BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

2.1.1 *Nematode Identification using Artificial Neural Network* [4]

Penelitian dengan judul *"Nematode Identification using Artificial Neural Networks"* yang dilakukan oleh Jason Uhlemann, Oisin Cawley,dan Kakouli-Duarte berfokus pada identifikasi nematoda, dengan menekankan pentingnya nematoda sebagai bioindikator lingkungan. Penelitian ini mengusulkan penggunaan CNN untuk otomatisasi dan pengembangan proses identifikasi nematoda berdasarkan citra mikroskopik. Para peneliti menggunakan nematoda entomopatogenik (EPN), Tiga spesies EPN yang digunakan adalah *Heterorhabditis Bacteriophora*, *Steinernema Carpocapsae*, dan *Steinernema Feltiae*.

Metodologi yang diaplikasikan antaralain mengubah ukuran citra nematoda menjadi 224x224 untuk proses pelatihan, kemudian dilakukan rotasi acak, pembalikan secara vertikal dan horizontal sebesar 50%. Pada pelatihan model, peneliti menggunakan tiga metode pelatihan yang berbeda yaitu ekstraksi fitur, *fine-tuning*, dan inisialisasi acak. Peneliti juga menggunakan SGD, RMSProp, dan Adam sebagai tiga *optimizer* yang dibandingkan. Teknik lain yang diterapkan adalah *gradient clipping* dan *label smoothing*. Jumlah *epoch* pelatihan yang digunakan yaitu 2000 *epoch* dengan menyertakan *patience* pada *monitor early stopping* sebanyak 150 *epoch* sehingga jika model tidak mencapai tingkat kehilangan validasi yang lebih rendah daripada yang terbaiknya dalam waktu itu, pelatihan akan dihentikan.

Kemudian, lapisan tambahan diterapkan antara lapisan *global average pooling* dari model Xception dan lapisan dense akhir. Lapisan tambahan ini termasuk *dropout*, *batch normalization*, dan *dense layers*. Jumlah unit (*neuron*) yang digunakan untuk lapisan *dense* didasarkan pada *output* dari lapisan *global average pooling*, yang merupakan 2.048 unit. Model-model ini

diuji menggunakan lapisan *dense* dengan setengah jumlah unit, jumlah unit yang sama, dan dua kali jumlah unit. Perubahan lain yang diuji termasuk perubahan nilai *gradient clipping* dan nilai *patience early stopping*. Pada penelitian ini, model yang dilatih menggunakan *optimizer* Adam lebih sering mengalami kegagalan secara signifikan daripada model yang menggunakan *optimizer* lainnya.

Adapun poin penting pada penelitian ini adalah :

- Penelitian ini menggunakan CNN untuk otomatisasi dan pengembangan proses identifikasi nematoda berdasarkan citra mikroskopik.
- Augmentasi yang diterapkan antaralain mengubah ukuran citra nematoda (*resize*), melakukan rotasi acak dan pembalikan secara vertikal dan horizontal, serta menggunakan tiga metode pelatihan yang berbeda: ekstraksi fitur, *fine-tuning*, dan inisialisasi acak.
- SGD, RMSProp, dan Adam digunakan sebagai *optimizer* yang dibandingkan dalam penelitian ini.
- Model-model yang diuji menggunakan lapisan tambahan antara lapisan *global average pooling* dan lapisan *dense* akhir, termasuk *dropout*, *batch normalization*, dan *dense layers*.
- Model yang dilatih menggunakan *optimizer* Adam lebih sering mengalami kegagalan secara signifikan daripada model dengan *optimizer* lainnya.

2.1.2 *Deep Learning Models for Automatioc Identification of Plant Parasitic Nematode* [5]

Penelitian dengan judul "*Deep Learning Models for Automatic Identification Of Plant Parasitic Nematode"* yang dilakukan oleh Nabila Husna Shabrina, Ryukin Aranta Lika, dan Siwi Indarti, berfokus pada klasifikasi nematoda yang umum ditemukan di Indonesia. Penelitian ini

membandingkan empat model *deep learning* antaralain ResNet101v2, CoAtNet-0, EfficientNetV2B0, dan EfficientNetV2M.

Metodologi penelitian ini antaralain dimulai dengan pengumpulan dataset nematoda parasit tumbuhan dan mengklasifikasikannya ke dalam beberapa kelas. Proses pra-pemrosesan data diterapkan pada dataset menggunakan deteksi tepi (*edge detection*) dengan tujuan untuk menetapkan wilayah mana nematoda berada dalam gambar, kemudian dari hasil *edge detection* citra mikroskopik nematoda akan dilakukan proses pemotongan (*cropping*). Hal ini juga mengurangi informasi yang redundan dengan memotong area kosong yang tidak berisi nematoda. pemotongan (*cropping*). Sampel-sampel tersebut kemudian diubah menjadi gambar skala abu-abu (*grayscale*) karena klasifikasi hanya didasarkan pada fitur morfologi dari sampel-sampel tersebut. Untuk meningkatkan ukuran dataset, dilakukan augmentasi pemutaran gambar (*image flip*), penambahan *noise*, pengaburan gambar (*image blurring*), pencerahan (*brightening*), dan kontras (*contrast*). Ukuran citra yag digunakan adalah 224x224.

Penelitian ini menggunakan augmentasi *on-the-fly* (online augmentation) untuk meningkatkan keberagaman data. Beberapa teknik seperti translasi dan rotasi tidak dipilih karena ada kekhawatiran bahwa hasil augmentasi dapat menyembunyikan fitur diskriminatif penting pada sampel, yang dapat menurunkan akurasi model. Dalam penelitian ini, beberapa metrik evaluasi yang digunakan antaralain *test accuracy*, *F1-score*, *average precision*, *recall*, dan *mean class accuracy*.

Model-model terbaik berdasarkan akurasi pengujian (%) adalah: EfficientNetV2M menggunakan RMSProp dan augmentasi kecerahan; EfficientNetV2B0 menggunakan RMSProp dan augmentasi kecerahan; CoAtNet-0 menggunakan SGD tanpa augmentasi; EfficientNetV2B0 menggunakan SGD tanpa augmentasi; dan EfficientNetV2B0 menggunakan RMSProp tanpa augmentasi.

Adapun poin penting pada penelitian ini adalah :

- Penelitian ini fokus pada klasifikasi nematoda yang umum ditemukan di Indonesia.
- Empat model *pre-trained* yang dibandingkan dalam penelitian ini yaitu ResNet101v2, CoAtNet-0, EfficientNetV2B0, dan EfficientNetV2M.
- Pra-pemrosesan data melibatkan deteksi tepi (*edge detection*) untuk menentukan wilayah nematoda dalam gambar dan dilanjutkan dengan pemotongan (*cropping*) untuk menghilangkan area kosong yang tidak berisi nematoda. Dataset kemudian diubah menjadi gambar skala abu-abu (*grayscale*) karena klasifikasi hanya bergantung pada fitur morfologi.
- Untuk meningkatkan ukuran dataset, dilakukan augmentasi pemutaran gambar, penambahan *noise*, pengaburan gambar, pencerahan, dan kontras.
- Model-model terbaik berdasarkan akurasi pengujian adalah EfficientNetV2M dengan RMSProp dan augmentasi kecerahan, EfficientNetV2B0 dengan RMSProp dan augmentasi kecerahan, CoAtNet-0 dengan SGD tanpa augmentasi, EfficientNetV2B0 dengan SGD tanpa augmentasi, dan EfficientNetV2B0 dengan RMSProp tanpa augmentasi.
- Metrik evaluasi yang digunakan antaralain *test accuracy, F1-score, average precision, recall*.

2.1.3 *Nemanet: A Convolutional Neural Network Model for Identification of Nematodes Soybean Crop in Brazil* [6]

Penelitian dengan judul *"Nemanet: A Convolutional Neural Network Model for Identification of Nematodes Soybean Crop in Brazil"* yang dilakukan oleh Andre da Silva Abade, Flavio de Barros Vidal, Lucas Faria Porto, dan Paulo Afonso Ferreira, menyajikan sebuah set data publik baru

yang disebut NemaDataset yang berisi 3.063 gambar mikroskopis dari lima spesies nematoda yang memiliki relevansi kerusakan yang paling signifikan bagi tanaman kedelai. Selain itu, pada penelitian ini juga mengusulkan model CNN baru yang disebut NemaNet dan melakukan penilaian perbandingan dengan tiga belas model CNN lainnya.

Metodologi yang dilakukan pada penelitian ini adalah, pada langkah *pre-processing* difokuskan mempertahankan tingkat detail dan rasio sinyal-ke-noise yang cukup tinggi sambil menghindari aliasing dan melakukannya dengan tingkat shading, ketidakhomogenan fotometrik, dan distorsi geometri yang cukup rendah. Citra mikroskopik nematoda juga dilakukan proses pemangkasan dan pemusatan objek yang diperlukan untuk mengurangi fitur-fitur yang tidak menentukan klasifikasi patogen.

Penelitian ini menggunakan dua pendekatan, pendekatan pertama menggunakan teknik *transfer learning*, menggunakan jaringan saraf dasar sebagai ekstraktor fitur, di mana gambar-gambar dari dataset target diberikan kepada jaringan saraf dalam rangkaian mendalam. Fitur-fitur yang dihasilkan sebagai input pada lapisan klasifikasi akhir diekstraksi. Dengan fitur-fitur ini, dibangun klasifikasi baru dan model dibuat. Pada jaringan dasar, lapisan terakhir klasifikasinya diganti dalam proses *fine-tuning*, dan bobot lapisan sebelumnya juga dimodifikasi. Pendekatan kedua adalah *from scratch*, di mana bobot jaringan tidak diwarisi dari model sebelumnya, melainkan diinisialisasi secara acak. Pendekatan ini membutuhkan set pelatihan yang lebih besar. Untuk mengevaluasi kinerja klasifikasi arsitektur yang diusulkan adalah *loss*, *overall accuracy*, *F1-score*, *presisi*, *recall*, dan *specificity*. Selain itu peneliti juga menggunakan *confusion matrix*, *receiving operator characteristics* (ROC)-*Area Under Curve* (AUC) *Analysis*.

Di antara model-model CNN yang dilatih dan dievaluasi menggunakan teknik *from scratch*, model Xception dan InceptionResNetV2 yang menggunakan koneksi residual yang dikombinasikan dengan arsitektur Inception memiliki performa lebih rendah dibandingkan dengan jaringan

InceptionV3. Sementara untuk *transfer learning*, DenseNet menduduki lima besar dalam hal *average accuracy*, yaitu DenseNet201, DenseNet169, dan DenseNet121. Arsitektur DenseNet memiliki beberapa keunggulan yang menarik yaitu mereka mengurangi masalah menghilangnya gradien, memperkuat propagasi fitur, mendorong penggunaan ulang fitur, dan secara substansial mengurangi jumlah parameter.

Adapun poin penting pada penelitian ini adalah :

- Penelitian ini mengusulkan model CNN baru bernama NemaNet dan membandingkannya dengan tiga belas model CNN lainnya.
- Dua pendekatan digunakan dalam penelitian ini: pendekatan *transfer learning* dan pendekatan dari awal (*from scratch*).
- Pendekatan *transfer learning* menggunakan jaringan saraf dasar sebagai ekstraktor fitur dan memodifikasi lapisan klasifikasi akhir.
- *Optimizer* yang digunakan SGD dan evaluasi kinerja model menggunakan berbagai metrik, termasuk *loss*, *overall accuracy*, *F1-score*, *presisi*, *recall*, *specificity*, *confusion matrix*, dan ROC-AUC *analysis*.
- Dengan menggunakan teknik *transfer learning*, DenseNet201, DenseNet169, dan DenseNet121 menunjukkan performa yang baik dalam hal *average accuracy*. Arsitektur DenseNet memiliki keunggulan seperti mengatasi masalah menghilangnya gradien, memperkuat propagasi fitur, dan mengurangi jumlah parameter.

Berdasarkan poin penting yang telah disebutkan di atas, penulis mengambil beberapa hal yang dijadikan sebagai acuan penelitian, antara lain:

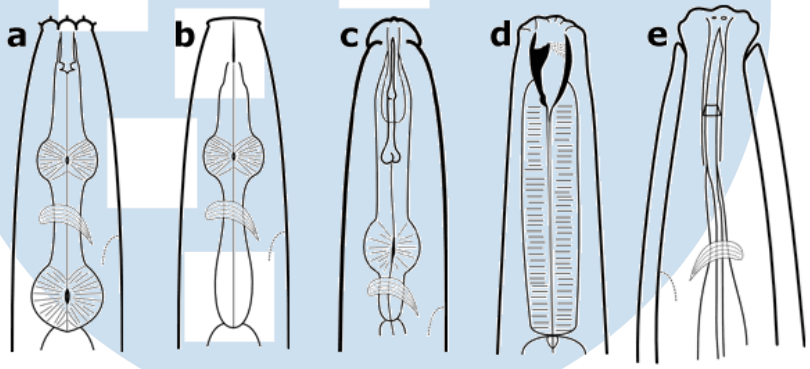
Tabel 2. 1 Poin penting yang diambil pada penelitian

No.	Nama Peneliti	Judul Penelitian (Tahun)	Poin yang dijadikan acuan penelitian
1	Jason Uhlemann, Oisin Cawley, dan Kakouli- Duarte	*Nematode Identification using Artificial Neural Networks* (2020)	• Pada penelitian tersebut, Xception dan ResNet50 dua dari beberapa model yang dibandingkan yang memiliki performa cukup baik dalam melakukan klasifikasi nematoda, Penulis akan menggunakan Xception dan ResNet50 adalah dua dari empat model yang akan digunakan pada penelitian ini. • Optimizer SGD akan digunakan untuk mengoptimalkan kinerja model. • Teknik *transfer learning* yang digunakan adalah ekstraksi fitur. • Layer tambahan yang digunakan adalah *dense* dan *dropout*.
2	Nabila Husna Shabrina, Ryukin Aranta Lika, dan SiwiIndarti	*Deep Learning Models for Automatic Identification of Plant Parasitic Nematode* (2023)	• Pada penelitian tersebut, model EfficientV2B0 dengan augmentasi *brightness* memiliki performa yang baik dalam melakukan klasifikasi nematoda. Penulis akan menggunakan *brightness* untuk melakukan augmentasi dataset. EfficientNetV2B0 adalah tiga dari empat model yang akan digunakan pada penelitian ini. • Metrik evaluasi yang digunakan antaralain *test accuracy*, *F1-score*, *average precision*, dan *recall*.
3	Andre da Silva Abade, Flavio de Barros Vidal,	*Nemanet: A Convolutional Neural Network*	• Pada penelitian tersebut, DenseNet menjadi 5 besar top model dengan teknik *transfer learning* untuk

	Lucas Faria Porto, dan Paulo Afonso Ferreira	*Model for Identification of Nematodes Soybean Crop in Brazil* (2021)	klasifikasi nematoda. DenseNet201 adalah model ke empat yang akan digunakan pada penelitian ini.

2.2 Tinjauan Teori

2.2.1 Nematoda



Gambar 2. 1 (a) *bacterivore*, (b) *fungivore*, (c) *herbivore* (*plant-parasitic*), (d) *predator*, (e) *omnivore* [7]

Nematoda merupakan organisme multiseluler yang dapat hidup di lingkungan tanah, air laut dan air tawar. Nematoda bergantung pada lapisan tipis air, hidup dan bergerak dalam jalur pori-pori tanah dengan diameter 25–100 µm. Nematoda parasitik (*herbivore*) hidup sebagai parasit tanaman, sedangkan nematoda non parasitik tidak berperan sebagai parasitik tanaman, melainkan keberadaannya dapat memengaruhi respirasi dan mineralisasi tanah [8]. Keberadaan nematoda ini juga berperan sebagai indikator kondisi jaring makanan [9].

Bersumber dari jurnal dan situs publik pertanian eOrganic [8], [9], [7] . *Bacterivore,* merupakan trofik nematoda non parasitik pemakan bakteri, keberadaan nematoda ini sangat melimpah di dalam tanah. Pada *bacterivore* umumnya memiliki mulut atau stoma yang berbentuk tabung berongga yang berfungsi untuk menelan bakteri. Genus nematoda non parasitik trofik

bacterivore antara lain *Genus Acrobeles, Genus Acrobeloides, Genus Rhabditis*, dan sebagainya. Nematoda non parasitik *bacterivore* ditunjukkan pada Gambar 2.2.

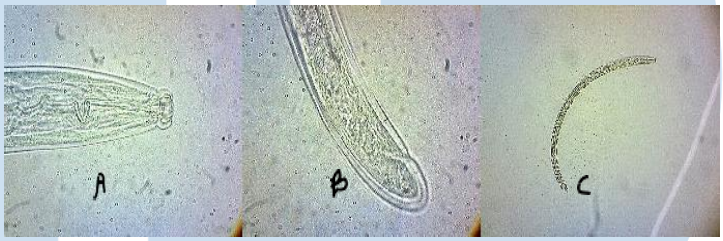
Gambar 2. 2 *anterior*, (b) *posterior*, (c) *whole body*

Fungivore, merupakan trofik nematoda non parasitik pemakan jamur. Pada *fungivore*, umumnya memiliki stilet untuk menusuk hifa jamur, kemudian tidak memiliki setae, bagian tubuh meruncing, memiliki metacorpus dan klep, serta ekornya membulat. Genus nematoda non parasitik trofik *fungivore* antaralain Genus Aphelenchida, *Genus Tylenchus*, dan sebagainya. Nematoda non parasitik *fungivore* ditunjukkan pada Gambar 2.3.

Gambar 2. 3 *anterior*, (b) *posterior*, (c) *whole body*

Predator dan *Omnivore*, merupakan trofik nematoda non parasitik yang memiliki kesamaan dalam perilaku makan mereka. Keduanya adalah pemakan pemangsa, yang berarti mereka memangsa organisme lain untuk memenuhi kebutuhan makanan mereka, namun mereka tidak menjadi parasit untuk tanaman. *Predator* secara khusus memakan nematoda lain atau organisme kecil lainnya sebagai sumber makanan mereka. Sedangkan *omnivore* tergantung pada kondisi lingkungan dan ketersediaan makanan; misalnya, nematoda *omnivore* dapat menjadi *predator*, tetapi jika tidak ada

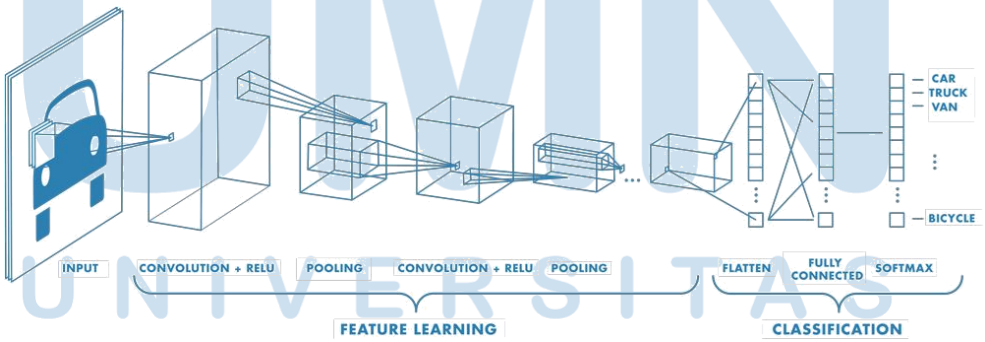
sumber makanan utama, mereka dapat memakan jamur atau bakteri. Pada *predator*, umumnya memiliki stoma berbentuk *thrust-shaped* yang lebih terbuka lebar dan tidak dilengkapi dengan gigi. Sedangkan *omnivore*, memiliki setae atau terkadang tidak jelas, memiliki stilet, tidak memiliki knob tetapi tidak memiliki basal bulbus, stilet berada di posisi tengah. Genus nematoda non parasitik trofik *predator* dan *omnivore* antaralain *Genus Dorylaimida, Genus Mononchida*, dan sebagainya.



Gambar 2. 4 *Predator; (a) anterior, (b) posterior, (c) whole body*

Gambar 2. 5 *Omnivore*; (a) *anterior*, (b) *posterior*, (c) *whole body*

2.2.2 *Convolutional Neural Network*



Gambar 2. 6 Arsitektur *Convolutional Neural Network* (CNN) [10]

Convolutional Neural Network (CNN) didasarkan pada konsep konvolusi, yang merupakan operasi matematis untuk menggabungkan dua fungsi dan menghasilkan fungsi baru. Dalam konteks CNN, konvolusi digunakan untuk memproses data citra dengan menerapkan filter atau *kernel* ke seluruh citra secara bergeser. CNN juga merupakan salah satu algoritma pembelajaran *deep learning* yang paling kuat untuk memodelkan proses yang kompleks dan melakukan pengenalan pola dalam aplikasi dengan data dalam jumlah besar dan pengenalan pola dalam gambar [6]. Berdasarkan tinjauan pustaka jurnal [11] berikut penjelasan masing-masing *layer* dan *function*-nya:

- *Convolutional layer*, *layer* ini bertanggung jawab untuk mengekstraksi fitur yang terdapat pada sebuah citra. Layer ini akan menghasilkan citra yang diperkecil dari ukuran aslinya. Umumnya, *convolutional layer* akan memiliki lebih dari satu filter, misalnya jika ada empat filter yang digunakan, maka akan ada empat *neuron* yang akan melihat bidang reseptif yang sama. Dengan cara ini, *overfitting* dapat dikontrol dan setiap filter akan mencoba menyesuaikan dengan satu fitur pada setiap posisi spasial.

  Parameter yang terdapat pada *layer* ini antaralain *stride* dan *padding*. *Stride* merupakan berapa *pixel* bidang reseptif akan berjalan, dengan arti lain parameter yang menentukan berapa jumlah pergeseran filter. Semakin kecil *stride*, maka informasi yang didapatkan dari sebuah input akan semakin detail, semakin besar juga komputasi yang dibutuhkan. *Padding* di sini akan menambahkan batas *pixel* karena *padding* yang menentukan jumlah *pixel* (berisi nilai 0), ini digunakan untuk menjaga dimensi *output* dari *convolutional layer* (*Feature Map*).
- *Pooling layer*, *layer* ini bertanggungjawab untuk mengontrol dimensi data dengan *downsampling* data, hal ini berfungsi untuk

mengontrol *overfitting*. Dalam arti lain, *pooling layer* berperan dalam mereduksi dimensi dari sebuah *output*. Pada umumnya, penggunaan *pooling layer* ini menggunakan *Max* atau *Average pooling*. Dalam *Max pooling*, dari setiap reseptif *field* akan diambil hanya akan diambil *value* yang terbesar untuk mendeskripsikan *field* yang ada. Sedangkan *Average pooling* mengambil rata-rata dari bidang reseptifnya (*kernel*).

- *Activation function*, fungsi aktivasi (*non-linear*) adalah pemetaan *input* ke *output*. Nilai *input* ditentukan dengan menghitung penjumlahan terbobot (*weight*) dari input *neuron* beserta biasnya (jika ada). Ini berarti bahwa fungsi aktivasi membuat keputusan apakah akan membebaskan *neuron* dengan mengacu pada input tertentu atau tidak dengan membuat *output* yang sesuai.

  Non-linear activation layer digunakan setelah semua *layer* dengan bobot (disebut dengan *layer* yang dapat dipelajari, seperti *fully connected layer* dan *convolutional layer*) dalam arsitektur CNN. Performa *non-linear* dari *activation layer* ini berarti pemetaan *input* ke *output* akan menjadi *non-linear*; selain itu, *layer-layer* ini memberi CNN kemampuan untuk mempelajari hal-hal yang sangat rumit. Beberapa *activation layer* pada CNN antaralain: sigmoid, tanh, ReLU, dan sebagainya.
- *Fully connected layer*, hasil *feature map* pada *convolutional layer* masih dalam bentuk *array* multidimensi, sehingga harus mengubahnya menjadi *array* satu dimensi (vektor) dengan melakukan *flatten* atau disebut juga *reshape feature map* agar dapat digunakan sebagai input dari *fully connected layer*.
- *Loss Function*, pada proses akhir klasifikasi di *output layer*, *loss function* digunakan untuk menghitung kesalahan prediksi yang dibuat di seluruh *training samples* dalam model CNN. Kesalahan

ini mengungkapkan perbedaan antara *output* sebenernya dan yang diprediksi.

2.2.3 *Imbalanced Dataset*

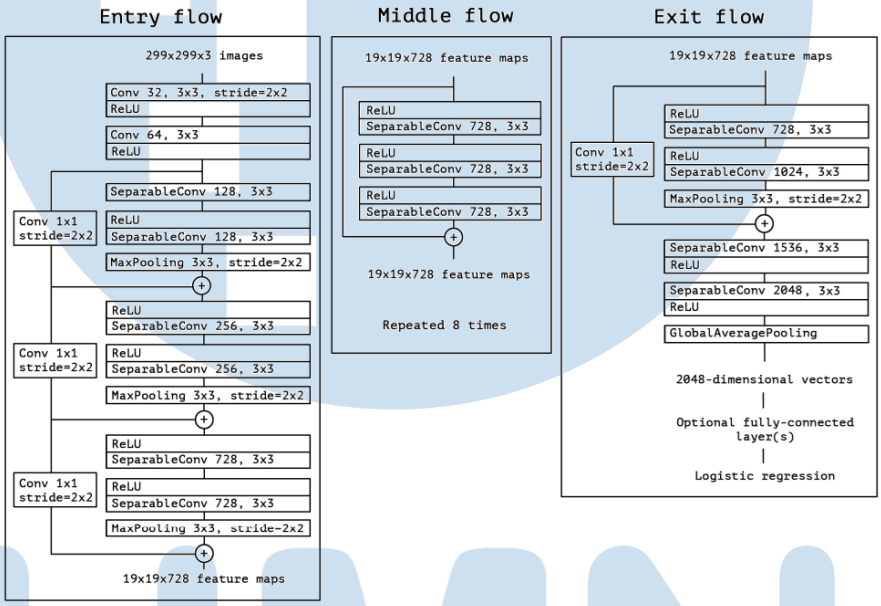
Imbalanced data atau ketidakseimbangan data yang dikenal dengan istilah ketidakseimbangan kelas merupakan kondisi yang menggambarkan tidak seimbangnya porsi data latih antara satu kelas dengan kelas lainnya. Distribusi kelas yang memiliki persentase lebih kecil dari kelas lainnya disebut kelas minoritas sedangkan kelas yang memiliki persentase besar dari semua data adalah kelas mayoritas [12]. Terdapat dua teknik umum untuk memecahkan masalah ketidakseimbangan kelas yaitu *undersampling* dan *oversampling*. *Undersampling* adalah proses secara acak mengurangi beberapa jumlah sampel dari kelas mayoritas untuk mencocokkan angka dengan kelas minoritas. Sedangkan *oversampling* adalah meningkatkan jumlah sampel dalam kelas minoritas [13]. Pada penelitian ini *oversampling* dipilih karena jumlah sampel pada kelas minoritas (*fungivore*) sangat sedikit yaitu sebanyak 80 citra nematoda non parasitik, apabila melakukan *undersampling*, sampel untuk pelatihan akan tidak cukup banyak dan kurang beragam, jadi model mungkin tidak memiliki informasi yang cukup untuk mempelajari pola dasar dataset. Pada penelitian ini *oversampling* dilakukan dengan cara melakukan augmentasi data dengan peningkatan cahaya (*brightness*).

2.2.4 Xception

Xception merupakan jaringan saraf konvolusional yang sepenuhnya didasarkan pada *depthwise separable convolution*. *depthwise separable convolution* disebut juga *separable convolution* terdiri dari *depthwise convolution* dan *pointwise convolution*. Pada konvolusi *depthwise*, setiap *channel* input diterapkan pada filter konvolusi terpisah secara independen. Ini memungkinkan deteksi fitur yang lebih baik karena setiap *channel* dapat fokus pada fitur-fiturnya sendiri. Pada konvolusi *spatial*, yaitu konvolusi 1x1,

memproyeksikan *output channel* dengan konvolusi *depthwise* ke *channel* baru.

Arsitektur Xception memiliki 36 lapisan konvolusional membentuk dasar jaringan *feature extractor*. 36 lapisan konvolusi disusun menjadi 14 modul, yang semuanya memiliki koneksi residual linier di sekitarnya, kecuali modul pertama dan terakhir. Xception juga memiliki blok residual, koneksi residual berperan penting dalam membantu konvergensi, di mana mengurangi jumlah parameter dan mempercepat pelatihan jaringan.

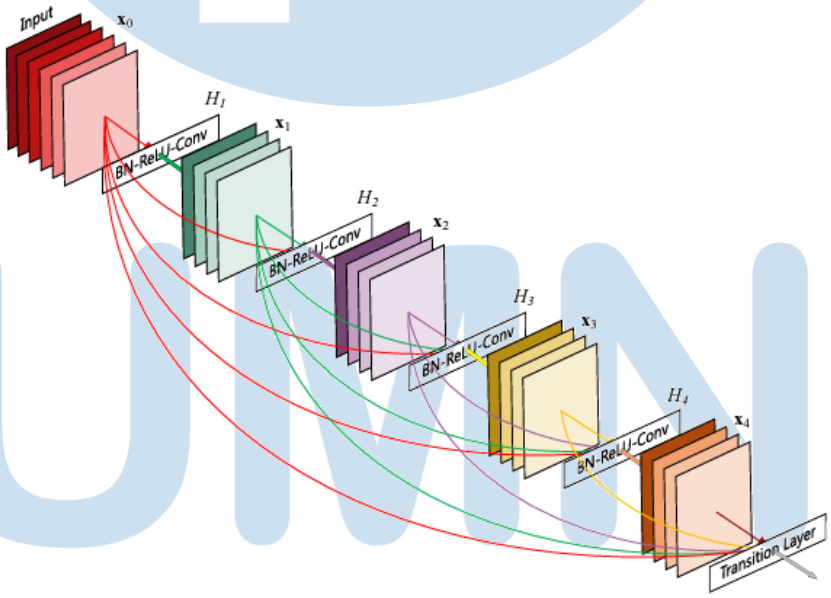


Gambar 2. 7 Arsitektur Xception [14]

Arsitektur Xception: data pertama melewati *entry flow*, kemudian melalui *middle flow* yang diulang delapan kali, dan terakhir melalui *exit flow*. Semua layer *Convolution* dan *SeparableConvolution* diikuti oleh *Batch Normalization* (tidak termasuk dalam diagram). Semua lapisan *SeparableConvolution* menggunakan pengganda kedalaman 1 (tanpa perluasan kedalaman) [14].

2.2.5 DenseNet

Konsep utama dalam DenseNet adalah koneksi langsung (*direct connection*) antara setiap lapisan, yang memungkinkan informasi dari lapisan sebelumnya langsung mengalir ke lapisan selanjutnya. Pada setiap lapisan, *input* dari lapisan sebelumnya dihubungkan secara langsung dengan lapisan tersebut dengan menggabungkan (*concatenating*) fitur dari lapisan sebelumnya dengan fitur saat ini. Dengan melakukan *concatenation* pada setiap lapisan, DenseNet dapat membangun jalur koneksi yang lebih kaya dan memungkinkan aliran informasi yang lebih langsung dan lebih efisien melalui jaringan. Hal ini membantu mengatasi masalah gradien yang melemah (*vanishing gradient*) dan mempromosikan pembelajaran fitur yang lebih baik dalam jaringan yang dalam [15].

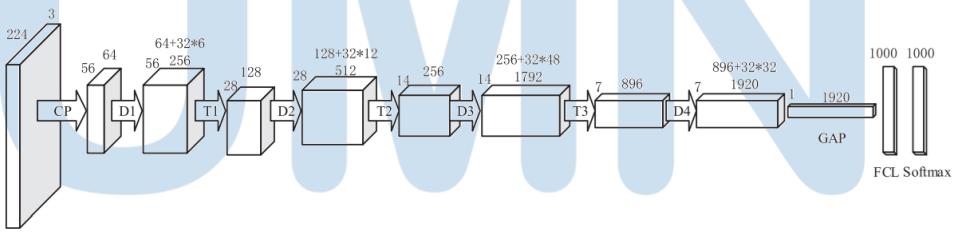


Gambar 2. 8 5 layer dense block [15]

Secara umum, DenseNet-201 memiliki arsitektur yang dalam (*deep*) dan kompleks dengan total 201 lapisan. Arsitektur DenseNet-201 terdiri dari

beberapa blok yang ditumpuk secara berurutan. Setiap blok terdiri dari beberapa lapisan konvolusi, biasanya dengan konvolusi 3x3, yang diikuti oleh fungsi aktivasi seperti ReLU.

Dalam DenseNet-201, struktur blok yang digunakan adalah "*Bottleneck layers*". *Bottleneck layer* terdiri dari tiga operasi, yaitu konvolusi 1x1 yang mengurangi dimensi fitur, konvolusi 3x3 yang dilakukan pada dimensi yang telah dikurangi, dan konvolusi 1x1 lagi yang mengembalikan dimensi fitur ke ukuran awal. Struktur *Bottleneck* ini membantu mengurangi jumlah parameter dan komputasi yang diperlukan dalam model. Selain itu, DenseNet-201 juga menggunakan teknik reduksi dimensi dengan memasukkan blok reduksi *(reduction block*) di antara blok-blok DenseNet. Blok reduksi ini membantu mengurangi dimensi fitur untuk mengontrol kompleksitas model dan mempercepat proses komputasi. Dengan kombinasi dari *concatenation*, *bottleneck layers*, dan reduksi dimensi, DenseNet-201 mampu membangun model yang sangat dalam dengan jumlah parameter yang relatif lebih sedikit dibandingkan dengan arsitektur lain. Pada standar ConvNet, *classifier* menggunakan fitur yang paling kompleks, di DenseNet, *classifier* menggunakan fitur dari semua tingkat kompleksitas. Ini cenderung memberikan batas keputusan yang lebih halus. Ini juga menjelaskan mengapa DenseNet bekerja dengan baik saat data pelatihan tidak mencukupi [16].

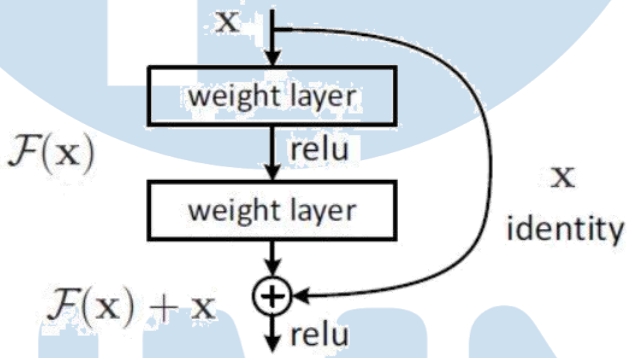


Gambar 2. 9 Arsitektur DenseNet201 [17]

2.2.6 ResNet

Konsep utama dalam ResNet adalah blok residu (*residual block*), yang mengatasi masalah pemudaran gradien (*vanishing gradient*) yang sering terjadi dalam jaringan yang sangat dalam. Pada dasarnya, blok residu adalah modul yang berfungsi untuk mengidentifikasi perbedaan atau "residu" antara *input* dan *output* lapisan. Blok Residu dibuat untuk mengatasi masalah gradien yang pelan-pelan menghilang/meledak.

Residual Connection, alih-alih mencoba membuat *layer* mempelajari fungsi identitas, idenya adalah membuat *input* dari *layer* sebelumnya tetap sama secara default, dan hanya mempelajari apa yang perlu diubah. Oleh karena itu, setiap fungsi tidak perlu belajar banyak dan pada dasarnya akan menjadi fungsi identitas [18].



Gambar 2. 10 *Building block residual learning* [18]

Lapisan-lapisan dalam jaringan residu lebih kecil daripada model VGG-19. Ada juga lebih banyak lapisan, tetapi mereka tidak perlu belajar banyak sehingga jumlah parameter lebih kecil. Alih-alih melakukan operasi *pooling*, jaringan saraf residu juga menggunakan *stride* dua. Dengan menambahkan lapisan, jaringan saraf polos 34 lapisan sederhana sebenarnya kehilangan performa, tetapi masalah ini dapat diatasi dengan menambahkan *skip connection*.

Menambahkan *skip connection* menciptakan masalah lain, setelah setiap konvolusi dengan *stride* dua, output memiliki ukuran setengah dari sebelumnya, dan pada saat yang sama jumlah filter dalam konvolusi berikutnya dua kali lebih besar dari yang sebelumnya. Tiga ide dieksplorasi untuk memecahkan masalah ini. Pertama adalah menambahkan *zero padding*, yang kedua adalah menambahkan konvolusi 1x1 ke koneksi-koneksi tertentu (yang ditandai dengan titik-titik), dan yang terakhir adalah menambahkan konvolusi 1x1 ke setiap koneksi. Ide yang lebih populer adalah yang kedua, karena yang ketiga tidak memberikan peningkatan yang signifikan dibandingkan dengan opsi kedua dan menambahkan lebih banyak parameter.

Ketika jaringan saraf semakin dalam, itu menjadi lebih mahal secara komputasi. Untuk memperbaiki masalah ini, mereka memperkenalkan *bottleneck block*. Ini memiliki tiga lapisan, dua lapisan dengan konvolusi 1x1, dan lapisan ketiga dengan konvolusi 3x3. Lapisan 1x1 pertama bertanggung jawab untuk mengurangi dimensi dan yang terakhir bertanggung jawab untuk mengembalikan dimensi, meninggalkan lapisan 3x3 dengan dimensi *input/output* yang lebih kecil dan mengurangi kerumitannya. Menambahkan lapisan 1x1 bukanlah masalah karena intensif komputasinya jauh lebih rendah daripada lapisan 3x3.

Gambar 2. 11 *A deeper residual function* [18]

Dengan desain *bottleneck*, ResNet 34 *layer* menjadi ResNet 50 *layer*, dan ada jaringan yang lebih dalam dengan desain *bottleneck*: ResNet-101 dan ResNet-152. Arsitektur keseluruhan untuk semua jaringan adalah sebagai berikut:

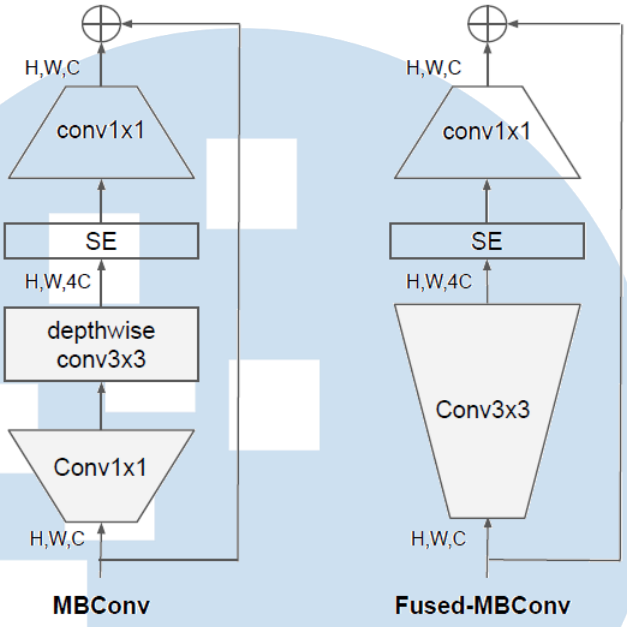
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Gambar 2. 12 Arsitektur *building blocks* ResNet [18]

Pada penelitian ini, penulis menggunakan ResNet50 untuk model ketiga.

2.2.7 EfficientNetV2

Efficient-V2 adalah *state-of-the-art* dari keluarga EfficientNet (dirilis oleh Google Research pada 2021) yang memiliki kecepatan pelatihan lebih cepat dan efisiensi parameter lebih baik. EfficientNet-V2 menyertakan kombinasi blok konvolusional yaitu, blok MBConv dan blok Fused MBConv yang keduanya menggunakan lapisan pemerasan dan eksitasi. Blok MBConv menggunakan lapisan *depth-wise* konvolusional 3x3 sementara blok Fused MBConv menggunakan lapisan konvolusional 3x3.

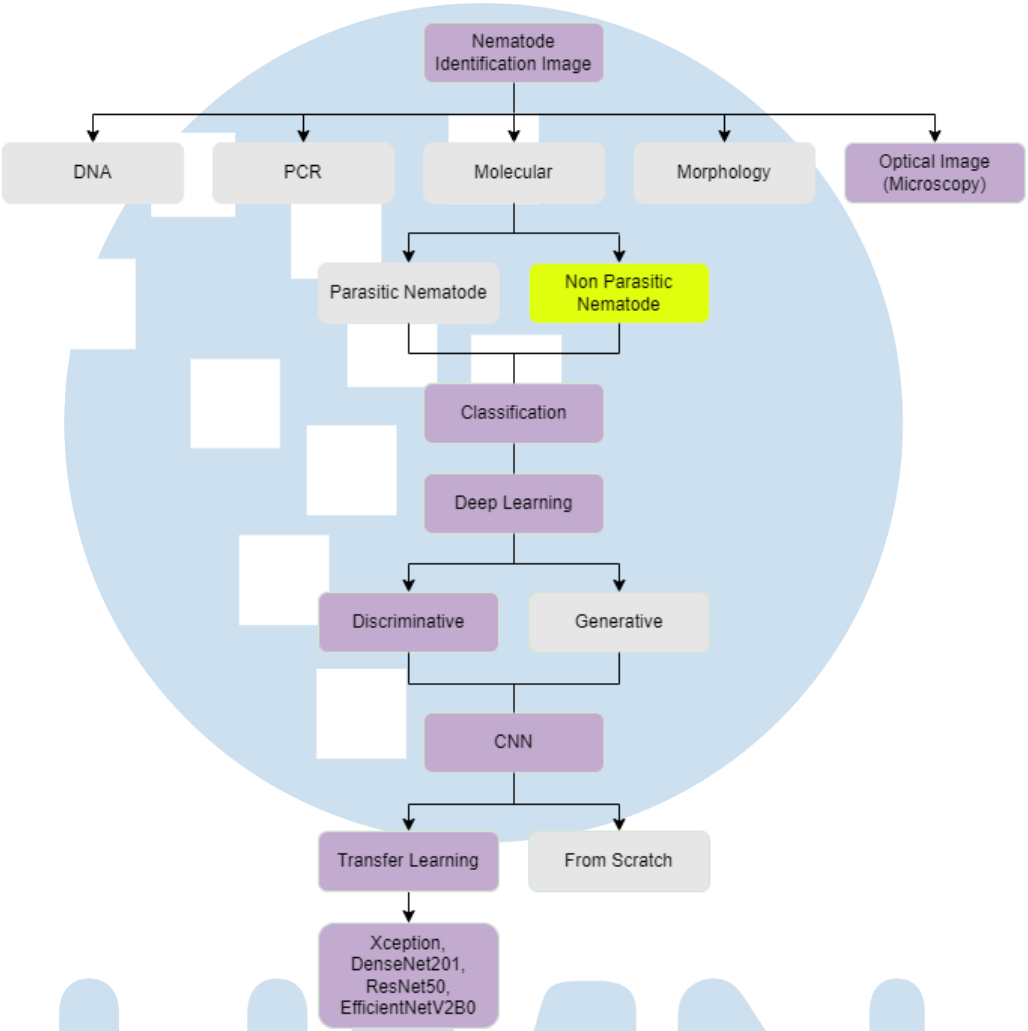


Gambar 2. 13 Struktur MBConv and Fused-MBConv [19]

Fused-MBConv diintegrasikan ke dalam EfficientNetV2 untuk mengatasi kelemahan EfficientNet-B0 layer MBConv pada tahap awal tetapi efektif pada tahap akhir. Meskipun memiliki jumlah parameter dan FLOP yang lebih rendah daripada layer *deep learning* lainnya, layer MBConv seringkali tidak dapat memanfaatkan *accelerator* terkini karena kompleksitasnya.

2.3 State-of-the-art

Seperti pembahasan pada Bab I, karena belum adanya penelitian secara spesifik yang membahas mengenai nematoda non parasitik berdasarkan trofiknya, pada penelitian ini penulis akan mencoba melakukan pengklasifikasian nematoda non parasitik dengan menggunakan model *pre-trained* CNN yang dipilih berdasarkan penelitian terdahulu yang sudah disebutkan pada Bab I. *State-of-the-art* pada penelitian ini ditunjukkan dengan warna kuning pada bagan di Gambar 2.15.



Gambar 2. 14 *State-of-the-art* penelitian