**PENERJEMAH BAHASA ISYARAT MENGGUNAKAN METODE *DEEP LEARNING MODEL MULTICHANNEL 2D CONVOLUTIONAL NEURAL NETWORK (M2D CNN)***

**SKRIPSI**



oleh

**Rizkika Zakka Palindungan**

**NIM E41170164**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI JEMBER**

**2020**

# BAB 1. PENDAHULUAN

## Latar Belakang

Manusia adalah makhluk sosial yang saling berkomunikasi dan berinteraksi satu dengan yang lainnya melalui bahasa. Bentuk komunikasi ini biasanya dilakukan secara verbal/lisan, yang artinya komunikasi dengan menggunakan kata-kata. Akan tetapi The World Health Organization (WHO) memberikan pernyataan bahwa lebih dari 5% populasi dunia atau 466 juta orang mengalami gangguan pendengaran (432 juta orang dewasa dan 34 juta anak). Diperkirakan pada tahun 2050 lebih dari 900 juta orang atau satu dari setiap sepuluh orang akan mengalami gangguan pendengaran. Dengan informasi tersebut dapat disimpulkan bahwa akan terjadi peningkatan populasi yang mengalami gangguan pendengaran maupun juga yang mengalami gangguan berbicara. Komunitas yang mengalami gangguan pendengaran disebut sebagai tunarungu. Tunarungu memiliki cara berkomunikasi sendiri tanpa menggunakan bahasa lisan yaitu dengan bahasa isyarat. Sedangkan bahasa isyarat ini sulit dipahami oleh masyarakat pada umumnya, dan membuat komunitas tersebut merasa terasingkan di lingkungan sekitarnya. Bahasa isyarat itu sendiri merupakan bahasa yang digunakan untuk berkomunikasi dengan menggunakan gerak bibir dan bahasa tubuh, termasuk ekspresi wajah dan pandangan mata. Selain itu, bahasa isyarat adalah gerakan-gerakan yang sudah disepakati maknanya dan digunakan untuk bertukar informasi.

Untuk mengatasi keterbatasan dalam hal berkomunikasi, teknologi adalah jawabannya. Dengan berkembangnya teknologi visi komputer (*computer vision*), komunikasi antara manusia bisa dijembatani oleh mesin. Teknologi visi komputer (*computer vision*) merupakan salah satu bidang pengetahuan dalam memanfaatkan kecerdasan buatan yang melatih komputer untuk menafsirkan dan memahami dunia visual. Teknologi ini menggunakan gambar digital dari kamera dan bisa juga dari sensor yang kemudian diolah menggunakan model *deep learning*. Maka dari itu mesin dapat secara akurat mengidentifikasi dan mengklasifikasikan objek kemudian bereaksi terhadap apa yang mesin “lihat”. Diharapkan teknologi visi komputer ini bisa dikembangkan untuk membuat sebuah piranti yang mampu menerjemahkan bahasa isyarat ke dalam tulisan bahkan suara. Meskipun harus melewati penelitian dan percobaan yang cukup panjang tetapi teknologi ini pasti akan membantu kehidupan manusia menjadi lebih baik dimasa depan.

Penelitian ini dilakukan dengan pendekatan berbasis visi computer. Pada pendekatan berbasis visi computer digunakan data yang berasal dari tangkapan kamera yang akan menangkap gerakan bahasa isyarat. Sebelum dilakukan proses pengenalan harus melakukan praproses terlebih dahulu. Ada beberapa praproses yang harus dilakukan seperti mengumpulkan data mentah dengan menggunakan kamera, deteksi objek, segmentasi dan konversi warna, thresholding, filtering, operasi morfologi. Setelah data berhasil didapatkan selanjutnya melakukan normalisasi untuk pembuatan model dataset. Normalisasi ini memastikan bahwa data yang berhasil dimodelkan bisa dengan efektif dan efisien digunakan oleh sistem cerdas. Semua praproses tersebut dilakukan untuk memisahkan objek yang sangat penting dari noise. Sampai pada akhirnya sistem komputer yang sudah ditanami dengan teknologi kecerdasan buatan (artificial inteligent) mampu memanfaatkan model dari dataset tersebut dengan maksimal.

Untuk membuat kecerdasan buatan. Salah satu pendekatan yang bisa digunakan adalah menggunakan Jaringan Syaraf Tiruan (JST) yang terinspirasi dari jaringan syaraf pada manusia. Konsep tersebut kemudian dikembangkan lebih lanjut dalam *Deep Learning*. *Deep Learning* telah menjadi salah satu topik hangat dalam dunia *Machine Learning* karena kapabilitasnya yang signifikan dalam memodelkan berbagai data kompleks seperti citra atau visual dan suara. Metode *Deep Learning* yang saat ini memiliki hasil paling signifikan dalam pengenalan citra adalah *Convolutional Neural Network* (CNN). Hal tersebut dikarenakan CNN berusaha meniru sistem pengenalan citra pada visual *cortex* manusia sehingga memiliki kemampuan mengolah informasi citra.

Sebernarnya konsep penelitian dan pengembangan sistem komputer vision untuk penerjemah bahasa isyarat sudah ada beberapa yang telah dilakukan antara lain : Sistem Pengenalan Bahasa Isyarat Indonesia Dengan Menggunakan Metode Fuzzy K-Nearest Neighbor dengan tingkat akurasi 88% (Gafar, 2017), Pengenalan Angka Sistem Isyarat Bahasa Indonesia Dengan Menggunakan Metode Convolutional Neural Network dengan menganalisis tingkat akurasi dari parameter jumlah total epoch pada model dataset yaitu 25 epoch, 50 epoch, 100 epoch yang memiliki akurasi 67.66%, 89.44%, 96.44% (Bakti., 2019), Model Penerjemah Bahasa Isyarat Indonesia (Bisindo) Menggunakan Convolutional Neural Network dengan melakukan perbandingan 2 buah model dataset yang memiliki akurasi 94.38% dan 30% (Fadillah, 2020). Sebagai pengembangan sistem lebih lanjut, penelitian ini akan menggunakan Bahasa Isyarat Indonesia (BISINDO) sebagai jenis bahasa isyarat yang dipakai didalam dataset. Selanjutnya akan menambah jumlah kamera *input* menjadi 2 buah untuk mendeteksi tangan dari 2 sisi yaitu sisi depan dan sisi samping hal ini bertujuan untuk meningkatkan kemampuan dan akurasi dalam mendeteksi bahasa isyarat yang lebih kompleks. Pada proses pengolahan data citra dari kamera akan dilakukan penghilangan bagian wajah dan leher dengan menggunakan bantuan *Haar Cascade Classifier*, kemudian melakukan *skin mask* atau deteksi warna kulit untuk memfokuskan pada objek tangan. Algoritma yang akan digunakan pada penelitian ini adalah salah satu model pengembangan dari CNN yaitu model arsitektur *Multichannel 2D Convolutional Neural Network*, algoritma ini dipilih berdasarkan tingkat kemampuan dan akurasi untuk mengolah 2 atau lebih data input citra dari sebuah objek dengan sudut pandang yang berbeda hal ini dibuktikan dari penelitian berjudul A Multichannel 2D Convolutional Neural Network Model for Task-Evoked fMRI Data Classification dengan objek input berupa gambar otak dari 3 sisi yaitu sumbu x, y, z yang memiliki akurasi 83.20% (Hu., 2019).

## Rumusan Masalah

Berdasarkan uraian dari latar belakang, terdapat beberapa permasalahan yang bisa dirumuskan yaitu sebagai berikut :

1. Bagaimana proses ekstraksi fitur warna untuk mendeteksi warna kulit dan dan objek tangan pada data citra dari kamera video?
2. Bagaimana pengolahan dataset yang digunakan untuk data training metode Deep Learning Model Multichannel 2D Convolutional Neural Network?
3. Bagaimana implementasi metode Deep Learning Model Multichannel 2D Convolutional Neural Network dalam menerjemahkan gerakan bahasa isyarat?

## Tujuan Penelitian

Adapun tujuan penelitian ini adalah sebagai berikut :

1. Untuk mengetahui teknik pengolahan citra yang tepat dalam ekstraksi fitur warna dan deteksi objek tangan data citra yang berasal dari kamera.
2. Untuk mendapatkan model dataset dari proses data training yang bisa digunakan oleh mechine learning (M2D CNN) secara efektif dan efisien.
3. Untuk mengetahui hasil metode Deep Learning Model Multichannel 2D Convolutional Neural Network dalam memahami dan menerjemahkan gerakan bahasa isyarat.

## Manfaat Penelitian

Adapun manfaat penelitian ini adalah sebagai berikut :

1. Meningkatkan efisiensi komputasi pada sistem sehingga memiliki tingkat fungsional dan akurasi yang baik didalam menerjemahkan bahasa isyarat.
2. Memberikan kemampuarn sebuah device untuk belajar mengingat dan memahami gerakan bahasa isyarat manusia.
3. Mampu mengembangkan software untuk memudahkan masyarakat luas dalam memahami bahasa isyarat dengan bantuan mechine learning.

# BAB 2. TINJAUAN PUSTAKA

## State Of The Art

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Penelitian 1 | Penelitian 2 | Penelitian 3 | Penelitian 4 |
| Judul | Sistem Pengenalan Bahasa Isyarat Indonesia dengan Menggunakan Metode Fuzzy K-Nearest Neighbor | Pengenalan Angka Sistem Isyarat Bahasa Indonesia Dengan Menggunakan Metode Convolutional Neural Network | Model Penerjemah Bahasa Isyarat Indonesia (BISINDO) Menggunakan Convolutional Neural Network | Sistem Penerjemah Bahasa Isyarat Menggunakan Metode Deep Learning Model Multichannel 2D Convolutional Neural Network (M2D CNN) |
| Penulis | Agum Agidtama Gafar | Mochamad Bagus Setiyo Bakti dan Yuliana Melita Pranoto | Riestiya Zain Fadillah | Rizkika Zakka Palindungan |
| Tahun | 2017 | 2019 | 2020 | 2020 |
| Metode | Fuzzy K-Nearest Neighbor (FKNN) | Convolutional Neural Network (CNN) | Convolutional Neural Network (CNN) | Multichannel 2D Convolutional Neural Network (M2D CNN) |
| Objek | Citra Bahasa Isyarat Tangan (SIBI) | Citra Bahasa Isyarat Tangan (SIBI) | Citra Bahasa Isyarat Tangan (BISINDO) | Citra Bahasa Isyarat Tangan (BISINDO) |

## Bahasa Isyarat

Bahasa isyarat merupakan jenis komunikasi non verbal karena merupakan bahasa yang tidak menggunakan suara tetapi menggunakan bentuk dan arah tangan, pergerakan tangan, bibir, badan serta ekspresi wajah untuk menyampaikan maksud dan pikiran dari seorang penutur. Kaum tunarungu adalah kelompok utama yang menggunakan bahasa ini. Bahasa isyarat biasanya pengkombinasian dari bentuk, orientasi dan gerak tangan, lengan, tubuh serta ekspresi wajah untuk mengungkapkan isi pikiran (Zuhir dan Amri, 2019).

Di Indonesia, ada dua sistem dari bahasa isyarat yang digunakan yaitu: Bahasa Isyarat Indonesia (BISINDO) dan Sistem Isyarat Bahasa Indonesia (SIBI). SIBI adalah bahasa isyarat yang diperkenalkan secara awal oleh Alm. Anton Widyatmoko yang merupakan mantan kepala sekolah SLB/B Widya Bakti di Semarang dalam proses penciptaannya tidak melalui musyawarah dan persetujuan dari Gerakan Kesejahteraan Tunarungu Indonesia (GERKATIN) tetapi berkolaborasi dengan mantan kepala sekolah SLB/B di Jakarta dan Surabaya dengan hasil akhir sebuah kamus SIBI. BISINDO adalah bahasa isyarat yang mengadopsi nilai budaya asli Indonesia dan mudah dapat digunakan untuk berkomunikasi diantara kaum tunarungu dalam kehidupan sehari - hari. Kecepatan dan kepraktisannya dari BISINDO membuat lebih mudah untuk memahami dan mengerti bagi kaum tunarungu walaupun tidak mengikuti faedah tata bahasa dari bahasa Indonesia (Yunanda, 2018).

### Sejarah Bahasa Isyarat Indonesia (BISINDO)

Menurut Laura Lesmana Wijaya selaku Ketua Pusat Bahasa Isyarat Indonesia (Pusbisindo), Bisindo dapat diartikan sebagai sebuah terminologi yang digunakan untuk menunjuk pada bahasa isyarat alami yang digunakan oleh komunitas Tuli di Indonesia. Berdasarkan penjelasan tersebut dapat disimpulkan bahwa sejarah Bisindo sejalan dengan kemunculan bahasa isyarat alami yang terdapat di Indonesia (Yohans, Arjawa dan Punia., 2019).

Kemunculan bahasa isyarat alami diyakini telah berlangsung sejak tahun 1933 ketika sekolah khusus Tuli pertama yaitu Sekolah Luar Biasa (SLB)/B Cicendo, Bandung, Jawa Barat berdiri. Selain itu, terdapat pula sekolah khusus Tuli lainnya yang berdiri pada tahun-tahun berikutnya seperti SLB/B Dena Upakara, Wonosobo, Jawa Tengah (sekolah khusus perempuan) pada tahun 1938, SLB/B Don Bosco, Wonosobo, Jawa tengah (sekolah khusus laki- laki) pada tahun 1955, dan SLB/B Santi Rama (Jakarta) pada tahun 1970 (Tim Produksi Bahasa Isyarat Jakarta, 2014: vii). Penjelasan ini diperkuat dengan keberadaan bahasa isyarat Jakarta yang variasinya berasal dari pencampuran bahasa isyarat asli, termasuk bahasa isyarat yang digunakan oleh orang orang Tuli yang pernah mendapatkan pendidikan formal di sekolah khusus Tuli tersebut (Tim Produksi Bahasa Isyarat Jakarta, 2014: vii).

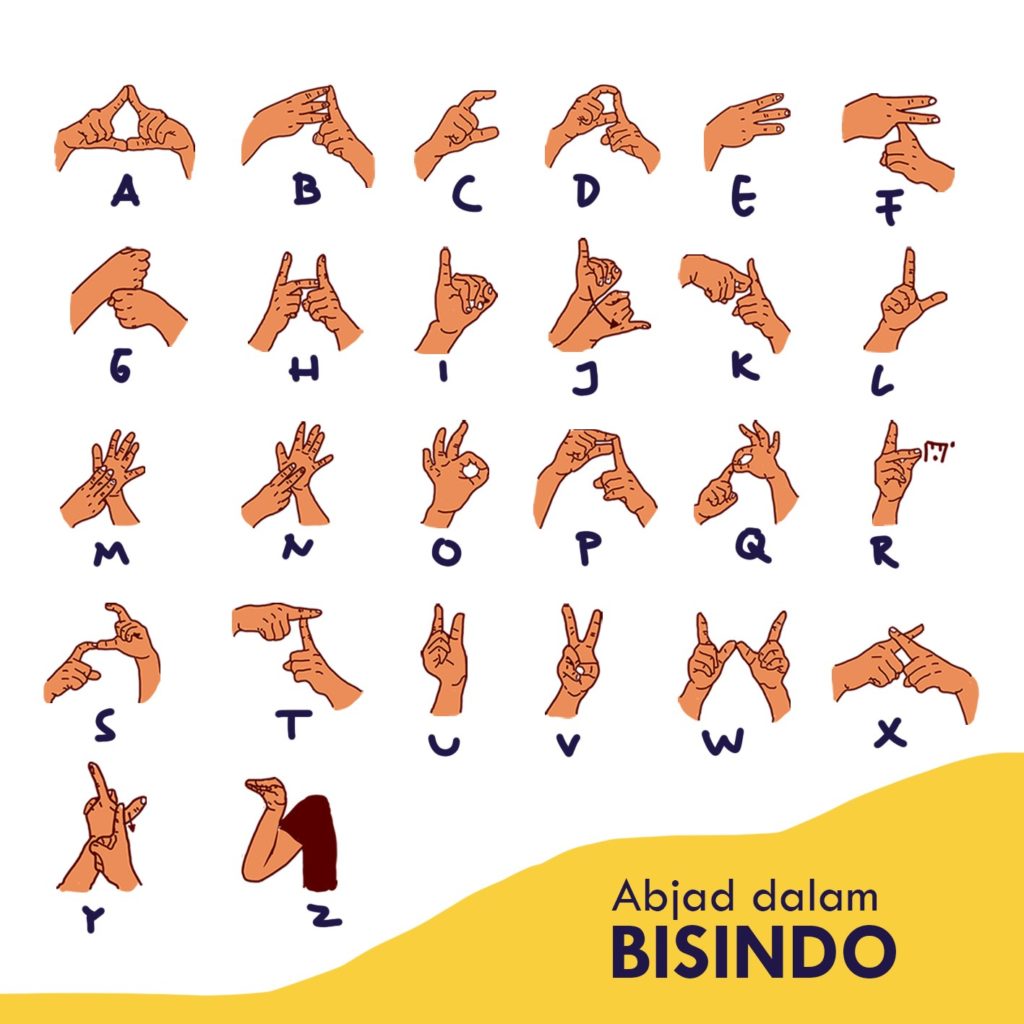
Perkembangan bahasa isyarat alami di Indonesia tidak serta merta mendapatkan pengakuan oleh pemerintah Indonesia. Sistem Isyarat Bahasa Indonesia (SIBI) merupakan sarana komunikasi yang terlebih dahulu diakui oleh pemerintah Indonesia. Pengakuan dan pembakuan atas penggunaan SIBI secara resmi ditetapkan pada tahun 1994 melalui Keputusan Mendikbud RI Nomor 0161/U/1994.

Disebarluaskan dan dibakukannya penggunaan SIBI sebagai sistem isyarat buatan yang bersifat nasional tidak sepenuhnya diterima oleh komunitas Tuli. Komunitas Tuli menilai bahwa keberadaan SIBI tidak merepresentasikan bahasa isyarat asli Indonesia, terdapat berbagai bentuk isyarat yang tidak sesuai dengan isyarat yang berkembang di komunitas Tuli. Salah satu isyarat yang banyak diterapkan pada kamus SIBI, yaitu isyarat yang terdapat pada sistem isyarat American Sign Language (ASL).

Permasalahan dan pertentangan atas penggunaan SIBI menjadi salah satu faktor yang memengaruhi munculnya penggunaan istilah Bisindo. Laura Lesmana Wijaya menjelaskan bahwa penggunaan istilah Bahasa Isyarat Indonesia atau disingkat Bisindo dimulai pada awal tahun 2000. Istilah tersebut muncul melalui pelaksanaan kongres yang dilaksanakan oleh Gerkatin. Penentuan istilah tersebut digunakan untuk menunjuk pada bahasa isyarat alami yang berkembang di komunitas Tuli.

### Isyarat Abjad Dalam Bisindo

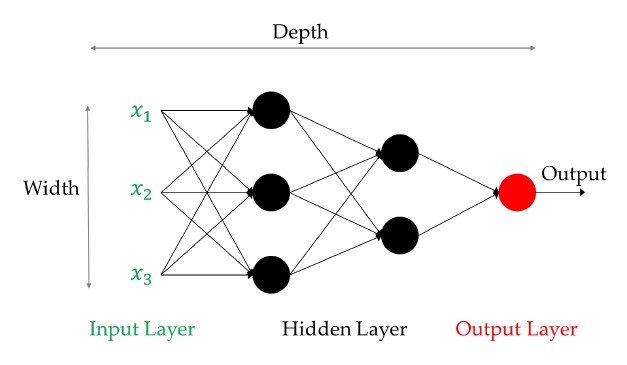
Berikut adalah gambar gerakan bahasa isyarat dari abjad mulai A hingga Z yang diterjemahkan kedalam Bahasa Isyarat Indonesian (BISINDO).



## Artificial Neural Network

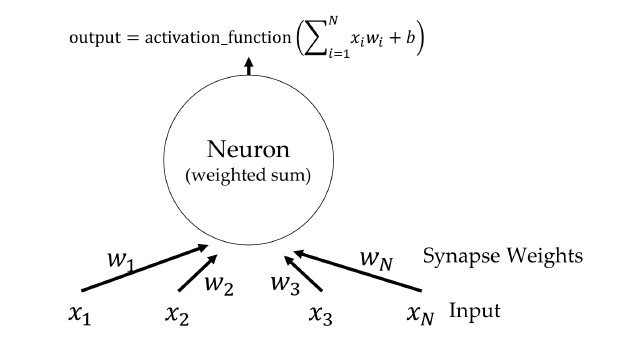
*Artificial neural network* adalah salah satu algoritma *supervised learning* yang populer dan bisa juga digunakan untuk *semi-supervised* atau *unsupervised* *learning* (Amir Atiya, 1994). Walaupun tujuan awalnya adalah untuk mensimulasikan jaringan saraf biologis, jaringan tiruan ini sebenenarnya simulasi yang terlalu disederhanakan, artinya simulasi yang dilakukan tidak mampu menggambarkan kompleksitas jaringan biologis manusia.

*Artificial Neural Network* (selanjutnya disingkat ANN), menghasilkan model yang sulit dibaca dan dimengerti oleh manusia karena memiliki banyak layer (kecuali *single perceptron*) dan sifat **non-linear** (merujuk pada fungsi aktivasi). Pada bidang riset ini, ANN disebut agnostik - kita percaya, tetapi sulit membuktikan kenapa konfigurasi parameter yang dihasilkan *training* bisa benar. Konsep matematis ANN itu sendiri cukup *solid*, tetapi *interpretability* (tingkat pemahaman dan ‘insight’) model rendah menyebabkan kita tidak dapat menganalisa proses inferensi(penyimpulan) yang terjadi pada model ANN. Secara matematis, ANN ibarat sebuah graf. ANN memiliki neuron/*node* *(vertex)*, dan sinapsis *(edge)*. Karena memiliki struktur seperti graf, operasi pada ANN mudah dijelaskan dalam notasi aljabar linear. Sebagai gambaran, ANN berbentuk seperti Gambar XX.X (*deep neural network*, salah satu varian arsitektur). *Depth* (kedalaman) ANN mengacu pada jumlah layer. Sementara width (lebar) ANN mengacu pada jumlah unit pada *layer*.



### Single Perceptron

Bentuk terkecil (minimal) sebuah ANN adalah *single perceptron* yang hanya terdiri dari sebuah neuron. Sebuah neuron diilustrasikan pada Gambar XX.X. Secara matematis, terdapat *feature vector* x yang menjadi *input* bagi neuron tersebut (*feature vector* merepresentasikan suatu *data point*, *event* atau *instans*). Neuron akan memproses *input* x melalui perhitungan jumlah perkalian antara nilai *input* dan *synapse weight*, yang dilewatkan pada fungsi non linear. Pada *training*, yang dioptimasi adalah nilai *synapse weight (learning parameter)*. Selain itu, terdapat juga bias *b* sebagai kontrol tambahan (materi *steepest gradient descent*). Output dari neuron adalah hasil fungsi aktivasi dari perhitungan jumlah perkalian antara nilai *input* dan *synapse weight*. Ada beberapa macam fungsi aktivasi, misal ***step function, sign function, rectifier*** dan ***sigmoid function***. Bila diplot menjadi grafik, fungsi ini memberikan bentuk seperti huruf S.



Perhatikan kembali, Gambar XX.X sesungguhnya adalah operasi aljabar linear. Single perceptron dapat dituliskan kembali sebagai XX.X.

Dimana o adalah *output* dan f adalah fungsi non-linear yang dapat diturunkan secara matematis (*differentiable non-linear function* - selanjutnya disebut “fungsi non linear” saja.). Bentuk ini tidak lain dan tidak bukan adalah persamaan model linear yang ditransformasi dengan fungsi non-linear. Secara filosofis, ANN bekerja mirip dengan model linear, yaitu mencari *decision boudary*. Apabila beberapa model non-linear ini digabungkan, maka kemampuannya akan menjadi lebih hebat. Yang menjadikan ANN "spesial" adalah penggunaan fungsi non-linear.

Untuk melakukan pembelajaran *single perceptron, training* dilakukan menggunakan ***perceptron training rule***. Prosesnya sebagai berikut:

1. Inisiasi nilai *synapse weights*, bisa *random* ataupun dengan aturan tertentu.
2. Lewatkan input pada neuron, kemudian kita akan mendapatkan nilai *output*. Kegiatan ini disebut ***feedforward***.
3. Nilai *output (actual output)* tersebut dibandingkan dengan *desired output*.
4. Apabila nilai output sesuai dengan *desired output*, tidak perlu mengubah apa-apa.
5. Apabila nilai *output* tidak sesuai dengan *desired output*, hitung nilai *error (loss)* kemudian lakukan perubahan terhadap *learning parameter (synapse weight)*.
6. Ulangi langkah-langkah ini sampai tidak ada perubahan nilai *error*, nilai *error* kurang dari sama dengan suatu *threshold* (biasanya mendekati 0), atau sudah mengulangi proses latihan sebanyak **T** kali (*threshold*).

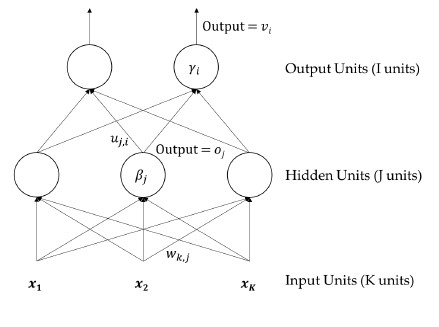
Error function diberikan pada persamaan (dapat diganti dengan absolute value) dan perubahan *synapse weight* diberikan pada persamaan , dimana y melambangkan d*esired output*, melambangkan actual output untuk x sebagai input. disebut sebagai learning rate.

Hasil akhir pembelajaran *(learning)* adalah konfigurasi *synapse weight*. Saat klasifikasi, kita melewatkan *input* baru pada jaringan yang telah dibangun, kemudian tinggal mengambil hasilnya. Pada contoh kali ini, seolah-olah *single perceptron* hanya dapat digunakan untuk melakukan *binary classification* (hanya ada dua kelas, nilai 0 dan 1). Untuk multi-class classification, kita dapat menerapkan berbagai strategi, misal thresholding, i.e., nilai output 0 - 0.2 mengacu pada kelas pertama, 0.2 - 0.4 untuk kelas kedua, dst.

### Multilayer Perceptron

*Multilayer perceptron* (MLP) yang juga dikenal sebagai *feedforward neural network* secara literal memiliki beberapa *layers*. Pada lecture note ini, secara umum ada tiga *layers*: *input*, *hidden*, dan *output* *layer*. *Input layer* menerima *input* (tanpa melakukan operasi apapun), kemudian nilai *input* (tanpa dilewatkan ke fungsi aktivasi) diberikan ke *hidden units*. Pada *hidden units*, *input* diproses dan dilakukan perhitungan hasil fungsi aktivasi untuk tiap-tiap neuron, lalu hasilnya diberikan ke layer berikutnya. *Output* dari *input layer* akan diterima sebagai *input* bagi *hidden layer*. Begitupula seterusnya *hidden layer* akan mengirimkan hasilnya untuk *output layer*.

Kegiatan ini dinamakan *feed forward*. Hal serupa berlaku untuk *artificial neural network* dengan lebih dari tiga *layers*. Parameter neuron dapat dioptimisasi menggunakan metode *gradient-based optimization*. Perlu diperhatikan, MLP adalah gabungan dari banyak fungsi *non-linear*. Gabungan banyak fungsi *non-linear* ini lebih hebat dibanding single perceptron. Masing-masing neuron terkoneksi dengan semua neuron pada *layer* berikutnya. Konfigurasi ini disebut sebagai ***fully connected***. MLP pada umumnya menggunakan konfigurasi *fully connected*.



oj =

vi = =

Perhatikan persamaan XX.X dan XX.X untuk menghitung *output* pada *layer* yang berbeda. u,w adalah *learning parameters*. , melambangkan *noise* atau *bias*. K adalah banyaknya *input units* dan J adalah banyaknya *hidden units*.

Persamaan XX.X dapat disederhanakan penulisannya sebagai persamaan XX.X. Persamaan XX.X terlihat relatif lebih “elegan”. Sehingga ANN dapat direpresentasikan dengan notasi operasi aljabar.

Untuk melatih MLP, algoritma yang umumnya digunakan adalah **backpropagation**. Arti kata *backpropagation* sulit untuk diterjemahkan ke dalam bahasa Indonesia. Peneliti memperbaharui parameter (*synapse weights*) secara bertahap (dari *output* ke *input* layer, karena itu disebut *backpropagation*) berdasarkan *error/loss* (*output* dibandingkan dengan *desired output*). Intinya adalah mengkoreksi *synapse weight* dari output layer ke *hidden layer*, kemudian *error* tersebut dipropagasi ke layer sebelum-sebelumnya. Artinya, perubahan *synapse weight* pada suatu layer dipengaruhi oleh perubahan synapse weight pada layer setelahnya. Backpropagation tidak lain dan tidak bukan adalah metode *gradient-based optimization* yang diterapkan pada ANN.

Pertama-tama diberikan pasangan *input* (x) dan *desired output* (y) sebagai *training data*. Untuk meminimalkan *loss*, algoritma *backpropagation* menggunakan prinsip *gradient descent*. Cara menurunkan *backpropagation* menggunakan teknik *gradient descent*, yaitu menghitung *loss* ANN pada Gambar XX.X **yang menggunakan fungsi aktivasi sigmoid**.

*Error*, untuk MLP diberikan oleh persamaan XX.XX (untuk satu data point), dimana I adalah banyaknya *output unit* dan adalah kumpulan *weight matrices* (semua *parameter* pada MLP). Kami inggatkan kembali perhitungan *error* bisa juga menggunakan nilai absolut.

E() =

Proses penurunan untuk melatih MLP, *error/loss* diturunkan terhadap tiap learning parameter. Diferensial uj,i diberikan oleh turunan sigmoid function.

=

=

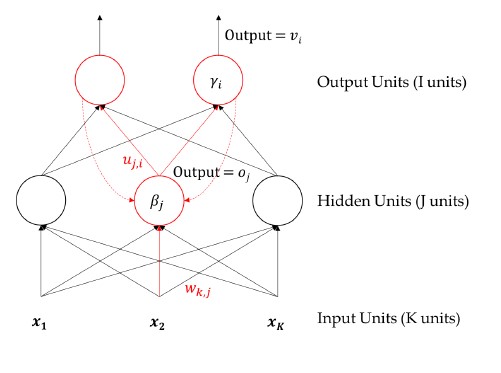
Diferensial wk,j diberikan oleh turunan *sigmoid function*

=

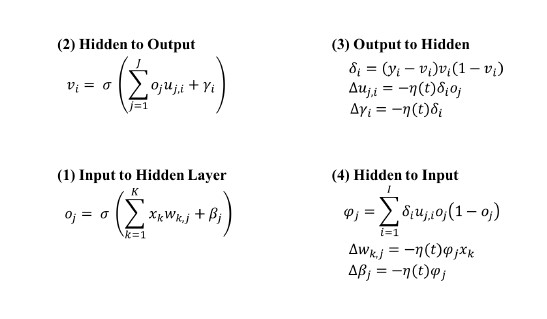
=

=

Perhatikan, *diferensial* wk,j memiliki ∑ sementara uj,i tidak ada. Hal ini disebabkan karena uj,I hanya berkorespondensi dengan satu *output* neuron. Sementara wk,j berkorespondensi dengan banyak *output* neuron. Dengan kata lain, nilai wk,j mempengaruhi hasil operasi yang terjadi pada banyak *output* neuron, sehingga banyak neuron mempropagasi error kembali ke wk,j. Ilustrasi diberikan pada Gambar XX.X.

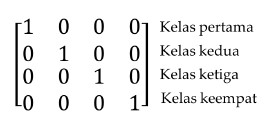


Metode penurunan serupa dapat juga digunakan untuk menentukan perubahan dan . Jadi proses *backpropagation* untuk kasus Gambar XX.X dapat diberikan seperti pada Gambar XX.X dimana adalah *learning rate*. Untuk *artificial neural network* dengan lebih dari 3 layers juga bisa menurunkan persamaannya. Secara umum, proses melatih ANN (apapun variasi arsitekturnya) mengikuti *framework* *perceptron training rule*.



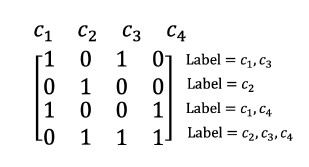
### Multi-class Classification

*Multilayer perceptron* dapat memiliki *output unit* berjumlah lebih dari satu. Seumpama mempunyai empat kelas, dengan demikian peneliti dapat merepresentasikan keempat kelas tersebut sebagai empat *output units*. Kelas pertama direpresentasikan dengan *unit* pertama, kelas kedua dengan *unit* kedua, dst. Untuk C kelas, kita dapat merepresentasikannya dengan C *output units*. Kita dapat merepresentasikan data harus dimasukkan ke kelas mana menggunakan *sparse vector*, yaitu bernilai 0 atau 1. Elemen ke-i bernilai 1 apabila data masuk ke kelas ci, sementara nilai elemen lainnya adalah 0 (ilustrasi pada Gambar XX.X).Output ANN dilewatkan pada suatu fungsi softmax yang melambangkan probabilitas *class-assignment*, i.e., kita ingin output agar semirip mungkin dengan *sparse vector* (*desired output*). Pada kasus ini, *output* ANN adalah sebuah distribusi yang melambangkan input di-*assign* ke kelas tertentu. Cross entropy cocok digunakan sebagai utility function ketika output berbentuk distribusi.



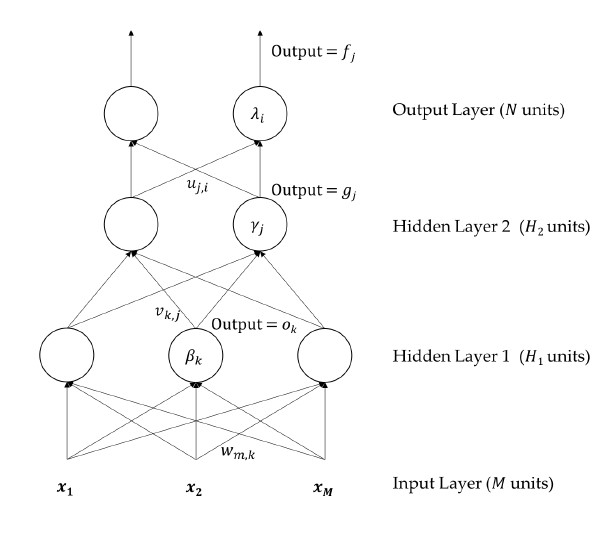
### Multi-label Classification

Seperti halnya *multi-class classification*, kita dapat menggunakan sejumlah C neuron untuk merepresentasikan C kelas pada *multi-label classification* Perbedaan *multi-class* dan *multilabel* terletak pada cara interpretasi *output* dan evaluasi *output*. Pada umumnya, *layer* terakhir diaktivasi dengan fungsi sigmoid, dimana tiap neuron ni merepresentasikan probabilitas suatu dapat diklasifikasikan sebagai kelas ci atau tidak. *Cross entropy* juga cocok untuk mengevaluasi (dan melatih) *multi-label classification*.



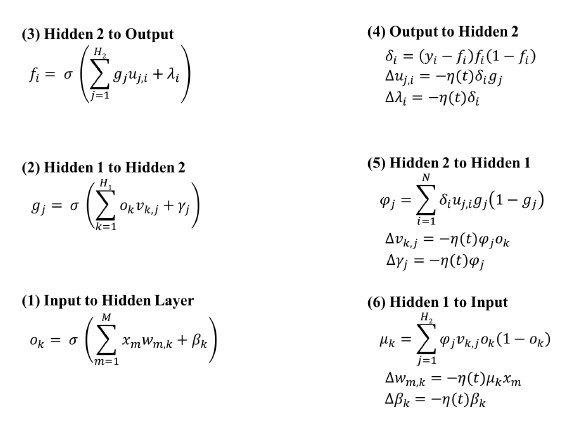
### Deep Neural Network

*Deep Neural Network* (DNN) adalah *artificial neural network* yang memiliki banyak *layer*. Pada umumnya, deep neural network memiliki lebih dari 3 *layers (input layer,* N *hidden layers, output layer* ), dengan kata lain adalah MLP dengan lebih banyak *layer*. Karena ada relatif banyak *layer*, disebutlah *deep*. Proses pembelajaran pada DNN disebut sebagai *deep learning*11. Jaringan neural *network* pada DNN disebut *deep neural network*. Perhatikan Gambar XX.XX yang memiliki 4 layers.



Cara menghitung *final output* sama seperti MLP, diberikan pada persamaan XX.XX dimana adalah *noise* atau *bias*.

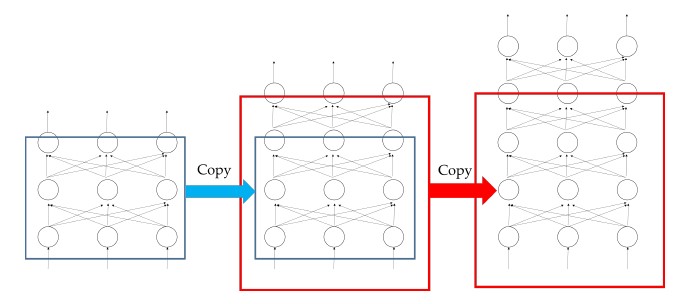
Cara melatih *deep neural network*, salah satunya dapat menggunakan backpropagation. Hanya perlu menurunkan rumusnya saja. Hasil proses penurunan dapat dilihat pada Gambar XX.XX.



*Deep network* terdiri dari banyak *layer* dan *synapse weight*, karenanya estimasi parameter susah dilakukan. Arti filosofisnya adalah susah/lama untuk menentukan relasi antara *input* dan *output*. Walaupun deep learning sepertinya kompleks, tetapi entah kenapa dapat bekerja dengan baik untuk permasalahan praktis. *Deep learning* dapat menemukan relasi "tersembunyi" antara input dan output, yang tidak dapat diselesaikan menggunakan *multilayer perceptron* (3 layers).

Banyak orang percaya *deep neural network* lebih baik dibanding *neural network* yang lebar tapi sedikit layer, karena terjadi lebih banyak transformasi. Maksud lebih banyak transformasi adalah kemampuan untuk merubah input menjadi suatu representasi (tiap hidden layer dapat dianggap sebagai salah satu bentuk representasi input) dengan langkah hierarchical. Seperti contoh permasalahan XOR, permasalahan *non-linearly separable* pun dapat diselesaikan apabila kita dapat mentransformasi data (representasi data) ke dalam bentuk *linearly separable* pada ruang yang berbeda. Keuntungan utama deep learning adalah mampu merubah data dari *non-linearly separable* menjadi *linearly separable* melalui serangkaian transformasi (*hidden layers*). Selain itu, deep learning juga mampu mencari *decision boundary* yang berbentuk *non-linear*, serta mengsimulasikan interaksi *non-linear* antar fitur.

Karena memiliki banyak parameter, proses latihan ANN pada umumnya lambat. Ada beberapa strategi untuk mempercepat pembelajaran menggunakan deep learning, misalnya: regularisasi, *successive learning*, dan penggunaan *autoencoder*. Sebagai contoh, arti *successive learning* adalah jaringan yang dibangun secara bertahap. Misal kita latih ANN dengan 3 *layers*, kemudian kita lanjutkan 3 *layers* tersebut menjadi 4 *layers*, lalu kita latih lagi menjadi 5 *layers*, dst (mulai dari hal kecil). Ilustrasinya dapat dilihat pada Gambar XX.XX. Menggunakan *deep learning* harus hati-hati karena pembelajaran cenderung *divergen* (artinya, *minimum squared error* belum tentu semakin rendah seiring berjalannya waktu - *swing* relatif sering).



### Regularization and Dropout

Pada model linear. Model harus mampu mengeneralisasi dengan baik (kinerja baik pada *training* data dan *unseen examples*). Peneliti dapat menambahkan fungsi regularisasi untuk mengontrol kompleksitas ANN. Regularisasi pada ANN cukup *straightforward* seperti regularisasi pada model linear.

Selain itu, agar ANN tidak “bergantung” pada satu atau beberapa *synapse weights* saja, peneliti dapat menggunakan *dropout*. **Dropout** berarti me-nol-kan nilai *synapse weights* dengan nilai rate tertentu. Misalkan me-nol-kan nilai 30% *synapse weights* (dropout rate= 0,3) secara random. Hal ini dapat dicapai dengan teknik *masking*, yaitu mengalikan *synapse weights* dengan suatu *mask*.

Ingat kembali ANN secara umum, persamaan XX.XX dimana W adalah *synapse weights*, x adalah input (dalam pembahasan saat ini, dapat merepresentasikan *hidden state* pada suatu *layer*), b adalah *bias* dan f adalah fungsi aktivasi (*non-linear*). Peneliti bisa buat suatu *mask* untuk *synapse weights* seperti pada persamaan XX.XX, dimana p adalah vektor dan pi = [0,1] merepresentasikan *synapse weight* diikutsertakan atau tidak. r% (*dropout rate*) elemen vektor p bernilai 0. Biasanya p diambil dari *bernoulli distribution*. Kemudian, saat *feed forward,* mengganti *synapse weights* menggunakan *mask* seperti pada persamaan XX.XX. Saat menghitung *backpropagation*, turunan fungsi juga mengikut sertakan *mask* (*gradient di-mask*). Teknik *regularization* dan *dropout* sudah menjadi metode yang cukup "standar" dan diaplikasikan pada berbagai macam arsitektur.

## Convolutional Neural Network

Kemampuan utama *convolutional neural network* (CNN) adalah arsitektur yang mampu mengenali informasi prediktif suatu objek (gambar, teks, potongan suara, dsb) walaupun objek tersebut dapat diposisikan dimana saja pada input. Kontribusi CNN adalah pada *convolution* dan *pooling* *layer*. *Convolution* bekerja dengan prinsip *sliding window* dan *weight sharing* (mengurangi kompleksitas perhitungan). *Pooling layer* berguna untuk merangkum informasi informatif yang dihasilkan oleh suatu *convolution* (mengurangi dimensi). Pada ujung akhir CNN, kita lewatkan satu vektor hasil beberapa operasi *convolution* dan *pooling* pada *multilayer perceptron (feed-forward neural network)*, dikenal juga sebagai *fully connected layer*, untuk melakukan suatu pekerjaan, e.g., klasifikasi. Pada umumnya CNN tidak berdiri sendiri, dalam artian CNN biasanya digunakan (dikombinasikan) pada arsitektur yang lebih besar.

### Activation Layer

*Activation layer* pada umumnya merupakan sebuah fungsi aktivasi di bagian atas sebuah *output layer*. Tujuan utama dari penggunaan fungsi aktivasi ini adalah untuk membentuk *non-linearity* (ketidaklinieran) pada *neural network*. Tanpa adanya fungsi aktivasi, *neural network* hanya akan melakukan transformasi linier dari input ke output. Secara matematis, fungsi aktivasi ditulis sebagai berikut.

xl = f(xl−1)

Beberapa jenis fungsi aktivasi yang umum digunakan adalah RelU *(Rectified-Linear Unit)*, Sigmoid dan TanH *(Hyperbolic Tangent)*.

**Sigmoid**

Fungsi Sigmoid akan memetakan input ke interval [0, 1]. Fungsi sigmoid didefinisikan sebagai berikut.

**TanH**

Fungsi TanH atau *hyperbolic tangent function* merupakan fungsi transformasi linier dari sigmoid ke interval [-1, 1]. Fungsi TanH dapat didefinisikan sebagai berikut.

**RelU**

Permasalahan yang ada pada fungsi sigmoid dan tanh yaitu output yang mudah jenuh ke angka 0 atau 1 pada sigmoid dan +1 dan -1 pada tanh. Output yang jenuh akan membuat hilangnya gradien dan menyebabkan turunnya kecepatan training dan memungkinkan untuk menjebak model pada *area local minimum*.

RelU diperkenalkan pada tahun 2010 untuk meningkatkan kecepatan *konvergensi* dari *training neural networks* (V. Nair, 2010). RelU memiliki kelebihan tidak akan jenuh pada suatu nilai dan memungkinkan untuk melakukan *training neural network* berukuran besar layaknya CNN. Relu didefinisikan sebagai berikut.

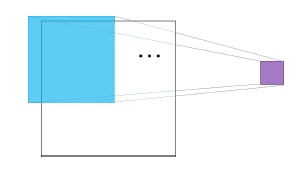
f(x) = max(0, x)

Pada CNN, layer output pada umumnya merupakan *feature maps* dimensi dua. Pada persamaan XX.XX , x adalah matrix dan max adalah elemen *twise* yang diterapkan pada setiap elemen matriks. Varian lain dari RelU yaitu LeakyRelU memungkinkan adanya nilai yang kecil, bukan nol meski unit sedang tidak dalam keadaan aktif. Leaky RelU didefinisikan sebagai berikut.

### Convolution

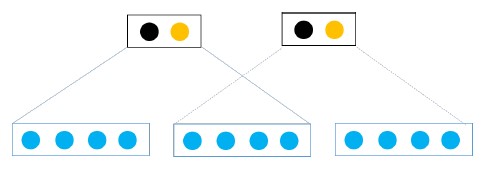
Motivasi CNN adalah untuk mampu mengenali aspek yang informatif pada regional tertentu (lokal). Dibanding meng-*copy* mesin pembelajaran beberapa kali untuk mengenali objek pada banyak regional, ide lebih baik adalah untuk menggunakan *sliding window*. Setiap operasi pada window bertujuan untuk mencari aspek lokal yang paling informatif.

Ilustrasi diberikan oleh Gambar. XX.X. Warna biru merepresentasikan satu *window*, kemudian kotak ungu merepresentasikan aspek lokal paling informatif (disebut *filter*) yang dikenali oleh *window*. Dengan kata lain, kita mentransformasi suatu window menjadi suatu nilai numerik (*filter* ). Kita juga dapat mentransformasi suatu *window* (regional) menjadi d nilai numerik (d channels, setiap elemen berkorespondensi pada suatu *filter* ). *Window* ini kemudian digeser-geser sebanyak T kali, sehingga akhirnya kita mendapatkan vektor dengan panjang d x T. Keseluruhan operasi ini disebut sebagai *convolution*.

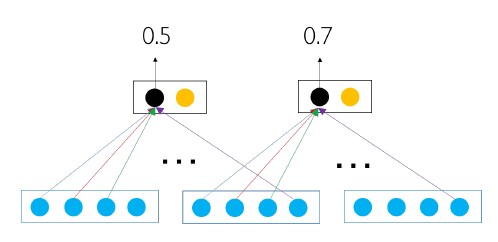


Pada gambar XX.XX menggunakan *window* selebar 2, satu *window* mencakup 2 data; i.e., window1 = (X1, X2), window2 = (X2; X3), … . Untuk suatu input X. Kita juga dapat mempergunakan *stride* sebesar s, yaitu seberapa banyak data yang digeser untuk *window* baru. Contoh yang diberikan memiliki *stride* sebesar satu. Apabila memiliki *stride*= 2, maka akan menggeser sebanyak 2 data setiap langkah, i.e., window1 = (X1, X2), window2 = (X3, X4), … .

Berikut contoh dalam bentuk 1-D pada gambar XX.X. Warna biru merepresentasikan *feature vector* (*regional*) untuk suatu input (e.g., *regional* pada suatu gambar, kata pada kalimat, dsb). Pada contoh ini, setiap 2 input ditransformasi menjadi vektor berdimensi 2 (2-*channels*); menghasilkan vektor berdimensi 4 (2 *window* x 2).

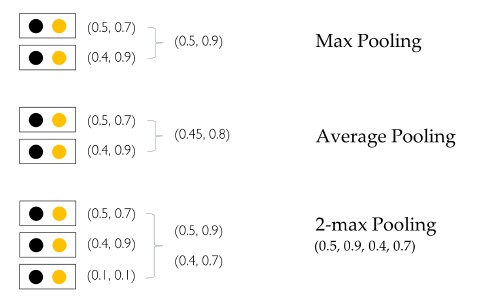


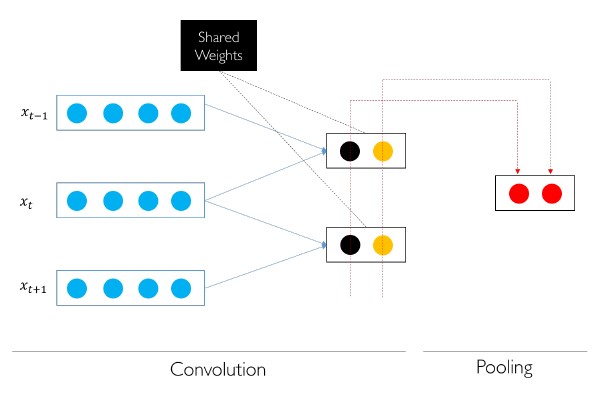
Selain *sliding window* dan *filter*, *convolutional layer* juga mengadopsi prinsip *weight sharing*. Artinya, *synapse weights* untuk suatu filter adalah sama walau filter tersebut dipergunakan untuk berbagai *window*. Sebagai ilustrasi, perhatikan Gambar. XX.X, warna yang sama pada *synapse weights* menunjukkan *synapse weights* bersangkutan memiliki nilai (*weight*) yang sama. Tidak hanya pada *filter* hitam, hal serupa juga terjadi pada *filter* berwarna oranye (i.e., *filter* berwarnya oranye juga memenuhi prinsip *weight sharing*). Walaupun memiliki konfigurasi bobot *synapse weights* yang sama, unit dapat menghasilkan output yang berbeda untuk input yang berbeda. Konsep *weight sharing* ini sesuai dengan pernyataan bahwa konfgurasi parameter untuk mengenali karakteristik informatif untuk satu objek bernilai sama walau pada lokasi yang berbeda. Dengan *weight sharing*, parameter neural network juga menjadi lebih sedikit dibanding menggunakan *multilayer perceptron (feed-forward neural network)*.



### Pooling

Pada tahap convolution, setiap k-sized window diubah menjadi satu vektor berdimensi d (yang dapat disusun menjadi matriks D). Semua vektor yang dihasilkan pada tahap sebelumnya dikombinasikan (pooled) menjadi satu vektor c. Ide utamanya adalah mengekstrak informasi paling informatif (semacam meringkas). Ada beberapa teknik *pooling*, diantaranya: *max pooling*, *average pooling*, dan *K-max pooling*; diilustrasikan pada Gambar. XX.X. *Max pooling* mencari nilai maksimum untuk setiap dimensi vektor. *Average pooling* mencari nilai rata-rata tiap dimensi. *K-max* pooling mencari K nilai terbesar untuk setiap dimensinya (kemudian hasilnya digabungkan). Gabungan operasi *convolution* dan *pooling* secara konseptual diilustrasikan pada Gambar. XX.X.





Setelah melewati berbagai operasi *convolution* dan *pooling*, pembuat akan memiliki satu vektor yang kemudian dilewatkan pada *multilayer perceptron* *(fully connected)* untuk melakukan sesuatu (tergantung permasalahan), misal klasifikasi gambar, klasifikasi sentimen, dsb (Ilustrasi pada Gambar. XX.X).

