid.*

4. Fungsi aktivasi Tanh (*Tangent hyperbolic*)

   Fungsi aktivasi tanh adalah operasi yang nilainya terpusat pada nilai nol dengan mengubah nilai *input* menjadi nilai ambang dari -1 hingga +1.



Gambar 2.14. Fungsi aktivasi Tanh.

5. Fungsi aktivasi *Softmax*

   Fungsi *softmax* adalah operasi yang merubah K dimensi vektor "x" berupa nilai riil ke bentuk sama akan tetapi bernilai rentang 0 hingga 1. Fungsi *softmax* dipakai dalam klasifikasi jumlah kelas yang cukup banyak, seperti *Naive Bayes Classification*, regresi logistik *multinomial* dan *Artificial Neural Network* (ANN). Fungsi ini berguna dalam *layer* yang memiliki *neural network* dan biasa terdapat pada lapisan akhir untuk menghasilkan *output*. Persamaan fungsi *softmax* adalah sebagai berikut :

$$fj(z) = \frac{e^z j}{\sum k e^z k} \quad (3.5)$$

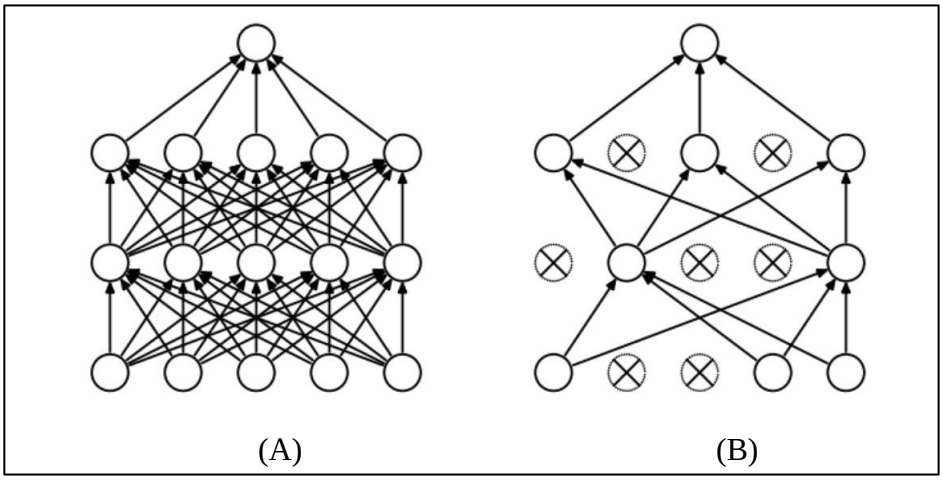
   Nilai *fj* merupakan hasil fungsi tiap elemen ke-j vektor *output* kelas. Nilai *z* merupakan dugaan yang dibuat oleh model *training* agar mampu diklasifikasikan fungsinya. Hasil ini menampilkan hasil yang lebih intuitif juga menjadikan intepretasi probabilistik yang lebih baik dibanding fungsi klasifikasi lainnya. *Softmax* dapat memperhitungan nilai probabilitas di semua label. Label tersebut kemudian dijadikan sebuah vektor nilai yang memiliki nilai riil serta merubahnya dengan nilai berentang 0 dan 1 yang mana apabila dijumlah akan menghasilkan nilai 1.

Softmax layer dapat menentukan probabilitas terbesar untuk hasil klasifikasinya. Neuron *softmax layer* tidak menerapkan fungsi aktivasi namun menerapkan fungsi operasi *softmax*. Sama hal dengan neuron pada umumnya, neuron pada fungsi *softmax* menerima masukan kemudian dilakukan penambahan bias dan pembobotan (Rena, 2019).

2.3.5 *Dropout Regulation*

Dropout adalah cara meregulasi jaringan syaraf yang bertujuan memilah neuron secara *random* untuk dibuang dan tidak digunakan selama proses *training* berlangsung. Kontribusi neuron yang dibuang pada tahap ini akan di berhentikan sementara oleh bobot dan jaringan baru tak diimplementasikan pada neuron saat kegiatan *back propragation* (Lina, 2019).

Proses *dropout* mempercepat proses *learning* dan mengurangi terjadinya *overfitting*. *Droupout* akan mencoba men*drop* neuron - neuron berupa *hidden* maupun lapisan yang *visible* dalam suatu jaringan. Suatu neuron yang dihilangkan membuat neuron tersebut sementara hilang dari jaringan yang ada. Neuron yang akan di*drop* dapat dipilih secara *random*. Probabilitas dari tiap neuron diberi nilai antara 0 dan 1 (Lina, 2019).



Gambar 2.15. Jaringan syaraf biasa (a) dan jaringan yang telah diaplikasikan regulasi *dropout* (b) (Lina, 2019).

Mengacu gambar 2.10, jaringan syaraf (a) merupakan jaringan syaraf biasa yang memiliki dua lapisan tersembunyi. Sedangkan pada bagian (b) telah diaplikasikan teknik regulasi *dropout* yang mana terdapat beberapa neuron aktivasi yang tidak digunakan lagi. Teknik ini mudah diimplementasikan pada model *convolutional neural network* dan berdampak pada performa model dalam melatih serta mengurangi *overfitting* (Srivastava, 2014).

2.3.6 *Cross Entropy Loss Function*

Loss function adalah operasi yang perlihatkan gambaran kerugian dugaan hasil dari model yang dibuat. *Loss function* akan berjalan ketika model pembelajaran yang dibuat memberikan *missing* sehingga perlu menjadi perhatian. *Loss function* yang efektif adalah operasi yang akan menghasilkan nilai *error* yang diharapkan paling rendah pada model yang dibuat.

Ketika suatu model mempunyai kelas yang banyak, diperlukan sebuah cara untuk menaksir perbedaan antar probabilitas sebenarnya dengan probabilitas hasil dugaan. Selama *training*, banyaknya algoritma yang mampu melakukan penyesuaian parameter dan nilai perbedaan probabilitas ini mampu diminimalisir. *Crossentropy* merupakan cara terbaik dalam melakukan perbandingan probabilitas. Gambaran dari algoritmanya yakni meminimalisir kemungkinan log negatif pada *dataset* berukuran performa prediksi model. Adapun fungsi dari *crossentropy* adalah sebagai berikut :

$$\frac{1}{N}\sum_x(target(x) - aktivasi(x))^2 = \frac{1}{N}\sum_x(target(x) - \max(0, \sum_i^{|x|} w_i x_i + b))^2 \quad (3.6)$$

2.3.7 *Forward Propagation*

Forward propagation pada jaringan *convolutional neural network* digunakan untuk melanjutkan nilai lapisan *input* hingga nilai lapisan *output*. Nilai ini dilanjutkan melalui *layer* konvolusi, *subsampling* dan *fully-connected* di sesuaikan dengan runtutan lapisan diplot pada jaringan yang dipakai.

Arsitektur CNN yang akan dipakai perlu dirancang dahulu. Urutan dari proses *forward propagation* pada CNN dapat disederhanakan sebagai berikut:

1. Inisiasi nilai pertama pada lapisan konvolusi serta lapisan *fully-connected* dengan nilai *random* dan nilai bias pertama bernilai 0.
2. Lakukan proses konvolusi *image input* sesuai filter konvolusi hingga mendapatkan hasil ke p (C_p^1) dari filter (k_{1p}^1) dan bias (b_p^1) yang dioperasikan pada *image input*. kode * mengartikan konvolusi dan σ(x) mengartikan fungsi aktivasi.

   $$C_P^1 = \sigma\,(I * K_{1p}^1 + b_p^1) \quad (3.7)$$

3. Hasil yang didapat kemudian dilakukan pengurangan ukuran untuk mengecilkan kompleksitas hitungan pada lapisan berikutnya. Proses berlaku dilapisan *subsampling* dengan operasi *max pooling*. Hasil dari lapisan *subsamping* merupakan nilai baru yang sudah dilakukan reduksi ukuran.
4. Nilai pada lapisan *subsampling* akhir menjadi *feature maps* yang akan dipakai pada *fully-connected layer* sebagai fitur dalam kegiatan klasifikasi. Fitur masukan *fully-connected layer* merupakan hasil vektorisasi (*F(x)*) dari hasil *subsampling layer* sebelumnya (S_p^1). Prosesnya menggabungkan seluruh n sebuah *feature maps*.

   $$f = F\,(\{S_p^1\} p = 1,2,3,..n) \quad (3.8)$$

5. Lakukan kegiatan memperhitungksn dugaan target dari fitur *input* yang menuju *fully-connected*. Prediksi kelas ($y(i)$) dapat dihitung dengan persamaan berikut :

   $$\hat{y}(i) = \sigma\,(\sum_{j=1}^{n} W\,(i,j) f(j) + b(i)) \quad (3.9)$$

6. Untuk mengetahui seberapa baik proses pembelajaran, maka hitung nilai loss dengan persamaan 3.2.

2.3.8 *Back Propagation*

Back propagation merupakan kegiatan memperbaharui nilai bobot dan filter pada jaringan. Nilai terbaru bobot dihitung mulai dari *fully-connecter layer*. Pada lapisan ini, nilai terbaru bobot dihitung dengan menghitung nilai

derivatif dari *loss function* dengan bobotnya (Zhang, 2016). Perhitungan perubahan ($\Delta W(i,j)$) yang terhubung dengan *node* penghasil citra f(j) berdasarkan sisa nilai dugaan kelas dari data ke *i* ($\hat{y}(i)$) dengan target aktual data *ke i* ($y(i)$) *fully-connected layer* dapat ditulis dalam persamaan sebagai berikut :

$$\Delta W(i,j) = \begin{cases} \hat{y}(i) - y(1).f(j), & jika\ \hat{y}(I) < 0 \\ 0, & lainnya \end{cases} \quad (3.10)$$

Perubahan bias dapat dijalankan dengan mencari nilai bias terhadap derivatif *loss function*. Untuk menghitung nilai bias digunakan persamaan berikut :

$$\Delta b(i) = \begin{cases} \hat{y}(i) - y(1), & jika\ \hat{y}(i) < 0 \\ 0, & lainnya \end{cases} \quad (3.11)$$

Perubahan nilai filter pada lapusan konvolusi didasari oleh *error* pada lapisan *subsampling*. Untuk melakukan perhitungan perubahan nilai bobot di lapisan konvolusi, perlu dijalankan *unsampling* dari *error* sebab sesudah dilakukan konvolusi *feature maps* melalui *subsampling* serta proses vektorisasi. Perubahan nilai *feature maps* dapat dihitung dengan persamaan berikut :

$$\Delta f = \begin{cases} \hat{y}(i) - y(1).W(i,j), & jika\ \hat{y}(i) < 0 \\ 0, & lainnya \end{cases} \quad (3.12)$$

Nilai perubahan *feature maps* yang masih berbentuk vektor kemudian dilakukan pembalikan vektor kedalam bentuk matriks. Perubahan ini dapat dihitung dengan persamaan :

$$\{S_p^1\}_{p=1,2,3,\ldots n} = f^{-1}(\Delta f) \quad (3.13)$$

Proses *unsumpling* merupakan proses merubah matriks $\{\Delta S_p^1\}$ hasil *subsampling* ke ukuran semula sebelum dilakukan proses *subsampling*. Hal ini dilakukan untuk meneruskan nilai matriks pada koordinat *feature maps* yang tidak diloloskan nilainya pada proses *subsampling*. Penerusan nilai ini dinotasikan oleh persamaan berikut :

$$\Delta b(i) = \begin{cases} \Delta S_p^1\left(\frac{i}{2}, \frac{j}{2}\right), & jika\ C_p^1(i,j)\ terkontribusi < 0 \\ 0, & lainnya \end{cases} \quad (3.14)$$

Setelah proses *unsampling*, nilai $\Delta C_p^1(i,j)$ dapat dipakai untuk perhitungan nilai filter konvolusi yang berubah tidak sama seperti di lapisan sebelumnya. Perubahan nilai filter dapat dikerjakan dengan menggunakan konvolusi fitur *input* menggunakan $\Delta C_{1,\sigma}^1$. Proses pencarian nilai filter konvolusi yang berubah dicari dengan menggunakan persamaan :

$$\Delta C_{1,\sigma}^1(i,j) = \begin{cases} \Delta C_p^1(i,j), & jika\ C_p^1(i,j) > 0 \\ 0, & lainnya \end{cases} \quad (3.15)$$

$$\Delta K_{1,p}^1 = I_{rot180} * \Delta C_{1,\sigma}^1 \quad (3.16)$$

Pada *layer* konvolusi, ada bias yang *diupdate* untuk pendukung proses pembelajaran. Perubahan nilai bias dapat dihitung seperti perubahan nilai filter, namun tak mengikutsertakan *input*. Perubahan nilai bias merupakan jumlah seluruh setelah di *unsampling*. Nilai bias dapat dicari dengan persamaan :

$$\Delta b_{1,p}^1 = \sum_{i=1}^{row\ of\ C^1} \sum_{j=1}^{column\ of\ C^1} \Delta C_p^1(i,j) \quad (3.17)$$

Setelah memperhitungkan perubahan tiap lapisan, proses *update* nilai filter, bias pada *layer* konvolusi, bobot pada *fully-connected layer*, serta bias sebelumnya kemudian dijalankan dengan menggunakan persamaan berikut :

$$k_{1,p}^1 = k_{1,p}^1 - \alpha \cdot k_{1,p}^1 \quad (3.18)$$

$$b_{1,p}^1 = b_{1,p}^1 - \alpha \cdot b_{1,p}^1 \quad (3.19)$$

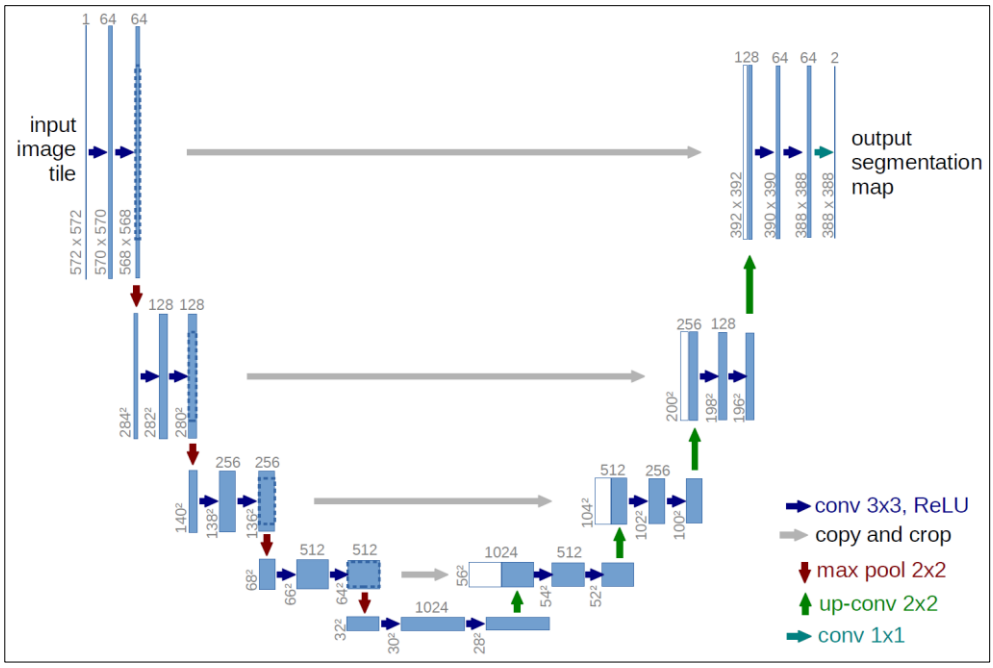
$$W = W - \alpha \cdot \Delta W \quad (3.20)$$

$$b = b - \alpha \cdot \Delta b \quad (3.21)$$

Proses ini berlaku sampai kondisi *stop* ditemukan, yakni kondisi berupa *epoch* bernilai akurat tertinggi tercapai atau nilai *loss* ada pada titik terendah. Proses mengubah nilai bias, filter dan bobot dijalankan tiap satu set data masuk kedalam sebuah jaringan.

2.3.9 Arsitektur U-Net

Arsitektur U-Net merupakan teknik pengembangan arsitektur *autoencoder*. U-Net dapat mempresentasikan gambaran wilayah lokal disekitar *pixels* pada segmentasi *image*. Mengacu pada gambar 2.16, tampilan arsitektur U-Net terdiri atas *contracting path* pada sisi kiri dan *expansive path* pada sisi kanan. *Contracting path* dibentuk oleh jaringan konvolusi yang berlapis dan diikuti operasi fungsi aktivasi ReLU serta *max pooling*. Saat proses *contracting path* bekerja, informasi yang disimpan pada *layer* sebelumnya akan berkurang dan informasi *fitur* yang semakin ditingkatkan. Pada sisi arsitektur biasa disebut *expansive path* yang dibentuk oleh penggabungan fitur berurutan *unsampling* dan gabungan lapisan beresolusi tinggi dari jaringan *contracting path* hasilkan *output unsampling* atau disebut *skip-connection. Skip-connection* adalah cara lain menutupi kekurangan *autoencoder* yang mengalami *loss* dari beberapa resolusi ketika proses *encoding* hingga menghasilkan *output image* yang blur (Purbaya, 2018).



Gambar 2.16. Arsitektur U-Net (Ronneberger, 2015)

2.4 Penelitian Sebelumnya

Berdasarkan penelitian yang akan dilakukan, referensi dari penelitian terdahulu sangat penting untuk melakukan sebuah penelitian agar bertujuan mengetahui hubungan antara penelitian yang akan dilakukan dengan penelitian terdahulu hingga dengan menambah referensi tersebut diupayakan menghindari duplikasi dalam penelitian ini.

Penelitian mengenai *image classification* dengan memanfaatkan algoritma dari *Convolutional Neural Network* (CNN) dari *image* wayang golek yang telah dilakukan oleh Triano Nurhikmat (2018). Penelitian ini menjelaskan penggunaan CNN untuk mengklasifikasikan gambar wayang golek 2D secara otomatis. Tingkat akurasi yang didapat sebesar 93% (Nurhikmat, 2018).

Penelitian tentang *image classification* di dunia geofisika menggunakan kemampuan *machine learning* telah banyak dilakukan salah satunya oleh Bergen dkk (2019). Penelitian ini menjelaskan aplikasi *machine learning* pada data geosains dalam memahami perilaku bumi (Bergen, 2019). Kemudian penelitian yang dilakukan oleh Wrona dkk (2018). Penelitian ini menjelaskan cara melakukan klasifikasi fasies seismik secara otomatis dengan metode SVM (Wrona, 2018). Penelitian lainnya yaitu dilakukan oleh Ketut Toto Suryahadinata (2020). Penelitian ini menjelaskan cara melakukan intepretasi seismik terintegrasi dengan metode CNN. (Suryahadinata, 2020).