MIND (Multimedia Artificial Intelligent Networking Database) Journal

MIND Journal | ISSN (p): 2528-0015 | ISSN (e): 2528-0902 | Vol. 8 | No. 2 | Halaman 158 - 172
DOI : https://doi.org/10.26760/mindjournal.v8i2.158-172 Desember 2023

Ornamental Plants Classification using Integration of Convolution with Capsule Network

MOHAMMAD FATONI[1], ERNASTUTI[2]

[1]Program Studi Magister Manajemen Sistem Informasi, Universitas Gunadarma
[2]Program Studi Doktor Teknologi Informasi, Universitas Gunadarma
Email : fatoni.hello@gmail.com

Received 26 Juni 2023 **|** *Revised* 14 Agustus 2023 **|** *Accepted* 21 September 2023

ABSTRAK

Klasifikasi tanaman hias bertujuan untuk mempermudah media sosial tanaman hias dalam mengategorikan citra, sehingga sistem dapat merekomendasikan konten sesuai dengan preferensi pengguna. Pengguna berpotensi merasa cepat bosan apabila konten hanya ditampilkan secara acak. Penelitian ini melakukan *Integration of Convolution with Capsule Network* (ICCN) dengan menggabungkan beberapa lapisan *strided convolution* dan *Capsule Network* (CapsNet) untuk menghasilkan model klasifikasi yang memiliki komputasi lebih rendah dibandingkan original CapsNet dan mampu mengatasi permasalahan *invariant of translation* pada *Convolutional Neural Network* (CNN). Sebanyak 3 lapisan *convolution* dengan *kernel* berukuran 3x3 dan *stride* 2 ditambahkan pada CapsNet untuk membantu mengekstraksi citra dan mengurangi jumlah parameter yang dilatih. Hasil penelitian menunjukkan ICCN yang diusulkan memiliki jumlah parameter 2 kali lebih sedikit daripada original CapsNet dan memiliki akurasi lebih tinggi dibandingkan dengan CNN yaitu sebesar 95% sementara CNN berakurasi 93%.

***Kata kunci**: tanaman hias, klasifikasi citra, cnn, capsnet, integrasi*

ABSTRACT

The aim of ornamental plant classification is to assist ornamental plant social media in categorizing images, so the system is able to recommend content based on user preferences. Showing content randomly can lead to user boredom. This research implements Integration of Convolution with Capsule Network (ICCN) by combining several layers of strided convolution with Capsule Network (CapsNet) to create a classification model that has lower computation compared to the original CapsNet and able to address the issue of invariant of translation in Convolutional Neural Network (CNN). There are 3 convolutional layers with 3x3 kernel and stride of 2 added to CapsNet to assist in image extraction and reduce the number of trainable parameters. The research results showed that the proposed ICCN has 2 times fewer trainable parameters than the original CapsNet and achieves higher accuracy than CNN, with 95% accuracy, while CNN has an accuracy of 93%.

***Keywords**: ornamental plants, image classification, cnn, capsnet, integration*

1. PENDAHULUAN

Budidaya tanaman hias menjadi salah satu kegiatan yang ramai diperbincangkan di berbagai media sosial selama pandemi covid-19 di Indonesia **(Setyawan, 2022)**. Pengguna banyak melakukan pengunggahan citra potret koleksi tanaman hias mereka. Media sosial umum maupun media sosial khusus seperti media sosial tanaman hias perlu mengklasifikasi jenis konten yang diunggah pengguna agar dapat meningkatkan pemahaman sistem mengenai konten **(Ali, 2023)**. Daya tarik pengguna terhadap konten yang ditampilkan akan berpotensi meningkat apabila sistem menampilkan konten yang sesuai dengan preferensi pengguna **(Eg, 2023)**. Konten yang hanya ditampilkan secara acak berpotensi membuat pengguna merasa lebih cepat bosan **(Hutmacher, 2023)**. Penggolongan konten merupakan permasalahan klasifikasi yang dapat diselesaikan dengan algoritma klasifikasi. Algoritma klasifikasi perlu dipilih sesuai dengan kebutuhan klasifikasi karena setiap algoritma memiliki kelebihan dan kekurangan **(Wang, 2021)**. Kekurangan dari sebuah algoritma dapat diatasi dengan melakukan pengintegrasian dengan algoritma lain. Salah satu contoh pengintegrasian algoritma yaitu pengintegrasian antara algoritma *Capsule Network* (CapsNet) dengan algoritma *convolution*. Pengintegrasian antara CapsNet dengan *convolution* dinamakan *Integration of Convolution with Capsule Network* (ICCN). ICCN dilakukan untuk mengatasi permasalahan komputasi pada CapsNet yang terlalu besar dalam menghasilkan model klasifikasi citra. CapsNet adalah sebuah algoritma yang menggunakan kapsul untuk mendapatkan fitur-fitur pada citra serta hubungan spasialnya **(Sabour, 2017)** sehingga dapat menghasilkan model yang tidak *invariant of translation*. *Invariant of translation* merupakan ketidakmampuan model dalam mengidentifikasi posisi suatu objek ke objek lain dengan baik **(Biscione, 2021).** Model *invariant of translation* dapat terbentuk pada salah satu algoritma citra populer bernama *Convolutional Neural Network* (CNN) **(Yu, 2023)** apabila arsitektur CNN memiliki lapisan yang kompleks dan dilatih dengan data yang telah diaugmentasi **(Kauderer-Abrams, 2017)**. Algoritma CapsNet menggunakan *dynamic routing* dan *reconstruction regularization* sehingga model yang dihasilkan tidak *invariant of translation* **(Mazzia, 2021)**, namun CapsNet cenderung membutuhkan waktu pelatihan yang lebih lama daripada CNN dikarenakan jumlah *kernel* yang besar dan penggunaan *dynamic routing* yang membutuhkan komputasi yang lebih tinggi. Jumlah parameter yang dilatih menjadi sangat besar jika CapsNet digunakan untuk mengidentifikasi citra yang berukuran besar **(Yao, 2021).** Pengimplementasian ICCN memungkinkan untuk menghasilkan model yang tidak *invariant of translation* dan memiliki komputasi yang lebih rendah dibandingkan dengan original CapsNet. Pengintegrasian dengan *convolution* dapat membantu dalam mengekstraksi fitur dan mengurangi jumlah parameter yang dilatih, kemudian fitur citra dari hasil *convolution* dapat digunakan pada CapsNet sebagai *input* untuk diproses lebih lanjut.

Studi terkait pengintegrasian algoritma *convolution* dengan algoritma CapsNet telah dilakukan oleh peneliti terdahulu. Penelitian yang dilakukan oleh Sharma (2023) menggunakan *convolution* CapsNet untuk mendeteksi covid-19 pada citra X-Ray rontgen torak. Bagian *convolution* yang diintegrasikan terdiri dari 4 lapisan *convolution* yang memiliki filter 16, 32, 64, 128 secara berurutan. Setiap lapisan *convolution* disandingkan dengan lapisan *max pooling* **(Sharma, 2023)**. Penelitian yang dilakukan oleh Tiwari (2021) melakukan pengintegrasian arsitektur VGG16 dengan CapsNet. VGG16 tersusun atas 13 lapisan *convolution* yang disandingkan dengan lapisan *max pooling* **(Tiwari, 2021)**. Penelitian yang dilakukan oleh Ans Mittal (2020) menggunakan algoritma *convolution* dan algoritma CapsNet untuk mengklasifikasi pneumonia pada citra X-ray rongga dada tubuh manusia. Pengintegrasian *convolution* dan CapsNet dilakukan dengan berbagai cara seperti menyusun secara *sequential* dan *ensemble*. Lapisan *convolution* juga disandingkan dengan lapisan *max pooling* untuk mereduksi dimensi **(Mittal, 2020)**.

Pengintegrasian *convolution* dengan CapsNet telah dilakukan di berbagai penelitian dengan melibatkan lapisan *max pooling* untuk mereduksi dimensi. Penelitian ini mengusulkan penggunaan *strided convolution* untuk menggantikan *max pooling* dalam mereduksi dimensi sekaligus tetap membantu dalam ekstraksi fitur. Lapisan *strided convolution* dipilih karena memiliki kemampuan untuk dilatih pada proses pelatihan, sementara lapisan *pooling* hanya menggunakan operasi tetap sehingga tidak dapat dilatih. *Strided convolution* sangat berguna saat informasi posisi objek dibutuhkan. Penggunaan memori *strided convolution* juga tergolong lebih efisien, selain itu penggunaan *strided convolution* dapat meningkatkan akurasi **(Ayachi, 2020)**.

Penelitian ini bertujuan menghasilkan model yang dapat digunakan untuk mengklasifikasi citra agar dapat mempermudah sistem dalam merekomendasikan konten media sosial tanaman hias. Penelitian menerapkan *Integration of Convolution with Capsule Network* (ICCN) dengan cara menggabungkan beberapa lapisan *strided convolution* dengan *Capsule Network* (CapsNet). Beberapa lapisan *strided convolution* ditempatkan pada *encoder* CapsNet untuk membantu dalam mengekstraksi citra dan mereduksi dimensi. Penerapan ICCN pada klasifikasi tanaman hias diharapkan dapat memperlihatkan kinerja CapsNet dengan *strided convolution* berdasarkan matriks evaluasi dan hasil pengujian.

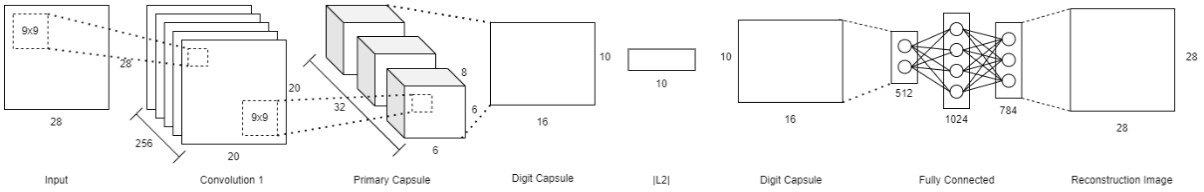
2. METODE PENELITIAN

2.1 Klasifikasi Citra

Klasifikasi citra merupakan sebuah teknik dalam bidang *computer vision* yang memungkinkan komputer untuk mengategorikan berbagai jenis citra. Klasifikasi citra dapat dilakukan dengan 2 cara yaitu *supervised learning* dan *unsupervised learning*. *Supervised learning* berarti melakukan pelatihan model dengan menggunakan *dataset* yang telah diberi label sedangkan *unsupervised learning* tidak memerlukan pelabelan *dataset* secara manual. Setiap citra pada *supervised learning* telah dikaitkan dengan label atau kategori tertentu yang telah ditentukan sebelumnya **(Sathya, 2013)**. Model yang dilatih dengan *supervised* maupun *unsupervised learning* akan menganalisis citra, mengidentifikasi pola visual dan karakteristik umum yang membedakan satu kategori dari kategori lainnya selama proses pelatihan. Pola-pola tersebut dapat berupa bentuk dasar, warna, tekstur, dan struktur yang lebih kompleks **(Aggarwal, 2018)**.

2.2 Capsule Network

Capsule Network (CapsNet) adalah arsitektur jaringan syaraf yang menggunakan kapsul untuk mendapatkan fitur-fitur pada citra serta hubungan spasialnya. Kemampuan CapsNet dalam menangkap hubungan spasial antara fitur-fitur dapat meningkatkan performa model dalam pengenalan citra dan deteksi objek **(Singh, 2020)**.

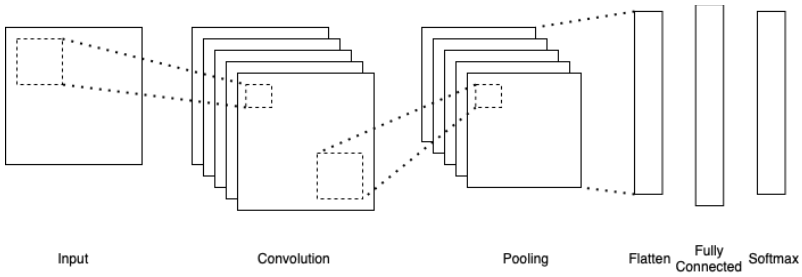


Gambar 1. Arsitektur Capsule Network

CapsNet terdiri dari 2 bagian yaitu *encoder* dan *decoder* seperti yang terlihat pada Gambar 1. Bagian *encoder* dimulai dari *input* hingga bagian perhitungan kesalahan setelah lapisan *digit capsule* dan untuk bagian *decoder* dimulai dari lapisan *digit capsule* hingga ke rekonstruksi

citra. *Encoder* menghasilkan representasi vektor dari *input* menggunakan lapisan-lapisan kapsul. Setiap kapsul merepresentasikan fitur-fitur tertentu beserta propertinya. Kapsul-kapsul ini menghasilkan vektor yang tidak hanya mengkodekan keberadaan fitur-fitur, tetapi juga orientasi, dan skala dari fitur tersebut. *Encoder* pada CapsNet terdiri dari 3 bagian yaitu lapisan *convolution*, lapisan *primary capsule*, dan lapisan *digit capsule*. Bagian *encoder* melakukan operasi *dynamic routing* yang menggunakan mekanisme *routing by agreement*. *Decoder* pada CapsNet berisi lapisan *fully connected* yang berfungsi untuk merekonstruksi citra berdasarkan vektor representasi kapsul. Vektor representasi kapsul merupakan *output* 16 dimensi dari kapsul tertentu pada lapisan *digit capsule* yang mewakili label sebenarnya dari citra. Mekanisme *masking* digunakan pada CapsNet untuk memastikan bahwa hanya kapsul yang relevan dengan *routing agreement* yang digunakan pada rekonstruksi. Mekanisme *masking* berperan untuk mengubah nilai kapsul lainnya di lapisan *digit capsule* menjadi 0 sehingga *decoder* fokus sepenuhnya pada kapsul yang terkait dengan label sebenarnya selama proses rekonstruksi **(Shah, 2018)**.

2.3 Convolutional Neural Network

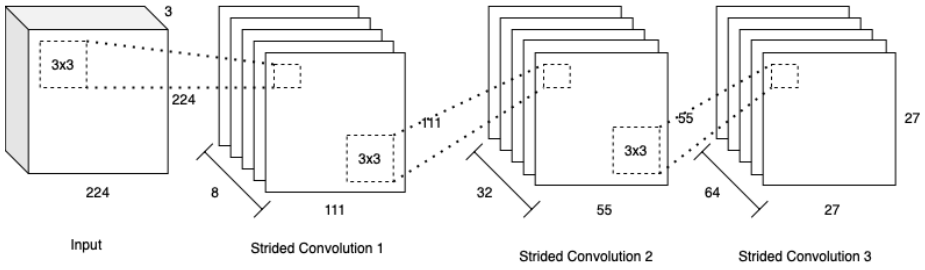


Gambar 2. Arsitektur Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan metode pembelajaran dalam *deep learning* yang terinspirasi oleh korteks visual otak manusia. CNN dirancang khusus untuk memproses data dengan struktur *grid*, seperti citra atau data spasial. CNN memiliki beberapa jenis lapisan yang digunakan untuk melakukan ekstraksi fitur dan pembelajaran. Lapisan-lapisan tersebut adalah lapisan *convolution*, *subsampling* dan *fully connected*. Lapisan *convolution* digunakan untuk mengenali pola dan fitur-fitur penting dalam data. Lapisan *subsampling* (lapisan *pooling*) digunakan untuk mengurangi dimensi. Lapisan *fully connected* bertanggung jawab untuk menghubungkan hasil ekstraksi fitur ke lapisan akhir yang menghasilkan *output*. Lapisan-lapisan CNN umumnya disusun seperti pada Gambar 2 **(O'Shea, 2015)**.

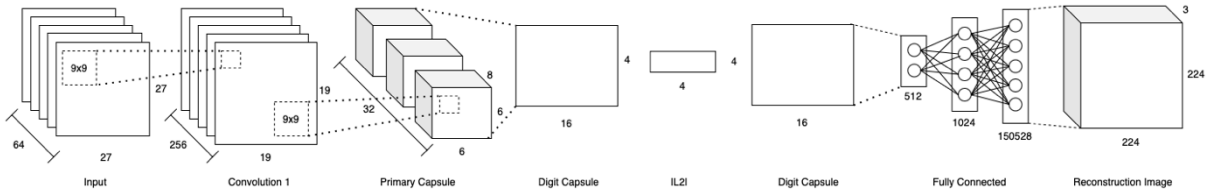
2.4 Integration of Convolution With Capsule Network

Integration of Convolution with Capsule Network (ICCN) merupakan integrasi yang dilakukan pada algoritma *Capsule Network* (CapsNet) dengan algoritma *convolution*. Arsitektur yang diusulkan pada penelitian ini mengintegrasikan beberapa lapisan tambahan pada bagian *encoder* CapsNet. Lapisan yang ditambahkan yaitu lapisan *strided convolution*. Lapisan *strided convolution* dapat menggantikan kegunaan lapisan *max pooling* dalam mengurangi jumlah parameter yang dilatih namun tetap mempertahankan fitur-fitur pada citra **(Springenbreg, 2014)**. Lapisan *strided convolution* memiliki kemampuan untuk dilatih sementara lapisan *pooling* menggunakan operasi yang tetap. Hal ini yang mendasari penggunaan *strided convolution*. Susunan *strided convolution* yang ditambahkan terlihat seperti Gambar 3.



Gambar 3. Susunan Strided Convolution

Lapisan *strided convolution* yang ditambahkan berjumlah 3 lapisan. Lapisan tersebut merupakan lapisan *convolution* yang memiliki filter berjumlah 8, 32, 64 dengan *kernel* berukuran 3x3, *stride* 2 dan menggunakan fungsi aktivasi ReLU.



Gambar 4. Encoder dan Decoder pada ICCN

Proses selanjutnya identik dengan proses yang ada pada CapsNet seperti yang terlihat pada Gambar 4. Perbedaannya terletak pada bagian *input* dan beberapa nilai pada beberapa lapisan. Original CapsNet menggunakan citra yang belum dilakukan proses ekstraksi sedangkan arsitektur yang diusulkan pada penelitian menggunakan hasil dari lapisan *strided convolution* sebagai *input* untuk dilakukan proses *convolution* dengan *kernel* berukuran 9x9. Dimensi *output* lapisan *digit capsule* menjadi 4x16 dikarenakan arsitektur ini digunakan untuk mengklasifikasi 4 kelas citra. Lapisan *fully connected* ke-3 memiliki jumlah neuron yang disesuaikan dengan ukuran dari citra dikarenakan *decoder* bertugas untuk merekonstruksi citra. Citra masukan berukuran 224x224 px dengan 3 *channel* sehingga lapisan *fully connected* ke-3 memiliki neuron sejumlah 150528.

2.5 Matriks Performa Model

Matriks performa model merupakan matriks yang digunakan untuk menilai kinerja model dalam *machine learning* atau *deep learning*. Matriks performa ini memberikan informasi tentang sejauh mana kemampuan model dalam mengklasifikasi data yang diberikan dengan baik. Informasi yang diberikan dapat membantu dalam pengambilan keputusan lanjutan terkait perbaikan untuk meningkatkan hasil prediksi **(Power, 2020)**. Salah satu contoh matriks performa model adalah *confusion matrix*. *Confusion matrix* menggunakan empat kategori untuk mewakili hasil dari prediksi yaitu *true positives, true negatives, false positives*, dan *false negatives*. Nilai dari *confusion matrix* dapat digunakan untuk menghitung matrik evaluasi seperti akurasi, presisi, *recall* dan skor f1 pada Persamaan (1) sampai dengan Persamaan (4). Jenis pengukuran yang umum diterapkan adalah sebagai berikut:

a. Akurasi mengukur kemampuan model dalam mengklasifikasi dengan tepat. Akurasi tidak selalu memberikan gambaran yang lengkap tentang kinerja model, terutama jika data tidak seimbang antara kelas yang berbeda.

$$\frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

b. Presisi berguna dalam mengevaluasi seberapa banyak prediksi positif yang sebenarnya benar dari keseluruhan prediksi positif yang dibuat oleh model.

$$\frac{TP}{TP + FP} \tag{2}$$

c. *Recall* menunjukkan berapa persentase dari data kategori positif diklasifikasikan dengan benar oleh sistem. *Recall* jarang diperhatikan dalam *Machine Learning* dan Linguistik Komputasional. *Recall* lebih sering digunakan dalam konteks kedokteran karena tujuannya adalah mengidentifikasi semua kasus positif nyata.

$$\frac{TP}{TP + FN} \tag{3}$$

d. Skor f1 menyatukan informasi tentang presisi dan *recall* menjadi satu ukuran. Penggunaan prediksi positif dan kemampuan model dalam mendeteksi *instance* positif membuat skor f1 memberikan gambaran yang lebih komprehensif tentang kinerja model.

$$2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

2.6 Tahapan Penelitian

Penelitian dibagi menjadi beberapa tahapan yaitu persiapan data, perancangan arsitektur, pelatihan model, dan pengujian model. Tahapan persiapan data dilakukan dengan mengumpulkan citra dengan format RGB dari internet. Citra dikumpulkan dengan menggunakan 2 cara yaitu mengunduh citra secara manual dan mengunduh citra secara otomatis menggunakan teknik *scraping*. Citra yang terkumpul diseleksi secara manual untuk mendapatkan citra yang sesuai dengan kebutuhan pelatihan dan pengujian model. Citra yang telah diseleksi cenderung memiliki ukuran yang beragam sehingga perlu dilakukan pengubahan ukuran citra.

Arsitektur CNN, CapsNet, dan ICCN dibangun pada tahap perancangan. Perancangan arsitektur CapsNet dilakukan untuk mengetahui perbedaan jumlah parameter yang dilatih pada arsitektur original CapsNet dengan jumlah parameter yang dilatih pada arsitektur ICCN. Arsitektur CNN dirancang untuk digunakan pada tahap pelatihan model CNN. Model CNN digunakan sebagai pembanding performa model ICCN. Tahapan perancangan diawali dengan menyusun lapisan - lapisan yang dibutuhkan sesuai dengan algoritma yang digunakan. Penyusunan lapisan - lapisan model dilakukan dengan bantuan pustaka keras dan tensorflow. Susunan lapisan utama masing-masing model terlihat pada Tabel 1.

Tabel 1. Susunan Lapisan Utama Arsitektur CNN, CapsNet dan ICCN

CNN		CapsNet		ICCN	
Lapisan	**Properti**	**Lapisan**	**Properti**	**Lapisan**	**Properti**
Convolution 1	Kernel 3x3 Stride 2 Filter 8	Convolution 1	Kernel 9x9 Stride 1 Filter 256	Convolution 1	Kernel 3x3 Stride 2 Filter 8
Convolution 2	Kernel 3x3 Stride 2 Filter 32	Primary Capsule	Kernel 9x9 Stride 2 Filter 256 Dimensi 8	Convolution 2	Kernel 3x3 Stride 2 Filter 32
Convolution 3	Kernel 3x3 Stride 2 Filter 64	Digit Capsule	Routing 3 Dimensi 16	Convolution 3	Kernel 3x3 Stride 2 Filter 64
Convolution 4	Kernel 9x9 Stride 1 Filter 256	Dense 1	Neuron 512	Convolution 4	Kernel 9x9 Stride 1 Filter 256
Convolution 5	Kernel 9x9 Stride 2 Filter 256	Dense 2	Neuron 1024	Primary Capsule	Kernel 9x9 Stride 2 Filter 256 Dimensi 8
Dense 1	Neuron 1152 Dropout 0.5	Dense 3	Neuron 150528	Digit Capsule	Routing 3 Dimensi 16
Dense 2	Neuron 1152 Dropout 0.5			Dense 1	Neuron 512
Dense 3	Neuron 1152 Dropout 0.5			Dense 2	Neuron 1024
Dense 4	Neuron 4			Dense 3	Neuron 150528

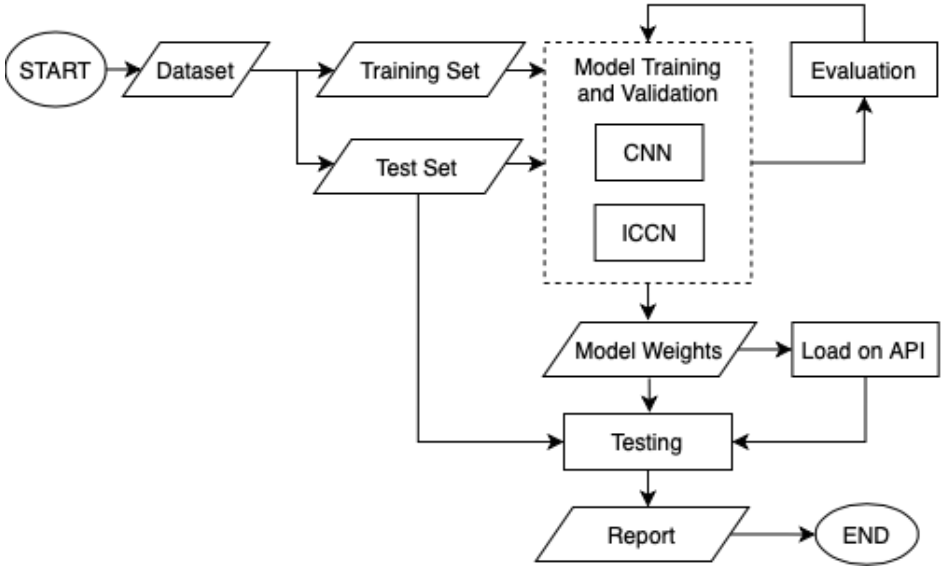
Tahapan pelatihan model dilakukan dengan menggunakan *hyperparameter* seperti *optimizer*, *learning rate* untuk *optimizer*, *learning rate decay*, *batch size*, dan *patience* untuk e*arly stopping*. *Hyperparameter* tersebut digunakan untuk melatih model CNN dan model ICCN. *Exponential decay* digunakan untuk menyesuaikan nilai *learning rate* secara gradual berdasarkan *epoch* dan konstanta *learning rate decay*. *Early stopping* diterapkan untuk otomatis menghentikan pelatihan model apabila kemampuan model dalam memprediksi tidak meningkat setelah beberapa *epoch*. Proses pelatihan model terplot dalam bentuk grafik pelatihan. Tahapan pelatihan menghasilkan sebuah model dengan bobot yang telah disesuaikan dengan data latih.

Tahapan pengujian dibagi menjadi 2 yaitu pengujian kemampuan model secara langsung dengan melakukan klasifikasi menggunakan data uji dan pengujian *web service* klasifikasi yang menggunakan model hasil dari pelatihan. Sebuah *web service* klasifikasi sederhana dibangun menggunakan flask agar dapat melakukan klasifikasi menggunakan *endpoint* yang memanfaatkan model CNN dan model ICCN. Pengujian model pada *web service* dilakukan menggunakan pustaka locust dengan memperhatikan *request* dan *response* pada laporan locust. Pengujian model secara langsung menghasilkan *output* berupa *confusion matrix* dan laporan klasifikasi. Pengujian model pada *web service* menghasilkan *output* berupa laporan pengujian *endpoint*. Analisa model dilakukan berdasarkan grafik pelatihan, laporan klasifikasi serta laporan pengujian *endpoint* dari locust. Analisa juga dilakukan terhadap perbandingan jumlah parameter yang dilatih antara arsitektur CapsNet dan arsitektur ICCN yang menggunakan *strided convolution*. Pengecekan terhadap hasil *feature map* pada *strided*

convolution yang ditambahkan pada bagian *encoder* juga dilakukan untuk memastikan fitur citra tidak hilang.

2.7 Arsitektur Sistem

Arsitektur sistem klasifikasi tanaman hias menggunakan data latih dan data uji terlihat pada Gambar 5.



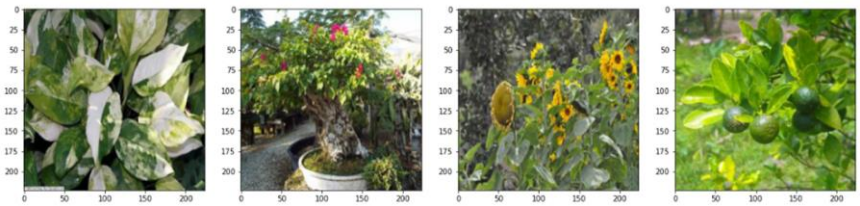
Gambar 5. Arsitektur Sistem

Model dilatih menggunakan data dari internet. Data diseleksi secara manual dan ukurannya diubah. *Dataset* dibagi proporsinya menjadi data latih dan data uji. Data latih digunakan dalam pelatihan model untuk penyesuaian bobot, sedangkan data uji digunakan untuk validasi kemampuan model dan pengujian model. Model pada proses pelatihan akan melakukan klasifikasi kemudian kesalahan dalam klasifikasi digunakan untuk melakukan pembaharuan bobot sebesar nilai kesalahan dan *learning rate*. Model dievaluasi menggunakan data uji untuk memvalidasi kemampuan model selama proses pelatihan. Proses pelatihan dilakukan dengan beberapa iterasi hingga hasil akurasi model tidak mengalami peningkatan. Model yang telah dihasilkan dari proses pelatihan kemudian diuji menggunakan data uji kembali. Pengujian yang dilakukan yaitu melakukan klasifikasi secara langsung menggunakan data uji dan melakukan pengujian terhadap performa *web service* yang menggunakan model CNN dan ICCN. *Web service* dibangun untuk dapat melakukan klasifikasi menggunakan *endpoint* dengan memanfaatkan model yang telah dilatih. *Web service* menerima *input* berupa *JavaScript Object Notation* (JSON) yang berisi *Uniform Resource Locator* (URL). Sistem pada *web service* melakukan pembacaan citra berdasarkan URL kemudian dilakukan klasifikasi oleh model yang telah dilatih. Hasil klasifikasi dikirimkan dalam bentuk *response* JSON. Analisa terhadap performa model dilakukan dengan memperhatikan grafik pelatihan, laporan klasifikasi (nilai akurasi, presisi, *recall*, dan skor f1), dan laporan pengujian dari pustaka locust.

3. HASIL DAN PEMBAHASAN

3.1 Dataset

Dataset yang digunakan berupa citra tanaman hias aglonema, bunga matahari, bonsai, dan kalamansi yang diambil dari google dan kaggle. Setiap tanaman mewakili kategori tanaman hias. Aglonema mewakili daun, bunga matahari mewakili bunga, bonsai mewakili batang, dan kalamansi mewakili buah. Pengumpulan citra dari google dilakukan dengan teknik *scraping* menggunakan pustaka icrawler, dan untuk pengambilan citra dari kaggle dilakukan dengan mengunduh *dataset*. Citra yang terkumpul dikelompokkan sesuai dengan jenis tanaman hias. Citra yang diambil dari google dan kaggle tidak langsung digunakan sebagai *dataset*, namun diseleksi secara manual terlebih dahulu untuk mendapatkan citra yang sesuai. Hasil citra masing-masing tanaman hias yang telah diseleksi sebanyak 1000 citra, sehingga total citra berjumlah 4000 citra. Citra yang telah terkumpul diubah ukurannya menjadi 224x224 px. *Dataset* yang berjumlah 4000 citra dibagi menjadi 80% untuk data latih dan 20% untuk data uji sehingga 3200 citra digunakan sebagai data latih dan 800 citra digunakan sebagai data uji. Sampel *dataset* terlihat seperti Gambar 6.



Gambar 6. Sampel Dataset

3.2 Implementasi Arsitektur

Model di bangun menggunakan GPU GTX 1050 4GB dan RAM DDR4 sebesar 16GB. Perancangan arsitektur disesuaikan dengan kemampuan dari perangkat yang digunakan. Implementasi arsitektur dilakukan menggunakan bahasa pemrograman python dengan bantuan pustaka keras dan tensorflow.

Tabel 2. Perbandingan Parameter CNN, CapsNet, dan ICCN

CNN		CapsNet		ICCN	
Layer	**Trainable Parameter**	**Layer**	**Trainable Parameter**	**Layer**	**Trainable Parameter**
Convolution 1	224	Convolution 1	62464	Convolution 1	224
Convolution 2	2336	Primary Capsule	5308672	Convolution 2	2336
Convolution 3	18496	Digit Capsule	177209344	Convolution 3	18496
Convolution 4	1327360	Dense 1	33280	Convolution 4	1327360
Convolution 5	5308672	Dense 2	525312	Primary Capsule	5308672
Dense 1	10617984	Dense 3	154291200	Digit Capsule	589824
Dense 2	1328256			Dense 1	33280
Dense 3	1328256			Dense 2	525312
Dense 4	4612			Dense 3	154291200
Total	**19936196**		**337430272**		**162096704**

Tabel 2 menunjukkan arsitektur ICCN memiliki total parameter 2 kali lebih sedikit dibandingkan dengan arsitektur CapsNet. Arsitektur ICCN memiliki jumlah parameter yang dilatih sebanyak 7246912 pada bagian *encoder* dan sebanyak 154849792 pada bagian *decoder*. Arsitektur ICCN berhasil dalam mengurangi jumlah parameter CapsNet, namun jumlah parameter yang dilatih tersebut memiliki perbedaan yang signifikan dengan jumlah parameter yang dilatih pada CNN. Arsitektur CNN hanya menghasilkan 1 *output* yaitu nilai klasifikasi sedangkan arsitektur ICCN menghasilkan 2 *output* yaitu nilai klasifikasi pada bagian *encoder* dan citra rekonstruksi pada bagian decoder. Proses rekonstruksi citra tersebut yang membuat perbedaan jumlah parameter yang dilatih menjadi signifikan.

Tabel 3. Perbandingan Out Shape CNN, CapsNet, dan ICCN

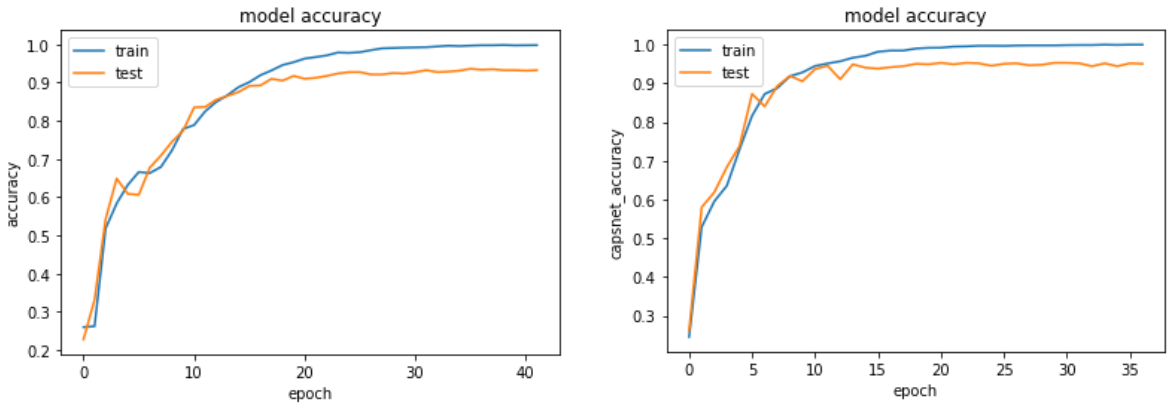
CNN		CapsNet		ICCN	
Layer	**Output Shape**	**Layer**	**Output Shape**	**Layer**	**Output Shape**
Input	224, 224, 3	Input	224, 224, 3	Input	224, 224, 3
Convolution 1	111, 111, 8	Convolution 1	216, 216, 256	Convolution 1	111, 111, 8
Convolution 2	55, 55, 32	Primary Capsule	346112, 8	Convolution 2	55, 55, 32
Convolution 3	27, 27, 64	Digit Capsule	4, 16	Convolution 3	27, 27, 64
Convolution 4	19, 19, 256	Dense 1	512	Convolution 4	19, 19, 256
Convolution 5	6, 6, 256	Dense 2	1024	Primary Capsule	1152, 8
Dense 1	1152	Dense 3	150528	Digit Capsule	4,16
Dense 2	1152			Dense 1	512
Dense 3	1152			Dense 2	1024
Dense 4	4			Dense 3	150528

Tabel 3 menunjukkan bahwa penambahan lapisan pada ICCN menyebabkan *output shape* yang dihasilkan memiliki ukuran yang lebih kecil. Lapisan *digit capsule* pada ICCN hanya memproses data sebanyak 1152 sedangkan lapisan *digit capsule* pada CapsNet memproses data 8 dimensi sebanyak 346112. Hal ini yang membuat jumlah parameter yang dilatih pada ICCN menjadi lebih sedikit.

3.3 Pelatihan Model

Pelatihan model CNN dan ICCN dilakukan dengan menggunakan *hyperparameter* yang sama. *Hyperparameter* yang digunakan yaitu *batch size* dengan nilai 20, *patience* dengan nilai 3, dan *epoch* maksimum dengan nilai 50. Pelatihan model otomatis berhenti apabila tidak terjadi peningkatan kemampuan pada model selama 3 *epoch* berturut-turut (sesuai dengan nilai *patience*). *Adaptive Moment Estimation* (ADAM) digunakan sebagai *optimizer* dengan *learning rate* sebesar 0,001 dan nilai *learning rate decay* sebesar 0,9. *Learning rate* dan *learning rate decay* mempengaruhi kecepatan pembelajaran model pada proses pelatihan. Pengukuran kinerja selama proses pelatihan dilakukan dengan mengevaluasi kinerja model setiap *epoch* menggunakan data uji.

Pelatihan model ICCN berhenti otomatis pada *epoch* ke-37 karena sudah tidak mengalami peningkatan kemampuan. Durasi pelatihan model per *epoch* sekitar 69 detik. Proses pelatihan menggunakan sumber daya RAM sebesar 70% dan GPU sebesar 60%. Model ICCN berhasil memperoleh akurasi evaluasi terakhir sebesar 95%. Model ICCN memiliki akurasi evaluasi lebih tinggi dibandingkan model CNN dengan susunan lapisan yang mirip. Grafik hasil pelatihan dan evaluasi seluruh *epoch* terlihat pada Gambar 7.



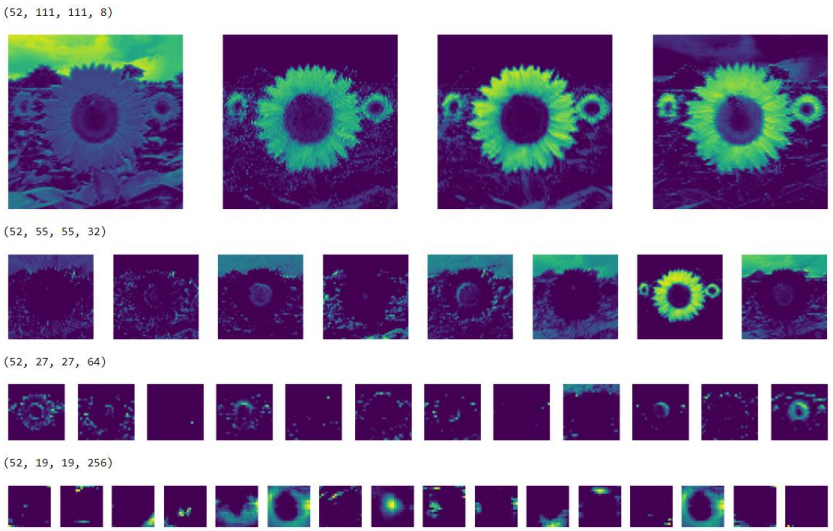
Pelatihan CNN Pelatihan ICCN

Gambar 7. Grafik Pelatihan Model

Model ICCN memiliki kemampuan belajar lebih cepat dibandingkan dengan model CNN seperti yang terlihat pada Gambar 7. Model CNN memiliki akurasi evaluasi terakhir sebesar 93%. Pelatihan model CNN berhenti otomatis pada *epoch* ke-43 karena sudah tidak mengalami peningkatan kemampuan. Model ICCN mengungguli model CNN dari segi tingkat keakuratan dalam klasifikasi. Model CNN mengungguli model ICCN pada waktu pelatihan yang lebih singkat yaitu sekitar 16 detik/*epoch*. Pelatihan model CNN menggunakan sumber daya sebesar 67% RAM dan 65% GPU.

3.3.1 Encoder

Penggunaan *strided convolution* efektif dalam mengurangi parameter yang dilatih tanpa menghilangkan fitur-fitur yang ada pada citra.

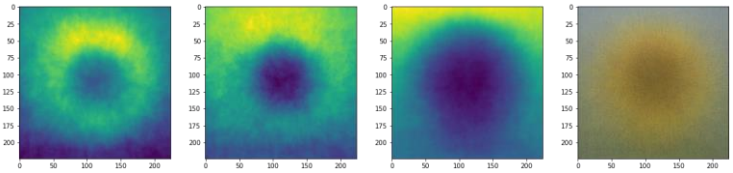


Gambar 8. Fitur Citra

Gambar 8 menunjukkan bahwa fitur pada citra tidak hilang pada proses *strided convolution* berlangsung. *Encoder* mendeteksi hubungan lokal antar fitur-fitur pada lapisan sebelumnya. Pendeteksian asosiasi lokal antar fitur-fitur membantu dalam mengidentifikasi pola, tekstur atau bagian penting lainnya pada citra.

3.3.1 Decoder

Decoder pada ICCN memiliki kemampuan untuk merekonstruksi citra sesuai dengan data pada lapisan *capsule digit* yang telah di *masking*. *Decoder* pada ICCN cenderung menghasilkan citra rekonstruksi yang buram dan memiliki tingkat kemiripan yang tinggi pada rekonstruksi citra di kelas yang sama. *Decoder* memiliki nilai *loss* sebesar 6,02% menggunakan data uji.



Gambar 9. Citra Rekonstruksi

Hasil rekonstruksi citra model ICCN ditunjukkan pada Gambar 9. Citra ke-1 hingga ke-3 pada merupakan visualisasi berdasarkan *channel* citra sedangkan citra ke-4 merupakan hasil visualisasi citra dalam bentuk RGB.

3.4 Pengujian Model

Pengujian Model secara langsung dilakukan terhadap model CNN dan ICCN dengan menggunakan data uji. Hasil dari pengujian berupa laporan klasifikasi yang berisi nilai matriks evaluasi. Laporan klasifikasi terlihat pada Tabel 4.

Tabel 4. Laporan Klasifikasi CNN dan ICCN

Kategori	CNN				ICCN			
	Presisi	Recall	Skor F1	Akurasi	Presisi	Recall	Skor F1	Akurasi
Aglonema	90	92	91	93%	93	95	94	95%
Bonsai	96	91	93		97	92	94	
Bunga Matahari	98	96	97		98	97	97	
Kalamansi	89	94	91		93	96	94	

Model ICCN memiliki nilai akurasi, presisi, *recall*, dan skor f1 yang lebih tinggi dibandingkan model CNN. Nilai akurasi menunjukkan persentase tanaman hias yang benar diklasifikasi sesuai dengan keseluruhan tanaman hias. CNN dan ICCN memperoleh akurasi masing-masing sebesar 93% dan 95%. Nilai presisi, *recall* dan skor f1 dihasilkan dari klasifikasi masing-masing kategori tanaman hias. Sebagai contoh model CNN dalam kategori aglonema memperoleh nilai presisi, *recall*, dan skor f1 sebesar 90%, 92%, dan 91%. Presisi menyatakan pengukuran sejauh mana model benar dalam mengidentifikasi aglonema dari semua prediksi yang dinyatakan sebagai aglonema, *recall* mengukur sejauh mana model memprediksi tanaman sebagai aglonema dibandingkan dengan data aglonema sebenarnya. Skor f1 memberikan gambaran mengenai kinerja model ketika ada ketidakseimbangan antara hasil diprediksi benar aglonema dan hasil diprediksi salah aglonema.

Tabel 5. Laporan Pengujian Locust

Model	Request	Fail	Average (milidetik)	Min (milidetik)	Max (milidetik)
CNN	1099	0	5804	3347	31087
ICCN - Encoder Decoder	1096	0	6385	2358	49747
ICCN - Encoder	1123	0	5563	2432	38776

Pengujian *web service* yang menggunakan model pelatihan dilakukan dengan bantuan pustaka locust. Model pelatihan yang digunakan yaitu model CNN dan model ICCN. Klasifikasi yang dilakukan pada *endpoint* tidak membutuhkan citra rekonstruksi, oleh sebab itu pengujian terhadap *web service* model ICCN dibagi menjadi 2 jenis yaitu menggunakan model ICCN yang terdiri dari *encoder* dan *decoder* serta menggunakan model ICCN yang hanya terdiri dari *encoder*. Pengujian *web service* dilakukan selama 1 menit dengan 4 *worker* dan 100 *virtual user*. Hasil pengujian yang terlihat pada Tabel 5 menunjukkan ICCN yang hanya menggunakan bagian *encoder* menangani *request* lebih banyak yaitu sebanyak 1123 *request* dan rata-rata *response* yang diberikan lebih cepat yaitu 5563 milidetik.

4. KESIMPULAN

Penelitian ini melakukan *Integration of Convolution with Capsule Network* (ICCN) untuk menghasilkan model klasifikasi agar dapat mempermudah media sosial dalam merekomendasikan konten yang sesuai dengan preferensi pengguna. ICCN diterapkan dengan menggabungkan *strided convolution* dengan *Capsule Network* (CapsNet). Hasil perancangan arsitektur model menunjukkan ICCN memiliki parameter yang dilatih sebanyak 2 kali lebih sedikit dibandingkan original CapsNet yaitu sejumlah 337430272 untuk CapsNet dan 162096704 untuk ICCN. Penurunan jumlah parameter yang dilatih dipengaruhi oleh pereduksian dimensi yang dilakukan menggunakan *strided convolution*. Jumlah parameter yang dilatih pada ICCN masih terpaut jauh apabila dibandingkan dengan CNN. CNN memiliki jumlah parameter yang dilatih sebanyak 19936196. CNN hanya berfokus untuk memprediksi citra tanpa melakukan proses rekonstruksi citra. Apabila CNN dibandingkan hanya dengan bagian *encoder* ICCN, maka jumlah parameter yang dilatih lebih sedikit yaitu sebanyak 7246912. Model ICCN membutuhkan 37 *epoch* untuk mencapai kemampuan yang maksimum sedangkan CNN membutuhkan epoch yang lebih banyak yaitu sebanyak 42 *epoch* namun CNN memiliki waktu pelatihan per *epoch* lebih cepat dibandingkan ICCN yaitu selama 16 detik/*epoch*. Kompleksitas operasi dari ICCN membuat model ICCN membutuhkan waktu yang lama untuk dilatih. ICCN mengungguli CNN dari segi keakuratan dalam mengklasifikasi citra. Model ICCN berhasil meraih akurasi sebesar 95% menggunakan data uji sedangkan model CNN meraih nilai akurasi sebesar 93%. Hasil pengujian terhadap *web service* yang menggunakan model terlatih menunjukkan model ICCN yang hanya menggunakan bagian *encoder* memiliki kinerja yang lebih baik dibandingkan dengan model CNN dan model ICCN yang menggunakan *encoder* dan *decoder*. Kompleksitas dan jumlah parameter yang dilatih sangat berpengaruh pada performa model dalam melakukan klasifikasi dan lamanya proses pelatihan model. Model ICCN memiliki performa baik pada saat diuji secara langsung dan digunakan pada *web service* sehingga model ini dapat digunakan sebagai model klasifikasi citra tanaman hias pada media sosial.

DAFTAR RUJUKAN

Aggarwal, V. (2018). A Review: Deep Learning Technique for Image Classification. *ACCENTS Transactions on Image Processing and Computer Vision, 4*(11), 21 - 25.

Ayachi, R., Afif, M., Said, Y., & Atri, M. (2020). Strided Convolution Instead of Max Pooling for Memory Efficiency of Convolutional Neural Networks. *International Conference on Sciences of Electronics,* (pp. 234-243).

Ali, M., Hassan, M., Kifayat, K., Kim, J. Y., Hakak, S., & Khan, M. K. (2023). Social Media Content Classification and Community Detection Using Deep Learning and Graph Analytics. *Technological Forecasting and Social Change, 188*(1), 1-13.

Biscione, V., & Bowers, J. S. (2021). Convolutional Neural Networks Are Not Invariant to Translation, but They Can Learn to Be. *The Journal of Machine Learning Research, 22*(1), 10407-10434.

Eg, R., Tønnesen, Ö. D., & Tennfjord, M. K. (2023). A Scoping Review of Personalized User Experiences on Social Media: The Interplay Between Algorithms and Human Factors. *Computers in Human Behavior Reports, 9*(1), 1-17.

Hutmacher, F., & Appel, M. (2023). The Psychology of Personalization in Digital Environments: From Motivation to Well-Being–a Theoretical Integration. *Review of General Psychology, 27*(1), 26-40.

Kauderer-Abrams, E. (2017, December 10). *Quantifying Translation-Invariance in Convolutional Neural Networks*. Retrieved from arxiv.org.

Mazzia, V., Salvetti, F., & Chiaberge, M. (2021). Efficient-CapsNet: Capsule Network With Self-Attention Routing. *Scientific Reports, 11*(1), 1-13.

Mittal, A., Kumar, D., Mittal, M., Saba, T., Abunadi, I., Rehman, A., & Roy, S. (2020). Detecting Pneumonia Using Convolutions and Dynamic Capsule Routing for Chest X-Ray Images. *Sensors, 20*(4), 1-30.

O'Shea, K., & Nash, R. (2015, November 26). *An Introduction to Convolutional Neural Networks*. Retrieved from arxiv.org.

Powers, D. M. (2020, October 11). *Evaluation: From Precision, Recall and F-Measure to Roc, Informedness, Markedness and Correlation*. Retrieved from arxiv.org.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules. *Neural Information Processing Systems,* (pp. 1-11).

Sathya, R., & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. *International Journal of Advanced Research in Artificial Intelligence, 2*(2), 34-38.

Sharma, P., Arya, R., Verma, R., & Verma, B. (2023). Conv-CapsNet: Capsule Based Network for COVID-19 Detection Through X-Ray Scans. *Multimedia Tools and Applications, 82*(18), 28521–28545.

Singh, C. K., Gangwar, V. K., Majumder, A., Kumar, S., Ambwani, P. C., & Sinha, R. (2020). A Light-Weight Deep Feature Based Capsule Network. *International Joint Conference on Neural Networks,* (pp. 1-8).

Setyawan, D. (2022). Tinjauan Peningkatan Penjualan Tanaman Hias Di Masa Pandemi Dengan Life Cycle Assesment (LCA). *National Multidisciplinary Sciences,* (pp. 185-193).

Shah, P. L., Gupta, T. K., Dhakad, J. B., & D'silva, M. R. (2018). A Review Paper on Understanding Capsule Networks. *International Journal of Engineering Development and Research, 6*(4), 58-65.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014, December 21). *Striving for Simplicity: The All Convolutional Net*. Retrieved from arxiv.org.

Tiwari, S., & Jain, A. (2021). Convolutional Capsule Network for COVID-19 Detection Using Radiography Images. *International Journal of Imaging Systems and Technology, 31*(2), 525-539.

Wang, P., Fan, E., & Wang, P. (2021). Comparative Analysis of Image Classification Algorithms Based on Traditional Machine Learning and Deep Learning. *Pattern Recognition Letters, 141*(1), 61-67.

Yao, H., Tan, Y., Xu, C., Yu, J., & Bai, X. (2021). Deep Capsule Network for Recognition and Separation of Fully Overlapping Handwritten Digits. *Computers & Electrical Engineering, 91*(1), 1-12.

Yu, Z., Wang, K., Wan, Z., Xie, S., & Lv, Z. (2023). Popular Deep Learning Algorithms for Disease Prediction: A Review. *Cluster Computing, 26*(2), 1231-1251.