YAYASAN PRIMA AGUS TEKNIK

# ALGORITMA dan TEORI KOMPUTASI TERUKUR dalam Pembelajaran Mesin (Machine Learning)

Dr. Joseph Teguh Santoso, M.Kom.

**Algoritma dan Teori Komputasi Terukur dalam Pembelajaran Mesin (Machine Learning)**

**Penulis :**
Dr. Joseph Teguh Santoso, S.Kom., M.Kom

**ISBN : 9 786238 120406**

**Editor :**
Muhammad Sholikan, M.Kom

**Penyunting :**
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

**Desain Sampul dan Tata Letak :**
Irdha Yunianto, S.Ds., M.Kom

**Penebit** :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

**Redaksi :**
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

**Distributor Tunggal :**
**Universitas STEKOM**
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

# KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa, yang telah memberikan kekuatan, ketekunan dan kesabaran sehingga buku ini akhirnya dapat diselesaikan tepat waktu tanpa kurang halangan apa pun. Penulisan buku yang berjudul ***"Alogaritma Dan Teori Komputasi Terukur Dari Pembelajaran Mesin (Machine Learning)"***. Teknlogi *Machine Learning* atau Pembelajaran Mesin adalah salahsatu cabang dari Artificial Intelligence yang sangat menarik, karena kecerdasan buatan ini dapat diaplikasikan dalam berbagai aspek kehidupan manusia. Dalam buku ini akan membahas algoritma dan Komputasi menggunakan *Machine Learning*.

Buku ini terdiri dari 7 bab. Bab pertama buku ini akan membahas tentang pengantar komputasi dan pembelajaran mesin. Bab 2 buku ini akan memberikan gambaran tentang Permodelan grafis hibrida yang akan memberikan contoh permodelan ini. Bab ke 3 buku ini akan membahas tentang wawasan DL (*Deep Learning*) dan mekanisme maupun Pembelajaran jauh terdistribusi. Bab 4 pada buku ini akan membahas tentang merancang algoritma Wake dan Sleep dalam Variasi AutoEncoder, serta akan menjelaskan tentang permodelan Variabel Laten dan Variabel Tampak. Bab 5 dalam buku ini akan menerangkan tentang lapisan Komputasi dan Pembelajaran Mendalam sebagai grafik aliran data. Bab 6 buku ini akan menjelaskan pembelajaran mendalam terdistribusi pada GPU (*Graphics Processing Unit*) tentang mengatasi kemacetan dan menangan memori yang terbatas. Bab terakhir buku ini akan menjelaskan tentang penyimpanan parameter dan mempelajari paradigma komunikasi menggunakan pembelajaran mendalam.

Penulis mengucapkan terimakasih kepada pihak yang telah membantu sehingga dapat diterbitkannya tulisan ini. penulis juga merasa bahwa buku ini jauh dari sempurna, oleh karena itu segala masukan baik berupa saran maupun kritik yang membangun sangat diharapkan. Akhir kata semoga tulisan ini dapat bermanfaat bagi siapa saja yang ingin belajar dan mendalaminya.

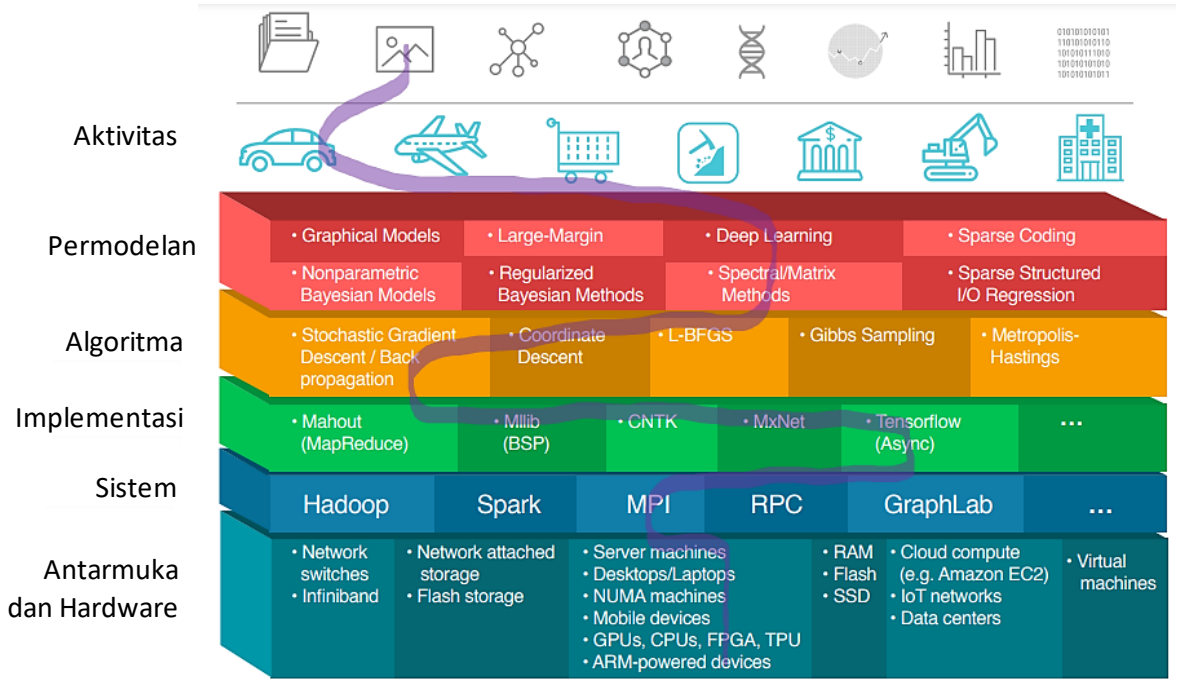
Semarang, Mei 2023
Penulis

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

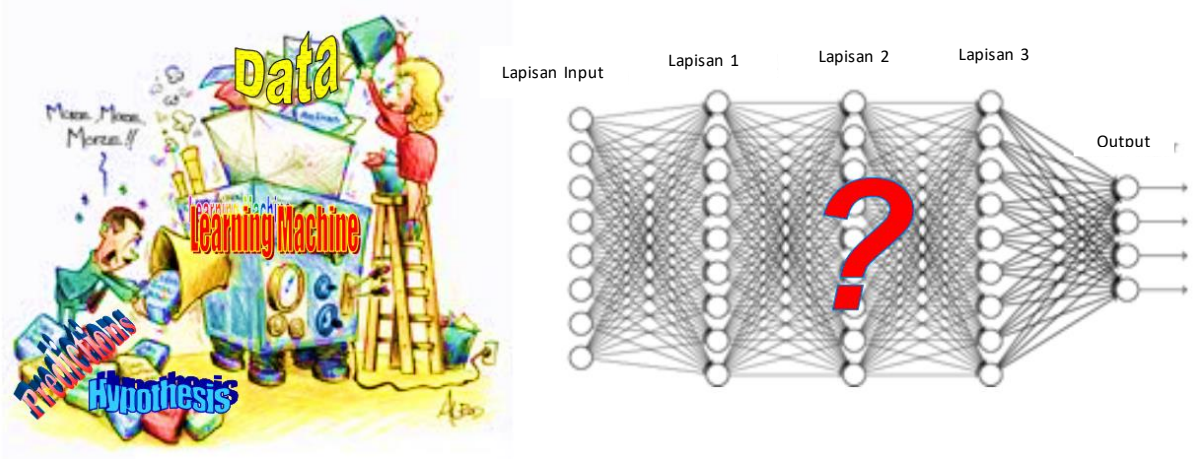
# DAFTAR ISI

# BAB I
# KOMPUTASI DAN *MACHINE LEARNING*

Segala sesuatu dimasa sekarang berbasis menggunakan mesin. Mulai dari melakukan hal yang ringan seperti menghitung pembayaran hingga hal besar dalam skala besar pula seperti perindustran. Sebelum kita belajar lebih jauh, marilah kita simak gambar dibawah ini. Dari gambar dibawah ini segala kegiatan manusia sudah di bantu mesin yang sudah mampu belajar dengan sendirinya, atau kita sering sebut dengan Kecerdasan Buatan atau *Artificial Intelligence*. Dari menyimpan file yang akan dihubungkan ke Lembaga yang lain semua bisa dilakukan dengan mudah.




**Gambar 1.1** Elemen dalam AI *(Artificial Intelligent)*

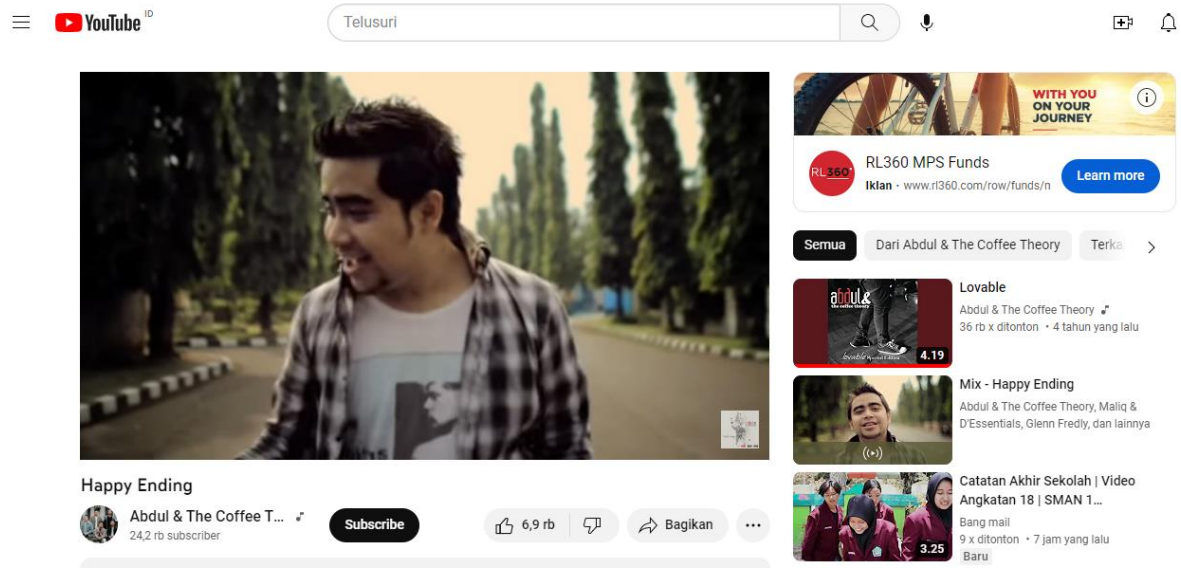



**Gambar 1.2** ML *(Machine Learning)* Vs DL *(Deep Learning)*

Merujuk pada gambar diatas (Gambar 1.2), menjelaskan tentang perbedaan antara ML *(Machine Learning)* dan DL *(Deep Learning)*. Perbedaan diantaranya adalah pada machine learning hanya mampu mengolah dan menganalisis data yang terstruktur. Sedangkan, deep learning mampu mengolah dan menganalisis data yang tidak terstruktur seperti gambar, video, maupun audio, karena pengolahan data bersifat lapisan demi lapisan. ML dan DL juga memiliki performa dalam penggolahan data yang berbeda, terutama saat jumlah data yang terus meningkat.

## 1.1 PENGANTAR

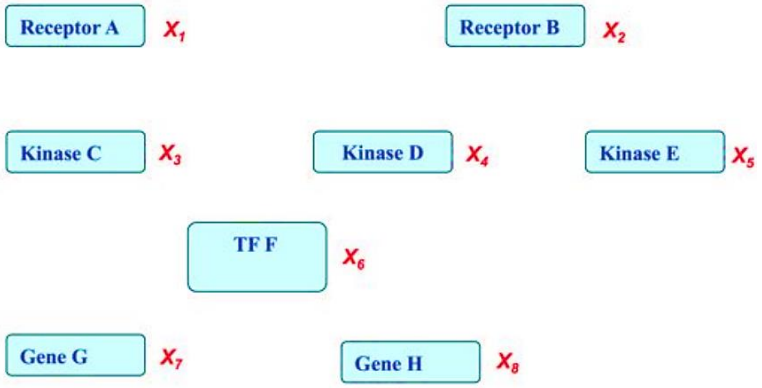
Dalam buku ini akan membahas tentang algoritma dan komputasi Terukur dalam DL. Salah satu perbedaan lainnya dari ML dan DL adalah ML tidak perlu menggunakan program berulang, sedangkan pada DL harus memakai jaringan syaraf tiruan manusia. Output yang dihasilkan pada DL bervarisi dari platform yang digunakan, dapat berupa audio, video, teks sesuai dari platform yang digunakan. Tujuan di ciptakan DL adalah untuk menyelesaikan masalah pada kegiatan sehari-hari dengan data yang terukur, deep learning mampu melakukan unstructured (data tidak terstuktur) menjadi lebih baik dan lebih efisien. Deep Learning mampu menganalisis *big data* yang kompleks dan mampu mengambil keputusan dari program kecerdasan buatan dengan algoritma pengolahan data. Salah satu contoh penerapan deep learning dalam kehidupan sehari hari salah satunya dapat dilihat saat kita membuka youtube atau social media, kita akan menemukan rekomendasi yang kita sukai atau sering kita putar.




**Gambar 1.3** Contoh DL dari youtube tentang video yang anda sukai

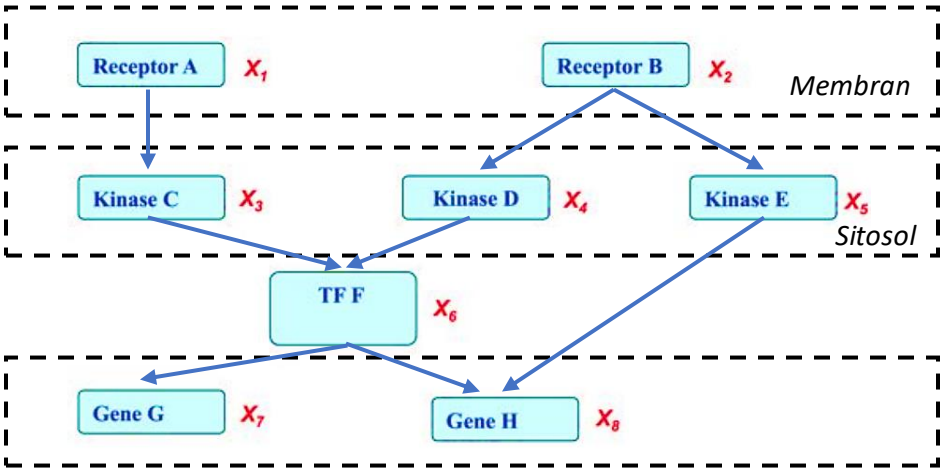
## 1.2 APA ITU MODEL GRAFIS?

Sebelum kita mengenal lebih jauh tentang permodelan grafis, mari kita lihat gambar 1.4 dibawah ini. Gambar dibawah ini merupakan struktur yang merepresentasikan sinyal seluler sebelum disederhanakan menggunakan permodelan grafik.




**Gambar 1.4** Kemungkinan dunia transduksi sinyal seluler

Dari gambar diatas jika kita implementasikan deep learning yang memiliki algoritma pada tiap lapisan-lapisan nya maka hasilnya sebagai berikut:

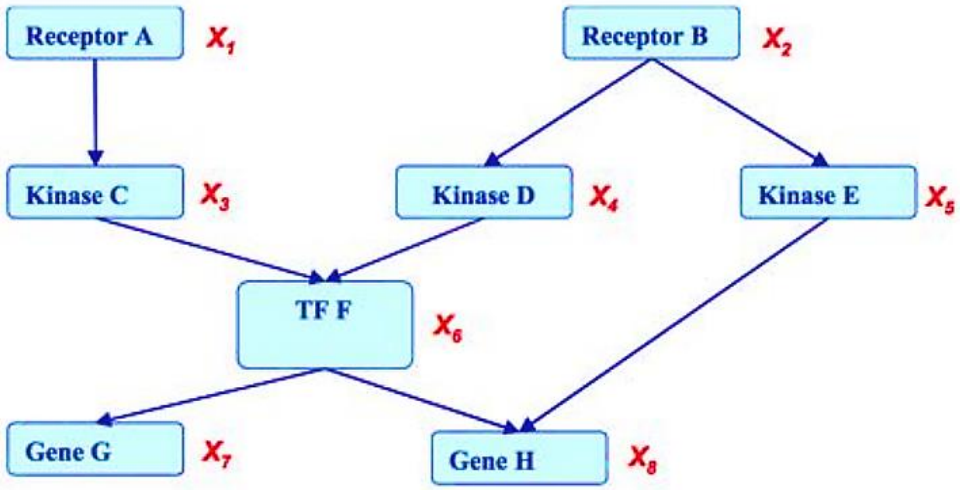



**Gambar 1.5** Pengimplementasian lapisan algoritma dalam *Deep Learning*

## 1.3 MODEL GRAFIS PROBABILISTIK

Model grafis probabilistik / PGM *(Probabilistic Graphical Models)* adalah kerangka kerja yang kuat untuk merepresentasikan domain kompleks menggunakan distribusi probabilitas, dengan banyak aplikasi dalam pembelajaran mesin, visi komputer, pemrosesan bahasa alami, dan biologi komputasi. Model grafis menyatukan teori grafik dan teori probabilitas, dan menyediakan kerangka kerja yang fleksibel untuk memodelkan kumpulan besar variabel acak dengan interaksi yang kompleks. Permodel ini akan memberikan survei komprehensif dan teknik utama yang digunakan untuk membangunnya, membuat prediksi, dan mendukung pengambilan keputusan di bawah ketidakpastian.

Mengacu pada gambar 1.4 Jika $X_o$ bebas bersyarat, maka gabungan dapat difaktorkan menjadi produk dengan persyaratan yang lebih sederhana:



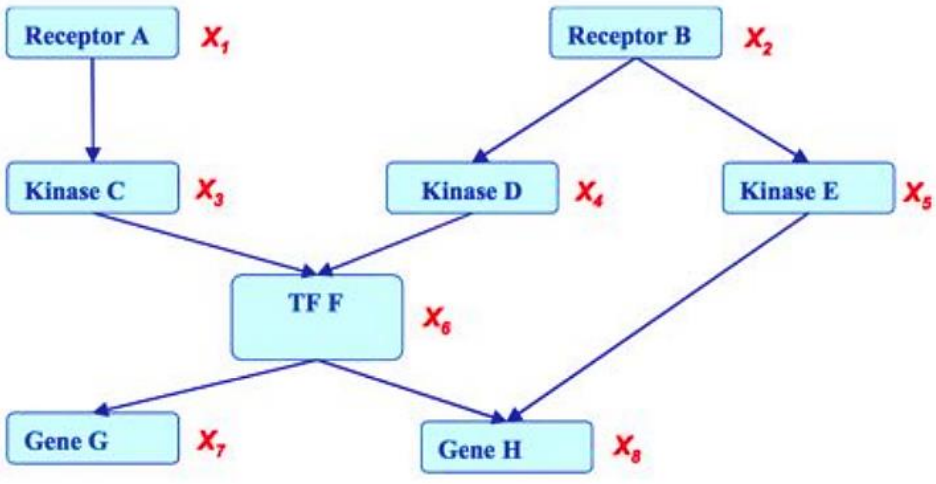

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = P(X_1)P(X_2)P(X_3|X_1)P(X_4|X_2)P(X_5|X_2) P(X_6|X_3, X_4)P(X_7|X_6)P(X_8|X_5, X_6)$$

Alasan penggunaan PGM antara lain mudah menggabungkan pengetahuan domain dan struktur kausal (logis). Alasan lain nya adalah pengurangan biaya perwakilan yang signifikan ($2^8$ dikurangi menjadi 18).

***Jenis Pemodelan Grafik***

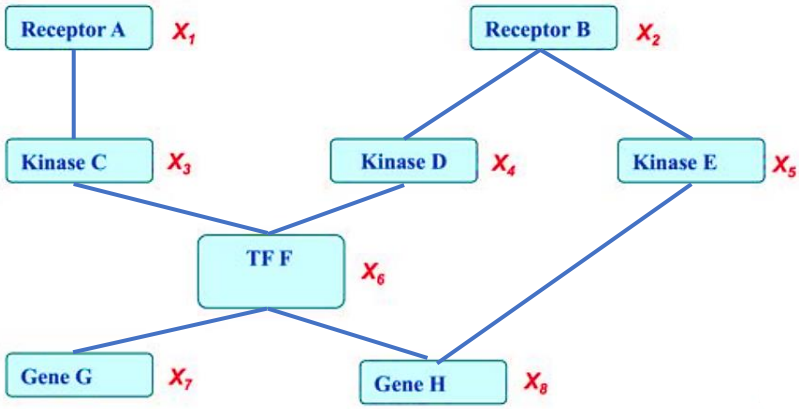
1. Tepi terarah memberikan makna kausal pada hubungan (*Bayesian Networks* atau *Directed Graphical Models*)

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = P(X_1)P(X_2)P(X_3|X_1)P(X_4|X_2)P(X_5|X_2) P(X_6|X_3, X_4)P(X_7|X_6)P(X_8|X_5, X_6)$$




2. Sisi tak berarah mewakili korelasi antar variabel (Bidang Acak Markov atau Model Grafis Tak Berarah)

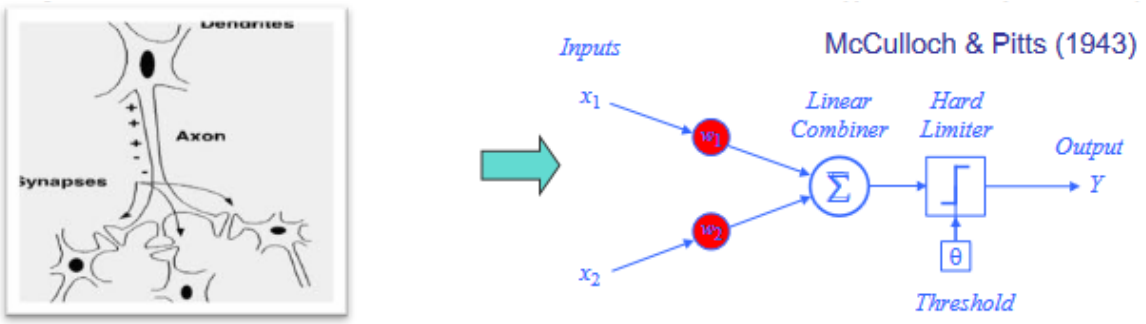
$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) = \frac{1}{Z}\exp\{E(X_1) + E(X_2) + E(X_1, X_3) + E(X_2, X_4) + E(X_5, X_2) + E(X_3, X_4, X_6) + E(X_6, X_7) + E(X_5, X_6, X_8)\}$$




## 1.4 JARINGAN SYARAF

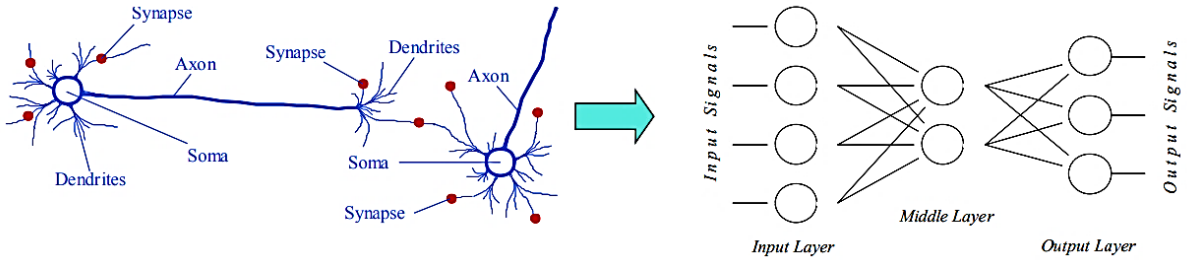
Neural network (NN) membantu mengelompokkan dan mengklasifikasikan data yang disimpan dan dikelola. Neural network (NN) dapat memahami data yang tidak berlabel dan tidak terstruktur, kemudian mengelompokkannya berdasarkan *similarity* data input dengan data pelatihan.

Perceptron adalah jenis neural network tertua dan dikenal sebagai salah satu jenis algoritma machine learning. Pada awalnya perceptron merupakan mesin untuk mengenali gambar. Cara kerja ini terinspirasi dari fungsi persepsi manusia dalam melihat dan mengenali objek. Perceptron adalah neural network yang hanya terdiri dari satu lapisan node, dan hanya memahami hubungan linear antara data input dan output yang disediakan. Yang akan diperlihatkan pada gambar dibawah ini



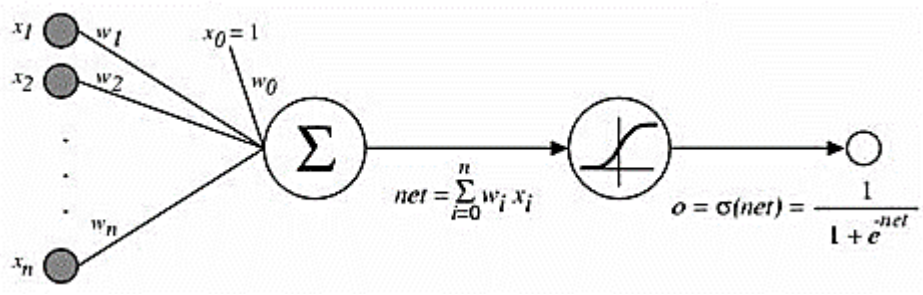

**Gambar 1.6** Neuron biologis ke neuron buatan (perceptron)

Jaringan syaraf yang memiliki banyak lapisan node sehingga mampu memahami pola data yang lebih kompleks. Kemampuan itu bisa ditemukan di *multi-layer perceptrons* (MLP).




**Gambar 1.7** Jaringan saraf biologis ke jaringan saraf tiruan

***Algoritma Pembelajaran Perceptron***




Ingat rumus dari fungsi sigmoid

$$\frac{d\sigma}{dt} = \sigma(1-\sigma)$$

Pertimbangkan masalah regresi f: X→Y, untuk skalar

$$Y:\ y = f(x) + \epsilon$$

Sedangkan rumus memaksimalkan kemungkinan data bersyarat

$$\vec{w} = \arg\max_{\vec{w}} \ln \prod_i P(y_i|x_i;\vec{w})$$

Dimana

$$\vec{w} = \arg\min_{\vec{w}} \sum_i \frac{1}{2}(y_i - \hat{f}(x_i;\vec{w}))^2$$

$X_o$ = Input
$t_d$ = Output Target
$o_d$ = Output yang diamati
$w_i$ = berat i

$$\begin{aligned}
\frac{\partial E_D[\vec{w}])}{\partial w_j} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_t (t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\
&= \sum_d (t_d - o_d)\left(-\frac{\partial o_d}{\partial w_i}\right) \\
&= -\sum_d (t_d - o_d)\frac{\partial o_d}{\partial net_i}\frac{\partial net_d}{\partial w_i} \\
&= -\sum_d (t_d - o_d)o_d(1-o_d)x_d^i
\end{aligned}$$

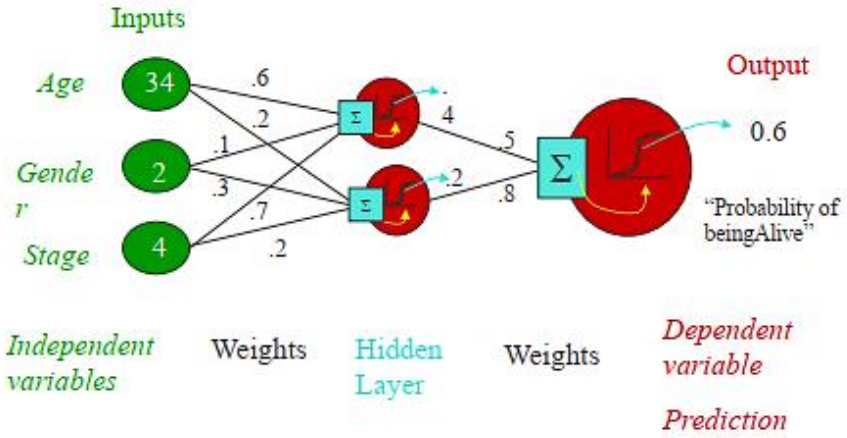
Mode Tambahan:
Lakukan sampai Konvergen:
Untuk masing masing contoh *d* di *D*
1. Hitung Gradient $\nabla E_d\,[w]$
2. $\vec{w} = \vec{w} - \eta\,\nabla Ed[\vec{w}]$

Dimana:
$\nabla Ed[\vec{w}] = -(t_d - o_d)o_d(1-o_d)\vec{x_d}$

Mode Batch:
Lakukan sampai Konvergen:
1. Hitung Gradient $\nabla E_d\,[w]$
2. $\vec{w} = \vec{w} - \eta\,\nabla Ed[\vec{w}]$

**Model Jaringan Neural**

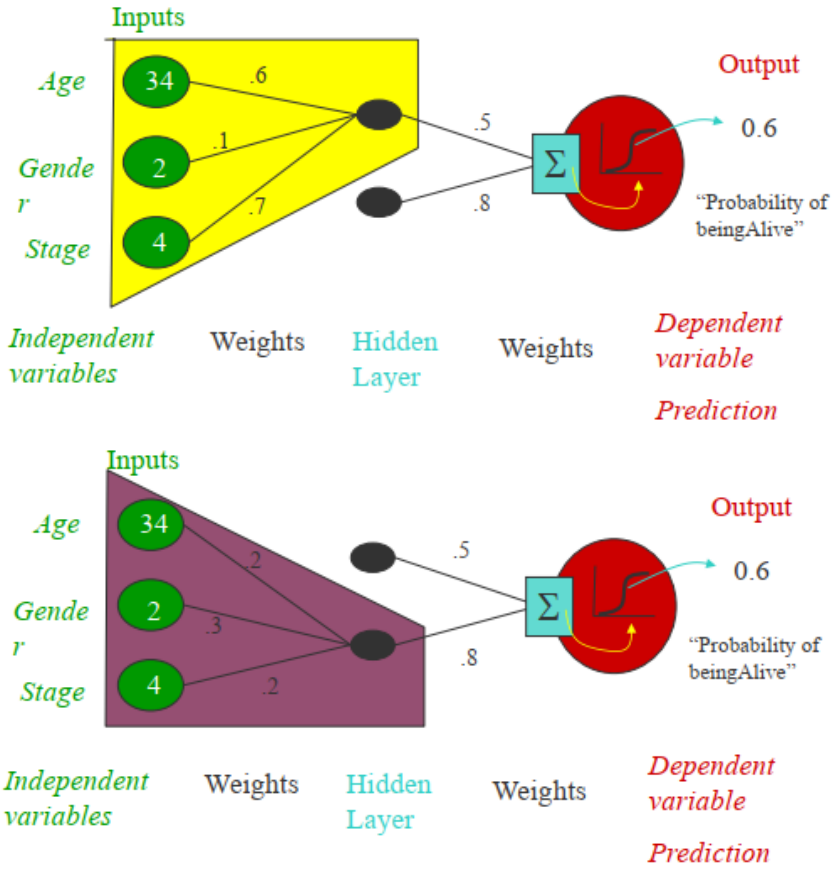
Mode jaringan ini terinpirasi dari syaraf jaringan otak manusia yang terdiri dari impuls atau inputan yang akan diolah pada masing-masing layer. Salah satu contoh penggunaan ini adalah sistem prediksi hidup manusia yang terkena kangker. Untuk masing-masing inputan memiliki bobot yang akan dikalkulasi sebagai output prediksi.

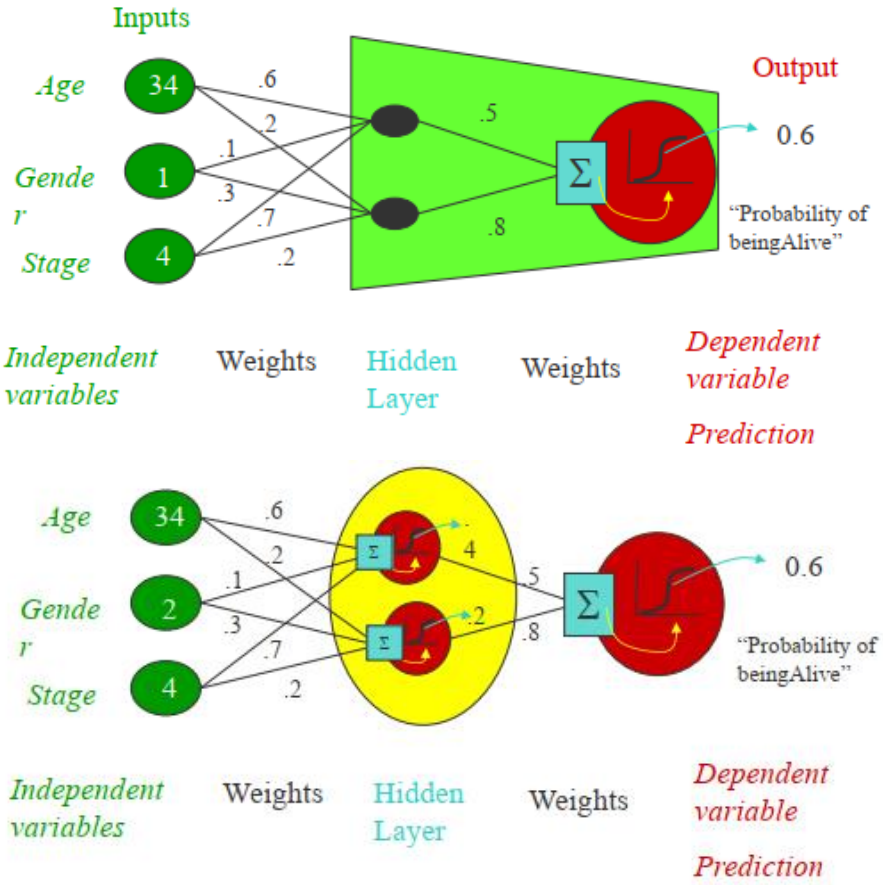



**Gambar 1.8** Bagan permodelan jaringan terstruktur sistem prediksi kangker

**Model logistik gabungan**

Algoritma pada permodelan ini terletak pada letak algoritma tersebut diimplikasikan pada suatu sistem. Gambar dibawah ini akan memberikan kita contoh dari permodelan logistic gabungan (bagian bangun ruang yang berwarna).

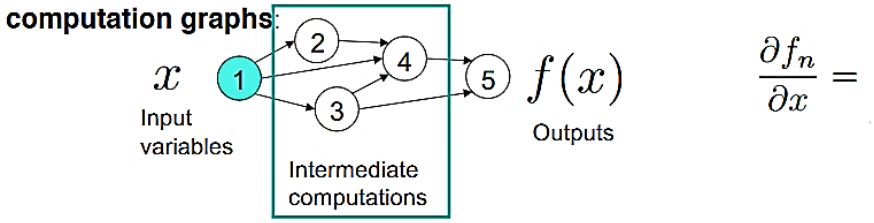


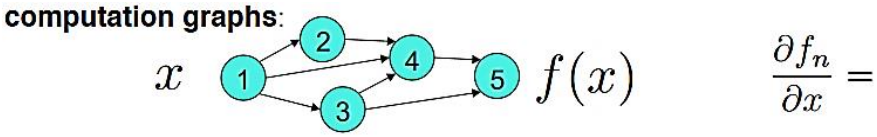



**Gambar 1.8** Model Logistik Gabungan

## 1.5 *BACKPROPAGATION:* DIFERENSIASI MODE TERBALIK

Backpropagation merupakan salah satu bagian dari Neural Network. Backpropagation adalah metode pembelajaran yang terawasi (supervised learning), yang berarti memiliki target yang akan dicari. Ciri Backpropagation yaitu minimalisir error pada output yang akan dihasilkan oleh jaringan. Dalam metode ini, biasanya menggunakan jaringan multilayer. Jaringan saraf tiruan ini tidak lebih dari komposisi fungsional kompleks yang dapat diwakili oleh grafik perhitungan:



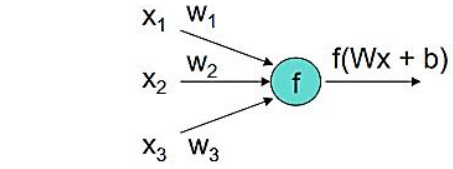



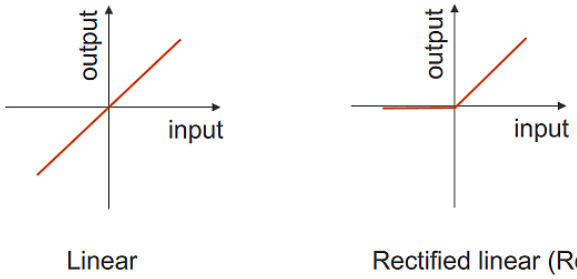
Dengan menerapkan aturan rantai dan menggunakan akumulasi terbalik, kita dapatkan

$$\frac{\partial f_n}{\partial x} = \sum_{i_1 \in \pi(n)} \frac{\partial f_n}{\partial f_{i_1}} \frac{\partial f_{i_1}}{\partial x} = \sum_{i_1 \in \pi(n)} \frac{\partial f_n}{\partial f_{i_1}} \sum_{i_2 \in \pi(i_1)} \frac{\partial f_{i_1}}{\partial f_{i_2}} \frac{\partial f_{i_1}}{\partial x} = \dots$$

Dalam buku ini kita akan mempelajari algoritma Blok bangunan modern dari jaringan dalam

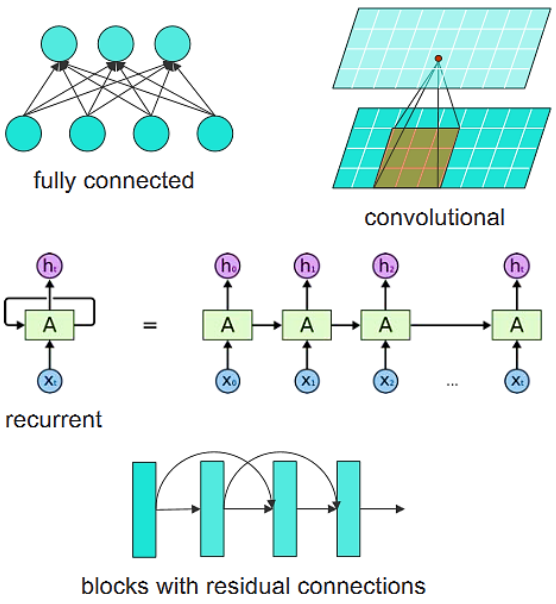
- Fungsi aktivasi
- Linear dan ReLU
- Sigmoid dan tanh
- Dll.



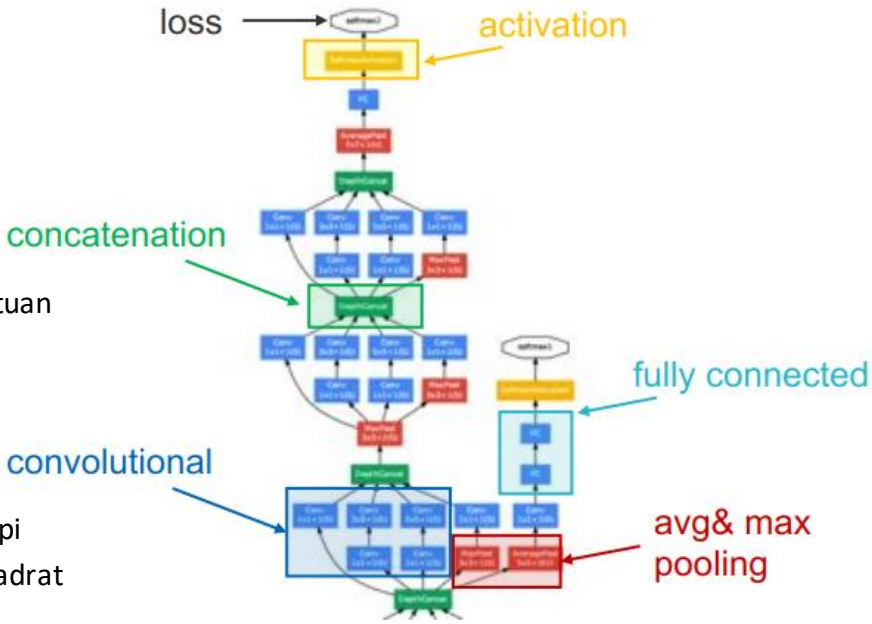

**Blok bangunan modern dari jaringan dalam**

- Fungsi aktivasi
  - Linear dan ReLU
  - Sigmoid dan tanh
  - Dll.
- Lapisan
  - Sepenuhnya terhubung
  - Convolutional & penyatuan
  - Berulang
  - ResNet
  - Dll.



**Contoh lain dari implementasi dari Blok bangunan modern dari jaringan dalam**

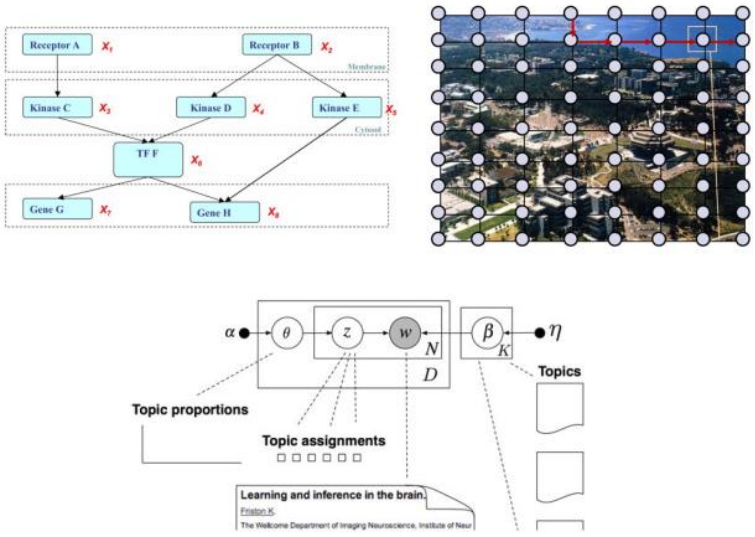
- Fungsi aktivasi
  - Linear dan ReLU
  - Sigmoid dan tanh
  - Dll.
- Lapisan
  - Sepenuhnya terhubung
  - Convolutional & penyatuan
  - Berulang
  - ResNet
  - Dll.
- Fungsi kerugian
  - Kehilangan lintas-entropi
  - Rata-rata kesalahan kuadrat
  - Dll.




Dari gambar diatas dapat dilihat Kombinasi yang tidak terstruktur dari blok bangunan dasar. Beberapa fungsi kerugian – prediksi multi-target, pembelajaran transfer, dan banyak lagi. Dengan data yang cukup, arsitektur yang lebih dalam terus meningkat. Pembelajaran representasi: jaringan semakin mempelajari representasi abstrak dari data yang "terurai," yaitu, setuju untuk pemisahan linier.

## 1.6 MODEL GRAFIS VS. JARING DALAM

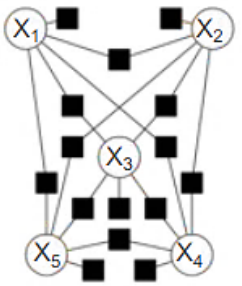
Pada permodelan grafis direpresentasi untuk pengkodean pengetahuan yang bermakna dan ketidakpastian terkait dalam bentuk grafis. Representasi untuk pengkodean pengetahuan yang bermakna dan ketidakpastian terkait dalam bentuk grafis. Pembelajaran dan inferensi didasarkan pada kotak alat yang kaya dari teknik yang dipelajari dengan baik (bergantung pada struktur) (misalnya EM, penyampaian pesan, VI, MCMC, dll.). Grafik mewakili model



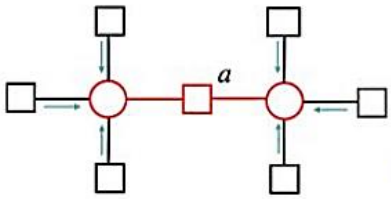

**Gambar 1.9** permodelan grafis direpresentasi

**Keuntungan permodelan grafik**

1. Sarana untuk mensintesis fungsi kerugian global dari struktur lokal, seperti fungsi potensial, fungsi fitur, dll.
2. Sarana untuk merancang algoritme inferensi yang baik dan efisien, misalnya Produk-jumlah, bidang rata-rata, dll.
3. Sarana untuk menginspirasi pendekatan dan hukuman, seperti Structured MF, Tree-approximation, dll.
4. Juga dapat diartikan sarana untuk memantau perilaku teoretis dan empiris serta akurasi inferensi.



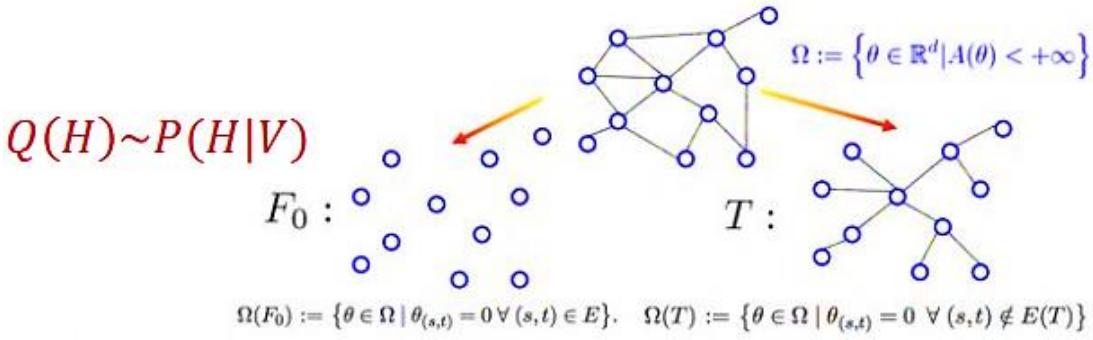

$$\log P(X) = \sum_i \log \phi(x_i) + \sum_{i,j} \log \psi(x_i, x_j)$$




$$m_{i\to a}(x_i) = \prod_{c\in N(i)\setminus a} m_{c\to i}(x_i)$$

$$b_a(X_a) \propto f_a(X_a) \prod_{i\in N(a)} m_{i\to a}(x_i)$$

$$m_{a\to i}(x_i) = \sum_{X_a\setminus x_i} f_a(X_a) \prod_{j\in N(a)\setminus i} m_{j\to a}(x_j)$$

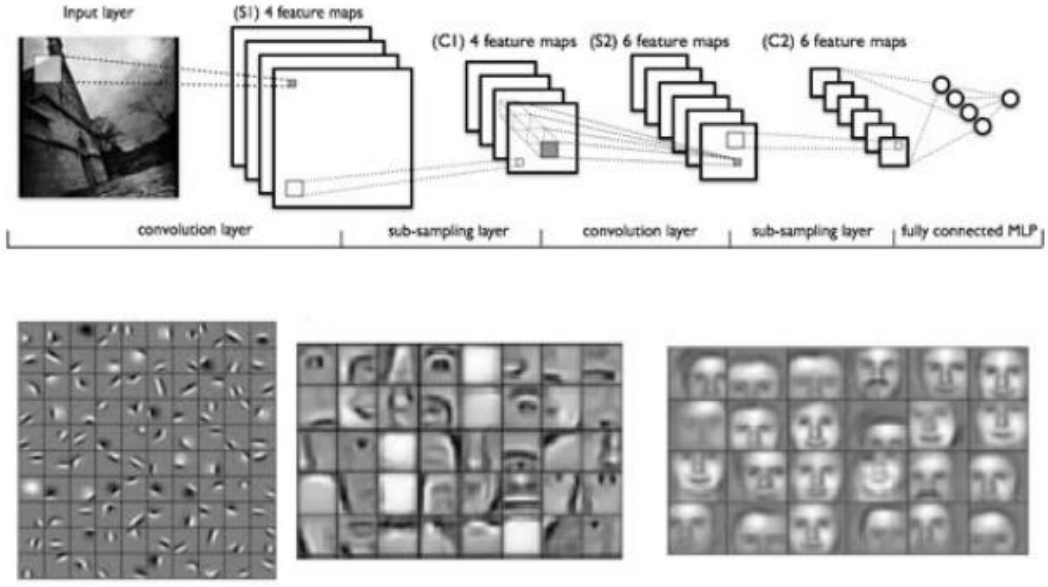



Kerugian permodelan grafik adalah ukuran utama kualitas algoritma pembelajaran dan model yang direpresentasikan dengan algoritma

$$\theta = \text{argmax}_\theta P_\theta(V)$$

**Jaringan saraf yang dalam**

Jaringan ini memrepresentasikan dan memfasilitasi perhitungan dan kinerja pada metrik akhir. Sebagian besar didasarkan pada metode penurunan gradien (alias backpropagation); Inferensi seringkali sepele dan dilakukan melalui "forward pass". Setiap Grafik mewakili perhitungan




**Keuntungan jaringan dalam**

1. Sarana untuk mensintesis hipotesis keputusan kompleks secara konseptualatau sebagai proyeksi dan agregasi bertahap
2. Sarana untuk mengatur operasi komputasi atau pembaruan bertahap dari status laten
3. Sarana untuk merancang langkah-langkah pemrosesan dan modul komputasi Paralelisasi berlapis-lapis
4. Tidak ada utilitas yang jelas dalam mengevaluasi algoritma inferensi DL

**Kerugian Jaringan dalam**

Kurang mampu untuk melakukan operasi-operasi numerik dengan presisi tinggi serta kurang mampu melakukan operasi algoritma aritmatik, operasi logika dan simbolis

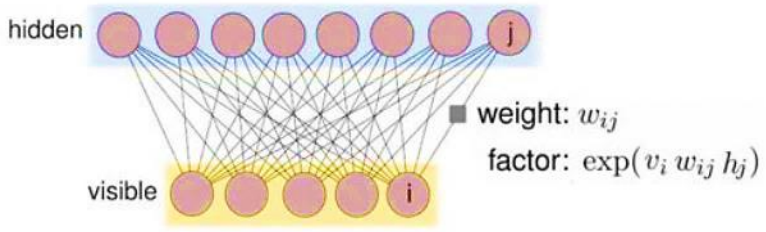
**Tabel 1.1** skema perbedaan Deep Learning dan Permodelan Grafik

| | ***Deep Learning*** | ***Permodelan Grafik*** |
|---|---|---|
| Tujuan empiris: | misalnya, klasifikasi, pembelajaran fitur | misalnya, inferensi variabel laten, transfer pembelajaran |
| Struktur: | Grafis | Grafis |
| Objektif: | Sesuatu yang dikumpulkan dari fungsi lokal | Sesuatu yang dikumpulkan dari fungsi lokal |
| Kosakata: | Neuron, fungsi aktivasi, … | Variabel, fungsi potensial, … |
| Algoritma: | Algoritma inferensi tunggal yang tidak tertandingi<br>–<br>Backpropagation (BP) | Fokus utama penelitian terbuka, banyak algoritme, dan banyak lagi yang akan datang |
| Evaluasi: | Pada skor kotak hitam – kinerja akhir | Di hampir setiap kuantitas menengah |
| Penerapan: | Banyak trik | Lebih atau kurang standar |

| | | |
|---|---|---|
| Eksperimen: | Data masif dan nyata (GT tidak diketahui) | Sederhana, data sering disimulasikan (GT dikenal) © Petuum,Inc. 33 |

## 1.7 MESIN BOLTZMANN

RBM (*Restricted Boltzmann Machines*) adalah bidang acak Markov yang direpresentasikan dengan grafik bi-partit. Algoritma dalam mesin ini adalah emua node dalam satu lapisan/bagian grafik terhubung ke semua yang lain; tidak ada koneksi antar-lapisan




Distribusi bersama:

$$P(v,h) = \frac{1}{Z}\exp\left\{\sum_{i,j} w_{ij}v_i h_i + \sum_i b_i v_i + \sum_j c_j h_j\right\}$$

Log-kemungkinan satu titik data (tidak dapat diamati terpinggirkan):

$$\log L(v) = \log \sum_h \exp\left\{\sum_{i,j} w_{ij}v_i h_i + \sum_i b_i v_i + \sum_j c_j h_j - \log(Z)\right\}$$

Gradien log-kemungkinan w.r.t. parameter model:

$$\frac{\partial}{\partial w_{ij}}\log L(v) = \sum_h P(h|v)\frac{\partial}{\partial w_{ij}}P(v,h) - \sum_{v,h} P(v,h)\frac{\partial}{\partial w_{ij}}P(v,h)$$

di mana kita memiliki rata-rata di *atas posterior* dan *di atas sendi.*

Gradien log-kemungkinan w.r.t. parameter (bentuk alternatif):

$$\frac{\partial}{\partial w_{ij}}\log L(v) = \mathbb{E}_{P(h|v)}\left[\frac{\partial}{\partial w_{ij}}P(v,h)\right] - \mathbb{E}_{P(v,h)}\left[\frac{\partial}{\partial w_{ij}}P(v,h)\right]$$

Kedua ekspektasi dapat didekati melalui sampling, Pengambilan sampel dari *posterior* tepat (faktor RBM di atas $h$ diberikan $v$). Pengambilan sampel dari *Sendi / Sambungan* dilakukan melalui MCMC (misalnya, pengambilan sampel Gibbs).

Dalam literatur jaringan saraf:

- Menghitung suku pertama disebut fase ***klem / wake / positif***
  (jaringan "terjaga" karena mengkondisikan variabel yang terlihat)
- Menghitung suku kedua disebut fase ***unclamped / sleep / free / negative***
  (jaringan "tertidur" karena mengambil sampel variabel yang terlihat dari sambungan; secara metaforis, "memimpikan" input yang terlihat)

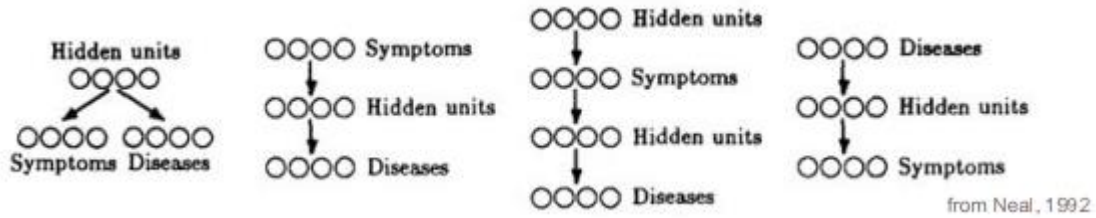
Pembelajaran dilakukan dengan mengoptimalkan log-likelihood model untuk data tertentu melalui stochastic gradient descent (SGD). Estimasi suku kedua (fase negatif) sangat bergantung pada sifat pencampuran rantai Markov. Hal ini sering menyebabkan konvergensi lambat dan membutuhkan perhitungan ekstra.

## 1.8 JARINGAN SIGMOID

Jaring sigmoid hanyalah jaringan Bayesian atas variabel biner dengan probabilitas bersyarat yang diwakili oleh fungsi sigmoid:

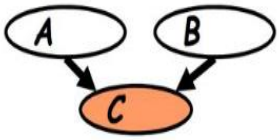
$$P(x_i|\pi(x_i)) = \sigma\left(x_i \sum_{x_j \in \pi(x_i)} w_{ij}x_j\right)$$

Jaringan ini memiliki fungsi matematika yang berfungsi untuk mengubah nilai input menjadi nilai output yang berada dalam rentang 0 hingga 1. Rumus ini sering digunakan dalam algoritma machine learning dan neural network untuk menghasilkan prediksi yang akurat.




**Gambar 1.10** Algoritma jaringan sigmoid

Jaringan Bayesian menunjukkan fenomena yang disebut "Sebab Akibat"




***Catatan:***
Karena "Penjelasan Sebab Akibat", saat kita mengkondisikan pada lapisan yang terlihat di jaringan kepercayaan, semua variabel tersembunyi menjadi dependen.

Jika A berkorelasi dengan C, maka kemungkinan B berkorelasi dengan C berkurang. → A dan B menjadi berkorelasi mengingat C.

Approximated with Gibbs sampling

Conditional distributions:

$$P(S_i = x \mid S_j = s_j : j \neq i) \propto \sigma\left(x^* \sum_{j<i} s_j w_{ij}\right) \prod_{j>i} \sigma\left(s_j^*\left(x w_{ji} + \sum_{k<j, k\neq i} s_k w_{jk}\right)\right)$$

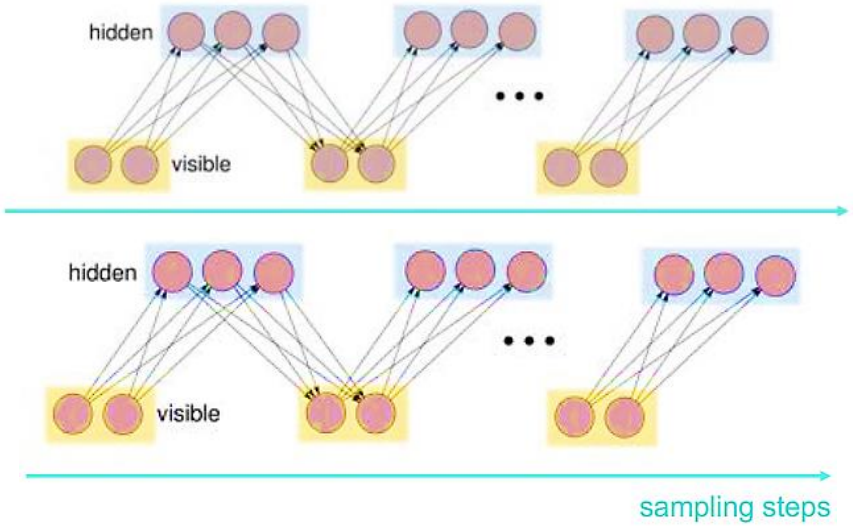
*Konvergensi Sangat rendah,*
Khususnya untuk jaringan yang sangat besar, karena rumit nya efek "Sebab Akibat"

$$\begin{aligned}
\frac{\partial L}{\partial w_{ij}} &= \sum_{\tilde{v}\in\mathcal{T}} \frac{1}{P(\tilde{V}=\tilde{v})} \frac{\partial P(\tilde{V}=\tilde{v})}{\partial w_{ij}} \\
&= \sum_{\tilde{v}\in\mathcal{T}} \frac{1}{P(\tilde{V}=\tilde{v})} \sum_{\tilde{h}} \frac{\partial P(\tilde{S}=\langle\tilde{h},\tilde{v}\rangle)}{\partial w_{ij}} \\
&= \sum_{\tilde{v}\in\mathcal{T}} \sum_{\tilde{h}} P(\tilde{S}=\langle\tilde{h},\tilde{v}\rangle \mid \tilde{V}=\tilde{v}) \cdot \frac{1}{P(\tilde{S}=\langle\tilde{h},\tilde{v}\rangle)} \frac{\partial P(\tilde{S}=\langle\tilde{h},\tilde{v}\rangle)}{\partial w_{ij}} \\
&= \sum_{\tilde{v}\in\mathcal{T}} \sum_{\tilde{s}} P(\tilde{S}=\tilde{s} \mid \tilde{V}=\tilde{v}) \frac{1}{P(\tilde{S}=\tilde{s})} \frac{\partial P(\tilde{S}=\tilde{s})}{\partial w_{ij}} \\
&= \sum_{\tilde{v}\in\mathcal{T}} \sum_{\tilde{s}} P(\tilde{S}=\tilde{s} \mid \tilde{V}=\tilde{v}) \cdot \frac{1}{\sigma\left(s_i^* \sum_{k<i} s_k w_{ik}\right)} \frac{\partial \sigma\left(s_i^* \sum_{k<i} s_k w_{ik}\right)}{\partial w_{ij}} \\
&= \sum_{\tilde{v}\in\mathcal{T}} \sum_{\tilde{s}} P(\tilde{S}=\tilde{s} \mid \tilde{V}=\tilde{v})\, s_i^* s_j \sigma\left(-s_i^* \sum_{k<i} s_k w_{ik}\right).
\end{aligned}$$

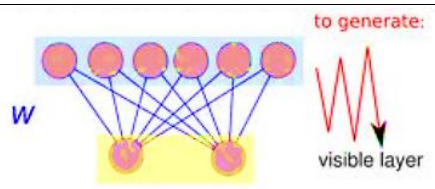
Ingat ekspresi gradien kemungkinan log untuk RBM (*Restricted Boltzmann Machines*) atau mesin Boltzman:

$$\frac{\partial}{\partial w_{ij}} \log L(v) = \mathbb{E}_{P(h|v)}\left[\frac{\partial}{\partial w_{ij}} P(v,h)\right] - \mathbb{E}_{P(v,h)}\left[\frac{\partial}{\partial w_{ij}} P(v,h)\right]$$

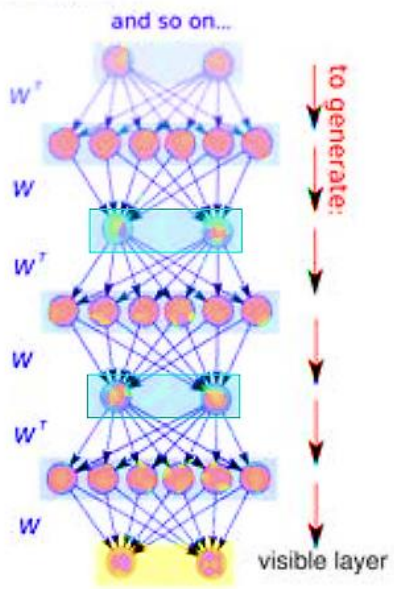
Untuk membuat pembaruan gradien parameter model, kita perlu menghitung ekspektasi melalui pengambilan sampel. Kita dapat mengambil sampel persis dari posterior pada term pertama. Kami menjalankan blok pengambilan sampel Gibbs ke sekitar sampel dari distribusi Bersama.

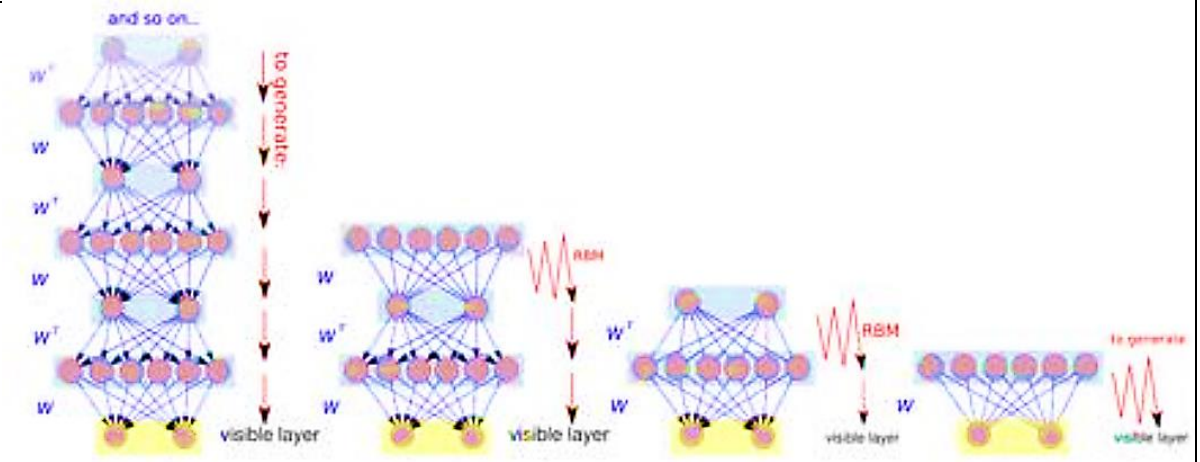



Distribusi bersyarat $P(v|h)$ dan $P(h|v)$ diwakili oleh sigmoid. Dengan demikian, kita dapat menganggap pengambilan sampel Gibbs dari distribusi bersama yang diwakili oleh mesin Boltzman sebagai propagasi top-down dalam jaringan kepercayaan sigmoid yang sangat dalam. Mesin Boltzman setara dengan jaringan kepercayaan yang sangat dalam sehingga didapatkan algoritma seperti dibawah ini:

| | |
|---|---|
| *Mesin Boltzman setara dengan DBN (Deep Belief Network) Tak Hingga.* |   |

*Pengambilan sampel dari sini sama dengan pengambilan sampel dari jaringan di sebelah kanan.*



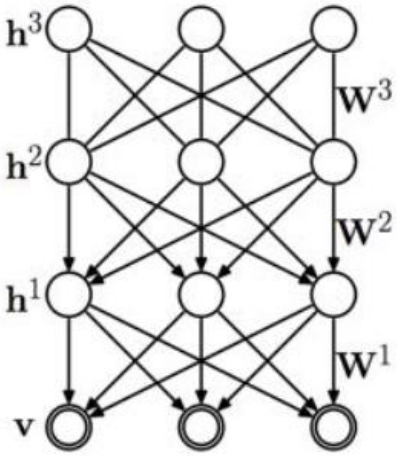




*Saat kita menggunakan Mesin Boltzman, kita benar-benar melatih jaring singkat yang sangat dalam. Hanya saja bobot semua layer diikat dan dihitung. Jika bobotnya "tidak terikat" sampai batas tertentu, kita mendapatkan Deep Belief Network (DBN).*

# BAB II
# PERMODELAN GRAFIS HYBRID

DBN *(Deep Belief Network)* adalah model grafis hibrid atau grafik berantai yang memanfaatkan tumpukan atau stack RBM (Restricted Boltzmann Machines) atau Autoencoder secara berkala. Sedangkan yang dimaksud dengan Autoencoder adalah permodelan jaringan syaraf yang memiliki input dan output yang sama.

Permodelan ini tersusun atas multiple layer dari layer tersembunyi dimana masing-masing lapisan saling terhubung satu samalain, tetapi sambungan (node) intra RBM tidak saling terhubung dengan sabungan dalam lainnya.




**Gambar 2.1** Lapisan Deep Belief Network

Dimana DBN *(Deep Belief Network)* mewakili distribusi probabilitas bersama, yang bisa kita tulis rumus

$$P(v, h^1, h^2, h^3) = P(h^2, h^3)P(h^1|h^2)P(v|h^1)$$

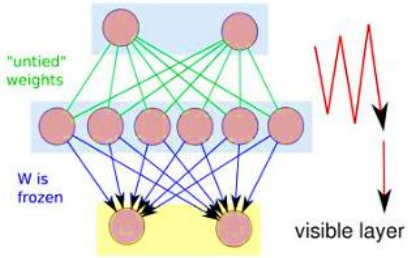
Perhatikan bahwa $P(h^{\&},h^3)$ adalah RBM dan kondisional $P(h^1|h^2)$ dan $P(v|h^1)$ direpresentasikan dalam bentuk sigmoid. Model dilatih dengan mengoptimalkan kemungkinan log untuk log data tertentu $P\ v$

**Tantangan dalam Permodelan ini**

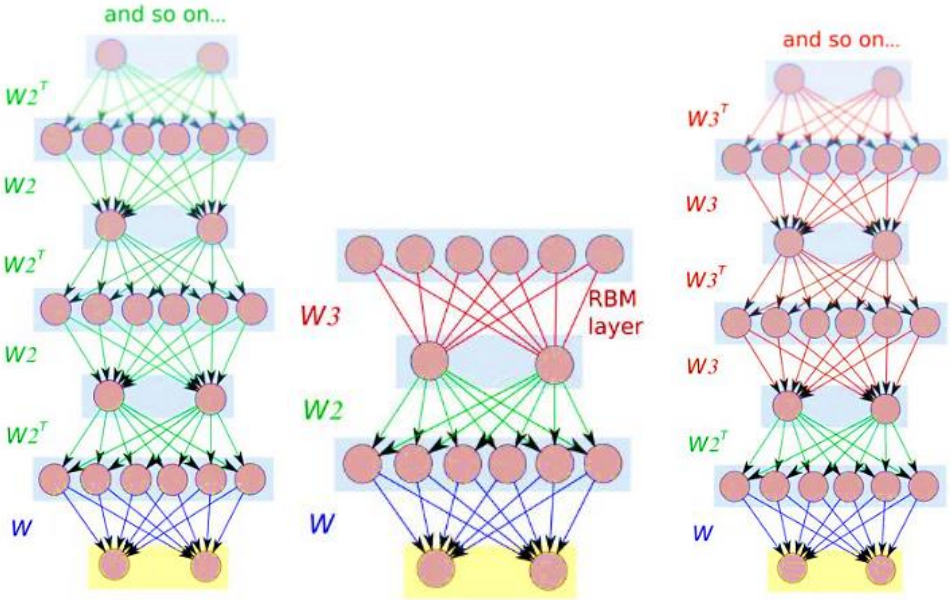
- Inferensi yang tepat dalam DBN bermasalah karena menjelaskan sebab akibat.
- Data Training dilakukan dalam dua tahap:
    - Kebutuhan data yang sangat besat yaitu: data pra-training + penyempurnaan ad-hoc;
    - tidak ada pelatihan bersama yang tepat
- Perkiraan inferensi bersifat feed-forward (bottom-up)

## 2.1 ANTAR LAPISAN

- Data PraTraining dan bekukan RBM ke-1
- Susun RBM lain di atasnya dan latih



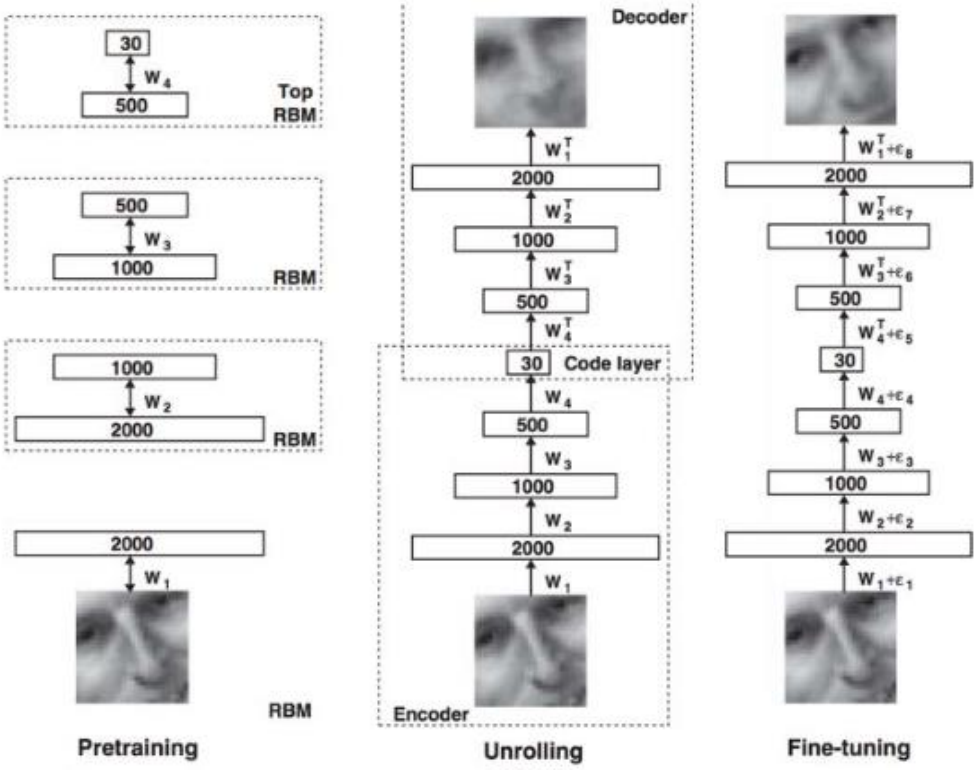

- Bobot pemberat 2+ lapisan tetap diikat
  Tiap lapisan mengulangi prosedur ini: data pra-training dan lepaskan beban lapis demi lapis
- Bobot 3+ lapisan tetap terikat Dan seterusnya
  Dari perspektif pengoptimalan, prosedur ini secara longgar sesuai dengan perkiraan aksen koordinat blok pada kemungkinan log




- Pra-pelatihan cukup ad-hoc dan tidak mungkin menghasilkan model probabilistik yang baik. Namun, lapisan representasi mungkin berguna untuk beberapa tugas dilapisan lainnya. Kita dapat “menyempurnakan” lebih lanjut DBN terlatih untuk beberapa tugas lain

Pengaturan A: Pembelajaran tanpa pengawasan (DBN → autoencoder)

1. Pra-training tumpukan RBM (*Restricted Boltzmann Machine*) dengan cara menyeluruh.
2. "Buka gulungan" RBM untuk membuat penyandi otomatis
3. Sempurnakan parameter dengan mengoptimalkan kesalahan rekonstruksi

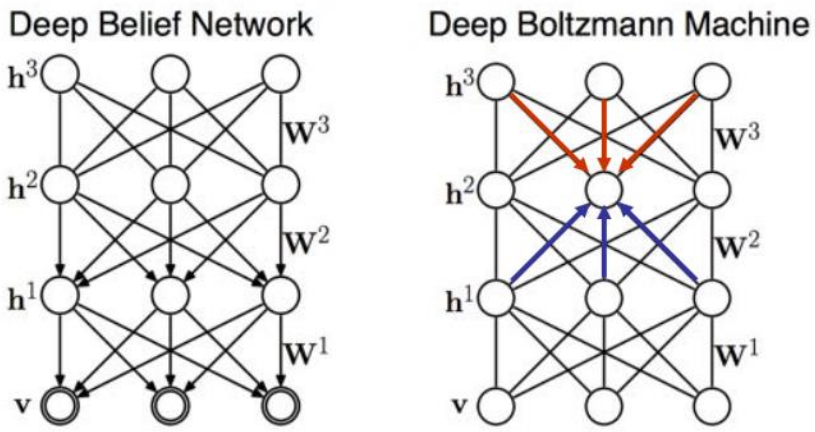



- Pra-training cukup ad-hoc dan tidak mungkin menghasilkan model probabilistik yang baik
- Namun, lapisan representasi mungkin berguna untuk beberapa tugas hilir lainnya!
- Kita selanjutnya dapat "menyempurnakan" DBN yang telah dilatih sebelumnya untuk beberapa tugas lainnya

Pengaturan B: Pembelajaran yang diawasi (DBN → classifier)

1. Data Pra-training berupa tumpukan RBM dengan cara penuh.
2. "Buka gulungan" RBM untuk membuat pengklasifikasi feedforward
3. Sempurnakan parameter dengan mengoptimalkan kesalahan rekonstruksi

**DBN (Deep Belief Nets) Dan DBM (Deep Boltzmann Machine)**



Perbedaan dalam arsitektur antara DBN dan DBM adalah bahwa dalam DBN, dua lapisan teratas mengikuti model grafis tidak terarah dan lapisan bawah membentuk model generatif terarah, namun dalam DBM memiliki banyak lapisan unit tersembunyi di mana unit yang ditempatkan di lapisan bernomor ganjil tidak bergantung pada lapisan bernomor genap sehingga inferensi tidak dapat dipecahkan. Seperti DBN, pretraining lapis demi lapis digunakan oleh DBM untuk menginisialisasi bobot jaringan yang membantu menghindari minima lokal yang buruk.

Seperti di DBN, DBM menggabungkan bidang acak Markov untuk pra-pelatihan berlapis untuk data besar tanpa label dan kemudian memberikan umpan balik dari lapisan atas ke lapisan belakang. Dengan menerapkan metode backpropagation, algoritma pelatihan disesuaikan. Proses pelatihan di DBM perlu diadaptasi untuk menentukan informasi pelatihan, inisialisasi bobot, dan parameter penyesuaian. Diamati dari DBM bahwa kendala kompleksitas waktu akan terjadi ketika mengatur parameter secara optimal. Metode optimisasi pemusatan membuat mekanisme pembelajaran lebih stabil dan juga untuk DBM menengah untuk tujuan merancang model yang generatif, lebih cepat, dan diskriminatif.

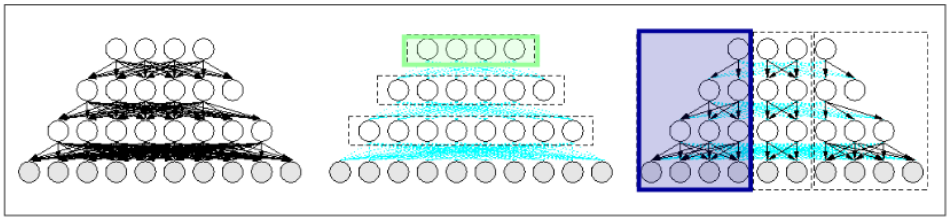
- DBN adalah model grafis hibrid (grafik berantai):
  - Inferensi dalam DBN bermasalah karena menjelaskan efek jauh
  - Data pra-pelatihan sangatbesar + penyempurnaan ad-hoc; tidak ada pelatihan bersama yang tepat
  - Perkiraan inferensi bersifat feed-forward
- DBM adalah model yang sepenuhnya tidak diarahkan:
  - Bisa dilatih sama seperti RBM melalui MCMC *(Markov Chain Monte Carlo)*
  - Gunakan perkiraan variasi distribusi data untuk pelatihan yang lebih cepat
  - Demikian pula, dapat digunakan untuk menginisialisasi jaringan lain untuk tugas hilir

Beberapa hal penting yang perlu diperhatikan tentang semua model ini:

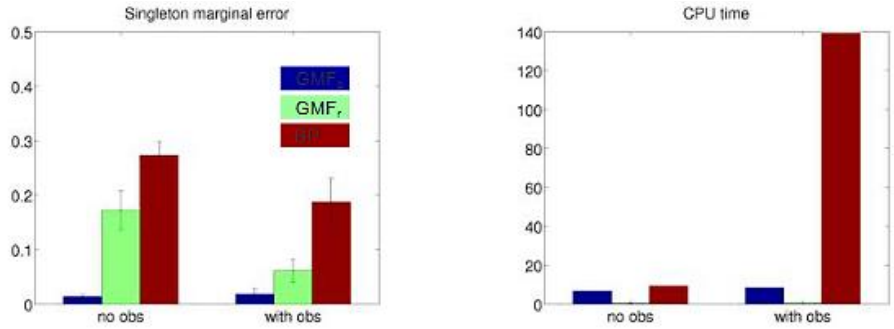
- Tujuan utama dari model generatif mendalam adalah untuk mewakili distribusi variabel yang dapat diamati. Menambahkan lapisan variabel tersembunyi memungkinkan untuk mewakili distribusi yang semakin kompleks.
- Variabel tersembunyi adalah elemen sekunder (tambahan) yang digunakan untuk memfasilitasi pembelajaran ketergantungan kompleks antara yang dapat diamati.
- Pelatihan model bersifat ad-hoc, tetapi yang penting adalah kualitas representasi tersembunyi yang dipelajari.
- Representasi dinilai berdasarkan kegunaannya pada tugas hilir (makna probabilistik model sering dibuang di bagian akhir).
- Sebaliknya, model grafis klasik sering memperhatikan kebenaran pembelajaran dan inferensi semua variabel

## 2.2. STUDI PERMODELAN GRAFIS HYBRID

Partisi rata-rata dari jaringan kepercayaan sigmoid untuk inferensi GMF selanjutnya



berfokus pada inferensi/keakuratan pembelajaran, kecepatan, dan partisi



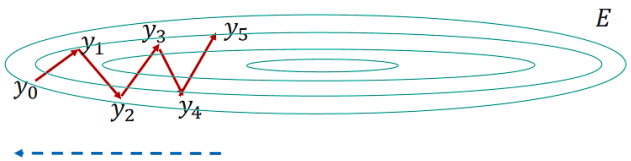

Pemodelan output terstruktur (CRF) berbasis energi

$$\mathbf{y}^*(\mathbf{x};\mathbf{w}) := \arg\min_{\mathbf{y}} E(\mathbf{y},\mathbf{x};\mathbf{w})$$

Buka gulungan algoritme pengoptimalan untuk sejumlah langkah tetap

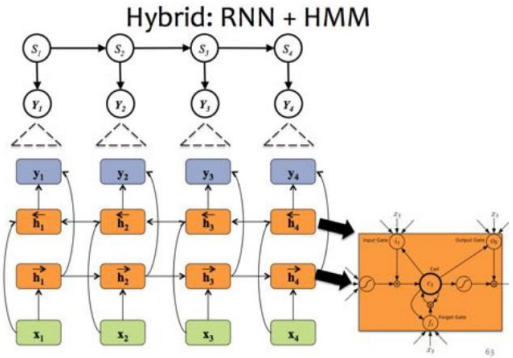
$$\mathbf{y}^*(\mathbf{x};\mathbf{w}) = \text{opt-alg}_{\mathbf{y}} E(\mathbf{y},\mathbf{x};\mathbf{w})$$

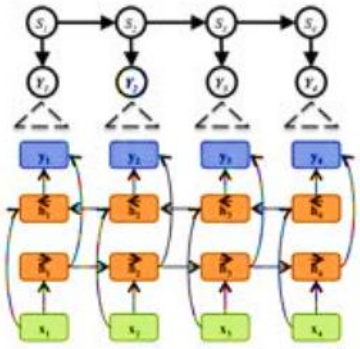
- Kita dapat menganggap y* sebagai beberapa fungsi non-linear yang dapat dibedakan dari input dan bobot → memaksakan beberapa kerugian dan mengoptimalkannya seperti grafik komputasi standar lainnya menggunakan backprop!
- Demikian pula, algoritma inferensi berbasis pengiriman pesan dapat dipotong dan diubah menjadi grafik komputasi.




Kami dapat melakukan backprop melalui langkah-langkah pengoptimalan karena ini hanyalah urutan perhitungan

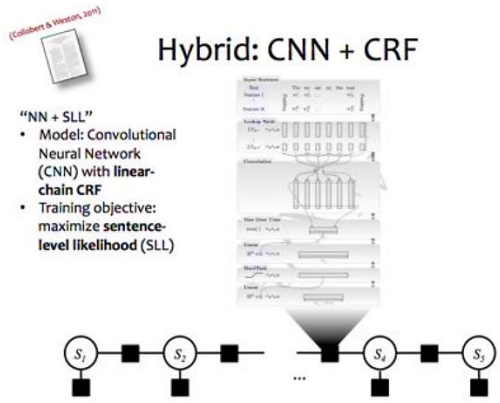
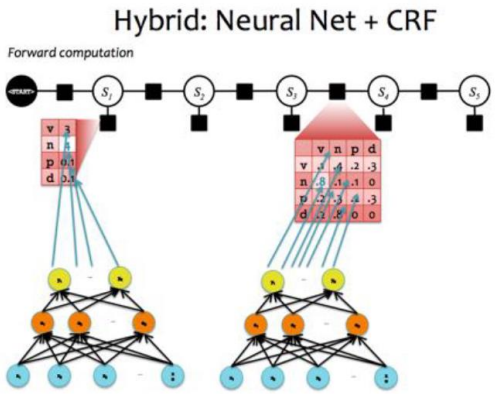
## 2.3 PENGGABUNGAN ANTARA JARINGAN SYARAF DAN PERMODELAN GRAFIS



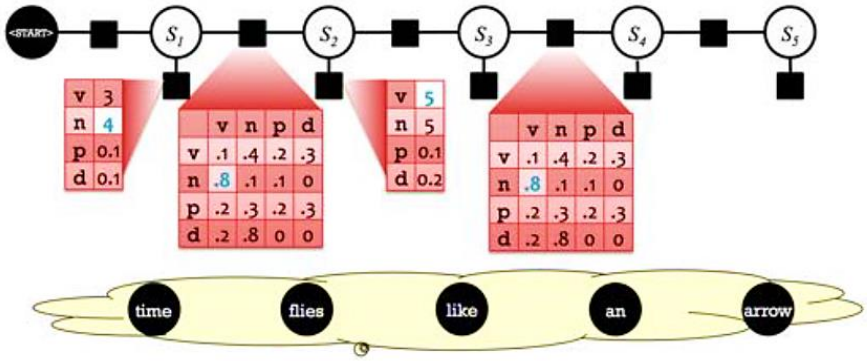


Permodelan, inferensi dan pemelajaran dapat dianalogikan dengan Hybrid NN + HNN dengan tujuan kemungkinan Log, inferensi menggunakan algoritma Forward – Backward.

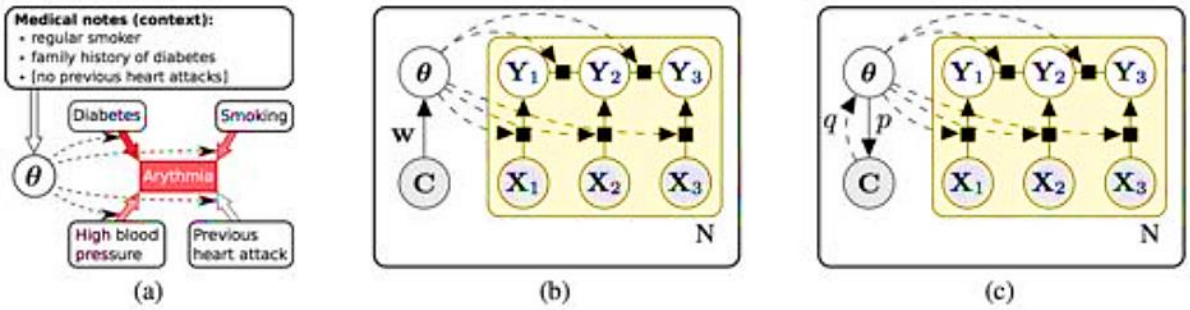
Berikut ini adalah contoh permodelan Neural Network yang dikombinasikan dengan permodelan yang lain:

| Hybrid CNN dan CRF | Hybrid Neural network dan CRF |
|---|---|
|  |  |

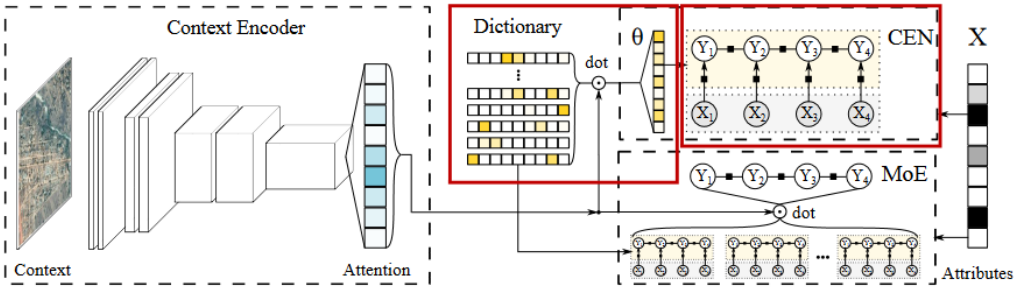
Hybrid NN + GM bersyarat



## 2.4 MENGGUNAKAN GM SEBAGAI PREDIKSI



Ide nya adalah dalam menggunakan jaring saraf dalam untuk menghasilkan parameter model grafis untuk konteks tertentu (mis., contoh atau kasus tertentu). GM yang dihasilkan digunakan ini untuk membuat prediksi akhir. GM dibuat di atas variabel yang dapat ditafsirkan dan dapat digunakan sebagai penjelasan kontekstual untuk setiap prediksi.




Implementasinya adalah mempertahankan data dan parameter GM yang tersedia. Juga Memproses input yang kompleks (gambar, teks, deret waktu, dll.) dalam penggunaan layer atau jaringan dalam; perhatikan untuk memilih atau menggabungkan model dari kamus, serta gunakan GM yang dirancang (misalnya CRF) untuk membuat prediksi, setelah itu periksa GM untuk memahami alasan di balik prediksi yang muncul.

**Pembelajaran Bayesian tentang NN**

Jaringan saraf tiruan (NNs) telah menjadi standar de facto dalam pembelajaran mesin. Jaringan ini memungkinkan mempelajari transformasi yang sangat nonlinier dalam banyak aplikasi. Namun, NN biasanya hanya memberikan perkiraan titik tanpa menghitung ketidakpastian terkait secara sistematis. Dalam buku ini pendekatan baru terhadap NN sepenuhnya Bayesian diusulkan, di mana pelatihan dan prediksi perceptron dilakukan dalam kerangka inferensi Bayesian dalam bentuk tertutup. Bobot dan prediksi perceptron dianggap sebagai variabel acak Gaussian. Ekspresi analitik untuk memprediksi output perceptron dan untuk mempelajari bobot disediakan untuk fungsi aktivasi yang umum digunakan seperti sigmoid atau ReLU.

Bayesiyan Memperkirakan parameter distribusi perkiraan ini tidak dapat dilakukan dalam bentuk tertutup pada umumnya. Alih-alih, pengambilan sampel Monte Carlo dan penurunan gradien biasanya digunakan, yang membuat pelatihan Bayesian NN menjadi ekspansif secara komputasional dan menyebabkan masalah dalam mengontrol varian tinggi estimasi gradien Monte Carlo.

Alternatif inferensi variasional untuk mendekati posterior Bayesian di atas bobot adalah dropout. Proses ini memungkinkan kuantifikasi ketidakpastian dan sesuai dengan perkiraan distribusi variasional. Bayesian NN diperlakukan sebagai masalah pemfilteran Kalman, tetapi seperti inferensi variasional, penurunan gradien diperlukan untuk menghitung matriks pemfilteran. Pendekatan ini tidak memerlukan perhitungan gradien yang mahal secara komputasi dan selanjutnya memungkinkan pembelajaran berurutan. Jaringan saraf sebagai model probabilistik dapat dihitung dengan:

$$P(y|x,\theta)$$

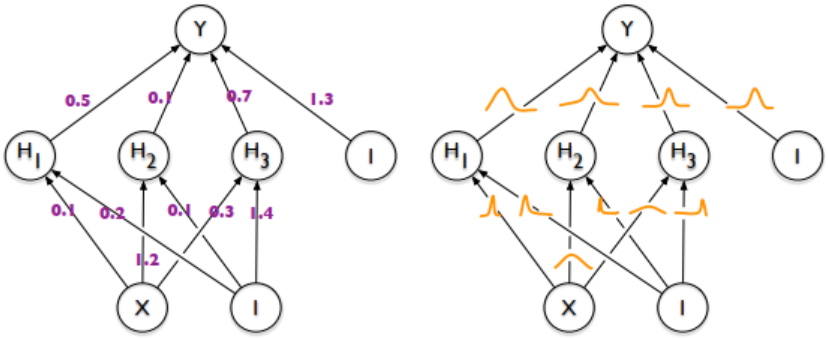
$P$ adalah Prediksi yang akan dicari.

$y$ diartikan sebagai distribusi kategorikal untuk klasifikasi ⇒ kehilangan lintas-entropi
$x$ diartikan sebagai distribusi Gaussian untuk regresi ⇒ kuadrat kerugian

Sebelumnya pada parameter: $p(\theta)$, Maka solusi maksimum a posteriori (MAP) dapat dihitung dengan:

$$\boldsymbol{\theta}^{MAP} = \text{argmax}_{\theta} \log p(y|x, \boldsymbol{\theta}) p(\boldsymbol{\theta})$$

Jika Regularisasi Gaussian sebelumnya ⇒ L2, sedangkan Laplace sebelum regularisasi ⇒ L1, dengan Inferensi variasi dengan perkiraan posterior $q$(c), maka kita dapat mempolakan perhitungan dengan pola seperti dibawah ini:




- Inferensi variasi (singkatnya):

$$\min_q F(D, \boldsymbol{\theta}) = \text{KL}\big(q(\boldsymbol{\theta}) || \, p(\boldsymbol{\theta}|D)\big) - \text{E}_{q(\boldsymbol{\theta})}[\log p(D|\boldsymbol{\theta})]$$

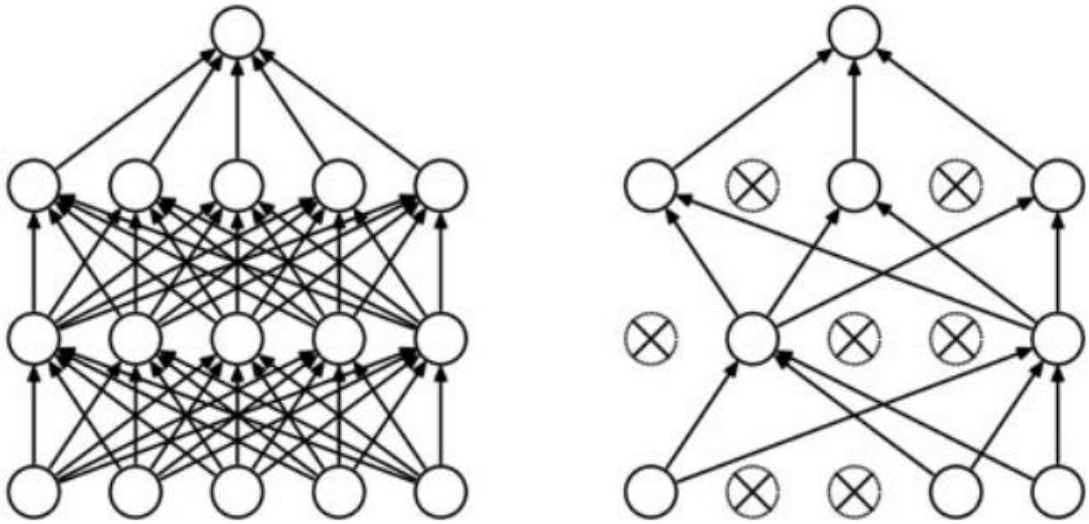
$$\min_q F(D, \boldsymbol{\theta}) = \text{KL}\big(q(\boldsymbol{\theta}) || \, p(\boldsymbol{\theta}|D)\big) - \sum_i \log p(D|\boldsymbol{\theta}_i)$$

dimana $c_i \sim q$(c); Istilah KL dapat didekati dengan cara yang sama

Kita dapat mendefinisikan $q$ sebagai Gaussian diagonal atau Gaussian kovarian penuh. Sebagai alternatif, $q$ dapat didefinisikan secara implisit.

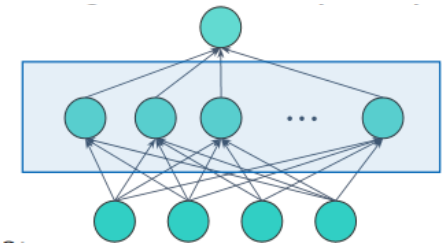
$$\boldsymbol{\theta} = \boldsymbol{M} \cdot \text{diag}(\boldsymbol{z}),$$
$$\boldsymbol{z} \sim \text{Bernoulli}(p)$$

Mengeluarkan neuron setara dengan mengosongkan kolom matriks parameter (yaitu, bobot) $z_i$ = 0 sesuai dengan kolom ke-i dari $M$ dikeluarkan ⇒ prosedurnya sama dengan putus unit I, Parameter variasi adalah $\{M, p\}$



**Model Dalam yang "Sangat Dalam"**

Kita telah melihat bahwa jaringan yang "sangat dalam" dapat dijelaskan oleh GM yang tepat, Bagaimana dengan jaringan yang "sangat luas"? dengan mempertimbangkan jaringan saraf dengan Gaussian sebelum bobotnya tak terhingga banyak neuron tersembunyi di lapisan menengah. Ternyata, jika kita memiliki Gaussian sebelumnya pada bobot jaringan tak terbatas tersebut, itu akan setara dengan proses Gaussian.



Dapat kita lihat pada bagan diatas bahwa banyak sekali unit tersembunyi, karena pada jaringan ini memiliki parameter yang jauh lebih banyak daripada titik data training, yang memungkinkannya mencapai kesalahan pelatihan mendekati nol secara bersamaan — untuk beberapa alasan yang tidak ditentukan. Saat digunakan untuk prediksi, GP memperhitungkan korelasi antara titik data dan dapat menghasilkan estimasi ketidakpastian prediksi yang terkalibrasi dengan baik. Dari bagan diatas Proses Gaussian (GP) adalah distribusi fungsi:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \\ f(\mathbf{x}) &\sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \end{aligned}$$

## 2.5 PROSES GAUSSIAN DAN DEEP KERNEL LEARNING

Kelas-kelas tertentu dari prior Gaussian untuk jaringan saraf dengan banyak unit tersembunyi yang tak terhingga menyatu ke proses Gaussian. Menggabungkan bias induktif arsitektur model dalam dengan fleksibilitas non-parametrik proses Gaussian seperti yang ditulis pada rumus dibawah ini:

$$k(x_i, x_j|\phi) \rightarrow k(g(x_i, \theta), g(x_j, \theta)|\phi, \theta) \quad \text{where } K_{ij} = k(x_i, x_j)$$

Mulai dari kernel dasar

$$k(x_i, x_j|\phi),$$

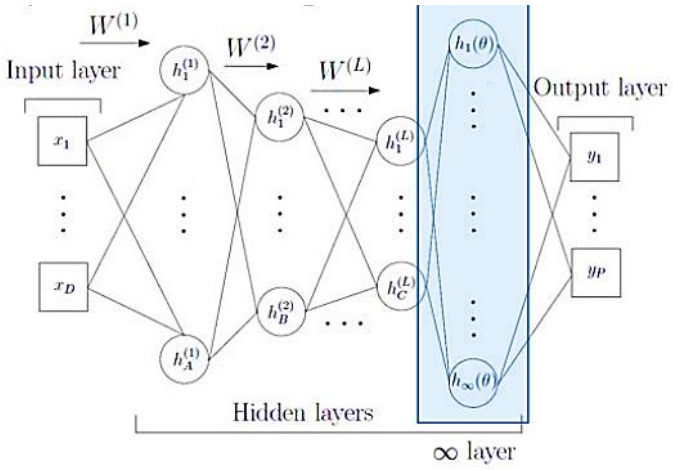
ubah input $x$ menjadi

$$p(y|f) = \mathcal{N}(y|f, \beta^{-1})$$
$$p(f|\phi) = \mathcal{N}(f|m(x), K)$$

Pelajari parameter kernel dan saraf $^{\{\phi,\theta\}}$ bersama-sama dengan mengoptimalkan kemungkinan log marjinal (atau batas bawah variasinya).

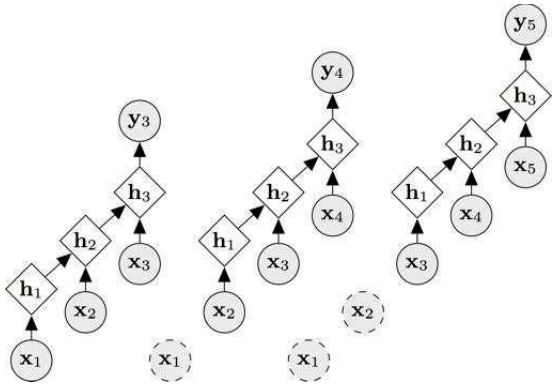
Pembelajaran cepat dan inferensi dengan interpolasi kernel lokal, titik induksi terstruktur, dan perkiraan. Dengan menambahkan Gausian Proses sebagai lapisan ke jaringan saraf yang dalam, kita dapat menganggapnya sebagai menambahkan lapisan tersembunyi yang tak terbatas dengan bobot sebelumnya tertentu. Pembelajaran kernel yang mendalam yaitu menggabungkan bias induktif model dalam dengan fleksibilitas non-parametrik proses

Gaussian menambahkan regularisasi yang kuat ke jaringan selain itu, mereka memberikan estimasi ketidakpastian prediktif.




**Pembelajaran kernel yang mendalam pada data berurutan**

Sebelum membahas tentang pembelajaran ini, pasti akan terlintas pertanyaan, Bagaimana jika kita memiliki data yang bersifat berurutan? Bisakah kita tetap menerapkan alasan yang sama dan membangun model nonparametrik yang kaya di atas jaring berulang?

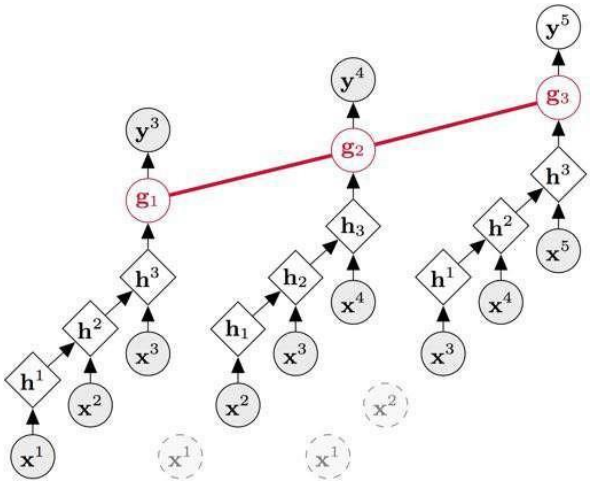


Jawabannya iya!

Dengan menambahkan lapisan GP ke jaringan berulang, kami secara efektif mengkorelasikan sampel sepanjang waktu dan mendapatkan prediksi beserta estimasi ketidakpastian yang terkalibrasi dengan baik. Namun untuk melatih model seperti itu menggunakan teknik stokastik membutuhkan perhatian tambahan.

Model berulang dengan memori jangka pendek panjang (LSTM) baru-baru ini muncul sebagai pendekatan utama untuk memodelkan struktur sekuensial. LSTM adalah metode berbasis gradien yang efisien untuk melatih jaringan berulang. LSTM menggunakan node dalam setiap unit tersembunyi dan mekanisme gerbang khusus yang menstabilkan aliran kesalahan yang disebarkan kembali, meningkatkan proses pembelajaran model. Sementara LSTM memberikan hasil mutakhir pada data ucapan dan teks, mengukur ketidakpastian atau mengekstraksi distribusi prediktif penuh dari model yang dalam masih merupakan bidang penelitian aktif. Mengukur ketidakpastian prediktif model dalam dengan mengikuti pendekatan nonparametrik Bayesian. Secara khusus, mengusulkan fungsi kernel yang

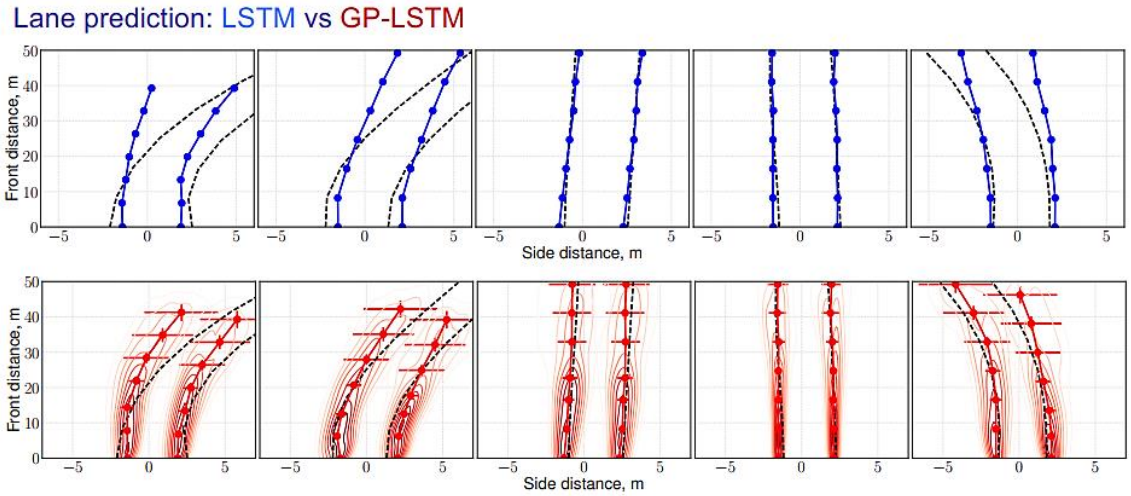
sepenuhnya merangkum sifat struktural LSTM, untuk digunakan dengan proses Gaussian. Model yang dihasilkan memungkinkan proses Gaussian untuk mencapai kinerja canggih pada tugas regresi berurutan, sementara juga memungkinkan representasi prinsip ketidakpastian dan fleksibilitas non-parametrik. Selanjutnya, kami mengembangkan algoritme pengoptimalan semi-stokastik konvergen yang terbukti memungkinkan pembaruan mini-batch dari kernel berulang. Kami secara empiris menunjukkan bahwa pendekatan semi-stokastik ini secara signifikan meningkatkan metode orde pertama non-stokastik standar dalam waktu proses dan dalam kualitas solusi konvergen. Untuk skalabilitas tambahan, kami mengeksploitasi struktur aljabar dari kernel ini, menguraikan matriks kovarians yang relevan menjadi produk. Model ini tidak hanya dapat diinterpretasikan sebagai proses Gaussian dengan kernel berulang, tetapi juga sebagai jaringan berulang yang dalam dengan keluaran probabilistik, banyak unit tersembunyi, dan fungsi utilitas yang kuat untuk overfitting.




Pada bagian berikut, kami memformalkan masalah belajar dari data berurutan, memberikan latar belakang jaringan berulang dan LSTM, dan menyajikan evaluasi empiris yang luas dari model kami. Secara khusus, kami menerapkan model kami ke sejumlah tugas, termasuk identifikasi sistem, prakiraan energi, dan aplikasi mobil tanpa pengemudi. Secara kuantitatif, model ini dinilai berdasarkan ukuran data mulai dari ratusan titik hingga hampir satu juta dengan berbagai rasio signal-to-noise yang menunjukkan kinerja canggih dan penskalaan linier dari pendekatan kami. Secara kualitatif, model diuji pada aplikasi self-driving konsekuensial: estimasi lajur dan prediksi posisi kendaraan di depan. Memang, fokus utama dari makalah ini adalah pada pencapaian kinerja canggih pada aplikasi konsekuensial yang melibatkan data berurutan, mengikuti pendekatan langsung dan terukur untuk membangun proses Gaussian yang sangat fleksibel.

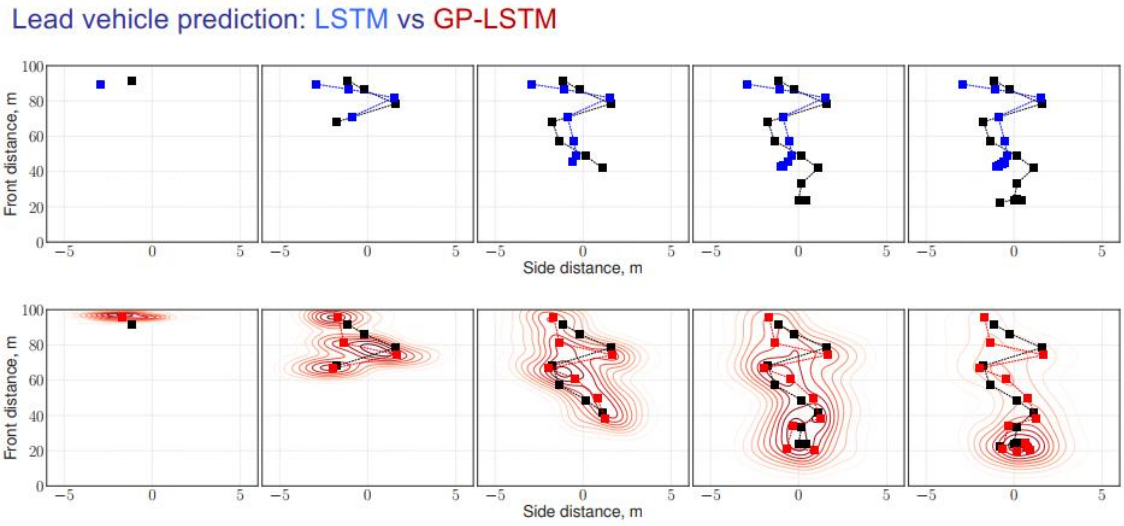
Kita bisa mengkombinasikan GP dengan model sequential seperti RNN atau LSTM. Dalam model sekuensial reguler, kami memiliki input dengan panjang tetap yang menghasilkan satu output. Saat memasukkan ke depan melalui urutan, kita harus membuang bagian urutan sebelumnya karena model mengambil input dengan panjang tetap. Output tidak menggunakan informasi dari seluruh urutan. Kita dapat menambahkan layer GP yang menggunakan representasi laten dari h sebagai input dan memetakan ke output. Dengan

lapisan GP yang ditambahkan, kami menggunakan semua input hingga titik tersebut dalam urutan. Prediksi diturunkan menggunakan GP bersyarat dengan rata-rata dan kovarian seperti yang dijelaskan sebelumnya. Kami mendapat manfaat dari NN dan GP berurutan (yang memberikan korelasi dan ketidakpastian).




Prediksi jalur: Biru adalah LSTM standar, dan merah adalah GP-LSTM. Titik menunjukkan prediksi.

Gambar diatas merupakan perbedaan grafik prediksi antara LSTM dan grafik prediksi LSTM. Kami ingin memprediksi lokasi jalur di depan, Terlihat perbedaan kualitas/akurasi prediksi, dengan GP-LSTM memiliki akurasi yang lebih tinggi. GP-LSTM juga memberikan error bound (ketidakpastian) pada prediksi, bukan hanya prediksi titik. Memprediksi jarak kendaraan utama. Membandingkan LSTM dan GP-LSTM, GP-LSTM memiliki akurasi yang lebih tinggi dan memberikan ketidakpastian. Ketidakpastian berguna karena kita dapat mengambil tindakan tambahan berdasarkan ketidakpastian. Saat ketidakpastian besar, kita mungkin ingin menjauh dari pengemudi.

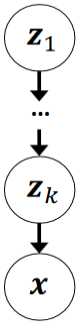



Prediksi kendaraan utama: Biru adalah LSTM standar, dan merah adalah GP-LSTM. Kurva GP-LSTM mewakili ketidakpastian.

# BAB III
# *DEEP GENERATIVE MODELS* (DGM)

## 3.1 PENGANTAR

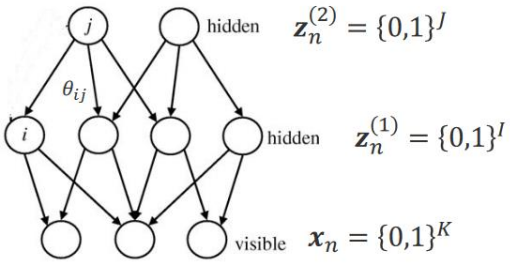
*Deep generative model (DGM)* adalah jaringan saraf dengan banyak lapisan tersembunyi yang dilatih untuk memperkirakan distribusi probabilitas berdimensi tinggi yang rumit menggunakan sejumlah besar sampel. Ketika berhasil dilatih, kita dapat menggunakan DGM untuk memperkirakan kemungkinan setiap pengamatan dan membuat sampel baru dari distribusi yang mendasarinya. Mengembangkan DGM telah menjadi salah satu bidang yang paling banyak diteliti dalam kecerdasan buatan dalam beberapa tahun terakhir. Literatur tentang DGM telah menjadi luas dan berkembang pesat. Beberapa kemajuan bahkan telah mencapai ruang publik, misalnya keberhasilan baru-baru ini dalam menghasilkan gambar, suara, atau film yang terlihat realistis; yang disebut deep fakes. Terlepas dari keberhasilan ini, beberapa masalah matematis dan praktis membatasi penggunaan DGM secara lebih luas: mengingat kumpulan data tertentu, tetap menantang untuk merancang dan melatih DGM dan bahkan lebih menantang lagi untuk mengetahui mengapa model tertentu efektif atau tidak.




**Bentuk awal model generatif dalam**

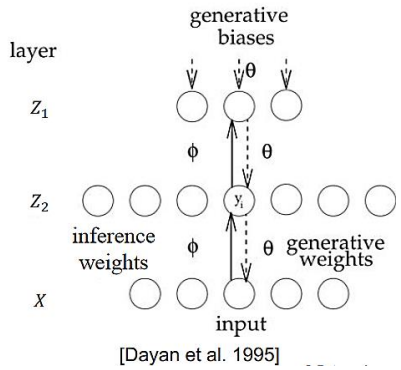
*Deep Generative Models* (DGM), memberikan janji tambahan karena menyandikan struktur material dan/atau properti ke dalam ruang laten, dan melalui eksplorasi dan manipulasi ruang laten dapat menghasilkan material baru. Pendekatan ini mempelajari representasi struktur material dan kimia atau fisika yang sesuai untuk mempercepat penemuan material, yang berbeda dari metode AI/ML tradisional yang menggunakan penyaringan statistik dan kombinatorial dari material yang ada melalui hubungan struktur-properti yang berbeda. Namun, aplikasi DGM untuk bahan anorganik lebih sulit daripada molekul organik karena struktur anorganik seringkali lebih kompleks untuk dikodekan. Dalam karya ini kami meninjau inovasi terbaru yang memungkinkan DGM mempercepat penemuan bahan anorganik. Kami fokus pada representasi struktur material yang berbeda, dampaknya pada strategi desain terbalik menggunakan autoencoder variasional atau jaringan adversarial generatif, dan menyoroti potensi pendekatan ini untuk menemukan material dengan properti

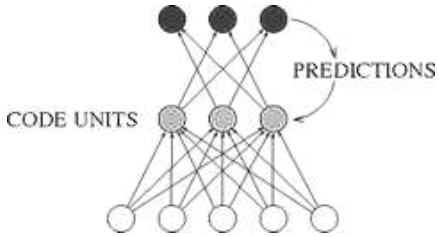
target yang diperlukan untuk inovasi teknologi. Permodelan ini mengacu pada Model Bayesian hierarkis atau Jaring singkat sigmoid




$$p\left(x_{kn} = 1 \middle| \boldsymbol{\theta}_k, \boldsymbol{z}_n^{(1)}\right) = \sigma\left(\boldsymbol{\theta}_k^T \boldsymbol{z}_n^{(1)}\right)$$
$$p\left(z_{in}^{(1)} = 1 \middle| \boldsymbol{\theta}_i, \boldsymbol{z}_n^{(2)}\right) = \sigma\left(\boldsymbol{\theta}_i^T \boldsymbol{z}_n^{(2)}\right)$$

Salah satu pendekatan permodelan ini mengunakan permodel jaringan saraf Mesin Helmholtz. Mesin Helmholtz adalah jaringan saraf tiruan yang, melalui banyak siklus penginderaan dan kemungkinan, secara bertahap belajar membuat prediksi menyatu dengan kenyataan, dan, dalam prosesnya, menciptakan model internal ringkas dari dunia yang berfluktuasi.




Pelatihan DGM melalui kerangka gaya EM, Sampling / augmentasi data dapat ditulis dengan rumus:

$$\boldsymbol{z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2\}$$
$$\boldsymbol{z}_1^{new} \sim p(\boldsymbol{z}_1 | \boldsymbol{z}_2, \boldsymbol{x})$$
$$\boldsymbol{z}_2^{new} \sim p(\boldsymbol{z}_2 | \boldsymbol{z}_1^{new}, \boldsymbol{x})$$

Sedangkan perhitungan Inferensi variasi dihitung dengan rumus:

$$\log p(\boldsymbol{x}) \geq \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z})) \coloneqq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$$
$$\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$$

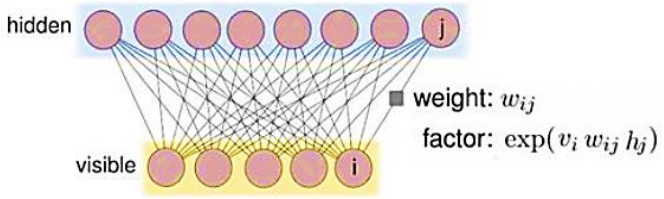
Wake Sleep

Wake: $\min_\theta \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$

Sleep: $\min_\phi \mathbb{E}_{p_\theta(x|z)}[\log q_\phi(z|x)]$

*Restricted Boltzmann machines* (RBMs)

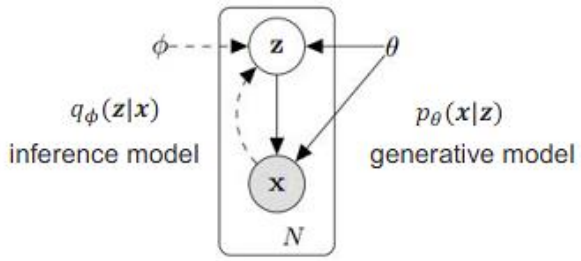
Membangun blok model probabilistik mendalam, metode pembelajaran terawasi digunakan ketika tujuannya adalah mempelajari pemetaan antara atribut dan target dalam data. Ketika tujuannya adalah untuk mengidentifikasi struktur yang mendasari atau pola dalam data, metode pembelajaran tanpa pengawasan berguna. Beberapa metode pembelajaran tanpa pengawasan yang populer adalah Pengelompokan, Pengurangan dimensi, Penambangan asosiasi, Deteksi anomali, dan Model generatif. Masing-masing teknik ini memiliki tujuan pengenalan pola yang berbeda seperti mengidentifikasi pengelompokan laten, mengidentifikasi ruang laten, menemukan ketidakteraturan dalam data, estimasi kepadatan atau menghasilkan sampel baru dari data. Pada artikel kali ini kita akan fokus pada model generatif, khususnya Boltzmann Machine (BM), varian Restricted Boltzmann Machine (RBM) yang populer, cara kerja RBM dan beberapa aplikasinya. Sebelum mendalami lebih dalam tentang BM, kita akan membahas beberapa konsep dasar yang penting untuk memahami BM.




- Autoencoder variasi (VAE)

Autoencoder adalah jaringan saraf yang mempelajari dua fungsi; 1) Penyandian: membuat representasi terkompresi atau penyandian dari data input, 2) Dekode: membuat ulang data masukan dari representasi penyandian. Representasi yang dibuat ulang harus dekat dengan input asli. Fungsi encoder biasanya disebut sebagai pengurangan data dalam ruang yang diamati ke ruang laten. Gambar dibawah ini menunjukkan arsitektur tipikal dari autoencoder. Dalam arsitektur ini, diindikasikan bahwa input ruang observasi enam dimensi direduksi menjadi ruang laten dua dimensi. Autoencoder mempelajari parameter jaringan selama propagasi balik mirip dengan jaringan pembelajaran yang diawasi tetapi perbedaannya ada pada fungsi biaya. Sementara jaringan pembelajaran yang diawasi menggunakan nilai variabel target dalam fungsi biaya, autoencoder menggunakan nilai input.

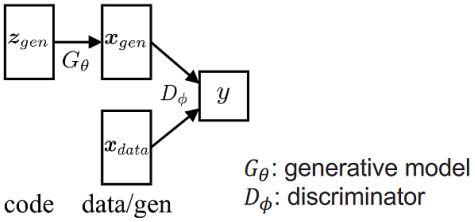
Setelah autoencoder dilatih, bagian encoder dari jaringan dapat dibuang dan bagian decoder dapat digunakan untuk menghasilkan data baru di ruang yang diamati dengan membuat sampel data acak di ruang laten dan memetakannya ke ruang yang diamati. Ini adalah ide inti dari model generatif. Penjelasan singkat tentang autoencoder disajikan karena kesamaan antara autoencoder dan Mesin Boltzmann (BM). Seperti Autoencoder, BM berguna untuk mengekstrak ruang laten dari data. Perbedaannya terletak pada arsitektur, representasi ruang laten, dan proses pelatihan.




*Generative Adversarial Networks (GANs)*

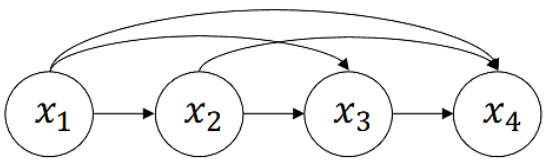
Pelatihan permusuhan telah sepenuhnya mengubah cara kami mengajarkan jaringan saraf untuk melakukan tugas tertentu. Jaringan Adversarial Generatif tidak bekerja dengan estimasi kepadatan eksplisit apa pun seperti Variational Autoencoder. Sebaliknya, ini didasarkan pada pendekatan teori permainan dengan tujuan untuk menemukan keseimbangan Nash antara dua jaringan, Generator dan Diskriminator. Idenya adalah mengambil sampel dari distribusi sederhana seperti Gaussian dan kemudian belajar mengubah noise ini menjadi distribusi data menggunakan penaksir fungsi universal seperti jaringan saraf.

Pada dasarnya tugas Generator adalah menghasilkan gambar yang tampak alami dan tugas Diskriminator adalah memutuskan apakah gambar tersebut palsu atau asli. Ini dapat dianggap sebagai permainan dua pemain mini-max di mana kinerja kedua jaringan meningkat seiring waktu. Generator mencoba mengelabui diskriminator dengan menghasilkan gambar nyata sejauh mungkin dan diskriminator berusaha untuk tidak dikelabui oleh generator dengan meningkatkan kemampuan diskriminatifnya. Gambar di bawah menunjukkan arsitektur dasar GAN.




***Jaringan Pencocokan Momen Generatif (GMMNs)***

Masalah lama dan inheren dalam pembelajaran tanpa pengawasan adalah menentukan metode yang baik untuk evaluasi. Model generatif menawarkan kemampuan untuk mengevaluasi generalisasi dalam ruang data, yang juga dapat dinilai secara kualitatif. GMMN adalah jaringan saraf generatif yang dimulai dengan prior sederhana yang mudah diambil sampelnya. Ini disebarkan secara deterministik melalui lapisan tersembunyi jaringan dan hasilnya adalah sampel dari model. Jadi, dengan GMMN mudah untuk mengambil sampel acak independen dengan cepat, berlawanan dengan prosedur MCMC mahal yang diperlukan dalam model lain seperti mesin Boltzmann.



## 3.2 INFERENSI VARIASI DGM

Variational Inference (VI) telah mendapatkan popularitas dalam fisika statistik, pemodelan data, dan jaringan saraf. Masalahnya melibatkan penggunaan metrik untuk memilih perkiraan yang dapat dilacak ke kepadatan probabilitas posterior. Metodologi ini merumuskan masalah inferensi statistik sebagai masalah optimisasi. Dengan demikian, kami mendapatkan manfaat kecepatan dari estimasi a posteriori (MAP) maksimum dan dapat dengan mudah menskalakan kumpulan data yang besar.

Pertimbangkan model generatif $p_\theta(x|z)$ dan sebelumnya $p(z)$. Dengan distribusi bersama: $p_\theta(\boldsymbol{x},\boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$

- Asumsikan distribusi variasional $q_\phi(\boldsymbol{z}|\boldsymbol{x})$

Tujuannya adalah memaksimalkan batas bawah untuk kemungkinan log

$$\begin{aligned}&\log p(\boldsymbol{x})\\&= KL\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right) + \int_{\boldsymbol{z}} q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\\&\geq \int_{\boldsymbol{z}} q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\\&:= \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x})\end{aligned}$$

Ekuivalen, meminimalkan energi bebas

$$F(\boldsymbol{\theta},\phi;\boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

Maksimalkan variasi batas bawah

$$\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x})$$

Langkah Selanjutnya: maksimalkan $\mathcal{L}$ wrt. $\emptyset$ dengan $\theta$ tetap

$$\max_\phi \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x}) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + KL(q_\phi(z|x)||p(z))$$

Jika dengan solusi bentuk tertutup

$$q_\phi^*(z|x) \propto \exp[\log p_\theta(x,z)]$$

maksimalkan $\mathcal{L}$ wrt. $\theta$ dengan $\emptyset$ tetap

$$\max_\theta \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x}) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + KL(q_\phi(z|x)||p(z))$$

**Inferensi Variasi Diamortisasi**

Inferensi diamortisasi mengacu pada penggunaan solusi inferensi sebelumnya (atau pra-perhitungan lainnya) untuk menyelesaikan masalah inferensi selanjutnya dengan lebih cepat. Ada bukti eksperimental bahwa orang memanfaatkan pengalaman dari tugas inferensi sebelumnya ketika diminta untuk menyelesaikan tugas terkait. Ide ini telah mengilhami penelitian untuk mengembangkan sistem inferensi diamortisasi untuk jaringan Bayesian. Model sistem ini dengan membalikkan topologi jaringan dan mencoba mempelajari distribusi kondisional lokal dari model grafis terbalik ini. Inferensi diamortisasi juga dapat dicapai melalui inferensi variasional. Distribusi variasional sebagai model inferensi $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ dengan parameter $\emptyset$. Amortisasi biaya inferensi dengan mempelajari satu model inferensi yang bergantung pada data. Model inferensi terlatih dapat digunakan untuk inferensi cepat pada data baru. Dengan cara maksimalkan variasi batas bawah $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x})$

- Langkah-E: maksimalkan $\mathcal{L}$ wrt. $\emptyset$ dengan $\theta$ tetap
- Langkah-M: maksimalkan $\mathcal{L}$ wrt. $\theta$ dengan $\emptyset$ tetap

**Model generatif mendalam dengan inferensi diamortisasi**

Variable yang diamortisasi mengasumsikan bahwa variabel laten ini dapat diprediksi oleh fungsi data yang berparameter. Jadi, setelah fungsi ini diestimasi, variabel laten dapat diperoleh dengan melewatkan titik data baru melalui fungsi tersebut. Jaringan saraf dalam yang digunakan dalam konteks ini juga disebut jaringan inferensi. Inferensi diamortisasi pasti cepat, tetapi parameter variasi didekati dengan fungsi parametrik dari data masukan,

*Neural Variation Inference and Learning* (NVIL) adalah pilihan yang sangat alami untuk permodelan ini, karena melatih jaringan inferensi, jaringan saraf yang secara langsung memetakan dokumen ke perkiraan distribusi posterior, tanpa perlu menjalankan pembaruan variasional lebih lanjut. Ini secara intuitif menarik karena dalam model ini, kami berharap pemetaan dari dokumen ke distribusi posterior berperilaku baik, yaitu, perubahan kecil pada dokumen hanya akan menghasilkan perubahan kecil pada topik. Ini persis jenis pemetaan yang harus direpresentasikan dengan baik oleh penaksir fungsi universal seperti jaringan saraf. Pada dasarnya, jaringan inferensi belajar meniru efek inferensi probabilistik, sehingga pada data uji, kita dapat menikmati manfaat pemodelan probabilistik tanpa membayar biaya lebih lanjut untuk inferensi. Kita akan melihat nanti bahwa pendekatan permusuhan juga termasuk dalam daftar:

- Minimisasi prediktabilitas (PM)
- Jaringan permusuhan generatif (GAN)

## 3.3 ALGORITMA WAKE DAN SLEEP

Algoritma W-S (Wake-Sleep) adalah aturan pembelajaran sederhana untuk model dengan variabel tersembunyi. Terlihat bahwa algoritma ini dapat diterapkan pada model analisis faktor yang merupakan versi linear dari mesin Helmholtz. Tetapi bahkan untuk model analisis faktor, konvergensi umum tidak terbukti secara teoritis. Pada buku ini menjelaskan pemahaman geometris dari algoritma W-S yang berbeda dengan algoritma EM (Expectation Maximization) dan algoritma EM. Dalam algoritme Wake-Sleep, tujuannya adalah untuk mempelajari representasi yang ekonomis untuk dideskripsikan tetapi memungkinkan input direkonstruksi secara akurat.

Kita dapat mengukur tujuan ini dengan membayangkan permainan komunikasi di mana setiap vektor input sensorik mentah dikomunikasikan ke penerima dengan terlebih dahulu mengirimkan representasi tersembunyinya dan kemudian mengirimkan perbedaan antara vektor input dan rekonstruksi top-down dari representasi tersembunyi tersebut. Latih model inferensi terpisah bersama dengan model generatif. Umumnya berlaku untuk berbagai model generatif, misalnya mesin Helmholtz .

Maksimalkan kemungkinan log data dengan dua langkah relaksasi kerugian:

- Maksimalkan batas bawah log-kemungkinan, atau ekuivalen, minimalkan energi bebas
  $F(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_{\phi}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$
- Minimalkan tujuan yang berbeda (KLD terbalik) wrt Ø untuk memudahkan pengoptimalan dengan memutuskan ke kerugian batas bawah variasional asli, dapat dihitung dengan rumus:
  $F'(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(p_{\theta}(\boldsymbol{z}|\boldsymbol{x}) \,||\, q_{\phi}(\boldsymbol{z}|\boldsymbol{x}))$

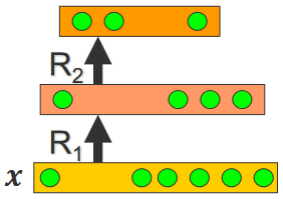
Energi bebas:

$$F(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

- Minimalkan energi bebas wrt.$\theta$ dari $p_\theta$ → *Fase Wake*

  $\max_{\boldsymbol{\theta}} \mathrm{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} [\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})]$

Mengacu pada rumus diatas, dapatkan sampel dari $q_\phi(z|x)$ melalui inferensi pada variabel tersembunyi, Gunakan sampel sebagai target untuk memperbarui model generatif $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, lalu sesuai dengan langkah M variasional




Berdasarkan gambar diatas, kita dapat menghitung perhitungan seperti dibawah ini

Energi bebas:

$$F(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

- Minimalkan energi bebas wrt. Ø dari $q_\phi(\boldsymbol{z}|\boldsymbol{x})$

  Sesuai dengan langkah E variasional, Kesulitanyang akan dihadapi:

  - Pengoptimalan data yang dihitung

    $$q_{\boldsymbol{\phi}}^*(\boldsymbol{z}|\boldsymbol{x}) = \frac{\boldsymbol{p_\theta}(\boldsymbol{z}, \boldsymbol{x})}{\int \boldsymbol{p_\theta}(\boldsymbol{z}, \boldsymbol{x})\, \boldsymbol{dz}}$$

  - Dengan variasi tinggi estimasi gradien langsung

    $$\nabla_{\boldsymbol{\phi}} F(\theta, \phi; x) = \cdots + \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} [\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})] + \cdots$$

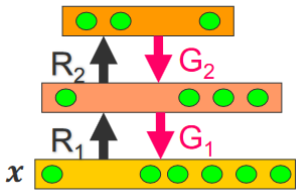
    - Estimasi gradien dengan trik turunan log:

      $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}} [\log p_{\boldsymbol{\theta}}] = \int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}} = \int q_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}}\, \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}} = \mathbb{E}_{q_{\boldsymbol{\phi}}} [\log p_{\boldsymbol{\theta}}\, \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}]$

    - Estimasi Monte Carlo:

      $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}} [\log p_{\boldsymbol{\theta}}] \approx \mathbb{E}_{z_i \sim q_{\boldsymbol{\phi}}} [\log p_{\boldsymbol{\theta}}(x, z_i)\, \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(z_i|x)]$

    - Faktor skala $\log p_\theta$ dari turunan $\nabla_\phi \log q_\phi$ dapat memiliki magnitudo besar sembarang




Berdasarkan gambar diatas, kita dapat menghitung perhitungan seperti dibawah ini

Energi bebas:

$$F(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

- WS mengatasi kesulitan dengan perkiraan fase Wake
- Minimalkan tujuan berikut ➔ Fase Sleep

  $F'(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(p_\theta(\boldsymbol{z}|\boldsymbol{x}) \,||\, q_\phi(\boldsymbol{z}|\boldsymbol{x}))$

  $\max_{\boldsymbol{\phi}} \mathrm{E}_{p_\theta(\boldsymbol{z},\boldsymbol{x})} [\log q_\phi(\boldsymbol{z}|\boldsymbol{x})]$

  "mengambil" sampel dari $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ melalui top-down pass, kemudian gunakan sampel sebagai target untuk memperbarui model inferensi.

| *Algoritma Wake – Sleep* | *Variasional EM* |
|---|---|
| • Model Interferensi parameter $q_{\boldsymbol{\phi}}(\boldsymbol{z}\|\boldsymbol{x})$<br>• Fase Wake:<br>  • Minimalkan<br>  $KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}\|\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{z}\|\boldsymbol{x}))$ wrt. $\theta$<br>  $\mathrm{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}\|\boldsymbol{x})} [\nabla_\theta \log p_{\boldsymbol{\theta}}(\boldsymbol{x}\|\boldsymbol{z})]$<br>• Fase Sleep:<br>  • Minimalkan<br>  $KL(p_\theta(\boldsymbol{z}\|\boldsymbol{x}) \,\|\, q_\phi(\boldsymbol{z}\|\boldsymbol{x}))$ wrt. $\phi$<br>  $\mathrm{E}_{p_\theta(\boldsymbol{z},\boldsymbol{x})} [\nabla_\phi \log q_\phi(\boldsymbol{z}, \boldsymbol{x})]$<br>  • varian data rendah<br>  • Belajar dengan sampel yang dihasilkan dari *x*<br>• Dua tujuan, tidak dijamin bertemu | • Distribusi variasi $q_\phi(\boldsymbol{z}\|\boldsymbol{x})$<br>• Langkah M variasi:<br>  • Minimalkan<br>  $KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}\|\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{z}\|\boldsymbol{x}))$ wrt. $\theta$<br>  $\mathrm{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}\|\boldsymbol{x})} [\nabla_\theta \log p_{\boldsymbol{\theta}}(\boldsymbol{x}\|\boldsymbol{z})]$<br>• Langkah E variasi:<br>  • Minimalkan<br>  $KL(q_{\boldsymbol{\phi}}(\boldsymbol{z}\|\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{z}\|\boldsymbol{x}))$ wrt. $\phi$<br>  $q_{\boldsymbol{\phi}}^* \propto \exp[\log p_\theta]$ jika dengan bentuk tertutup<br>  $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(z, x)]$<br>  • membutuhkan pengurangan varians data dalam praktiknya<br>  • Belajar dengan data nyata *x*<br>• Tujuan tunggal, dijamin menyatu |

## 3.4 (VAE) VARIATIONAL AUTOENCODER

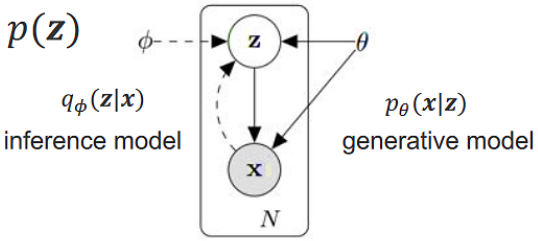
Dua pendekatan yang paling umum digunakan dan efisien adalah Variational Autoencoder (VAE) dan Generative Adversarial Networks (GAN). VAE bertujuan untuk memaksimalkan batas bawah kemungkinan log data dan GAN bertujuan untuk mencapai keseimbangan antara Generator dan Diskriminator.

Penulis berasumsi bahwa pembaca sudah terbiasa dengan cara kerja Variational autoencoder. Kita tahu bahwa kita dapat menggunakan autoencoder untuk menyandikan gambar input ke representasi dimensi yang jauh lebih kecil yang dapat menyimpan informasi laten tentang distribusi data input. Tetapi dalam autoencoder, vektor yang disandikan hanya dapat dipetakan ke input yang sesuai menggunakan dekoder. Itu pasti tidak dapat digunakan untuk menghasilkan gambar serupa dengan beberapa variabilitas.

Untuk mencapai ini, model perlu mempelajari distribusi probabilitas dari data pelatihan. VAE adalah salah satu pendekatan paling populer untuk mempelajari distribusi data yang rumit seperti gambar menggunakan jaringan saraf tanpa pengawasan. Ini adalah model grafis probabilistik yang berakar pada inferensi Bayesian yaitu, model tersebut bertujuan untuk mempelajari distribusi probabilitas yang mendasari data pelatihan sehingga dapat

dengan mudah mengambil sampel data baru dari distribusi yang dipelajari tersebut. Idenya adalah untuk mempelajari representasi laten dimensi rendah dari data pelatihan yang disebut variabel laten (variabel yang tidak diamati secara langsung tetapi lebih disimpulkan melalui model matematika) yang kami asumsikan telah menghasilkan data pelatihan aktual kami. Variabel laten ini dapat menyimpan informasi berguna tentang jenis keluaran yang perlu dihasilkan model. Distribusi probabilitas variabel laten z dilambangkan dengan P(z). Distribusi Gaussian dipilih sebagai pendahuluan untuk mempelajari distribusi P(z) agar dapat dengan mudah mengambil sampel titik data baru selama waktu inferensi.

- Model generatif $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, dan sebelumnya $p(\boldsymbol{z})$
  - Distribusi bersama $p_\theta(\boldsymbol{x},\boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$
- Model inferensi $q_\phi(\boldsymbol{z}|\boldsymbol{x})$




- Variasi batas bawah

  $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x}) = \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x},\boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$

- Optimalkan
  $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x})$ wrt. $\theta$ of $p_\theta(\boldsymbol{x}|\boldsymbol{z})$
  - Sama dengan fase bangun
- Optimalkan
  $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x})$ wrt. $\phi$ of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$

  $\nabla_\phi \mathcal{L}(\theta,\phi;x) = \cdots + \nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + \cdots$
- Gunakan trik reparameterisasi untuk mengurangi varians

Hal terbaik dari VAE adalah ia mempelajari model generatif dan model inferensi. Meskipun VAE dan GAN merupakan pendekatan yang sangat menarik untuk mempelajari distribusi data yang mendasarinya menggunakan pembelajaran tanpa pengawasan, tetapi GAN memberikan hasil yang lebih baik dibandingkan dengan VAE. Di VAE, kami mengoptimalkan batas variasi yang lebih rendah sedangkan di GAN, tidak ada asumsi seperti itu. VAE dan GAN terutama berbeda dalam cara pelatihan. Hal terbaik dari VAE adalah ia mempelajari model generatif dan model inferensi. Meskipun VAE dan GAN merupakan pendekatan yang sangat menarik untuk mempelajari distribusi data yang mendasarinya menggunakan pembelajaran tanpa pengawasan, tetapi GAN memberikan hasil yang lebih baik dibandingkan dengan VAE. Di VAE, kami mengoptimalkan batas variasi yang lebih rendah sedangkan di GAN, tidak ada asumsi seperti itu. Faktanya, GAN tidak berurusan dengan estimasi kepadatan probabilitas eksplisit apa pun. Kegagalan VAE dalam menghasilkan gambar yang tajam menyiratkan bahwa model tidak dapat mempelajari distribusi posterior yang sebenarnya. VAE dan GAN terutama berbeda dalam cara pelatihan.

**Gradien yang diparametrikan ulang**

Gradien reparameterisasi berlaku ketika fungsi kerugian dapat dibedakan dan sampel dari kepadatan yang mendasarinya dapat dihasilkan oleh transformasi sampel deterministik yang dapat dibedakan dari distribusi yang lebih sederhana yang tidak bergantung pada parameter model. Biasanya, gradien reparameterisasi memiliki varian yang lebih rendah daripada gradien fungsi skor, asalkan fungsi kerugiannya cukup mulus. Oleh karena itu, mengembangkan estimator gradien reparameterisasi yang efektif telah menjadi bidang penelitian aktif dalam analisis sensitivitas dan backpropagation stokastik.

Namun, gradien reparameterisasi terutama terbatas pada distribusi dengan konstanta normalisasi yang dapat ditelusuri. Ini menghalangi penggunaannya untuk model minat yang kompleks seperti model berbasis energi dan model Bayesian non-konjugasi. Dengan demikian, menggeneralisasi reparameterisasi dengan MCMC ke distribusi yang tidak dinormalisasi merupakan arah penting untuk mengembangkan estimator gradien yang efektif untuk model yang rumit. Memang, beberapa pekerjaan baru-baru ini berfokus pada gradien reparameterisasi untuk distribusi yang tidak dinormalisasi dalam masalah khusus.

- Optimalkan $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ wrt. $\phi$ of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
  - Rekap: estimasi gradien dengan trik log-derivatif:
    $\nabla_\phi \mathbb{E}_{q_\phi}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] = \mathbb{E}_{q_\phi}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) \nabla_\phi \log q_\phi]$
    - Varian tinggi: $\nabla_\phi \mathbb{E}_{q_\phi}[\log p_\theta] \approx \mathbb{E}_{z_i \sim q_\phi}[\log p_\theta(x, z_i) \nabla_\phi q_\phi(z_i|x)]$
      - Faktor skala $\log p_\theta(x, z_i)$ dari derivatif $\nabla_\phi \log q_\phi$ dapat memiliki magnitudo besar yang sewenang-wenang
  - estimasi gradien dengan trik reparameterization
    $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}) \quad \Leftrightarrow \quad \mathbf{z} = \mathrm{g}_\phi(\boldsymbol{\epsilon}, \boldsymbol{x}), \qquad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$
    $\nabla_\phi \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] = \mathrm{E}_{\boldsymbol{\epsilon} \sim p(\epsilon)}\left[\nabla_\phi \log p_\theta\left(\boldsymbol{x}, \boldsymbol{z}_\phi(\boldsymbol{\epsilon})\right)\right]$
    - (Secara empiris) varian estimasi gradien yang lebih rendah
    - Misalnya, $\boldsymbol{z} \sim N\left(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{L}(\boldsymbol{x})\boldsymbol{L}(\boldsymbol{x})^T\right) \quad \Leftrightarrow \quad \boldsymbol{\epsilon} \sim N(0,1), \; \boldsymbol{z} = \boldsymbol{\mu}(\boldsymbol{x}) + \boldsymbol{L}(\boldsymbol{x})\boldsymbol{\epsilon}$

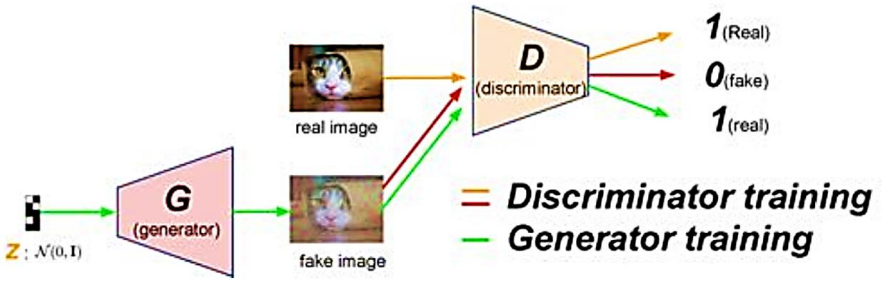
**Contoh hasil dari algoritma VAE**

VAE cenderung menghasilkan gambar buram karena perilaku mode yang menutupi (lebih lanjut nanti). Beberapa kelas (misalnya empat dan sembilan) bersinggungan satu sama lain. Jika Anda mengambil sampel suatu titik dari wilayah ruang laten, di mana empat dan sembilan sering terjadi, VAE pada dasarnya akan menghasilkan superposisi sembilan dan empat. Ini menghasilkan gambar yang buram.

# BAB 4
# GENERATIVE ADVERSARIAL NETWORK (GAN)

Jaringan Adversarial Generatif tidak bekerja dengan estimasi kepadatan eksplisit apa pun seperti Variational Autoencoder. Sebaliknya, ini didasarkan pada pendekatan teori permainan dengan tujuan untuk menemukan keseimbangan Nash antara dua jaringan, Generator dan Diskriminator. Idenya adalah mengambil sampel dari distribusi sederhana seperti Gaussian dan kemudian belajar mengubah noise ini menjadi distribusi data menggunakan penaksir fungsi universal seperti jaringan saraf.

. Model generator G belajar untuk menangkap distribusi data dan model diskriminator D memperkirakan probabilitas bahwa sampel berasal dari distribusi data daripada distribusi model. Pada dasarnya tugas Generator adalah menghasilkan gambar yang tampak alami dan tugas Diskriminator adalah memutuskan apakah gambar tersebut palsu atau asli. Ini dapat dianggap sebagai permainan dua pemain mini-max di mana kinerja kedua jaringan meningkat seiring waktu. Dalam permainan ini, generator mencoba mengelabui diskriminator dengan menghasilkan gambar nyata sejauh mungkin dan diskriminator berusaha untuk tidak dikelabui oleh generator dengan meningkatkan kemampuan diskriminatifnya. Gambar di bawah menunjukkan arsitektur dasar GAN.




Kita mendefinisikan variabel kebisingan input P(z) dan kemudian generator memetakan ini ke distribusi data menggunakan fungsi terdiferensiasi kompleks dengan parameter ɵg. Selain itu, kami memiliki jaringan lain yang disebut Discriminator yang mengambil input x dan menggunakan fungsi terdiferensiasi lainnya dengan parameter ɵd menghasilkan nilai skalar tunggal yang menunjukkan probabilitas bahwa x berasal dari distribusi data sebenarnya Pdata(x). Fungsi tujuan GAN didefinisikan sebagai

$$\max_D \mathcal{L}_D = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1-D(\boldsymbol{x}))\right]$$
$$\min_G \mathcal{L}_G = \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1-D(\boldsymbol{x}))\right].$$

Dalam persamaan di atas, jika input ke Diskriminator berasal dari distribusi data yang sebenarnya maka D(x) harus menghasilkan 1 untuk memaksimalkan fungsi tujuan di atas w.r.t D sedangkan jika gambar dihasilkan dari Generator maka D(G(z)) harus menampilkan 1 untuk meminimalkan fungsi tujuan w.r.t G. Yang terakhir pada dasarnya menyiratkan bahwa G harus menghasilkan gambar realistis yang dapat menipu D. Kami memaksimalkan parameter w.r.t

fungsi di atas Diskriminator menggunakan Gradient Ascent dan meminimalkan parameter w.r.t yang sama dari Generator menggunakan Gradient Descent . Namun ada masalah dalam mengoptimalkan tujuan generator. Pada awal permainan ketika generator belum mempelajari apapun, gradien biasanya sangat kecil dan ketika bekerja dengan sangat baik, gradiennya sangat tinggi. Tapi kami menginginkan perilaku sebaliknya. Oleh karena itu kami memaksimalkan E[log(D(G(z))] daripada meminimalkan E[log(1-D(G(z))]

Proses pelatihan terdiri dari penerapan Stochastic Gradient Descent secara simultan pada Diskriminator dan Generator. Saat pelatihan, kami bergantian antara k langkah mengoptimalkan D dan satu langkah mengoptimalkan G pada mini-batch. Proses pelatihan berhenti ketika Discriminator tidak dapat membedakan ρg dan ρdata yaitu D(x, ɵd) = ½ atau ketika ρg = ρdata.

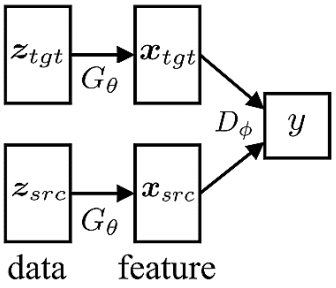
Contoh hasil yang didapatkan dari Algoritma ini adalah gambar dibawah ini

Kamar tidur yang dihasilkan

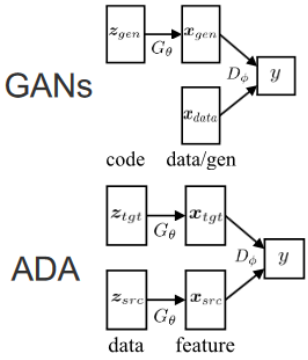
**Generative Adversarial Networks (GANs).**

Generative Adversarial Networks (GANs) mengintegrasikan pembelajaran permusuhan dan adaptasi domain dalam permainan dua pemain yang serupa dengan Generative Adversarial Networks (GANs). Diskriminator domain dipelajari dengan meminimalkan kesalahan klasifikasi dalam membedakan sumber dari domain target, sedangkan model klasifikasi mendalam mempelajari representasi yang dapat dipindahkan yang tidak dapat dibedakan oleh diskriminator domain. Setara dengan pendekatan tingkat fitur ini, model adaptasi tingkat piksel generatif melakukan penyelarasan distribusi dalam ruang piksel mentah, dengan menerjemahkan data sumber ke gaya domain target menggunakan teknik terjemahan Gambar ke Gambar. Baris pekerjaan lain menyelaraskan distribusi fitur dan kelas secara terpisah menggunakan diskriminator domain yang berbeda. Pembelajaran ini, ide kunci untuk mengaktifkan Generative Adversarial Networks (GANs) telah berhasil dieksplorasi untuk meminimalkan perbedaan lintas-domain

Generative Adversarial Networks (GANs) telah disematkan ke dalam jaringan yang dalam untuk mempelajari representasi yang dapat dipisahkan dan ditransfer untuk adaptasi domain. Metode ini yang ada mungkin tidak secara efektif menyelaraskan domain yang berbeda dari distribusi multimodal asli dalam masalah klasifikasi. Metode adaptasi domain menjembatani sumber dan target dengan mempelajari representasi fitur invarian atau memperkirakan kepentingan instance menggunakan data sumber berlabel dan data target tidak berlabel. Kemajuan terbaru dari metode adaptasi domain dalam memanfaatkan jaringan dalam untuk mempelajari representasi yang dapat ditransfer dengan menyematkan modul adaptasi dalam arsitektur yang dalam, sekaligus menguraikan faktor penjelas variasi di balik data dan pencocokan distribusi fitur di seluruh domain.



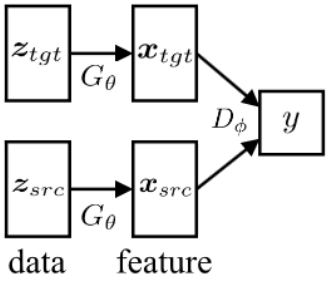

| Elements | GANs | ADA |
|---|---|---|
| $x$ | data/generation | features |
| $z$ | code vector | Data from src/tgt domains |
| $y$ | Real/fake indicator | Source/target domain indicator |




## 4.1 (ADA) ADVERSARIAL DOMAIN ADAPTATION

Adaptasi domain adversarial Metode adaptasi domain dalam mencoba menggeneralisasi jaringan saraf dalam di berbagai domain. Pendekatan yang paling umum digunakan didasarkan pada minimisasi perbedaan atau pelatihan permusuhan. Pelatihan permusuhan, terinspirasi oleh pemodelan generatif di GAN. Kerangka kerja ADA untuk identifikasi pemicu peristiwa. Ini terdiri dari tiga komponen: i) representasi pelajar (R) ii) event classifier (E) dan iii) prediktor domain (D). Pelajar representasi menghasilkan representasi tokenlevel, sedangkan pengklasifikasi peristiwa dan prediktor domain menggunakan representasi ini untuk mengidentifikasi pemicu peristiwa dan memprediksi domain tempat urutan tersebut berada. Ide kuncinya adalah untuk melatih pelajar representasi untuk menghasilkan representasi yang prediktif untuk identifikasi pemicu tetapi tidak prediktif untuk prediksi domain, sehingga lebih bersifat invarian domain. Manfaat penting di sini adalah bahwa satu-satunya data yang kami butuhkan dari domain target adalah data yang tidak berlabel. Untuk memastikan pembelajaran representasi domain-invarian, ADA menggunakan pelatihan.

- Data $z$ dari dua domain dilambangkan dengan $y \in \{0,1\}$
  - Domain sumber *($y = 1$)*
  - Domain target *($y = 0$)*
- ADA mentransfer pengetahuan prediksi yang dipelajari dari domain sumber ke domain target
  - Pelajari ekstraktor fitur $G_\theta$: $\boldsymbol{x} = G_\theta(\boldsymbol{z})$
  - Ingin *x* tidak dapat dibedakan oleh pembeda domain: $D_\phi(\boldsymbol{x})$
- Aplikasi dalam klasifikasi
  - Misalnya, kami memiliki label data domain sumber
  - Latih classifier melalui *x* dari data domain sumber untuk memprediksi label
  - *x* adalah invarian domain ⇒ *x* bersifat prediktif untuk data domain target




- Data Training $D_\phi$ untuk membedakan antar domain

$$\max_{\boldsymbol{\phi}} \mathcal{L}_\phi = \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=1)} [\log D_\phi(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=0)} [\log(1 - D_\phi(\boldsymbol{x}))]$$

- Data Training $G_\theta$ untuk membohongi $D_\phi$

$$\max_{\boldsymbol{\theta}} \mathcal{L}_\theta = \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=1)} [\log(1 - D_\phi(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=0)} [\log D_\phi(\boldsymbol{x})]$$

**Formula Baru Dari ADA**

Pengembangan formula baru ini berfungsi untuk mengungkap hubungan dengan pendekatan variasional konvensional, mari kita tulis ulang tujuan dalam format yang menyerupai EM variasional

- Distribusi implisit berakhir $\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|y)$

  $\boldsymbol{x} = G_\theta(\boldsymbol{z}),\ \boldsymbol{z} \sim p(\boldsymbol{z}|y)$
- Distribusi diskriminator $q_\phi(y|\boldsymbol{x})$

  $q_\phi^r(y|\boldsymbol{x}) = q_\phi(1 - y|\boldsymbol{x})$
- Tulis ulang tujuan dalam bentuk baru (hingga faktor skala konstan)

  $\max_{\boldsymbol{\phi}} \mathcal{L}_\phi = \mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)} [\log q_\phi(y|\boldsymbol{x})]$

  $\max_{\boldsymbol{\theta}} \mathcal{L}_\theta = \mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)} [\log q_\phi^r(y|\boldsymbol{x})]$
  - *z* dikemas dalam distribusi implisit $p_\theta(\boldsymbol{x}|y)$

$$\begin{aligned}\max_{\boldsymbol{\phi}} \mathcal{L}_\phi &= \mathbb{E}_{p_\theta(\boldsymbol{x}|y=0)p(y=0)} [\log q_\phi(y=0|\boldsymbol{x})] + \mathbb{E}_{p_\theta(\boldsymbol{x}|y=1)p(y=1)} [\log q_\phi(y=1|\boldsymbol{x})] \\ &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=0)} [\log(1 - D_\phi(\boldsymbol{x}))] + \frac{1}{2}\mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=1)} [\log D_\phi(\boldsymbol{x})]\end{aligned}$$

(Abaikan faktor skala konstanta 1/2) maka akan kita dapat formulasi baru satu-satunya perbedaan antara $\theta$ dan $\phi$: $q$ vs. $q^r$

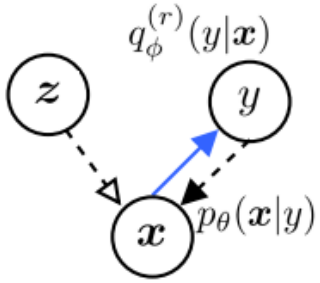
$$\max_{\boldsymbol{\phi}} \mathcal{L}_{\phi} = \mathbb{E}_{p_{\theta}(\boldsymbol{x}|y)p(y)} \left[\log q_{\phi}(y|\boldsymbol{x})\right]$$
$$\max_{\boldsymbol{\theta}} \mathcal{L}_{\theta} = \mathbb{E}_{p_{\theta}(\boldsymbol{x}|y)p(y)} \left[\log q_{\phi}^{r}(y|\boldsymbol{x})\right]$$

- Di sinilah mekanisme permusuhan muncul

Perbandingan antara ADA dan Variasi EM akan kita lihat pada table dibawah ini

| EM Variasi | ADA |
|---|---|
| • Tujuan<br>$\max_{\phi} \mathcal{L}_{\boldsymbol{\phi},\theta} = \mathbb{E}_{q_{\phi}(z\|x)}[\log p_{\theta}(x\|z)] + KL\left(q_{\phi}(z\|x)\|\|p(z)\right)$<br>$\max_{\theta} \mathcal{L}_{\boldsymbol{\phi},\theta} = \mathbb{E}_{q_{\phi}(z\|x)}[\log p_{\theta}(x\|z)] + KL\left(q_{\phi}(z\|x)\|\|p(z)\right)$<br>• Tujuan tunggal untuk Ç dan Å<br>• Regularisasi ekstra sebelumnya oleh $p(z)$<br>• Istilah rekonstruksi: maksimalkan kemungkinan log bersyarat dari $x$ dengan distribusi generatif $p\theta\ (x\|z)$ pengkondisian pada kode laten $z$ disimpulkan oleh $q\phi\ (z\|x)$<br>• $p\theta\ (x\|z)$ adalah model generatif<br>• $q\phi\ (z\|x)$ adalah model inferensi | • Tujuan<br>$\max_{\boldsymbol{\phi}} \mathcal{L}_{\phi} = \mathbb{E}_{p_{\theta}(\boldsymbol{x}\|y)p(y)} \left[\log q_{\phi}(y\|\boldsymbol{x})\right]$<br>$\max_{\boldsymbol{\theta}} \mathcal{L}_{\theta} = \mathbb{E}_{p_{\theta}(\boldsymbol{x}\|y)p(y)} \left[\log q_{\phi}^{r}(y\|\boldsymbol{x})\right]$<br>• Dua tujuan<br>• Memiliki keadaan optimal global dalam pandangan teori permainan<br>• Tujuan:memaksimalkan kemungkinan log bersyarat dari $y$ (atau $1 - y$) dengan distribusi $q\phi\ (y\|x)$ pengkondisian pada fitur laten $x$ disimpulkan oleh $p\theta\ (x\|y)$<br>• Tafsirkan $q\phi\ (y\|x)$ sebagai model generatif<br>• Interpretasikan $p\theta\ (x\|y)$ sebagai model inferensi |

**ADA dalam permodelan grafis**



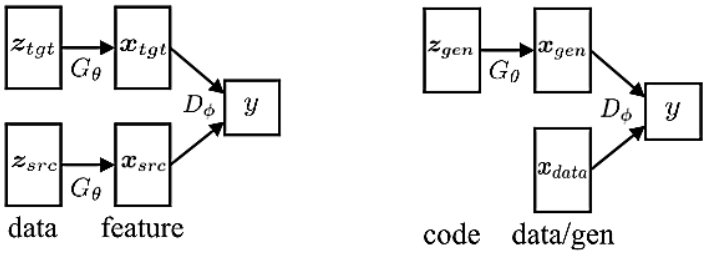

Panah garis padat ($x \rightarrow y$) merupakan proses generatif, sedangkan panah garis putus-putus (y, z $\rightarrow x$) adalah interferensi. Sedangkan Tanda panah berongga (z $\rightarrow x$) berarti transformasi deterministik atau mengarah ke distribusi implisit. Panah biru ($x \rightarrow y$) berarti mekanisme advisarial melibatkan $q_{\phi}(y|\boldsymbol{x})$ dan $q_{\phi}^{r}(y|\boldsymbol{x})$

## 4.2 VARIAN DARI GAN

GAN bersyarat digunakan dalam berbagai tugas seperti pembuatan teks ke gambar, terjemahan gambar ke gambar, penandaan gambar otomatis, dll. Dalam kasus ini mentransfer properti domain sumber ke domain target dapat

- Domain sumber: mis. bayangan nyata, $y$ = 1

- Target domain: mis. gambar yang dihasilkan, $y = 0$




- Distribusi implisit berakhir $\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|y)$

$$p_\theta(\boldsymbol{x}|y) = \begin{cases} p_{g_\theta}(\boldsymbol{x}) & y = 0 \quad \text{(distribution of generated images)} \\ p_{data}(\boldsymbol{x}) & y = 1. \quad \text{(distribution of real images)} \end{cases}$$

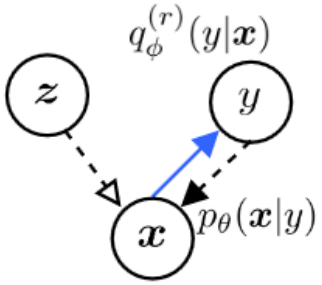
  - $\boldsymbol{x} \sim p_{g_\theta}(\boldsymbol{x}) \Leftrightarrow \boldsymbol{x} = G_\theta(\boldsymbol{z}),\ \boldsymbol{z} \sim p(\boldsymbol{z}|y = 0)$
  - $\boldsymbol{x} \sim p_{data}(\boldsymbol{x})$
- ruang kode dari *z* mengalami degenerasi dengan menggunakan sampel langsung dari data

Tugas diskriminator adalah membedakan gambar asli dari gambar yang dihasilkan (palsu), sedangkan jaringan generator mencoba mengelabui diskriminator dengan menghasilkan gambar yang semakin realistis. Namun, jika generator terlalu mudah atau terlalu sulit untuk dikelabui, mungkin gagal memberikan sinyal pembelajaran yang berguna untuk generator, oleh karena itu melatih GAN biasanya dianggap sebagai tugas yang sulit.

Namun, menerapkan augmentasi data ke GAN tidaklah mudah. Karena generator diperbarui menggunakan gradien diskriminator, jika gambar yang dihasilkan ditambah, pipa augmentasi harus dapat dibedakan dan juga harus kompatibel dengan GPU untuk efisiensi komputasi. Sekali lagi, tulis ulang tujuan GAN dalam format "variational-EM".

- Rekap: formulasi konvensional:

$$\max_{\boldsymbol{\phi}} \mathcal{L}_\phi = \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=0)} [\log(1 - D_\phi(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})} [\log D_\phi(\boldsymbol{x})]$$
$$\max_{\boldsymbol{\theta}} \mathcal{L}_\theta = \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=0)} [\log D_\phi(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})} [\log(1 - D_\phi(\boldsymbol{x}))]$$
$$= \mathbb{E}_{\boldsymbol{x}=G_\theta(\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z}|y=0)} [\log D_\phi(\boldsymbol{x})]$$




Tulis ulang dalam bentuk baru

$$\max_{\boldsymbol{\phi}} \mathcal{L}_\phi = \mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)} [\log q_\phi(y|\boldsymbol{x})]$$
$$\max_{\boldsymbol{\theta}} \mathcal{L}_\theta = \mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)} [\log q_\phi^r(y|\boldsymbol{x})]$$

**GAN: meminimalkan Kullback-Leibler Divergence (KDL)**

Divergensi Kullback-Leibler (KL) dapat dihitung dengan dua cara, maju atau mundur, dan karenanya asimetris. Bergantung pada apakah distribusinya kontinu atau diskrit, divergensi KL maju mereka adalah sebagai berikut. Seperti pada EM Variasi, kita dapat menulis ulang lebih lanjut dalam bentuk meminimalkan KLD untuk mengungkap lebih banyak wawasan tentang masalah pengoptimalan

- Untuk setiap langkah pengoptimalan dari $p_\theta(\boldsymbol{x}|y)$ pada titik $(\theta = \theta_0, \phi = \phi_0)$, membiarkan

  $p(y)$: distribusi awal yang seragam

  $p_{\theta=\theta_0}(\boldsymbol{x}) = \mathrm{E}_{p(y)}\left[p_{\theta=\theta_0}(\boldsymbol{x}|y)\right]$

  $q^r(\boldsymbol{x}|y) \propto q^r_{\phi=\phi_0}(y|\boldsymbol{x})p_{\theta=\theta_0}(\boldsymbol{x})$
- Lemma 1: Pembaruan $\theta$ di $\theta_0$ miliki

  $$\nabla_\theta\left[-\mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)}\left[\log q^r_{\phi=\phi_0}(y|\boldsymbol{x})\right]\right]\Big|_{\boldsymbol{\theta=\theta_0}} =$$

  $$\nabla_\theta\left[\mathbb{E}_{p(y)}\left[KL\left(p_\theta(\boldsymbol{x}|y)\|q^r(\boldsymbol{x}|y)\right)\right] - JSD\left(p_\theta(\boldsymbol{x}|y=0)\|p_\theta(\boldsymbol{x}|y=1)\right)\right]\Big|_{\boldsymbol{\theta=\theta_0}}$$

  - KL: divergensi KL
  - JSD: divergensi Jensen-shannon

**Bukti Lemma 1**

Bukti

$$\mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)}\left[\log q^r(y|\boldsymbol{x})\right] =$$
$$-\mathbb{E}_{p(y)}\left[\mathrm{KL}\left(p_\theta(\boldsymbol{x}|y)\|q^r(\boldsymbol{x}|y)\right) - \mathrm{KL}(p_\theta(\boldsymbol{x}|y)\|p_{\theta_0}(\boldsymbol{x}))\right],$$

Di mana

$$\mathbb{E}_{p(y)}\left[\mathrm{KL}(p_\theta(\boldsymbol{x}|y)\|p_{\theta_0}(\boldsymbol{x}))\right]$$
$$= p(y=0)\cdot\mathrm{KL}\left(p_\theta(\boldsymbol{x}|y=0)\|\frac{p_{\theta_0}(\boldsymbol{x}|y=0)+p_{\theta_0}(\boldsymbol{x}|y=1)}{2}\right)$$
$$+ p(y=1)\cdot\mathrm{KL}\left(p_\theta(\boldsymbol{x}|y=1)\|\frac{p_{\theta_0}(\boldsymbol{x}|y=0)+p_{\theta_0}(\boldsymbol{x}|y=1)}{2}\right)$$

Perhatikan bahwa $p_\theta(x|y=0) = p_{g\theta}(x)$ dan $p_\theta(x|y=1) = p_{data}(x)$ Membiarkan $p_{Me} = \frac{P_{9\theta}+p_{data}}{2}$. Persamaan sebelumnya dapat disederhanakan menjadi:

$$\mathbb{E}_{p(y)}\left[\mathrm{KL}(p_\theta(\boldsymbol{x}|y)\|p_{\theta_0}(\boldsymbol{x}))\right] = \frac{1}{2}\mathrm{KL}\left(p_{g\theta}\|p_{M_{\theta_0}}\right) + \frac{1}{2}\mathrm{KL}\left(p_{data}\|p_{M_{\theta_0}}\right)$$

Di samping itu,

$$\begin{aligned}
\mathrm{JSD}(p_{g_\theta}\|p_{data}) &= \frac{1}{2}\mathbb{E}_{p_{g_\theta}}\left[\log\frac{p_{g\theta}}{p_{M_\theta}}\right] + \frac{1}{2}\mathbb{E}_{p_{data}}\left[\log\frac{p_{data}}{p_{M_\theta}}\right] \\
&= \frac{1}{2}\mathbb{E}_{p_{g_\theta}}\left[\log\frac{p_{g\theta}}{p_{M_{\theta_0}}}\right] + \frac{1}{2}\mathbb{E}_{p_{g_\theta}}\left[\log\frac{p_{M_{\theta_0}}}{p_{M_\theta}}\right] \\
&\quad + \frac{1}{2}\mathbb{E}_{p_{data}}\left[\log\frac{p_{data}}{p_{M_{\theta_0}}}\right] + \frac{1}{2}\mathbb{E}_{p_{data}}\left[\log\frac{p_{M_{\theta_0}}}{p_{M_\theta}}\right] \\
&= \frac{1}{2}\mathbb{E}_{p_{g_\theta}}\left[\log\frac{p_{g\theta}}{p_{M_{\theta_0}}}\right] + \frac{1}{2}\mathbb{E}_{p_{data}}\left[\log\frac{p_{data}}{p_{M_{\theta_0}}}\right] + \mathbb{E}_{p_{M_\theta}}\left[\log\frac{p_{M_{\theta_0}}}{p_{M_\theta}}\right] \\
&= \frac{1}{2}\mathrm{KL}\left(p_{g_\theta}\|p_{M_{\theta_0}}\right) + \frac{1}{2}\mathrm{KL}\left(p_{data}\|p_{M_{\theta_0}}\right) - \mathrm{KL}\left(p_{M_\theta}\|p_{M_{\theta_0}}\right).
\end{aligned}$$

Perhatikan bahwa

$$\nabla_\theta \mathrm{KL}\left(p_{M_\theta} \| p_{M_{\theta_0}}\right)|_{\theta=\theta_0} = 0.$$

Mengambil turunan dari Persamaan w.r.t $\theta$ di $\theta_0$ kita dapatkan

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{p(y)}\left[\mathrm{KL}(p_\theta(\boldsymbol{x}|y)\|p_{\theta_0}(\boldsymbol{x}))\right]|_{\theta=\theta_0} \\
&= \nabla_\theta\left(\frac{1}{2}\mathrm{KL}\left(p_{g_\theta}\|p_{M_{\theta_0}}\right)|_{\theta=\theta_0} + \frac{1}{2}\mathrm{KL}\left(p_{data}\|p_{M_{\theta_0}}\right)\right)|_{\theta=\theta_0} \\
&= \nabla_\theta \mathrm{JSD}(p_{g_\theta}\|p_{data})|_{\theta=\theta_0}\ .
\end{aligned}$$

Mengambil turunan dari kedua sisi Persamaan di *w.r.t* $\theta$ di $\theta_0$ dan memasukkan persamaan terakhir dari Persamaan , kita mendapatkan hasil yang diinginkan.

Lemma 1: Pembaruan $\theta$ di $\theta_0$ miliki Koneksi ke inferensi variasional

$$\nabla_\theta\left[-\mathbb{E}_{p_\theta(\boldsymbol{x}|y)p(y)}\left[\log q^r_{\phi=\phi_0}(y|\boldsymbol{x})\right]\right]\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} =$$

$$\nabla_\theta\left[\mathbb{E}_{p(y)}\left[\mathrm{KL}\left(p_\theta(\boldsymbol{x}|y)\|q^r(\boldsymbol{x}|y)\right)\right] - \mathrm{JSD}\left(p_\theta(\boldsymbol{x}|y=0)\|p_\theta(\boldsymbol{x}|y=1)\right)\right]\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}$$

- Lihat *x* sebagai variabel laten, $y$ sebagai terlihat
- $p_{\theta=\theta_0}(\boldsymbol{x})$: distribusi sebelumnya
- $q^r(\boldsymbol{x}|y) \propto q^r_{\phi=\phi_0}(y|\boldsymbol{x})p_{\theta=\theta_0}(\boldsymbol{x})$ : distribusi posterior
- $p_\theta(\boldsymbol{x}|y)$: distribusi variasional
  - Inferensi diamortisasi: memperbarui parameter model $\theta$

Menyarankan hubungan dengan VAE, seperti yang akan kita bahas sebentar lagi
Meminimalkan drive KLD $p_{g_\theta}(\boldsymbol{x})$ to $p_{data}(\boldsymbol{x})$

- Menurut definisi: $p_{\theta=\theta_0}(\boldsymbol{x}) = \mathrm{E}_{p(y)}[p_{\theta=\theta_0}(\boldsymbol{x}|y)] = \left(p_{g_{\theta=\theta_0}}(\boldsymbol{x}) + p_{data}(\boldsymbol{x})\right)/2$

  $\mathrm{KL}(p_\theta(x|y=1)||q^r(x|y=1)) = \mathrm{KL}(p_{data}(x)||q^r(x|y=1))$ : konstan, tidak ada parameter bebas

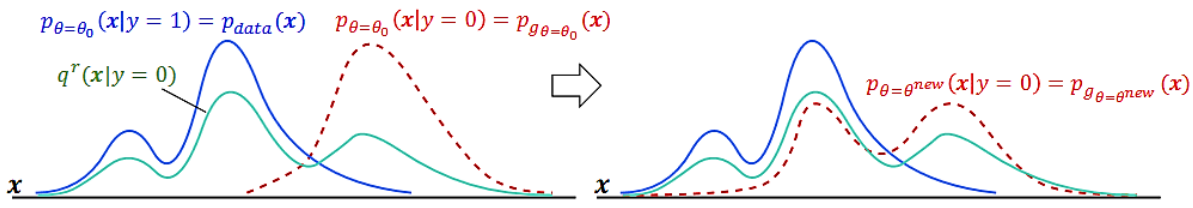
  $\mathrm{KL}(p_\theta(x|y=0)||q^r(x|y=0)) = \mathrm{KL}\left(p_{g_\theta}(x)||q^r(x|y=0)\right)$ : parameter $\theta$ untuk mengoptimalkan

$$q^r(\boldsymbol{x}|y=0) \propto q^r_{\phi=\phi_0}(y=0|\boldsymbol{x})p_{\theta=\theta_0}(\boldsymbol{x})$$

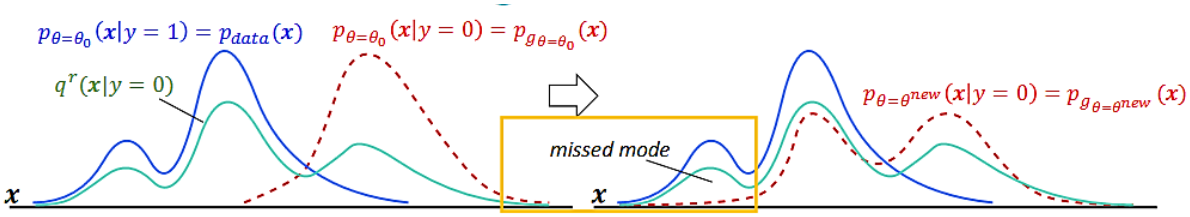
  - dilihat sebagai campuran dari $p_{g_{\theta=\theta_0}}(\boldsymbol{x})$ dan $p_{data}(\boldsymbol{x})$
  - berat pencampuran yang diinduksi dari $q^r_{\phi=\phi_0}(y=0|\boldsymbol{x})$
- Drive $p_{g_\theta}(\boldsymbol{x}|y)$ untuk campuran $p_{g_{\theta=\theta_0}}(\boldsymbol{x})$ dan $p_{data}(\boldsymbol{x})$

  ⇒ Drive $p_{g_\theta}(\boldsymbol{x})$ untuk $p_{data}(\boldsymbol{x})$

Dari perhitungan diatas maka akan ditemukan grafik seperti dibawah ini



Fenomena mode GAN yang hilang




- Asimetri KLD

Asimetri KLD adalah kekurangan untuk deteksi kerusakan, untuk mengatasi kekurangan ini, dua divergensi lain dan satu distribusi statistik diusulkan. Kemudian identifikasi kerusakan oleh KLD dan ketiga gambarannya dari sudut pandang simetris diselidiki. Berkonsentrasi $p_\theta(x|y=0)$ hingga mode besar $q^r(x|y) \to p_{g\theta}(x)$ merindukan mode $p_{data}(x)$

- Simetri JSD

Jensen-Shannon Divergence (JSD) adalah ukuran yang digunakan untuk mengukur perbedaan antara dua distribusi probabilitas, memainkan peran penting dalam pembelajaran mesin, statistik, dan pemrosesan sinyal. Keberhasilan tugas-tugas ini sangat bergantung pada pemilihan ukuran divergensi yang sesuai. Sementara banyak divergensi telah diusulkan dan dianalisis, ada kekurangan kriteria objektif untuk memilih divergensi yang optimal untuk tugas tertentu. Tidak mempengaruhi perilaku mode yang hilang

$$\mathrm{KL}\left(p_{g_\theta}(x)||q^r(x|y=0)\right) = \int p_{g_\theta}(x) \log \frac{p_{g_\theta}(x)}{q^r(x|y=0)} dx$$

Kontribusi positif yang besar bagi KLD di daerah ruang $x$ di mana $q^r(x|y=0)$ kecil, kecuali $p_{g_\theta}(x)$ juga kecil.

- Lemma 1: Pembaruan $\theta$ di $\theta_0$ miliki

$$\nabla_\theta \left[ -\mathbb{E}_{p_\theta(x|y)p(y)} \left[\log q^r_{\phi_0}(y|x)\right] \right]\Big|_{\theta=\theta_0} =$$
$$\nabla_\theta \left[\mathbb{E}_{p(y)} \left[KL\left(p_\theta(x|y)||q^r(x|y)\right)\right] - JSD\left(p_\theta(x|y=0)||p_\theta(x|y=1)\right)\right]\Big|_{\theta=\theta_0}$$

- Tidak ada asumsi tentang diskriminator optimal $q^r_{\phi_0}(y|x)$
  - Hasil sebelumnya biasanya mengandalkan diskriminator (hampir) optimal

    $q^*(y=1|x) = p_{data}(x)/(p_{data}(x) + p_g(x))$

    Asumsi optimalitas tidak praktis: ekspresi $D_\phi$ terbatas. Hasil kami adalah generalisasi dari teorema sebelumnya. Masukkan diskriminator optimal ke dalam persamaan di atas, kita pulihkan teorema

$$\nabla_\theta \left[ -\mathbb{E}_{p_\theta(x|y)p(y)} \left[\log q^r_{\phi_0}(y|x)\right] \right]\Big|_{\theta=\theta_0} = \nabla_\theta \left[\frac{1}{2}\mathrm{KL}\left(p_{g_\theta}||p_{data}\right) - \mathrm{JSD}\left(p_{g_\theta}||p_{data}\right)\right]\Big|_{\theta=\theta_0}$$

**InfoGAN**

InfoGAN adalah jenis jaringan permusuhan generatif yang memodifikasi tujuan GAN untuk mendorongnya mempelajari representasi yang dapat ditafsirkan dan bermakna. Ini

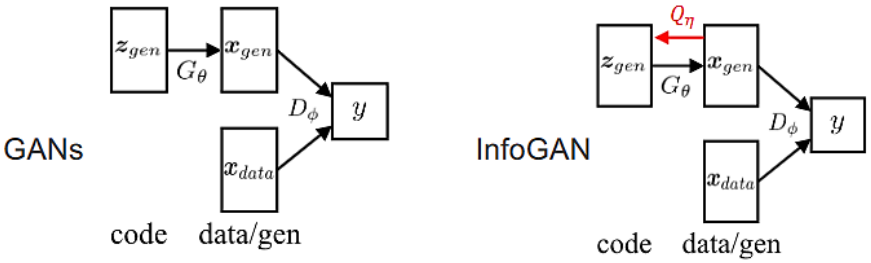
dilakukan dengan memaksimalkan informasi timbal balik antara subset kecil tetap dari variabel kebisingan GAN dan pengamatan.

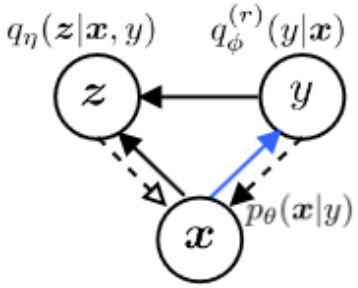
InfoGAN adalah ekstensi ke arsitektur GAN yang memperkenalkan variabel kontrol yang secara otomatis dipelajari oleh arsitektur dan memungkinkan kontrol atas gambar yang dihasilkan, seperti gaya, ketebalan, dan jenis dalam hal menghasilkan gambar dari angka tulisan tangan.

Model ini merupakan ekstensi teoretis-informasi ke Generative Adversarial Network yang mampu mempelajari representasi terurai dengan cara yang sepenuhnya tanpa pengawasan. InfoGAN adalah jaringan adversarial generatif yang juga memaksimalkan informasi timbal balik antara sebagian kecil variabel laten dan observasi.

Perkenalkan model inferensi $Q_\eta(z|x)$ dengan parameter $\boldsymbol{\eta}$. Tingkatkan tujuan GAN dengan menyimpulkan *z*

$$\max_D \mathcal{L}_D = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x\sim G(z),z\sim p(z)}[\log(1-D(x))],$$
$$\max_{G,Q} \mathcal{L}_{G,Q} = \mathbb{E}_{x\sim G(z),z\sim p(z)}[\log D(x) + \log Q(z|x)].$$



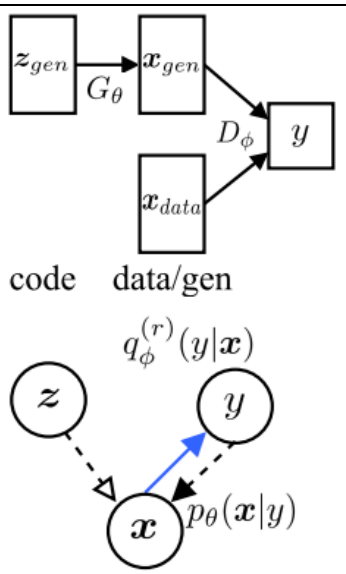
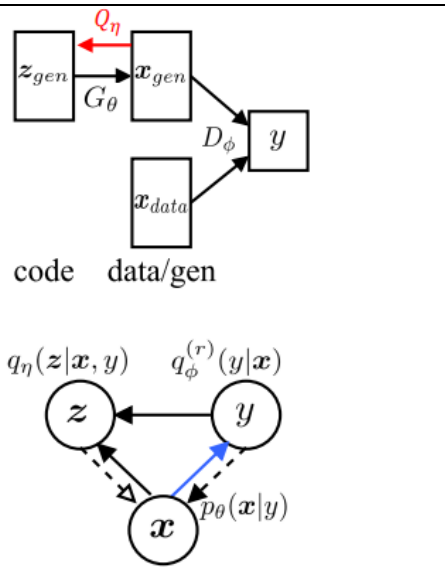

**formulasi baru InfoGAN**

Mendefinisikan bersyarat $q_\eta(z|x,y)$, $q_\eta(z|x,y=1)$diperbaiki tanpa parameter gratis untuk dipelajari. Karena GAN mengasumsikan ruang kode dari data nyata mengalami degenerasi. Parameter $\eta$ hanya diasosiasikan dengan $q_\eta(z|x,y=0)$

Maka akan dapat ditulis ulang dalam bentuk baru:

$$\max_{\boldsymbol{\phi}} \mathcal{L}_\phi = \mathbb{E}_{p_\theta(x|y)p(y)}[\log q_\eta(z|x,y)q_\phi(y|x)]$$
$$\max_{\boldsymbol{\theta},\boldsymbol{\eta}} \mathcal{L}_{\theta,\eta} = \mathbb{E}_{p_\theta(x|y)p(y)}[\log q_\eta(z|x,y)q_\phi^r(y|x)]$$

**Perbedaan antara GAN dan InfoGAN**

| GAN | InfoGAN |
|---|---|
|  |  |
| $\max_\phi \mathcal{L}_\phi = \mathbb{E}_{p_\theta(x\|y)p(y)}\left[\log q_\phi(y\|x)\right]$<br>$\max_\theta \mathcal{L}_\theta = \mathbb{E}_{p_\theta(x\|y)p(y)}\left[\log q_\phi^r(y\|x)\right]$ | $\max_\phi \mathcal{L}_\phi = \mathbb{E}_{p_\theta(x\|y)p(y)}\left[\log q_\eta(z\|x,y) q_\phi(y\|x)\right]$<br>$\max_{\theta,\eta} \mathcal{L}_{\theta,\eta} = \mathbb{E}_{p_\theta(x\|y)p(y)}\left[\log q_\eta(z\|x,y) q_\phi^r(y\|x)\right]$ |

- Hasil serupa seperti di GAN:
  - Membiarkan $q^r(x|z,y) \propto q_{\eta=\eta_0}(z|x,y) q^r_{\phi=\phi_0}(y|x) p_{\theta=\theta_0}(x)$
  - Kita punya:

$$\nabla_\theta \Big[ -\mathbb{E}_{p_\theta(x|y)p(y)} \left[\log q_{\eta_0}(z|x,y) q^r_{\phi_0}(y|x)\right] \Big]\Big|_{\theta=\theta_0} =$$

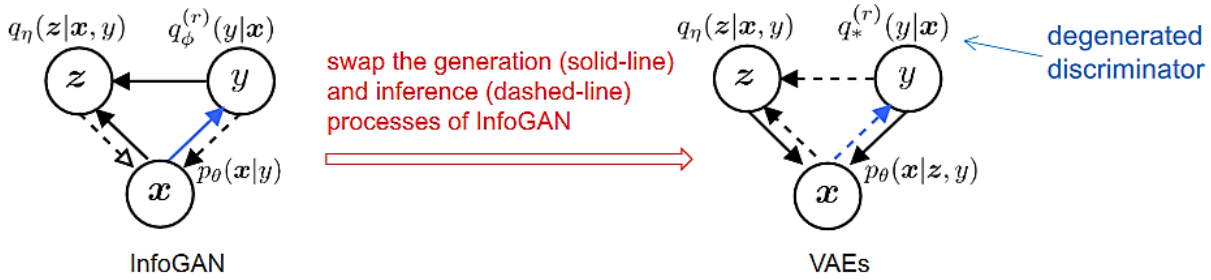
$$\nabla_\theta \Big[\mathbb{E}_{p(y)} \left[\mathrm{KL}\left(p_\theta(x|y) \| q^r(x|z,y)\right)\right] - \mathrm{JSD}\left(p_\theta(x|y=0) \| p_\theta(x|y=1)\right) \Big]\Big|_{\theta=\theta_0}$$

## 4.3 MENGHUBUNGKAN VAE DENGAN GAN

Autoencoder variasional (VAE) dan jaringan permusuhan generatif (GAN), telah mencapai kesuksesan luar biasa dalam menghasilkan dan memanipulasi gambar dimensi tinggi. VAE unggul dalam mempelajari representasi gambar yang tidak terurai, sementara GAN unggul dalam menghasilkan gambar yang realistis. Secara sistematis menilai kinerja pemisahan dan generasi pada data ekspresi gen sel tunggal dan menemukan bahwa kekuatan dan kelemahan VAE dan GAN ini berlaku untuk data ekspresi gen sel tunggal dengan cara yang serupa.

Dua jenis model generatif dalam yang paling banyak digunakan adalah autoencoder variasional (VAE) dan jaringan adversarial generatif (GAN). VAE menggunakan pendekatan Bayesian untuk memperkirakan distribusi posterior dari jaringan enkoder probabilistik, berdasarkan kombinasi kesalahan rekonstruksi dan probabilitas sebelumnya dari distribusi yang disandikan. Sebaliknya, kerangka kerja GAN terdiri dari permainan dua pemain antara jaringan generator dan jaringan diskriminator. GAN dan VAE memiliki kekuatan dan kelemahan yang saling melengkapi: GAN menghasilkan sampel yang jauh lebih baik daripada VAE, tetapi pelatihan VAE jauh lebih stabil dan mempelajari representasi laten "terurai" yang

lebih bermanfaat. Memang, VAE pada dasarnya meminimalkan KLD dengan arah yang berlawanan, dan dengan diskriminator permusuhan yang merosot. Lihat gambar dibawah ini




**VAE: formulasi baru**

- Asumsikan pembeda yang sempurna $q_*(y|x)$
  $q_*(y=1|x)=1$ jika *x* adalah contoh nyata
  $q_*(y=0|x)=1$ jika *x* dihasilkan sampel
  $q_*^r(y|x) := q_*(1-y|x)$
- Distribusi generatif

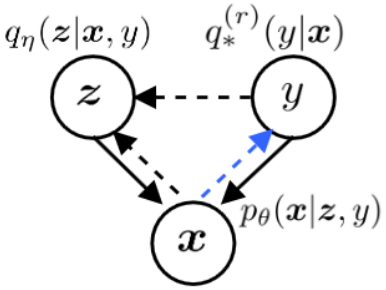
$$p_\theta(x|z,y) = \begin{cases} p_\theta(x|z) & y=0 \\ p_{data}(x) & y=1. \end{cases}$$

- Membiarkan
  $p_\theta(z,y|x) \propto p_\theta(x|z,y)p(z|y)p(y)$

**Lemma 2: sketsa pembuktian**

**Lemma 2**

$$\begin{aligned}\mathcal{L}_{\theta,\eta}^{vae} &= 2\cdot\mathbb{E}_{p_{\theta_0}(x)}\left[\mathbb{E}_{q_\eta(z|x,y)q_*^r(y|x)}\left[\log p_\theta(x|z,y)\right] - KL(q_\eta(z|x,y)q_*^r(y|x)\|p(z|y)p(y))\right] \\ &= 2\cdot\mathbb{E}_{p_{\theta_0}(x)}\left[-KL\left(q_\eta(z|x,y)q_*^r(y|x)\|p_\theta(z,y|x)\right)\right].\end{aligned}$$




Pembuktian dari rumus diatas adalah

1) Perluas $\mathbb{E}_{p_{\theta_0}(x)}[.] = \frac{1}{2}\mathbb{E}_{p_{\theta_0}(x|y=1)}[.] + \frac{1}{2}\mathbb{E}_{p_{\theta_0}(x|y=0)}[.]$
2) $\frac{1}{2}\mathbb{E}_{p_{\theta_0}(x|y=0)}[.]$ konstan
   - Karena pembeda yang sempurna $q_*^r(y|x)$
   - Memblokir sampel yang dihasilkan dalam kehilangan pelatihan
3) $\frac{1}{2}\mathbb{E}_{p_{\theta_0}(x|y=1)}[.] = \frac{1}{2}\mathbb{E}_{p_{data}(x)}[.]$

- Memulihkan formulasi konvensional

**Bukti Lemma 2**

Untuk istilah rekonstruksi:

$$\begin{aligned}
&\mathbb{E}_{p_{\theta_0}(\boldsymbol{x})}\left[\mathbb{E}_{q_\eta(\boldsymbol{z}|\boldsymbol{x},y)q_*^r(y|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z},y)\right]\right] \\
&= \frac{1}{2}\mathbb{E}_{p_{\theta_0}(\boldsymbol{x}|y=1)}\left[\mathbb{E}_{q_\eta(\boldsymbol{z}|\boldsymbol{x},y=0),y=0\sim q_*^r(y|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z},y=0)\right]\right] \\
&+ \frac{1}{2}\mathbb{E}_{p_{\theta_0}(\boldsymbol{x}|y=0)}\left[\mathbb{E}_{q_\eta(\boldsymbol{z}|\boldsymbol{x},y=1),y=1\sim q_*^r(y|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z},y=1)\right]\right] \\
&= \frac{1}{2}\mathbb{E}_{p_{data}(\boldsymbol{x})}\left[\mathbb{E}_{\tilde{q}_\eta(\boldsymbol{z}|\boldsymbol{x})}\left[\log \tilde{p}_\theta(\boldsymbol{x}|\boldsymbol{z})\right]\right] + const,
\end{aligned}$$

di mana $y=0\sim q_*^r(y|x)$ berarti $q_*^r(y|x)$ memprediksi *y=0* dengan probabilitas 1. Perhatikan bahwa $q_\eta(z|x,y=1)$ dan $p_\theta(x|z,y=1)$ adalah distribusi konstan tanpa parameter bebas untuk mempelajari $q_\eta(z|x,y=0)=\tilde{q}_\eta(z|x)$, dan $p_\theta(x|z,y=0)=\tilde{p}_\theta(x|z)$.

Untuk masa regularisasi KL sebelumnya:

$$\begin{aligned}
&\mathbb{E}_{p_{\theta_0}(\boldsymbol{x})}\left[\mathrm{KL}(q_\eta(\boldsymbol{z}|\boldsymbol{x},y)q_*^r(y|\boldsymbol{x})\|p(\boldsymbol{z}|y)p(y))\right] \\
&= \mathbb{E}_{p_{\theta_0}(\boldsymbol{x})}\left[\int q_*^r(y|\boldsymbol{x})\mathrm{KL}\left(q_\eta(\boldsymbol{z}|\boldsymbol{x},y)\|p(\boldsymbol{z}|y)\right)dy + \mathrm{KL}\left(q_*^r(y|\boldsymbol{x})\|p(y)\right)\right] \\
&= \frac{1}{2}\mathbb{E}_{p_{\theta_0}(\boldsymbol{x}|y=1)}\left[\mathrm{KL}\left(q_\eta(\boldsymbol{z}|\boldsymbol{x},y=0)\|p(\boldsymbol{z}|y=0)\right) + const\right] + \frac{1}{2}\mathbb{E}_{p_{\theta_0}(\boldsymbol{x}|y=1)}\left[const\right] \\
&= \frac{1}{2}\mathbb{E}_{p_{data}(\boldsymbol{x})}\left[\mathrm{KL}(\tilde{q}_\eta(\boldsymbol{z}|\boldsymbol{x})\|\tilde{p}(\boldsymbol{z}))\right].
\end{aligned}$$

**Persamaan antara InfoGAN dan VAEs**

| | GANs (InfoGAN) | VAEs |
|---|---|---|
| Generative distribution | $p_\theta(\boldsymbol{x}\|y)=\begin{cases}p_{g_\theta}(\boldsymbol{x}) & y=0\\ p_{data}(\boldsymbol{x}) & y=1.\end{cases}$ | $p_\theta(\boldsymbol{x}\|\boldsymbol{z},y)=\begin{cases}p_\theta(\boldsymbol{x}\|\boldsymbol{z}) & y=0\\ p_{data}(\boldsymbol{x}) & y=1.\end{cases}$ |
| Discriminator distribution | $q_\phi(y\|\boldsymbol{x})$ | $q_*(y\|\boldsymbol{x})$, perfect, degenerated |
| $z$-inference model | $q_\eta(\boldsymbol{z}\|\boldsymbol{x},y)$ of InfoGAN | $q_\eta(\boldsymbol{z}\|\boldsymbol{x},y)$ |
| KLD to minimize | $\min_\theta \mathrm{KL}\left(p_\theta(\boldsymbol{x}\|y)\,\|\|\,q^r(\boldsymbol{x}\|\boldsymbol{z},y)\right)$ $\sim \min_\theta \mathrm{KL}(P_\theta \,\|\|\, Q)$ | $\min_\theta \mathrm{KL}\left(q_\eta(\boldsymbol{z}\|\boldsymbol{x},y)q_*^r(y\|\boldsymbol{x})\,\|\|\,p_\theta(\boldsymbol{z},y\|\boldsymbol{x})\right)$ $\sim \min_\theta \mathrm{KL}(Q \,\|\|\, P_\theta)$ |

## 4.4 AAVAE (ADVERSARY ACTIVATED VAEs)

Rumus permodelan ini dapat ditulis dengan:

$$\max_{\boldsymbol{\theta},\boldsymbol{\eta}} \mathcal{L}^{\text{vae}}_{\theta,\eta} = \mathbb{E}_{p_{\theta_0}(\boldsymbol{x})}\left[\mathbb{E}_{q_\eta(\boldsymbol{z}|\boldsymbol{x},y)q_*^r(y|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z},y)\right] - \mathrm{KL}(q_\eta(\boldsymbol{z}|\boldsymbol{x},y)q_*^r(y|\boldsymbol{x})\|p(\boldsymbol{z}|y)p(y))\right]$$

Ganti $q_*(y|\boldsymbol{x})$ dengan yang dapat dipelajari $q_\phi(y|\boldsymbol{x})$ dengan parameter $\phi$

Seperti biasa, nyatakan distribusi terbalik $q_\phi^r(y|x)=q_\phi(y|x)$ maka

$$\max_{\theta,\eta} \mathcal{L}^{\text{aavae}}_{\theta,\eta} = \mathbb{E}_{p_{\theta_0}(x)} \left[ \mathbb{E}_{q_\eta(z|x,y) q^r_\phi(y|x)} [\log p_\theta(x|z,y)] - \mathrm{KL}(q_\eta(z|x,y) q^r_\phi(y|x) \| p(z|y)p(y)) \right]$$

Untuk prumusan pemilihan data adaptif

$$\max_{\theta,\eta} \mathcal{L}^{\text{aavae}}_{\theta,\eta} = \mathbb{E}_{p_{\theta_0}(x)} \left[ \mathbb{E}_{q_\eta(z|x,y) q^r_\phi(y|x)} [\log p_\theta(x|z,y)] - \mathrm{KL}(q_\eta(z|x,y) q^r_\phi(y|x) \| p(z|y)p(y)) \right]$$

Mekanisme pemilihan data yang efektif:

- Baik sampel yang dihasilkan maupun contoh nyata diberi pembobotan $q^r_\phi(y=0|x) = q_\phi(y=1|x)$
- Hanya sampel yang menyerupai data nyata dan mengecoh diskriminator yang akan digunakan untuk pelatihan
- Contoh nyata menerima bobot besar $q^r_\phi(y|x)$
  ⇒ Mudah dikenali oleh pembeda sebagai nyata
  ⇒ Sulit disimulasikan dari generator
  ⇒ Contoh sulit mendapatkan bobot yang lebih besar

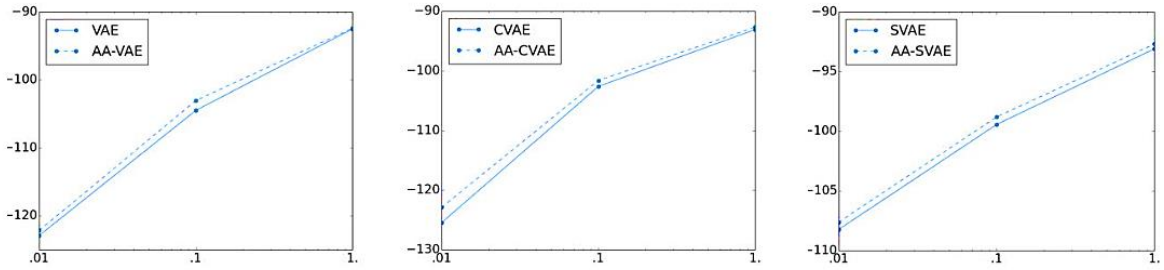
**AAVAE: pembelajaran diskriminator**

digunakan tujuan klasifikasi biner seperti di GAN, dapat dirumuskan:

$$\max_\phi \mathcal{L}^{\text{aavae}}_\phi = \mathbb{E}_{p_\theta(x|z,y)p(z|y)p(y)} [\log q_\phi(y|x)]$$

Pengujian AAVAE

- Batas bawah variasi rangkaian uji yang dievaluasi pada MNIST
  - Semakin tinggi semakin baik




Sumbu X: rasio data pelatihan untuk pembelajaran (0,01, 0,1, 1.)
Sumbu Y: nilai batas bawah test-set

Eksperimen menunjukkan bahwa model berbasis AA-SVAE secara drastis mengungguli model deterministik dalam hal presisi dan bahwa mekanisme rekonstruksi meningkatkan daya ingat model berbasis AA-SVAE tanpa mengorbankan presisi.

evaluasi akurasi klasifikasi SAVE dan A-SAVE

| | 1% | 10% |
|---|---|---|
| SVAE | 0.9412±.0039 | 0.9768±.0009 |
| AASVAE | **0.9425±.0045** | **0.9797±.0010** |

Menggunakan label data 1% dan 10% di MNIST

## 4.5 IWAE (IMPORTANCE WEIGHTED VAE)

Autoencoder tertimbang kepentingan (IWAE), model generatif yang berbagi arsitektur VAE, tetapi dilatih dengan batas bawah kemungkinan log yang lebih ketat yang berasal dari pembobotan kepentingan. Jaringan pengenalan menghasilkan beberapa sampel posterior perkiraan, dan bobotnya dirata-rata. Saat jumlah sampel meningkat, batas bawah mendekati kemungkinan log yang sebenarnya. Penggunaan banyak sampel memberi IWAE fleksibilitas tambahan untuk mempelajari model generatif yang distribusi posteriornya tidak sesuai dengan asumsi pemodelan VAE. Pendekatan ini terkait dengan reweighted wake sleep, tetapi IWAE dilatih menggunakan satu tujuan terpadu. Dibandingkan dengan VAE, IWAE kami dapat mempelajari representasi yang lebih kaya dengan dimensi laten yang lebih banyak, yang berarti kemungkinan log yang jauh lebih tinggi pada tolok ukur estimasi kepadatan.
Pada pembelajaran generator di variasi dari GAN dihitung dengan rumus:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim p_\theta(\boldsymbol{x}|y)p(y)} \left[\log q^r_{\phi_0}(y|\boldsymbol{x})\right]$$

Sedangkan untuk Pembelajaran generator di IWGAN

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k\sim p_\theta(\boldsymbol{x}|y)p(y)} \left[\sum_{i=1}^{k} \frac{q^r_{\phi_0}(y|\boldsymbol{x}_i)}{q_{\phi_0}(y|\boldsymbol{x}_i)} \log q^r_{\phi_0}(y|\boldsymbol{x}_i)\right]$$

Memberikan bobot yang lebih tinggi pada sampel yang lebih realistis dan membodohi pembeda dengan lebih baik

| | MNIST | SVHN |
|---|---|---|
| GAN | 8.34±.03 | 5.18±.03 |
| IWGAN | **8.45±.04** | **5.34±.03** |

- Akurasi klasifikasi sampel dari CGAN dan IW

| | MNIST | SVHN |
|---|---|---|
| CGAN | 0.985±.002 | 0.797±.005 |
| IWCGAN | **0.987±.002** | **0.798±.006** |

**Rekap: Inferensi Variasi**

Maksimalkan variasi batas bawah $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$, minimalkan energi bebas

$$F(\boldsymbol{\theta}, \phi; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

- E-langkah: maksimalkan $\mathcal{L}$ wrt. $\phi$ dengan $\theta$ tetap

$$\max_\phi \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + KL(q_\phi(z|x)||p(z))$$

  - Jika dengan solusi bentuk tertutup

$$q^*_\phi(z|x) \propto \exp[\log p_\theta(x, z)]$$

- M-langkah: maksimalkan $\mathcal{L}$ wrt. $\theta$ dengan $\phi$ tetap

$$\max_\theta \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + KL(q_\phi(z|x)||p(z))$$

# BAB 5
# INFERENSI DAN PEMBELAJARAN

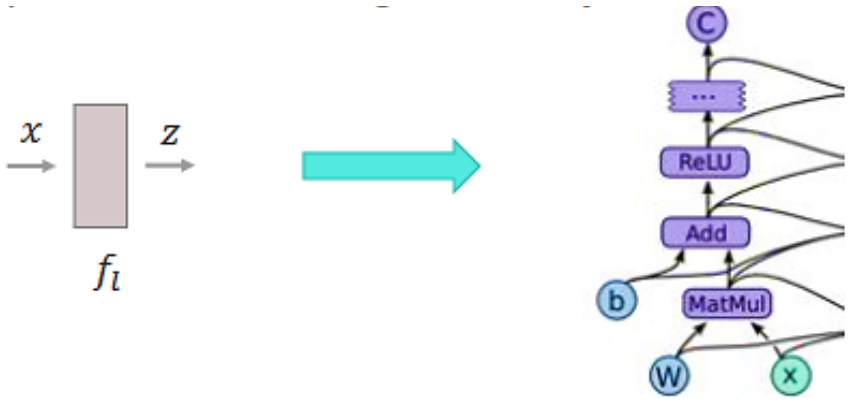
## 5.1 LAPISAN KOMPUTASI DI *DEEP LEARNING*

Neural Network dibangun dari 3 jenis lapisan:

1. Lapisan input — data awal untuk jaringan saraf.
2. Lapisan tersembunyi — lapisan perantara antara lapisan input dan output dan tempat di mana semua komputasi dilakukan.
3. Lapisan keluaran — menghasilkan hasil untuk masukan yang diberikan.

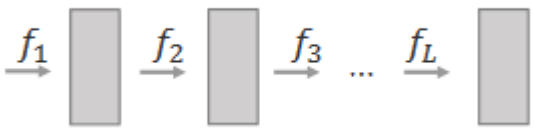
Setiap node terhubung dengan setiap node dari layer berikutnya dan setiap koneksi (panah hitam) memiliki bobot tertentu. Bobot dapat dilihat sebagai dampak yang dimiliki simpul tersebut pada simpul dari lapisan berikutnya. Lapisan dalam jaringan saraf terdiri dari beberapa operasi komputasi yang lebih baik

- Lapisan $l$ memiliki input $x$ dan output $z$, dan mengubah $x$ menjadi $z$ sebagai berikut:
  $y = Wx + b, z = ReLU(y)$
- Nyatakan transformasi layer $l$ sebagai f$l$, yang dapat direpresentasikan sebagai grafik aliran data: input $x$ mengalir melalui layer




**Dari Lapisan ke Jaringan**

- Sebuah neural network dengan demikian merupakan beberapa layer yang ditumpuk $l = 1, \ldots, L,$ di mana setiap layer merepresentasikan transformasi fungsi $f_l$
  - Komputasi maju berlangsung dengan mengeksekusi secara berurutan $f_1, f_2, f_3, \ldots, f_L$ maka akan digambarkan sebagai berkut:

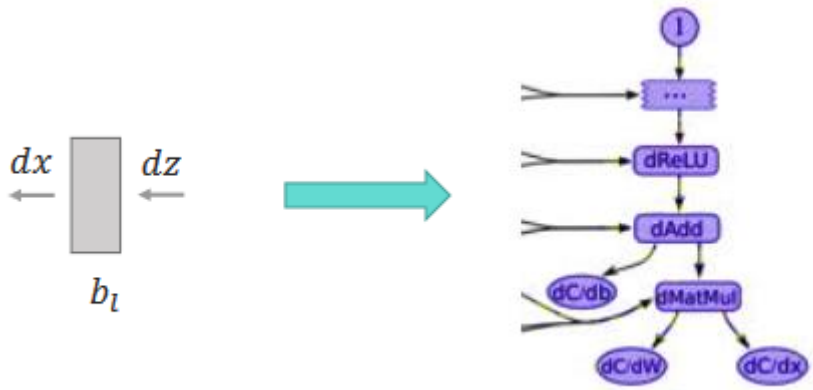



Melatih jaringan saraf meliputi menurunkan gradien parameternya dengan lintasan mundur (slide berikutnya)

**Lapisan Komputasi di DL**

Nyatakan lintasan mundur melalui lapisan $l$ sebagai $bl$

- $bl$ menurunkan gradien dari input $x$(d$x$), mengingat gradien dari $z$ sebagai d$z$, serta gradien parameter W,b
- d$x$ akan menjadi input mundur dari layer sebelumnya $l-1$
- Backward pass dapat dianggap sebagai aliran data mundur dimana gradien mengalir melalui layer

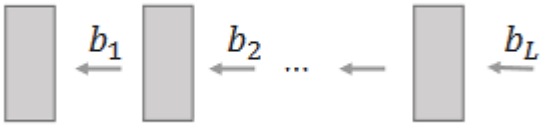



**Backpropagation melalui NN**

Backpropagation adalah salah satu konsep penting dari jaringan saraf. Tugas kita adalah mengklasifikasikan data kita dengan sebaik-baiknya. Untuk ini, kita harus memperbarui bobot parameter dan bias, tetapi bagaimana kita bisa melakukannya di jaringan saraf yang dalam? Dalam model regresi linier, kami menggunakan penurunan gradien untuk mengoptimalkan parameter. Demikian pula di sini kami juga menggunakan algoritma penurunan gradien menggunakan Backpropagation.

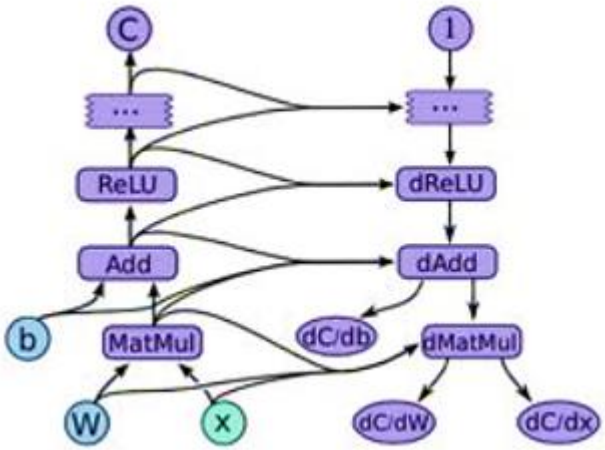
Untuk contoh pelatihan tunggal, algoritma Backpropagation menghitung gradien dari fungsi error. Backpropagation dapat ditulis sebagai fungsi dari jaringan saraf. Algoritma backpropagation adalah seperangkat metode yang digunakan untuk melatih jaringan syaraf tiruan secara efisien mengikuti pendekatan penurunan gradien yang mengeksploitasi aturan rantai.

Fitur utama Backpropagation adalah metode berulang, rekursif, dan efisien untuk menghitung bobot yang diperbarui untuk meningkatkan jaringan hingga tidak dapat melakukan tugas yang dilatihnya. Turunan dari fungsi aktivasi untuk diketahui pada saat perancangan jaringan diperlukan Backpropagation. Perhitungan mundur berlangsung dengan mengeksekusi secara berurutan $b_L, b_{L-1}, b_{L-2}, \ldots, b_1$



**Lapisan sebagai Grafik Aliran Data**

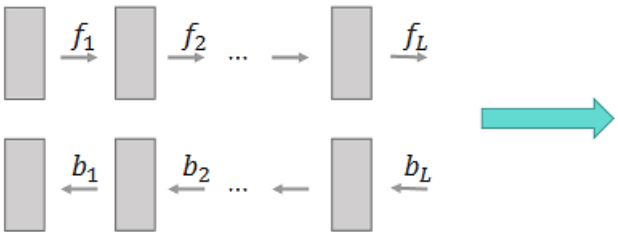
Kerangka pembelajaran mesin modern umumnya mengadopsi paradigma pemrograman gaya aliran data. Grafik aliran data dari suatu aplikasi memaparkan informasi berharga untuk mengoptimalkan aplikasi. Berikan aliran perhitungan maju, gradien dapat dihitung dengan diferensiasi otomatis. Secara otomatis mendapatkan grafik aliran gradien mundur dari grafik aliran data maju

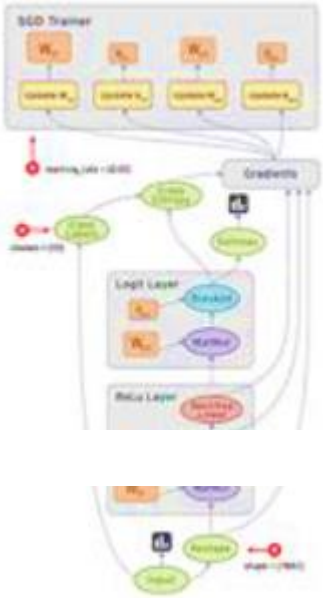



**Jaringan sebagai Grafik Aliran Data**

Diagram aliran jaringan memetakan aliran data melalui jaringan. Sistem digital sering melibatkan sistem yang terhubung ke jaringan dengan fungsionalitas yang didistribusikan di beberapa node.

Diagram aliran jaringan menunjukkan rute yang dilalui data, node internal dan eksternal tempat data disimpan atau diproses, dan tujuan dari node tersebut. Diagram alur jaringan sangat penting untuk memahami lingkungan yang menampung data sensitif serta mitigasi risiko dan penegakan kebijakan keamanan informasi. Gradien dapat dihitung dengan diferensiasi otomatis. Dan secara otomatis menurunkan grafik aliran gradien dari grafik aliran data maju



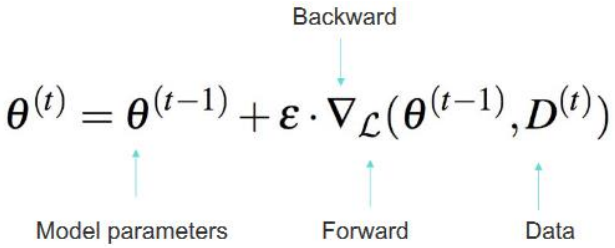


**Penurunan Gradien melalui Backpropagation**

Back-propagation disebut seperti ini karena untuk menghitung turunannya Anda menggunakan aturan rantai dari lapisan terakhir (yang terhubung langsung ke fungsi kerugian, karena merupakan salah satu yang memberikan prediksi) ke lapisan pertama, yaitu yang mengambil data masukan. Dalam penurunan gradien mencoba untuk mencapai fungsi kerugian minimum sehubungan dengan parameter menggunakan turunan yang dihitung dalam propagasi balik. Cara termudah adalah menyesuaikan parameter dengan mengurangkan turunannya yang sesuai dikalikan dengan laju pembelajaran, yang mengatur seberapa banyak Anda ingin bergerak dalam arah gradien. Alur kerja komputasi deep learning

- Forward, yang biasanya juga kita sebut inference: forward dataflow
- Mundur, yang menurunkan gradien: aliran gradien mundur
- Terapkan/perbarui gradien dan ulangi

Secara matematis,




## 5.2 MEMPROGRAM JARINGAN SARAF

Jaringan saraf adalah sistem buatan yang terinspirasi oleh jaringan saraf biologis. Sistem ini belajar melakukan tugas dengan terpapar ke berbagai kumpulan data dan contoh tanpa aturan khusus tugas apa pun. Idenya adalah bahwa sistem menghasilkan karakteristik pengenal dari data yang telah mereka lewati tanpa diprogram dengan pemahaman yang telah diprogram sebelumnya dari kumpulan data ini. Jaringan saraf didasarkan pada model

komputasi untuk logika ambang batas. Threshold logic adalah kombinasi dari algoritma dan matematika. Jaringan saraf didasarkan pada studi otak atau penerapan jaringan saraf untuk kecerdasan buatan. Pekerjaan telah menyebabkan perbaikan dalam teori automata terbatas. Komponen jaringan saraf yang khas melibatkan neuron, koneksi yang dikenal sebagai sinapsis, bobot, bias, fungsi propagasi, dan aturan pembelajaran. Menentukan operasi dan lapisan: terhubung sepenuhnya? Lilitan? Berulang?

Lapisan konvolusi adalah komponen mendasar dari arsitektur CNN yang melakukan ekstraksi fitur, yang biasanya terdiri dari kombinasi operasi linier dan nonlinier. Konvolusi adalah jenis operasi linier khusus yang digunakan untuk ekstraksi fitur, di mana sejumlah kecil angka, yang disebut kernel, diterapkan pada input, yang merupakan larik angka, yang disebut tensor. Produk elemen-bijaksana antara setiap elemen kernel dan tensor input dihitung di setiap lokasi tensor dan dijumlahkan untuk mendapatkan nilai output pada posisi tensor output yang sesuai,

Operasi konvolusi yang dijelaskan di atas tidak memungkinkan bagian tengah setiap kernel tumpang tindih dengan elemen terluar tensor masukan, dan mengurangi tinggi dan lebar peta fitur keluaran dibandingkan dengan tensor masukan. Padding, biasanya zero padding, adalah teknik untuk mengatasi masalah ini, di mana baris dan kolom nol ditambahkan di setiap sisi tensor input, agar pas dengan bagian tengah kernel pada elemen terluar dan mempertahankan in-plane yang sama dimensi melalui operasi konvolusi.

Lapisan penyatuan menyediakan operasi downsampling tipikal yang mengurangi dimensi in-plane dari peta fitur untuk memperkenalkan invarian terjemahan ke pergeseran kecil dan distorsi, dan mengurangi jumlah parameter yang dapat dipelajari berikutnya. Perlu dicatat bahwa tidak ada parameter yang dapat dipelajari di salah satu lapisan penyatuan, sedangkan ukuran filter, langkah, dan padding adalah hyperparameter dalam operasi penyatuan, mirip dengan operasi konvolusi.

Fully connected layer Peta fitur keluaran dari konvolusi akhir atau pooling layer biasanya diratakan, yaitu diubah menjadi array angka (atau vektor) satu dimensi (1D), dan terhubung ke satu atau lebih lapisan yang terhubung sepenuhnya, juga dikenal sebagai lapisan padat, di mana setiap input terhubung ke setiap output dengan bobot yang dapat dipelajari. Setelah fitur yang diekstrak oleh lapisan konvolusi dan diturunkan sampelnya oleh lapisan penyatuan dibuat, fitur tersebut dipetakan oleh subset dari lapisan yang terhubung sepenuhnya ke keluaran akhir jaringan, seperti probabilitas untuk setiap kelas dalam tugas klasifikasi.

Penurunan gradien biasanya digunakan sebagai algoritme pengoptimalan yang secara literal memperbarui parameter yang dapat dipelajari, yaitu kernel dan bobot, dari jaringan untuk meminimalkan kerugian.

Gradien dari fungsi kerugian memberi kita arah di mana fungsi tersebut memiliki laju peningkatan paling tajam, dan setiap parameter yang dapat dipelajari diperbarui ke arah negatif gradien dengan ukuran langkah arbitrer yang ditentukan berdasarkan hyperparameter yang disebut laju pembelajaran. Gradien, secara matematis, merupakan turunan parsial dari kerugian sehubungan dengan setiap parameter yang dapat dipelajari. Dalam praktiknya, untuk alasan seperti keterbatasan memori, gradien fungsi kerugian terkait dengan parameter

dihitung dengan menggunakan subset dari dataset pelatihan yang disebut mini-batch, dan diterapkan ke pembaruan parameter.

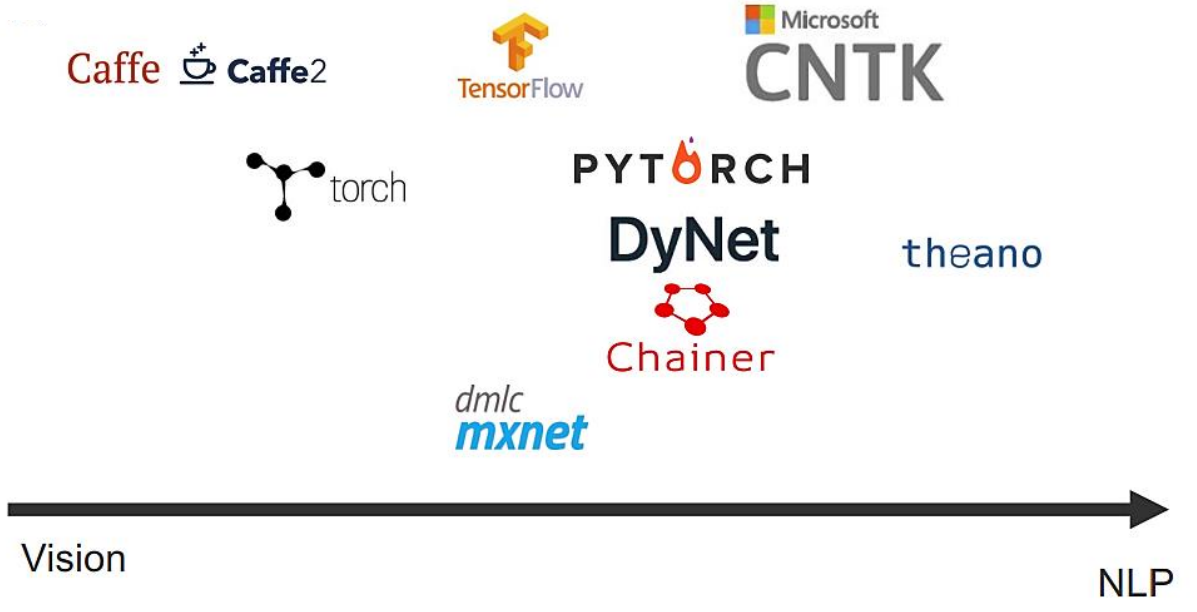
**Pustaka diferensial otomatis**

Auto-diferensial Pustaka secara otomatis memperoleh gradien mengikuti aturan propagasi balik. Kontribusi diferensiasi otomatis pada jaringan saraf mungkin merupakan penggunaannya yang paling dipopulerkan saat ini, tetapi gagasan tersebut memiliki sejarah yang panjang, dan kisahnya terus berlanjut. Teknik ini akan diungkapkan dalam kaitannya dengan kegunaannya untuk jaringan saraf, pelajaran yang diperoleh dari mempelajarinya jauh lebih dapat diterapkan, menggambarkan beberapa kebenaran yang lebih dalam tentang diferensiasi, pemrograman dinamis, dan seni pemecahan masalah secara umum.

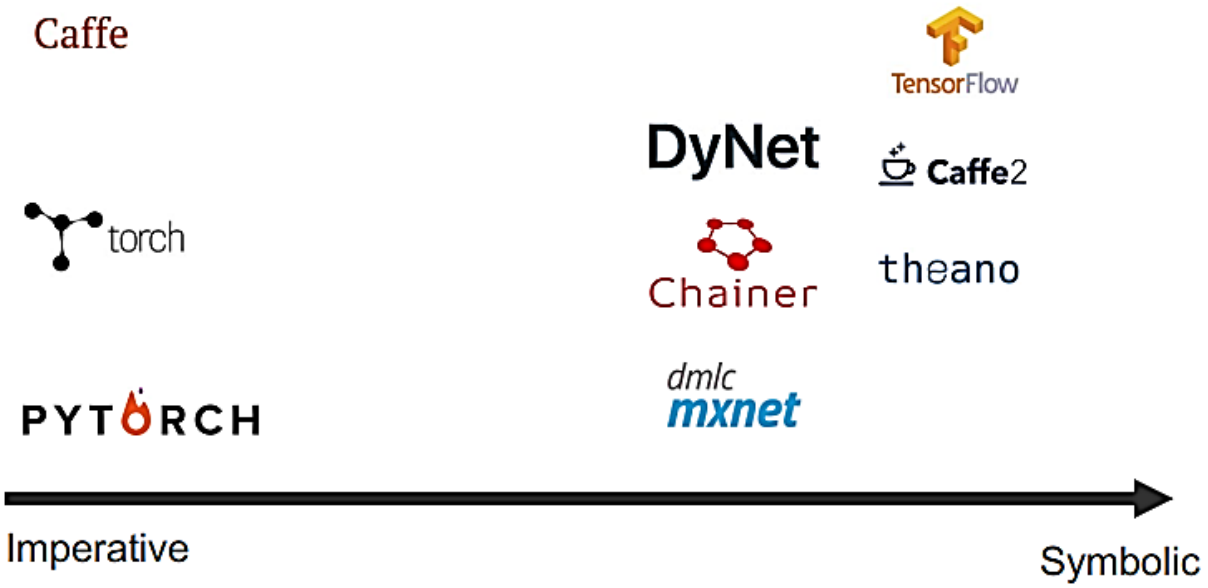
Derivatif dari fungsi variabel tunggal adalah pengukuran seberapa kecil variasi ruang input suatu fungsi sesuai dengan variasi yang dihasilkan dalam ruang outputnya. Dan hal yang sama berlaku untuk fungsi multivariabel, kecuali sekarang kita memiliki lebih banyak cara untuk memvariasikan ruang input kita. Diferensiasi adalah tulang punggung untuk metode optimasi berbasis gradien yang digunakan dalam pembelajaran mendalam (DL) atau pemodelan berbasis optimasi pada umumnya. Contoh metode pengoptimalan berbasis gradien adalah penurunan gradien. Saat menggunakan penurunan gradien atau penurunan gradien stokastik, kami secara iteratif meminimalkan kerugian atau fungsi objektif dengan menyetel parameter model ke arah gradien negatif. Autodiff menghitung diferensial dengan presisi mesin. Ini menghasilkan nilai numerik dan menggunakan variabel perantara atau bagian dari fungsi asli. Artinya, definisikan program (fungsi asli atau fungsi yang diinginkan) sebagai komposisi operasi primitif yang kita ketahui cara menurunkannya. Banyak pustaka diferensiasi otomatis telah dikembangkan.

**Perangkat Pembelajaran Mendalam**

Berikut ini adalah macam-macam dari toolkit dari pembelajaran mendalam yang bisa kita gunakan. Dari masing-masing toolkit diadopsi secara berbeda di domain yang berbeda:



Jika dilihat dari gambar diatas adalah perbandigan antara tampilan dengan Domain nya. Perbandingan lainnya juga bisa kita lihat dari symbol dan pemrograman imperative nya.




Jika kita lihat dari gambar diatas, Yang dimaksud Simbolik adalah penulisan simbol untuk menyusun jaringan terlebih dahulu kemudian evaluasi kemudian. Sedangkan Imperatif maksudnya adalah pengevaluasian segera dalam penyusunan program.

```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + Constant(1)
# compiles the function
f = compile(D)
d = f(A=np.ones(10), B=np.ones(10)*2)
```

Symbolic

```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
d = c + 1
```

Imperative

*Simbolik*

Keuntungan nya adalah mudah dioptimalkan (mis. didistribusikan, batching, paralelisasi) untuk pengembang. Sedangkan kekurangannya adalah cara pemrograman mungkin kontra-intuitif, Sulit di-debug untuk program pengguna dan kurang fleksibel maksudnya kita perlu menulis simbol sebelum benar-benar melakukan sesuatu

*Imperatif*

Keuntungan nya adalah lebih fleksibel karena penulisan satu baris dan pengevaluasian tiap baris, Mudah diprogram dan mudah di-debug: karena cocok dengan cara kita menggunakan C++ atau python. Kekurangan nya disini adalah Kurang efisien dan lebih sulit dioptimalkan

*Sebagai contoh perbandingan lain*

Perbandingan Toolkit antara grafik aliran data vs. konstruksi lapis demi lapis akan ditampikan dalam gambar dibawah ini:


*Grafik Aliran Data*

Grafik aliran data tampaknya menjadi pilihan dominan untuk merepresentasikan model pembelajaran mendalam. Apa yang baik untuk grafik aliran data:

- Cocok untuk alur kerja statis: tentukan sekali, jalankan untuk kumpulan/data arbitrer
- Kenyamanan pemrograman: mudah diprogram setelah Anda terbiasa.
- Mudah diparalelkan/digabungkan untuk grafik tetap
- Mudah dioptimalkan: banyak teknik pengoptimalan siap pakai untuk grafik

Sedangkan keburukan untuk grafik aliran data

- Tidak bagus untuk alur kerja dinamis: perlu menentukan grafik untuk setiap sampel pelatihan -> biaya overhead
- Sulit untuk memprogram jaringan saraf dinamis: bagaimana Anda bisa mendefinisikan grafik dinamis menggunakan bahasa untuk grafik statis? (misalnya LSTM, pohon-LSTM).
- Tidak mudah untuk debugging.
- Sulit untuk memparalelkan/menggabungkan beberapa grafik: setiap grafik berbeda, tidak ada pengelompokan alami.

**Grafik Aliran Data Statis vs. Dinamis**

Pengujian aliran data adalah teknik pengujian kotak putih yang memeriksa aliran data sehubungan dengan variabel yang digunakan dalam kode. Ini memeriksa inisialisasi variabel dan memeriksa nilainya di setiap instance.

Jenis pengujian aliran data

Ada dua jenis pengujian aliran data:

1. Pengujian aliran data statis: Deklarasi, penggunaan, dan penghapusan variabel diperiksa tanpa mengeksekusi kode. Grafik aliran kontrol sangat membantu dalam hal ini.
2. Pengujian aliran data dinamis: Variabel dan aliran data diperiksa dengan eksekusi kode.

*Keuntungan dari pengujian aliran data*

Pengujian aliran data membantu menangkap berbagai jenis anomali dalam kode. Anomali tersebut antara lain:

- Menggunakan variabel tanpa deklarasi
- Menghapus variabel tanpa deklarasi
- Mendefinisikan variabel dua kali
- Menghapus variabel tanpa menggunakannya dalam kode
- Menghapus variabel dua kali
- Menggunakan variabel setelah menghapusnya
- Tidak menggunakan variabel setelah mendefinisikannya

*Kerugian dari pengujian aliran data*

Beberapa kelemahan dari pengujian aliran data adalah:

- Pengetahuan yang baik tentang pemrograman diperlukan untuk pengujian yang tepat
- Memakan waktu

*Grafik Aliran Data Statis*

Pengujian yang hanya ditentukan sekali tetapi eksekusinya dilakukan berkali-kali. Misalnya: jaringan saraf convolutional. Eksekusi: Setelah ditentukan, semua komputasi berikut akan mengikuti komputasi yang ditentukan

- Keuntungan
  - Tidak ada upaya ekstra untuk pengoptimalan batching, karena secara alami dapat di-batch
  - Itu selalu mudah untuk menangani grafik aliran data komputasi statis dalam semua aspek, karena struktur tetap
  - Penempatan node, runtime terdistribusi, manajemen memori, dll.
  - Menguntungkan para pengembang

*Grafik Aliran Data Dinamis*

Eksekusinya kapan kita membutuhkan? Dalam semua kasus grafik aliran data statis tidak bekerja dengan baik, Ukuran masukan bervariasi tergantung dari kebutuhan nya. Input yang terstruktur secara bervariasi. Keluaran yang terstruktur secara bervariasi.

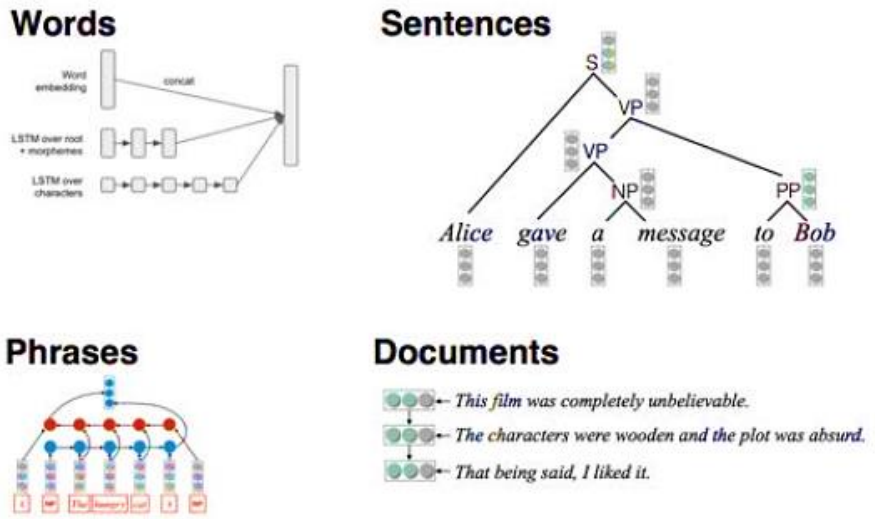
Masalah yang akan timbul ketika kita melakukan pengujian aliran data adalah sebagai berikut

- Kesulitan dalam mengungkapkan logika kontrol aliran yang rumit
- Kompleksitas implementasi grafik perhitungan
- Kesulitan dalam debugging

## 5.3 DYNET

DyNet merupakan perangkat untuk mengimplementasikan model jaringan saraf berdasarkan deklarasi dinamis struktur jaringan. Dalam strategi deklarasi statis yang digunakan dalam toolkit seperti Theano, CNTK, dan TensorFlow, pengguna pertama-tama menentukan grafik komputasi (representasi simbolis dari komputasi), lalu contoh dimasukkan ke dalam mesin yang menjalankan komputasi ini dan menghitung turunannya. Dirancang untuk alur kerja pembelajaran mendalam yang dinamis, mis.

- Tree-LSTM untuk terjemahan mesin saraf, di mana setiap kalimat mendefinisikan struktur yang sesuai dengan aliran komputasi
- Graph-LSTM untuk image parsing, dimana setiap image memiliki koneksi khusus antar segmen




**Bahan Utama di DyNet**

Dynet adalah kerangka kerja pertama yang melakukan pengelompokan dinamis dalam deklarasi dinamis. DyNet mengusulkan/mengadopsi dua metode cerdas untuk menemukan operasi yang tersedia yang dapat dikelompokkan bersama: pengelompokan berbasis kedalaman dan pengelompokan berbasis agenda. Pengelompokan berbasis kedalaman dilakukan dengan menghitung kedalaman setiap node dalam grafik komputasi asli, yang didefinisikan sebagai panjang maksimum dari node daun ke node itu sendiri, dan mengelompokkan node yang memiliki kedalaman dan tanda tangan yang identik. Dengan konstruksi, node dengan kedalaman yang sama tidak bergantung satu sama lain, karena semua node memiliki kedalaman yang lebih tinggi daripada inputnya. Dengan demikian, strategi batching ini dijamin bisa berjalan.

Pengelompokan berbasis agenda adalah metode yang tidak hanya bergantung pada kedalaman. Inti dari metode ini adalah agenda yang melacak node "tersedia" yang tidak memiliki dependensi yang belum terselesaikan. Jumlah dependensi yang belum terselesaikan dari setiap node dipertahankan dan diinisialisasi menjadi jumlah input ke node. Agenda diinisialisasi dengan menambahkan node yang tidak memiliki input masuk (dan dengan demikian tidak ada dependensi yang belum terselesaikan). Pada setiap iterasi, kami memilih node dari agenda bersama dengan semua node yang tersedia dalam tanda tangan yang sama, dan mengelompokkannya ke dalam satu operasi batch. Node ini kemudian dihapus dari agenda, dan penghitung ketergantungan dari semua penerusnya dikurangi. Setiap node tanpa ketergantungan baru ditambahkan ke agenda dan proses ini diulang sampai semua node telah diproses.

Pengelompokan dinamis DyNet dalam praktiknya jauh lebih kompleks. Strategi pengelompokan yang berbeda memiliki pro dan kontra yang berbeda, dan penting juga untuk

mempertimbangkan bidang masalah lain seperti memeriksa tanda tangan, menjamin kesinambungan memori, dan menjadwalkan kernel komputasi.

- Pisahkan deklarasi parameter dan konstruksi grafik
- Mendeklarasikan parameter yang dapat dilatih dan membuat model terlebih dahulu
  - Parameter, mis. matriks bobot dalam unit LSTM.
  - Bangun model sebagai kumpulan parameter yang dapat dilatih
- Membangun grafik perhitungan
  - Alokasikan beberapa node untuk komputasi kita (node dapat dilihat sebagai layer di NN)
  - Tentukan grafik aliran data dengan menghubungkan node bersama-sama
  - Jika perlu, grafik yang berbeda untuk sampel masukan yang berbeda
- Kesimpulan: Tentukan parameter sekali, tetapi tentukan grafik secara dinamis tergantung pada input

```
model = dy.Model()

pW = model.add_parameters((20,4))
pb = model.add_parameters(20)

dy.renew_cg()
x = dy.inputVector([1,2,3,4])
W = dy.parameter(pW) # convert params to expression
b = dy.parameter(pb) # and add to the graph

y = W * x + b
```

- Backend dan model pemrograman

Konstruksi grafik, Di TensorFlow, membuat grafik memiliki beban yang cukup besar. Pengguna TensorFlow menghindari penentuan grafik berulang kali. DyNet mendefinisi grafik yang sangat optimals. Sedikit overhead mendefinisikan grafik: baik untuk jaringan saraf dinamis. Mudah untuk menulis program rekursif untuk mendefinisikan grafik (sangat efektif untuk banyak jaringan dinamis, seperti pohon-LSTM atau grafik-LSTM).

```
class TreeRNNBuilder(object):
    def __init__(self, model, word_vocab, hdim):
        self.W = model.add_parameters((hdim, 2*hdim))
        self.E = model.add_lookup_parameters((len(word_vocab),hdim))
        self.w2i = word_vocab

    def encode(self, tree):
        if tree.isleaf():
          return self.E[self.w2i.get(tree.label,0)]
        elif len(tree.children) == 1: # unary node, skip
          expr = self.encode(tree.children[0])
          return expr
        else:
          assert(len(tree.children) == 2)
          e1 = self.encode(tree.children[0])
          e2 = self.encode(tree.children[1])
          W = dy.parameter(self.W)
          expr = dy.tanh(W*dy.concatenate([e1,e2]))
          return expr

model = dy.Model()
U_p = model.add_parameters((2,50))
tree_builder = TreeRNNBuilder(model, word_vocabulary, 50)
trainer = dy.AdamTrainer(model)
for epoch in xrange(10):
  for in_tree, out_label in read_examples():
    dy.renew_cg()
    U = dy.parameter(U_p)
    loss = dy.pickneglogsoftmax(U*tree_builder.encode(in_tree), out_label)
    loss.forward()
    loss.backward()
    trainer.update()
```

DyNet TreeLSTM (30 LoC)

TensorFlow TreeLSTM (200 LoC)

# BAB 6
# PEMBELAJARAN MENDALAM TERDISTRIBUSI

Perkembangan terbaru dalam pembelajaran mendalam telah menghasilkan beberapa hasil canggih yang menarik terutama di bidang seperti pemrosesan bahasa alami dan visi komputer. Beberapa alasan keberhasilan biasanya berasal dari ketersediaan data dalam jumlah besar dan ukuran model deep learning (DL) yang semakin besar. Algoritma ini mampu mengekstraksi pola yang bermakna dan menurunkan korelasi antara input dan output. Tetapi juga benar bahwa mengembangkan dan melatih algoritme kompleks ini dapat memakan waktu berhari-hari bahkan berminggu-minggu.

Untuk mengelola masalah ini, diperlukan pendekatan yang cepat dan efisien untuk merancang dan mengembangkan model baru. Seseorang tidak dapat melatih model ini pada satu GPU karena akan mengakibatkan kemacetan informasi. Untuk mengatasi masalah kemacetan informasi pada GPU inti tunggal, kami perlu menggunakan GPU multi-inti. Di sinilah ide pelatihan terdistribusi muncul.

Pelatihan DL biasanya bergantung pada skalabilitas, yang berarti kemampuan algoritme DL untuk mempelajari atau menangani sejumlah data. Pada dasarnya skalabilitas algoritma DL tergantung pada tiga faktor:

1. Ukuran dan kompleksitas model deep learning
2. Jumlah data pelatihan
3. Ketersediaan infrastruktur yang mencakup perangkat keras seperti GPU dan unit penyimpanan, serta integrasi yang mulus antar perangkat ini

Pelatihan terdistribusi memenuhi ketiga elemen tersebut. Ini menangani ukuran dan kompleksitas model, menangani data pelatihan dalam batch, dan membagi serta mendistribusikan proses pelatihan di antara banyak prosesor yang disebut node. Lebih penting lagi, ini mengurangi waktu pelatihan secara signifikan membuat waktu iterasi lebih pendek dan dengan demikian membuat eksperimen dan penyebaran lebih cepat.

Pelatihan terdistribusi terdiri dari dua jenis:

1. Pelatihan paralel data
2. Pelatihan model-paralel

Kriteria untuk memilih kerangka kerja yang tepat untuk pelatihan yang didistribusikan. Sebelum kita menyelami kerangka kerja, ada beberapa poin yang harus dipertimbangkan saat memilih kerangka kerja dan alat yang tepat:

1. Jenis grafik komputasi: Seluruh komunitas pembelajaran mendalam sebagian besar dibagi menjadi dua faksi, satu yang menggunakan PyTorch atau grafik komputasi dinamis dan yang lainnya menggunakan TensorFlow atau grafik komputasi statis. Oleh karena itu, bukan berita bahwa sebagian besar kerangka kerja terdistribusi dibangun di atas dua pustaka ini. Jadi jika Anda lebih memilih salah satu dari yang lain maka setengah dari keputusan Anda sudah dibuat.

2. Biaya pelatihan: Keterjangkauan merupakan perhatian penting saat Anda berurusan dengan komputasi terdistribusi, mis. sebuah proyek yang melibatkan pelatihan BigGAN dapat memerlukan sejumlah GPU dan biayanya dapat meningkat secara proporsional seiring bertambahnya jumlah ini. Karenanya, alat dengan harga sedang selalu menjadi pilihan yang tepat.
3. Jenis pelatihan: Bergantung pada kebutuhan pelatihan Anda, yaitu paralelisme data atau paralelisme model, Anda dapat memilih satu alat daripada yang lain.
4. Efisiensi: Ini pada dasarnya mengacu pada jumlah baris yang perlu Anda tulis untuk mengaktifkan pelatihan terdistribusi, semakin sedikit semakin baik.
5. Fleksibilitas: Bisakah kerangka pilihan Anda digunakan di berbagai platform? Terutama saat Anda perlu berlatih di tempat atau di platform cloud.

### 6.1.1 TOOLKIT DL (DEEP LEARNING) PADA MESIN TUNGGAL

Toolkit mesin tunggal menggunakan GPU adalah suatu keharusan. Sejumlah kecil mesin yang dilengkapi GPU dapat mencapai kecepatan yang memuaskan dibandingkan dengan kluster CPU dengan ribuan inti. Akan ditunjukan pada gambar dibawah ini:

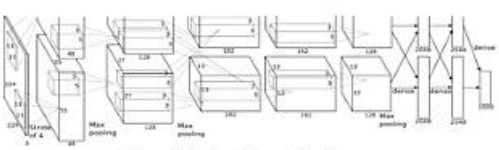


Lebih mudah tersedia untuk peneliti misalnya:

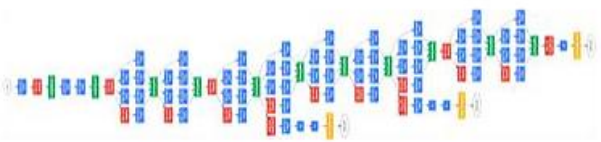
- Sekelompok 8 mesin yang dilengkapi GPU
- Sekelompok 2000 core CPU

Namun, menggunakan satu GPU masih jauh dari cukup

- jaringan dalam berukuran rata-rata membutuhkan waktu berhari-hari untuk berlatih menggunakan satu GPU saat menghadapi data 100 GB hingga TB
- Menuntut pelatihan jaringan saraf yang lebih cepat pada kumpulan data yang semakin besar



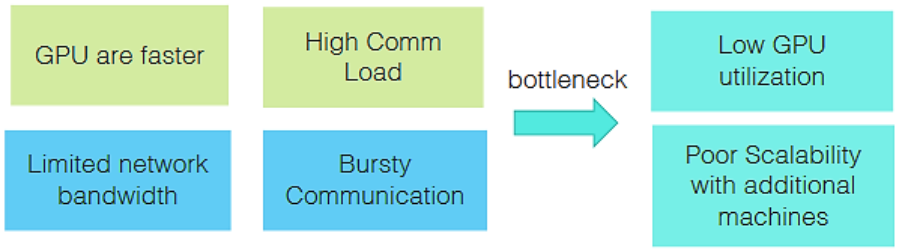
AlexNet, 5 – 7 days



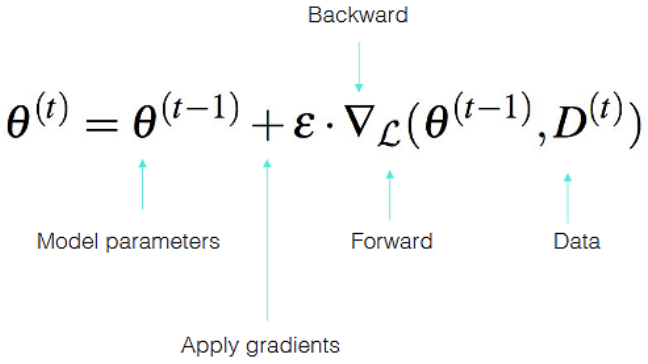
GoogLeNet, 10+ days

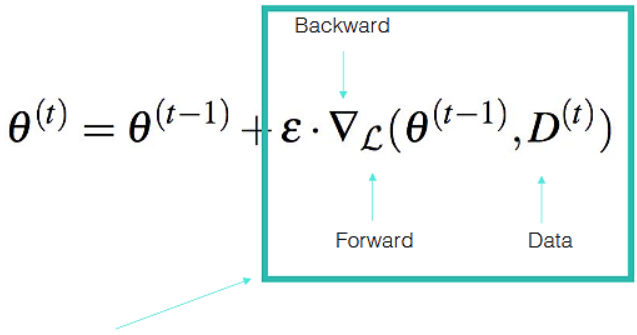
Namun, implementasi DL terdistribusi saat ini (mis. di TensorFlow) dapat diskalakan dengan buruk karena sinkronisasi parameter substansial melalui jaringan (kami akan tunjukkan nanti)

Tantangan komunikasi dalam single machine adalah GPU setidaknya satu kali lipat lebih cepat daripada CPU




Beban komunikasi yang tinggi meningkatkan komunikasi jaringan sebagai bottleneck utama mengingat bandwidth komoditas Ethernet yang terbatas. Mengelola komputasi dan komunikasi dalam klaster GPU terdistribusi sering memperumit desain algoritma. Masalah yang akan dihadapi pada mesin ini adalah Deep Learning pada satu node – formulasi iteratif-konvergen dapat dihitung pada rumus berikut ini



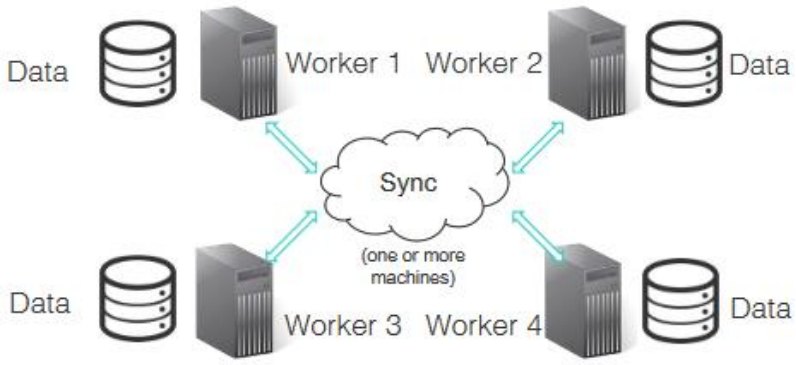




Maju dan mundur adalah komputasi utama (99%) beban kerja program deep learning.

## 6.2 DATA PARALEL DENGAN SGD *(STOCHASTIC GRADIENT DESCENT)*

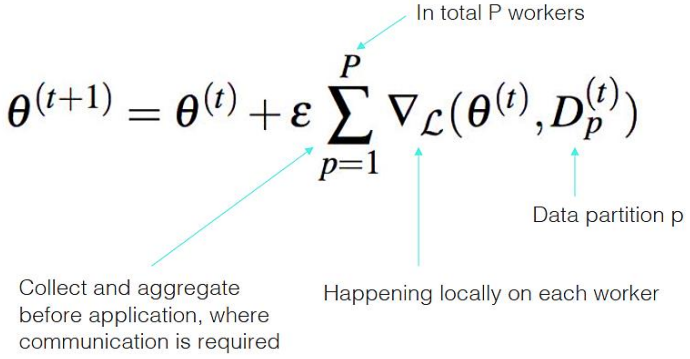
Merancang metode paralelisasi yang mempertimbangkan tingginya biaya komunikasi yang ditemukan di komputer cluster telah dipelajari secara ekstensif. SGD Paralel, adalah salah satu teknik tersebut dan dapat dilihat sebagai perbaikan rata-rata model. Konvergensi rata-rata model tergantung pada tingkat konveksitas sebagai hasil dari regularisasi. Namun, regularisasi berkurang dengan bertambahnya ukuran data yang membuatnya kurang bermanfaat dalam praktiknya, terutama karena ukuran data terus bertambah. SGD Paralel memperbaiki hal ini dengan menggabungkan manfaat dari komunikasi jaringan rendah rata-

rata model dan pembelajaran online. Dalam praktiknya, Parallel SGD adalah metode Paralel Data dan diimplementasikan seperti itu. Ada dua jenis komputer (atau node) berbeda yang digunakan dalam pengoptimal ini, server parameter dan node pekerja. Server parameter adalah tempat model Deep Learning ditentukan, tempat agregasi dari node pekerja terjadi dan bertanggung jawab untuk mengkomunikasikan model Deep Learning dan data pelatihan ke node pekerja. node pekerja bertanggung jawab untuk melatih model pada data pelatihan, mengkomunikasikan perubahan pada model kembali ke server parameter dan melakukan sebagian besar perhitungan selama proses pelatihan. Implementasi yang ketat dari Parallel SGD menggunakan pengoptimal SGD untuk pelatihan yang dilokalkan. kita biasanya mencari strategi paralelisasi yang disebut paralelisme data, berdasarkan SGD mempartisi data menjadi bagian yang berbeda, Biarkan mesin yang berbeda menghitung pembaruan gradien pada partisi data yang berbeda. Kemudian agregat/sinkronisasi atau lakukan pengujian.




Data turunan gradien stokastik paralel. Paralelisme data mengharuskan setiap pekerja memiliki akses baca dan tulis ke parameter model bersama $\theta$, yang menyebabkan komunikasi antar pekerja. SGD Paralel memperbaiki hal ini dengan menggabungkan manfaat dari komunikasi jaringan rendah rata-rata model dan pembelajaran online.

Setelah semua pekerja selesai dengan proses pelatihan mereka, semua node mengirimkan, ke server parameter, model baru yang baru saja mereka latih. Karena, melatih model jaringan saraf hanya mengubah bobot antar node, hanya perubahan bobot yang dikirim ke server parameter. Secara khusus, pembaruan bobot setiap node dibagi dengan jumlah node. Hasilnya adalah model jaringan saraf baru (dilatih secara bertahap) yang kemudian didistribusikan kembali ke node pekerja dan proses kemudian berulang untuk jumlah iterasi yang diinginkan.



**Bagaimana cara berkomunikasi**

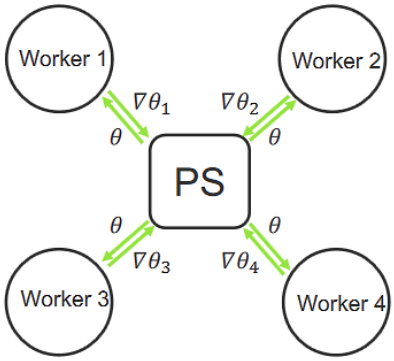
Pada setiap pekerja, model perlu didefinisikan secara identik dengan pekerja lainnya, kemudian bobotnya perlu ditetapkan. Bobot akan berupa bobot yang diinisialisasi, atau bobot dari model yang dilatih sebagian. Dalam kasus Parallel SGD, semua pekerja memulai dengan bobot yang sama. Bobot kemudian dikembalikan setelah latihan seperti yang ditunjukkan pada baris terakhir.

Setelah data dalam format yang dapat dikonsumsi oleh Keras, pelatihan model dapat dimulai. Di Keras, proses pelatihan jaringan saraf sangat fleksibel. Berbagai pengaturan pengoptimal, seperti ukuran batch, jumlah epoch, logging, antara lain diteruskan ke fungsi Keras oleh penangan meta-data. Implementasi handler ini dirancang agar fleksibel dan dapat diperluas untuk dapat menangani berbagai jenis model dan skenario pelatihan.

Pada titik ini, semua node telah menyelesaikan pelatihan dan bobot perlu digabungkan. Cara bobot digabungkan bergantung pada skema paralelisasi mana yang ditentukan pengguna. Skema dapat memiliki model yang dilatih secara rekursif untuk beberapa zaman lagi, atau kinerjanya dapat dievaluasi pada data uji yang tidak terlihat. Untuk mengevaluasi model dengan benar, itu harus digunakan untuk membuat prediksi pada data uji yang tidak terlihat. Data uji yang tidak terlihat hanyalah data yang belum digunakan secara langsung dalam melatih model.

Metrik kinerja, bergantung pada sifat model, digunakan untuk menunjukkan seberapa baik jaringan saraf dapat menggeneralisasi. Evaluasi hanya memprediksi kelas berdasarkan fitur dalam data uji dan membandingkannya dengan kebenaran dasar dari titik data tersebut Server parameter, mis. Bosen, SSP

- Server parameter (PS) adalah sistem memori bersama yang menyediakan akses bersama untuk parameter model global $\theta$
- Pembelajaran mendalam dapat dengan mudah diparalelkan data dengan pekerja terdistribusi menggunakan PS dengan 3 langkah:
  - Setiap pekerja menghitung gradien (∇L) pada partisi data mereka sendiri ($Dp$) dan mengirimkannya ke server jarak jauh;
  - server menerima pembaruan dan menerapkan (+) pada parameter yang dibagikan secara global;
  - Setiap pekerja menarik kembali parameter yang diperbarui ($\theta\_t$)



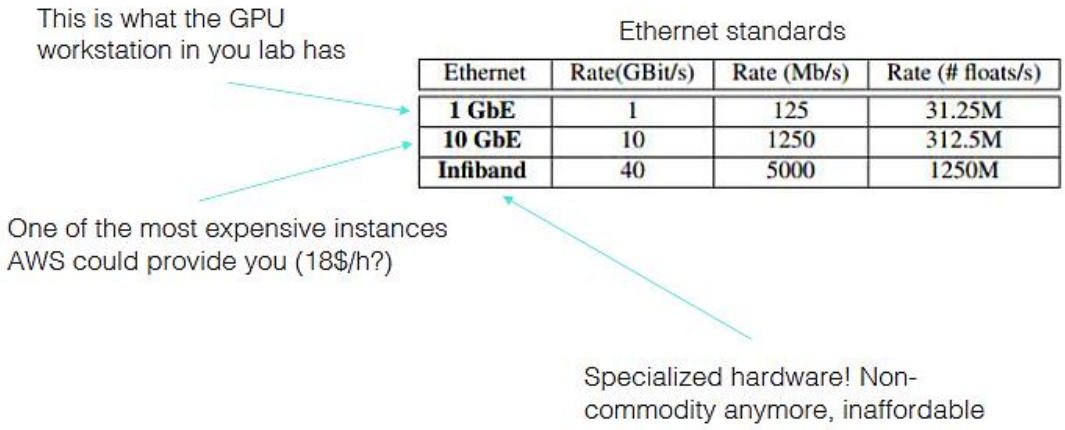
## 6.3 SERVER PARAMETER

Arsitektur server parameter, memiliki dua kelas node: Node server mempertahankan partisi dari parameter yang dibagikan secara global (parameter lokal mesin tidak disinkronkan secara default). Mereka berkomunikasi satu sama lain untuk mereplikasi dan/atau memigrasikan parameter untuk keandalan dan penskalaan. Node klien melakukan sebagian besar perhitungan; node server terutama melakukan langkah-langkah pembukuan dan agregasi global. Setiap klien biasanya menyimpan secara lokal sebagian dari data pelatihan, menghitung statistik lokal seperti gradien. Klien hanya berkomunikasi dengan node server, memperbarui dan mengambil parameter bersama. Klien dapat ditambahkan atau dihapus; melakukan hal itu memerlukan pengiriman bagian yang sesuai dari kumpulan data pelatihan ke mesin baru dan menanyakan set parameter masing-masing. Menerapkan server parameter secara langsung untuk pembelajaran mendalam terdistribusi berbasis GPU akan berkinerja buruk (seperti yang akan ditampilkan nanti).

Contoh server parameter maksimalzAlexNet: parameter float 61,5 juta, 0,25 detik/iterasi pada Geforce Titan X (ukuran batch = 256)

Tingkat pembangkitan gradien: 240M float/(s*GPU)

- Paralelasikan lebih dari 8 mesin masing-masing dengan satu GPU menggunakan PS.
- Untuk memastikan perhitungan tidak diblokir pada GPU (yaitu percepatan linier dengan node tambahan)
  - Sebagai pekerja: kirim 240M pelampung/dtk dan tarik kembali 240M pelampung/dtk (setidaknya)
  - Sebagai server: terima 240M * 8 float/dtk dan kirim kembali 240M * 8/dtk (setidaknya)




Masalahnya lebih parah dari yang dijelaskan di atas

Kami hanya menggunakan 8 node (yang kecil). Bagaimana dengan 32.128, atau bahkan 256?

- Kami belum mempertimbangkan masalah lain (yang mungkin juga merepotkan), mis.
  - Salinan memori antara DRAM dan GPU akan memiliki biaya yang tidak kecil
  - Ethernet mungkin dibagi dengan tugas lain, yaitu bandwidth yang tersedia bahkan lebih sedikit.
  - Burst communication sangat sering terjadi pada GPU (yang akan dijelaskan nanti).

Atasi Kemacetan Komunikasi

- Fakta sederhana: Waktu komunikasi bisa dikurangi, tapi tidak bisa dihilangkan (tentu saja)
- Oleh karena itu, ide-ide yang mungkin untuk mengatasi kemacetan komunikasi
  - Sembunyikan waktu komunikasi dengan menimpanya dengan waktu komputasi
  - Kurangi ukuran pesan yang diperlukan untuk menjadi komunikasi

Pendekatan server parameter untuk optimasi terdistribusi. Dalam model ini, node komputasi dipartisi menjadi klien dan server. Setiap klien "memiliki" sebagian dari data dan beban kerja, dan server bersama-sama mempertahankan parameter yang dibagikan secara global. Ide arsitektur ini bukanlah hal baru: Ini telah diterapkan pada beberapa aplikasi pembelajaran mesin termasuk model variabel laten, inferensi terdistribusi pada grafik, dan pembelajaran mendalam. Tujuannya adalah untuk membangun sistem tujuan umum dengan fitur yang hanya didukung sebagian oleh pekerjaan sebelumnya:

~ *Kemudahan penggunaan*. Parameter yang dibagikan secara global direpresentasikan sebagai vektor dan matriks (berpotensi jarang), yang merupakan struktur data yang lebih nyaman untuk aplikasi pembelajaran mesin daripada penyimpanan atau tabel (kunci, nilai) yang banyak digunakan. Operasi aljabar linier multi-utas berkinerja tinggi dan nyaman, seperti perkalian vektor-matriks antara parameter dan data pelatihan lokal, disediakan untuk memfasilitasi pengembangan aplikasi.

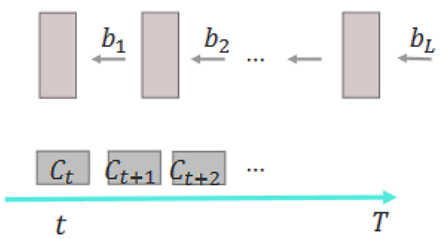
~ *Efisiensi*. Komunikasi antar node tidak sinkron. Yang penting, sinkronisasi tidak memblokir komputasi. Kerangka kerja ini memungkinkan perancang algoritme untuk menyeimbangkan laju konvergensi algoritmik dan efisiensi sistem, di mana pertukaran terbaik bergantung pada data, algoritme, dan perangkat keras.

~ *Skalabilitas Elastis.* Node baru dapat ditambahkan tanpa memulai ulang kerangka kerja yang sedang berjalan. Properti ini diinginkan, mis. untuk sketsa streaming atau saat menggunakan server parameter sebagai layanan online yang harus tetap tersedia untuk waktu yang lama.

~ *Toleransi Kesalahan dan Daya Tahan.* Sebaliknya, kegagalan node tidak dapat dihindari, terutama pada skala besar yang menggunakan server komoditas.

**Komputasi dan Komunikasi Tumpang Tindih**

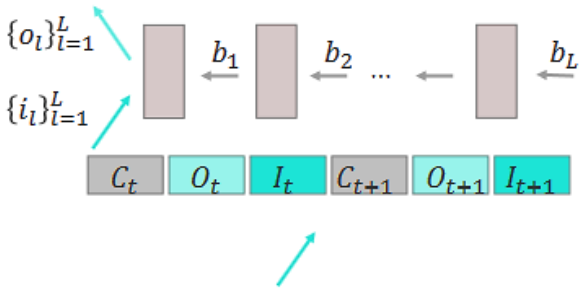
- Tinjau kembali aliran perhitungan BP pada satu node
  - $b_l$: komputasi backpropagation melalui layer l
  - $C_t$ : perhitungan maju dan mundur pada iterasi t




$$\theta^{(t)} = \theta^{(t-1)} + \varepsilon \cdot \nabla_{\mathcal{L}}(\theta^{(t-1)}, D^{(t)})$$

- Pada beberapa node, saat komunikasi terlibat
- Memperkenalkan dua operasi komunikasi
  €$_l$: kirim gradien di layer $l$ ke remote
  i$_l$: tarik kembali parameter lapisan $l$ yang dibagikan secara global dari jarak jauh
  $O_t$ : himpunan $\{o_l\}_{l=1}^{L}$ pada iterasi t
  $I_t$ : himpunan $\{i_l\}_{l=1}^{L}$ pada iterasi t

$$\theta^{(t)} = \theta^{(t-1)} + \varepsilon \cdot \nabla_{\mathcal{L}}(\theta^{(t-1)}, D^{(t)})$$




Komputasi dan komunikasi terjadi secara berurutan!

- Perhatikan independensi berikut
  a. Operasi pengiriman $o_l$ tidak bergantung pada operasi mundur
  b. Operasi read-in i$_l$ dapat memperbarui parameter lapisan sepanjang $b_l$ selesai, tanpa memblokir operasi mundur berikutnya $b_i\ (i < l)$

Ide nya adalah perhitungan dan komunikasi yang tumpang tindih dengan memanfaatkan konkurensi Pipelining pembaruan dan operasi perhitungan

$$\theta^{(t)} = \theta^{(t-1)} + \varepsilon \cdot \nabla_{\mathcal{L}}(\theta^{(t-1)}, D^{(t)})$$

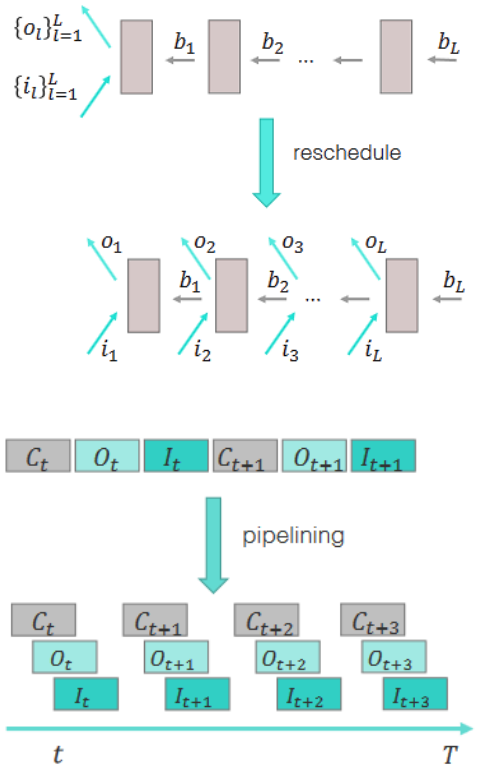
## 6.4 WAIT-FREE BACKPROPAGATION (WFBP)

WFBP paling bermanfaat untuk melatih model DL yang parameternya terkonsentrasi pada lapisan atas (lapisan FC) tetapi komputasi terkonsentrasi pada lapisan bawah (lapisan CONV) karena tumpang tindih komunikasi lapisan atas (90% waktu komunikasi) dengan perhitungan bawah layer (90% dari waktu komputasi). Selain NN seperti rantai, WFBP umumnya berlaku untuk struktur seperti non-rantai lainnya (misalnya, struktur seperti pohon), karena pengoptimalan parameter untuk jaringan saraf dalam bergantung pada lapisan yang berdekatan (dan bukan seluruh jaringan), selalu ada peluang untuk optimasi parameter (yaitu, perhitungan) dan komunikasi dari lapisan yang berbeda untuk dilakukan secara bersamaan.

Beberapa framework DL, seperti TensorFlow, merepresentasikan dependensi data program DL menggunakan grafik, oleh karena itu secara implisit mengaktifkan paralelisasi otomatis. Namun, mereka gagal mengeksplorasi peluang potensial paralelisasi antara iterasi. Misalnya, TensorFlow perlu mengambil parameter yang diperbarui dari penyimpanan jarak jauh di awal setiap iterasi, sementara prosedur komunikasi ini mungkin tumpang tindih

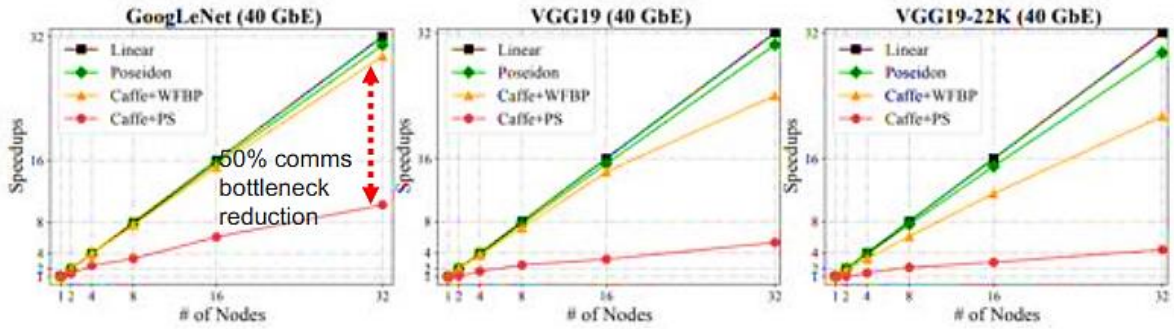
dengan prosedur komputasi dari iterasi sebelumnya. Idenya adalah perhitungan dan komunikasi yang tumpang tindih dengan memanfaatkan konkurensi

- Overhead komunikasi disembunyikan di bawah komputasi
- Hasil: lebih banyak perhitungan dalam satuan waktu

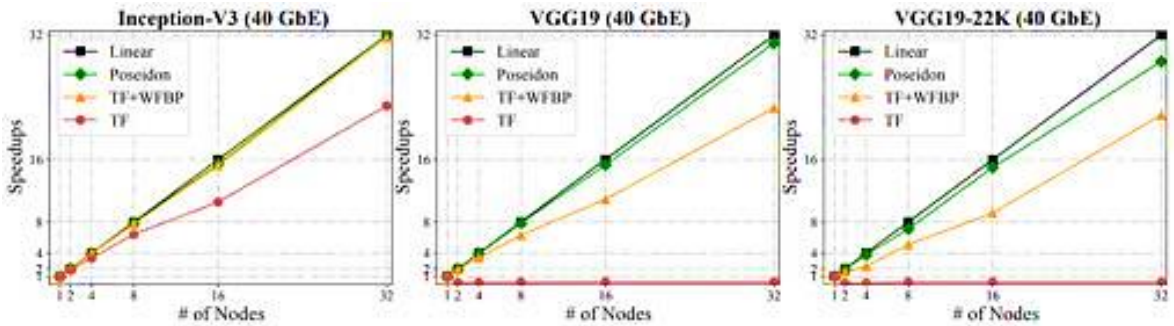



Bagaimana kinerja WFBP?

1. Menggunakan Caffe sebagai mesin:




2. Menggunakan TensorFlow sebagai mesin:



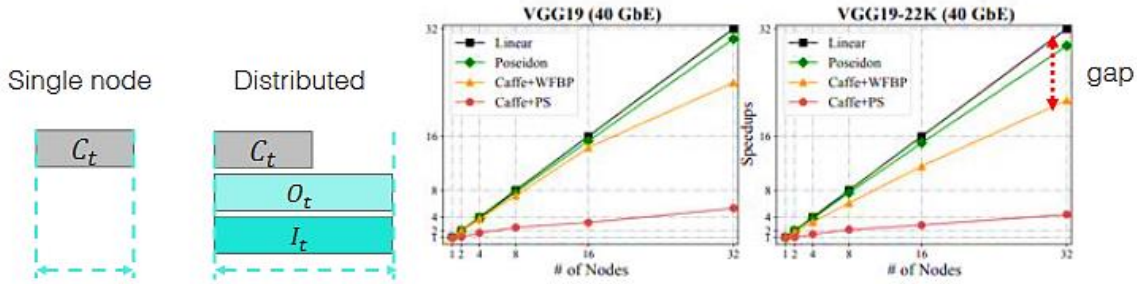
- Mengapa DWBP efektif, Lebih banyak statistik CNN modern, berikut ini perhitungan dari CNN Modern

Distribusi Params/FLOP dari CNN modern

| Parameters | CONV Layers (#/% ) | FC Layers (#/% ) |
|---|---|---|
| AlexNet | 2.3M / 3.75 | 59M / 96.25 |
| VGG-16 | 7.15M / 5.58 | 121.1M / 94.42 |
| FLOPs | CONV Layers (#/% ) | FC Layers (#/% ) |
| AlexNet | 1,352M / 92.0 | 117M / 8.0 |
| VGG-16 | 10,937M / 91.3 | 121.1M / 8.7 |

  - 90% komputasi terjadi pada lapisan bawah
  - 90% komunikasi terjadi di lapisan atas
  - WFBP tumpang tindih 90% dan 90%

- Apakah komunikasi dan komputasi yang tumpang tindih menyelesaikan semua masalah?
  ~ Ketika waktu komunikasi lebih lama dari komputasi, tidak (lihat gambar di bawah).
  ~ Katakanlah, jika komunikasi dan perhitungan benar-benar tumpang tindih, berapa banyak skalabilitas yang dapat kita capai?




**Sufficient Factor Broadcasting**

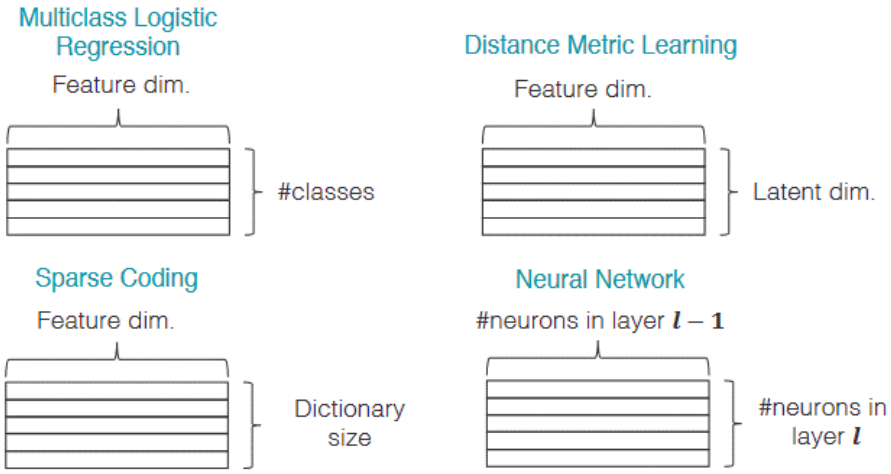
Model komputasi Sufficient Factor Broadcasting (SFB) untuk pembelajaran terdistribusi yang efisien dari keluarga besar model berparameter matriks, yang berbagi properti berikut: pembaruan parameter yang dihitung pada setiap sampel data adalah matriks peringkat-1, yaitu produk luar dari dua "faktor yang cukup" (SFs). Dengan menyiarkan SF di antara mesin pekerja dan merekonstruksi matriks pembaruan secara lokal di setiap pekerja, SFB meningkatkan efisiensi komunikasi — biaya komunikasi bersifat linier dalam dimensi matriks parameter, bukan kuadrat — tanpa memengaruhi kebenaran komputasi.

Kontribusi utama dari Sufficient Factor Broadcasting (SFB) adalah sebagai berikut:

- Mengidentifikasi properti faktor yang cukup dari keluarga besar model matriks-parametrized ketika diselesaikan dengan dua algoritma populer: penurunan gradien stokastik dan pendakian koordinat ganda stokastik.
- Mengingat properti faktor yang memadai, model perhitungan penyiaran faktor yang memadai (SFB). Melalui arsitektur peer-to-peer yang terdesentralisasi dengan

penyiaran parsial asinkron terbatas, SFB sangat mengurangi kompleksitas komunikasi sambil mempertahankan kinerja empiris yang sangat baik.

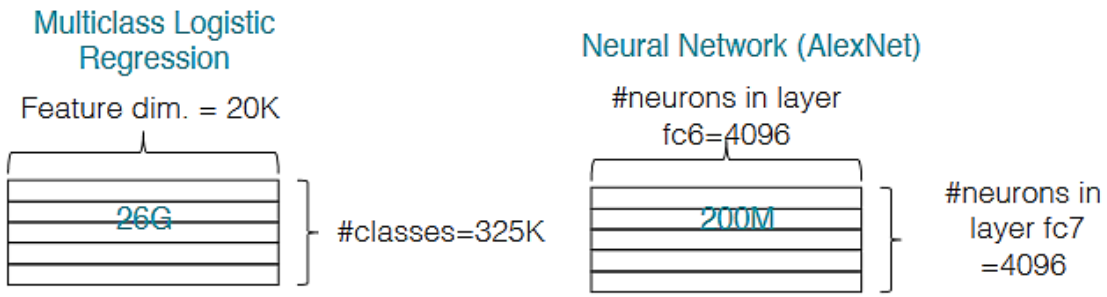
- Untuk lebih mengurangi biaya komunikasi, kami menyelidiki skema penyiaran parsial dan mengusulkan pendekatan berbasis grafoptimasi untuk menentukan topologi jaringan komunikasi.
- Menganalisis biaya komunikasi dan komputasi SFB dan memberikan jaminan konvergensi algoritme SGD minibatch berbasis SFB, di bawah eksekusi sinkron massal dan eksekusi asinkron terbatas.
- Secara empiris mengevaluasi SFB pada empat model populer, dan mengonfirmasi efisiensi dan kompleksitas komunikasi SFB yang rendah. Model matriks-parametrized sebagai berikut:




**Pembelajaran MPM Terdistribusi**

Mempelajari MPM dalam masalah ML skala besar memang menantang: Skala aplikasi ML telah meningkat secara dramatis. Untuk memastikan waktu berjalan yang cepat saat meningkatkan MPM ke masalah sebesar itu, sebaiknya beralih ke komputasi terdistribusi; namun, tantangan unik untuk MPM adalah bahwa matriks parameter berkembang pesat dengan ukuran masalah, menyebabkan strategi paralelisasi langsung menjadi kurang ideal. Pertimbangkan algoritme paralel data, di mana setiap pekerja menggunakan subset data untuk memperbarui parameter — paradigma umumnya adalah menyinkronkan matriks parameter lengkap dan memperbarui matriks di antara semua pekerja.

Mempelajari MPM dengan mengkomunikasikan matriks parameter antara server dan pekerja. Biaya komunikasi yang tinggi dan penundaan sinkronisasi yang besar




Pembaruan matriks parameter lengkap ΔW dapat dihitung sebagai produk luar dari dua vektor $uv^T$ (disebut faktor yang cukup). Contohnya: Penurunan gradien stokastik primal (SGD)

$$\min_W \frac{1}{N}\sum_{i=1}^{N} f_i(Wa_i;b_i)+h(W)$$

$$\Delta W = uv^T \quad u = \frac{\partial f(Wa_i,b_i)}{\partial(Wa_i)} \quad v = a_i$$

Contoh lain adalah sebagai berikut: Pendakian koordinat ganda stokastik (SDCA)

$$\min_Z \frac{1}{N}\sum_{i=1}^{N} f_i^*(-z_i)+h^*(\frac{1}{N}ZA^T)$$

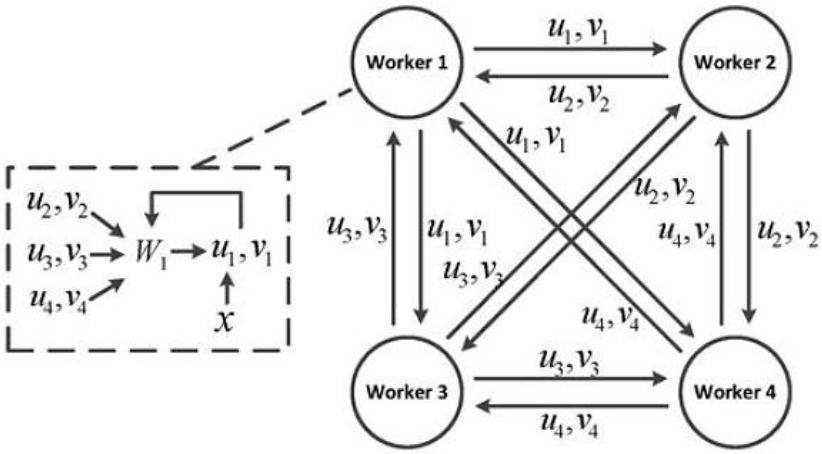
$$\Delta W = uv^T \quad u = \Delta z_i \quad v = a_i$$

Kirim pembaruan SF ringan (u, v), alih-alih pembaruan matriks penuh ΔW yang mahal!

**Topologi Peer to Peer + SF Updates**

Topologi peer-to-peer SFB dapat dikontraskan dengan arsitektur clientserver "full-matrix" untuk sinkronisasi parameter, untuk mempelajari jaringan saraf, server terpusat mempertahankan matriks parameter global, dan setiap klien menyimpan salinan lokal. Klien menghitung faktor yang cukup dan mengirimkannya ke server, yang menggunakan SF untuk memperbarui matriks parameter global; server kemudian mengirimkan matriks parameter lengkap yang diperbarui kembali ke klien. topologi SFB peer-to-peer tidak pernah mengirimkan matriks penuh; hanya SF yang dikirim melalui jaringan.

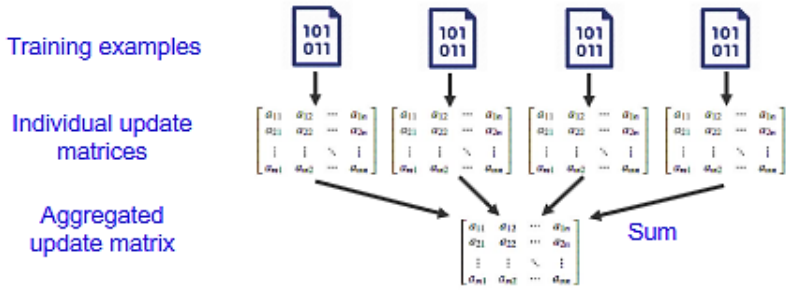
SFB adalah model komputasi terdesentralisasi peer-to-peer, kami perlu menunjukkan bahwa salinan parameter pada pekerja yang berbeda menyatu ke titik batas yang sama tanpa koordinasi terpusat, bahkan di bawah penundaan komunikasi karena eksekusi asinkron terbatas. Dalam hal ini, kami berbeda dari analisis sistem server parameter terpusat, yang sebaliknya menunjukkan konvergensi parameter global pada server pusat. Untuk lebih jelasnya mari kita lihat skema dibawah ini:



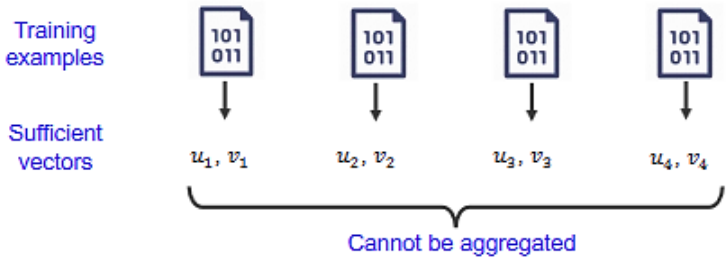

**Pembaruan Komunikasi dan Komputasi**

Setelah dilakukan pembaruan komputasi dan topologi maka kita akan memperoleh hasil matrik sebagai berikut

- Pembaruan penuh:

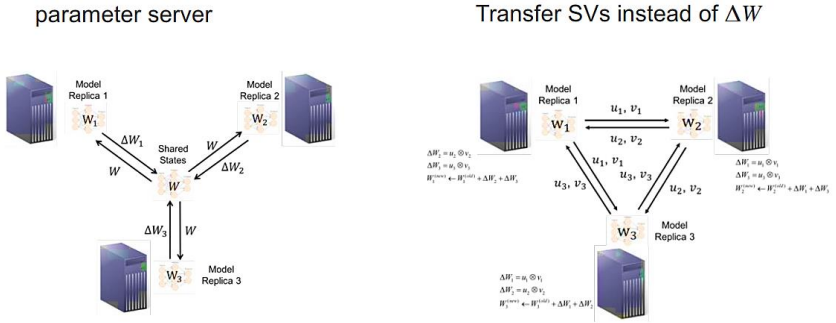



- Pra-pembaruan




## 6.5 SINKRONISASI REPLIKA PARAMETER

Berdasarkan pembaruan dari topologi disub-bab sebelumnya maka replika yang akan kita rancang adalahse bagai berikut



Dari skema diatas maka kita akan mendapatkan perbandingan biaya seperti tabel dibawah ini

| | Ukuran satu Pesan | Nomer Pesan | Lalulintas Jaringan |
|---|---|---|---|
| P2P SV-Transfer | *O(J + K)* | *O($P^2$)* | *O((J + K)$P^2$)* |
| Parameter Server | *O (JK)* | *O(P)* | *O(JKP)* |

**Percepatan Konvergensi**


Grafik diatas merupakan tolok ukur 3 Program Machine learning.

Matriks parameter besar dengan entri 6,5-8,6b (30+GB), berjalan pada 12- & 28- kluster mesin SFB 28 mesin selesai dalam 2-7 jam, Hingga 5,6x lebih cepat dari 28 mesin PS, 12,3x lebih cepat dari Spark 28 mesin, PS tidak dapat mendukung komunikasi SF, yang memerlukan penyimpanan terdesentralisasi. Dengan mengasumsikan, Staleness Synchronous Parallel (SSP) dengan parameter staleness $s$ dan Paralel Sinkron Massal adalah kasus khusus SSP ketika $s$ = 0 dan Metode komunikasi Partial broadcast (PB) mengirim pesan ke subset mesin $Q\ (Q < P-1)$ maka akan mendapatkan hasil berikut ini:

**Theorem 1.** *Let Assumption 1 hold, and let* $\{\mathbf{W}_p^c\}$, $p = 1, \ldots, P$, $\{\mathbf{W}^c\}$ *be the local sequences and the auxiliary sequence, respectively.*

*Under full broadcasting (i.e.,* $Q = P - 1$*) and set the learning rate* $\eta := \eta_c = O(\sqrt{\frac{1}{L\sigma^2 Psc}})$*, we have*

- $\liminf_{c\to\infty} \mathbb{E}\|\nabla F(\mathbf{W}^c)\| = 0$*, hence there exists a subsequence of* $\nabla F(\mathbf{W}^c)$ *that almost surely vanishes;*
- $\lim_{c\to\infty} \max_p \|\mathbf{W}^c - \mathbf{W}_p^c\| = 0$*, i.e., the maximal disagreement between all local sequences and the auxiliary sequence converges to 0 (almost surely);*
- *There exists a common subsequence of* $\{\mathbf{W}_p^c\}$ *and* $\{\mathbf{W}^c\}$ *that converges almost surely to a stationary point of* $F$*, with the rate* $\min_{c\le C} \mathbb{E}\|\sum_{p=1}^P \nabla F_p(\mathbf{W}_p^c)\|_2^2 \le O\left(\sqrt{\frac{L\sigma^2 Ps}{C}}\right)$

*Under partial broadcasting (i.e.,* $Q < P - 1$*) and set a constant learning rate* $\eta = \frac{1}{CLG(P-Q)}$*, where* $C$ *is the total number of iterations. Then we have*

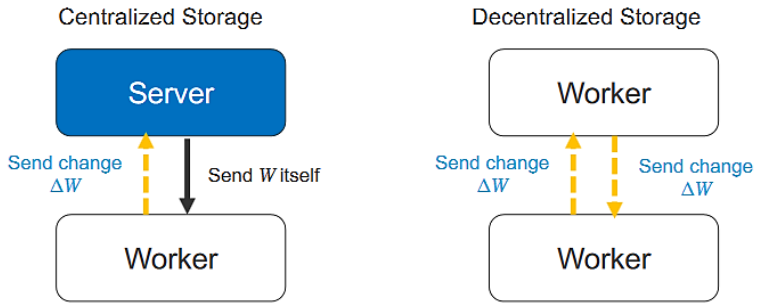
$$\min_{c\le C} \mathbb{E}\left[\|\sum_{p=1}^P \nabla F_p(\mathbf{W}_p^c)\|_2^2\right] \le O\left(LG(P-Q) + \frac{P(sG+\sigma^2)}{CG(P-Q)}\right).$$

*Hence, the algorithm converges to a* $O(LG(P-Q))$ *neighbourhood if* $C \to \infty$.

# BAB 7
# PENYIMPANAN PARAMETER DAN PARADIGMA KOMUNIKASI

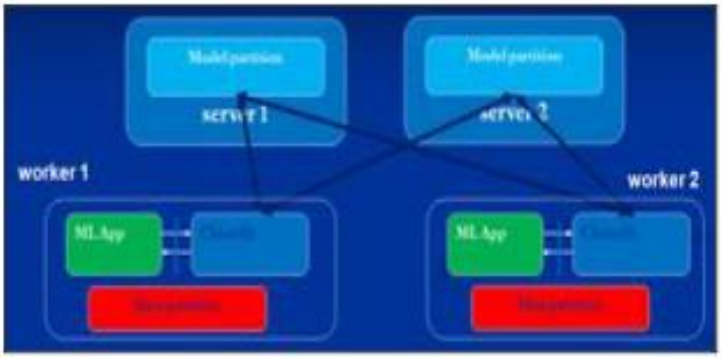
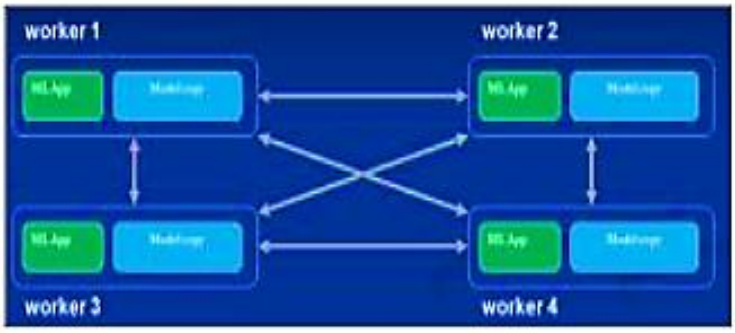
Jenis wewenang atau otoritas pada sebuah manajemen database terbagi menjadi 2, yaitu Sentralisasi dan Desentralisasi. Terpusat atau sentralisasi: mengirim parameter W sendiri dari server ke pekerja. Keuntungan: memungkinkan topologi komunikasi yang ringkas, mis. Bipartit Kelebihan sistem sentralisasi Kemudahan dalam berkoordinasi karena adanya unity of command. Kemudahan dalam mengendalikan sistem manajerial. Terjadi pemusatan keahlian (expertise), dimana hal tersebut dapat dimanfaatkan secara maksimal karena pemberian wewenang dari pemimpin. Implementasi kebijakan umum terhadap keseluruhan dapat dilaksana dengan lebih mudah.

Tujuan dari sentralisasi adalah: Mencegah setiap daerah menjadi mandiri yang berpotensi pada konflik kepentingan atau bahkan memisahkan diri. Memudahkan penerapan kebijakan umum dan pelaksanaannya di setiap daerah. Memudahkan dan mempercepat proses pengambilan keputusan yang secara tidak langsung menunjukkan suatu kepemimpinan yang kuat. Untuk lebih jelasnya marilah kita perhatikan skema dibawah ini.u



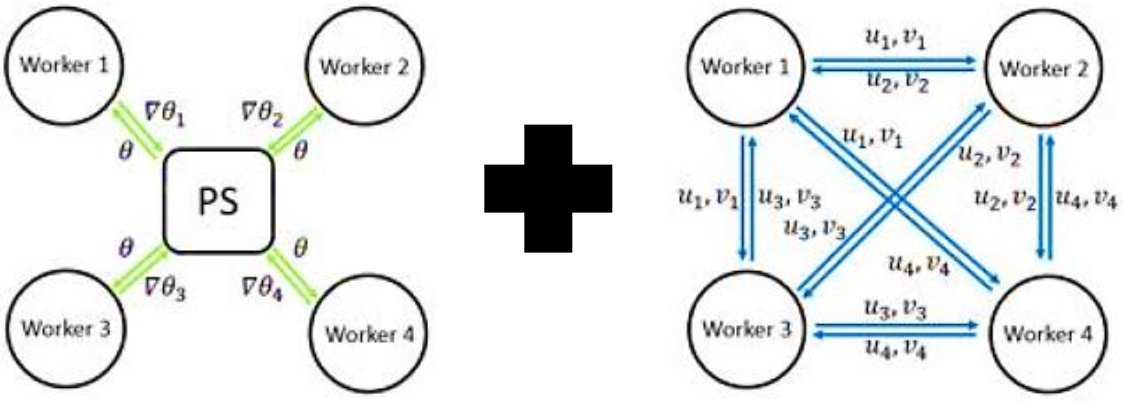

Penyimpanan terdesentralisasi adalah sistem penyimpanan data di beberapa cloud dan server terdesentralisasi alih-alih satu lokasi terpusat. Tidak seperti model terpusat, di mana penyimpanan data terkonsentrasi di tangan beberapa raksasa teknologi, penyimpanan terdesentralisasi adalah solusi penyimpanan cloud peer-to-peer (P2P) di mana pengguna biasa mengelola sistem. Terdesentralisasi: selalu mengirim perubahan ΔW antar pekerja. Keunggulannya adalah kode lebih kuat, homogen, komunikasi rendah. Keuntungan: kode yang sama untuk semua pekerja; tidak ada satu titik kegagalan, elastisitas tinggi terhadap penyesuaian sumber daya.

Berikut ini topologi yang disarankan untuk pengaplikasian penyimpanan Terpusat dan desentralisasi.

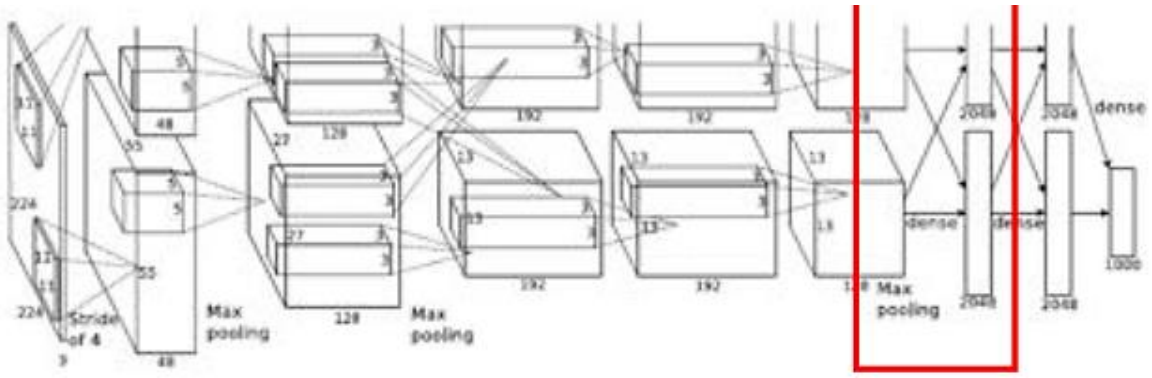
| | |
|---|---|
| *Sentralisasi* |  |
| *Desentralisasi* |  |

## 7.1 MODEL HYBRID PS (*PARAMETER STORAGE*) DAN SFB (*SUFFICIENT FACTOR BROADCASTING*)

Permodelan yang akan dibahas adalah Komunikasi hibrid yang mencakup Server Parameter + Sufficient Factor broadcasting, topologi yang digunakan dalam parameter ini adalah topologi Sentralisasi atau Terpusat, sedangkan untuk SFB atau penyiaran data menggunakan topologi peer yo peer. Jadi untuk data campuran matriks besar dan kecil, berarti pengiriman matriks kecil melalui topologi terpusat sedangkan matriks besar menggunakan topologi peer to peer. Mengapa demikian, agar lalulintas dalam pertukaran data tidak terjadi kemacetan.



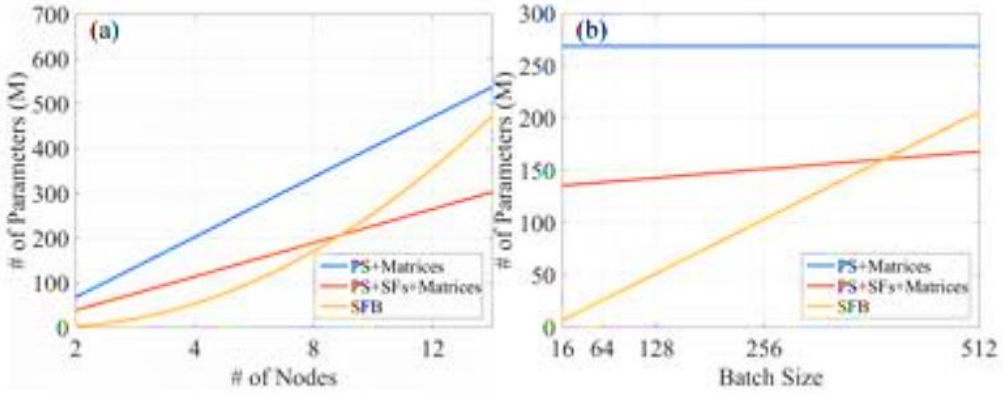

Contoh Hibrida yang akan kita bahas adalah permodelan AlexNet CNN adalah Poseidon
Lapisan akhir = matriks 4096 * 30000 (parameter 120M)
Komunikasi hibrida ini menggunakan komunikasi peer to peer yang dipecah menjadi dua vektor 4096: u, v, lalu mengirimkan ke 2 vektor tersebut.




***Komunikasi Hibrid***

Ide komunikasi hibrid ini adalah Sinkronkan lapisan FC menggunakan SFB dan Sinkonisasi Convergensi tiap lapisan menggunakan penyimanan terpusat. Efektifitas kominikasi hybrid ini secara langsung mengurangi ukuran pesan dalam banyak situasi. Apakah SFB selalu optimal?

Jawabanya Tidak, karena beban komunikasinya meningkat secara kuadra. Untuk permasalahan seperti ini, strategi yang tepat adalah pilihlah PS setiap kali menghasilkan komunikasi yang kurang.




Berikut adalah Algoritma komunikasi hibrid

**Algorithm 1** Get the best comm method of layer $l$

```
function BESTSCHEME(l)
    layer_property = Query(l.name)
    P1, P2, K = Query('n_worker', 'n_server', 'batchsize')
    if layer_property.type == 'FC' then
        M = layer_property.width
        N = layer_property.height
        if 2K(P1 − 1)(M + N) ≤ 2MN(P1+P2−2)/P2 then
            return 'SFB'
        end if
    end if
    return 'PS'
end function
```

Jenis lapisan: CONV atau FC?

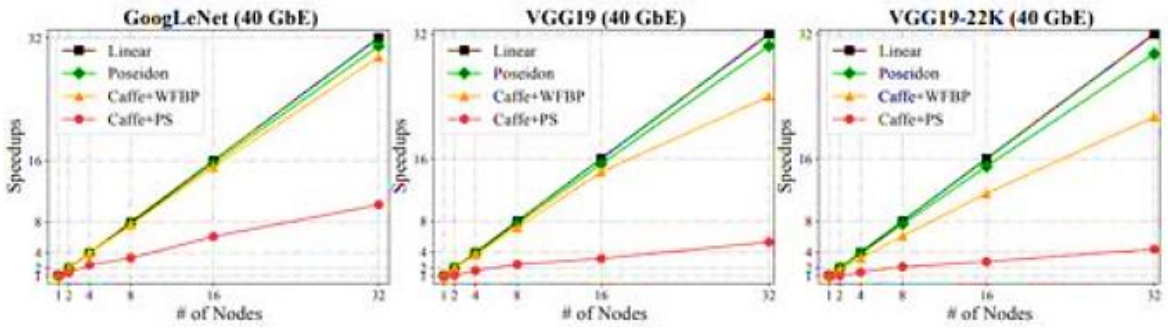
Ukuran lapisan: M, N

Ukuran batch: K

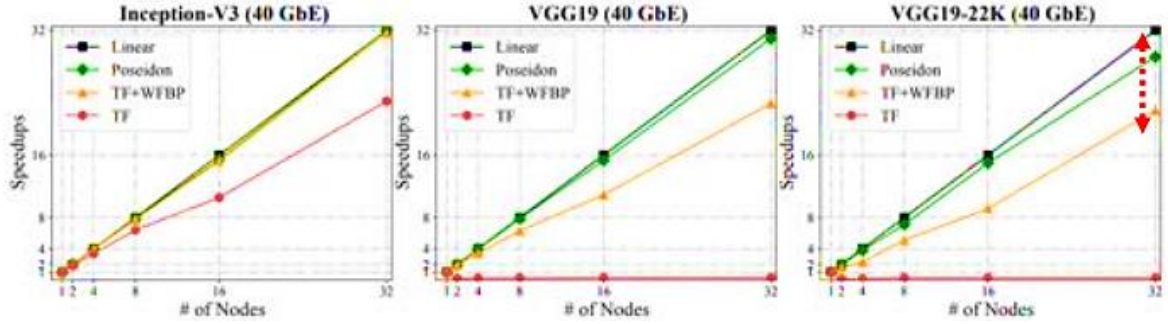
# node Gugus: $P_1$, $P_2$

Menggunakan algoritma, penulis telah mencoba ke dalam beberapa toolkit. Hasil mencapai skalabilitas linier di berbagai model/data dengan bandwidth 40GbE

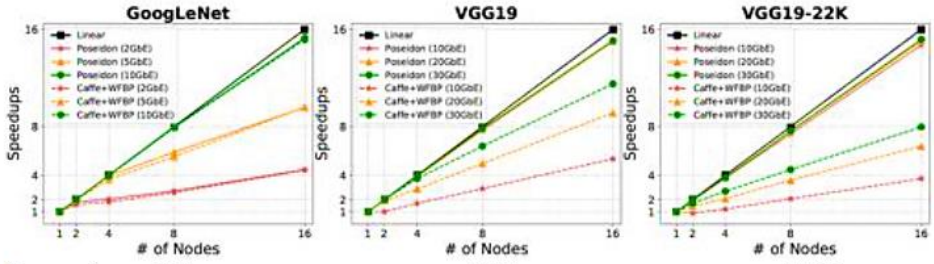
- Menggunakan Caffe sebagai mesin:




- Menggunakan TensorFlow sebagai engine

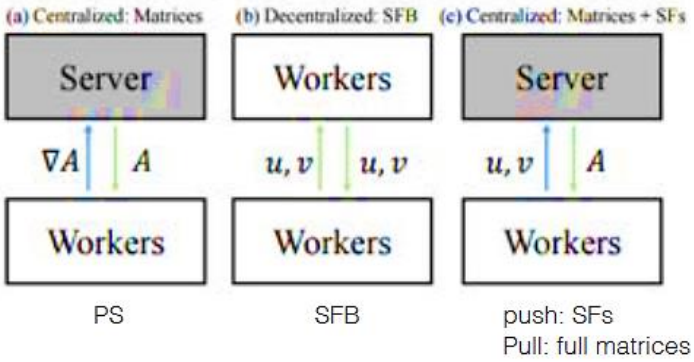



Skalabilitas linear pada throughput, bahkan dengan bandwidth terbatas, tetapi membuat pembelajaran mendalam terdistribusi menjadi terjangkau.



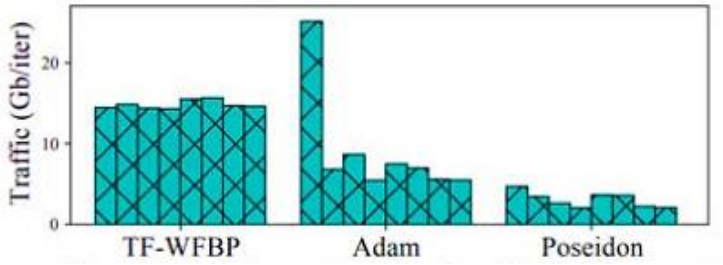

# parameters 5M 143M 229M

| Ethernet | Rate(GBit/s) | Rate (Mb/s) | Rate (# floats/s) |
|---|---|---|---|
| **1 GbE** | 1 | 125 | 31.25M |
| **10 GbE** | 10 | 1250 | 312.5M |
| **Infiband** | 40 | 5000 | 1250M |




Menggunakan microsoft Adams sebagai Strategi Adam menyebabkan kemacetan komunikasi

- Mendorong SF ke server baik-baik saja
- Menarik matriks penuh kembali akan membuat hambatan pada node server.




Komunikasi hibrid menghasilkan penyeimbangan beban komunikasi, yang penting untuk mengatasi masalah burst communication.

## 7.2 POSEIDON

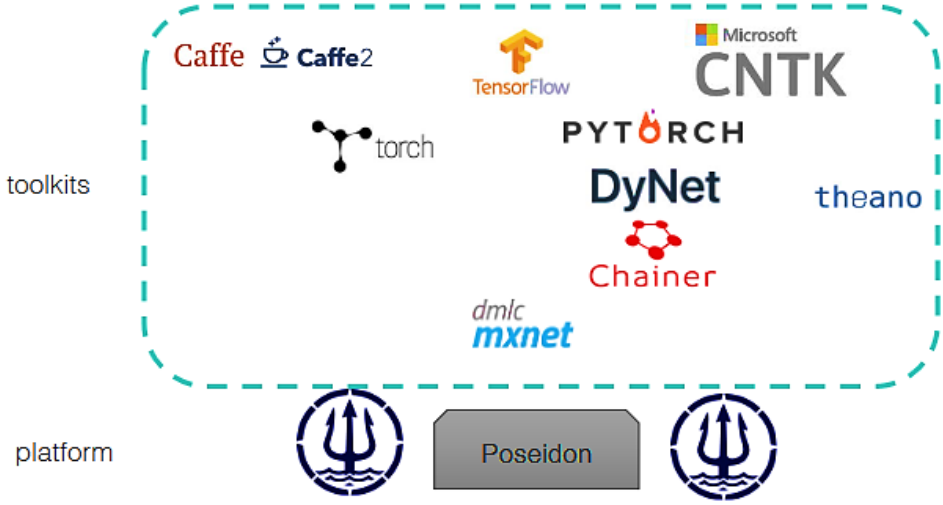
Poseidon adalah arsitektur komunikasi yang efisien untuk DL terdistribusi pada GPU. Poseidon mengeksploitasi struktur model berlapis dalam program DL untuk tumpang tindih komunikasi dan perhitungan, mengurangi komunikasi jaringan yang meledak. Selain itu, Poseidon menggunakan skema komunikasi hibrid yang mengoptimalkan jumlah byte yang diperlukan untuk menyinkronkan setiap lapisan, menurut properti lapisan dan jumlah mesin. Kami menunjukkan bahwa Poseidon dapat diterapkan pada kerangka kerja DL yang berbeda dengan menghubungkan Poseidon ke Caffe dan TensorFlow. Kami menunjukkan bahwa Poseidon memungkinkan Caffe dan TensorFlow mencapai kecepatan 15,5x pada 16 mesin GPU tunggal.

Poseidon mengeksploitasi struktur lapis demi lapis berurutan dalam program DL, menemukan operasi komputasi GPU independen dan operasi komunikasi jaringan dalam algoritme pelatihan, sehingga keduanya dapat dijadwalkan bersama untuk mengurangi komunikasi jaringan yang meledak. Selain itu, Poseidon menerapkan skema komunikasi hibrid yang memperhitungkan sifat matematika setiap lapisan program DL serta konfigurasi kluster, untuk menghitung biaya jaringan dari berbagai metode komunikasi, dan memilih yang termurah – saat ini, Poseidon mengimplementasikan dan mendukung parameter skema server yang cocok untuk matriks kecil, dan skema penyiaran faktor yang memadai yang bekerja dengan baik pada matriks besar.

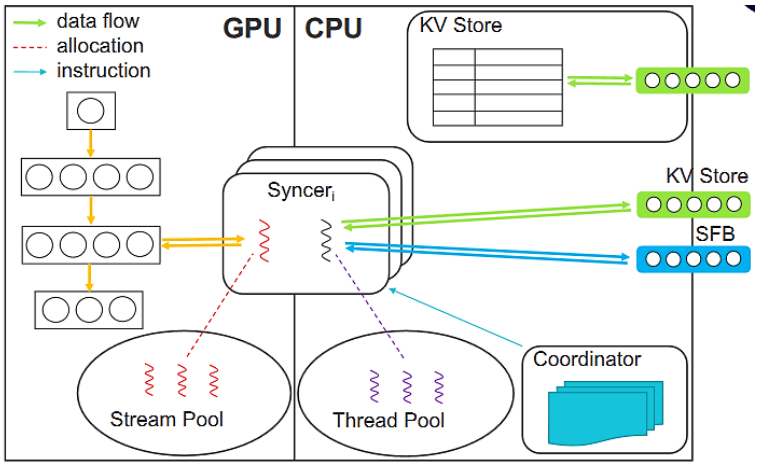
Poseidon secara efektif mengurangi overhead komunikasi dengan secara otomatis mengkhususkan metode komunikasi terbaik untuk setiap lapisan, dan mampu menjaga

penskalaan linier dengan throughput. Dibandingkan dengan metode pengurangan komunikasi lainnya, Poseidon menunjukkan keunggulan sistem (peningkatan keluaran algoritme) atau keunggulan statistik (lebih sedikit langkah algoritme atau iterasi untuk mencapai kriteria terminasi tetap). Poseidon tidak banyak menderita beban komunikasi yang tidak seimbang, yang kami temukan terjadi ketika menggunakan strategi faktor yang cukup yang digunakan dalam Proyek Adam.

Penerapan Poseidon ke beberapa kerangka kerja DL, kami mengimplementasikannya ke dalam dua kerangka kerja DL yang berbeda: Caffe dan TensorFlow, dan menunjukkan bahwa Poseidon memungkinkan mereka untuk menskalakan hampir linier dalam throughput algoritme dengan mesin tambahan, sambil mengeluarkan sedikit biaya tambahan bahkan dalam pengaturan mesin tunggal.




**Arsitektur Sistem Poseidon**




Arsitektur Poseidon: pustaka komunikasi C++ yang mengelola komunikasi parameter untuk program DL yang berjalan pada GPU terdistribusi. Ini memiliki tiga komponen utama: koordinator, yang memelihara model dan konfigurasi cluster; Penyimpanan KV, penyimpanan nilai kunci memori bersama yang menyediakan dukungan untuk komunikasi berbasis server

parameter; pustaka klien, yang dicolokkan ke program DL untuk menangani komunikasi parameter.

| Method | Owner | Arguments | Description |
|---|---|---|---|
| `BestScheme` | Coordinator | A layer name or index | Get the best communication scheme of a layer |
| `Query` | Coordinator | A list of property names | Query information from coordinators' information book |
| `Send` | Syncer | None | Send out the parameter updates of the corresponding layer |
| `Receive` | Syncer | None | Receive parameter updates from either parameter server or peer workers |
| `Move` | Syncer | A GPU stream and an indicator of move direction | Move contents between GPU and CPU, do transformations and application of updates if needed |
| `Send` | KV store | updated parameters | Send out the updated parameters |
| `Receive` | KV store | parameter buffer of KV stores | Receive gradient updates from workers |

**Memperkuat kotak alat DL Menggunakan Poseidon**

Untuk menyiapkan pelatihan terdistribusi, program klien (mis., Caffe) pertama-tama membuat Poseidon dengan membuat koordinator dalam prosesnya. Koordinator pertama-tama akan mengumpulkan informasi yang diperlukan, termasuk informasi klaster (misalnya, jumlah pekerja dan node server, alamat IP mereka) dan arsitektur model (misalnya, jumlah lapisan, jenis lapisan, jumlah neuron, dan cara mereka terhubung, dll.). Dengan informasi tersebut, koordinator akan menginisialisasi penyimpanan KV dan pustaka klien dengan dua langkah: (1) mengalokasikan port komunikasi yang tepat untuk setiap PS shard dan peer worker; (2) menentukan parameter apa yang harus ditransmisikan melalui penyimpanan KV dan apa yang oleh SFB, dan hash parameter secara merata ke setiap penyimpanan KV jika perlu, dan simpan pemetaan dalam buku informasi, yang, selama seluruh pelatihan, dipertahankan dan disinkronkan lintas node, dan dapat diakses di tempat lain melalui Query API koordinator. Selain itu, koordinator menyediakan Skema Terbaik API lain yang mengambil lapisan dan mengembalikan skema komunikasi yang optimal untuknya sesuai dengan strategi yang dijelaskan dalam Algoritma.

**Algorithm 1** Get the best comm method of layer $l$

```
1:  function BESTSCHEME(l)
2:      layer_property = Query(l.name)
3:      P1, P2, K = Query('n_worker', 'n_server', 'batchsize')
4:      if layer_property.type == 'FC' then
5:          M = layer_property.width
6:          N = layer_property.height
7:          if 2K(P1 − 1)(M + N) ≤ 2MN(P1+P2−2)/P2 then
8:              return 'SFB'
9:          end if
10:     end if
11:     return 'PS'
12: end function
```

KV diimplementasikan berdasarkan server parameter sinkron massal, dan dibuat oleh koordinator pada daftar mesin "server" yang ditentukan pengguna. Setiap instance penyimpanan KV menyimpan satu shard dari parameter model yang dibagikan secara global dalam bentuk sekumpulan pasangan KV, yang setiap pasangan KV disimpan di potongan DRAM. Poseidon menetapkan ukuran pasangan KV ke ukuran tetap kecil (mis., 2MB), untuk mempartisi dan mendistribusikan parameter model ke node server secara setara, mengurangi

risiko kemacetan Ethernet. Setiap instance toko KV mengelola buffer parameter pada RAM, dan menyediakan API mirip PS, seperti Terima dan Kirim, untuk menerima dan menerapkan pembaruan dari pustaka klien, atau mengirimkan parameter. Ini akan secara teratur memeriksa status parameter saat ini untuk toleransi kesalahan.

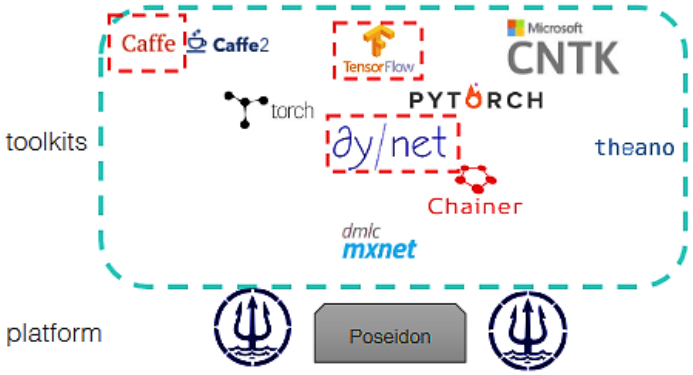
Poseidon berkoordinasi dengan program DL melalui pustaka kliennya. Khususnya, pengguna menyambungkan pustaka klien ke dalam program pelatihan mereka, dan pustaka klien akan membuat sinkronisasi untuk setiap lapisan NN selama perakitan jaringan (sehingga setiap lapisan memetakan satu-ke-satu ke satu sinkronisasi), memperhitungkan sinkronisasi parameternya. Setiap sinkronisasi kemudian diinisialisasi, misalnya, mengatur koneksi ke pecahan PS yang sesuai atau sinkronisasi peer (jarak jauh) sesuai dengan buku informasi koordinator, dan mengalokasikan buffer memori kecil untuk menerima matriks atau SF parameter jarak jauh, dll. Pustaka klien mengelola kumpulan thread CPU dan kumpulan aliran GPU pada mesin pekerja, yang dapat dialokasikan oleh API sinkronisasi saat ada pekerjaan sinkronisasi yang dibuat. GPU terus berkembang pesat dalam daya komputasi, dan terus menghasilkan pembaruan parameter lebih cepat daripada yang dapat disinkronkan secara naif melalui jaringan. Sinkronisasi memiliki tiga API utama, Kirim, Terima, dan Pindahkan, untuk digunakan dalam program klien. Move API menangani pergerakan memori antara RAM dan memori GPU, dan melakukan komputasi yang diperlukan, misalnya, transformasi antara SF dan gradien, dan aplikasi pembaruan. Ini multi-utas menggunakan API asinkron CUDA, dan akan memicu alokasi dari kumpulan utas/aliran perpustakaan klien saat pekerjaan sinkronisasi dimulai (lihat algoritma dibawah ini). Kirim dan Terima adalah API komunikasi yang menyinkronkan parameter lapisan di seluruh replika model yang berbeda. Send API tidak memblokir; itu mengirimkan pembaruan parameter selama backpropagation setelah dihasilkan, mengikuti protokol yang dikembalikan oleh BestScheme API koordinator. Terima API akan dipanggil setelah Kirim selesai. Itu meminta matriks parameter baru dari toko KV atau SF dari sinkronisasi rekannya, dan akan memblokir utasnya saat ini hingga menerima semua yang diminta. Pesan yang diterima dimasukkan ke dalam buffer memori sinkronisasi untuk diambil oleh Move API.

**Algorithm 2** Parallelize a DL library using Poseidon

```
1:  function TRAIN(net)
2:      for iter = 1 → T do
3:          sync_count = 0
4:          net.Forward()
5:          for l = L → 1 do
6:              net.BackwardThrough(l)
7:              thread_pool.Schedule(sync(l))
8:          end for
9:          wait_until(sync_count == net.num_layers)
10:     end for
11: end function
12: function SYNC(l)
13:     stream = stream_pool.Allocate()
14:     syncers[l].Move(stream, GPU2CPU)
15:     syncers[l].method = coordinator.BestScheme(l)
16:     syncers[l].Send()
17:     syncers[l].Receive()
18:     syncers[l].Move(stream, CPU2GPU)
19:     sync_count++
20: end function
```

Sistem DL Terdistribusi berbasis PS. Berdasarkan arsitektur parameter server, sejumlah sistem DL terdistribusi berbasis CPU telah dikembangkan, seperti dan Adam. Mereka murni sistem berbasis PS pada kluster khusus CPU, sedangkan kami menangani kasus kluster GPU yang lebih menantang. Meningkatkan DL pada GPU terdistribusi adalah bidang penelitian yang aktif. Membangun sistem multimesin berbasis GPU untuk DL menggunakan paralelisme model daripada paralelisme data, dan penerapannya agak dikhususkan untuk struktur model tetap sambil menuntut perangkat keras khusus, seperti jaringan InfiBand. TensorFlow adalah platform ML terdistribusi Google yang menggunakan grafik aliran data untuk merepresentasikan model DL, dan menyinkronkan parameter model melalui PS. Oleh karena itu, Poseidon tidak dapat secara dinamis menyesuaikan metode komunikasinya tergantung pada lapisan dan informasi kluster seperti yang dilakukan Poseidon. MXNet adalah sistem DL lain yang menggunakan PS untuk eksekusi terdistribusi, dan mendukung TensorFlow seperti representasi grafik untuk model DL. Dengan autoparalelisasi subgraf independen, kedua kerangka secara implisit tumpang tindih dengan komunikasi dan komputasi.

Sebaliknya, Poseidon memiliki cara yang lebih eksplisit untuk tumpang tindih melalui pustaka kliennya. Karenanya, Poseidon juga dapat digunakan untuk memparalelkan kerangka kerja berbasis non-grafik. Selain itu, MXNet dan TensorFlow tidak mengatasi hambatan yang disebabkan oleh bandwidth jaringan yang terbatas, yang merusak skalabilitasnya saat melatih model besar dengan lapisan padat. Berikut ini skema dari toolkits dan platform Poseidon.



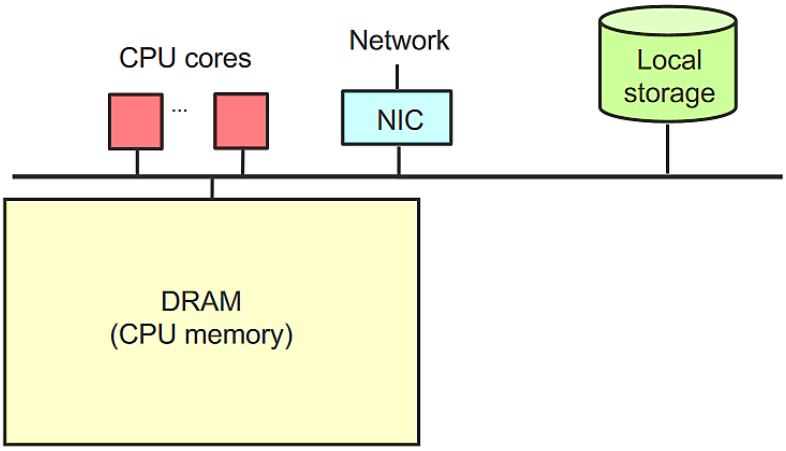

**Masalah yang sering Dihadapi**

Masalah yang sering dihadapi dalam komputasi deep learning adalah ketidakefisiensian. Ketidakefisienan ini terutama disebabkan oleh sinkronisasi parameter melalui jaringan. Dibandingkan dengan CPU, GPU jauh lebih efisien dalam perhitungan matriks; produksi gradien pada GPU jauh lebih cepat daripada yang dapat disinkronkan secara naif melalui jaringan. Akibatnya, perhitungan pelatihan biasanya terhambat oleh komunikasi. Salah satu masalah lain yaitu pada Penyimpanan (Storage), GPU memiliki memori khusus, agar program pelatihan DL menjadi efisien, datanya harus ditempatkan pada memori GPU. Sedangkan yang disediakan pada memori GPU terbatas, dibandingkan dengan CPU, mis. maksimal 12Gb. Masalah lain yang sering tombul adalah Memcopi antara CPU dan GPU mahal karena memerlukan waktu yang sama dengan peluncuran kernel komputasi GPU.

## 7.3 DEEP LEARNING TANPA GPU *(GRAPHICS PROCESSING UNIT)*

Artificial Intelligence (AI) Deep Learning melibatkan jaringan saraf yang perlu dididik (dilatih) untuk memprediksi (menyimpulkan) output secara akurat. Aspek jaringan saraf yang, setelah dilatih, dapat menyimpulkan wawasan tentang data baru yang diberikan padanya. Biasanya ketika orang berpikir tentang pembelajaran mendalam dan daya komputasi yang diperlukan untuk melakukannya secara tepat waktu, mereka biasanya berasumsi bahwa Unit Pemrosesan Grafis (GPU) diperlukan. Mereka juga mengerti bahwa ini bisa sangat mahal. Hal ini berlaku sehubungan dengan pelatihan jaringan saraf, tetapi terkait dengan inferensi, penulis menemukan bahwa, dalam beberapa kasus, kami dapat memperoleh hasil yang lebih baik dengan menggunakan CPU terbaru dari Intel®.

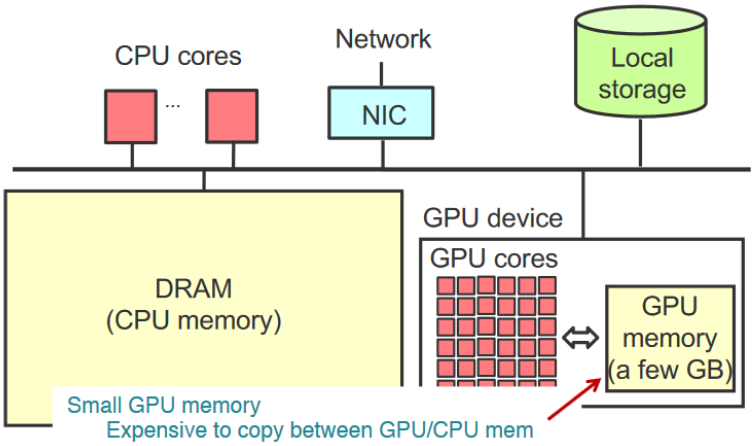
Pelatihan jaringan saraf mencakup banyak propagasi maju dan mundur melewati jaringan saraf untuk meminimalkan fungsi kerugian, sehingga melibatkan perkalian matriks dalam jumlah besar. Pelatihan jaringan saraf memerlukan GPU karena GPU memiliki perangkat keras khusus yang dirancang khusus untuk perkalian matriks. GPU dapat melakukan pelatihan jaringan saraf ratusan kali lebih cepat daripada yang dapat dilakukan CPU. Melatih jaringan saraf yang kompleks dengan kumpulan data pelatihan yang besar biasanya memerlukan beberapa GPU. Namun, setelah model saraf dilatih dan diterapkan, model tersebut dapat menyimpulkan wawasan baru tentang kumpulan data baru berdasarkan pelatihannya. Deep learning inference (DLI) ini hanya melibatkan satu forward pass melalui neural network, dan dengan demikian, komputasinya kurang intensif, dan biasanya hanya membutuhkan satu GPU. Dengan hadirnya CPU terbaru dari Intel, mulai dari prosesor Skylake hingga prosesor Cascade Lake terbaru, DLI kini dapat dilakukan pada CPU canggih ini alih-alih GPU, memberikan penghematan biaya di area ini. Berikut ini adalah skema Deep Learning tanpa mesin GPU




*Mengapa Menggunakan GPU untuk Deep Learning?*

GPU dapat melakukan banyak perhitungan secara bersamaan. Hal ini memungkinkan distribusi proses pelatihan dan dapat mempercepat operasi pembelajaran mesin secara signifikan. Dengan GPU, Anda dapat mengumpulkan banyak inti yang menggunakan lebih sedikit sumber daya tanpa mengorbankan efisiensi atau daya. Saat merancang arsitektur deep learning, keputusan Anda untuk menyertakan GPU bergantung pada beberapa faktor:

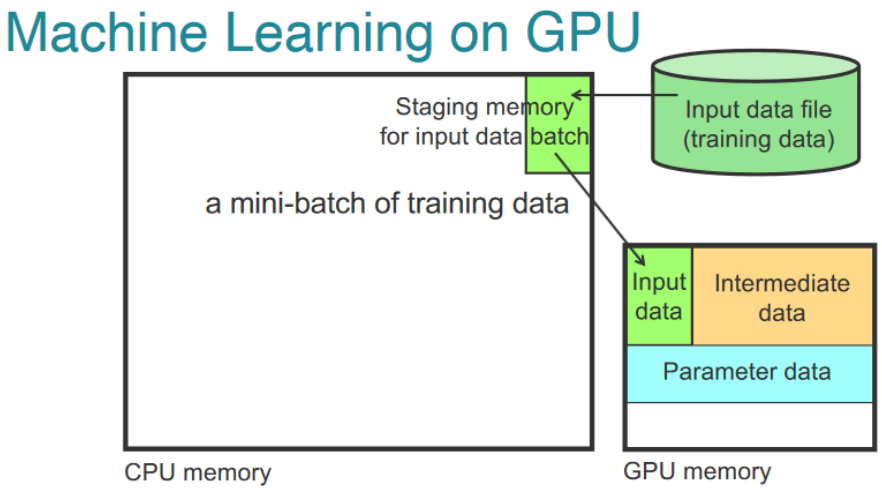
- Bandwidth memori—termasuk GPU dapat menyediakan bandwidth yang diperlukan untuk mengakomodasi kumpulan data besar. Ini karena GPU menyertakan RAM video khusus (VRAM), memungkinkan Anda mempertahankan memori CPU untuk tugas lain.
- Ukuran set data—GPU secara paralel dapat diskalakan lebih mudah daripada CPU, memungkinkan Anda memproses set data besar dengan lebih cepat. Semakin besar kumpulan data Anda, semakin besar manfaat yang dapat Anda peroleh dari GPU.
- Pengoptimalan—kelemahan GPU adalah pengoptimalan tugas individu yang berjalan lama terkadang lebih sulit dibandingkan dengan CPUSkema dibawah ini adalah mesin deep learning dengan GPU




## 7.4 PEMBELAJARAN MESIN PADA GPU *(GRAPHICS PROCESSING UNIT)*

Apakah kita memerlukan GPU baik pribadi atau di Cloud untuk Deep Learning atau kita dapat bekerja dengan CPU saja?

Jika bekerja dengan gambar, kita pasti membutuhkan GPU karena memproses gambar di GPU jauh lebih cepat daripada di CPU. kita juga dapat bekerja dengan CPU tetapi akan lambat. Dan jika kita akan bermain dengan arsitektur jaringan dan parameter lainnya, kita akan melakukan banyak trial and error dan akan cukup membuat frustrasi jika kita hanya menggunakan CPU karena akan memakan banyak waktu.



**Pembelajaran Mendalam di GPU**

Bagian penting dari algoritma pembelajaran pelatihan adalah penggunaan data pelatihan. Pemanfaatan penyimpanan data besar-besaran di lingkungan cloud memberi pengembang banyak sumber daya untuk tujuan itu. Bagian penting lainnya dari pembelajaran mesin adalah menggunakan kekuatan pemrosesan yang cukup untuk memproses sejumlah besar informasi untuk mengajari mesin cara bertindak dan cara memberi daya pada mesin saat mereka beroperasi dalam skenario dunia nyata.

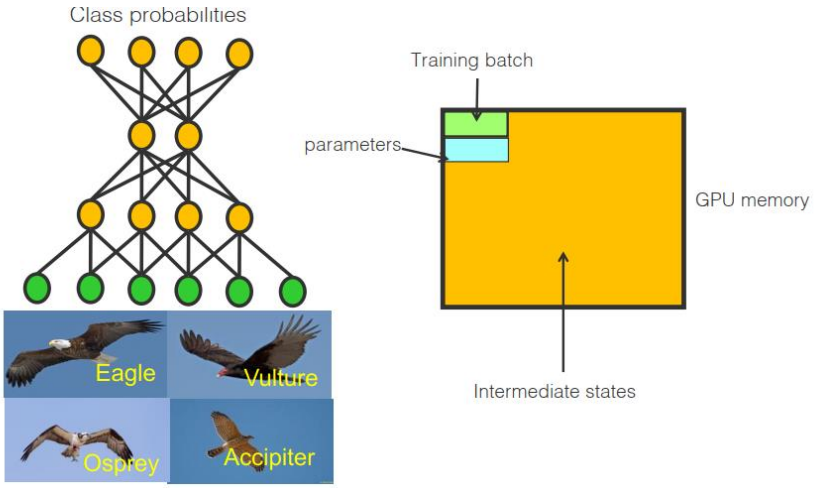
GPU mengatasinya dengan menerapkan perangkat keras dan konfigurasi komputasi untuk masalah tertentu. Pendekatan komputasi paralel ini, yang dikenal sebagai arsitektur Single Instruction, Multiple Data (SIMD), memungkinkan para insinyur untuk mendistribusikan tugas dan beban kerja dengan operasi yang sama secara efisien di seluruh inti GPU.

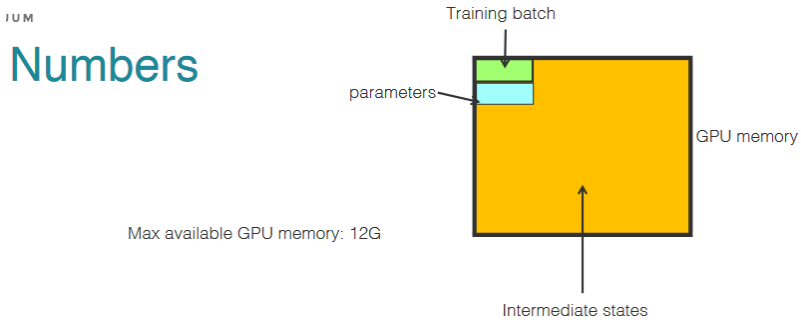
Jadi mengapa GPU cocok untuk pembelajaran mesin? Karena inti dari pelatihan mesin adalah permintaan untuk memasukkan kumpulan data berkelanjutan yang lebih besar untuk memperluas dan menyempurnakan apa yang dapat dilakukan oleh algoritme. Semakin banyak data, semakin baik algoritme ini dapat belajar darinya. Hal ini terutama terjadi pada algoritme pembelajaran mendalam dan jaringan saraf, di mana komputasi paralel dapat mendukung proses multi-langkah yang kompleks. Karena teknologi GPU telah menjadi produk yang sangat dicari tidak hanya untuk industri pembelajaran mesin tetapi juga untuk komputasi pada umumnya, ada beberapa GPU kelas konsumen dan perusahaan di pasaran.

Secara umum, jika Anda mencari GPU yang sesuai dengan konfigurasi perangkat keras pembelajaran mesin, maka beberapa spesifikasi yang lebih penting untuk unit tersebut akan mencakup hal berikut:

- Bandwidth memori tinggi: Karena GPU mengambil data dalam operasi paralel, mereka memiliki bandwidth memori yang tinggi. Tidak seperti CPU yang bekerja secara berurutan (dan meniru paralelisme melalui pengalihan konteks), GPU dapat mengambil banyak data dari memori secara bersamaan. Bandwidth lebih tinggi dengan VRAM lebih tinggi biasanya lebih baik, tergantung pekerjaan Anda.
- Inti tensor: Inti tensor memungkinkan penggandaan matriks yang lebih cepat dalam inti, meningkatkan throughput dan mengurangi latensi. Tidak semua GPU hadir dengan inti tensor, tetapi seiring kemajuan teknologi, mereka menjadi lebih umum, bahkan pada GPU tingkat konsumen.
- Memori bersama yang lebih signifikan: GPU dengan cache L1 yang lebih tinggi dapat meningkatkan kecepatan pemrosesan data dengan membuat data lebih tersedia — namun harganya mahal. GPU dengan lebih banyak cache umumnya lebih disukai, tetapi ini merupakan trade-off antara biaya dan kinerja (terutama jika Anda mendapatkan GPU dalam jumlah besar.)
- Interkoneksi: Solusi cloud atau lokal yang menggunakan GPU untuk beban kerja berperforma tinggi biasanya akan memiliki beberapa unit yang saling terhubung satu sama lain. Namun, tidak semua GPU bekerja dengan baik satu sama lain, jadi pahamilah bahwa pendekatan terbaik adalah memastikan bahwa keduanya dapat bekerja sama dengan mulus.

Penting untuk dicatat bahwa pembelian GPU bukanlah sesuatu yang biasanya dilakukan oleh operasi skala besar kecuali mereka memiliki cloud pemrosesan khusus sendiri. Sebagai gantinya, organisasi yang menjalankan beban kerja pembelajaran mesin akan membeli ruang cloud (baik publik maupun hybrid) yang disesuaikan untuk komputasi performa tinggi. Penyedia cloud ini (idealnya) akan menyertakan GPU berkinerja tinggi dan memori cepat di platform mereka. Berikut ini adalah skema dari pembelajaran mesin yang menggunakan GPU.




| Network | Batch size | Input size | Parameters + grads | Intermediate states |
|---|---|---|---|---|
| AlexNet | 256 | 150MB | <500M | 4.5G |
| GoogLeNet | 64 | 19MB | <40M | 10G |
| VGG19 | 16 | 10MB | <1.2G | 10.8G |

**Menghemat Memory.**

Ide dasar dalam menghemat memoru adalah ukuran batch besar dengan mengumpulkan gradien yang dihasilkan oleh ukuran batch lebih kecil yang terjangkau dalam memori GPU. Karena faktanyakeadaan antara sebanding dengan ukuran bets K. Solusi yang digunakan adalah membagi K menjadi M bagian, setiap bagian memiliki sampel K/M, Untuk iter = 1:M, ujikan dengan ukuran mini-batch K/M, lalu Akumulasi gradien pada GPU tanpa memperbarui parameter model dan perbarui semua parameter model saat semua bagian M selesai. Kelemahannya adalah bagaimana jika GPU masih tidak mampu membeli kondisi peralihan meskipun K=1? Dan Ukuran batch yang kecil biasanya menyebabkan penggunaan kemampuan komputasi GPU yang tidak memadai

# DAFTAR PUSTAKA


Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Et Al. Tensorflow: A System For Large-Scale Machine Learning. Arxiv Preprint Arxiv:1605.08695 (2016).

Chen, J., Monga, R., Bengio, S., And Jozefowicz, R. Revisiting Distributed Synchronous Sgd. Arxiv Preprint Arxiv:1604.00981 (2016).

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., And Zhang, Z. Mxnet: A Flexible And Efficient Machine Learning Library For Heterogeneous Distributed Systems. Arxiv Preprint Arxiv:1512.01274 (2015).

Chilimbi, T., Apacible, Y. S. J., And Kalyanaraman, K. Project Adam: Building An Efficient And Scalable Deep Learning Training System. In Osdi (2014).

Coates, A., Huval, B., Wang, T., Wu, D. J., Ng, A. Y., And Catanzaro, B. Deep Learning With Cots Hpc Systems. In Icml (2013).

Collobert, R., Kavukcuoglu, K., And Farabet, C. Torch7: A Matlab-Like Environment For Machine Learning. In Nipsw (2011).

Cui, H., Zhang, H., Ganger, G. R., Gibbons, P. B., And Xing, E. P. Geeps: Scalable Deep Learning On Distributed Gpus With A Gpuspecialized Parameter Server. In Proceedings Of The Eleventh European Conference On Computer Systems (2016), Acm, P. 4.

Dai, W., Kumar, A., Wei, J., Ho, Q., Gibson, G., And Xing, E. P. Analysis Of Highperformance Distributed Ml At Scale Through Parameter Server Consistency Models. In Proceedings Of The 29th Aaai Conference On Artificial Intelligence (2015).

Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A., Tucker, P., Yang, K., And Ng, A. Y. Large Scale Distributed Deep Networks. In Nips (2012).

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., And Darrell, T. Caffe: Convolutional Architecture For Fast Feature Embedding. In Mm (2014).

Krizhevsky, A. Learning Multiple Layers Of Features From Tiny Images. Master's Thesis, University Of Toronto, 2009.

Moritz, P., Nishihara, R., Stoica, I., And Jordan, M. I. Sparknet: Training Deep Networks In Spark. Arxiv Preprint Arxiv:1511.06051 (2015).

Simonyan, K., And Zisserman, A. Very Deep Convolutional Networks For Large-Scale Image Recognition. In Iclr (2015).

Williams Billy M. And Hoel Lester A. 2003. Modeling And Forecasting Vehicular Traffic Flow As A Seasonal Arima Process: Theoretical Basis And Empirical Results. Journal Of Transportation Engineering 129, 6 (2003), 664–672

Yan, Z., Zhang, H., Wang, B., Paris, S., And Yu, Y. Automatic Photo Adjustment Using Deep Neural Networks. Acm Transactions On Graphics (Tog) 35, 2 (2016), 11.

# ALGORITMA dan TEORI KOMPUTASI TERUKUR dalam Pembelajaran Mesin

## (Machine Learning)

Dr. Joseph Teguh Santoso, M.Kom.

## BIODATA PENULIS

Dr. Joseph Teguh Santoso, S.Kom, M.Kom adalah Rektor dari Universitas Sains & Teknologi Komputer (Universitas STEKOM) Semarang yang memiliki banyak pengalaman praktis dalam bidang *e-commerce*sejak Tahun 2002. Beliau mempunyai 3 (tiga) toko *Official Online Store* di China untuk merek Sepeda Raleigh, dengan omzet tahunan pada Tahun 2019 mencapai lebih dari Rp. 35 Milyar rupiah dan terus meningkat. Dr. Joseph T.S memiliki lisensi tunggal sepeda merek "Raleigh" untuk penjualan *Online* di seluruh China. Di samping itu beliau juga memiliki pabrik sepeda dan sepeda listrik merek "Fengjiu", yaitu Pabrik Sepeda Listrik yang masih tergolong kecil di China. Pengalaman beliau malang melintang di dunia *online store* di China seperti Alibaba, Tmall, Taobao, JD, Aliexpress sangat membantu mahasiswa untuk memiliki pengalaman teknis dan praktis untuk membuka toko *online* bersama beliau.

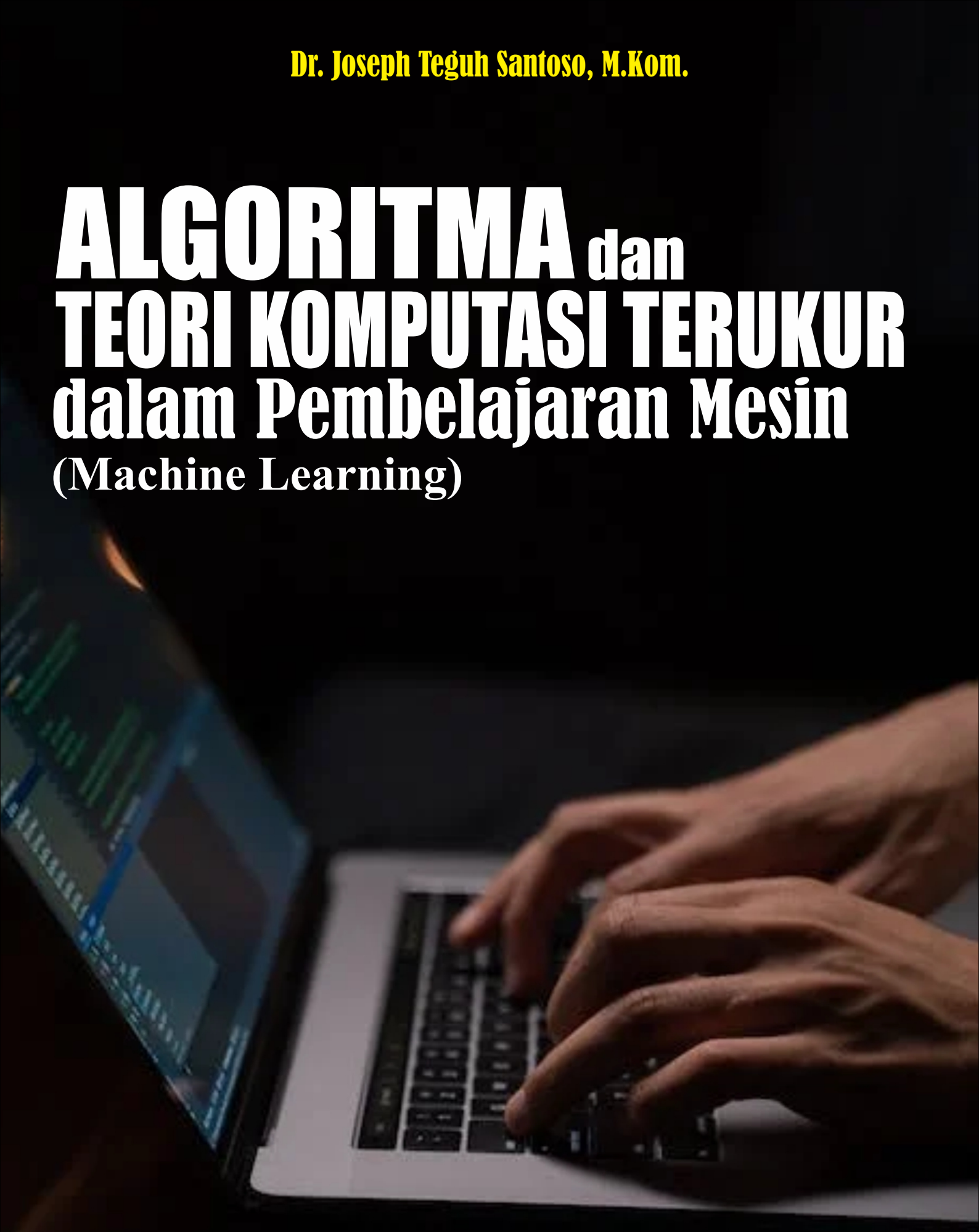
YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8120-40-6 (PDF)

Dr. Joseph Teguh Santoso, M.Kom.

# ALGORITMA dan TEORI KOMPUTASI TERUKUR dalam Pembelajaran Mesin (Machine Learning)



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id