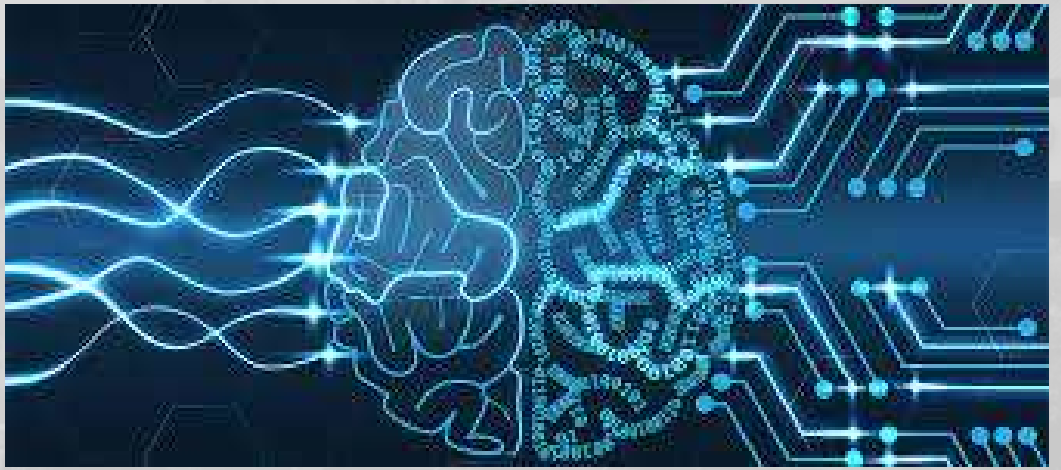
14620323

DEEP LEARNING



Regularization for Deep Learning

PENGAMPU

Dr. Fajar Astuti Hermawati, S.Kom.,M.Kom.

Elsen Ronando, S.Si.,M.Si

Bagus Hardiansyah, S.Kom.,M.Si

Andrey Kartika Widhy H., S.Kom., M.Kom.

Capaian Pembelajaran

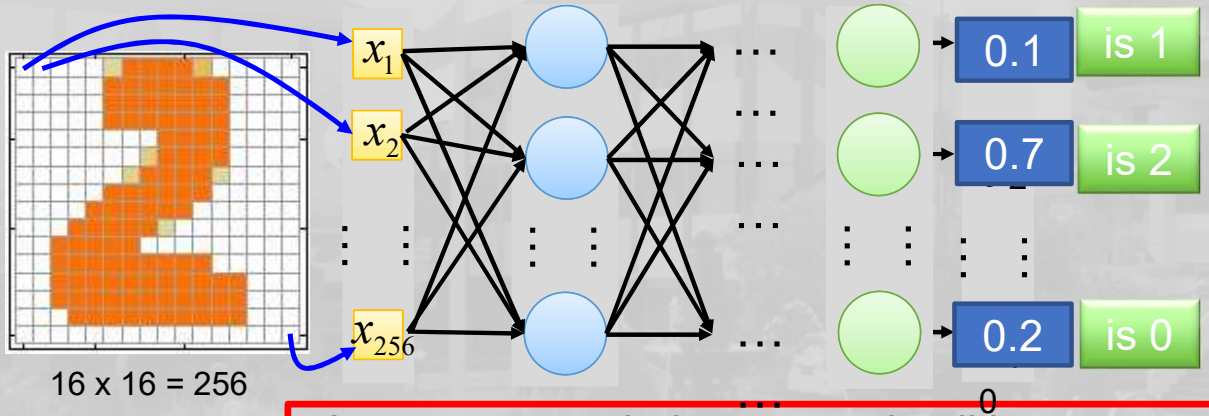
- Sub-CPMK-3: Mampu mengidentifikasi konsep dasar jaringan syaraf tiruan dalam (deep feedforward network) serta regularisasi dan optimisasi pembelajaran dalam deep learning dan mampu mengaplikasikan pemodelan serta evaluasinya untuk menyelesaikan contoh permasalahan yang diberikan [C3, A3]

Bahan Kajian

- Neural Network Parameters
- **Parameters vs Hyperparameters**
- Train / Dev / Test sets
- Performa dari Model
- Classification Model Error
- Problem of fitting
- Parameter Norm Penalties
  - L^2 Parameter Regularization
  - L^1 Parameter Regularization
- Dataset Augmentation
- Multitask Learning
- Early Stopping
- Sparse Representations
- Bagging
- Dropout

Neural Network Parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



16 x 16 = 256

Ink → 1

No ink → 0

Atur parameter jaringan θ sedemikian rupa ……

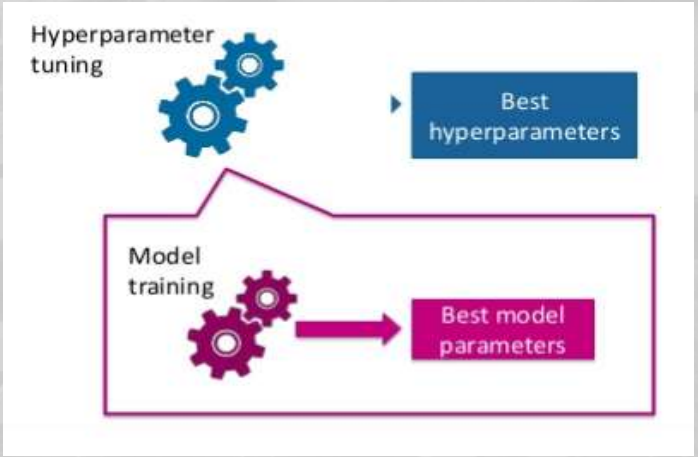
Input:  → y_1 memiliki nilai maksimum

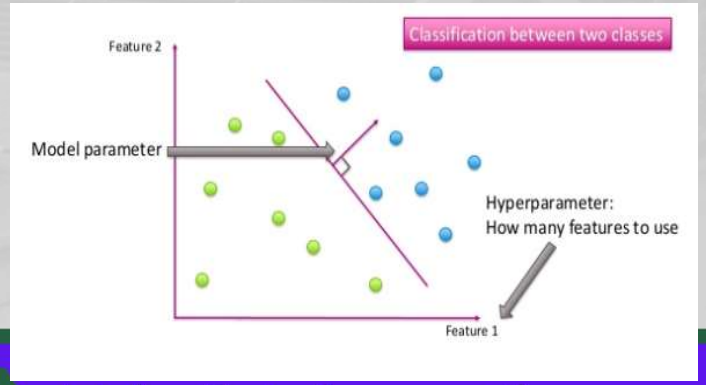
Input:  → y_2 memiliki nilai maksimum

Bagaimana mengatur jaringan saraf mencapai ini

Parameters vs Hyperparameters

- Parameter model adalah variabel dari model yang dipilih yang dapat diestimasi dengan menyesuaikan data yang diberikan ke model.
- Hyperparameter adalah parameter dari distribusi sebelumnya; yang menangkap keyakinan sebelumnya sebelum data diamati.
  - Ini adalah parameter yang mengontrol parameter model
  - Dalam algoritma pembelajaran mesin apa pun, parameter ini perlu diinisialisasi sebelum melatih model.



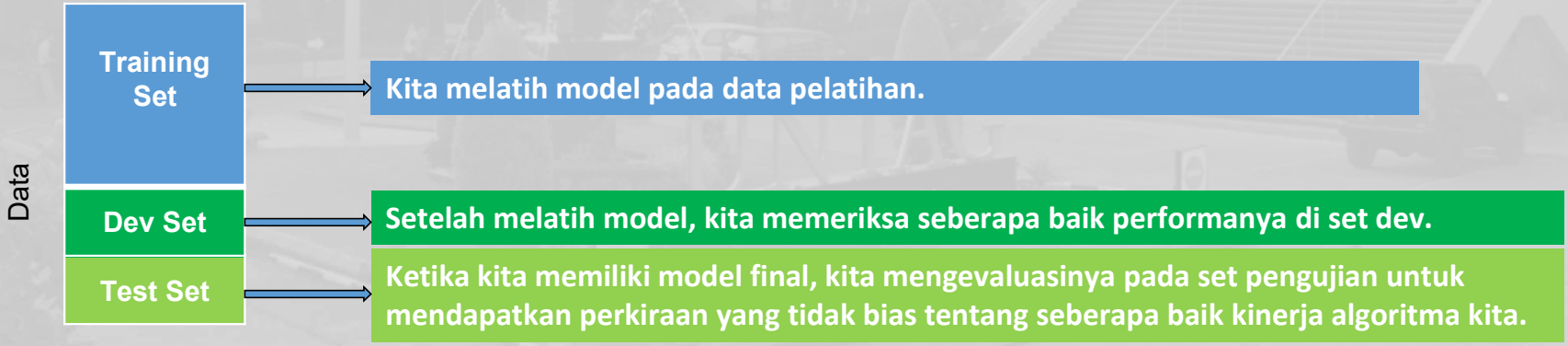


Parameters vs Hyperparameters

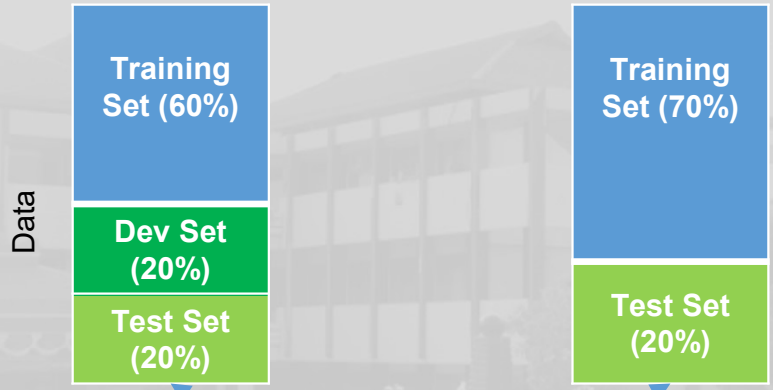
- Parameters:
  - $W^1, b^1, W^2, b^2, \cdots W^L, b^L$
- Hyperparameters:
  - Learning rate (α) in gradient descent
  - Number of iterations in gradient descent
  - Number of layers in a Neural Network
  - Number of neurons per layer in a Neural Network
  - Activations Functions
  - Mini-batch size
  - Regularizations parameters

Train / Dev / Test sets

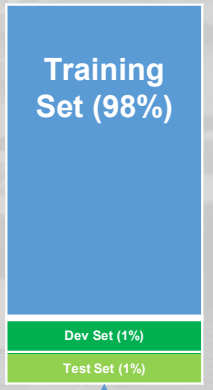
- Penyesuaian hyperparameter adalah proses yang sangat berulang, di mana kita:
  - mulai dengan sebuah ide, yaitu mulai dengan sejumlah lapisan tersembunyi, tingkat pembelajaran tertentu, dll.
  - mencoba ide dengan menerapkannya
  - bereksperimen seberapa baik ide itu berhasil
  - menyempurnakan ide dan mengulangi proses ini
- Sekarang bagaimana kita mengidentifikasi apakah ide itu berhasil? Di sinilah himpunan train / dev / test berperan.



Train / Dev / Test sets



Sebelumnya, ketika kita memiliki kumpulan data kecil, paling sering distribusi kumpulan yang berbeda



Karena ketersediaan data telah meningkat dalam beberapa tahun terakhir, kita dapat menggunakan sebagian besar data untuk melatih model

Train / Dev / Test sets

- Pastikan distribusi set dev/test sama dengan set pelatihan
  - Bagilah set pelatihan, dev, dan pengujian sedemikian rupa sehingga distribusinya serupa
  - Lewati set pengujian dan validasi model menggunakan set dev saja

Performa dari Model

- Untuk mengevaluasi performa dari model yang dibangun, perlu dilakukan pengukuran performa, yaitu pengukuran akurasi (***accuracy***) atau tingkat kesalahan (***error rate***). Jika f_{ij} menotasikan jumlah record dari kelas *i* yang berada di kelas *j* pada saat pengujian, maka pengukuran akurasi (*accuracy*) dapat dituliskan dengan persamaan sebagai berikut :
- $$Accuracy = \frac{jumlah\_prediksi\_yg\_benar}{jumlah\_prediksi\_keseluruhan} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$
- Sedangkan tingkat kesalahan (***error rate***) didefinisikasi sebagai berikut :
- $$Error = \frac{jumlah\_prediksi\_yg\_salah}{jumlah\_prediksi\_keseluruhan} = \frac{f_{01} + f_{10}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Classification Model Error

- Ada dua tipe Classification Model Error:
  - Re-substitution errors (training error): error pada saat pembelajaran (*training*):Σ e(t)
  - Generalization errors: error pada saat pengujian (*testing*): Σ e'(t)
- Model yang baik **harus mempunyai training error dan generalization error yang rendah.**

Problem of fitting

- **Underfitting model** adalah model yang mempunyai **training error dan test error yang tinggi**. Hal ini terjadi karena model belum mempelajari struktur data yang sebenarnya.
- **Overfitting model** adalah model yang mempunyai **generalization error lebih tinggi dibandingkan dengan training error**.

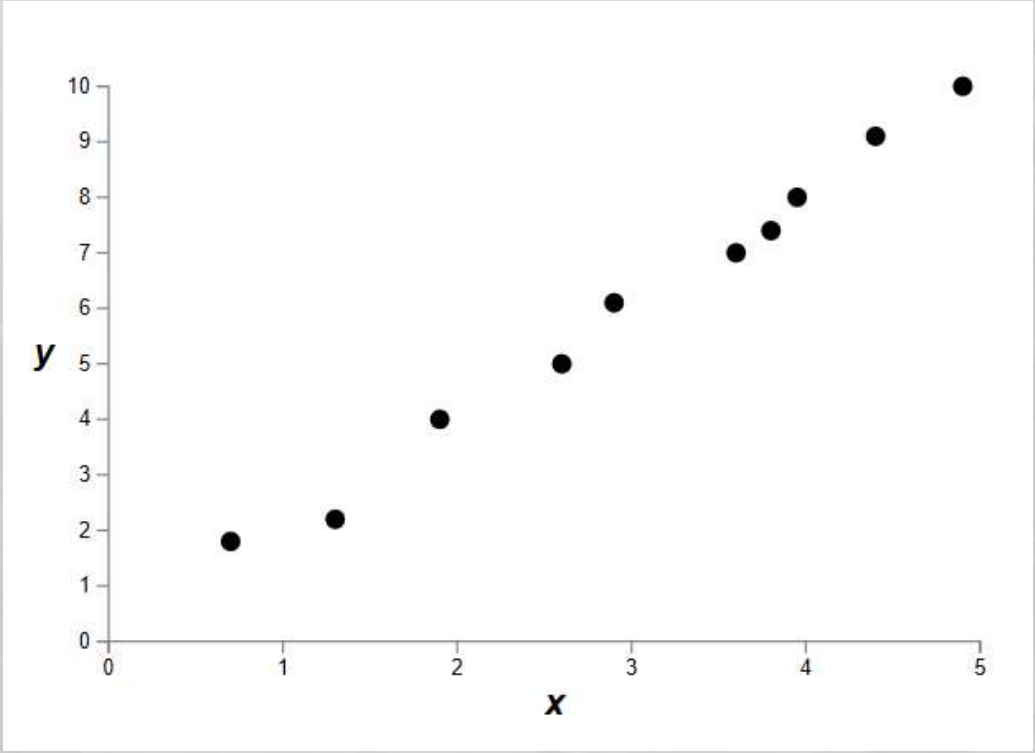
Problem of fitting

- Terlalu banyak parameter = overfitting
- Parameter tidak cukup = underfitting

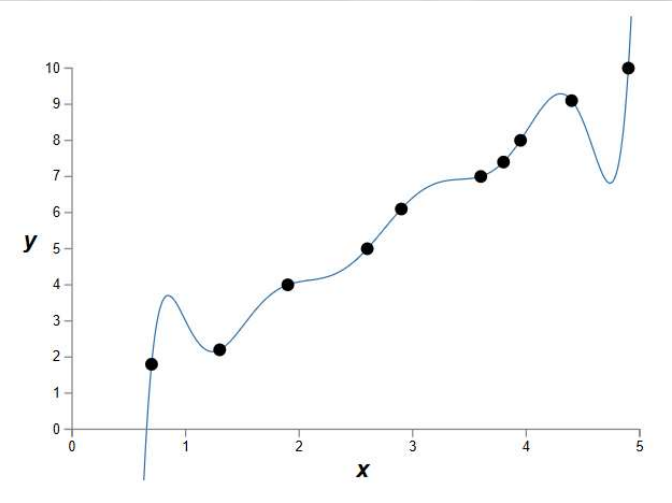
- Lebih banyak data = lebih sedikit kesempatan untuk overfit

- Bagaimana kita tahu apa yang dibutuhkan?

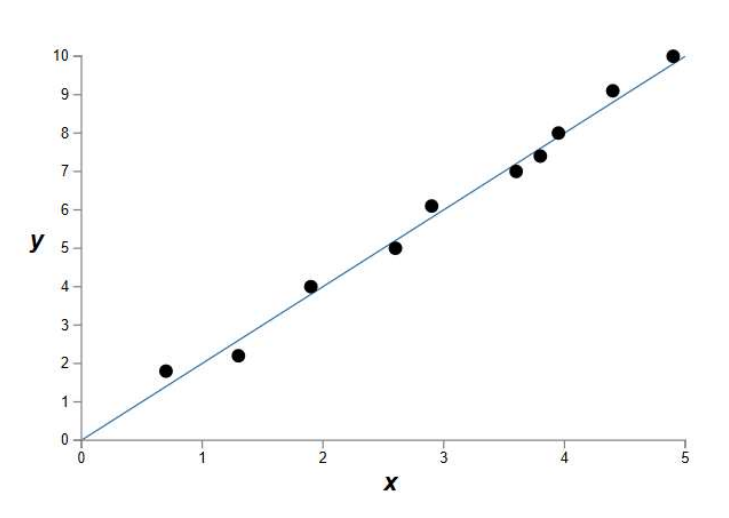
Data fitting problem



Which is better?



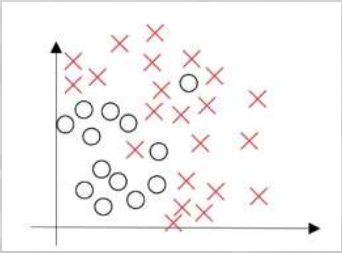
9th order polynomial

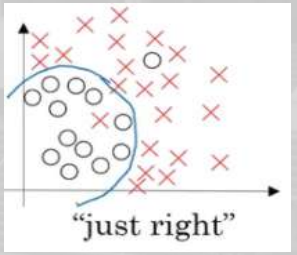


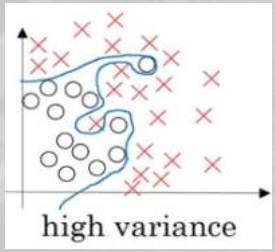
1st order polynomial

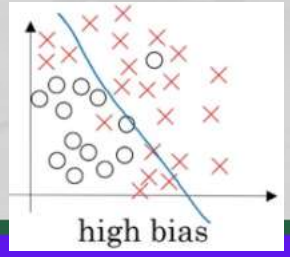
Bias / Variance Trade-off

- Kita ingin **model kita tepat**, yang berarti **memiliki bias rendah dan varians rendah**
- **Overfitting**: Jika kesalahan set dev jauh lebih banyak daripada kesalahan set train, model overfitting dan **memiliki varian yang tinggi**
- **Underfitting**: Ketika kesalahan set train dan dev tinggi, model underfitting dan **memiliki bias tinggi**









Overfitting in Deep Neural Nets

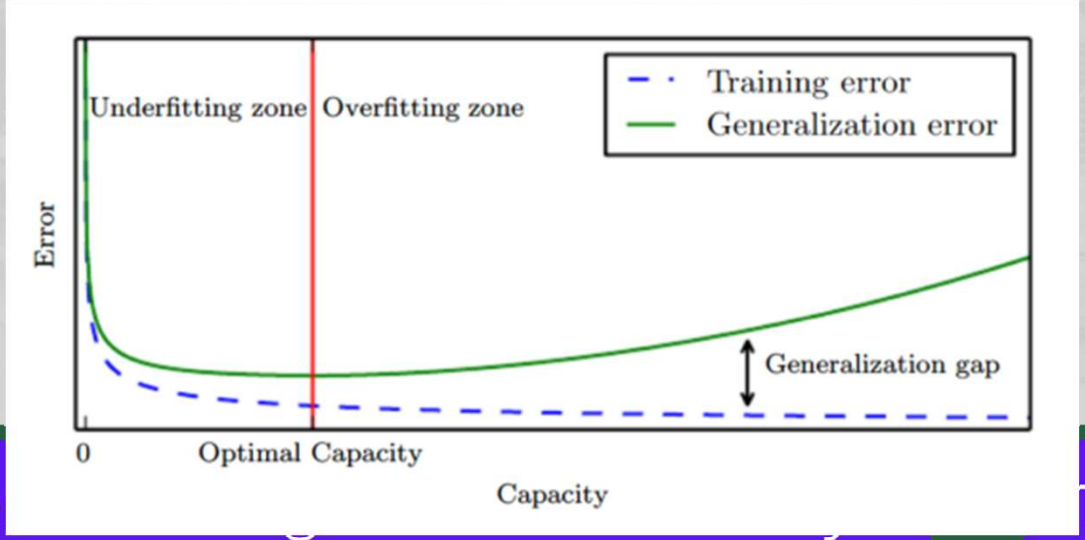
- Jaringan saraf dalam berisi banyak lapisan tersembunyi non-linear
  - Ini membuat mereka menjadi model yang sangat ekspresif yang dapat mempelajari hubungan yang sangat rumit antara input dan output mereka.
  - Dengan kata lain, model mempelajari bahkan detail terkecil yang ada dalam data.
- Tetapi dengan data pelatihan yang terbatas, banyak dari hubungan yang rumit ini akan menjadi hasil dari sampling noise
  - Jadi mereka akan ada di set pelatihan tetapi tidak di data uji nyata meskipun diambil dari distribusi yang sama.
  - Jadi setelah mempelajari semua kemungkinan pola yang dapat ditemukannya, model cenderung bekerja dengan sangat baik pada set pelatihan tetapi gagal menghasilkan hasil yang baik pada set dev dan pengujian.

Regularization

- Mencoba untuk mencari solusi agar tidak overfit
- Namun tetap memberikan kebebasan dengan banyak parameter

Regularization

- Kurangi kesalahan generalisasi bahkan dengan mengorbankan peningkatan kesalahan pelatihan.
- Misalnya, Membatasi kapasitas model adalah metode regularisasi



Regularization

- "**Regularisasi** adalah modifikasi apa pun yang kita buat pada algoritma pembelajaran yang dimaksudkan untuk mengurangi kesalahan generalisasinya (**generalization error**) tetapi bukan kesalahan pelatihannya (**training error**)."

Berbagai bentuk 'regularisasi'

- **Menambahkan lebih banyak data** adalah semacam regularisasi
- **Pooling** adalah semacam regularisasi
- **Augmentasi data** adalah semacam regularisasi

Parameter Norm Penalties

- Ide: Hukuman (**penalize**) penggunaan parameter untuk memilih bobot yang kecil *dan menambahkan biaya untuk memiliki bobot yang tinggi*
- Banyak pendekatan regularisasi didasarkan pada **pembatasan kapasitas model**, seperti **jaringan saraf, regresi linier, atau regresi logistik**, dengan **menambahkan penalti norma parameter** $\Omega(\theta)$ ke fungsi tujuan *J*.

Parameter Norm Penalties

- Fungsi tujuan dinyatakan sebagai:

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta}),$$

- di mana $\alpha \in [0, \infty)$ adalah **hyperparameter** yang memberi bobot kontribusi relatif dari **Norm Penalties**, Ω, relatif terhadap fungsi tujuan standar *J*.
  - Menetapkan $\alpha = 0$ **tidak menghasilkan regularisasi**.
  - Nilai ***α* yang lebih besar**, **lebih banyak regularisasi**

Parameter Norm Penalties

- Pada jaringan saraf, biasanya menggunakan norma parameter, **penalti Ω**,
  - yang hanya mempenalti **bobot** (weight) dari **transformasi affine** pada setiap lapisan dan
  - membiarkan **bias (nilai ambang / threshold) tidak diatur**.
- Oleh karena itu, kita menggunakan vektor $\boldsymbol{w}$ untuk menunjukkan semua bobot yang harus dipengaruhi oleh penalti norma, sedangkan vektor $\boldsymbol{\theta}$ menunjukkan semua parameter, termasuk $\boldsymbol{w}$ dan parameter yang tidak diatur.

L^2 Parameter Regularization

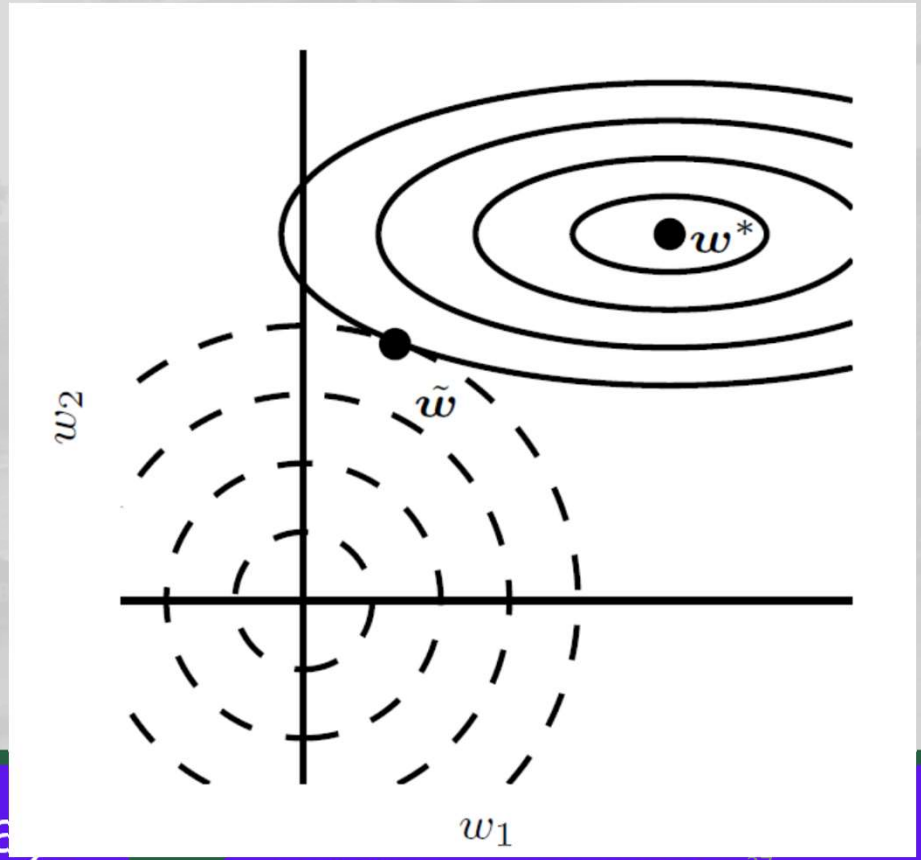
- Parameter Norm Penalties L^2 umumnya dikenal sebagai peluruhan bobot (***weight decay***).
- Strategi regularisasi ini **mendorong bobot lebih dekat ke titik asal** (origin) dengan menambahkan ke fungsi tujuan suku regularisasi:

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

- Di komunitas akademik lain, regularisasi L^2 juga dikenal sebagai **regresi *ridge*** atau **regularisasi Tikhonov**.

L^2 Parameter Regularization

- Ilustrasi pengaruh regularisasi L^2 (atau peluruhan bobot) pada nilai $\boldsymbol{w}$ optimal.
- Elips padat mewakili kontur dengan nilai yang sama dari tujuan yang tidak diregularisasi.
- Lingkaran putus-putus mewakili kontur dengan nilai yang sama dari pengatur L^2.
- Pada titik $\tilde{w}$, tujuan-tujuan yang bersaing ini mencapai keseimbangan.



L^2 Parameter Regularization

- Dalam dimensi pertama,
  - **nilai eigen dari Hessian** of J kecil.
  - Fungsi tujuan tidak bertambah banyak saat bergerak menjauhi $\boldsymbol{w}^*$ secara horizontal.
  - Karena fungsi objektif tidak menunjukkan preferensi yang kuat sepanjang arah ini, regularizer memiliki efek yang kuat pada sumbu ini.
  - Regularizer menarik $\boldsymbol{w}_1$ mendekati nol.
- Pada dimensi kedua,
  - fungsi tujuan sangat sensitif terhadap pergerakan menjauhi $\boldsymbol{w}^*$.
  - Nilai eigen yang sesuai besar, menunjukkan kelengkungan yang tinggi.
  - Akibatnya, penurunan bobot (*weight*) mempengaruhi posisi $\boldsymbol{w}_2$ relatif kecil

L^1 Regularization

- Secara formal, regularisasi L^1 pada parameter model w didefinisikan sebagai:

$$\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |w_i|,$$

- yaitu, sebagai penjumlahan dari nilai absolut dari masing-masing parameter

L^1 Regularization

- Dibandingkan dengan regularisasi L^2, regularisasi L^1 menghasilkan solusi yang lebih jarang (sparse).
- Ketersebaran dalam konteks ini mengacu pada fakta bahwa **beberapa parameter memiliki nilai optimal nol**.
- Jarangnya regularisasi L^1 adalah perilaku yang berbeda secara kualitatif dibandingkan dengan regularisasi L^2.

L^1 Regularization

- Properti sparsity yang disebabkan oleh regularisasi L^1 telah digunakan secara luas sebagai **mekanisme pemilihan fitur**.
- **Pemilihan fitur** menyederhanakan masalah pembelajaran mesin dengan memilih subset mana dari fitur yang tersedia yang harus digunakan.
- Model LASSO (Tibshirani, 1995) (***least absolute shrinkage and selection operator***) yang terkenal mengintegrasikan penalti ^{1} dengan model linier dan fungsi biaya kuadrat terkecil. *L*
- **Penalti L^1 menyebabkan subset dari bobot menjadi nol**, menunjukkan bahwa fitur yang bersesuaian dapat dibuang dengan aman

L^1 Regularization

- Regularisasi L^2 setara dengan inferensi MAP Bayesian dengan Gaussian sebelumnya pada bobot.
- Untuk regularisasi L^1, untuk mengatur fungsi biaya setara dengan istilah ***log-prior*** yang dimaksimalkan oleh inferensi Bayesian MAP ketika yang sebelumnya adalah distribusi ***Laplace isotropik*** pada $w \in \mathbb{R}^n$ digunakan penalty: $\alpha\Omega(\boldsymbol{w}) = \alpha \sum_i |w_i|$
- Dengan formula log-prior:

$$\log p(\boldsymbol{w}) = \sum_i \log \text{Laplace}(w_i; 0, \frac{1}{\alpha}) = -\alpha||\boldsymbol{w}||_1 + n \log \alpha - n \log 2.$$

Dataset Augmentation

- Cara terbaik untuk membuat model pembelajaran mesin **menggeneralisasi lebih baik adalah dengan melatihnya pada lebih banyak data**.
- Tentu saja, dalam praktiknya, jumlah data yang kita miliki terbatas.
- Salah satu cara untuk mengatasi masalah ini adalah dengan **membuat data palsu** dan **menambahkannya ke set pelatihan**.
- Untuk beberapa tugas pembelajaran mesin, cukup mudah untuk membuat data palsu baru.

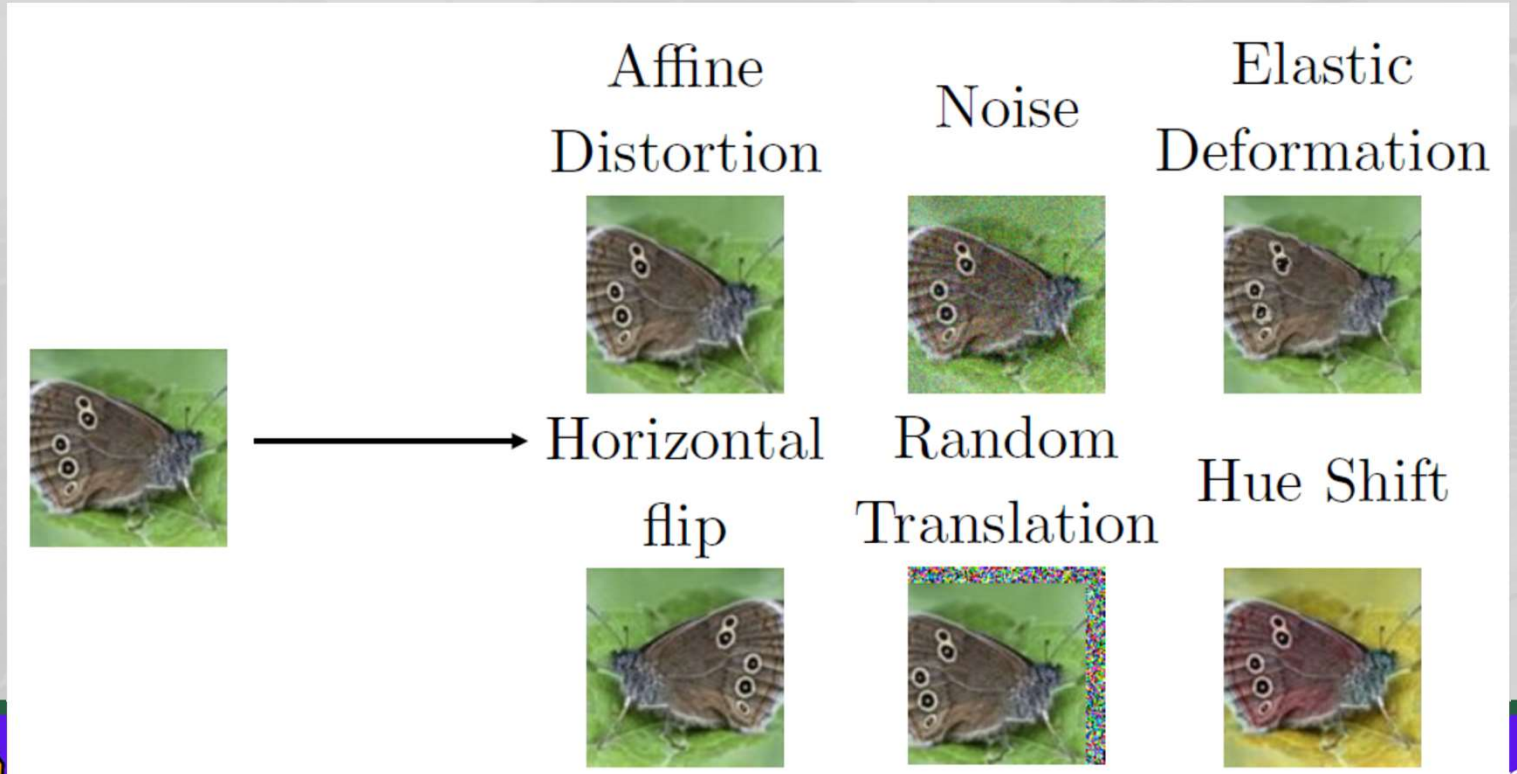
Dataset Augmentation

- Pendekatan ini paling mudah untuk klasifikasi.
- Pengklasifikasi perlu mengambil input x yang rumit dan berdimensi tinggi dan meringkasnya dengan identitas kategori tunggal y.
- Ini berarti bahwa tugas utama yang dihadapi pengklasifikasi adalah menjadi **invarian terhadap berbagai macam transformasi**.
- Kita dapat menghasilkan pasangan (x,y) baru dengan mudah hanya dengan mengubah input x di set pelatihan kita

Dataset Augmentation

- Penambahan dataset telah menjadi teknik yang sangat efektif untuk masalah klasifikasi tertentu: pengenalan objek.
- Gambar berdimensi tinggi dan mencakup sejumlah besar faktor variasi, banyak di antaranya dapat dengan mudah disimulasikan.
- Operasi seperti menerjemahkan gambar pelatihan beberapa piksel di setiap arah seringkali dapat sangat meningkatkan generalisasi, bahkan jika model telah dirancang untuk sebagian invariant translation dengan menggunakan teknik konvolusi dan pooling.
- Banyak operasi lainnya, seperti memutar gambar atau penskalaan gambar, juga terbukti cukup efektif

Dataset Augmentation

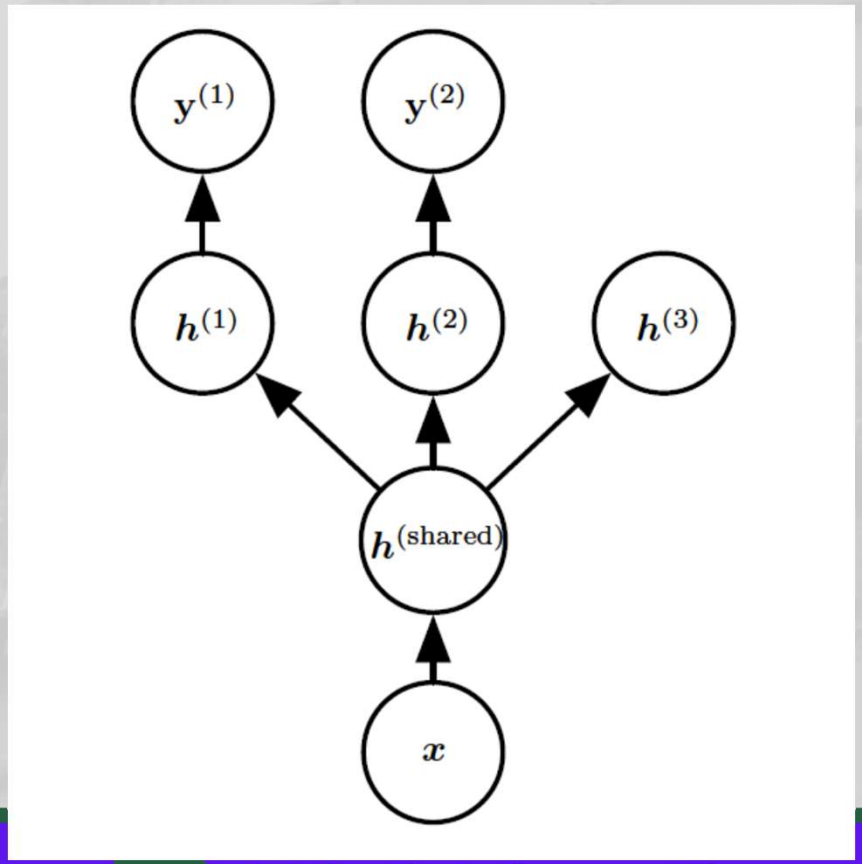


Multitask Learning

- Pembelajaran multitugas (Multitask Learning) (Caruana, 1993) adalah **cara untuk meningkatkan generalisasi dengan menggabungkan contoh-contoh** (yang dapat dilihat sebagai kendala lunak yang dikenakan pada parameter) yang muncul dari beberapa tugas.
- Dengan cara yang sama bahwa contoh pelatihan tambahan memberi lebih banyak tekanan pada parameter model menuju nilai yang digeneralisasi dengan baik, ketika bagian dari model dibagikan di seluruh tugas, bagian model itu lebih dibatasi menuju nilai yang baik (dengan asumsi pembagian itu dibenarkan).), seringkali menghasilkan generalisasi yang lebih baik

Multitask Learning

- Gambar mengilustrasikan bentuk pembelajaran multitugas yang sangat umum, di mana berbagai tugas yang diawasi (memprediksi $y^{(i)}$ yang diberikan x) berbagi input x yang sama, serta beberapa representasi tingkat menengah h (dibagi), menangkap kumpulan faktor yang sama .

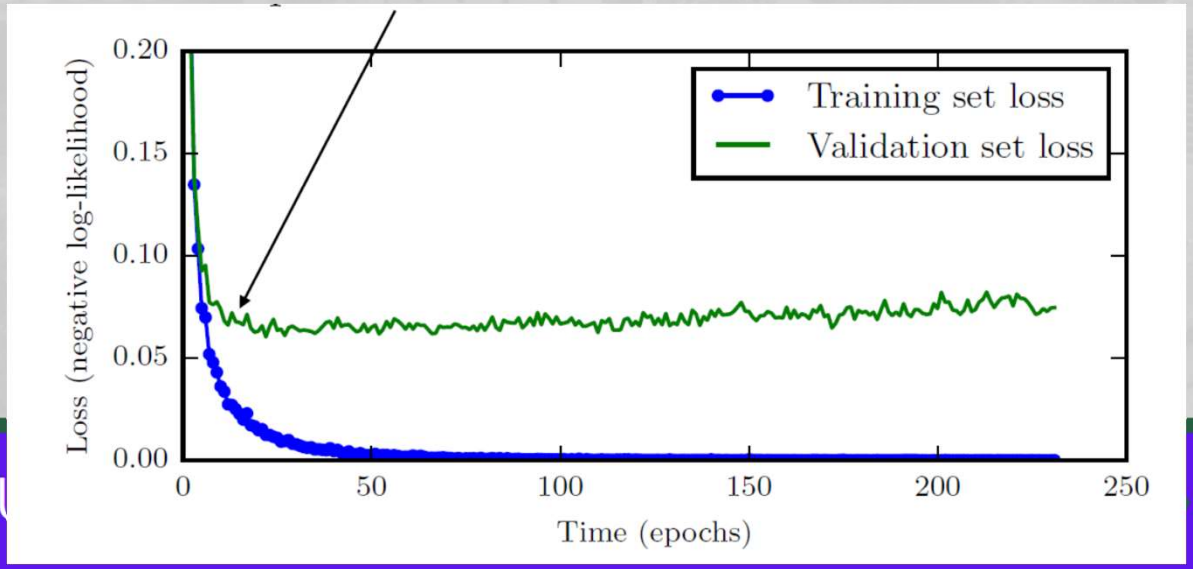


Multitask Learning

- Model umumnya dapat dibagi menjadi dua jenis bagian dan parameter terkait:
  1. ***Task-specific parameters*** - Parameter khusus tugas (yang hanya mendapat manfaat dari contoh tugas mereka untuk mencapai generalisasi yang baik). Ini adalah lapisan atas dari jaringan saraf pada Gambar.
  2. ***Generic parameters*** - Parameter generik, dibagikan di semua tugas (yang diuntungkan dari kumpulan data semua tugas). Ini adalah lapisan bawah jaringan saraf pada Gambar.

Early Stopping

- Saat melatih model besar dengan kapasitas representasional yang cukup untuk mengatasi tugas, kita sering mengamati bahwa **kesalahan pelatihan terus menurun seiring waktu**, tetapi **kesalahan set validasi mulai meningkat lagi**.



Early Stopping

- Ini berarti kita dapat memperoleh model dengan kesalahan set validasi yang lebih baik (dan dengan demikian, diharapkan, kesalahan set pengujian yang lebih baik) dengan kembali ke pengaturan parameter pada titik waktu dengan kesalahan set validasi terendah.
- **Setiap kali kesalahan pada set validasi membaik**, **kita menyimpan salinan parameter model**.
- Saat **algoritma pelatihan dihentikan**, **kita mengembalikan parameter ini**, bukan parameter terbaru.
- Algoritme berhenti ketika tidak ada parameter yang diperbaiki dari kesalahan validasi yang tercatat terbaik untuk sejumlah iterasi yang ditentukan sebelumnya.

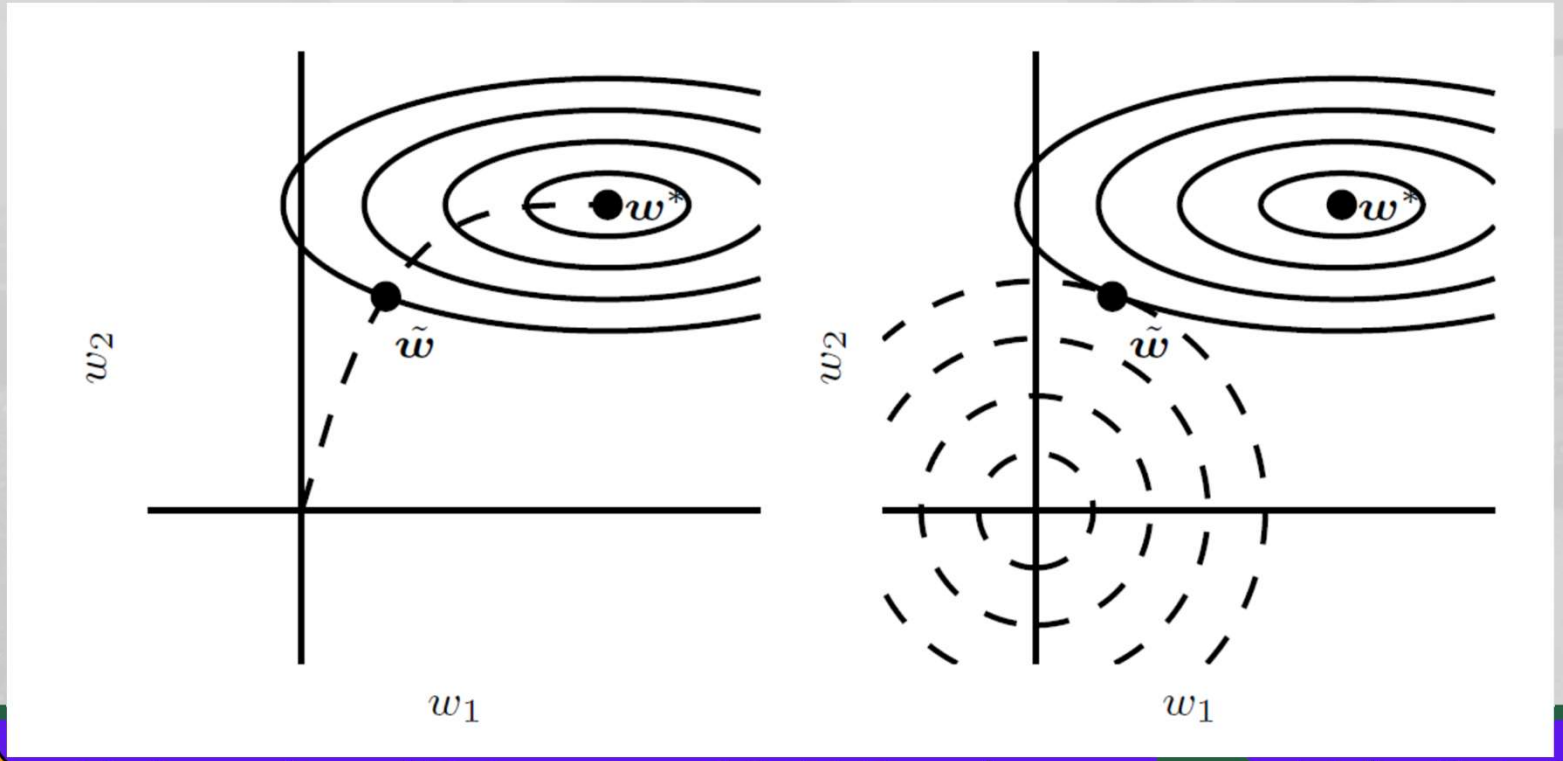
Early Stopping

- Strategi ini dikenal sebagai **penghentian awal**.
- Ini mungkin bentuk regularisasi yang paling umum digunakan dalam pembelajaran mendalam.
- Popularitasnya adalah karena efektivitas dan kesederhanaannya.
- Salah satu cara untuk memikirkan penghentian awal adalah sebagai algoritme pemilihan hyperparameter yang sangat efisien.
- Dalam tampilan ini, jumlah langkah pelatihan hanyalah hyperparameter lainnya.

Early Stopping

- Kita dapat melihat pada Gambar bahwa hyperparameter ini memiliki kurva kinerja set validasi berbentuk U.
- Sebagian besar hyperparameter yang mengontrol kapasitas model memiliki kurva kinerja set validasi berbentuk U.
- Dalam kasus penghentian awal, kita **mengontrol kapasitas efektif** model dengan menentukan berapa banyak langkah yang dapat dilakukan agar sesuai dengan set pelatihan.

Early Stopping and Weight Decay



Early Stopping and Weight Decay

- Ilustrasi efek berhenti lebih awal.
  - (Kiri) Garis kontur padat menunjukkan kontur kemungkinan log negatif. Garis putus-putus menunjukkan lintasan yang diambil oleh SGD mulai dari titik asal. Daripada berhenti di titik $\boldsymbol{w}^*$ yang meminimalkan biaya, penghentian lebih awal menghasilkan lintasan yang berhenti di titik lebih awal $\widetilde{w}$.
  - (Kanan) Ilustrasi efek regularisasi L^2 untuk perbandingan. Lingkaran putus-putus menunjukkan kontur penalti L^2, yang menyebabkan total biaya minimum lebih dekat ke asal daripada biaya minimum yang tidak diatur

Sparse Representations

- Peluruhan bobot bertindak dengan menempatkan penalti langsung pada parameter model.
- Strategi lain adalah memberikan penalti pada aktivasi unit di jaringan saraf, mendorong aktivasi mereka menjadi jarang.
- Ini secara tidak langsung memberikan penalti yang rumit pada parameter model.
- Kita telah membahas bagaimana hukuman L1 menginduksi parameterisasi yang jarang—artinya banyak parameter menjadi nol (atau mendekati nol).

Sparse Representations

- Ketersebaran representasional, di sisi lain, menggambarkan representasi di mana **banyak elemen representasi adalah nol** (atau mendekati nol).
- Pandangan yang disederhanakan dari perbedaan ini dapat diilustrasikan dalam **konteks regresi linier** sebagai berikut: (next slide)

Sparse Representations

- Pada ekspresi pertama, adalah contoh model regresi linier parametrik jarang.
- Yang kedua, adalah regresi linier dengan representasi jarang ***h*** dari data ***x***. dimana, ***h*** adalah fungsi dari ***x*** yang, dalam arti tertentu, merepresentasikan informasi yang ada dalam ***x***, tetapi melakukannya dengan **vektor sparse**

$$\begin{matrix} 18 \\ 5 \\ 15 \\ -9 \\ -3 \end{matrix} = \begin{matrix} 4 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & -1 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & -4 \\ 1 & 0 & 0 & 0 & -5 & 0 \end{matrix} \begin{matrix} 2 \\ 3 \\ -2 \\ -5 \\ 1 \\ 4 \end{matrix}$$

$$\boldsymbol{y} \in \mathrm{R}^{\mathrm{m}} \qquad \boldsymbol{A} \in \mathrm{R}^{\mathrm{m}\times\mathrm{n}} \qquad \boldsymbol{x} \in \mathrm{R}^{\mathrm{n}}$$

$$\begin{matrix} -14 \\ 1 \\ 19 \\ 2 \\ 23 \end{matrix} = \begin{matrix} 3 & -1 & 2 & -5 & 4 & 1 \\ 4 & 2 & -3 & -1 & 1 & 3 \\ -1 & 5 & 4 & 2 & -3 & -2 \\ 3 & 1 & 2 & -3 & 0 & -3 \\ -5 & 4 & -2 & 2 & -5 & -1 \end{matrix} \begin{matrix} 0 \\ 2 \\ 0 \\ 0 \\ -3 \\ 0 \end{matrix}$$

$$\boldsymbol{y} \in \mathrm{R}^{\mathrm{m}} \qquad \boldsymbol{B} \in \mathrm{R}^{\mathrm{m}\times\mathrm{n}} \qquad \boldsymbol{h} \in \mathrm{R}^{\mathrm{n}}$$

Sparse Representations

- Regularisasi representasi dicapai dengan jenis mekanisme yang sama yang telah kita gunakan dalam regularisasi parameter
- Regularisasi penalti norma dari **representasi** dilakukan dengan menambahkan ke fungsi kerugian (loss function) ***J*** sebuah penalty pada representasi.
- Penalty dilambangkan dengan Ω(h). Seperti sebelumnya, fungsi loss:

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{h}),$$

- di mana $\alpha \in [0,\infty)$ memberi bobot pada kontribusi relatif dari ketentuan penalti norma, dengan **nilai α yang lebih besar** sesuai dengan regularisasi yang lebih banyak

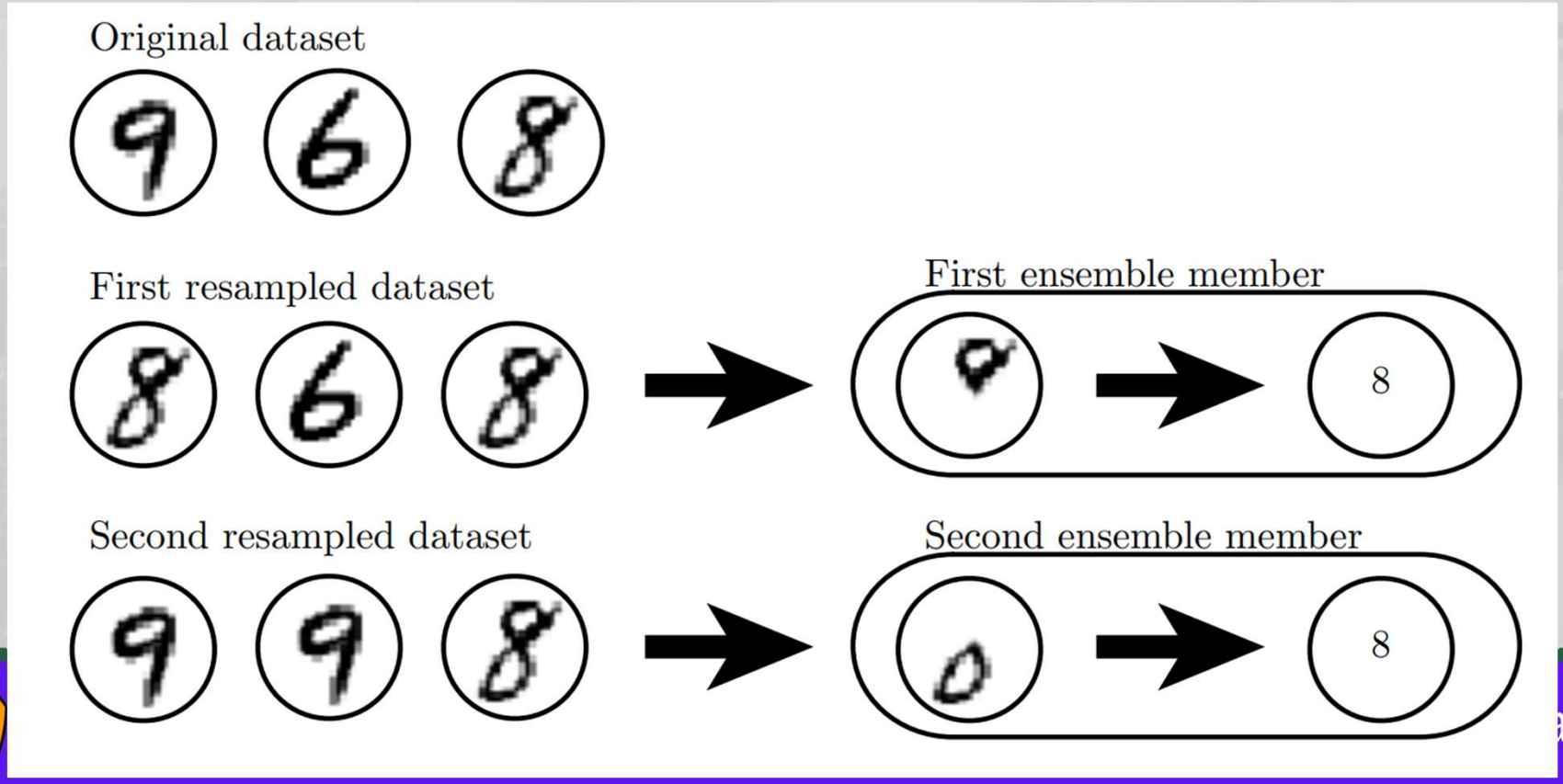
Bagging

- **Bagging** (kependekan dari ***bootstrap aggregating***) adalah teknik untuk mengurangi kesalahan generalisasi dengan **menggabungkan beberapa model** (Breiman, 1994).
- Idenya adalah untuk melatih beberapa model yang berbeda secara terpisah, kemudian meminta semua model memberikan pilihan (***vote***) pada keluaran untuk contoh pengujian.
- Ini adalah contoh strategi umum dalam pembelajaran mesin yang disebut rata-rata model (***model averaging***).
- Teknik yang menggunakan strategi ini dikenal sebagai metode ansambel (***ensemble methods***).

Bagging

- Misalkan kita melatih detektor **8** pada kumpulan data yang digambarkan berikut, yang berisi 8, 6, dan 9.
- Misalkan kita membuat dua kumpulan data sampel ulang yang berbeda.
- Prosedur pelatihan **bagging** adalah untuk membangun masing-masing dataset ini dengan pengambilan sampel dengan penggantian (***replacement***).
  - Dataset pertama menghilangkan angka 9 dan mengulang angka 8. Pada dataset ini, detektor mengetahui bahwa lingkaran di atas digit sesuai dengan angka 8.
  - Pada dataset kedua, kita ulangi angka 9 dan hilangkan angka 6. Dalam hal ini, detektor belajar bahwa loop di bagian bawah digit sesuai dengan 8.
- Masing-masing aturan klasifikasi individu ini rapuh, tetapi jika kita rata-rata outputnya, maka detektornya kuat (***robust***), mencapai kepercayaan maksimal hanya ketika kedua loop dari 8 hadir.

Bagging

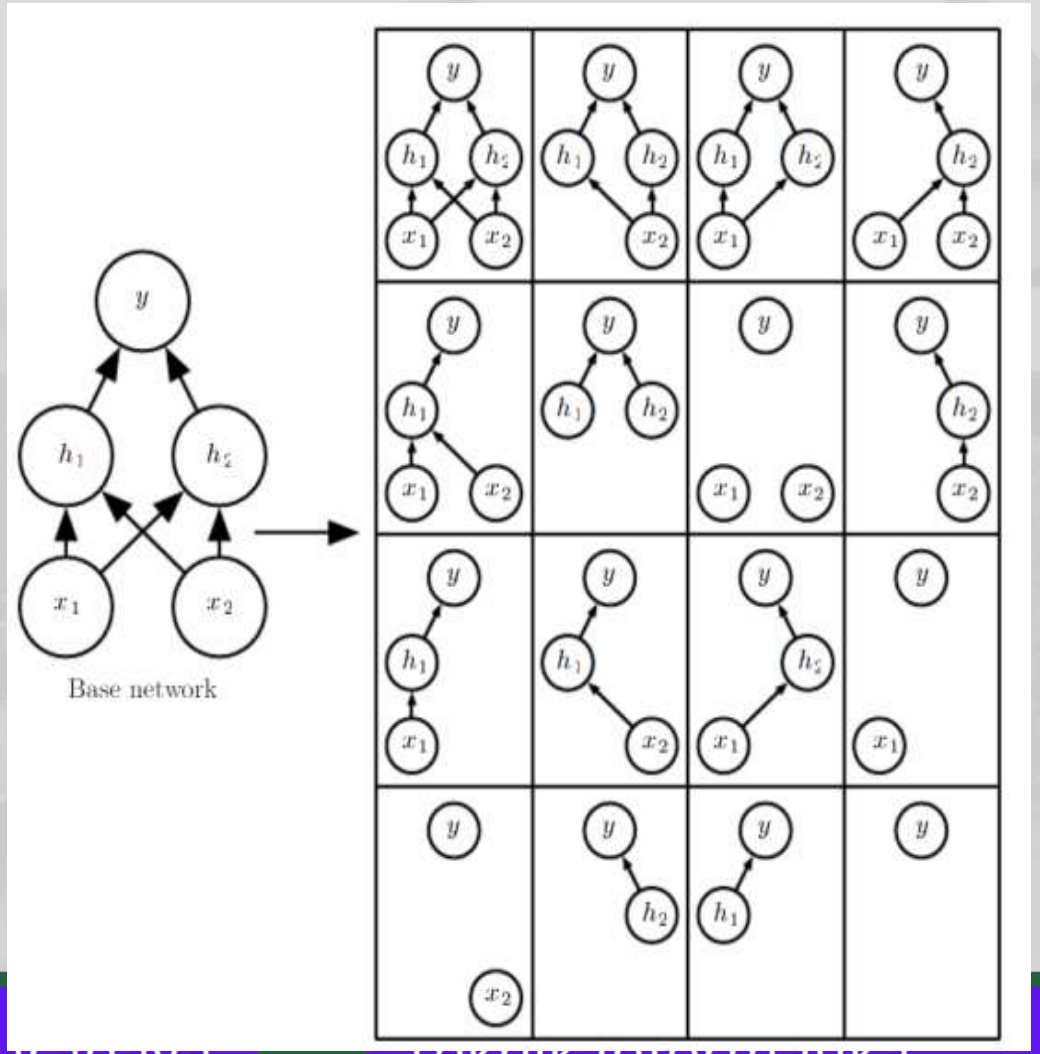

Original dataset
First resampled dataset
First ensemble member
8
Second resampled dataset
Second ensemble member
8

Dropout

- Dropout adalah metode bagging
  - Bagging adalah metode rata-rata atas beberapa model untuk meningkatkan generalisasi, yang
    - Tidak praktis untuk melatih banyak jaringan saraf karena mahal dalam hal waktu dan memori
  - Ini adalah metode bagging yang diterapkan pada jaringan saraf
- Dropout adalah metode yang murah namun kuat untuk mengatur keluarga model yang luas

Dropout

- Secara khusus, dropout melatih ansambel yang terdiri dari semua **sub-jaringan** yang dapat dibentuk dengan **menghapus unit non-output** dari jaringan dasar yang mendasarinya.

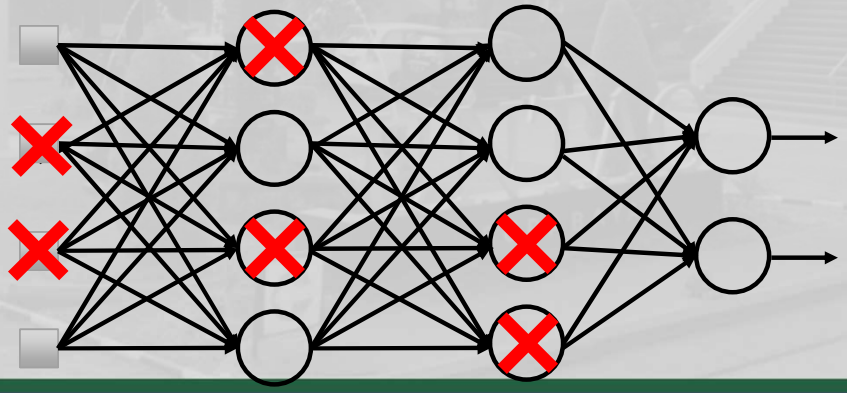


Dropout

- Pada gambar diatas, kita mulai dengan jaringan dasar dengan dua unit yang terlihat dan dua unit yang tersembunyi (hidden).
- Ada **enam belas himpunan bagian** yang mungkin dari empat unit ini.
- Ditunjukkan semua enam belas subnetwork yang dapat dibentuk dengan mengeluarkan subset unit yang berbeda dari jaringan asli.
- Dalam contoh kecil ini, **sebagian besar jaringan yang dihasilkan tidak memiliki unit masukan atau jalur yang menghubungkan masukan ke keluaran**.
- Masalah ini menjadi tidak signifikan untuk jaringan dengan lapisan yang lebih luas, di mana kemungkinan menjatuhkan semua kemungkinan jalur dari input ke output menjadi lebih kecil.

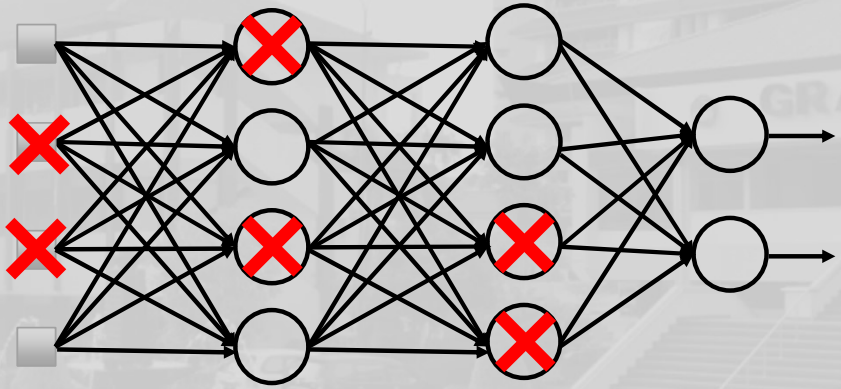
Dropout - Intuitive Reason

- Saat bekerja sama, jika semua orang berharap pasangannya akan melakukan pekerjaan, akhirnya tidak ada yang akan dilakukan.
- Namun, jika Anda tahu pasangan Anda akan dropout, Anda akan melakukannya dengan lebih baik.
- Saat pengujian, sebenarnya tidak ada yang dropout, sehingga memperoleh hasil yang baik pada akhirnya.



Dropout

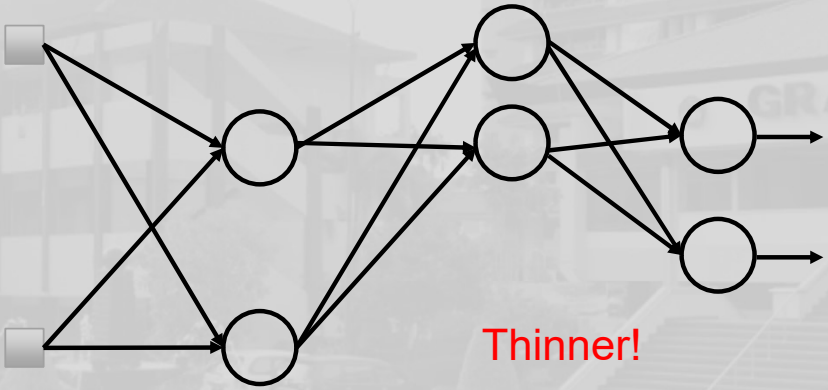
Training:



- **Setiap kali sebelum menghitung gradien**
  - Setiap neuron memiliki ***p***% untuk dropout

Dropout

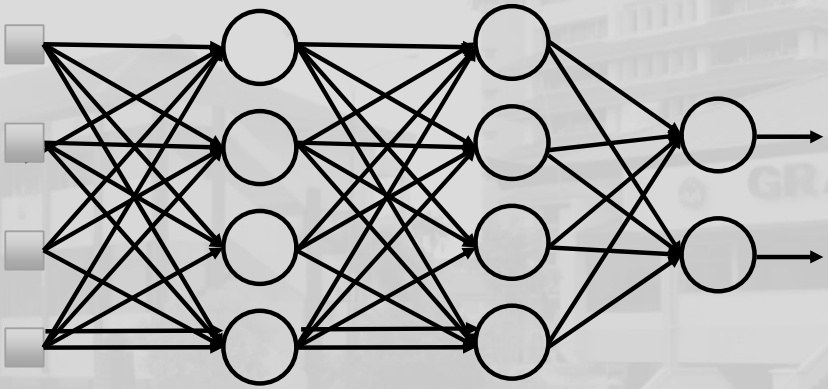
Training:



- **Setiap kali sebelum menghitung gradien**
  - Setiap neuron memiliki ***p***% untuk dropout
    - **Struktur jaringan diubah.**
  - Menggunakan jaringan baru untuk pelatihan

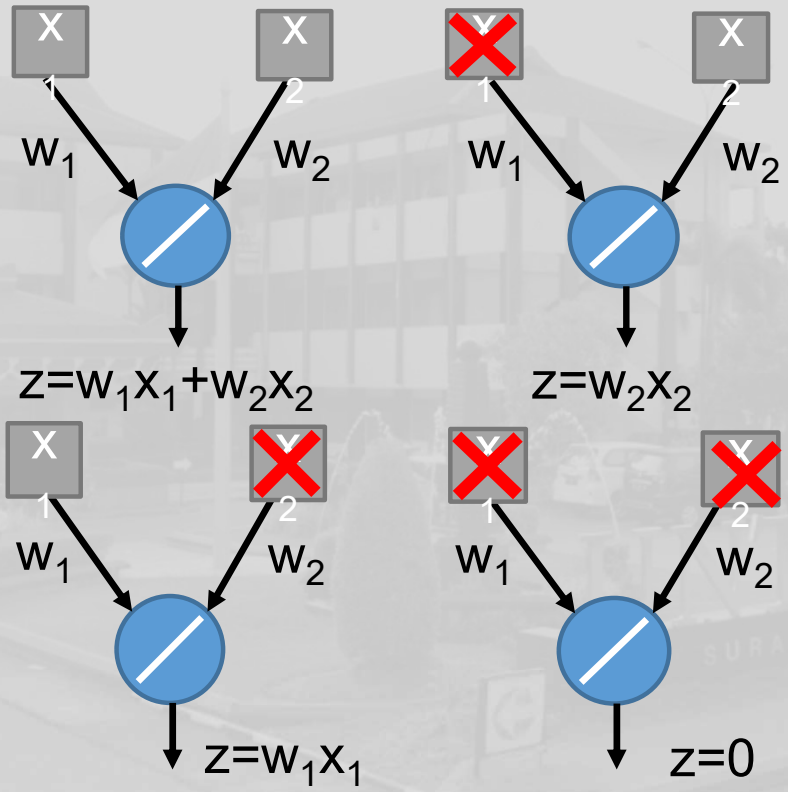
Dropout

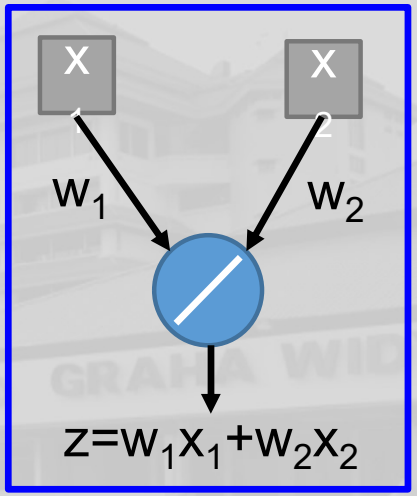
Testing:

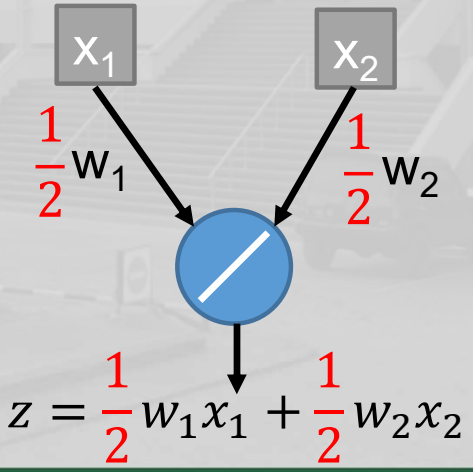


- **No dropout**
  - Jika tingkat dropout pada pelatihan adalah ***p***%, semua bobot dikalikan (1-***p***)%
  - Asumsikan bahwa tingkat dropout adalah 50%.
  - Jika bobot ***w***=1 dengan pelatihan, atur *w*=0,5 untuk pengujian.

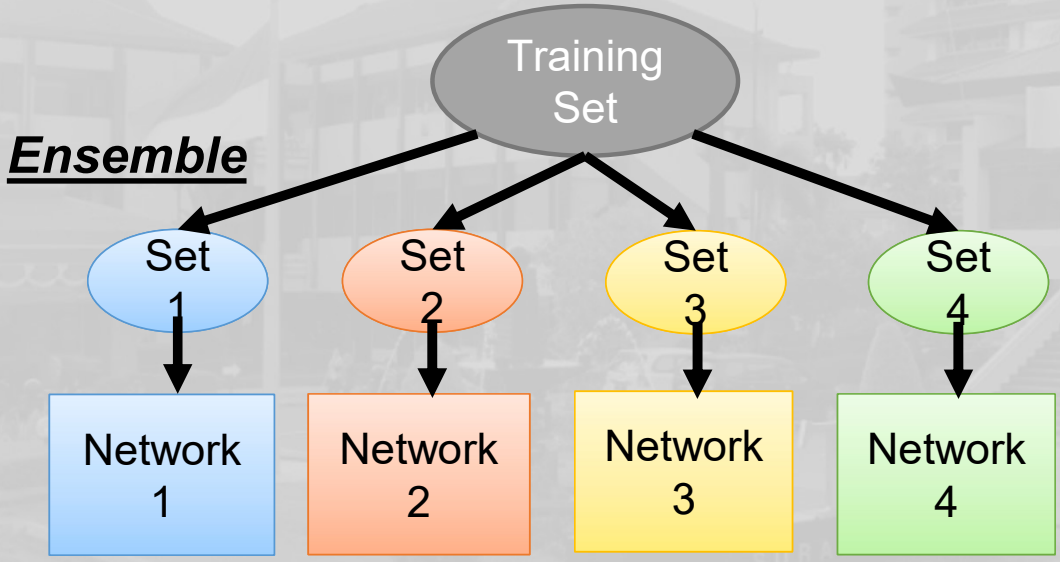
Mengapa bobot harus dikalikan (1-p)% (tingkat dropout) saat pengujian?







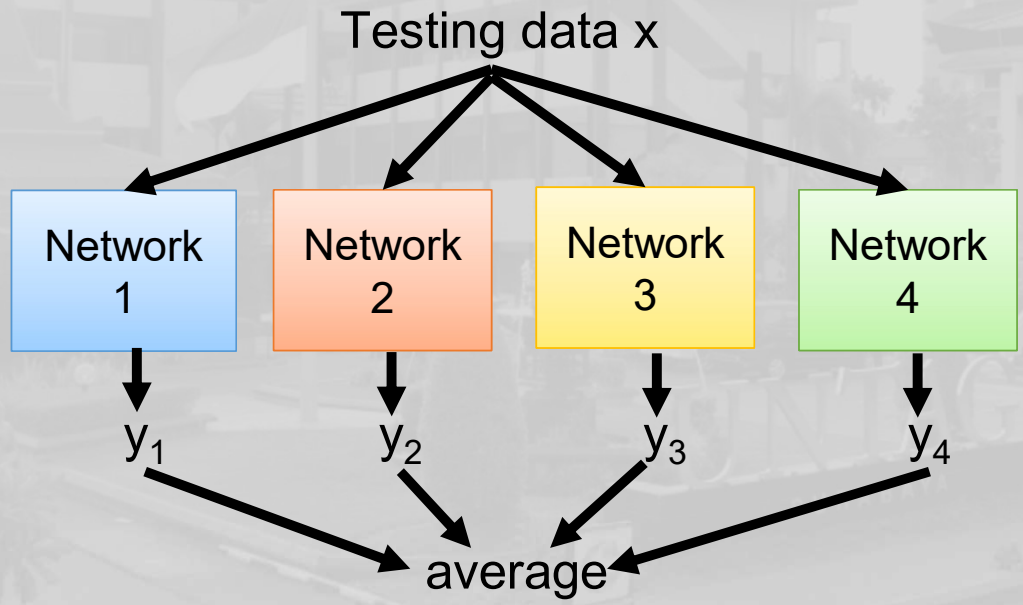
Dropout adalah sejenis ansambel.



Latih banyak jaringan dengan struktur berbeda

Dropout adalah sejenis ansambel.

Ensemble



References

- https://www.slideshare.net/AliceZheng3/evaluating-machine-learning-models-a-beginners-guide
- https://curaj.ac.in/sites/default/files/NITW_Improving%20Deep%20Neural%20Networks.pptx
- Goodfellow, I; Bengio,Y.; Courville, A (2016). Deep Learning. MIT Press pp: 224 - 270