

Implementation of Quantum Kernel Estimation for Supervised Classification on a Two-Qubit Quantum Simulator

Rizqi Aditya ^{1*}, Erwin Yudi Hidayat ^{2*}

* Teknik Informatika, Universitas Dian Nuswantoro

111202214411@mhs.dinus.ac.id¹, erwin@dsn.dinus.ac.id²

Article Info

Article history:

Received 2026-09-08

Revised 2026-09-25

Accepted 2026-10-18

Keyword:

Feature Map,
Quantum Kernel Estimation,
Quantum Machine Learning,
Qiskit,
Support Vector Machine.

ABSTRACT

Quantum kernel estimation (QKE) lets a quantum computer supply the similarity values a classical support vector machine needs, but the original hardware demonstration is hard to reproduce and gives little guidance on the number of shots a usable kernel requires. This study implements QKE on a two-qubit simulator and quantifies its shot and noise requirements. A second-order ZZ feature map encodes each data point, kernel entries are measured from an overlap circuit, and a classical support vector machine is trained on the resulting matrix. An artificial dataset of 40 training and 40 test points was generated with a separation gap of 0.3, the shot count was varied from 128 to 50,000, and a depolarising and readout noise model was applied; classical kernels and a variational classifier served as baselines. With an exactly computed kernel the classifier labels every training and test point correctly using 12 support vectors. Perfect test accuracy is reached at 1,024 shots per pair, far below the 50,000 used on hardware, and the deviation from the exact kernel decays with a fitted exponent of -0.52. The weight of negative eigenvalues falls from 2.019 to 0.096 as shots increase, making it a practical warning sign of too few shots. Under noise the kernel values drop by 0.014 on average, yet accuracy stays at 100% because the drop is nearly uniform. Classical kernels reach at most 70% test accuracy and the variational classifier 98.3%. The work contributes a reproducible open-source procedure and shot-budget guidance for quantum kernel experiments.



This is an open access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

I. PENDAHULUAN

Komputer kuantum adalah komputer yang bekerja memakai sifat partikel sangat kecil, seperti superposisi dan keterjeratan. Sifat ini membuatnya bisa mengolah banyak kemungkinan sekaligus. Beberapa tahun terakhir, banyak peneliti mencoba memakai komputer kuantum untuk pembelajaran mesin, yaitu untuk tugas mengelompokkan dan mengenali pola pada data [1]. Namun komputer kuantum yang ada sekarang masih kecil dan mudah terganggu derau, sehingga hanya program pendek yang bisa dijalankan dengan baik [2]. Karena itu, bentuk program yang paling banyak dipakai adalah gabungan rangkaian kuantum pendek dengan pengoptimal biasa di komputer klasik [3].

Di sisi komputer biasa, support vector machine (SVM) adalah salah satu metode klasifikasi yang paling sering dipakai. SVM mencari garis atau bidang pemisah antara dua

kelompok data. Kelebihannya, SVM tidak perlu mengetahui posisi data di ruang berdimensi tinggi; ia hanya perlu tahu nilai kemiripan antar pasangan data yang disebut kernel [4]. Sifat inilah yang membuka peluang bagi komputer kuantum: komputer kuantum cukup dipakai untuk menghitung satu angka kemiripan untuk setiap pasangan data. Usulan awal memang menjanjikan, tetapi mensyaratkan data sudah berbentuk keadaan kuantum sehingga sulit dipakai untuk data biasa [5]. Usulan yang lebih masuk akal adalah menjadikan ruang keadaan kuantum sebagai tempat menaruh data, lalu mengukur kemiripannya secara langsung [6].

Havlicek dan rekan-rekannya menyusun dua cara kerja yang bisa langsung dipakai pada data biasa dan sudah diuji pada komputer kuantum dua qubit milik IBM [7]. Cara pertama disebut quantum variational classifier (VQC), yaitu rangkaian kuantum berparameter yang dilatih seperti jaringan saraf [8] memakai susunan lapisan yang hemat gerbang [9].

Cara kedua disebut quantum kernel estimation (QKE), yaitu komputer kuantum hanya dipakai untuk mengisi tabel kemiripan antar data, sedangkan pencarian bidang pemisah tetap dikerjakan SVM biasa. Pada percobaan mereka dipakai teknik penekan derau [10] dan pengoptimal SPSA [11], dengan hasil klasifikasi benar 100% untuk dua kumpulan data dan 94,75% untuk satu kumpulan lainnya. Kajian sesudahnya bahkan menunjukkan bahwa hampir semua model pembelajaran kuantum berparameter sebenarnya dapat dipandang sebagai metode kernel [12].

Penelitian lanjutan berkembang ke berbagai arah. Ada yang menegaskan bahwa keunggulan model kuantum harus dibandingkan secara adil dengan model klasik yang juga punya data latih [13]. Ada yang membuktikan percepatan yang benar-benar terbukti secara matematis untuk satu masalah khusus [14], ada yang memperluas bentuk kernel agar cocok untuk data berstruktur khusus [15], dan ada yang melatih parameter peta fitur agar kernel menyesuaikan diri dengan data [16]. Sebaliknya, ditemukan pula kelemahan: pada jumlah qubit yang besar, nilai kernel cenderung menumpuk di angka yang hampir sama sehingga tidak lagi bisa membedakan data [17], dan percobaan pada data berdimensi tinggi menunjukkan derau perangkat keras masih menjadi hambatan utama [18].

Persoalannya, hasil percobaan tersebut diperoleh pada perangkat keras khusus dengan prosedur penekan derau tertentu, sehingga sulit diulang oleh peneliti lain. Padahal saat ini tersedia perangkat lunak gratis untuk menulis dan mensimulasikan rangkaian kuantum [19] serta pustaka pembelajaran mesin yang dapat menerima tabel kernel siap pakai [20]. Dengan dua perangkat lunak itu, metode QKE dapat dijalankan di komputer biasa, dan jumlah pengulangan pengukuran maupun tingkat derau dapat diatur sesuka kita.

Penelitian ini menerapkan metode QKE pada dua qubit di simulator, lalu memeriksa tiga hal: seberapa tepat hasil klasifikasinya, seberapa banyak pengulangan pengukuran yang dibutuhkan agar tabel kernel cukup akurat, dan seberapa tahan metode ini terhadap derau. Sebagai pembanding dipakai kernel klasik dan metode variasional. Hasilnya diharapkan menjadi panduan praktis bagi siapa pun yang ingin mencoba kernel kuantum tanpa perangkat keras kuantum.

II. METODE

Penelitian ini dikerjakan secara berurutan. Mula-mula disiapkan peta fitur kuantum, yaitu cara memindahkan data biasa menjadi keadaan kuantum. Selanjutnya dibuat data uji coba beserta labelnya, lalu dihitung tabel kemiripan atau matriks kernel di simulator. Tabel tersebut kemudian dipakai untuk melatih SVM. Setelah itu diuji pengaruh jumlah pengulangan pengukuran dan pengaruh derau, dan terakhir hasilnya dibandingkan dengan kernel klasik serta dengan metode variasional. Urutan yang sama dipakai kembali pada bab hasil agar mudah diikuti.

A. Peta Fitur Kuantum

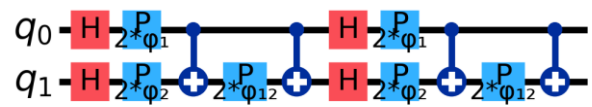
Peta fitur adalah aturan untuk mengubah sepasang angka menjadi sebuah keadaan kuantum. Aturan yang dipakai sama dengan aturan pada penelitian Havlicek dkk. [7], yaitu:

$$|\Phi(x)\rangle = U_{\phi(x)} H^{\otimes n} U_{\phi(x)} H^{\otimes n} \quad (1)$$

Bagian H adalah gerbang Hadamard yang membuat qubit berada dalam keadaan campuran, sedangkan bagian $U_{\phi(x)}$ adalah gerbang putaran fase yang menyisipkan nilai data ke dalam keadaan kuantum:

$$U_{\phi(x)} = \exp(i \sum_{S \subseteq [n]} \phi_S(x) \prod_{i \in S} Z_i) \quad (2)$$

Dipakai dua qubit, sehingga satu titik data berisi dua angka, yaitu x_1 dan x_2 . Sudut putaran untuk qubit pertama adalah x_1 , untuk qubit kedua adalah x_2 , dan sudut putaran gabungan kedua qubit adalah $(\pi - x_1)(\pi - x_2)$. Sudut gabungan inilah yang membuat hubungan antara data dan keadaan kuantum menjadi tidak linear, sehingga pola yang rumit tetap bisa dipisahkan. Bentuk rangkaianannya ditampilkan pada Gambar 1 dan hanya memerlukan 4 gerbang Hadamard, 6 gerbang fase, serta 4 gerbang CNOT.



Gambar 1. Rangkaian peta fitur untuk dua qubit; $\phi_1 = x_1$, $\phi_2 = x_2$, dan $\phi_{12} = (\pi - x_1)(\pi - x_2)$.

Rangkaian pada gambar tersebut dibaca dari kiri ke kanan. Blok yang sama diulang dua kali, dan pengulangan inilah yang membuat nilai kemiripan antar data sulit dihitung oleh komputer biasa. Jika blok hanya dipakai satu kali, nilai kemiripannya masih dapat ditiru dengan perhitungan biasa sehingga tidak ada gunanya memakai komputer kuantum.

Nilai kemiripan antara dua titik data dihitung sebagai kuadrat perkalian dalam antara dua keadaan kuantum:

$$K(x, z) = |\langle \Phi(x) | \Phi(z) \rangle|^2 \quad (3)$$

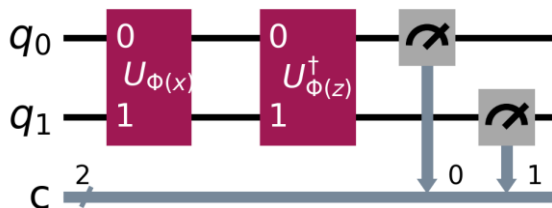
Nilainya selalu berada di antara 0 dan 1. Nilai 1 berarti dua data dipetakan ke keadaan yang persis sama, sedangkan nilai 0 berarti keduanya sama sekali berbeda.

B. Pembuatan Data Uji Coba

Data dibuat sendiri agar hasilnya bisa dibandingkan dengan penelitian Havlicek dkk. [7]. Caranya, diambil sebuah pengukuran acak pada dua qubit, lalu setiap titik data diberi label +1 bila nilai pengukurannya cukup besar dan label -1 bila cukup kecil. Titik yang nilainya berada di tengah dibuang agar dua kelompok data terpisah jelas dengan jarak aman sebesar 0,3. Titik diambil acak dari daerah $(0, 2\pi]$ untuk kedua sumbu sampai terkumpul 20 titik per label untuk data latih dan 20 titik per label untuk data uji. Jadi ada 40 data latih dan 40 data uji.

C. Penghitungan Matriks Kernel di Simulator

Untuk mengukur kemiripan dua data, rangkaian peta fitur data pertama dijalankan lebih dahulu, lalu disambung dengan kebalikan rangkaian peta fitur data kedua. Kalau kedua data sama, semua qubit akan kembali ke keadaan nol. Karena itu kemiripan diukur dari seberapa sering hasil pengukuran berupa '00'. Rangkaannya ditampilkan pada Gambar 2 dan setelah diterjemahkan ke gerbang dasar hanya memiliki kedalaman 14 dengan 6 gerbang CNOT.



Gambar 2. Rangkaian untuk mengukur kemiripan dua data.

Semakin sering hasil '00' muncul, semakin mirip kedua data tersebut. Satu kali pengukuran belum cukup karena hasilnya acak, sehingga pengukuran diulang sebanyak R kali; jumlah pengulangan ini disebut shots.

Karena tabel kemiripan bersifat simetris dan bagian diagonalnya pasti bernilai satu, cukup dihitung 780 pasangan data untuk 40 data latih. Nilai R yang dicoba adalah 128, 512, 1.024, 4.096, 8.192, dan 50.000. Angka terakhir sama dengan yang dipakai Havlicek dkk. [7] pada perangkat keras.

Selain itu dibuat pula percobaan dengan derau buatan agar mendekati keadaan komputer kuantum sungguhan. Derau yang dipasang berupa gangguan 0,1% pada gerbang satu qubit, 2% pada gerbang CNOT, dan kesalahan pembacaan 4% pada tiap qubit. Angka terakhir dipilih agar ketepatan pembacaan sekitar 95%, mirip dengan perangkat keras yang dipakai Havlicek dkk. [7]. Tidak ada teknik penekan derau berbasis ekstrapolasi derau-nol [10] yang dipasang.

D. Pelatihan SVM

Tabel kemiripan yang diperoleh pada langkah sebelumnya diserahkan kepada SVM biasa yang menerima kernel siap pakai. Nilai parameter penalti dibuat besar, yaitu 1000, karena data memang terpisah sempurna. SVM akan memilih beberapa titik data yang paling menentukan letak bidang pemisah; titik-titik ini disebut support vector. Untuk menebak label data baru, SVM menjumlahkan kemiripan data baru terhadap seluruh support vector, lalu melihat tanda hasil penjumlahannya.

E. Pengujian Jumlah Shots dan Derau

Pada pengujian ini dicatat empat hal untuk setiap nilai R. Pertama, akurasi latih dan akurasi uji, yaitu persentase data yang labelnya ditebak dengan benar. Kedua, selisih terbesar antara tabel kernel hasil pengukuran dan tabel kernel yang dihitung tepat. Ketiga, ukuran selisih rata-rata seluruh tabel.

Keempat, bobot nilai eigen negatif, yaitu penanda seberapa jauh tabel kernel menyimpang dari bentuk yang seharusnya. Tabel kernel yang benar tidak boleh punya nilai eigen negatif, sehingga munculnya angka ini menandakan hasil pengukuran belum cukup teliti.

F. Perbandingan dengan Metode Lain

Sebagai pembanding pertama, SVM dilatih langsung pada data mentah memakai kernel klasik, yaitu linear, polinomial derajat dua, polinomial derajat tiga, dan RBF. Sebagai pembanding kedua, metode VQC dari penelitian yang sama ikut dijalankan. Pada VQC, keadaan hasil peta fitur diolah lagi oleh rangkaian berparameter yang berisi putaran Y dan Z serta gerbang CZ, lalu parameternya dilatih dengan algoritma SPSSA sebanyak 250 langkah. Kedalaman rangkaian dicoba dari 0 sampai 4, dan tiap kedalaman dilatih tiga kali dengan tebakan awal berbeda agar hasilnya tidak kebetulan. Ringkasan seluruh pengaturan percobaan ada pada Tabel I.

TABEL I
PENGATURAN PERCOBAAN

Pengaturan	Nilai
Jumlah qubit	2
Lapisan peta fitur	2
Jarak aman antar kelas	0,3
Jumlah data latih / uji	40 / 40
Jumlah shots R	128 - 50.000
Pasangan kernel dihitung	780
Penalti SVM	1000
Derau	0,1% / 2% / baca 4%
Langkah SPSSA	250
Perangkat lunak	Qiskit 2.5, Aer 0.17, scikit-learn 1.8

Seluruh percobaan pada tabel tersebut memakai benih acak tetap. Artinya, bila percobaan dijalankan ulang dengan pengaturan yang sama, angka yang keluar akan sama persis sehingga hasilnya dapat diperiksa oleh orang lain.

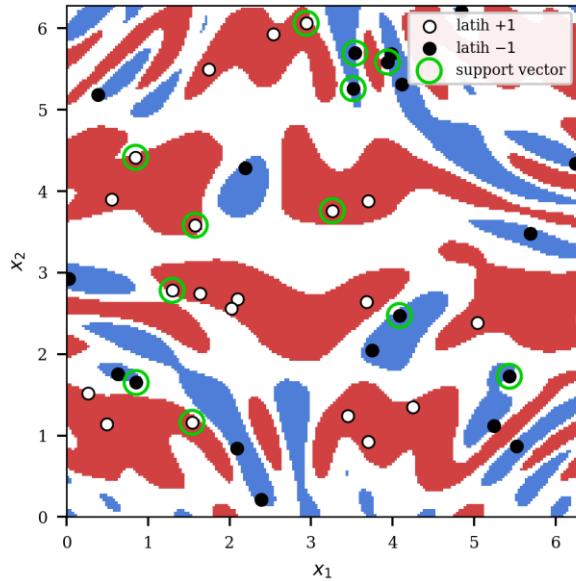
III. HASIL DAN PEMBAHASAN

Bagian ini disusun mengikuti urutan pada bab sebelumnya, mulai dari hasil pembuatan data sampai hasil perbandingan dengan metode lain.

A. Peta Fitur dan Data

Gambar 3 memperlihatkan bentuk pemisahan yang dihasilkan aturan pelabelan, beserta 40 titik data latih.

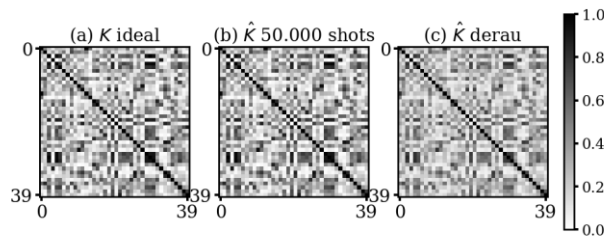
Pada gambar tersebut, warna merah menandai daerah berlabel +1, warna biru menandai daerah berlabel -1, dan warna putih adalah jarak aman yang sengaja dikosongkan. Lingkaran hijau menandai support vector. Bentuk daerahnya berpilin dan tidak dapat dipisahkan oleh sebuah garis lurus, sehingga data ini memang sulit bagi metode klasifikasi biasa.



Gambar 3. Daerah label pada seluruh daerah data, titik data latih, dan support vector.

B. Matriks Kernel

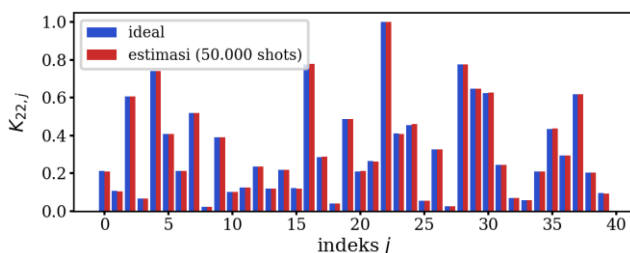
Gambar 4 membandingkan tabel kernel yang dihitung tepat, tabel hasil pengukuran 50.000 shots, dan tabel hasil pengukuran dengan derau.



Gambar 4. Matriks kernel data latih: (a) hitungan tepat, (b) hasil 50.000 shots, (c) hasil dengan derau pada 8.192 shots.

Pada ketiga tabel warna itu, kotak yang makin gelap berarti dua data makin mirip. Garis diagonal yang gelap muncul karena setiap data selalu mirip sempurna dengan dirinya sendiri. Ketiga gambar hampir tidak dapat dibedakan, artinya hasil pengukuran di simulator sudah sangat dekat dengan hitungan yang tepat.

Agar selisihnya terlihat jelas, Gambar 5 menampilkan satu baris tabel kernel yang selisihnya paling besar.



Gambar 5. Perbandingan satu baris matriks kernel dengan selisih terbesar.

Batang biru pada gambar tersebut adalah nilai yang benar dan batang merah adalah hasil pengukuran. Selisih keduanya kecil dan tersebar merata, bukan menumpuk pada nilai tertentu. Hal ini wajar karena penyebab selisihnya hanya keacakan pengukuran, bukan kesalahan sistematis.

C. Hasil Pelatihan SVM

Dengan tabel kernel yang dihitung tepat, SVM menebak benar seluruh data latih dan seluruh data uji, atau akurasi 100%. SVM memilih 12 support vector dari 40 data latih dengan nilai bias $-0,053$. Sepuluh support vector dengan bobot terbesar disajikan pada Tabel II.

TABEL II
SUPPORT VECTOR DENGAN BOBOT TERBESAR

x_1	x_2	Bobot α	Label
0,8471	4,4164	4,6929	+1
5,4331	1,7269	3,9345	-1
1,5792	3,5821	3,0538	+1
3,9382	5,5867	2,5855	-1
3,5163	5,2590	1,7311	-1
1,2965	2,7817	1,5513	+1
3,5389	5,6959	1,2388	-1
0,8550	1,6501	0,7497	-1
1,5407	1,1611	0,6855	+1
4,0842	2,4727	0,6558	-1

Pada tabel tersebut, makin besar bobot α , makin besar pengaruh titik itu terhadap letak bidang pemisah. Titik dengan bobot terbesar, yaitu 4,6929, adalah titik yang paling menentukan hasil klasifikasi. Jumlah 12 support vector ini sebanding dengan 11 sampai 13 support vector yang dilaporkan Havlicek dkk. [7], sehingga peta fitur dan cara pelabelan dapat dipastikan sudah dibuat dengan benar.

D. Pengaruh Jumlah Shots

Tabel III memperlihatkan apa yang terjadi bila jumlah pengulangan pengukuran diubah.

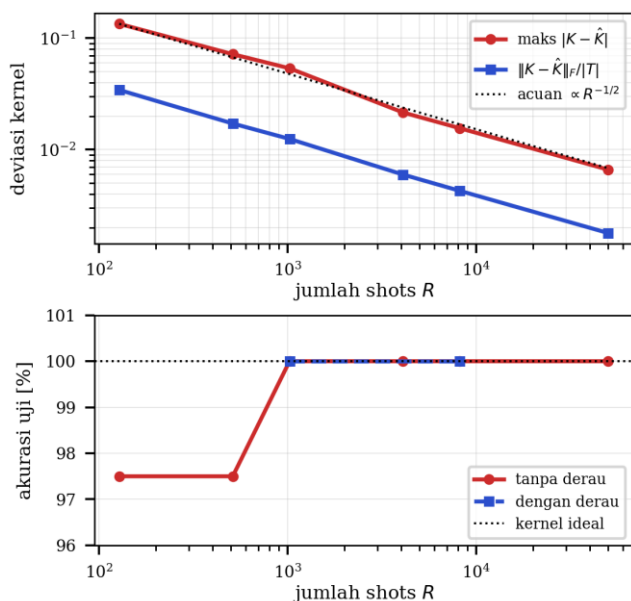
TABEL III
PENGARUH JUMLAH SHOTS

R	Akurasi uji	Selisih terbesar	Eigen negatif
128	97,50%	0,1347	2,019
512	97,50%	0,0719	0,945
1.024	100%	0,0537	0,665
4.096	100%	0,0215	0,298
8.192	100%	0,0156	0,226
50.000	100%	0,0066	0,096
Tepat	100%	0	0

Akurasi latih pada seluruh baris tabel tersebut selalu 100% sehingga tidak ikut ditampilkan. Mulai $R = 1.024$ akurasi uji sudah sempurna. Pada $R = 128$ dan $R = 512$ masih ada satu data uji yang salah tebak sehingga akurasinya 97,5%. Angka

itu pun masih lebih baik daripada 94,75% yang dilaporkan Havlicek dkk. [7] untuk kumpulan data tersulitnya.

Gambar 6 menunjukkan bagaimana selisih tabel kernel mengecil saat shots ditambah.



Gambar 6. Atas: selisih tabel kernel terhadap jumlah shots. Bawah: akurasi uji terhadap jumlah shots.

Kedua sumbu pada gambar bagian atas memakai skala logaritma, sehingga hubungan berbentuk pangkat akan tampak sebagai garis lurus. Kemiringan garisnya $-0,52$ dan $-0,50$. Artinya, untuk menekan selisih menjadi setengahnya, jumlah shots harus diperbanyak sekitar empat kali lipat. Gambar bagian bawah memperlihatkan akurasi yang sudah mendatar di angka 100% sejak $R = 1.024$, baik pada percobaan tanpa derau maupun dengan derau.

Kolom terakhir Tabel III menyampaikan hal yang jarang dibahas. Bobot nilai eigen negatif turun dari 2,019 pada $R = 128$ menjadi 0,096 pada $R = 50.000$. Havlicek dkk. [7] melaporkan angka 1,40, 1,27, dan 2,41 untuk tiga kumpulan datanya, dan angka terbesar dimiliki kumpulan data yang akurasi paling rendah. Pola yang sama muncul di sini: dua percobaan dengan bobot di atas satu justru yang gagal mencapai akurasi penuh. Karena itu bobot eigen negatif dapat dipakai sebagai lampu peringatan bahwa jumlah shots masih kurang, tanpa perlu mengetahui nilai kernel yang sebenarnya.

E. Pengaruh Derau

Dengan derau dipasang, nilai kemiripan turun rata-rata sebesar 0,0144, dengan selisih terbesar 0,171 pada $R = 1.024$ dan 0,162 pada $R = 8.192$. Selisih ini sepuluh kali lebih besar daripada selisih akibat keacakan pengukuran pada jumlah shots yang sama, sehingga pada keadaan ini deraulah penyebab utama kesalahan. Meskipun begitu, akurasi uji tetap 100%.

Penyebabnya, derau menurunkan hampir semua nilai kemiripan dengan besar yang mirip. Urutan mana yang lebih

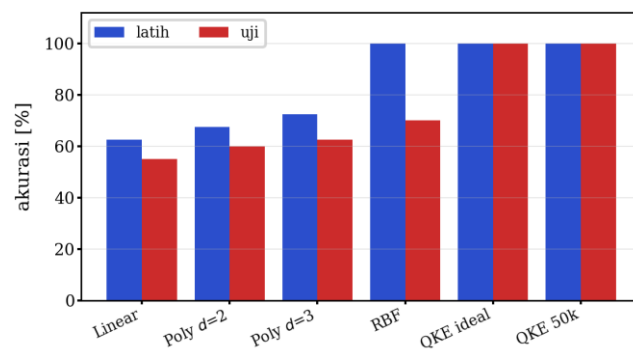
mirip dan mana yang kurang mirip tetap terjaga, sehingga bidang pemisah hampir tidak bergeser. Temuan ini sejalan dengan laporan Havlicek dkk. [7] bahwa akurasi tetap tinggi meski ada derau. Perlu dicatat, penurunan merata seperti ini adalah bentuk ringan dari gejala penumpukan nilai kernel yang pada jumlah qubit besar dapat menghapus kemampuan membedakan data [17].

F. Perbandingan dengan Metode Lain

Tabel IV membandingkan QKE dengan kernel klasik pada data mentah yang sama.

TABEL IV
PERBANDINGAN DENGAN KERNEL KLASIK

Kernel	Akurasi latih	Akurasi uji
Linear	62,50%	55,00%
Polinomial derajat 2	67,50%	60,00%
Polinomial derajat 3	72,50%	62,50%
RBF	100%	70,00%
QKE (hitungan tepat)	100%	100%
QKE (50.000 shots)	100%	100%



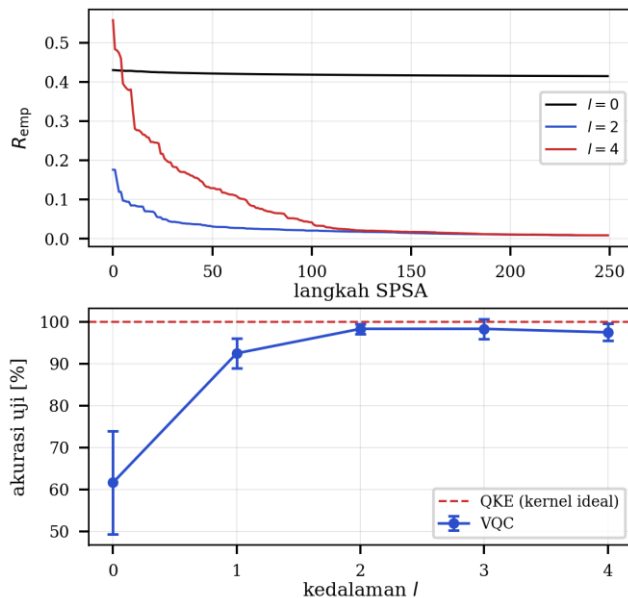
Gambar 7. Perbandingan akurasi latih dan uji antara kernel klasik dan QKE.

Dari tabel dan gambar tersebut terlihat kernel RBF hafal seluruh data latih tetapi hanya benar 70% pada data uji; keadaan seperti ini disebut overfitting. Kernel linear dan polinomial bahkan tidak mampu menghafal data latih sekalipun. Perbandingan ini tidak boleh dibaca sebagai bukti bahwa komputer kuantum lebih unggul, karena datanya memang sengaja dibuat agar cocok dengan peta fitur kuantum yang dipakai. Yang dibuktikan di sini hanyalah bahwa metode QKE berjalan sesuai rancangannya. Penilaian keunggulan yang adil menuntut perbandingan terhadap model klasik yang juga memiliki data latih [13], atau konstruksi masalah dengan percepatan yang terbukti secara matematis [14].

Tabel V dan Gambar 8 menampilkan hasil metode VQC sebagai pembanding kedua.

TABEL V
HASIL METODE VARIASIONAL

Kedalaman	Akurasi latih	Akurasi uji	Nilai kesalahan
0	60,83%	61,67%	0,414
1	93,33%	92,50%	0,014
2	100%	98,33%	0,009
3	100%	98,33%	0,011
4	100%	97,50%	0,009



Gambar 8. Atas: penurunan nilai kesalahan selama pelatihan SPSA. Bawah: akurasi uji terhadap kedalaman rangkaian.

Angka akurasi pada Tabel V adalah rata-rata dari tiga kali pelatihan. Pada kedalaman 0 rangkaian belum memiliki gerbang penjerat sehingga tidak mampu memisahkan data, dan nilai kesalahannya bertahan di sekitar 0,41 seperti terlihat pada kurva paling atas Gambar 8. Mulai kedalaman 2 akurasi sudah di atas 98% dan tidak banyak berubah setelahnya. Pola ini sama dengan yang dilaporkan Havlicek dkk. [7]. QKE tetap sedikit lebih unggul karena tidak memerlukan pelatihan yang hasilnya bergantung pada tebakan awal.

G. Perbandingan Menyeluruh dan Keterbatasan

Tabel VI merangkum kesesuaian hasil penelitian ini dengan angka yang dilaporkan Havlicek dkk. [7].

TABEL VI
PERBANDINGAN DENGAN HASIL HAVLICEK

Aspek	Havlicek	Penelitian ini
Akurasi uji QKE	94,75-100%	100%
Jumlah support vector	11-13	12
Shots per pasangan	50.000	1.024 sudah cukup
Bobot eigen negatif	1,27-2,41	0,10-2,02
Akurasi VQC kedalaman 0	sekitar 50-70%	61,7%
Akurasi VQC kedalaman 2 ke atas	mendekati 100%	97,5-98,3%

Baris pertama membandingkan akurasi uji. Havlicek dkk. [7] memperoleh 100% pada dua kumpulan data dan 94,75% pada kumpulan ketiga, sedangkan penelitian ini memperoleh 100% pada seluruh percobaan yang memakai shots memadai. Sekilas hasil penelitian ini terlihat lebih baik, tetapi perbandingannya tidak setara. Angka 94,75% pada Havlicek muncul karena percobaan mereka berjalan di perangkat keras sungguhan yang mengalami dekoherensi, pergeseran

kalibrasi, dan ketepatan pembacaan yang turun sampai sekitar 94%. Simulator yang dipakai di sini tidak mengalami gangguan tersebut, sehingga wajar bila hasilnya lebih rapi. Jadi baris ini bukan menunjukkan metodenya lebih unggul, melainkan menunjukkan bahwa kesalahan pada percobaan aslinya memang berasal dari perangkat keras.

Baris kedua membandingkan jumlah support vector, yaitu 11 sampai 13 berbanding 12. Di sini tidak ada yang lebih baik atau lebih buruk, karena jumlah support vector ditentukan oleh bentuk sebaran data dan lebar jarak aman, bukan oleh kualitas perangkat. Justru kemiripan angka inilah bukti terkuat bahwa peta fitur, aturan pelabelan, dan jarak aman 0,3 sudah dibuat sama persis dengan penelitian aslinya. Kalau peta fitur salah dibuat, jumlah support vector akan melenceng jauh, misalnya hampir seluruh titik data ikut menjadi support vector.

Baris ketiga membandingkan jumlah shots per pasangan data, yaitu 50.000 berbanding 1.024, atau sekitar empat puluh sembilan kali lebih hemat. Angka 1.024 sudah cukup di sini karena satu-satunya sumber kesalahan adalah keacakan pengukuran, dan pada jumlah itu selisih kernel sudah turun ke 0,0537, cukup kecil dibanding jarak antar nilai kernel yang perlu dibedakan. Havlicek membutuhkan 50.000 karena pengukuran mereka juga harus melawan galat gerbang dan galat pembacaan, ditambah teknik penekan derau yang justru memperbesar keacakan hasil sehingga perlu diimbangi jumlah cuplikan yang jauh lebih besar. Dengan kata lain, penghematan ini berlaku untuk simulator dan belum tentu berlaku bila metode yang sama dijalankan di perangkat keras.

Baris keempat membandingkan bobot nilai eigen negatif, yaitu 1,27 sampai 2,41 berbanding 0,10 sampai 2,02. Untuk ukuran ini semakin kecil semakin baik, karena nilai eigen negatif menandakan tabel kernel sudah menyimpang dari bentuk yang seharusnya. Nilai terbaik penelitian ini, yaitu 0,10 pada 50.000 shots, jauh lebih baik daripada nilai terbaik Havlicek yang 1,27. Penyebabnya sama seperti sebelumnya: pada perangkat keras, galat gerbang dan penekan derau dapat membuat nilai kemiripan yang seharusnya mendekati nol menjadi negatif, dan penyimpangan itu tidak hilang meski shots ditambah. Sebaliknya, nilai terburuk penelitian ini yaitu 2,02 pada $R = 128$ hampir menyamai nilai terburuk Havlicek yaitu 2,41, dan pada kedua kasus akurasi uji sama-sama tidak mencapai 100%. Kesamaan ini memperkuat dugaan bahwa bobot eigen negatif di atas satu dapat dipakai sebagai penanda bahwa hasil pengukuran belum layak dipakai.

Baris kelima dan keenam membandingkan metode variasional. Pada kedalaman 0 keduanya sama-sama gagal, yaitu sekitar 50 sampai 70% pada Havlicek dan 61,7% pada penelitian ini. Kegagalan ini memang sudah diperkirakan, sebab tanpa gerbang penjerat rangkaian hanya dapat memutar tiap qubit sendiri-sendiri sehingga bidang pemisah yang bisa dibentuk sangat terbatas. Rentang angkanya lebar karena hasil pada kedalaman 0 sepenuhnya bergantung pada tebakan awal. Pada kedalaman 2 ke atas, Havlicek melaporkan hasil mendekati 100% sedangkan penelitian ini 97,5 sampai 98,3%. Selisih sekitar dua poin ini sebenarnya sangat kecil: karena

data uji hanya berisi 40 titik, satu titik yang salah tebak sudah menurunkan akurasi sebesar 2,5 poin. Jadi selisih tersebut setara dengan satu titik uji saja, dan wajar muncul karena pelatihan di sini hanya diulang tiga kali dengan tebakan awal acak lalu dirata-ratakan.

Secara keseluruhan, seluruh temuan utama Havlicek muncul kembali pada penelitian ini, dan pada perbandingan antar metode urutannya juga sama: QKE memberi hasil terbaik, metode variasional menyusul di belakangnya, dan kernel klasik tertinggal jauh. QKE lebih unggul daripada metode variasional karena bidang pemisahannya dicari secara langsung oleh SVM, sehingga hasilnya tidak bergantung pada tebakan awal maupun keberhasilan proses pelatihan.

Ada tiga perbedaan yang perlu disebutkan. Pertama, derau yang dipakai di sini lebih sederhana dan tidak mencakup semua gangguan pada perangkat keras sungguhan, sehingga selisih kernel yang teramati lebih kecil. Kedua, teknik penekan derau tidak dipasang karena tanpa teknik itu pun akurasinya sudah penuh. Ketiga, jumlah shots yang benar-benar dibutuhkan ternyata jauh lebih sedikit daripada 50.000. Ini menunjukkan bahwa angka 50.000 pada penelitian aslinya lebih ditujukan untuk melawan galat perangkat keras, bukan sekadar melawan keacakan pengukuran.

Keterbatasan penelitian ini adalah ukurannya yang kecil, yaitu dua qubit dan 40 titik data, serta pemakaian data buatan yang menguntungkan kernel kuantum. Karena itu hasilnya belum bisa dipakai untuk menyimpulkan kinerja pada data nyata. Penelitian berikutnya dapat mencoba data nyata berdimensi lebih besar seperti pada [18], menambahkan langkah perbaikan agar tabel kernel selalu berbentuk benar, melatih parameter peta fitur seperti pada [16], serta menambah jumlah qubit untuk mengamati gejala penumpukan nilai kernel [17].

IV. KESIMPULAN

Metode quantum kernel estimation berhasil diterapkan pada simulator dua qubit memakai Qiskit dan scikit-learn. Dengan tabel kernel yang dihitung tepat, SVM menebak benar seluruh data latih dan data uji memakai 12 support vector. Dengan tabel kernel hasil pengukuran, akurasi penuh sudah tercapai mulai 1.024 shots per pasangan data. Selisih tabel kernel mengecil mengikuti hukum pangkat dengan kemiringan $-0,52$, sehingga menekan selisih menjadi setengahnya membutuhkan sekitar empat kali lipat shots. Bobot nilai eigen negatif ikut mengecil dan dapat dipakai sebagai penanda praktis bahwa jumlah shots sudah memadai.

Pada percobaan dengan derau, nilai kemiripan turun rata-rata 0,0144 dengan selisih terbesar 0,171, tetapi akurasi tetap 100% karena penurunannya hampir merata. Kernel klasik pada data yang sama hanya mencapai akurasi uji tertinggi 70%, sedangkan metode variasional naik dari 61,7% pada kedalaman 0 menjadi 98,3% pada kedalaman 2. Semua temuan utama penelitian Havlicek dkk. [7] muncul kembali, dengan catatan bahwa keunggulan yang terlihat berlaku

husus untuk data buatan yang dibangkitkan oleh peta fitur itu sendiri.

DAFTAR PUSTAKA

- [1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195-202, 2017.
- [2] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.
- [3] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nat. Rev. Phys.*, vol. 3, no. 9, pp. 625-644, 2021.
- [4] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273-297, 1995.
- [5] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Phys. Rev. Lett.*, vol. 113, no. 13, p. 130503, 2014.
- [6] M. Schuld and N. Killoran, "Quantum machine learning in feature Hilbert spaces," *Phys. Rev. Lett.*, vol. 122, no. 4, p. 040504, 2019.
- [7] V. Havlicek, A. D. Corcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209-212, 2019.
- [8] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Phys. Rev. A*, vol. 98, no. 3, p. 032309, 2018.
- [9] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242-246, 2017.
- [10] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Phys. Rev. Lett.*, vol. 119, no. 18, p. 180509, 2017.
- [11] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. Autom. Control*, vol. 37, no. 3, pp. 332-341, 1992.
- [12] M. Schuld, "Supervised quantum machine learning models are kernel methods," *arXiv preprint arXiv:2101.11020*, 2021.
- [13] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, "Power of data in quantum machine learning," *Nat. Commun.*, vol. 12, p. 2631, 2021.
- [14] Y. Liu, S. Arunachalam, and K. Temme, "A rigorous and robust quantum speed-up in supervised machine learning," *Nat. Phys.*, vol. 17, no. 9, pp. 1013-1017, 2021.
- [15] J. R. Glick, T. P. Gujarati, A. D. Corcoles, Y. Kim, A. Kandala, J. M. Gambetta, and K. Temme, "Covariant quantum kernels for data with group structure," *Nat. Phys.*, vol. 20, no. 3, pp. 479-483, 2024.
- [16] T. Hubregtsen, D. Wierichs, E. Gil-Fuster, P.-J. H. S. Derks, P. K. Faehrmann, and J. J. Meyer, "Training quantum embedding kernels on near-term quantum computers," *Phys. Rev. A*, vol. 106, no. 4, p. 042431, 2022.
- [17] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes, "Exponential concentration in quantum kernel methods," *Nat. Commun.*, vol. 15, p. 5200, 2024.
- [18] E. Peters, J. Caldeira, A. Ho, S. Leichenauer, M. Mohseni, H. Neven, P. Spentouris, D. Strain, and G. N. Perdue, "Machine learning of high dimensional data on a noisy quantum processor," *npj Quantum Inf.*, vol. 7, p. 161, 2021.
- [19] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with Qiskit," *arXiv preprint arXiv:2405.08810*, 2024.
- [20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825-2830, 2011.