PERBANDINGAN ALGORITMA KLASIFIKASI UNTUK HURUF TULISAN TANGAN

[1)]Intan Yuniar Purbasari, [2)]Fetty Tri Anggraeny
Teknik Informatika UPN "Veteran" Jawa Timur
[1)]intan.yuniar@gmail.com, [2)]fetty.ta@gmail.com

Abstrak. *Pengenalan tulisan tangan adalah kemampuan komputer untuk mengenali input tulisan tangan dari berbagai sumber seperti dokumen, gambar/foto, layar sentuh, serta alat input lainnya. Pengenalan tulisan tangan dapat dimanfaatkan antara lain untuk pembacaan alamat pada pengiriman paket atau pengenalan tulisan pada sebuah cek bank secara otomatis. Ada tiga metode klasifikasi dasar yang banyak digunakan untuk pengenalan; yakni pohon keputusan, neural network, dan Naïve Bayes. Penelitian ini membandingkan ketiga metode klasifikasi dasar tersebut dan kombinasinya dengan algoritma boosting varian awal AdaBoost.M1 untuk mengklasifikasikan 5605 sampel huruf tulisan tangan. Data input divariasikan dalam 4 komposisi training-test set dan output yang dibandingkan adalah tingkat akurasi dan runtime. Hasil yang didapat menunjukkan bahwa masing-masing algoritma yang diujicoba memiliki keunggulan masing-masing. Algoritma pohon keputusan yang dikombinasikan dengan AdaBoost dapat memberikan peningkatan akurasi hingga 15%, neural network multilayer tanpa boosting mampu memberikan tingkat akurasi yang cukup baik (rata-rata 79.3%), sedangkan Naïve-Bayes tanpa boosting memiliki runtime yang sangat baik (rata-rata 0.8225 detik) dibandingkan dengan kedua algoritma lainnya.*

Kata kunci: perbandingan algoritma, klasifikasi huruf tulisan tangan, neural network, pohon keputusan, Naïve Bayes, algoritma boosting.

Pengenalan tulisan tangan adalah kemampuan komputer untuk mengenali input tulisan tangan dari berbagai sumber seperti dokumen, gambar/foto, layar sentuh, serta alat input lainnya. Pengenalan tulisan tangan dapat dimanfaatkan antara lain untuk pembacaan alamat pada pengiriman paket atau pengenalan tulisan pada sebuah cek secara otomatis.

Terdapat dua bentuk pengenalan tulisan tangan, yakni secara *offline* dan *online*. Secara *offline*, komputer membaca tulisan dari selembar kertas dengan pemindaian optik (*optical scanning*). Citra yang terbaca lalu diubah ke dalam bentuk biner sehingga menghasilkan piksel citra yang bernilai 1 atau 0. Secara *online*, komputer membaca pergerakan dari alat input (dapat berupa *touch pen* atau *mouse*) yang digoreskan ke sebuah layar dalam bentuk serangkaian koordinat (x,y) dan hasil pengenalan langsung ditampilkan.

Topik pengenalan tulisan tangan secara *offline* telah diteliti sebelumnya dengan menggunakan beragam metode. Pada [**1**] telah dilakukan perbandingan algoritma klasifikasi Naive Bayes, *Instance Based Learner*, Pohon Keputusan, dan Neural Network untuk pengenalan digit tunggal secara *offline* terhadap dataset MNIST dan secara *online* terhadap sejumlah data penulisan digit dengan menggunakan sebuah pena *digitizer*. Akurasi yang didapat cukup tinggi pada algoritma *Instance Based Learner*.

Pada [**2**] digunakan modifikasi algoritma Neural Network Back Propagation untuk mengenali tulisan tangan dan dapat mempercepat proses konvergensi error dibandingkan dengan algoritma Back Propagation biasa.

Koerich dan Kalva dalam [**3**] menggunakan empat multilayer perceptron dan menerapkan tiga strategi klasifikasi yang berbeda pada database NIST SD19 dan menunjukkan peningkatan kinerja pengenalan pada classifier metaclass dimana huruf *uppercase* dan *lowercase* digabung menjadi satu class.

Adapun studi yang menggunakan algoritma *boosting* untuk meningkatkan akurasi algoritma klasifikasi juga telah dilakukan, antara lain dalam [**4**] yang melakukan boosting pada algoritma dasar *decision stump*, *product learner*, *Haar filter* dan pohon keputusan dengan algoritma AdaBoost.MH.

Ting dan Zheng dalam [**5**] telah melakukan studi penggunaan algoritma AdaBoost terhadap algoritma Naive Bayes. Walaupun pada hasil awal tidak menunjukkan peningkatan akurasi, mereka menambahkan unsur tree ke dalam Naive Bayes modifikasi mereka dan berhasil menurunkan error rata-rata secara signifikan.

Pada penelitian ini dilakukan perbandingan tiga algoritma klasifikasi dasar yang dalam penelitian sebelumnya juga telah digunakan baik tanpa maupun dengan modifikasi. Selain itu akan diujicoba pula ketiga algoritma tersebut dengan penambahan algoritma boosting terhadap dataset yang digunakan. Variabel output tiap algoritma yang akan dibandingkan adalah tingkat akurasi dan *run time*.

I. Metodologi

Sebelum data melalui proses klasifikasi, terlebih dahulu dilakukan proses *filter* terhadap seluruh atributnya untuk mengubah nilai numerik (0 dan 1) menjadi nominal (*false* dan *true*). Hal ini perlu dilakukan karena sifat salah satu algoritma klasifikasi (yakni ID3) yang hanya dapat dijalankan pada dataset dengan atribut nominal saja.

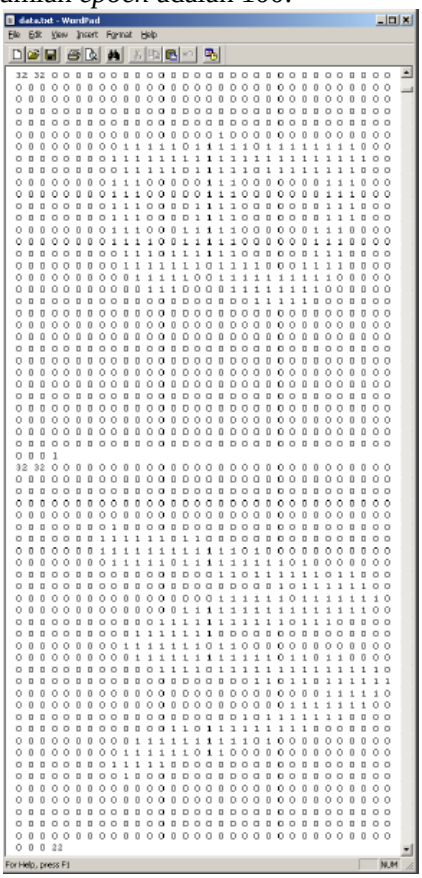
Setelah melalui proses filter, data diklasifikasi dengan menggunakan 6 teknik klasifikasi: kombinasi masing-masing 3 (tiga) algoritma klasifikasi klasik (ID3, Naïve-Bayes, Neural Network) dengan algoritma boosting AdaBoost (AB). Metodologi umum dari penelitian ini terdapat pada gambar 2.

Dataset

Dataset yang digunakan dalam penelitian ini terdiri dari 5605 sampel bersih huruf tulisan tangan (*uppercase* dan *lowercase*) dari huruf A sampai Z. Dataset tersimpan dalam file berformat .txt, masing-masing sampel huruf diwakili oleh matriks berukuran 32 x 32 yang bernilai numerik 0 dan 1 (Gambar 1). Jumlah total atribut adalah 1025, dimana 1024 adalah jumlah atribut piksel tiap data dan atribut ke-1025 adalah atribut class-nya. Dataset ini bertipe multiclass dengan jumlah data untuk tiap classnya adalah ~200 data.

Untuk setiap penerapan algoritma, dataset dibagi ke dalam komposisi *training set-test set* dalam 4 variasi, yakni 70-30, 80-20, 90-10, dan 100-100. Untuk variasi yang terakhir, semua *training set* menjadi *test set*.

Tiga algoritma klasifikasi klasik yang digunakan adalah Iterative Dichotomizer-3 (ID3), Naïve-Bayes (NB), dan Neural Network (NN). Untuk NN, network yang digunakan adalah Perceptron dengan *hidden layer* (*multilayer perceptron*). Nilai parameter *learning rate* adalah 0.3, jumlah *hidden layer* sebanyak 26 (sebanyak jumlah class output), serta jumlah *epoch* adalah 100.

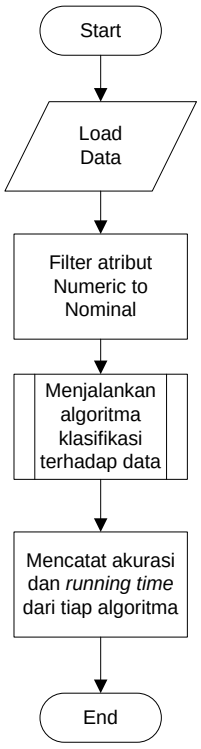


Gambar 1. Dataset

Untuk kombinasi tiap algoritma dengan algoritma boosting AdaBoost.M1, jumlah iterasi yang digunakan adalah 100 iterasi dengan nilai ambang bobot adalah 100.

Iterative Dichotomizer-3 (ID3)

Pohon keputusan adalah struktur tree dimana setiap internal node merepresentasikan fitur dan eksternal node (leaf) merepresentasikan konklusi class dari suatu data. Informai gain digunakan untuk menentukan fitur yang harus diletakkan di setiap internal node. Semakin tinggi posisi node fitur, maka fitur tersebut memiliki nilai informasi gain lebih tinggi. Semakin besar informasi gain, menunjukkan semakin besar suatu fitur memiliki peranan dalam menentukan keluaran, dalam hal ini konklusi class data.



Gambar 2. Metodologi

Gain ratio merupakan pengembangan dari informasi gain. Informasi gain digunakan untuk membentuk induksi pohon keputusan (ID3), sedangkan gain ratio digunakan pada C4.5, yang merupakan pengembangan dari ID3 [7]. Informasi gain menghasilkan bias, informasi gain lebih memilih fitur dengan banyak variasi nilai daripada fitur yang memilik sedikit variasi nilai meskipun lebih informatif [7]. Contoh, fitur unik pada suatu data seperti id siswa dalam tabel siswa di database. Pemisahan menggunakan id siswa menghasilkan sangat banyak partisi, karena setiap record data memiliki nilai unik yaitu id siswa [8].
Misal *S* adalah himpunan data sampel dan *m* adalah class. Maka entropi atau perkiraan informasi untuk mengklasifikasi sample:

$$\boldsymbol{I(S) = -\sum_{i=1}^{m} p_i log_2(p_i)} \quad (1)$$

dimana $\boldsymbol{p_i}$ adalah probabilitas sample dengan konklusi $\boldsymbol{class_i}$.
Misal fitur/atribut A memiliki variasi nilai sebanyak v. Misal $\boldsymbol{s_{ij}}$ adalah jumlah sampel class $\boldsymbol{C_i}$ dalam subset $\boldsymbol{S_j}$. $\boldsymbol{S_j}$ teridiri dari sampel dalam S yang memiliki nilai $\boldsymbol{a_j}$ dari A. Maka entropi berdasarkan pembagian menjadi subset atribut A:

$$\boldsymbol{E(A) = -\sum_{i=1}^{m} I(S) \frac{s1i+s2i+\cdots smi}{s}} \quad (2)$$

Informasi gain untuk mencabangkan atribut A adalah:

Gain(A)=I(S)-E(A) (3)

C4.5 menggunakan gain ratio dengan mengaplikasikan normalisasi terhadap informasi gain dengan nilai yang diperoleh dari:

$$\boldsymbol{SplitInfo(S) = -\sum_{i=1}^{v} (|S_i|/|S|) log_2 (|S_i|/|S|)} \quad (4)$$

Gain ratio dihitung menggunakan rumusan berikut:

GainRatio(A)=Gain(A)/SplitInfo(S) (5)

Atribut dengan nilai gain ratio tertinggi terpilih sebagai atribut pemisah (*splitting attribute*).

***Naïve-Bayes* (NB)**
Model statistik merupakan salah satu model yang terpercaya sangat andal sebagai pendukung pengambilan keputusan. Konsep probabilitas merupakan salah satu bentuk model statistik. Salah satu metode yang menggunakan konsep probabilitas adalah Naive Bayesian Classification (NBC). Pada metode ini, semua atribut akan memberikan kontribusinya dalam pengambilan keputusan, dengan bobot atribut yang sama penting dan setiap atribut saling bebas satu sama lain. Apabila diberikan k atribut yang saling bebas (independence), nilai probabilitas dapat diberikan sebagai berikut [9].

$$\boldsymbol{P(x_1, \ldots, x_k|C) = P(x_1|C) \times \ldots \times P(x_k|C)} \quad (6)$$

Jika atribut ke-*i* bersifat diskret, maka P(xi|C) diestimasi sebagai frekuensi relatif dari sampel yang memiliki nilai xi sebagai atribut ke i dalam kelas C. Namun, jika atribut ke-i bersifat kontinu, maka P(xi|C) diestimasi dengan fungsi densitas Gauss.

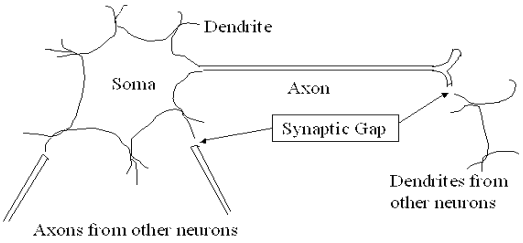
$$\boldsymbol{f(x) = \frac{1}{\sqrt{2\mu\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}} \quad (7)$$

Neural Network (NN)
Jaringan Syaraf Tiruan merupakan salah satu representasi buatan dari otak manusia yang

selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia. Istilah buatan digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran [10].

Otak manusia berisi berjuta-juta sel syaraf yang bertugas untuk memproses informasi. Setiap sel syaraf (neuron) akan memiliki satu inti sel, inti sel ini yang akan bertugas untuk melakukan pemrosesan informasi. Misal ketika kulit kita terkena panas, maka informasi yang diterima oleh sel-sel syaraf pada kulit akan diteruskan sampai ke otak untuk diproses, setelah itu dialirkan kembali ke sel-sel syaraf gerak sehingga menghasilkan gerak reflek. Begitu pula dengan algoritma jaringan syaraf tiruan, terdiri dari banyak neuron yang saling berhubungan untuk mengolah data masukan menjadi data keluaran.

Mengadopsi sel syaraf (neuron) pada manusia, dimana setiap neuron memiliki inti sel, bertugas sebagai pemrosesan informasi, dan sinapsis, yang bertugas sebagai penghubung antar neuron dan juga sebagai penerima masukan dan menghasilkan keluaran.



Gambar 3. Sel syaraf pada manusia

Sehingga komponen yang dimiliki jaringan syaraf tiruan sebagai berikut :

a. Neuron, sel syaraf yang akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju neuron-neuron yang lain. Di dalam setiap neuron terdapat fungsi aktivasi untuk menggantikan proses elektrokimiawi.
b. Bobot, pada jaringan syaraf tiruan, hubungan antar neuron-neuron dikenal dengan nama bobot yang menggantikan fungsi sinapsis.

Pada jaringan syaraf, neuron-neuron akan dikumpulkan dalam lapisan-lapisan (layer) yang disebut dengan lapisan neuron *(neuron layers).* Informasi yang diberikan pada jaringan syaraf akan dirambatkan lapisan ke lapisan, mulai dari input sampai kelapisan output melalui lapisan yang lainnya, yang dikenal dengan lapisan tersembunyi*(hidden layer).* Tergantung pada algoritma pembelajarannya, bisa jadi informasi tersebut akan dirambatkan secara mundur pada jaringan.

Metode jaringan syaraf tiruan yang digunakan adalah *backpropagation* yang merupakan algoritma pembelajaran yang terawasi dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyi (lihat Gambar 4). Metode pembelajaran pada jaringan syaraf disebut terawasi jika output yang diharapkan telah diketahui sebelumnya.

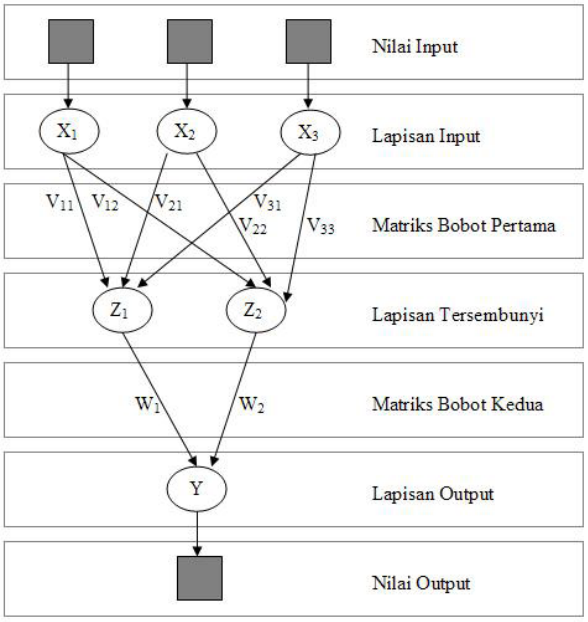
Jaringan syaraf tiruan dengan metode backpropagation terdiri dari 2 tahapan, yaitu tahap pelatian dan tahap pengujian.

Tahap Pembelajaran/Pelatihan

Tahap ini ditujukan untuk membuat jaringan syaraf tiruan menjadi "cerdas" dengan cara memasukkan beberapa data yang sudah terbukti kebenarannya. "cerdas" dalam hal ini adalah untuk mengkonfigurasi nilai bobot agar jaringan dapat menghasilkan keluaran yang sesuai/mendekati dengan target. Fungsi aktivasi yang digunakan adalah fungsi logsig. Serangkain proses dalam tahap pelatihan adalah seperti gambar 4.

Langkah 1 Inisialisasi nilai bobot dan bias dapat *diset* dengan sembarang angka (acak) Inisialisasi learning rate, maksimal iterasi dan toleransi *error*.

Langkah 2 Lakukan selama kondisi berhenti masih belum terpenuhi.



Gambar 4. Arsitektur jaringan lapis banyak

Langkah 3 Setiap unit *input* (X_i, i = 1,...,n) menerima sinyal *input* dan menyebarkannya pada seluruh *hidden* unit.

Langkah 4 Setiap hidden unit (Z_j, j = 1,...,p) akan menghitung sinyal-sinyal *input* dengan bobot dan biasnya.

$$z\_in_j = v0_j + \sum_{i=1}^{n} x_i v_{ij} \quad (8)$$

Kemudian dengan menggunakan fungsi aktivasi yang telah ditentukan diperoleh sinyal *output* dari *hidden* unit tersebut.

$$z_j = f(z\_in_j) \quad (9)$$

Langkah 5 Setiap unit *output* (Y_k, k = 1,...,m) akan menghitung sinyal-sinyal dari *hidden* unit dengan bobot dan biasnya.

$$y\_in_k = w0k + \sum_{j=1}^{p} Z_j W_{jk} \quad (10)$$

Kemudian dengan menggunakan fungsi aktivasi yang telah ditentukan diperoleh sinyal *output* dari unit *output* tersebut.

$$y_k = f(y\_in_k) \quad (11)$$

Langkah 6 penghitungan faktor koreksi *error* (δ_k).

$$\delta_k = (t_k - y_k) f'(y\_in_k) \quad (12)$$

Faktor koreksi *error* digunakan untuk menghitung koreksi *error* (ΛW_{jk}) untuk memperbaharui W_{jk}.

$$\Delta W_{jk} = \alpha \delta_k z_j \quad (13)$$
$$\Delta W_{0k} = \alpha \delta_k$$

Langkah 7 Setiap hidden unit (Z_j, j = 1,...,p) akan menghitung bobot yang dikirimkan *output* unit. Jika kondisi iterasi pertama yang rumus yang digunakan

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (14)$$

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi untuk mendapatkan faktor koreksi *error*

$$\delta_j = \delta\_in_j f'(z\_in_j) \quad (15)$$
$$\Delta V_{ij} = \alpha \delta_j x_i$$
$$V_{0j} = \alpha \delta_j$$

Langkah 8 Setiap unit *output* (Y_k, k = 1,...,m) akan memperbaharui bobotnya dari setiap hidden unit.

$$W_{jk}(baru) = W_{jk}(lama) + \Delta W_{jk} \quad (16)$$

Demikian pula setiap hidden unit (Z_j, j = 1,...,p) akan memperbaharui bobotnya dari setiap unit *input*.

$$V_{ij}(baru) = V_{ij}(lama) + \Delta V_{ij} \quad (17)$$

Langkah 9 Memeriksa kondisi berhenti

Tahap Pengujian

Tahap ini dilakukan untuk menguji apakah konfigurasi jaringan syaraf tiruan sudah dapat menghasilkan keluaran sesuai target. Proses dalam tahap pengujian merupakan langkah *feedforward* pada tahap pembelajaran, yaitu langkah 1 (satu) sampai langkah 5 (lima).

ADABOOST

AdaBoost merupakan sebuah algoritma meta yang diajukan oleh Freund dan Schapire **[** 6 **]** yang dapat melakukan *boosting* terhadap algoritma klasifikasi yang lemah dan dapat melakukan adaptasi terhadap error hipotesis.

Algoritma ini menerima input berupa *training set* (x_1, y_1), …, (x_m, y_m) dimana setiap x_i adalah input data dan setiap y_i adalah label/class dari input x_i. AdaBoost menjalankan sebuah algoritma klasifikasi lemah secara berulang

sebanyak *t* kali. Ide dasar dari algoritma ini adalah pencatatan bobot *D* } sampel pada *training set*. Pada awalnya, tiap sampel memiliki bobot yang sama. Namun pada setiap iterasi, bobot dari sampel yang salah diklasifikasi akan dinaikkan (dan sebaliknya, bobot dari sampel yang benar diklasifikasi akan diturunkan) agar pada iterasi berikutnya algoritma akan lebih memperhatikan sampel yang salah klasifikasi.

Sejalan dengan perkembangannya, algoritma AdaBoost memiliki banyak varian. Gambar 5 menunjukkan versi awal dari algoritma AdaBoost.

Input: $(x_1, y_1) \ldots, (x_m, y_m)$ dimana $x_i \in X, y_i \in Y = \{-1, +1\}$

Inisialisasi: $D_1(i) = 1/m$

For $t = 1, \ldots, T$:

Latih *classifier* lemah menggunakan distribusi D_t

Cari hipotesis lemah $h_t : X \rightarrow \{-1, +1\}$ dengan error $\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$

Pilih $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$

Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} x \begin{cases} e^{-\alpha_t} \; jika \; h_t(x_i) = y_i \\ e^{\alpha_t} \; jika \; h_t(x_i) \neq y_i \end{cases}$$

dimana Z_t adalah faktor normalisasi (yang dipilih sedemikian hingga D_{t+1} merupakan distribusi).

Output hipotesis finalnya:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Gambar 5. Algoritma Boosting AdaBoost

II. Hasil dan Pembahasan

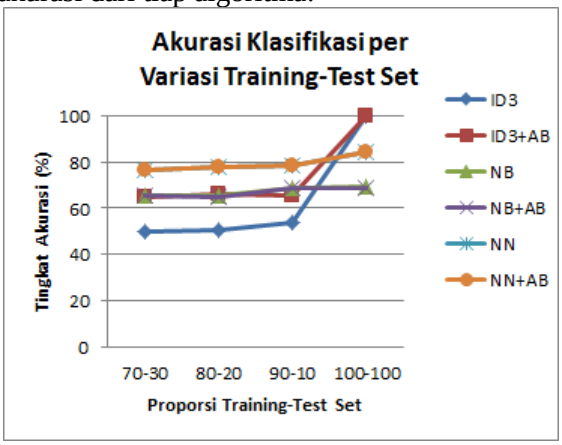
Dari hasil percobaan yang telah dilakukan, output hasil klasifikasi setiap algoritma dicatat. Dua parameter output yang dicatat adalah akurasi klasifikasi dan *running time*. Tabel 1 menunjukkan tingkat akurasi hasil klasifikasi untuk setiap skenario ujicoba yang telah dilakukan, sedangkan tabel 2 menunjukkan *runtime* dari tiap ujicoba.

Dari tabel 1 dapat diketahui bahwa akurasi tertinggi untuk semua algoritma dicapai saat variasi training-test set adalah 100-100. Hal ini tergolong sangat wajar karena seluruh data yang digunakan dalam proses pelatihan juga ikut diujikan. Sedangkan akurasi terendah dicapai oleh algoritma ID3 tanpa boosting pada variasi training-test set 70-30 yakni sebesar 49.76%.

Tabel 1. Akurasi Hasil Klasifikasi

Algoritma	Akurasi dalam % (untuk training-test set)			
	70-30	80-20	90-10	100-100
ID3	49.76	50.31	53.65	100
NB	65.78	65.55	68.95	69.37
NN	76.40	77.88	78.43	84.48
ID3 + AB	64.57	66.10	65.77	100
NB + AB	65.28	64.85	68.45	68.96
NN + AB	76.40	77.88	78.43	84.48

Gambar 6 menunjukkan perbandingan akurasi dari tiap algoritma:



Gambar 6. Grafik Akurasi Klasifikasi per Variasi Training-Test Set

Dari tabel 1 dan gambar 6 dapat diketahui bahwa pada algoritma Neural Network, tidak ada peningkatan akurasi ketika dikombinasikan dengan AdaBoost. Pada algoritma Naïve-Bayes, pengkombinasian dengan AdaBoost justru menurunkan tingkat akurasi antara 0.4 sampai 0.7%. Algoritma yang paling baik ketika dikombinasikan dengan AdaBoost adalah ID3, yang mengalami peningkatan akurasi antara 12 hingga 15%.

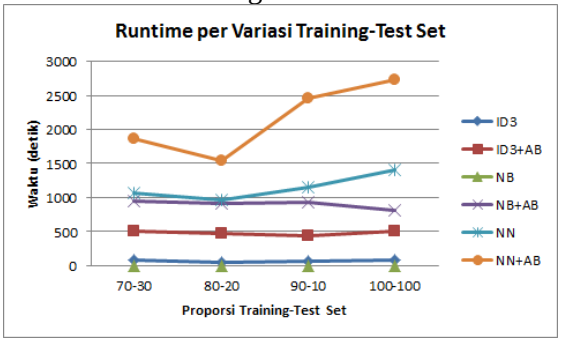
Tabel 2 menunjukkan *runtime* tiap algoritma beserta kombinasinya dengan AdaBoost untuk tiap variasi training-test set.

Tabel 2. Runtime Tiap Algoritma

Algoritma	*Runtime* Dalam detik (untuk training-test set)			
	70-30	80-20	90-10	100-100
ID3	74.13	46.55	56.84	74.16
NB	1.01	0.67	0.94	0.67
NN	1065.42	971.89	1153.44	1410.79
ID3 + AB	514.44	473.69	434.01	509.71
NB + AB	946.04	905.48	925.84	810.29
NN + AB	1862.14	1537.38	2452.12	2721.62

Gambar 7 menampilkan perbandingan runtime dari setiap algoritma.

Dari tabel 2 dan gambar 7 dapat diketahui bahwa *runtime* algoritma Naïve-Bayes memiliki perbedaan yang sangat signifikan dengan kedua algoritma lainnya, khususnya pada skenario non-boosting. Sedangkan pada skenario boosting, algoritma ID3 memiliki *runtime* yang tercepat di antara ketiganya pada semua variasi training-test set.



Gambar 7. *Runtime* per Variasi Training-Test Set

Terkait dengan tingkat akurasinya, algoritma Neural Network dengan skenario boosting walaupun memiliki kenaikan *runtime* dari skenario non-boosting yang hampir dua kali lipat, tidak memberikan peningkatan akurasi sama sekali. Begitu pula yang terjadi pada algoritma Naïve-Bayes. Meskipun mengalami kenaikan *runtime* sebesar lebih dari 800 kali lipat, tingkat akurasi justru mengalami penurunan.

Berdasarkan hasil ujicoba yang telah dilakukan, dapat disimpulkan bahwa dari ketiga algoritma klasifikasi yang diuji, algoritma Naïve-Bayes merupakan algoritma yang paling cepat dalam memberikan hasil klasifikasi, dengan rata-rata waktu runtime 0.8225 detik, namun memiliki tingkat akurasi yang sedang dan relatif stabil pada variasi jumlah training-test set. Algoritma ini juga kurang sesuai jika dikombinasikan dengan algoritma boosting AdaBoost.M1 yang justru menyebabkan penurunan tingkat akurasi.

Kombinasi algoritma AdaBoost.M1 dengan Neural Network Multilayer juga tidak memberikan kontribusi kenaikan tingkat akurasi sama sekali, walaupun memiliki *runtime* yang sangat besar.

Algoritma ID3 merupakan algoritma yang paling baik (dari ketiga algoritma klasifikasi ujicoba) ketika dipasangkan dengan AdaBoost.M1 yang memberikan peningkatan tingkat akurasi hingga 15%. Patut dicatat pula prestasi sempurna dari algoritma ini pada variasi training-test set 100-100 yang tidak dicapai oleh kedua algoritma yang lain. Dalam hal ini, algoritma ID3 sanggup menyesuaikan (belajar) dengan data input sehingga mencapai akurasi yang sempurna, sesuai data latih. Namun, dari sisi lain, hal ini juga berarti algoritma ID3 bersifat *overfitting*.

III. Simpulan

Secara keseluruhan, untuk mendapatkan tingkat akurasi yang baik, alternatif algoritma yang dapat digunakan adalah Neural Network Multilayer non-boosting yang memiliki rata-rata akurasi 79.3%. Sedangkan untuk mendapatkan *runtime* yang lebih singkat dengan tingkat akurasi yang masih cukup baik, alternatifnya adalah Naïve-Bayes non-boosting. Penggunaan algoritma boosting dengan algoritma ID3 masih merupakan alternatif yang juga patut dilihat, dengan melakukan beberapa kustomisasi baik terhadap data maupun algoritma itu sendiri. Mengingat kustomisasi algoritma ID3 yang sangat terbatas, maka pilihan yang lebih baik adalah melakukan kustomisasi pada data, misalnya dengan melakukan preproses seleksi atribut terhadap dataset.

Penelitian lanjutan yang dapat dilakukan adalah dengan mengikuti langkah yang telah dilakukan oleh [5] yang memodifikasi Naïve Bayes dengan menambahkan tree yang terbukti dapat memberikan peningkatan akurasi jika digunakan dengan algoritma *boosting*.

IV. Daftar Pustaka

[1] Elijah Olusayo Omidiora, Ibrahim Adepoju Adeyanju, and Olusayo Deborah Fenwa, "Comparison of Machine Learning Classifiers for Recognition of Online and Offline Handwritten Digits," *Computer Engineering and Intelligent Systems*, vol. 4, no. 13, pp. 39-47, 2013.

[2] Ashok Kumar and Pradeep Kumar Bhatia, "Offline Handwritten Character Recognition Using Improved Back-Propagation Algorithm," *International Journal of Advances in Engineering Sciences*, vol. III, no. 3, pp. 1-6, July 2013.

[3] Alessandro L. Koerich and Kalva P.R., "Unconstrained Handwritten Character Recognition Using Different Classification Strategies," in *IEEE International*

Conference on Image Processing (ICIP), Genova, 2005, pp. II - 542-545.

[4] Balazs Kegl and Robert Busa-Fekete, "Boosting Products of Base Classifiers," in *26th International Conference on Machine Learning (ICML)*, New York, 2009, pp. 497-504.

[5] Kai Ming Ting and Zijian Zheng, "A Study of AdaBoost with Naive Bayesian Classifiers: Weakness and Improvement," *Computational Intelligence*, vol. 19, no. 2, pp. 186-200, May 2003.

[6] Yoav Freund and Robert E. Schapire, "A Decision-theoretic Generalization of On-line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, August 1997.

[7] Asha Gowda Karegowda, A. S. Manjunath, M.A.Jayaram. Comparative Study of Attribute Selection Using Gain Ratio Correlation based feature selection. International Journal of Information Technology and Knowledge Management, July-December 2010, Volume 2, No. 2, pp. 271-277.

[8] Vege Sri Harsha. Ensemble of Feature Selection Techniques for High Dimensional Data. Masters Theses. Western Kentucky University. 2012.

[9] Sri Kusumadewi, Klasifikasi Status Gizi Menggunakan Naive Bayesian Classification, CommIT, Vol. 3 No. 1 Mei 2009, hlm. 6 – 11.

[10] Arief Hermawan, "Jaringan Saraf Tiruan Teori dan Aplikasi", Penerbit ANDI Yogyakarta, 2006.