Model Documentation: Strawberry Leaf Health Classifier

🧠 Model Purpose

This Convolutional Neural Network (CNN) is designed to classify images of strawberry leaves as either:

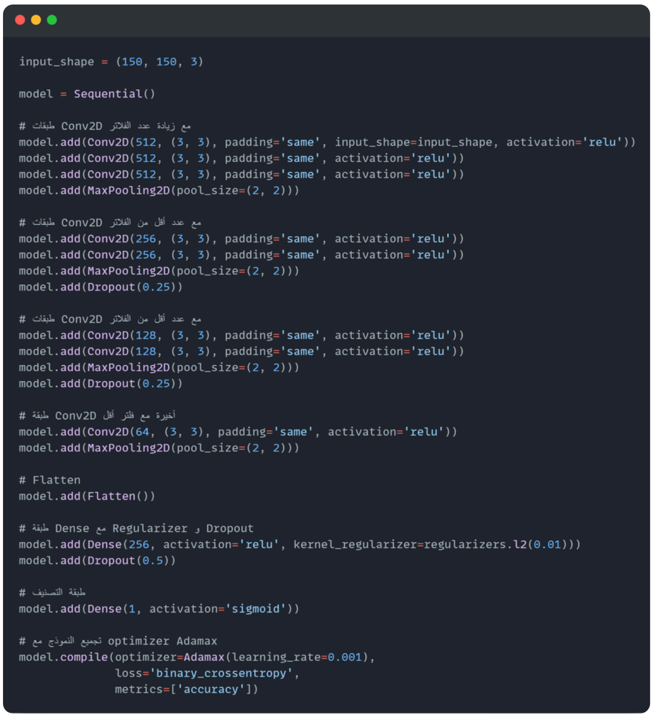
- ✅ Healthy
- ❌ Unhealthy

If the leaf is predicted to be unhealthy, a secondary detection stage (e.g., Roboflow) can classify the specific disease affecting the leaf.

📦 Model Details

Attribute	Value
Model Type	Binary Classification CNN
Input Shape	(height, width, 3) (RGB)
Output	Single sigmoid value in [0,1]
Threshold	0.75 (above = healthy)
File Format	Keras HDF5 .h5
Filename	my_model.h5

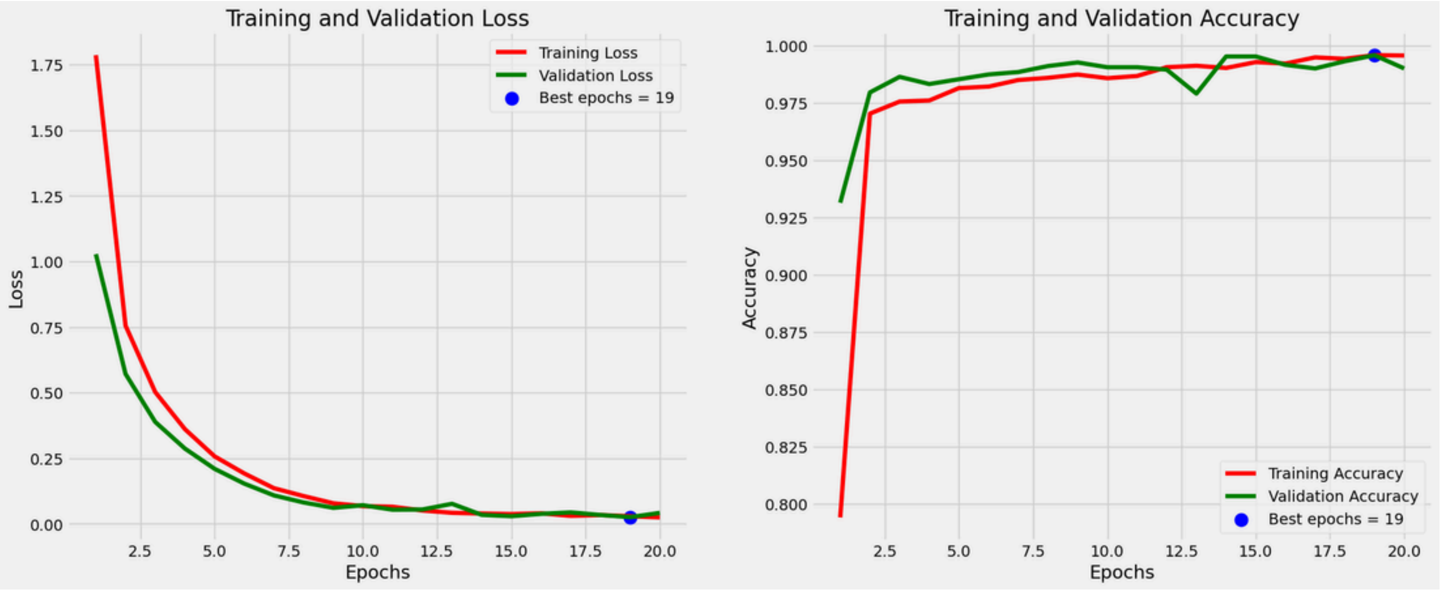
🔧 Model Layers



```
input_shape = (150, 150, 3)

model = Sequential()

# طبقات Conv2D مع زيادة عدد الفلاتر
model.add(Conv2D(512, (3, 3), padding='same', input_shape=input_shape, activation='relu'))
model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# طبقات Conv2D مع عدد أقل من الفلاتر
model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# طبقات Conv2D مع عدد أقل من الفلاتر
model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# طبقة Conv2D أخيرة مع فلتر أقل
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten
model.add(Flatten())

# طبقة Dense مع Regularizer و Dropout
model.add(Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dropout(0.5))

# طبقة التصنيف
model.add(Dense(1, activation='sigmoid'))

# تجميع النموذج مع optimizer Adamax
model.compile(optimizer=Adamax(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

📈 Training & Validation Performance

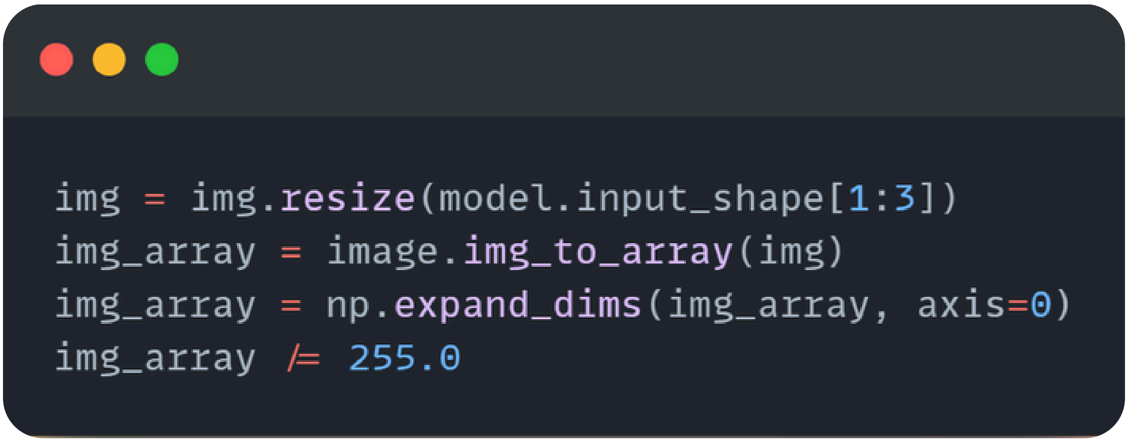
- **Model shows stable training with minimal overfitting.**
- **Loss decreases consistently and accuracy plateaus after ~10 epochs.**



🧾 Input Requirements

- **Input Type: RGB image of a strawberry leaf**
- **Input Size: Must be resized to match the model’s expected shape (e.g. (150, 150) or whatever model.input_shape[1:3] returns)**
- **Preprocessing:**
  - **Resize to model input shape**

  - **Normalize pixel values: divide by 255.0**
  - **Expand dimensions to match shape (1, height, width, 3)**



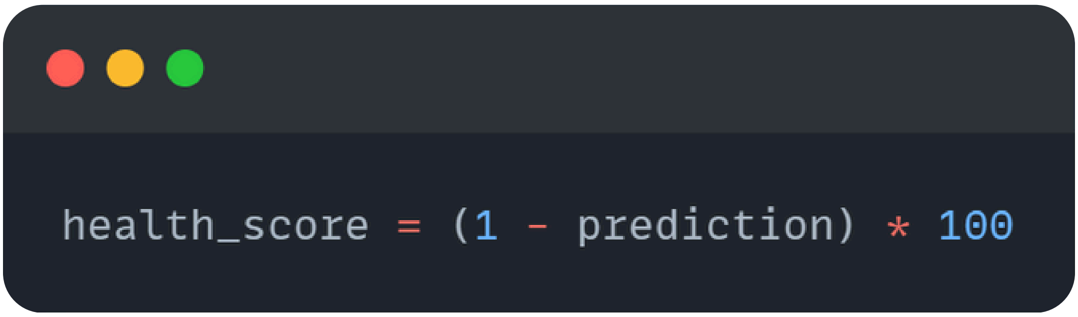
```
img = img.resize(model.input_shape[1:3])
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.0
```

🔮 Output

The model returns a single float between 0 and 1:

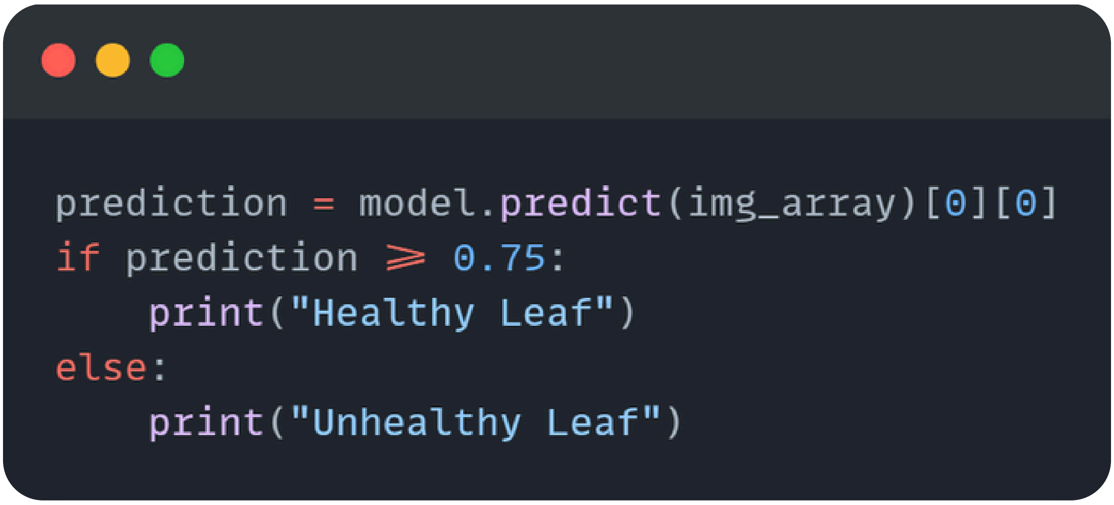
- **0.0 → 0.74: interpreted as Unhealthy**
- **0.75 → 1.0: interpreted as Healthy**

You can convert the output to a health score:



```
health_score = (1 - prediction) * 100
```

🧪 Example Inference



```
prediction = model.predict(img_array)[0][0]
if prediction >= 0.75:
    print("Healthy Leaf")
else:
    print("Unhealthy Leaf")
```

🧪 Extended Diagnosis

If the leaf is unhealthy, the system integrates with Roboflow's Object Detection API to identify the exact disease class. Example response classes include:

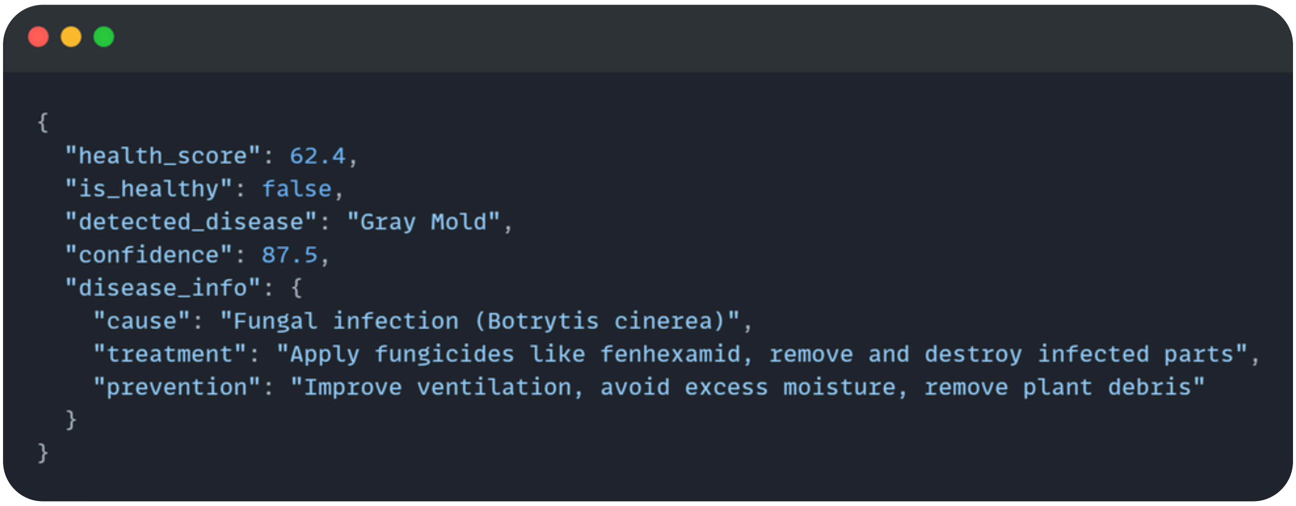
- *Angular Leafspot*
- *Anthracnose Fruit Rot*
- *Gray Mold*
- etc.

Each class is linked to metadata (cause, treatment, prevention) stored in a Python dictionary for real-time feedback.

⚙️ Deployment Notes

- This model is built using TensorFlow/Keras, easily portable to backends using:
    - Flask / FastAPI (via load_model and predict)
    - TensorFlow.js (after conversion using tensorflowjs_converter)
- Avoid Streamlit in production; instead:
    - Load and use the model inside a Python API
    - Accept image files from React frontend
    - Send predictions + disease analysis JSON back to the client

🔐 Example Response (to frontend)



```
{
  "health_score": 62.4,
  "is_healthy": false,
  "detected_disease": "Gray Mold",
  "confidence": 87.5,
  "disease_info": {
    "cause": "Fungal infection (Botrytis cinerea)",
    "treatment": "Apply fungicides like fenhexamid, remove and destroy infected parts",
    "prevention": "Improve ventilation, avoid excess moisture, remove plant debris"
  }
}
```

🧰 Dependencies

- tensorflow>=2.x
- PIL
- numpy
- requests (for Roboflow API)
- inference_sdk (optional)

📌 Notes

- Adjust the health threshold based on model performance if needed.
- Ensure consistent image quality in production (e.g., daylight, clear background).
- Roboflow detection is optional but adds high value for disease-specific interventions.

🔗 References & Resources

- 📂 Kaggle Notebook: Strawberry Health CNN by abdomogahed
- 📦 Roboflow Docs: https://docs.roboflow.com
- 🔬 TensorFlow Documentation: https://www.tensorflow.org/api_docs
- 🖼️ Pillow Docs: https://pillow.readthedocs.io
- 💡 NumPy Docs: https://numpy.org/doc
- 🌐 Requests Library: https://docs.python-requests.org
- 🏁 Keras API: https://keras.io/api