ISSN (Online) 2747-0563
Volume 1 Tahun 2020

Pengenalan Tulisan Tangan Huruf Hijaiyah Menggunakan Convolutional Neural Network Dengan Augmentasi Data

Sunu Ilham Pradika[1], Budi Nugroho[2*], Eva Yulia Puspaningrum[3]

[1] Informatika, Universitas Pembangunan Nasional "Veteran" Jawa Timur
[1]sunuilhamp@gmail.com
[1]evapuspaningrum.if@upnjatim.ac.id
*Corresponding author email: budinugroho.if@upnjatim.ac.id

***Abstrak*— Sistem pengenalan tulisan tangan huruf hijaiyah diperlukan untuk melakukan koreksi otomatis terhadap seseorang yang tengah belajar menulisnya. Dalam pengimplementasiannya terdapat beberapa tantangan seperti banyaknya bentuk variasi tulisan tangan huruf hijaiyah, pemilihan arsitektur yang tepat, dan banyak data pelatihan yang dibutuhkan agar sistem dapat memprediksi secara akurat. *Convolutional Neural Network (CNN)* merupakan salah satu algoritma *deep learning* yang efektif dalam mengolah citra yang dapat dilatih baik secara *supervised learning* maupun *unsupervised learning*. Model CNN dilatih menggunakan *dataset* Hijaiyah1SKFI. *Dataset* tersebut terdiri dari 2100 data dengan 30 kelas yakni huruf alif hingga ya yang ditulis oleh 4 orang berbeda dengan 80% digunakan sebagai data latih dan 20% adalah data tes. Dalam penelitian ini dilakukan optimisasi berupa augmentasi data karena data yang tidak banyak sehingga dengan data yang sedikit maka variasi data pelatihan akan bertambah. Arsitektur yang diusung di penelitian ini bernama SIP-Net mendapatkan akurasi pada data uji sebesar 99,7%.**

***Kata Kunci*— *Deep learning*, *Convolutional Neural Network*, Pengenalan Huruf Hijaiyah, Tulisan Tangan, *Supervised Learning*, Augmentasi Data.**

I. PENDAHULUAN

Huruf hijaiyah adalah huruf yang digunakan untuk membuat kata dalam bahasa arab yang berjumlah antara 28, 29, dan 30 [1]. Hal itu tergantung apakah huruf yang tidak asli diikutkan dalam hitungan huruf atau tidak. Huruf yang tidak asli misalnya lamalif sehingga berjumlah 29. Kemudian yang mengatakan 28 adalah alif dan hamzah dianggap sama ditambah dengan lamalif yang tidak asli. Sehingga terdapat 3 versi jumlah huruf hijaiyah. Mempelajari huruf hijaiyah adalah tahap pertama seseorang untuk bisa membaca Al Quran [2]. Al Quran sendiri dibuat dengan menggunakan Bahasa Arab [3].

Bahasa Arab merupakan bahasa dengan pengguna pada tahun 2019 sebesar 313 juta orang di dunia [4]. Bahasa Arab merupakan bahasa yang digunakan oleh sebagian besar umat Islam. Itu merupakan bahasa ke-6 terbanyak yang digunakan di seluruh dunia [5]. Sehingga mempelajari dasar dari Bahasa Arab yakni huruf hijaiyah menjadi langkah pertama yang harus dilakukan. Namun, dalam mempelajari sesuatu seringkali harus didampingi seorang pengajar. Di saat sekarang, dunia sedang mengalami pandemi *coronavirus disease* 2019 (Covid-19) [6]. WHO menganjurkan untuk menerapkan protokol kesehatan yang diantaranya adalah memakai masker, mencuci tangan, dan *physical distancing* [7]. Oleh karena itu mendatangkan seorang guru di saat pandemi seperti ini sangat beresiko tertular virus Covid-19.

Dari permasalahan di atas dibutuhkan sebuah sistem yang dapat mengenali tulisan tangan huruf hijaiyah. Hal tersebut diperlukan agar seseorang dapat belajar secara mandiri di rumah masing-masing namun tetap seperti diajari oleh pengajar karena apabila terdapat salah penulisan maka sistem akan memberitahukannya. Namun, untuk mencapai titik itu dibutuhkan sebuah model yang memiliki akurasi mendekati 100%. Penelitian tentang pengenalan huruf hijaiyah sudah pernah dilakukan oleh El-sawy et al. pada tahun 2017 menggunakan metode *Convolutional Neural Network (CNN)* mendapatkan akurasi sebesar 94,9% dengan kesalahan klasifikasi sebesar 5,1% pada *dataset Arabic Handwritten Characters Dataset (AHCD)* [8]. Selanjutnya, Younis et al. menggunakan metode yang sama dan pada tahun yang sama berhasil mendapatkan akurasi sebesar 94,7% pada AHCD dan 94,8% pada AIA9K *dataset* [9]. Selain itu, Altwaijry et al. juga melakukan penelitian menggunakan metode yang sama pada tahun 2020 berhasil mendapatkan akurasi sebesar 98% pada AHCD dan 88% pada *dataset* Hijja yang mereka buat sendiri [10]. Penelitian lain yang menggunakan CNN dengan akurasi 99,3% adalah yang dilakukan oleh Latif et al. [11] pada tahun 2018. Selain menggunakan CNN ada juga penelitian yang menggunakan metode *Multi Layer Perceptron (MLP)* seperti yang dilakukan oleh Ashiquzzaman et al. [12] pada tahun 2017 dengan akurasi sebesar 93,8% dan Das et al. [13] pada tahun 2006 dengan akurasi sebesar 94,93%. Tak hanya CNN dan MLP juga ada penelitian yang menggunakan metode *Time Delay Neural Network (TDNN)* yang dilakukan oleh Mars et al. [14] pada tahun 2016 dengan akurasi sebesar 98,5%.

Dari beberapa penelitian di atas, bahwa dalam implementasi pengenalan huruf hijaiyah menggunakan beberapa metode seperti CNN, MLP, dan TDNN dengan hasil urutan akurasi dari mulai yang terbaik yakni CNN, TDNN, dan MLP. CNN memiliki akurasi antara 94,7% hingga 99,3%, hal tersebut menunjukkan apabila metode CNN dibuat dengan baik maka dapat mencapai akurasi 99,3% selain faktor lainnya seperti *dataset*. Selanjutnya adalah TDNN yang memiliki akurasi sebesar 98,5% dan terakhir MLP mencapai akurasi 94,93%. Sehingga pada penelitian ini, dipilih metode CNN karena

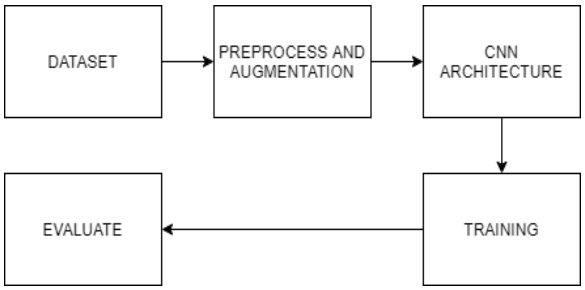
ISSN (Online) 2747-0563
Volume 1 Tahun 2020

memiliki akurasi tinggi dibandingkan dengan MLP dan TDNN.

Pada penelitian ini kami mengusulkan sebuah arsitektur yang dirancang oleh penulis dengan nama SIP-Net yang berbeda dari penelitian-penelitian yang sebelumnya. Selain itu, penelitian yang sebelumnya kebanyakan menggunakan *dataset* AHCD, dimana setelah dilakukan percobaan menggunakan data tersebut, dimensi citranya yang hanya 32×32 tidak cocok apabila digunakan menggunakan arsitektur SIP-Net. Sehingga pada penelitian ini menggunakan *dataset* yang dibuat 4 orang dengan nama Hijaiyah1SKFI. Dimensi citra dari datanya lebih besar dibandingkan dengan *dataset* AHCD dan CMATERDB yakni sebesar 664×373 dan ada yang sebesar 584×744.

II. METODOLOGI PENELITIAN

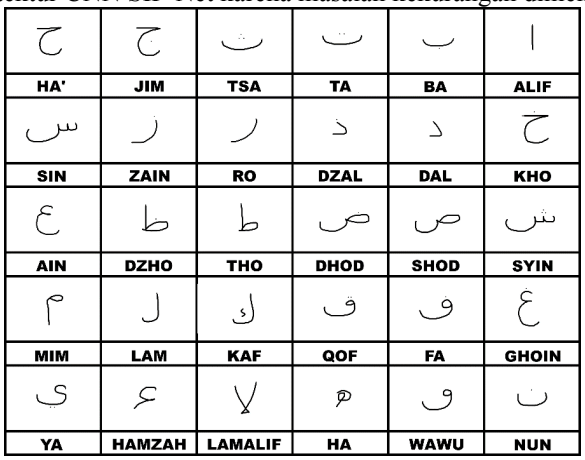
Metodologi yang dilakukan pada penelitian ini dapat diilustrasikan pada Gbr. 1. Memulai dengan menyiapkan *dataset*. Dilanjutkan dengan melakukan praproses dan augmentasi data. Kemudian membuat arsitektur CNN yang akan digunakan sebagai model. Lalu, melakukan pelatihan menggunakan *dataset* yang sudah dipraproses dan diaugmentasi. Terakhir, melakukan evaluasi model.



Gbr. 1 Metode penelitian.

A. Dataset

Dataset yang digunakan adalah tulisan tangan yang dibuat oleh 4 orang berjumlah sebesar 2100 data yang diberi nama Hijjaiyah1SKFI. Beberapa sampel *dataset* dapat dilihat pada Gbr. 2. Saat ini, data tersebut masih bersifat pribadi dan belum publik. Sayangnya, *dataset* AHCD tidak dapat diuji di arsitektur CNN SIP-Net karena masalah kekurangan dimensi.



Gbr. 2 Sampel *dataset*.

Datanya berformat .jpg sehingga tidak perlu dilakukan konversi ke citra dan bisa langsung dimuat ke dalam sistem pelatihan. Tidak seperti *dataset* AHCD yang berjumlah 16800 data dan *dataset* AIA9K yang berjumlah 8737 data, *dataset* Hijjaiyah1SKFI berjumlah lebih sedikit. Namun, jumlah data tersebut tidak terlalu jauh berbeda dengan *dataset* CMATERDB yang berjumlah 3000 data [12] [13] [15][16].

TABEL I
DESKRIPSI *DATASET*

	Karakter	Data Latih	Data Uji
1	Alif	56	14
2	Ba	56	14
3	Ta	56	14
4	Tsa	56	14
5	Jim	56	14
6	Ha’	56	14
7	Kho	56	14
8	Dal	56	14
9	Dzal	56	14
10	Ro	56	14
11	Zain	56	14
12	Sin	56	14
13	Syin	56	14
14	Shod	56	14
15	Dhod	56	14
16	Tho	56	14
17	Dzho	56	14
18	Ain	56	14
19	Ghoin	56	14
20	Fa	56	14
21	Qof	56	14
22	Kaf	56	14
23	Lam	56	14
24	Mim	56	14
25	Nun	56	14
26	Wawu	56	14
27	Ha	56	14
28	Lamalif	56	14
29	Hamzah	56	14
30	Ya	56	14
Total		1680	420

Dataset Hijjaiyah1SKFI terdiri dari 30 kelas yaitu alif, ba, ta, tsa, jim, ha’, kho, dal, dzal, ro, zain, sin, syin, shod, dhod, tho, dzho, ain, ghoin, fa, qof, kaf, lam, mim, nun, wawu, ha, lamalif, hamzah, dan ya seperti pada Tabel I. Masing-masing kelas terdiri dari 70 data. Data akan dibagi menjadi 80% dan 20% masing-masing untuk data latih dan data uji. Pembagian *dataset* tersebut mengikuti aturan Pareto [17]. Jadi, data latih sebanyak 1680 dan data uji sebanyak 420.

B. Preprocess and Augmentation

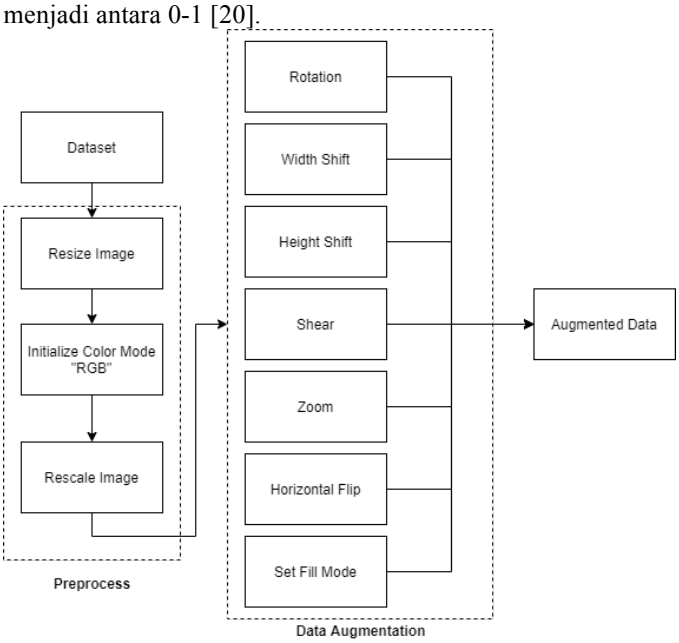
Preprocess and augmentation diilustrasikan pada Gbr. 3. Data akan dilakukan praproses berupa merubah ukuran citra dari 664×373 dan 584×744 menjadi 150×150 dan menetapkan mode warna *Red, Green, and Blue* (RGB) yang memiliki 3 *channel* [18]. Hal tersebut dilakukan walaupun

ISSN (Online) 2747-0563
Volume 1 Tahun 2020

warna yang dapat kita lihat adalah hitam dan putih maka mungkin akan terfikirkan bahwa akan menggunakan 1 *channel* yakni *grayscale* [19]. Namun, sebenarnya walaupun begitu, citra tersebut masih dapat dikategorikan mode warna RGB. Sehingga, dihasilkan sebuah *shape* dari setiap citra adalah $H \times W \times C$ untuk *height, weight,* dan *channel* yaitu $150 \times 150 \times 3$. Tidak lupa untuk melakukan *rescaling input* yang berupa *RGB coefficients* bernilai antara 0-255 dengan perhitungan berikut:

$$Output = input \times \frac{1}{255} \quad (1)$$

Rumus di atas digunakan untuk mengubah nilai tersebut menjadi antara 0-1 [20].

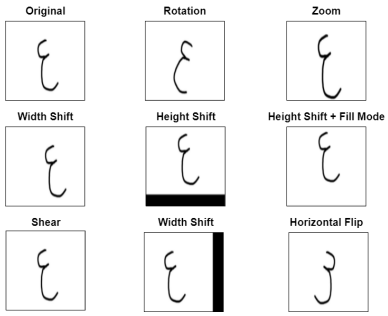


Gbr. 3 Praproses dan augmentasi data.

Dengan 2100 data jika dibandingkan dengan *dataset* yang sudah disebutkan di atas pada pembahasan *dataset* maka akan terlihat sedikit. Namun, saat ini sudah ditemukan sebuah cara untuk memperkaya variasi fitur menggunakan *data augmentation* [21]. Adapun beberapa teknik data augmentasi yang dilakukan yaitu:

1. Rotasi *(rotation),*
2. Pergeseran lebar *(width shift),*
3. Pergeseran tinggi *(height shift),*
4. Mencukur *(shear),*
5. Memperbesar *(zoom),*
6. Membalik secara horizontal *(horizontal flip),*

Mengisi area kosong *(filling)*. Biasanya dibutuhkan saat pergesaran baik lebar maupun tinggi untuk mengisi pixel baru yang terbuat. Hasil praproses dan augmentasi diilustrasikan pada Gbr. 4.

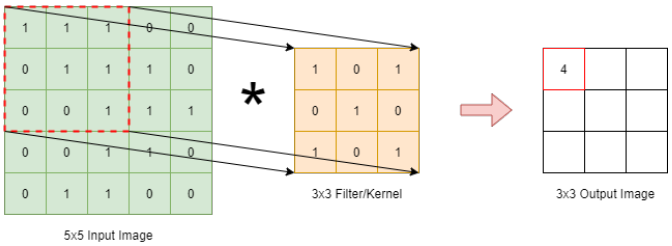


Gbr. 4 Praproses dan augmentasi data.

C. *CNN Architecture*

CNN adalah sebuah salah satu metode dalam *deep learning* yang populer [22][23]. CNN dapat digunakan untuk pelatihan baik secara *supervised learning* maupun *unsupervised learning* [24]. CNN adalah metode yang biasanya digunakan untuk mengklasifikasikan citra [22].

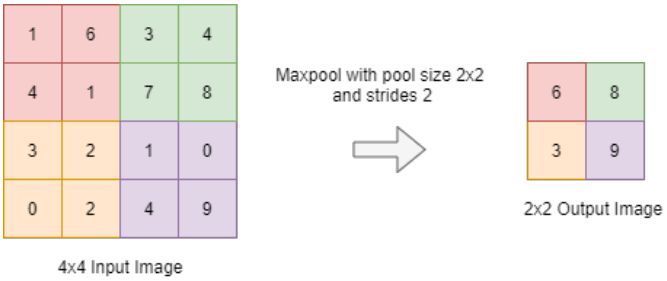
Dalam CNN, terdapat beberapa arsitektur yang terkenal di kalangan peneliti karena keunikan dan kemampuannya untuk memperoleh akurasi yang tinggi. Diantaranya adalah LeNet-5 yang dibuat oleh LeCun et al. [25], AlexNet yang dibuat oleh Krizhevsky et al. [26], *Network In Network (NN)* yang dibuat oleh Lin et al. [27], VGG-16 yang dibuat oleh Simonyan et al. [28], *Inception* yang dibuat oleh Szegedy et al. [29], dan ResNet yang dibuat oleh He et al. [30]. Masing-masing arsitektur CNN di atas memiliki keunikan. Namun, terdapat persamaan di hampir semua arsitektur tersebut yaitu semakin dalam sebuah arsitektur maka dimensi $H \times W$ akan semakin mengecil dan C akan semakin besar.



Gbr. 5 *Convolution layer.*

Seperti namanya, *convolution layer* memegang peranan penting di CNN. *Convolution layer* adalah *layer* yang berfungsi untuk mengekstrak fitur dari sebuah citra dengan *filter/kernel* tertentu yang bertipe *forward propagation* pada data latih dan *learnable parameters* seperti *kernels* dan *weights* diperbaharui berdasarkan nilai *loss* melalui *backpropagation* dengan optimisasi *gradient descent* [31]. Seperti pada Gbr. 5, citra masukan dengan ukuran 5x5 dilakukan *dot operation* antara *input* dan filter, setelah selesai pada bagian pertama maka akan digeser hingga seluruh bagian telah dihitung hingga menghasilkan sebuah *output* berupa citra berukuran 3x3.

ISSN (Online) 2747-0563
Volume 1 Tahun 2020



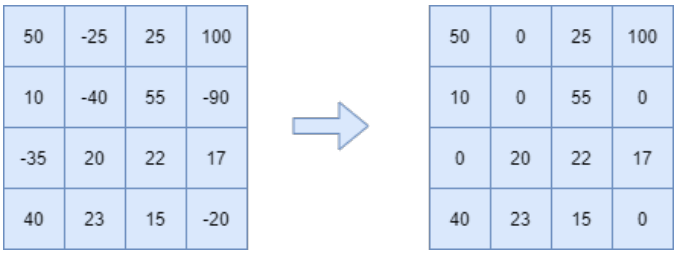
Gbr. 6 *Pooling layer.*

Pada Gbr. 6 adalah sebuah ilustrasi dari *pooling layer*. *Pooling layer* digunakan untuk mengurangi parameter dengan tetap mempertahankan imformasi penting yang merepresentasikan citra [32]. Dalam *pooling layer* terdapat beberapa tipe yakni *MaxPooling, AveragePooling, dan SumPooling* [33]. Tipe *pooling* yang digunakan pada penelitian ini adalah *MaxPooling*. Citra masukan akan dicek menggunakan *pooling layer* dengan size 2x2 dan akan mencari elemen terbesar.

Rectified Linear Unit (ReLU) digunakan untuk operasi non-linear [32]. Jadi, nilai keluaran dari operasi ReLU adalah nilai masukan itu sendiri atau 0. Rumus yang digunakan adalah sebagai berikut [33]:

$$f(x) = \max(0, x) \quad (2)$$

Rumus di atas dapat diilustrasikan seperti pada Gbr 7. Jadi setiap elemen yang bernilai minus maka akan menjadi 0.



Gbr. 7 Operasi ReLU.

Softmax adalah sebuah fungsi aktivasi yang digunakan pada perhitungan jaringan yang digunakan untuk menghitung distribusi sebuah vektor dari bilangan real [34]. *Softmax* menghasilkan sebuah luaran yang memiliki nilai antara 0 dan 1, dengan total semua kemungkinan sama dengan 1. Biasanya fungsi aktivasi *softmax* digunakan untuk model multi kelas dengan target mendapatkan kemungkinan tertinggi. *Softmax* digunakan pada *output layer* dari arsitektur *deep learning* seperti yang digunakan oleh A. Krizhevsky et al. [35] dan V. Badrinarayanan et al. [36]. Fungsi aktivasi *softmax* dihitung menggunakan rumus berikut [37]:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (3)$$

$\vec{z}$ = vektor masukan untuk fungsi *softmax*, terdiri dari (z0, …, zK)

z_i = semua nilai z_i dari vektor masukan untuk fungsi *softmax*, dan itu terdiri dari semua bilangan real, positif, nol, atau negative. Misalnya nilai vektor berupa (-0.62, 8.12, 2.53) yang merupakan distribusi kemungkinan yang tidak valid, oleh karena itu kenapa *softmax* dibutuhkan.

e^{z_i} = standar fungsi eksponensial diaplikasikan ke tiap elemen dari vektor masukan. Ini memberikan nilai positif lebih dari 0 yang akan sangat kecil apabila masukannya negatif dan akan sangat tinggi apabila masukannya positif.

K = banyaknya kelas.

$\sum_{j=1}^{K} e^{z_j}$ = normalisasi yang memastikan nilai keluaran jika ditotal menghasilkan nilai satu dan masing-masing nilai distribusi berada antara 0 dan 1, maka itulah distribusi kemungkinan yang valid.

Arsitektur yang digunakan pada penelitian ini diilustrasikan seperti pada Gbr. 8 disebut dengan nama SIP-Net juga memiliki ciri khas yang sama dengan arsitektur yang telah disebutkan pada paragraf sebelumnya. Arsitektur tersebut memiliki 6.826.846 parameter. Dapat dilihat pada saat *input* maka *shape* dari citranya adalah 150×150×3 kemudian masuk ke *convolution layer* pertama dengan perhitungan menggunakan rumus sebagai berikut [38]:

$$Output = \frac{(\mathrm{I} - \mathrm{F} + 2\times\mathrm{P})}{\mathrm{S}} + 1 \quad (4)$$

I = *Input* dari citra berupa tinggi (H) atau lebar (W),
F = *Filter* yang digunakan ($F\times F$) bernilai 3,
P = *Padding* yang digunakan bernilai 0,
S = *Strides* yang digunakan bernilai 1

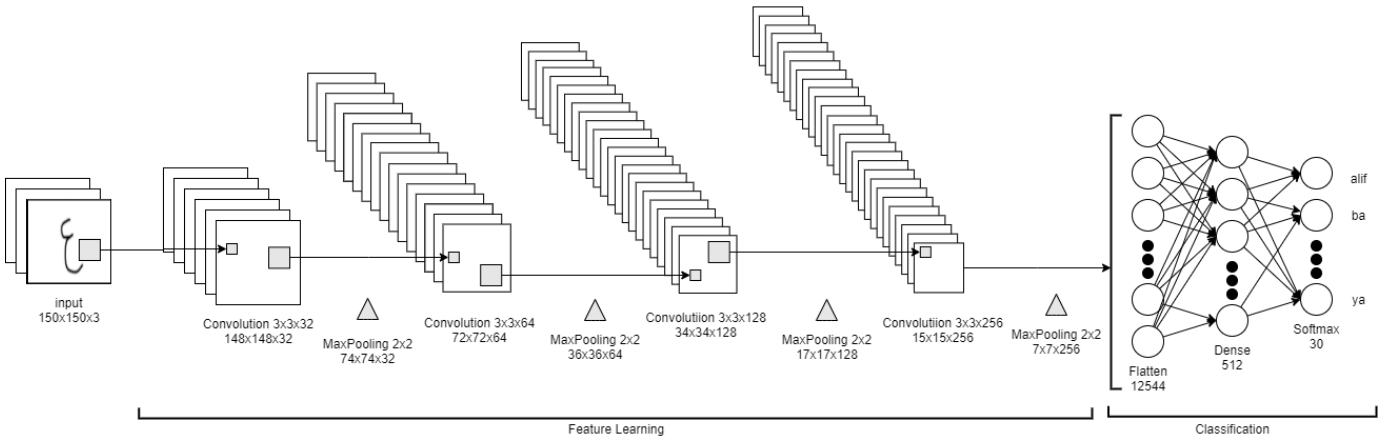
Sehingga setelah dihitung menjadi 148×148×32 lalu masuk ke dalam *pooling layer* pertama dengan perhitungan menggunakan rumus sebagai berikut [39]:

$$Output = \frac{\mathrm{I} - \mathrm{F} + 1}{\mathrm{S}} \quad (5)$$

I = *Input* dari citra berupa tinggi (H) atau lebar (W),
F = *Pool size* yang digunakan ($F\times F$) bernilai 2,
S = *Strides* yang digunakan di penelitian ini bernilai 2

Sehingga setelah dihitung menjadi 74×74×32 karena pembulatan ke atas, begitu seterusnya hingga *convolution layer* terakhir sebelum *Fully Connected Layer (FCL)* menjadi 7×7×256. Terlihat bahwa $H\times W\times C$ dimana H, W mengecil sampai 7 yang asalnya 150 dan C membesar sampai 256 yang awalnya 3. Setiap *convolution layer* diberikan aktivasi ReLU untuk mengurangi resiko penurunan peforma arsitektur karena hasil perhitungan negatif.

ISSN (Online) 2747-0563
Volume 1 Tahun 2020

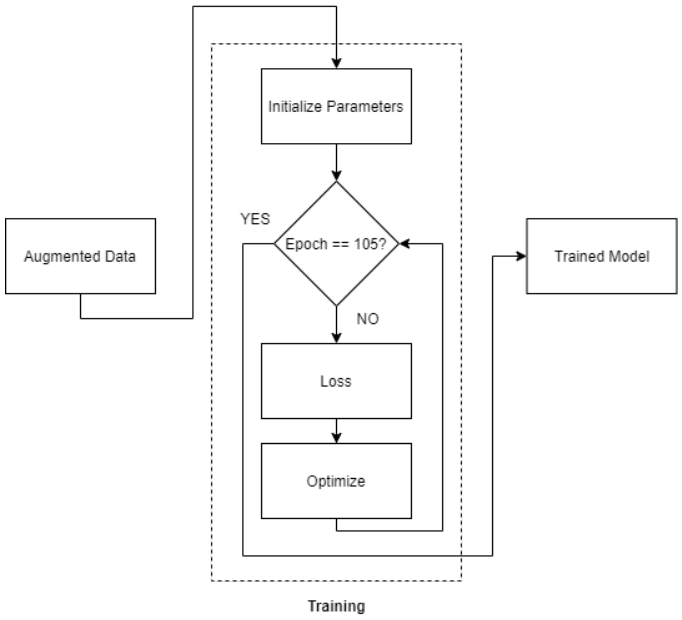


Gbr. 8 Arsitektur CNN SIP-Net.

Selanjutnya adalah FCL yang diletakkan pada bagian akhir CNN setelah semua *convolution layer* yang digunakan untuk mengklasifikasi berdasarkan fitur yang telah dipelajari pada *convolution layer* [40]. Pada layer pertama fitur akan dilakukan *flatten* dan didapatkan 12544 parameter yang didapatkan dari perkalian $H \times W \times C$ pada *pooling layer* terakhir. Selanjutnya terdapat *hidden layer* berjumlah 512 *neurons* dengan aktivasi ReLU. Terakhir, *output layer* dengan 30 *neurons* yang sejumlah kelas prediksi dari huruf hijaiyah dengan menggunakan aktivasi *softmax*.

D. Training

Proses *training* diilustrasikan pada Gbr. 9. Dimulai dengan menginisialisasi parameter-parameter seperti *epoch* sebanyak 105, *steps per epoch* berdasarkan banyak data uji dan pelatihan dan *batch size* sebesar 56, *optimizer* menggunakan Adam, fungsi *loss* menggunakan SCCE, metrik berupa akurasi, data yang akan digunakan untuk validasi, dan kebutuhan lain disesuaikan masing-masing peneliti.



Gbr. 9 Pelatihan model.

Sparse Categorical Cross Entropy (SCCE) sebenarnya sama saja dengan *Categorical Cross Entropy (CCE)* yang merupakan sebuah fungsi untuk menghitung *error/loss* dalam pelatihan *deep learning* [41]. Keduanya baik SCCE dan CCE dapat dihitung menggunakan rumus yang sama berikut ini [42]:

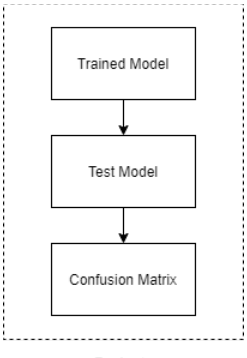
$$L(\theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i) \quad (6)$$

Lalu, yang membedakan antara keduanya adalah CCE digunakan apabila target yang diinginkan adalah *One-Hot Encoded (OHE)* dan SCCE digunakan apabila targetnya adalah *integer* [43]. Target berupa OHE misalnya terdapat 3 kelas maka [1, 0, 0], [0, 1, 0], dan [0, 0, 1]. Namun apabila berupa *integer* maka [1, 2, 3].

Adam adalah algoritma optimisasi kombinasi *RMSprop* dan *Stochastic Gradient Descent (SGD)* dengan momentum [44]. Metode ini banyak digunakan karena mudah untuk diterapkan, efisien secara komputasi sehingga tidak membutuhkan banyak memori, dan cocok untuk data/parameter yang besar [45]. Pada penelitian ini digunakan *learning rate* (lr= 0,001).

E. Evaluate

Proses *evaluate* diilustrasikan pada Gbr. 10. Menggunakan *confusion matrix* untuk melihat sebaran hasil prediksi dibandingkan dengan label sebenarnya dari 30 kelas mulai dari alif hingga ya.



Gbr. 10 Evaluasi model.

ISSN (Online) 2747-0563
Volume 1 Tahun 2020

Confusion matrix digunakan untuk melihat performa dari suatu model yang telah dibuat yaitu *accuracy, precision, recall,* dan *f1 score* [46]. Terdapat beberapa kategori dari kasus yang mungkin terjadi [47]:

- *True Positive (TP)* : kasus dimana huruf hijaiyah diprediksi sesuai (Positif) kelasnya, dan sebenarnya memang sesuai *(True)* kelasnya.
- *True Negative (TN)* : kasus dimana huruf hijaiyah diprediksi tidak sesuai (Negatif) kelasnya dan sebenarnya memang tidak sesuai *(True)* kelasnya.
- *False Positive (FP)* : kasus dimana huruf hijaiyah diprediksi sesuai (Positif) kelasnya, dan ternyata tidak sesuai *(False)* kelasnya.
- *False Negative (FN)* : kasus dimana huruf hijaiyah diprediksi tidak sesuai (Negatif) kelasnya, dan ternyata sesuai *(True)* kelasnya.

Accuracy adalah rasio dari prediksi benar dari keseluruhan data. Akurasi menjelaskan "Berapa persen huruf hijaiyah yang diprediksi sesuai kelasnya dari keseluruhan data".

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (7)$$

Precision adalah rasio prediksi yang benar dibandingkan dengan keseluruhan hasil yang diprediksi positif. *Precision* menjelaskan "Berapa persen huruf hijaiyah yang memang sesuai kelasnya dari keseluruhan huruf hijiayah yang diprediksi sesuai kelasnya"

$$Precision = \frac{TP}{TP + FP} \quad (8)$$

Recall adalah rasio dari prediksi yang benar dibandingkan dengan kesuluruhan data yang memang benar. *Recall* menjelaskan "Berapa persen huruf hijaiyah yang diprediksi sesuai kelasnya dibandingkan kesuluruhan huruf hijaiyah yang memang sesuai kelasnya".

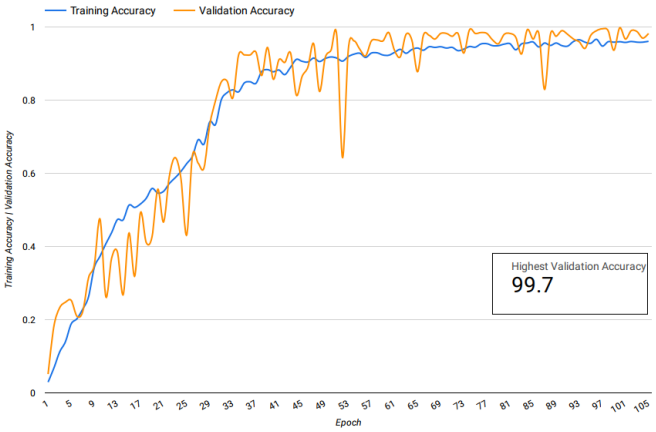
$$Recall = \frac{TP}{TP + FN} \quad (9)$$

F1 Score adalah rasio perbandingan rata-rata *precision* dan *recall* yang dibobotkan.

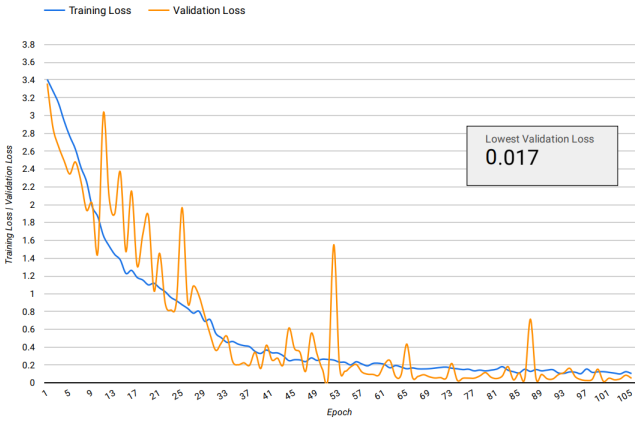
$$F1\ Score = \frac{2 \times (Recall \times Precision)}{(Recall + Precision)} \quad (10)$$

III. Hasil dan Pembahasan

Lingkungan percobaan yang digunakan adalah menggunakan sistem operasi Windows 10 64-bit dengan bahasa pemrograman Python dan GPU NVIDIA GeForce MX 150 menggunakan *deep learning framework* yakni *Tensorflow.Keras*. Beberapa hasil dari penelitian ini diantaranya:



Gbr. 11 Metrik akurasi.



Gbr. 12 Metrik *loss*.

Dapat dilihat pada Gbr. 11 dan Gbr. 12 yang merupakan metrik pada saat pelatihan model menggunakan arsitektur SIP-Net. Hasilnya cukup menjanjikan dimana dapat dilihat dari grafiknya tidak terjadi *overfitting*. Validasi baik akurasi dan *loss* terjadi sebuah lonjakan naik dan turun atau *spike* pada 89 *epoch* pertama, selanjutnya cukup stabil.

Metrik yang fluktuaktif itu disebabkan oleh augmentasi data, oleh karenanya sampel pelatihan akan teracak setiap *epoch* nya. Akurasi validasi tertinggi terjadi pada *epoch* ke-100 mencapai 99,7% bersamaan dengan akurasi pelatihan sebesar 95,9%. Lalu, untuk *loss* nya mencapai titik terendah pada *epoch* ke-100 bersamaan dengan akurasi validasi yakni sebesar 1,7% untuk *loss* validasi dan 12,5% untuk *loss* pelatihan.

Tabel II
Confusion Matrix

ISSN (Online) 2747-0563
Volume 1 Tahun 2020

	Karakter	Precision	Recall	F1 Score
1	Alif	1,00	1,00	1,00
2	Ba	1,00	1,00	1,00
3	Ta	1,00	1,00	1,00
4	Tsa	1,00	1,00	1,00
5	Jim	1,00	1,00	1,00
6	Ha'	1,00	1,00	1,00
7	Kho	1,00	1,00	1,00
8	Dal	1,00	1,00	1,00
9	Dzal	1,00	1,00	1,00
10	Ro	0,93	1,00	0,97
11	Zain	1,00	1,00	1,00
12	Sin	1,00	1,00	1,00
13	Syin	1,00	1,00	1,00
14	Shod	1,00	1,00	1,00
15	Dhod	1,00	1,00	1,00
16	Tho	1,00	1,00	1,00
17	Dzho	1,00	1,00	1,00
18	Ain	1,00	1,00	1,00
19	Ghoin	1,00	1,00	1,00
20	Fa	1,00	1,00	1,00
21	Qof	1,00	1,00	1,00
22	Kaf	1,00	1,00	1,00
23	Lam	1,00	1,00	1,00
24	Mim	1,00	1,00	1,00
25	Nun	1,00	1,00	1,00
26	Wawu	1,00	1,00	1,00
27	Ha	1,00	1,00	1,00
28	Lamalif	1,00	1,00	1,00
29	Hamzah	1,00	1,00	1,00
30	Ya	1,00	0,93	0,96
Akurasi pelatihan		**0,959**		
Akurasi pengujian		**1,00 (0,997)**		
Macro AVG		1,00	1,00	1,00
Weighted AVG		1,00	1,00	1,00

Hasil evaluasi menggunakan *confusion matrix* dapat dilihat pada Tabel II. Akurasi yang didapatkan hampir 100% tepatnya 99,7% pada data uji yang digunakan sebagai validasi. Hanya saja terdapat kekurangan pada huruf ro dan zain. Ro dengan kekurangan pada *precision* dan *f1 score*, kemudian zain pada *recall* dan *f1 score*. Hal tersebut terjadi karena memang untuk penulisan ro dan zain memiliki kesamaan bentuk dan hanya dibedakan dengan penanda sebuah titik yang kecil pada huruf zain. Selain itu, huruf ya yang juga sulit untuk dituliskan karena bentuknya yang sedikit rumit. Sehingga, dalam ekstraksi fitur bergantung pada penulis apakah dapat menuliskannya dengan baik.

IV. KESIMPULAN

Pengenalan tulisan tangan huruf hijaiyah bukanlah hal yang baru oleh akademisi. Pada penelitian ini berhasil dibuat sebuah sistem yang dapat mengenali huruf hijaiyah sehingga mampu memprediksi hasil tulisan tangan huruf hijaiyah seseorang. Tidak hanya itu, juga mendapatkan akurasi yang sangat tinggi untuk data latih sebesar 95,9% dan untuk data uji sebesar 99,7%. Pengembangan dapat dilakukan dengan cara memperbanyak data pelatihan agar tidak memerlukan augmentasi data, sehingga data yang dimiliki benar-benar dari orang yang nyata. Selain itu, juga dapat melakukan *tuning* pada *hyperparameter* untuk mendapatkan akurasi maksimal sebesar 100%.

UCAPAN TERIMA KASIH

Kami sampaikan terima kasih kepada seluruh pihak yang telah memberi dukungan dalam penyelesaian penelitian ini baik berupa semangat maupun pengetahuan.

REFERENSI

[1] T. M. Iqbal, "Huruf Hijaiyah: 30 Huruf Arab yang Luar Biasa [PENJELASAN LENGKAP]," 2020. [Online], https://hasana.id/huruf-hijaiyah/, tanggal akses: 04 Oktober 2020.

[2] R. M. Fauzi, Adiwijaya, and W. Maharani, "The recognition of Hijaiyah letter pronunciation using mel frequency cepstral coefficients and Hidden Markov Model," *Adv. Sci. Lett.*, 2016, doi: 10.1166/asl.2016.7769.

[3] R. Dharmawati and H. Destiana, "Interactive Animation Design of Hijaiyah Letters in Early Age Children at Al-Hidayah Kindergarten Bekasi," *SinkrOn*, 2019, doi: 10.33395/sinkron.v3i2.10033.

[4] D. Doochin, "How Many People Speak Arabic Around The World, And Where?," 2019. [Online], https://www.babbel.com/en/magazine/how-many-people-speak-arabic, tanggal akses: 05 Oktober 2020.

[5] I. Ghosh, "Ranked: The 100 Most Spoken Languages Around the World," 2020. [Online], https://www.visualcapitalist.com/100-most-spoken-languages/, tanggal akses: 05 Oktober 2020.

[6] WHO, "Considerations for school-related public health measures in the context of Annex to Considerations in adjusting public health and social measures in the context of COVID-19," no. May, pp. 1–6, 2020.

[7] WHO, "Coronavirus," 2020. [Online], https://www.who.int/health-topics/coronavirus#tab=tab_2, tanggal akses: 05 Oktober 2020.

[8] A. El-sawy, M. Loey, and H. El-Bakry, "Arabic Handwritten Characters Recognition using Convolutional Neural Network," *WSEAS Trans. Comput. Res.*, 2017.

[9] K. Younis and A. Khateeb, "Arabic Hand-Written Character Recognition Based on Deep Convolutional Neural Networks," *Jordanian J. Comput. Inf. Technol.*, vol. 3, no. 3, p. 186, 2017, doi: 10.5455/jjcit.71-1498142206.

[10] N. Altwaijry and I. Al-Turaiki, "Arabic handwriting recognition system using convolutional neural network," *Neural Comput. Appl.*, vol. 8, 2020, doi: 10.1007/s00521-020-05070-8.

[11] G. Latif, J. Alghazo, L. Alzubaidi, M. M. Naseer, and Y. Alghazo, "Deep Convolutional Neural Network for Recognition of Unified Multi-Language Handwritten Numerals," *2018 IEEE 2nd Int. Work. Arab. Deriv. Scr. Anal. Recognit.*, pp. 90–95, 2018, doi: 10.1109/ASAR.2018.8480289.

[12] A. Ashiquzzaman and A. K. Tushar, "Handwritten Arabic numeral recognition using deep learning neural networks," in *2017 IEEE International Conference on Imaging, Vision and Pattern Recognition, icIVPR 2017*, 2017, doi: 10.1109/ICIVPR.2017.7890866.

[13] N. Das, A. F. Mollah, S. Saha, and S. S. Haque, "Handwritten Arabic Numeral Recognition using a Multi Layer Perceptron Computer Science and Engineering Department , Computer Science and Engineering Department , Corresponding Author ' s email : nibs_breath@yahoo.com," *Inf. Syst.*, pp. 200–203, 2006.

[14] A. Mars and G. Antoniadis, "Arabic Online Handwriting Recognition Using Neural Network," *Int. J. Artif. Intell. Appl.*, vol. 7, no. 5, pp. 51–59, 2016, doi: 10.5121/ijaia.2016.7504.

[15] A. A. Alani, "Arabic handwritten digit recognition based on restricted Boltzmann machine and convolutional neural networks," *Inf.*, 2017, doi: 10.3390/info8040142.

[16] A. Ashiquzzaman, A. K. Tushar, A. Rahman, and F. Mohsin, "An efficient recognition method for handwritten arabic numerals using CNN with data augmentation and dropout," in *Advances in Intelligent Systems and Computing*, 2019, doi: 10.1007/978-981-13-1402-5_23.

[17] R. Dunford, Q. Su, E. Tamang, A. Wintour, and Project, "The Pareto Principle Puzzle," *Plymouth Student Sci.*, vol. 7, no. 1, pp. 140–148, 2014.

[18] J. Park, E. S. Jang, and J. W. Chong, "Demosaicing method for digital cameras with white-RGB color filter array," *ETRI J.*, vol. 38, no. 1, pp. 164–173, 2016, doi: 10.4218/etrij.16.0114.1371.
[19] [19]T. Wu and A. Toet, "Color-to-grayscale conversion through weighted multiresolution channel fusion," *J. Electron. Imaging*, vol. 23, no. 4, p. 043004, 2014, doi: 10.1117/1.jei.23.4.043004.
[20] F. Chollet, "Building powerful image classification models using very little data," 2016. [Online], https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html, tanggal akses: 07 Oktober 2020.
[21] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 International Interdisciplinary PhD Workshop, IIPhDW 2018*, 2018, doi: 10.1109/IIPHDW.2018.8388338.
[22] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, 2018, doi: 10.1109/ICEngTechnol.2017.8308186.
[23] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, 2019, doi: 10.1016/j.neucom.2018.09.038.
[24] J. Guérin, O. Gibaru, S. Thiery, and E. Nyiri, "CNN Features are also Great at Unsupervised Classification," pp. 83–95, 2018, doi: 10.5121/csit.2018.80308.
[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.
[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Handbook of approximation algorithms and metaheuristics," *Handb. Approx. Algorithms Metaheuristics*, pp. 1–1432, 2007, doi: 10.1201/9781420010749.
[27] M. Lin, Q. Chen, and S. Yan, "Network in network," *2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc.*, pp. 1–10, 2014.
[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.
[29] C. Szegedy *et al.*, "Going deeper with convolutions," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, doi: 10.1109/CVPR.2016.90.
[31] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging*, vol. 9, no. 4, pp. 611–629, 2018, doi: 10.1007/s13244-018-0639-9.
[32] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 120–147, 2018, doi: 10.1016/j.isprsjprs.2017.11.021.
[33] Prabhu, "Understanding of Convolutional Neural Network (CNN) — Deep Learning," 2018. [Online], https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148, tanggal akses: 09 Oktober 2020.
[34] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," pp. 1–20, 2018.
[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, 2017, doi: 10.1145/3065386.
[36] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017, doi: 10.1109/TPAMI.2016.2644615.
[37] T. Wood, "What is the Softmax Function?," 2020. [Online], https://deepai.org/machine-learning-glossary-and-terms/softmax-layer, tanggal akses: 10 Oktober 2020.
[38] V. Kohir, "Calculating Output dimensions in a CNN for Convolution and Pooling Layers with KERAS," 2020. [Online], https://medium.com/@kvirajdatt/calculating-output-dimensions-in-a-cnn-for-convolution-and-pooling-layers-with-keras-682960c73870, tanggal akses: 09 Oktober 2020.
[39] Keras.io, "MaxPooling2D layer," 2020. [Online], https://keras.io/api/layers/pooling_layers/max_pooling2d/, tanggal akses: 09 Oktober 2020.
[40] S. H. S. Basha, S. R. Dubey, V. Pulabaigari, and S. Mukherjee, "Impact of fully connected layers on performance of convolutional neural networks for image classification," *Neurocomputing*, vol. 378, pp. 112–119, 2020, doi: 10.1016/j.neucom.2019.10.008.
[41] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, no. NeurIPS, pp. 8778–8788, 2018.
[42] T. Jethwani, "Difference Between Categorical and Sparse Categorical Cross Entropy Loss Function," 2020. [Online] https://leakyrelu.com/2020/01/01/difference-between-categorical-and-sparse-categorical-cross-entropy-loss-function/, tanggal akses: 10 Oktober 2020.
[43] Chris, "How to use sparse categorical crossentropy in Keras?," 2019. [Online], https://www.machinecurve.com/index.php/2019/10/06/how-to-use-sparse-categorical-crossentropy-in-keras/#categorical-crossentropy, tanggal akses: 10 Oktober 2020.
[44] V. Bushaev, "Adam — latest trends in deep learning optimization.," 2018. [Online], https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c, tanggal akses: 10 Oktober 2020.
[45] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.
[46] S. Ghoneim, "Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on?," 2019. [Online], https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124, tanggal akses: 10 Oktober 2020.
[47] R. Arthana, "Mengenal Accuracy, Precision, Recall dan Specificity serta yang diprioritaskan dalam Machine Learning," 2019. [Online], https://medium.com/@rey1024/mengenal-accuracy-precission-recall-dan-specificity-serta-yang-diprioritaskan-b79ff4d77de8, tanggal akses: 10 Oktober 2020.