IMPLEMENTASI METODE CONVOLUTIONAL NEURAL NETWORK UNTUK PENEGENALAN HURUF AKSARA SASAK PADA ANDROID

(*Implementation Convolutional Neural Network Method for Recognition of Sasak Characters in Android)*

Reza Rismawandi, I Gede Pasek Suta Wijaya, Gibran Satya Nugraha
Program Studi Teknik Informatika, Fakultas Teknik, Universitas Mataram
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA
Email: xreza75@gmail.com, [gpsutawijaya, gibransn]@unram.ac.id

Abstract

Huruf aksara merupakan salah satu warisan dari suku sasak dan perlu untuk dijaga keberadaannya, salah satunya adalah aksara sasak. seiring berkembangnya zaman huruf aksara sasak mulai semakin jarang dipelajari untuk itu diperlukan cara agar tetap dapat lestari diantaranya dengan membuat mesin untuk mengenali aksara sasak. Pada penelitian ini akan mengenali huruf aksara sasak dari model CNN yang telah dilatih untuk diimplementasikan ke perangkat android. CNN merupakan suatu model pembelajaran mesin yang dapat digunakan untuk melakukan klasifikasi pada suatu gambar. Dataset yang digunakan yaitu sebanyak 10.800 yang terdiri dari 5400 data tulisan tangan yang diambil pada empat kategori yaitu SD, SMP, SMA dan Perguruan Tinggi. Pada penelitian ini model terbaik yang diperoleh menghasilkan akurasi sebesar 99.31%. Hasil implementasi model ke android diujikan kepada 45 responden dan memperoleh rata-rata akurasi sebesar 80.83% dengan akurasi tertinggi 95.56% dan terendah sebesar 51.11%. Hasil akurasi yang diperoleh dipengaruhi oleh bentuk tulisan pada perangkat android.

Keywords: *Pengenalan pola, Tulisan tangan, aksara sasak, Convolutional Neural Network, Android*

1. PENDAHULUAN

Huruf aksara sasak adalah huruf peninggalan suku sasak yang digunakan untuk menulis oleh suku sasak pada masa lalu. Huruf aksara sasak memiliki 18 karakter dengan masing-masing karakter memiliki bentuk yang unik. Dewasa ini Huruf aksara sasak sudah jarang dipelajari karena generasi muda sekarang lebih tertarik untuk belajar bahasa asing daripada mempelajari budayanya sendiri untuk itu diperlukan suatu cara agar huruf aksara sasak dapat terus dilestarikan[1]. Salah satu cara yang bisa digunakan untuk melestarikan suatu pengenalan aksara sasak yang selanjutnya dapat dikembangkan menjadi aplikasi pembelajaran untuk aksara sasak.

Penelitian sebelumnya mengenai pengenalan aksara sasak pernah dilakukan diantaranya dengan menggunakan metode *moment invariant* dan *support vector machine*. Pada penelitian ini menggunakan dataset 2700 tulisan tangan aksara sasak diperoleh rata-rata akurasi sebesar 92.25%[1]. Penelitian ini berhasil mendapat akurasi yang baik untuk mengenali huruf aksara sasak.

Penelitian lain yang menggunakan metode *integral projection* dan *neural network* mendapatkan akurasi sebesar 41,38%[2]. Pada penelitian ini hasil akurasi yang didapatkan yang masih di bawah 50% . Metode *integral projection* dan *neural network* masih belum optimal dalam melakukan pengenalan pada huruf aksara sasak. Kedua penelitian sebelumnya masih hanya sebatas menemukan model pembelajara mesin untuk mengenali aksara sasak.

Seiring dengan perkembangan zaman metode pembelajaran mesin semakin berkembang dan terdapat metode baru yaitu *Convolutional Neural Network* (CNN). CNN adalah salah satu jenis neural network yang biasa digunakan pada data image. CNN bisa digunakan untuk mendeteksi dan mengenali object pada sebuah image. Secara garis besar CNN tidak jauh beda dengan *neural network* biasanya. CNN terdiri dari neuron yang memiliki weight, bias dan activation function. Proses yang dilakukan oleh CNN, yaitu: *convolutional layer*, *non-linearity layer* (*relu layer*), *pooling layer*, dan *fully connected layer*[3].

Penggunaan metode CNN dalam pembelajaran mesin telah banyak dilakukan dan memperoleh hasil yang baik. Salah satu penelitian yang menggunakan metode CNN yaitu pengenalan aksara jawa dengan menggunakan metode CNN untuk mengenali huruf aksara jawa. Penelitian dilakukan pada aksara jawa bersambung sehingga perlu dilakukan segmentasi

sebelum dilakukan training. penelitian ini menghasilkan akurasi sebesar 95.04%[4].

Pada zaman sekarang ini banyak perangkat elektronik khususnya smartphone yang menggunakan sistem operasi Android. Android merupakan sistem operasi mobile dengan jumlah pengguna terbanyak. Android memiliki market share sebesar 39,16% meninggalkan Windows Desktop yang hanya 38.04%. Meningkatnya populasi pengguna Android didorong oleh ekosistem aplikasi pada Android yang mudah dimana developer dapat dengan mudah memasukkan aplikasinya ke dalam Google PlayStore. Selain itu di era serba cepat dan instan saat ini membuat kebanyakan orang lebih sering menggunakan mobile daripada desktop dengan alasan lebih praktis untuk dibawa-bawa. Maka dari itu dengan membuat aplikasi pada Android dapat digunakan oleh lebih banyak pengguna[5].

Berdasarkan pemaparan tersebut, penulis mengajukan sebuah penelitian untuk merancang sebuah model pembelajaran mesin untuk mengenali aksara sasak menggunakan metode CNN untuk diterapkan pada android. Penelitian ini diharapkan dapat menghasilkan aplikasi android untuk pengenalan aksara sasak dengan tingkat akurasi yang tinggi.

2. TINJAUAN PUSTAKA

Penelitian tentang pengenalan aksara sasak sebelumnya telah dilakukan beberapa kali. Penelitian-penelitian yang dimaksud antara lain pengenalan pola tulisan tangan huruf sasak menggunakan metode integral projection dan neural network[2], pengenalan pola tulisan tangan suku kata aksara sasak menggunakan metode moment invariant dan support vector machine[1].

Penelitian menggunakan metode moment invariant dan support vector machine mendapat hasil yang baik dimana didapatkan rata-rata akurasi sebesar 92.52% namun untuk menghasilkan hasil tersebu dibutuhkan banyak fitur dari hasil ekstraksinya. Kemudian penelitian dengan metode integral projection dan neural network memiliki tingkat akurasi masih dibawah 50%. Seiring berkembangnya zaman terdapat sebuah metode tanpa adanya proses ekstraksi yaitu metode Convolutional Neural Network (CNN).

Penelitian menggunakan metode convolution neural network telah banyak dilakukan oleh para peneliti untuk dapat mengenali atau mengklasifikasikan pola suatu citra dalam beberapa terakhir. Penelitian-penelitian sebelumnya akan dijadikan sebagai rujukan ketika pelaksanaan penelitian ini.

Penelitian menggunakan metode CNN untuk mengklasifikasikan citra sudah pernah dilakukan beberapa kali pada interval tahun 2016-2020. Penelitian-penelitian tersebut antara lain tentang pengenalan spesies ikan berdasarkan kontur *otolith* menggunakan *convolutional neural network*[6] penelitian menggunakan 403 data citra yang terdiri dari 14 kelas dan hasilnya mendapat akurasi sebesar 95.4%, selanjutnaya penelitian untuk pendeteksian tulang patah pada citra ultrasonic *B-mode* menggunakan *convolutional neural network*[7] hasil akurasi yang diperoleh pada penilitian ini sebesar 95.3%,

Peneilitan lain yaitu pengenalan objek video pada CCTV[8] data yang digunakan adalah data hasil ekstraksi video kemudian dimasukan ke program untuk dikenali hasil akurasi yang didapatkan pada penelitian ini sebesar 80% , Penelitan pada untuk pengenalan aksara jawa dengan menggunakan metode convolutional neural network[9] menggunakan data set berjumlah 53423 berhasil mendapat akurasi sebesar 95.04% untuk melakukan pengenalan pada aksaa jawa ,

pengenalan pola motif kain grinsing menggunakan convolutional neural network[10] total dataset yang digunakan berjumlah 3014 ditambah dengan 150 data kain non grinsing penelitian ini mendapatkan akurasi sebesar 76%, Pengenalan jenis tumbuhan berdasarkan citra daun[11] dilkaukan menggunakan dataset berjumlah 20000 dengan jumlah kelas sebanyak 20 penelitian ini mendapat akurasi sebesar 90.8% .

Berdasarkan penelitian-penelitian yang sudah dilakukan sebelumnya, dapat dilihat bahwa convolution neural network dapat bekerja dengan baik untuk pengklasifikasian citra. Penelitian sebelumnya masih menggunakan proses ekstraksi seblum klasisfikasinya dan belum menerapkan metodenya pada perangkat android. Oleh karena itu, penulis bermaksud untuk menggunakan metode ini melakukan pengenalan terhadap tulisan aksara sasak berbasis android.

3. METODE PENELITIAN

3.1. Proses Penelitian

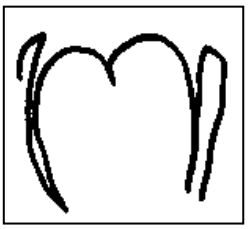
Terdapat beberapa tahap yang dilakukan dalam penelitian yang dimulai dari studi literatur yang dimaksudkan untuk menambah pengetahuan penulis dalam memahami apa yang akan dilakukan. Studi literatur dilakukan dengan membaca refrensi mengenai apa yang dilakukan baik berupa buku, jurnal, maupun penelitian-penelitian sebelumnya. Kemudian dilakukan pengumpulan dataset yang akan digunakan sebagai data untuk melatih sekaligus menguji model pemebeljaran mesin yang dibuat. Selanjutnya akan dilakukan pembangunan model pembelajaran mesin dengan metode CNN yang akan digunakan untuk

melakukan pengenalan pada huruf aksara sasak. Hasil yang didapat dari model kemudian akan dianalisa untuk mengetahui performa dari model yang telah dilatih. Jika model yang dihasilkan belum memiliki performa yang baik maka akan dilakukan perancangan ulang dari model yang dibangun. Dan jika sudah mendapat hasil sesuai dengan yang diinginkan maka akan diambil kesimpulan mengenai performa dari model. Terakhir, dilakukan penyelesaian dokumentasi penelitian dalam bentuk laporan akhir.

3.2. Pengumpulan Dataset

Pada penelitian ini,dataset dikumpulkan dengan meminta orang pada jenjang SD, SMP, SMA, serta Perguruan Tinggi untuk menulis aksara sasak pada kertas A4 yang telah diberikan tabel berukuran 4x4cm untuk ditulis oleh huruf aksara sasak. Setiap jenjang diwakili oleh 10 orang dan setiap orang akan menulis huruf aksara sasak sebanyak 15 kali. Hasil tulisan tangan tersebut selanjutnya di *scan* hasilnya dapat dilihat pada Gambar 2 kemudian hasil scan tadi akan di *crop* untuk memisahkan masing-masing huruf aksara agar tidak berada pada satu gambar hasil *crop* dapat dilihat pada Gambar 3.

Gambar 1. Hasil scan tulisan tangan

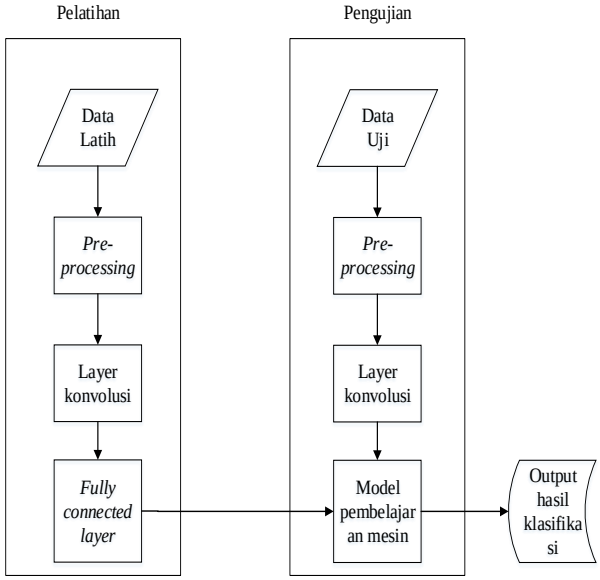


Gambar 2. Contoh hasil crop

Dataset yang terkumpul berjumlah 10800 data yang terdiri dari 18 kelas dan tiap kelasnya terdapat 600 buah gambar dalam format png. Hasil dataset ini selanjutny akan digunakan untuk mencarai model CNN untuk melakukan pengenalan pada aksara sasak.

3.3. Pembangunan Model

Terdapat dua tahap dalam pembuatan model yaitu tahap pelatihan dan pengujian. Tahap pelatihan adalah tahap pertama yang dilakukan yang dilakukan dimana pada proses ini akan dilakukan pengolahan citra dari data latih untuk dijadikan model pembelajaran mesin dengan CNN yang akan digunakan untuk melakukan pengenalan terhadap aksara sasak. Tahap kedua adalah tahap pengujian dimana pada tahap ini akan dilakukan pengujian dari model yang didapat dari proses pelatihan untuk melihat seberapa baik model yang didapat dalam melakukan pengenalan aksara sasak. Pengujian dilakukan dengan menggunakan data uji dari citra aksara sasak. Lebih jelasnya, alur klasifikasi dapat dilihat pada Gambar 3.



Gambar 3. Contoh hasil crop

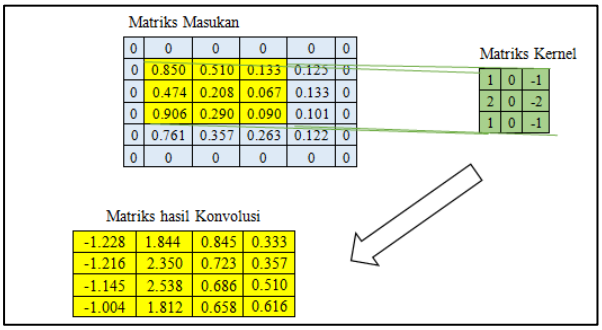
3.3.1. Preprocessing

Preprocessing dilakukan untuk mengolah dataset yang ada agar bisa mendapat hasil yang optimal ketika masuk ke dalam model CNN. Dataset yang ada akan dilakukan *resize* untuk menyamakan ukuran gambarnya. Ukuran gambar yang digunakan yaitu 32x32, 64x64, dan 96x96. Hasil *resize* nanti akan digunakan pada saat melakukan *training* pada data. Pembuatan gambar menjadi 3 variasi ukuran dimaksudkan untuk melihat pengaruh ukuran citra terhadap mode yang dibuat.

3.3.2. Struktur model

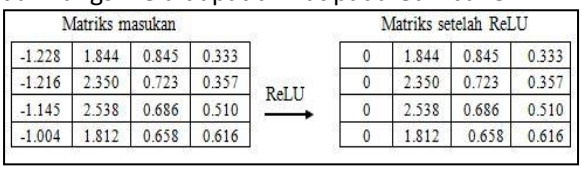
1. Layer konvolusi

   Pada layer konvolusi dilakukan proses konvolusi yang bertujuan untuk melakukan filter terhadap matriks masukan. Kemudian agar semua matriks dapat di konvolusi dan ukurannya tetap maka dilakukan *zero padding*. Konvolusi dilakukan dengan menggunakan matriks ukuran 3x3. Keluaran dari layer konvolusi ini akan menjadi masukan pada layer pooling[12]. Proses konvolusi dapat dilihat pada Gambar 4.



Gambar 4. Proses Konvolusi

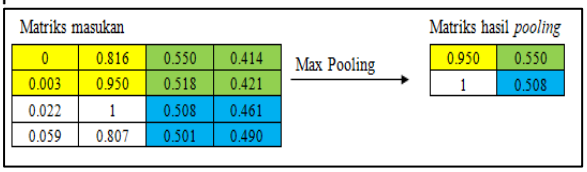
Setelah dilakukan konvolusi matriks citra akan memasuki *pooling layer* namun sebelum itu akan dilakukan fungsi aktivasi terlebih dahulu agar nilai dari matriks tidakada yang bernilai negatif. Aktivasi ReLU (Rectified Linear Unit) merupakan lapisan aktivasi pada model CNN yangmengaplikasikan fungsi f(x) = max(0,x) dimana nantinya nilai yang dibawah 0 akan dijadikan 0[13]. Visualisasi matriks dari fungsi Relu dapat dilihat pada Gambar 5.



Gambar 5. Aktivasi ReLU

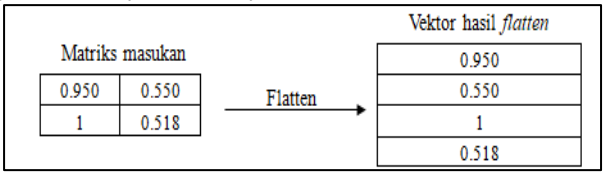
2. *Pooling layer*

Pada max pooling dilakukan proses untuk mengurangi ukuran citra namun tetap menyimpan informasi yang ada pada citra tersebut[12]. Penelitian ini menggunakan pooling dengan matriks 2x2 dengan stride sebesar 2. Jadi , pooling akan bergeser sebanyak 2 indeks dan mencari nilai terbesar dari pooling atau bisa disebut dengan istilah max pooling. Proses pooling dapat dilihat pada Gambar 6.



Gambar 6. Proses konvolusi

Sebelum memasuki *fully connected layer* matriks akan diubah menjadi vector atau di *flatten.* Proses *flatten d*dapat dilihat pada Gambar 7.



Gambar 7. Proses flatten

3. *Fully connected layer*

Layer yang digunakan pada model yang dibangun berjumlah 2 layer dimana masing-masing hidden layer memiliki 64 *neurons*. Akan tetapi, model *fully connected layer* kemungkinan akan menimbulkan overfitting. Oleh karena itu, diterapkan *dropout* pada model ini untuk mengurangi hal tersebut. *Dropout* nantinya akan melakukan pemilihan neuron secara acak dari tiap layer yang ada untuk di-nonaktif-kan sehingga pada perhitungan layer tidak terlalu banyak *weight* dan *neuron* yang terlibat. Hal itu dilakukan untuk mengurangi kemungkinan terjadi *overfitting*.

3.4. Pengujian Model

Pengujian dilakukan untuk menguji model yang telah dihasilkan dari hasil pelatihan sebelumnya. Model yang dibangun dengan memperhatikan dataset, serta parameter percobaan seperti ukuran kernel konvolusi, ukuran layer, *epoch*, serta skenario pelatihan. Pengujian dilakukan untuk melihat seberapa baik akurasi, presisi dan *recall* model yang dibuat dalam melakukan penegenalan huruf aksara sasak. Metode yang digunakan untuk melakukan pengujian adalah confusion matrix. Berikut adalah persamaan akurasi pada Persamaan (1), perhitungan presisi pada Persamaan (2) dan peerhitungan recall pada Persamaan (3)[14].

$$Akurasi = \frac{Jumlah\ data\ sesuai\ target}{Total\ seluruh\ data} \quad (1)$$

$$Presisi = \frac{jumlah\ data\ yang\ sesuai\ target\ sesuai\ kelas}{Jumlah\ seluruh\ data\ yang\ sesuai\ target} \quad (2)$$

$$Recall = \frac{jumlah\ data\ yang\ sesuai\ target\ di\ satu\ kelas}{\text{jumlah data di satu kelas}} \quad (3)$$

3.5. Implementasi Model Ke Android

Perancangan desain sistem ini dilakukan sesuai dengan fungsi aplikasi yang dibuat yaitu untuk melakukan penegenalan terhadap tulisan aksara sasak. Sistem akan dibuat menggunakan *software* IDE Android Studio. Perancangan desain sistem yang dibuat dapat dilihat pada Gambar 3.9.

Gambar 8. Desain sistem

Gambar 8 merupakan halaman utama dari aplikasi yang akan dibuat dimana nantinya aplikasi ini hanya akan melakukan pengenalan pada huruf aksara sasak. Pada bagian atas terdapat kotak untuk pengguna menuliskan huruf aksara sasak dimana nantinya hasil prediksi huruf tersebut akan ditampilkan pada kotak yang ada dibawahnya. Tombol hapus pada akan berfungsi untuk menghapus isi dari kotak tempat menulis apabila pengguna ingin menulis huruf aksara yang baru.

Model akan diimplementasikan menggunakan *tensorflow lite* dimana *tensorflowlite* terdiri dari *software* untuk menjalankan model sehingga dapat digunakan pada perangkat seluler[15].

4. HASIL DAN PEMBAHASAN

4.1. Skenario Pengujian

Penelitian kali ini akan membagi dataset menjadi 80% untuk pelatihan dan 20% untuk pengujian. Pembagian ini dilakukan untuk melakukan pengujian pada model yang akan dibuat. Dataset yang digunakan adalah dataset yang telah dikumpulkan sebelumnya dengan jumlah 10800 gambar.

Selanjutnya, dilakukan pengujian terhadap model CNN yang dibangun menggunakan dataset yang sudah disebutkan sebelumnya. Pengujian ini dilakukan dengan berbagai parameter dengan urutan pengujian sebagai berikut.

1. Jumlah kernel Konvolusi (1,2,3)
2. Ukuran kernel konvolusi (kernel 3x3 dan kernel 5x5)
3. Jumlah hidden layer neural network (1HL, 2HL, dan 3HL)
4. Ukuran neuron hidden layer (32 neuron, 64 neuron, dan 128 neuron)
5. Ukuran citra masukan (32x32, 64x64, dan 96x96)
6. Bobot learning rate (0.1, 0.01,dan 0.0001)
7. Batas epoch (10, 50, 100)

Model terbaik selanjutnya disimpan ke dengat format h5 untuk kemudian dikonversi ke model tensorflowlite dengan format tflite. Hasil konversi selanjutnya akan diterapkan pada sistem android untuk dilakukan pengujian. Pengujian ini dilakukan untuk melihat performa dari model pada sistem android.

4.2. Hasil Pengujian

Pengujian dilakukan menggunakan arsitektur awal menggunakan parameter dari scenario pengujian yaitu kernel ukuran 3x3, jumlah filter 64, jumlah *hidden layer* 1, jumlah *neuron hidden layer* 64, *dropout* 20%, learning rate 0.1 dengan *epoch* sebanyak 10. Karena keterbatasan sumber daya skenario pengujian yang dilakukan terbatas dan kurang variatif.

4.2.1. Pengujian Terhadap Ukuran Kernel

Kernel digunakan sebagai pengali pada citra untuk mendapatkan fitur dari citra tersebut. Ukuran kernel biasanya ganjil agar memiliki titik pusat pada saat dilakukan pengalian terhadap citra. Pada pengujian ini akan digunakan 2 variasi kernel yaitu ukuran 3x3 dan 5x5. Hasil pengujian dapat dilihat pada Tabel II.

TABEL II. HASIL PENGUJIAN TERHADAP UKURAN KERNEL

Ukuran Kernel	Akurasi *Train* (%)	Akurasi *Testing* (%)	Presisi (%)	Recall (%)
3x3	90.87	94.63	93.35	88.19
5x5	92.06	93.38	94.16	90.13

Berdasarkan Tabel II Hasil performa terbaik didapatkan pada kernel ukuran 5x5 jika dibandingkan dengen filter ukuran 3x3 dengan selisih akurasi sebesar 1.19%. Hal tersebut menunjukan ukuran kernel penambahan ukuran kernel dapat meningkatkan akurasi. Selanjutnya ukuran kernel 5x5 akan digunakan sebagai parameter untuk mencari jumlah filter konvolusi terbaik yang digunakan dalam arsitektur.

4.2.2. Pengujian Terhadap Jumlah Filter Konvolusi

Jumlah filter konvolusi menunjukan banyak kernel yang digunakan untuk melakukan konvolusi pada citra. Jumlah kernel tidak memiliki ukuran yang pasti semakin banyak atau sedikit tidak menjamin bertambah atau berkurangnya performa. Pada pengujian ini akan menggunakan 3 variasi ukuran kernel yaitu 32, 64, dan 128 hail pengujian dapat dilihat pada Tabel III.

TABEL IIII. HASIL PENGUJIAN TERHADAP JUMLAH FILTER

Jumlah Filter	Akurasi *Train* (%)	Akurasi *Testing* (%)	Presisi (%)	Recall (%)
32	75.09	90.42	84.54	67.34
64	92.06	93.38	94.16	90.13
128	90.31	94.12	92.75	87.91

Berdasarkan Tabel III diperoleh performa terbaik pada filter dengan jumlah 64 jika dibandingkan dengan

filter yang berjumlah 32 dan 128. Pada jumlah filter 64 diperoleh akurasi sebesar 92.06% memiliki selisih sebesar 1.75% dengan kernel ukuran 128 dan 15.22% dengan ukuran 32. Berdasarkan hasil tersebut semakin banyak atau sedikit jumlah filter tidak menjamin peningkatan akurasi. Selanjutnya jumlah filter konvolusi akan digunakan sebagai arsitektur untuk mencari jumlah hidden layer terbaik pada model yang dibuat.

4.2.3. Pengujian Terhadap Jumlah Hidden Layer

Hidden layer merupakan layer yang dimasuki setelah melewati layer konvolusi dan dilakukan *flatten*. Jumlah dari *hidden layer* tidak memiliki ukuran yang pasti, pada pengujian kali ini akan digunakan 3 variasi ukuran *hidden layer* yaitu 1, 2, dan 3. Hasil pengujian dapat dilihat pada tabel IV.

TABEL IV. HASIL PENGUJIAN TERHADAP JUMLAH *HIDDEN LAYER*

Jumlah HL	Akurasi *Train* (%)	Akurasi *Testing* (%)	Presisi (%)	Recall (%)
1	92.06	93.38	94.16	90.13
2	92.12	93.47	93.88	90.13
3	93.31	93.89	95.02	91.91

Berdasarkan tabel IV diperoleh performa terbaik pada hidden layerdengan jumlah 3 jika dibandingkan dengan *hidden layer* yang berjumlah 1 dan 2. Akurasi pada model dengan 3 hidden layer sebesar 93.31% memiliki selisih 1.19% dengan model dengan 2 *hidden layer* dan 1.25% dengan model dengan 1 *hidden layer*. Berdasarkan hasil tersebut penambahan jumlah *hidden layer* dapat meningkatkan akurasi. Selanjutnya jumlah 3 *hidden layer* akan dijadikan arsitektur dalam pengujian untuk mencari jumlah *neuron hidden layer* terbaik untuk model yang dibuat.

4.2.4. Pengujian Terhadap Jumlah Neuron pada Hidden Layer

Neuron merupakan bagian dari hidden layer, jumlah *neuron* tidak memiliki aturan yang pasti pada pengujian kali ini akan digunakan 3 jenis variasi jumlah *neuron* yairu 32, 64, dan 128 Hasil pengujian dapat dilihat pada Tabel V.

TABEL V. HASIL PENGUJIAN TERHADAP JUMLAH *NEURON*

Jumlah neuron	Akurasi *Train* (%)	Akurasi *Testing* (%)	Presisi (%)	Recall (%)
32	87.94	94.62	90.57	84.91
64	93.31	93.89	95.02	91.91
128	94.42	90.00	95.23	93.03

Berdasarkan Tabel V didapatkan performa terbaik pada *neuron* dengan jumlah 128 jika dibandingkan dengan *neuron* dengan jumlah 32 dan 64. Akurasi yang diperoleh dari model dengan jumlah *neuron* 128 sebesar 94.42% lebih besar 1.11% daripada *neuron* dengan jumlah 64 dan lebih besar 6.48% dari *neuron* dengan jumlah 32. Berdasarkan hal tersebut penambahan jumlha *neuron* meningkatkan akurasi dari model. Selanjutnya jumlah neuron 128 akan dijadikan arsitektur untuk mencari *learning rate* dengan performa terbaik pada model yang dibuat.

4.2.5. Pengujian Terhadap Learning Rate

Learning rate digunakan sebagai nilai yang untuk menghitung nilai koreksi bobot pada saat *training*. L*earning rate* tidak memiliki nilai yang pasti pada suatu model. Rentang nilai *learning rate* antara 0-1, pada pengujian kali ini digunakan 3 jenis variasi *learning rate* yaitu 0.1, 0.01, dan 0.0001 untuk dilihat performanya pada model yang dibuat. Hasil dari pengujian dapat dilihat pada Tabel VI.

TABEL VI. HASIL PENGUJIAN TERHADAP *LEARNING RATE*

Learning Rate	Akurasi *Train* (%)	Akurasi *Testing* (%)	Presisi (%)	Recall (%)
0.1	94.84	95.05	95.79	93.97
0.01	92.97	92.41	94.80	91.22
0.0001	83.53	88.89	88.85	78.94

Berdasarkan Tabel VI diperoleh performa terbaik pada model dengan *learning rate* 0.1 dibandingkan dengan *learning rate* dengaan nilai 0.01 dan 0.0001. Akurasi yang didapatkan pada model dengan *learning rate* 0.1 sebesar 98.48% sedangkan untuk *learning rate* 0.01 dan 0.0001 masing masing sebesar 92.97% dan 83.53%. Waktu komputasi yang dibutuhkan untuk *learning rate* 0.1, 0.01, dan 0.0001 masing masing sebesar 1342 detik, 1433 detik, dan 1022 detik. Waktu komputasi paling rendah diperoleh pada *learning rate* 0.0001 sementara tertinggi pada *learning rate* 0.01.

4.2.6. Pengujian Terhadap Bobot *Dropout*

Penggunaan *dropout* pada model dimaksudkan untuk mengurangi terjadinya overfitting atau akurasi testing jauh lebih tinggi daripada *testing*. Pada pengujian ini akan digunakan 3 variasi *dropout* yaitu 20%, 30% dan 50%. Hasil pengujian dapat dilihat pada Tabel VII.

TABEL VII. HASIL PENGUJIAN TERHADAP BOBOT *DROPOUT*

Bobot Dropout	Akurasi *Train* (%)	Akurasi *Testing* (%)	Presisi (%)	Recall (%)
20	94.42	90.00	95.23	93.03
30	93.31	91.94	94.69	92.00
50	94.84	95.05	95.79	93.97

Berdasarkan Tabel VII diperoleh akurasi tertinggi pada *dropout* dengan bobot 50% dibandingkan dengan *dropout* dengan bobot 20% dan 30%. Akurasi yang didapat pada *dropout* dengan bobot 50% sebesar 94.84% lebih besar 1.53% dari *dopout* dengan bobot 30% dan lebih besar 0.42% dari *dropout* dengan bobot 20%. Berdasarkan hasil pengujian penambahan bobot *dropout* menyebabkan selisih antara akurasi *train* dan

testing menjadi semakin kecil pada *dropout* dengan bobot 20%, 30%, dan 50% masing memiliki selisih akurasi *training* dan *testing* sebesar 4.42%, 1.37%, dan 0.21%. Selanjutnya bobot *dropout* 50% akan digunakan dalam model untuk menguji ukuran citra masukan pada model yang memiliki performa terbaik.

4.2.7. Pengujian Terhadap Ukuran Citra Masukan

Ukuran citra masukan didapatkan pada saat proses *pre-procesing* dimana semakin besar ukuran citra maka detail citra semakin baik namun waktu komputasi yang dibutuhkan semakin banyak sebaliknya semakin kecil ukuran citra detail dari citra menjadi berkurang namun waktu komputasi yang dibutuhkan semakin sedikit pada pengujian ini akan mencoba 3 variasi ukuran citra masukan yaitu 32x32, 64x64, dan 96x96. Hasil pengujian dapat dilihat pada Tabel VIII.

TABEL VIII. HASIL PENGUJIAN TERHADAP UKURAN CITRA MASUKAN

Ukuran Citra	Akurasi *Train* (%)	Akurasi *Testing* (%)	Presisi (%)	Recall (%)	Waktu komputasi (s)
32x32	84.31	84.31	89.00	80.37	370
64x64	94.84	95.05	95.79	93.97	1,342
96x96	94.06	94.95	95.48	92.41	4,300

Berdasarkan Tabel VIII diperoleh performa terbaik pada masukan citra dengan ukuran 64x64 dibandingkan dengan masukan citra dengan ukuran 32x32 dan 96x96. Nilai akurasi dari masukan citra dengan ukuran 64x64 memiliki akurasi sebesar 98.84% lebih besar 10.53% dibandingkan ukuran citra 32x32 dan lebih besar 0.78% dibandingkan ukuran citra 96x96.Hasil tersebut menunjukan ukuran citra masukan yang kecil memiliki waktu komputasi yang lebih sedikit namun akurasi yang didapatkan berkurang sedangkan pada citra masukan yang lebih besar yaitu 96x96 memiliki waktu komputasi yang lebih lama dan akurasi yang dihasilkan masih lebih kecil daripada masukan citra ukuran 64x64 yang artinya semakin besar atau kecil ukuran citra masukan tidak menjamin mendapatkan akurasi model yang lebih baik. Selanjutnya dilakukan percobaan untuk melihat pengaruh *epoch* terhadap model dan ukuran citra masukan yang digunakan adalah citra berukuran 64x64.

4.2.8. Pengujian Terhadap Jumlah Epoch

Epoch adalah ketika dataset sudah melalaui proses *training* dan kembali lagi ke awal. Setiap kali *epoch* akan memperbaharui bobot *weight,* semakin banyak *epoch* akan memerlukan lebih banyak waktu komputasi. Pada pengujian ini akan digunakan 3 variasi batas *epoch* yaitu 10, 50, dan 100. Hasil pengujian dapat dilihat pada Tabel IX.

TABEL IX. HASIL PENGUJIAN TERHADAP JUMLAH *EPOCH*

Jumlah Epoch	Akurasi *Train* (%)	Akurasi *Testing* (%)	Presisi (%)	Recall (%)
10	94.84	95.05	95.79	93.97
50	97.94	96.81	98.30	97.84
100	99.31	96.81	95.48	99.25

Berdasarkan Tabel IX diperoleh performa terbaik pada *epoch* dengan jumlah 100 dibandingkan dengan *epoch* dengan jumlah 10 dan 50. Pada *epoch* dengan jumlah 10 akurasi yang didapatkan 98.84% penambahan jumlah *epoch* ke 50 meningkatkan akurasi dari model sebesar 3.1% dan penambahan *epoch* dari 50 ke 100 meningkatkan akurasi sebesar 1.37%. Hal tersebut menunjukan penambahan jumlah *epoch* pada saat *train* meningkatkan akurasi dengan jumlah yang kecil.

4.2.9. Model hasil pengujian

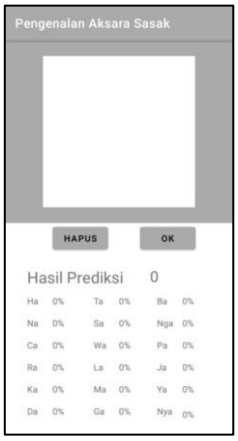
Berdasarkan pengujian yang telah dilakukan didapatkan model terbaik dengan ukuran kernel 5x5, jumlah filter pada kernel berjumlah 64, 3 *hidden layer* dengan masing tersidiri dari 128 *neuron*, *learning rate* 0.1, dan *dropout* 50% dengan ukuran citra masukan 64x64. *Summary* dari model yang didapatkan sebagai berikut.

- Ukuran citra masukan 64x64 dimensi citra RGB
- Layer konvolusi 1 (kernel 5x5, filter 128, aktivasi ReLU)
- Layer konvolusi 2 (kernel 5x5, filter 128, aktivasi ReLU)
- Layer konvolusi 3 (kernel 5x5, filter 128, aktivasi ReLU)
- *Hidden layer* 1 (*neuron* 128, *activation* ReLU)
- *Hidden layer* 2 (*neuron* 128, *activation* ReLU)
- *Hidden layer* 3 (*neuron* 128, *activation* ReLU)
- *Learning rate* 0.1 , *dropout* 50%

Waktu yang dibutuhkan untuk model melakukan prediksi pada dataset sebesar 348549.272 ms jika dirata-ratakan maka didapat hasil 32.273 ms untuk melakukan predict pada tiap gambar

4.3. Implementasi Model Pada Android

Implementasi model dilakukan dengan menggunakan Android Studio berdasarkan rancangan yang telah dibuat sebelumnya dengan mengimplementasikannya kedalam kode-kode program menggunakan bahasa java serta xml untuk tampilannya. Aplikasi yang dibuat hanya terdapat sebuah halaman untuk memprediksi huruf tulisan tangan yang ditulis oleh pengguna aplikasi. Pada halaman utama akan terdapat kotak untuk menulis, 2 buah tombol untuk memprediksi dan menghapus hasil tulisan tangan serta sebuah kotak untuk menampilkan hasil prediksi dari tulisan tangan seperti pada Gambar 9.



Gambar 9. Tampilan Utama

Aplikasi bekerja dengan cara pengguna menulis pada tempat yang disediakan kemudian menekan tombol "ok" untuk melakukan pengenalan huruf. Setelah mendapatkan hasilnya pengguna dapat menghapus dengan cara menekan tombol "hapus". Hasil pengenalan akan tampil pada kotak dibawah.

Pengenalan huruf pada aplikasi menggunakan model dari hasil pelatihan sebelumnya yang telah dikonversi ke format ".tflite". penggunaan model yang telah dikonversi tadi akan dimuat pada android menggunakan interpreter dari tensorflowlite; Masukan citra pada model menggunakan "ImageProcessor" dari tensorflowlite yang dilakukan dengan cara menyimpan hasil tulisan tangan pada canvas ke dalam BitMap kemudian memasukannya pada "ImagePrcessor" untuk kemudian diolah pada model untuk mendapatkan hasil prediksi dari gambar yang ada.

4.4. Hasil Pengujian

Pengujian dilakukan dengan memilih 45 responden sebagai responden untuk melakukan pengujian aplikasi. Setiap responden akan menulis huruf aksara sasak sebanyak 5 kali untuk tiap hurufnya. Hasil dari pengujian dicatat untuk melihat berapa jumlah dari tulisan yang diprediksi benar serta salah. Catatan hasil pengujian akan digunakan untuk menghitung akurasi, presisi, dan *recall* dari pengujian yang dilakukan oleh responden. Hasil pengujian yang telah dilakukan dapat dilihat pada Tabel IX.

TABEL X. *HASIL PENGUJIAN TERHADAP JUMLAH* EPOCH

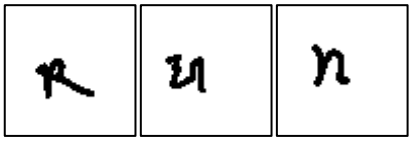
No	Nama	Akurasi (%)	Presisi (%)	Recall (%)
1	Responden 1	63.33	65.30	65.30
2	Responden 2	81.11	86.12	81.11
3	Responden 3	75.56	80.33	75.00
4	Responden 4	80.00	78.69	78.70
5	Responden 5	85.56	84.74	85.56
6	Responden 6	78.89	80.57	78.89
7	Responden 7	93.33	94.71	93.33
8	Responden 8	65.56	70.38	65.56
9	Responden 9	72.22	77.76	69.44
10	Responden 10	85.56	92.09	85.56
11	Responden 11	85.56	85.79	85.56
12	Responden 12	80.00	80.03	80.00
13	Responden 13	84.44	88.80	85.56
14	Responden 14	95.56	96.83	95.56
15	Responden 15	91.11	93.12	91.11
16	Responden 16	54.44	69.72	54.44
17	Responden 17	77.78	78.51	77.78
18	Responden 18	73.33	82.51	73.33
19	Responden 19	92.22	94.03	92.22
20	Responden 20	88.89	92.27	88.89
21	Responden 21	70.00	70.14	70.00
22	Responden 22	92.22	94.21	92.22
23	Responden 23	90.00	93.06	90.00
24	Responden 24	88.89	92.86	88.89
25	Responden 25	94.44	95.14	94.44
26	Responden 26	82.22	84.87	82.22
27	Responden 27	94.44	94.71	94.44
28	Responden 28	88.89	89.00	88.89
29	Responden 29	86.67	89.35	86.67
30	Responden 30	71.11	68.03	71.11
31	Responden 31	92.22	93.29	92.22
32	Responden 32	63.33	82.23	63.89
33	Responden 33	61.11	64.89	61.67
34	Responden 34	95.56	96.83	95.56
35	Responden 35	51.11	72.90	51.11
36	Responden 36	92.22	90.68	92.22
37	Responden 37	78.89	80.22	78.89
38	Responden 38	67.78	72.99	67.78
39	Responden 39	84.44	86.98	84.44
40	Responden 40	91.11	93.09	91.11
41	Responden 41	68.89	73.83	68.89
42	Responden 42	87.78	91.94	87.78
43	Responden 43	68.89	72.51	70.28
44	Responden 44	87.78	85.53	87.78
45	Responden 45	83.33	84.25	83.33

Berdasarkan Tabel X diperoleh rata akurasi, presisi, dan *recall* masing masing sebesar 80.84%, 84,13%, dan 80.86%. Akurasi tertinggi yang diperoleh sebesar 95.56% dengan presisi dan *recall* masing-masing sebesar 96.83% dan 95.56%. Akurasi terendah sebesar

51.11 % dengan presisi dan *recall* masing-masing 72.90% dan 51.11%.

Aplikasi yang dibuat mampu melakukan pengenalan terhadap aksara sasak dengan cukup baik. Waktu rata-rata yang dibutuhkan untuk melakukan prediksi dari tulisan tangan adalah 99.97 ms dengan waktu terlama 118 ms dan tercepat 65 ms. Meskipun akurasi yang didapatkan terendah mencapai 51.11% namun tetapi masih mendaoat rata-rata hasil yang baik dalam melakukan pengenalan terhadap aksara sasak.

Penyebab rendahnya akurasi dipengaruhi oleh tulisan responden pada saat pengujian yang tidak penuh pada tempat untuk menulis pada aplikasi seperti pada Gambar 10 serta gambar yang keluar dari tempat menulis seperti pada Gambar 11 mempengaruhi hasil prediksi dari aplikasi.



Gambar 10. Gambar huruf yang tidak memenuhi tempat gambar

Gambar 11. Gambar huruf yang keluar dari tempat gambar

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian yang sudah dilakukan, terdapat beberapa hal yang bisa penulis simpulkan antara lain sebagai berikut.:

1. Model yang dihasilkan pada penelitian ini menggunakan *input* citra dengan ukuran 64x64 *pixel*, *hidden layer* berjumlah 3 dengan *neuron* masing-masing sebanyak 128, kernel konvolusi 5x5, *learning rate* 0.1 dan normalisasi dengan fungsi aktivasi ReLU.
2. Waktu yang dibutuhkan untuk melakukan prediksi pada model dan hasil implementasi pada android memiliki perbedaan yang tidak terlalu jauh.
3. Semakin besar ukuran citra masukan semaikin lama waktu komputasi yang dibutuhkan, penambahan bobot *dropout* dapat mengurangi waktu komputasi.
4. Akurasi tertinggi yang didapatkan pada pengujian aplikasi 95.56% dengan presisi dan recall masing-masing sebesar 96.82% dan 95.56% sedangkan akurasi terendah d=54.44% dengan presesi dan recall masing-masing 69.72% dan 54.44%.
5. Rata-rata akurasi, presisi, dan recall yang diperoleh masing-masing 82.11%, 84.78%, dan 82.05% hal ini menunjukan model pada android memiliki performa yang tidak terlalu berbeda jauh dengan model aslinya.

5.2. Saran

Terdapat beberapa yang dapat penulis berikan apabila penelitian ini akan dikembangkan kembali antara lain sebagai berikut.

1. Menambah metode yang digunakan pada *pre processing* dan menambah scenario pengujian agar makin variatif sehingga dapat meningkatkan performa model.
2. Menggunakan dataset tulisan digital untukpenelitian selanjutnya agar tidak ada perbedaan ukuran ketebalan tulisan antara data tes pada android dan dataset pada saat training yang dapat mempengaruhi performa dari model.
3. Mengembangkan menjadi aplikasi untuk pembelajaran akasara sasak.

UCAPAN TERIMA KASIH

Terima kasih penulis ucapkan kepada rekan-rekan yang membantu penelitian ini pada penelitian ini serta pada responden untuk pengujian aplikasi atas kontribusinya pada penelitian kali ini.

DAFTAR PUSTAKA

[1] R. Yulianti, I. G. P. S. Wijaya, and F. Bimantoro, "Pengenalan Pola Tulisan Tangan Suku Kata Aksara Sasak Menggunakan Metode Moment Invariant dan Support Vector Machine," *J. Comput. Sci. Informatics Eng.*, vol. 3, no. 2, 2019.

[2] E. Dina Juliani U M, I. G. P. S. Wijaya, and F. Bimantoro, "Pengenalan Pola Tulisan Tangan Suku Kata Aksara Sasak Menggunakan Metode Integral Projection dan Neural Network," *J. Comput. Sci. Informatics Eng.*, vol. 3, no. 1, p. 19, 2019.

[3] A. N. Wangsaputra, Alfred Louis; Adipranata, Rudy; Tjondrowiguno, "Pengenalan Aksara Jawa dengan Menggunakan Metode Area Based Feature Extraction dan Support Vector Machine," *J. Infra Petra*, vol. 7, no. 1, pp. 140–146, 2019.

[4] A. Yudhana *et al.*, "Implementasi Algoritma Dijkstra Untuk Analisis Rute Transportasi Umum Transjogja Berbasis Android," *J. Sist. Inf. Bisnis 01(2019)*, vol. 01, pp. 32–38, 2019.

[5] T. C. Yunanto, K. Gunadi, A. N. Purbowo, P. S. Informatika, F. T. Industri, and U. K. Petra,

"Implementasi Convolutional Neural Network untuk Mengetahui Buah Tomat yang Matang pada Pohon Tomat Menggunakan Perangkat Android," *J. Infra*, vol. 8, no. 1, pp. 306–311, 2020.

[6] H. Darmanto, "Pengenalan Spesies Ikan Berdasarkan Kontur Otolith," *Joined J.*, vol. 2, no. 1, pp. 41–59, 2019.

[7] R. Rokhana, J. Priambodo, T. Karlita, I. M. G. Sunarya, and E. M. Yuniarno, "Convolutional Neural Network untuk Pendeteksian Patah Tulang Femur pada Citra Ultrasonik B – Mode," *JNTETI*, vol. 8, no. 1, 2019.

[8] V. Maha, P. Salawazo, D. Putra, J. Gea, F. Teknologi, and U. P. Indonesia, "Implementasi Metode Convolutional Neural Network (CNN) Pada Pengenalan Objek Video CCTV," *J. Mantik Penusa*, vol. 3, no. 1, pp. 74–79, 2019.

[9] A. N. Lorentius, Christopher Albert; Gunadi, Kartika; Tjondrowiguno, "Pengenalan Aksara Jawa dengan Menggunakan Metode Convolutional Neural Network," *J. Infra Petra*, vol. 7, no. 1, pp. 221–227, 2019.

[10] D. C. K. Putu Aryasuta Wicaksana, I Made Sudarma, "Pengenalan Pola Motif Kain Tenun Gringsing Menggunakan Metode Convolutional Neural Network Dengan Model Arsitektur," *J. SPEKTRUM*, vol. 6, no. 3, pp. 159–168, 2019.

[11] A. F. Saiful rahman, A. A. B, and S. D. Kurniawan, "Identifikasi Citra Daun Dengan Menggunakan Metode Deep Learning Convolutional Neural Network (Cnn)," *J. Tek. Elektro Uniba (JTE Uniba)*, vol. 4, no. 1, pp. 23–28, 2019.

[12] E. Gunawan, Verrell; Gunadi, Kartika; Setyati, "Identifikasi Buah-buahan Menggunakan Metode Convolutional Neural Network," *J. Infra Petra*, vol. 7, no. 1, pp. 33–38, 2019.

[13] S. Ilahiyah and A. Nilogiri, "Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network," *(Jurnal Sist. Teknol. Inf. Indones.*, vol. 3, no. 2, pp. 49–56, 2018.

[14] A. A. S. M. K. Maharani and F. Bimantoro, "Pengenalan Pola Tulisan Tangan Aksara Sasak Menggunakan Metode Linear Discriminant Analysis dan Jaringan Syaraf Tiruan Jenis Backpropagation," *J. Teknol. Informasi, Komputer, dan Apl. (JTIKA)*, vol. 2, no. 2, pp. 237–247, 2020.

[15] M. Fachriyan and D. Dharmayanti, "Pembangunan Aplikasi Pengenalan Objek Terdekat Untuk Penyandang Tunanetra Menggunakan Mlkit Dan Text To Voice Berbasis Android," *elibrary unikom*, 2019.