Learning Tomato-Leaf Dataset using Inception-V3 Architecture

[1]Denny Pratama Hardiono, [2]Fadhillah Putri Taha [3]Yustika Wahyuni
[1]H071171511, [2]H071171301, [3]H071171529
[1]dennypratama945@gmail.com & [2]dhilataha9@gmail.com
Program Studi Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Hasanuddin

1. Pendahuluan

Tomat adalah tanaman yang dibudidayakan secara luas di seluruh dunia, yang mengandung nutrisi yang kaya, rasa yang unik, dan efek kesehatan, sehingga ia memainkan peran penting dalam produksi pertanian dan perdagangan di seluruh dunia. Diberikan kredit dapur yang penting dalam konteks ekonomi, itu diperlukan untuk memaksimalkan daya tahan dan produktivitas produk dengan menggunakan teknik. Penyakit bercak daun Corynespora, penyakit busuk daun awal, penyakit busuk daun, penyakit jamur daun, bercak daun septoria, tungau laba-laba dua-bintik, penyakit virus, dan penyakit keriting daun kuning adalah 8 penyakit umum pada tomat. dengan demikian, teknologi pengenalan waktu dan tepat waktu sangat penting [1-8].

Baru-baru ini, karena CNN memiliki mekanisme belajar sendiri, yaitu, mengekstraksi fitur dan mengklasifikasikan gambar dalam satu prosedur, CNN telah berhasil menerapkan aplikasi beragam, seperti identifikasi penulis, deteksi objek yang menonjol, deteksi teks adegan, inference terpotong pembelajaran, roadcrackdetection, analisis gambar biomedis, memprediksi atribut wajah dari gambar web, dan deteksi pejalan kaki, dan mencapai kinerja yang lebih baik. Selain itu, CNN mampu mengekstraksi fitur yang lebih kuat dan diskriminatif dengan mempertimbangkan informasi konteks global wilayah, dan CNN hampir tidak terpengaruh oleh bayangan, distorsi, dan kecerahan gambar alami. Dengan perkembangan pesat CNN, banyak arsitektur CNN yang kuat muncul, seperti AlexNet, GoogLeNet, VGGNet, Inception-V3, Inception-V4, ResNet, dan Dense Nets [1].

Pembelajaran mendalam adalah teknik canggih dan modern untuk pemrosesan gambar dan analisis data yang menghasilkan hasil yang menjanjikan. Deep Learning dan Neural Networks dikombinasikan bersama disebut Deep Neural Network dan telah diterapkan ke berbagai bidang. Jaringan saraf menyediakan pemetaan antara input (gambar tanaman yang sakit) —untuk output— (sepasang tanaman ~ penyakit). Neuron pada dasarnya adalah fungsi matematika yang menerima input numerik dan memberikan output numerik. Jaringan neural yang dalam terdiri dari lapisan input, sejumlah lapisan pemrosesan, dan lapisan keluaran [10].

Pada pembelajaran ini, kami menggunakan prediksi otomatis dari dataset daun tomat dengan menggunakan *deep convolutional neural network* berdasarkan data gambar sekunder dari Kaggle. Dalam hal ini merupakan arsitektur jaringan Inception-V3 untuk melihat nilai akurasi datasaet.

2. Tinjauan Pustaka

2.1. *State of the Art*

Pendeteksian daun tomat telah banyak dilakukan sebelumnya menggunakan metode-metode dalam *deep learning*. Diantaranya K. Saiqa, dkk [9] dari Universitas Mumbai dan Amity University Rajasthan, India, yang membagi data *training* dan data *testing* dengan perbandingan sebesar 80% 20% yang diimplementasikan pada Torch7 *framework machine learning*. Penelitian

ini menggunakan 5 model arsitektur *neural network*, yaitu AlexNet dengan akurasi 99.06%, AlexNetOWTBn dengan akurasi 99.44%, GoogleNet dengan akurasi 97.27%, Overfeast dengan akurasi 98.96%, VGG Net dengan akurasi 99.48%.

Penelitian selanjutnya pada tahun 2019 yang dilakukan oleh H. Mosin, dkk. [10] yang menggunakan drone untuk mengambil gambar dataset dan disimpan menggunakan jaringan ZigBee. Sistem yang dimilikinya mampu mengklasifikasikan gambar berdasarkan kualitas daun menjadi baik, sedang, buruk, yang kemudian menggunakan reseptor untuk menyemprotkan cairan kimia. Arsitektur yang digunakan adalah model AlexNet yang ketika pemisahan dataset pelatihan dan pengujian adalah 45-55%, memberikan akurasi 72%. Sedangkan, semakin banyak data pelatihan, maka akurasi tertinggi didapatkan pada 99%, ketika mereka memberikan 90% data *training*.

Penelitian selanjutnya oleh A. Mohit, dkk. [20] pada tahun 2019 mendapatkan akurasi klasifikasi yang bervariasi dari 76% sampai 91%, yang menggunakan arsitektur model MobileNet dengan akurasi 63.75%, VGG Net dengan akurasi 77.2%, Inception-V3 dengan akurasi 63.4%, dan proposed Model dengan akurasi 91.2%.

Pada tahun 2019 lalu juga telah terdapat penelitian dalam *research article Hindawi, Advances in Multimedia* dari Northeast Agricultural University, China oleh Z. Keke, dkk. [1] yang menggunakan konstruksi model CNN untuk mengidentifikasi daun tomat dengan *transfer learning*. Penelitian ini mendapatkan tingkat akurasi 95.83% menggunakan arsitektur AlexNet (SGD), 13.86% menggunakan arsitektur AlexNet (Adam), 95.66% menggunakan arsitektur GoogleNet (SGD), 94.06% menggunakan arsitektur GoogleNet (Adam), 96.51% menggunakan arsitektur ResNet (SGD), dan 94.39% menggunakan arsitektur ResNet (Adam).

2.2. Landasan Teori

2.2.1. Artificial Intelligence

Artificial Intelligence atau kecerdasan buatan adalah kecerdasan yang dimiliki oleh mesin yang dapat bertindak seperti layaknya manusia[11].

*T*eknik kecerdasan buatan seperti pembelajaran mesin dan pembelajaran mendalam menjadi inti dari sistem CAD canggih dalam banyak aplikasi medis; misalnya, penyakit paru-paru [12], kardiologi, dan operasi otak [13].

Alat analisis gambar CT otomatis berbasis Artificial Intelligence (AI) untuk deteksi, kuantifikasi dan pemantauan coronavirus dan untuk membedakan pasien dengan coronavirus dari bebas penyakit telah dikembangkan [14].

2.2.2. Deep Learning

Teknik pembelajaran mendalam yang digunakan dalam beberapa tahun terakhir terus menunjukkan kinerja yang mengesankan di bidang pemrosesan citra medis, seperti di banyak bidang. Dengan menerapkan teknik pembelajaran mendalam pada data medis, dicoba untuk menarik hasil yang bermakna dari data medis[15].

Model pembelajaran dalam telah berhasil digunakan di banyak bidang seperti klasifikasi, segmentasi dan deteksi lesi data medis. Analisis data gambar dan sinyal diperoleh dengan teknik pencitraan medis seperti Magnetic Resonance Imaging (MRI), Computed Tomography (CT) dan X-ray dengan bantuan model pembelajaran yang mendalam. Sebagai hasil dari analisis ini, deteksi dan diagnosis penyakit seperti diabetes mellitus, tumor otak, kanker kulit dan kanker payudara disediakan dengan mudah [15].

2.2.3. Convolutional Neural Network

Pada tahun 1998, Yann LeCun merancang sebuah arsitektur yang disebut dengan convolutional neural network yang bekerja dengan data berupa citra. Convolution neural network sendiri terdiri dari layer konvolusi yang mengekstraksi tekstur dari sebuah citra. Diikuti dengan *backpropagation* untuk memperbarui bobotnya [16].

2.2.4. Inception-V3 Architechture

Inception-V3 adalah sebuah model *deep convolutional network* yang dikembangkan oleh Goolgle memenuhi I*magNet Large Visual Recognition Challenge* pada tahun 2012. Inception memiliki empat versi, yaitu Inception-V1, Inception-V2, Inception-V3 dan Inception-V4.

Model inception menggunakan beberapa *filter* pada layer yang biasa. Hasil dari beberapa *filter* tersebut dijadikan satu lagi menggunakan *channel concat* sebelum masuk kedalam iterasi berikutnya [17].

Jaringan Inception V3 memiliki beberapa blok bangunan simetris dan asimetris, di mana setiap blok memiliki beberapa cabang konvolusi, penyatuan rata-rata, *maxpooling*, digabungkan, putus, dan lapisan yang terhubung sepenuhnya [13].

Seiring berjalannya waktu, arsitektur dari *deep learning* semakin beragam. Salah satunya adalah inception V3 yang merupakan arsitektur pertama dengan parameter yang lebih sedikit dan komputasi yang efisien [18]. Inception-V3 yang merupakan gabungan dari perbaikan versi pertama dan kedua, dimana terdapat beberapa tambahan, seoerti *RMSProp optimizer, factorized 7x7 convolutions. BatchNorm in the auxillary classifier,* dan *label smooting*[19]. Pada inception V3 ini ada faktorisasi untuk mengurangi parameter. Meskipun komputasinya sudah rendah, inception V3 tidak dapat digunakan pada perangkat dengan ruang lingkup komputasi rendah. Inception V3 [17] adalah jaringan pembelajaran 42-lapisan dengan parameter lebih sedikit. Pengurangan parameter dilakukan dengan bantuan faktorisasi konvolusi. Misalnya, konvolusi 5x5 filter dapat dilakukan oleh dua konvolusi 3x3 filter. Parameter dalam proses ini berkurang dari $5 \times 5 = 25$ menjadi $3 \times 3 + 3 \times 3 = 18$. Dengan demikian, itu membawa pengurangan 28% dari jumlah parameter. Dengan angka lebih rendah dari jumlah parameter, mereka tidak akan melebihi dan dengan demikian meningkatkan akurasi. Keakuratannya bahkan lebih rendah dari VGG16 dan mencapai 63,4% ketika kami dilatih untuk 1000 zaman di NVIDIA. Model yang kami rasa akan bekerja dengan baik pada jumlah kelas yang lebih besar karena arsitektur 42-lapisan membuatnya lebih sesuai untuk sejumlah kecil kelas ketika fitur perbedaannya tidak jelas atau besar [20].

2.2.5. Tomato-Leaf Dataset

Gambar yang dijadikan dataset merupakan data sekunder yang diambil dari Kaggle dengan nama "Plantvillage Tomato Leaf Dataset"[1]. Dataset tersebut terdiri dari 6 kategori yang berbeda dengan 9 gambar per kelas digunakan sebagai set uji. Berikut beberapa kategori yang ada pada *link* tersebut :

- Tomato Bacterial Spot : 2127 images
- Tomato Early Blight : 1000 images
- Tomato Late Blight : 1916 images
- Tomato Leaf Mold : 952 images
- Tomato Septoria Leaf Spot : 1771 images
- Tomato Yellow Leaf Curl Virus : 2740 images

[1] https://www.kaggle.com/charuchaudhry/plantvillage-tomato-leaf-dataset

2.3. Ukuran Kinerja

Kinerja yang akan dicari disini adalah untuk mencari accuracy, loss, val accuracy dan val loss yang dimana akurasi adalah metode untuk mengukur kinerja model klasifikasi. Biasanya dinyatakan sebagai persentase. Akurasi adalah hitungan prediksi di mana nilai prediksi sama dengan nilai sebenarnya. Ini biner (benar / salah) untuk sampel tertentu. Akurasi sering digambarkan dan dipantau selama fase pelatihan meskipun nilainya sering dikaitkan dengan akurasi model keseluruhan atau akhir. Loss atau Fungsi kerugian, juga dikenal sebagai fungsi biaya, memperhitungkan probabilitas atau ketidakpastian prediksi berdasarkan seberapa besar prediksi tersebut berbeda dari nilai sebenarnya. Ini memberi kita pandangan yang lebih bernuansa tentang seberapa baik kinerja model.

Tidak seperti akurasi, kehilangan bukanlah persentase - itu adalah penjumlahan dari kesalahan yang dibuat untuk setiap sampel dalam pelatihan atau set validasi. Kehilangan sering digunakan dalam proses pelatihan untuk menemukan nilai parameter "terbaik" untuk model.

Dan Untuk validation loss dan validation accuracy memiliki keterikatan yang dimana

- Validation loss mulai meningkat dan Validation accuracy mulai berkurang. Ini berarti model mementingkan nilai-nilai.
- Validation loss mulai meningkat dan Validation accuracy juga meningkat. Ini bisa berupa overfitting atau beragam nilai probabilitas dalam kasus di mana softmax sedang digunakan di lapisan output.
- Validation loss mulai berkurang dan Validation accuracy mulai meningkat. Ini juga bagus karena model yang dibangun bekerja dengan baik.

	Prediksi Positif	Prediksi Negatif
Aktual Positif	True Positive	False Negative
Aktual Negatif	False Positive	True Negative

Tabel 1 Confussion Matrix

a) Akurasi

Akurasi adalah metrik paling penting untuk hasil pengklasifikasian pembelajaran kita yang mendalam, seperti yang diberikan pada rumus berilut. Model yang paling dapat diandalkan adalah yang terbaik tetapi penting untuk memastikan bahwa ada dataset simetris dengan nilai positif palsu hampir sama dan nilai negatif palsu.

$$\text{Akurasi} = \frac{TP + TN}{TP + FP + FN + TN} X\ 100\%$$

b) Presisi

Presisi direpresentasikan pada rumus berikut, untuk memberikan hubungan antara nilai prediksi positif sejati dan nilai prediksi positif penuh.

$$\text{Presisi} = \frac{TP}{TP + FP}$$

c) Recall

Recall atau sensitivitas adalah rasio antara nilai positif sejati prediksi dan penjumlahan dari nilai positif sejati yang diprediksi dan prediksi nilai negatif palsu.

$$\text{Presisi} = \frac{TP}{TP + FN}$$

d) F1-score

Skor F1 adalah ukuran keseluruhan dari keakuratan model yang menggabungkan ketepatan dan daya ingat, sebagaimana direpresentasikan pada rumus berikut. Skor F1 adalah dua kali rasio antara perkalian dengan penjumlahan metrik presisi dan *recall*.

$$\text{F1}-\text{score} = 2(\text{Presisi} \times \text{RecallPresisi} + \text{Recall})$$

3. Implementasi

Analisis ini dilakukan menggunakan bahasa pemrograman Python pada Google Colaboratory dengan menggunakan laptop Lenovo x230 dengan Processor Core (TM) i5, dengan RAM sebesar 6.00 GB dengan menggunakan Operating System Windows 10 Pro 64-bit.

Pada percobaan kali ini, kita akan menoba mengimplementasikan aritektur CNN yaitu Inception V3 dengan menggunakan dataset Tomato yang tersimpan di Google Drive

Percobaan kali ini, menggunakan data Tomato yang disimpan di Google Drive sebelumnya. Kemudian, dibuat model arsitektur untuk Inception V3. Pada percobaan ini, kami menurunkan versi TensorFlow dan Keras yang akan digunakan menjadi TensorFlow 2.0. Kemudian kita menggunakan kode berikut untuk membuat model dari arsitektur Inception V3: Pertama, meng*import* data mana yang digunakan, yang telah disimpan pada GoogleDrive.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] %cd drive/'My Drive'
```

```
[Errno 2] No such file or directory: 'drive/My Drive'
/content/drive/My Drive
```

Selanjutnya, meng*import package* apa saja yang digunakan pada model Inception V3.

```
[ ] import numpy as np
    import pickle
    import cv2
    import keras
    import tensorflow as tf

    from keras.applications.inception_v3 import InceptionV3
    from keras.preprocessing import image
    from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
    from keras.layers import Flatten, Dense
    from keras.models import Model
    from keras.layers import BatchNormalization

    from os import listdir
    from keras import backend as K
    from keras.layers import Input
    from keras.optimizers import Adam
    from keras.preprocessing.image import ImageDataGenerator
    from keras.preprocessing import image
    from keras.preprocessing.image import img_to_array
    from sklearn.preprocessing import MultiLabelBinarizer
    from sklearn.preprocessing import LabelBinarizer
    from sklearn.model_selection import train_test_split

    import matplotlib.pyplot as plt
```

Selanjutnya, mengkonvert gambar menjadi sebuah array dengan *epoch* sebanyak 50 kali pembelajaran, dan menggunakan *directory-root* dari dataset daun tomat, bernama "Dataset Tomato"[2] sesuai dengan nama folder pada Google Drive

```
[ ]  EPOCHS = 50
     INIT_LR = 1e-3
     BS = 32
     default_image_size = tuple((299, 299))
     image_size = 0
     directory_root = './Dataset Tomato/'
     width=299
     height=299
     depth=3
```

```
[ ]  def convert_image_to_array(image_dir):
         try:
             image = cv2.imread(image_dir)
             if image is not None :
                 image = cv2.resize(image, default_image_size)
                 return img_to_array(image)
             else :
                 return np.array([])
         except Exception as e:
             print(f"Error : {e}")
             return None
```

.

Berikut Fungsi untuk membuat gambar, yaitu *image list* memproses data apa saja yang terdapat pada dataset Tomato, dengan *wall time* selama 519 ds.

```
[ ]                 plant_disease_folder_list.remove(disease_folder)

            for plant_disease_folder in plant_disease_folder_list:
                print(f"[INFO] Processing {plant_disease_folder} ...")
                plant_disease_image_list = listdir(f"{directory_root}/{plant_folder}/{plant_disease_folder}/")

                for single_plant_disease_image in plant_disease_image_list :
                    if single_plant_disease_image == ".DS_Store" :
                        plant_disease_image_list.remove(single_plant_disease_image)

                for image in plant_disease_image_list[:200]:
                    image_directory = f"{directory_root}/{plant_folder}/{plant_disease_folder}/{image}"
                    if image_directory.endswith(".jpg") == True or image_directory.endswith(".JPG") == True:
                        image_list.append(convert_image_to_array(image_directory))
                        label_list.append(plant_disease_folder)
        %time print("[INFO] Image loading completed")
    except Exception as e:
        print(f"Error : {e}")
```

```
[INFO] Loading images ...
[INFO] Processing Tomato___Bacterial_spot ...
[INFO] Processing Tomato___Early_blight ...
[INFO] Processing Tomato___Late_blight ...
[INFO] Processing Tomato___Leaf_Mold ...
[INFO] Processing Tomato___Septoria_leaf_spot ...
[INFO] Processing Tomato___Tomato_Yellow_Leaf_Curl_Virus ...
[INFO] Image loading completed
CPU times: user 454 µs, sys: 57 µs, total: 511 µs
Wall time: 519 µs
```

Selanjutnya adalah kode untuk normalisasi Array pada Gambar yang dimana untuk batas yang diberikan yaitu 225.0

[2] https://drive.google.com/drive/folders/1lDz677HuCbqaAcfGS4bmxEjkzTOd1_tA?usp=sharing

```
[ ]  np_image_list = np.array(image_list, dtype=np.float16) / 225.0
```

Selanjutnya, memberikan label pada gambar yang dimana label yang digunakan adalah label *binariser.classes.*

```
[ ]  image_size = len(image_list)
     label_binarizer = LabelBinarizer()
     image_labels = label_binarizer.fit_transform(label_list)
     pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))
     n_classes = len(label_binarizer.classes_)
```

Data testing dan data training dengan *testing*nya diberi batas 0.2

```
[ ]  x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.2, random_state = 8)
```

Selanjutnya, menjalankan data *augmentation*, *image* yang digunakan adalah *image* data generator

```
[ ]  aug = ImageDataGenerator(
         rotation_range=25, width_shift_range=0.1,
         height_shift_range=0.1, shear_range=0.2,
         zoom_range=0.2,horizontal_flip=True,
         fill_mode="nearest")
```

Untuk menjalankan kode *base model*, kami menggunakan versi TensorFlow ke versi 2.0 agar dapat mengatur *shape* yang terdiri dari panjang, lebar dan tinggi data

.

```
[ ]  base_model = InceptionV3(weights=None, include_top=False, input_tensor=Input(shape = (width, height, depth)))

     x = base_model.output
     output = BatchNormalization()(x)
     x = GlobalAveragePooling2D()(x)
     x = Dense(1024, activation='relu')(x)
     predictions = Dense(1, activation='softmax')(x)

     model = Model(inputs=base_model.input, outputs=predictions)
```

Selanjutnya, menampilkan s*ummary model* yang digunakan pada Inception V3 untuk dataset Tomato

```
[ ] print(model.summary())
```

```
conv2d_89 (Conv2D)              (None, 8, 8, 384)    442368      activation_87[0][0]
__________________________________________________________________________________________________
conv2d_92 (Conv2D)              (None, 8, 8, 384)    442368      activation_91[0][0]
__________________________________________________________________________________________________
conv2d_93 (Conv2D)              (None, 8, 8, 384)    442368      activation_91[0][0]
__________________________________________________________________________________________________
average_pooling2d_9 (AveragePoo (None, 8, 8, 2048)   0           mixed9[0][0]
__________________________________________________________________________________________________
conv2d_86 (Conv2D)              (None, 8, 8, 320)    655360      mixed9[0][0]
__________________________________________________________________________________________________
batch_normalization_88 (BatchNo (None, 8, 8, 384)    1152        conv2d_88[0][0]
__________________________________________________________________________________________________
batch_normalization_89 (BatchNo (None, 8, 8, 384)    1152        conv2d_89[0][0]
__________________________________________________________________________________________________
batch_normalization_92 (BatchNo (None, 8, 8, 384)    1152        conv2d_92[0][0]
__________________________________________________________________________________________________
batch_normalization_93 (BatchNo (None, 8, 8, 384)    1152        conv2d_93[0][0]
__________________________________________________________________________________________________
conv2d_94 (Conv2D)              (None, 8, 8, 192)    393216      average_pooling2d_9[0][0]
__________________________________________________________________________________________________
batch_normalization_86 (BatchNo (None, 8, 8, 320)    960         conv2d_86[0][0]
__________________________________________________________________________________________________
activation_88 (Activation)      (None, 8, 8, 384)    0           batch_normalization_88[0][0]
__________________________________________________________________________________________________
activation_89 (Activation)      (None, 8, 8, 384)    0           batch_normalization_89[0][0]
__________________________________________________________________________________________________
activation_92 (Activation)      (None, 8, 8, 384)    0           batch_normalization_92[0][0]
__________________________________________________________________________________________________
activation_93 (Activation)      (None, 8, 8, 384)    0           batch_normalization_93[0][0]
__________________________________________________________________________________________________
```

Ini merupakan *training model* yang digunakan untuk mengetahui accuracy dan loss pada epoch 50 untuk dataset Tomato

```
[ ] opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
    # distribution
    model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])
    # train the network
    print("[INFO] training network...")
```

```
[INFO] training network...
```

```
[ ] history = model.fit_generator(
        aug.flow(x_train, y_train, batch_size=BS),
        validation_data=(x_test, y_test),
        steps_per_epoch=len(x_train) // BS,
        epochs=EPOCHS, verbose=1
        )
```

4. Hasil dan Pembahasan

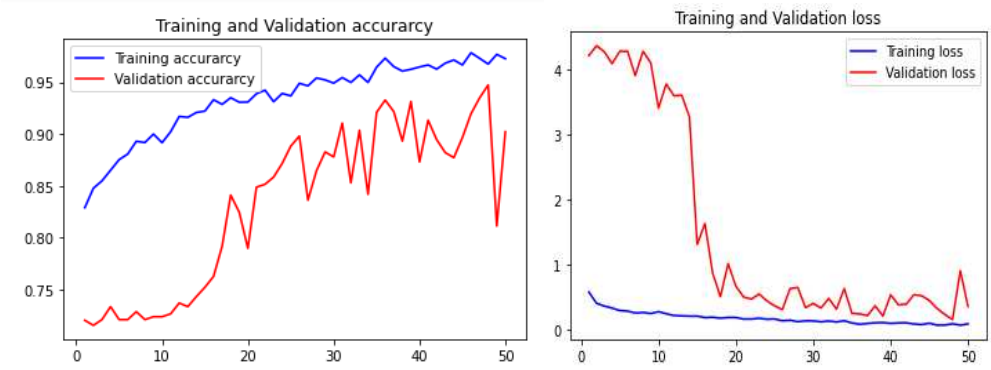
a. Perbandingan Akurasi Epoch 50 dan 100

Dalam penelitian ini menggunakan dataset daun tomat dalam pengaplikasian Inception-V3. *Epoch* atau iterasi pembelajarn dilakukan sebanyak 50 kali dan 100 kali dengan keluaran berupa tingkat akurasi dan *loss* dari arsitektur Inception-V3 terhadap dataset daun tomat.

1) Epoch 50

Berikut merupakan hasil yang kita dapat setelah melakukan *epoch*/perulangan sebanyak 50kali

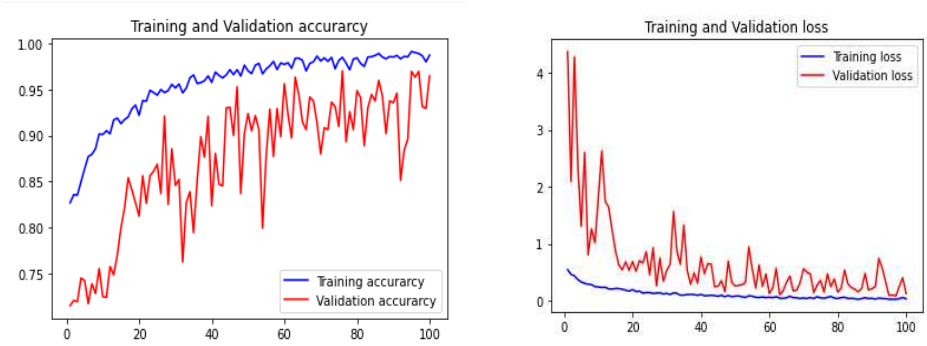
.



Dari gambar tersebut, hasil yang didapatkan kurang memuaskan. Dimana tingkat akurasi yang kit a dapatkan cukup tinggi yaitu hanya berkisar 85-95% sementara tingkat lossnya memiliki tingkat akurasi cukup tinggi

2) Epoch 100

Berikut merupakan hasil yang didapatkan setelah melakukan *epoch*/perulangan pembelajaran sebanyak 100 kali. Berikut grafiknya.



Dari gambar tersebut, hasil yang didapatkan kurang memuaskan. Dimana tingkat akurasi ya ng kita dapatkan cukup tinggi yaitu hanya berkisar 97-98% sementara tangka lossnya memiliki tingkat akurasi rendah

5. Kesimpulan

Prediksi daun pada tomat ini sangat penting untuk mengetahui penyakit apa saja yang terjangkit pada dataset tomat tersebut. Pada pembelajaran ini, kami mengusulkan pendekatan pembelajaran berbasis deep learning menggunakan gambar daun tomat (Tomato Leaf) yang diperoleh dari Kaggle dengan nama "Plantvillage Tomato Leaf Dataset. Jadi penelitian ini, melakukan pengujian dengan epoch sebanyak 50 dan 100. Dimana untuk epoch 50 didapatkan akurasi yang cukup tinggi, 85-95% sementara tingkat lossnya memiliki tingkat akurasi cukup tinggi .Sedangkan untuk epoch 100 didapatkan akurasi yang cukup tinggi antara 97-98% sementara tingkat lossnya memiliki tingkat akurasi yang cukup tinggi

6. Saran

Diharap pada penelitian selanjutnya mendapatkan menggunakan model arsitektur pembelajaran deep lainnya, sehingga mendapatkan nilai akurasi yang lebih tinggi dibandingkan dengan arsitektur Inception-V3. Penulis juga berharap agar pembelajaran ini, dapat menjadi motivasi sehingga akan ada kamera atau CCTV publik yang dapat melakukan scanning dada setiap orang dan menghasilkan gambar dada beserta kondisi orang tersebut, bahkan dalam berbagai penyakit sebagai pengembangan masa depan.

Daftar Pustaka

[1] Z. Keke, W. Qiufeng, L. Anwang, M. Xiangyan. “Can Deep Learning Identify Tomato Leaf Disease?”, Hindawi, Advances in Multimedia, vol. 2018, pp. 1, China, available on https://www.hindawi.com/journals/am/2018/6710865/, accessed 20 April 2020.

[2] A. M. Dickey, L. S. Osborne, and C. L. Mckenzie, “Papaya (Carica papaya, Brassicales: Caricaceae) is not a host plant of tomato yellow leaf curl virus (TYLCV; family Geminiviridae, genus Begomovirus),” Florida Entomologist, vol. 95, no. 1, pp. 211–213,2012, available on https://journals.flvc.org/flaent/article/view/78883, accessed 25 April 2020.

[3] G. Wei,L. Baoju,S. Yanxia, and X. Xuewen, “Studies on pathogenicity differentiation of corynespora cassiicola isolates, against cucumber, tomato and eggplant,” Acta Horticulturae Sinica, vol.38, no.3, pp.465–470, 2011, available on https://doi.org/10.1590/0100-5405/2220, accessed 25 April 2020.

[4] P. Lindhout, W. Korta, M. Cislik, I. Vos, and T. Gerlagh, “Further identification of races of Cladosporium fulvum (Fulvia fulva) on tomato originating from the Netherlands Franceand Poland,” Netherlands Journal of Plant Pathology, vol. 95, no. 3, pp. 143–148,1989, available on https://link.springer.com/article/10.1007/BF01999969, accessed 25 April 2020.

[5] K. Kubota,S. Tsuda, A. Tamai, and T. Meshi,“Tomato mosaic virus replication protein suppresses virus-targeted post transcriptional gene silencing, ” Journal of Virology, vol. 77, no. 20, pp.11016–11026, 2003, available on https://www.ncbi.nlm.nih.gov/pubmed/14512550, accessed 25 April 2020.

[6] M. Tian, B. Benedetti, and S. Kamoun, “A second Kazallike protease inhibitor from Phytophthora infestans inhibits and interacts with the apoplastic pathogenesis-relatedprotease P69B of tomato, ”Plant Physiology, vol.138, no.3, pp. 1785–1793, 2005, available on http://www.plantphysiol.org/content/138/3/1785.long, accessed 25 April 2020.

[7] L. E. Blum, “Reduction of incidence and severity of Septoria lycopersici leafs pot of tomato with bacteria and yeasts, ”Ciˆencia Rural, vol. 30, no.5, pp. 761–765, 2000, available on https://www.ncbi.nlm.nih.gov/pubmed/15980196, accessed 25 April 2020.

[8] E. A. Chatzivasileiadis and M. W. Sabelis, “Toxicity of methyl ketones from tomato trichomes to Tetranychus urticae Koch,” Experimental and Applied Acarology, vol. 21, no. 6-7, pp. 473– 484, 1997, available on https://www.researchgate.net/publication/227114743_Toxicity_of_methyl_ketones_from_tomato_trichomes_to_Tetranychu_urticae_Koch, accessed 25 April 2020.

[9] K. Saiqa, N. Meera, S. Anam Ayesha, A. Hera, A. Nida. “Disorder Detection in Tomato Plant Using Deep Learning”, International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM-2019), pp. 1, Februaty 2019, India, available on https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3358226&download=yes, accessed 25 April 2020.

[10] H. Mosin, T. Bhavesh, P. Krina J. “Deep Learning Precision Farming: Tomato Leaf Disease Detection by Transfer Learning”, 2nd INTERNATIONAL CONFERENCE ON ADVANCED COMPUTING AND SOFTWARE ENGINEERING (ICACSE-2019), pp. 174, BVM Engineering College, 2019, available on https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3349597, accessed 25 April 2020.

[11] "*Finite-State Machines: Theory and Implementation," Game Development Envato Tut*s. *Available*: tps://gamedevelopment.tutsplus.com/tutorials/finite-state-machinestheory-and-implementation--gamedev-11867. *Accessed*: 5 April 2020.

[12] C. H. Liang, Y. C. Liu, M. T. Wu, F. Garcia-Castro, A. Alberich-Bayarri, and F. Z. Wu. *Identifying pulmonary nodules or masses on chest radiography using deep learning: external validation and strategies to improve clinical practice.* Clinical Radiology, vol. 75, no. 1, pp. 38-45. *Accessed: 5 April 2020.*

[13] Ezz El-Din Hemdan, Marwa A. Shouma, Mohamed Esmail Karar, IEEE Member. *CovidX-Net: A Framework of Deep Learning Classifiers to Diagnose Covid-19 in X-Ray Images. Available:* https://arxiv.org/abs/2003.11055. *Accessed: 5 April 2020.*

[14] Gozes, O., Frid-Adar, M., Greenspan, H., Browning, P. D., Zhang, H., Ji, W., Bernheim, A., and Siegel, E. *Rapid AI Development Cycle for the Coronavirus (COVID-19) Pandemic: Initial Results for Automated Detection & Patient Monitoring using Deep Learning CT Image Analysis*. arXiv preprint arXiv:2003.05037. *Accessed: 5 April 2020.*

[15] Ali Narin, Ceren Kaya, Ziynet Pamuk. *Automatic Detection of Coronavirus Disease (COVID-19) Using X-ray Images and Deep Convolutional Neural Networks. Available:* https://arxiv.org/abs/2003.10849. Zonguldak Bulent Ecevit University, Turkey. *Accessed: 5 April 2020.*

[16] Harits Abdurrohman, Robih Dini & Arief Purnama Muharram. Evaluasi Performa metode Deep Learning untuk Klasifikasi Citra Lesi Kulit The HAM10000. Seminar Nasional Instrumentasi, Kontrol dan Otomasi (SNIKO) 2018. Bandung. *Accessed: 5 April 2020.*

[17] Nadia Ramadhani, Janson Hendryli, dan Dyah Erny Herwindianti. Pencarian Objek Wisata Bersejarah di Pulau Jawa Menggunakan *Convolutional Neural Network.* Jurnal Ilmu Komputer dan Sistem Operasi. Universitas Tarumanagara. Jakarta. *Accessed: 5 April 2020.*

[18] Szegedy, C. et al., 2016. *Rethinking the inception architecture for computer vision. Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818-2826. *Availabe*: https://www.semanticscholar.org/paper/Rethinking-the-Inception-Architecture-for-Computer-Szegedy-Vanhoucke/23ffaa0fe06eae05817f527a47ac3291077f9e58. *Accessed: 5 April 2020.*

[19] Aditya Yanuar. *Inception Network. Available:* http://machinelearning.mipa.ugm.ac.id/2018/08/10/inception-network/. *Accessed: 5 April 2020.*

[20] A. Mohit, S. Abhishek, A. Aiddartha, S. Amit, G, Suneet. "ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network", International Conference on Computational Intelligence and Data Science (ICCIDS 2019), pp. 297, India, 2020, available on (http://creativecommons.org/licenses/by-nc-nd/4.0, accessed 25 April 2020.