Revolutionizing Disease Detection

Mostafa Khalid Sallam

Topics

- INTRODUCTION
- DEEP LEARNING MODELS FOR IMAGE CLASSIFICATION
- INCEPTIONRESNET V2 ARCHITECTURE
- A QUICK COMPARISON OF THE MODELS THAT WERE USED
- PRACTICAL APPLICATION
- CONCLUSION

INTRODUCTION

Introduction

Challenge

Threat:

Plant diseases → Crop loss, economic damage, food

Insecurity:

Traditional Methods:

Manual inspection → Slow, ineffective for early diagnosis.

Introduction

Solution

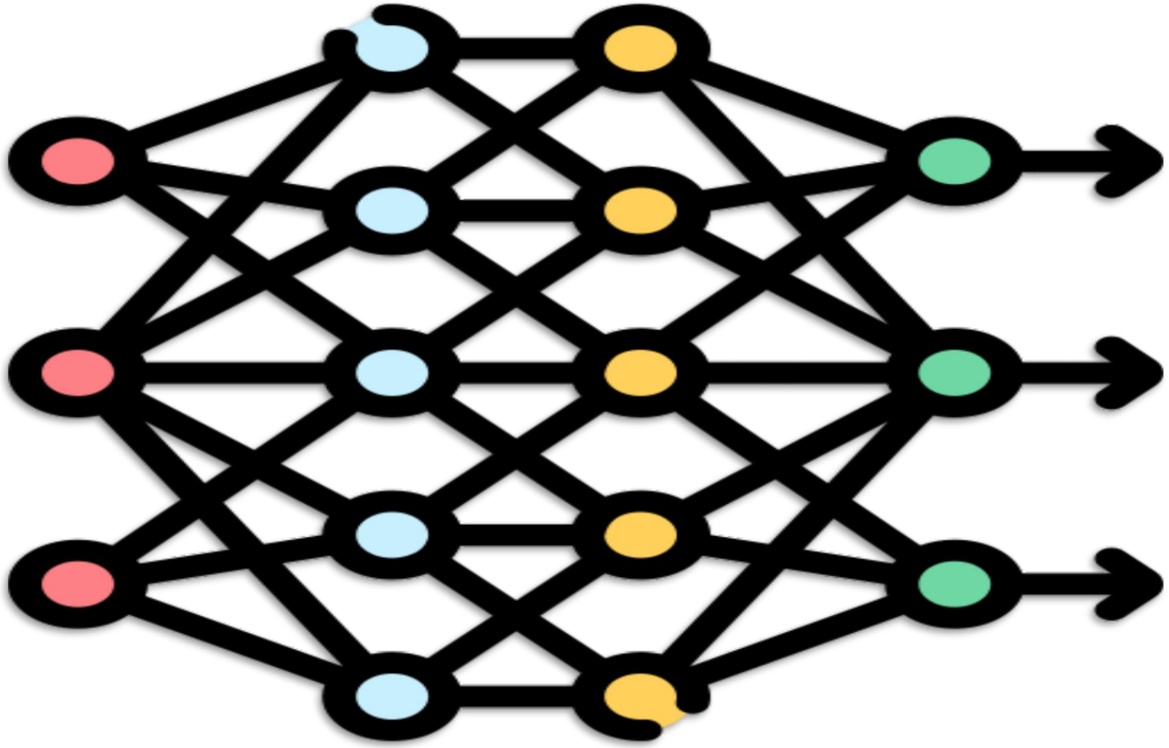
Deep Learning (DL):

Automates detection via image analysis → Speed, accuracy, scalability.

InceptionResNetV2:

Hybrid architecture (Inception + Residual Networks).

Detects subtle symptoms → Early-stage diagnosis.

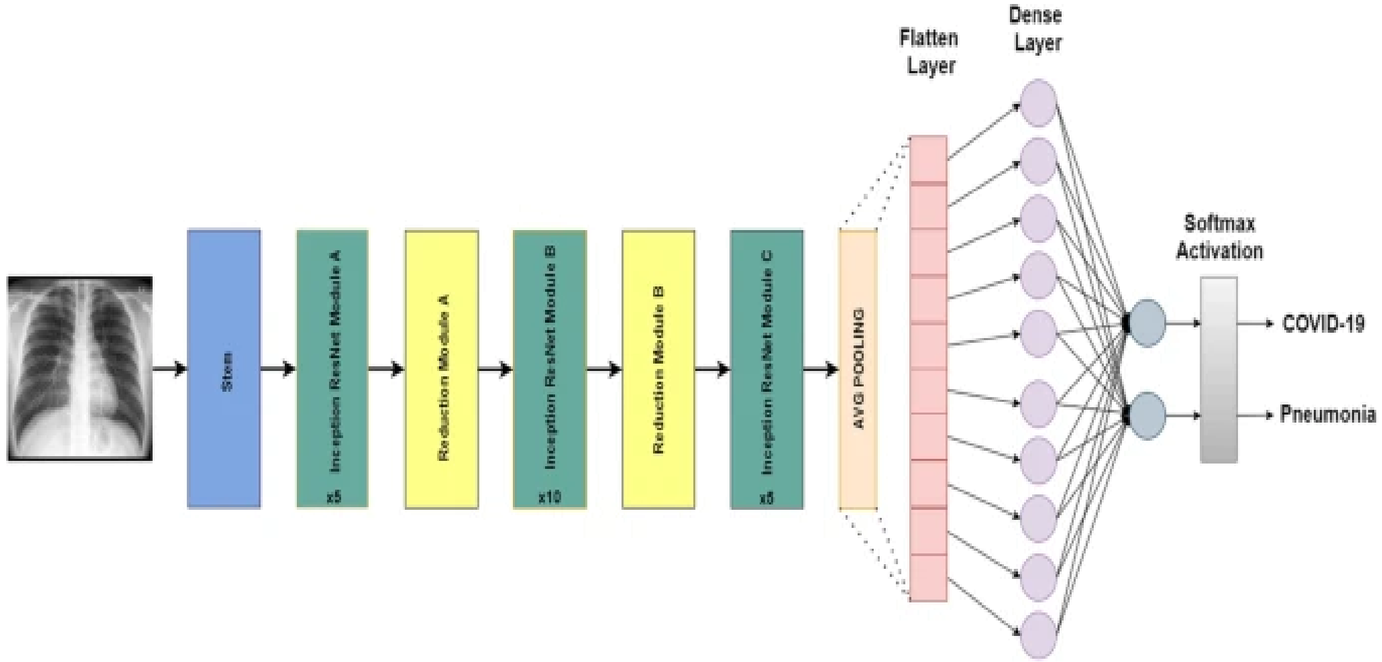


Introduction

Objective

Highlight AI's role in bridging agricultural gaps.

Showcase InceptionResNetV2 as a game-changer for sustainable farming.



DEEP LEARNING MODELS FOR IMAGE CLASSIFICATION

Deep learning models for image classification

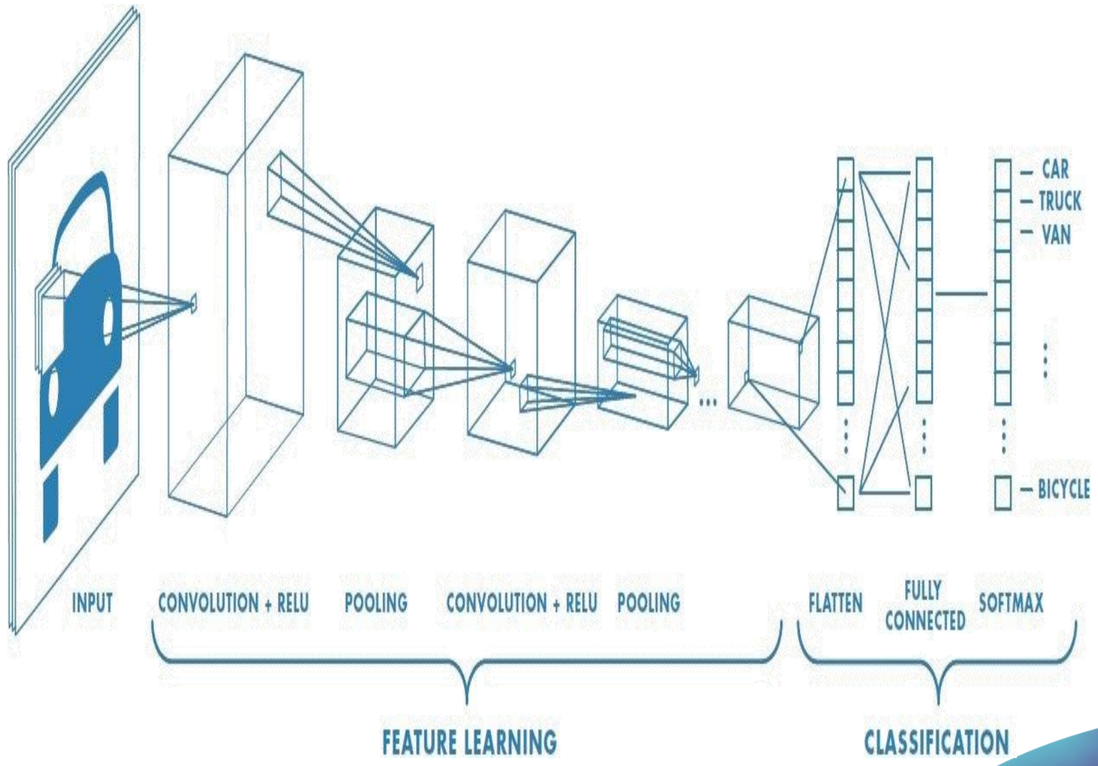
Convolutional Neural Networks (CNNs)

Description

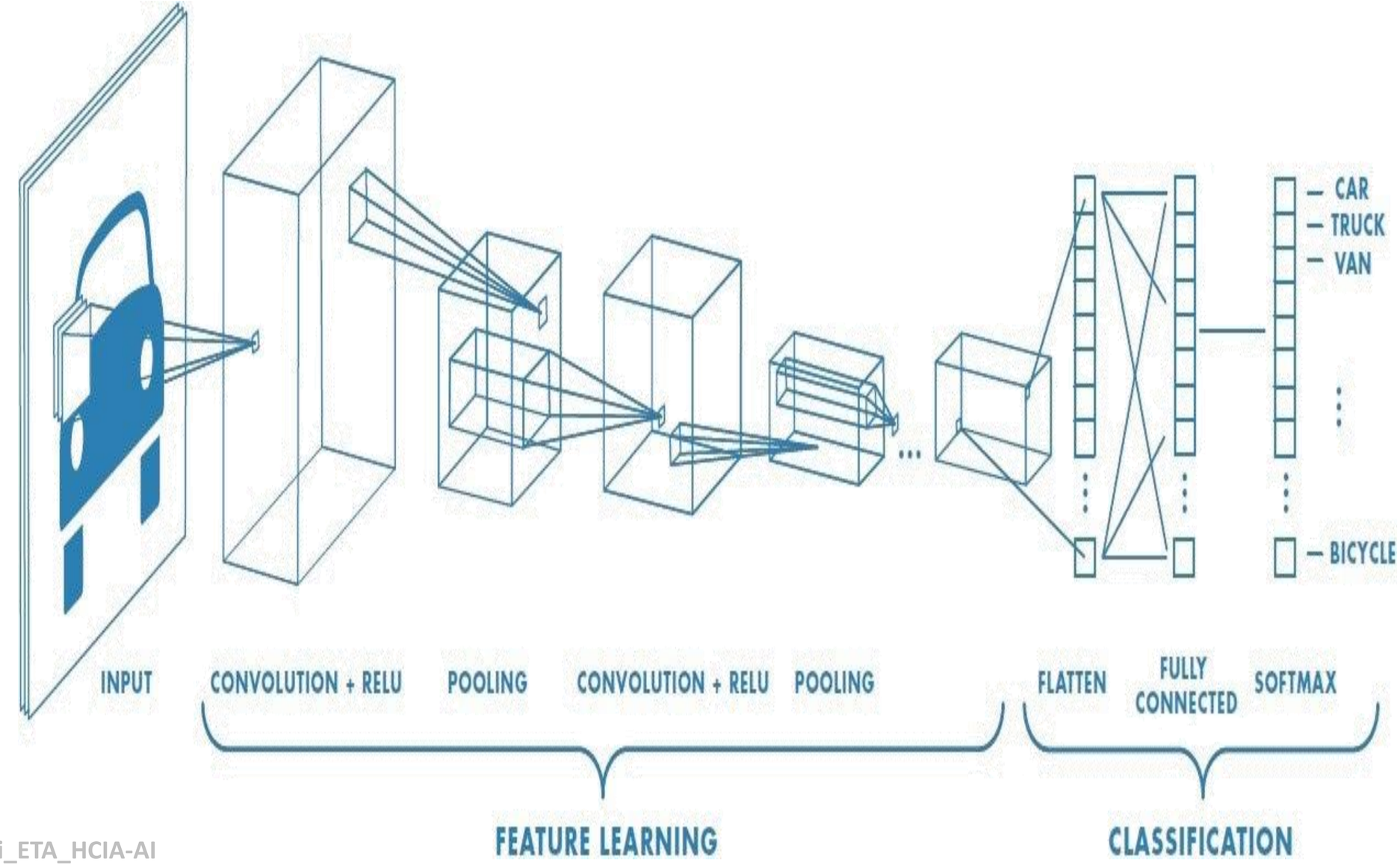
Convolutional Neural Networks (CNNs) are a specialized type of deep learning model designed to analyze visual data (as Images).

A typical CNN consists of:

- Convolutional layers
- Pooling layers
- Fully connected layers



Deep learning models for image classification



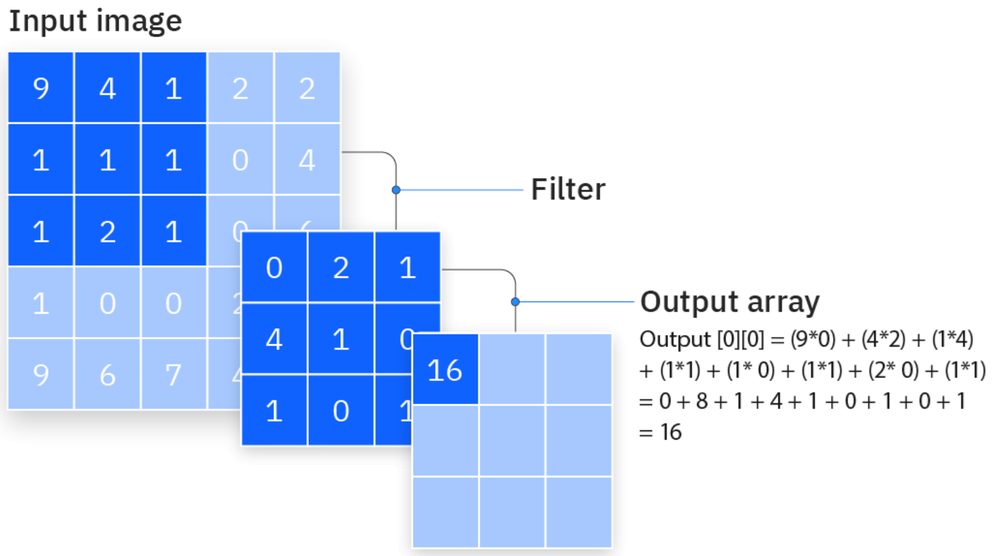
Deep learning models for image classification

1. Convolution Layer

How it works

The convolution layer applies a **set of filters (kernels)** to the input image.

Each filter slides across the image, performing a dot product **between** the filter and local regions of the image to produce a feature map.



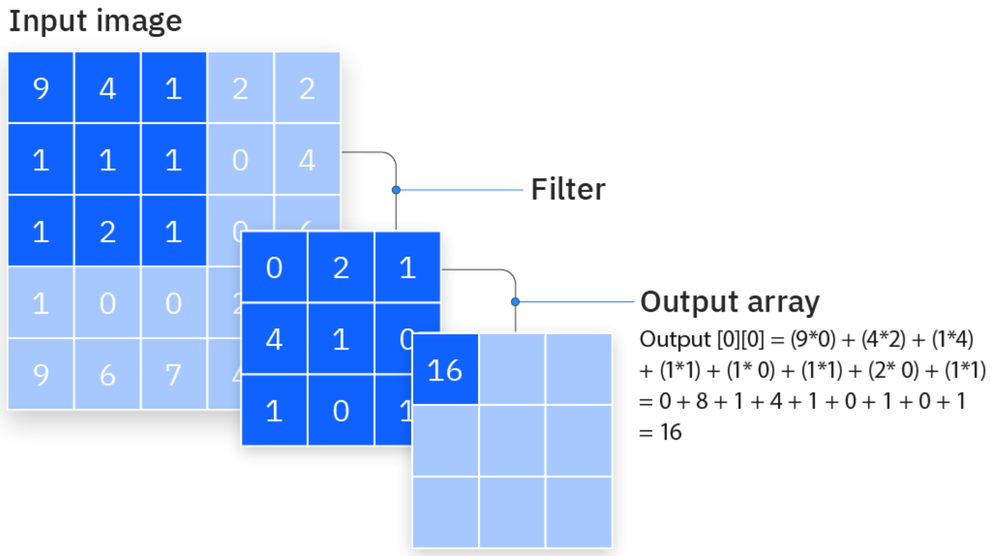
Deep learning models for image classification

1. Convolution Layer

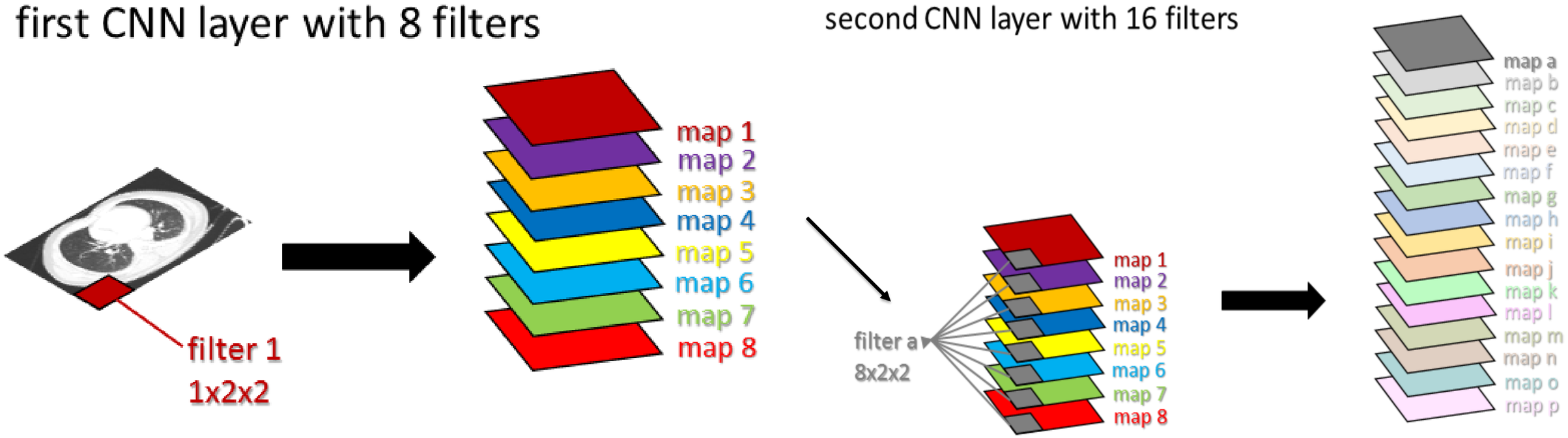
1. Filters & Output Depth

Role: Each filter extracts unique features (edges, textures).

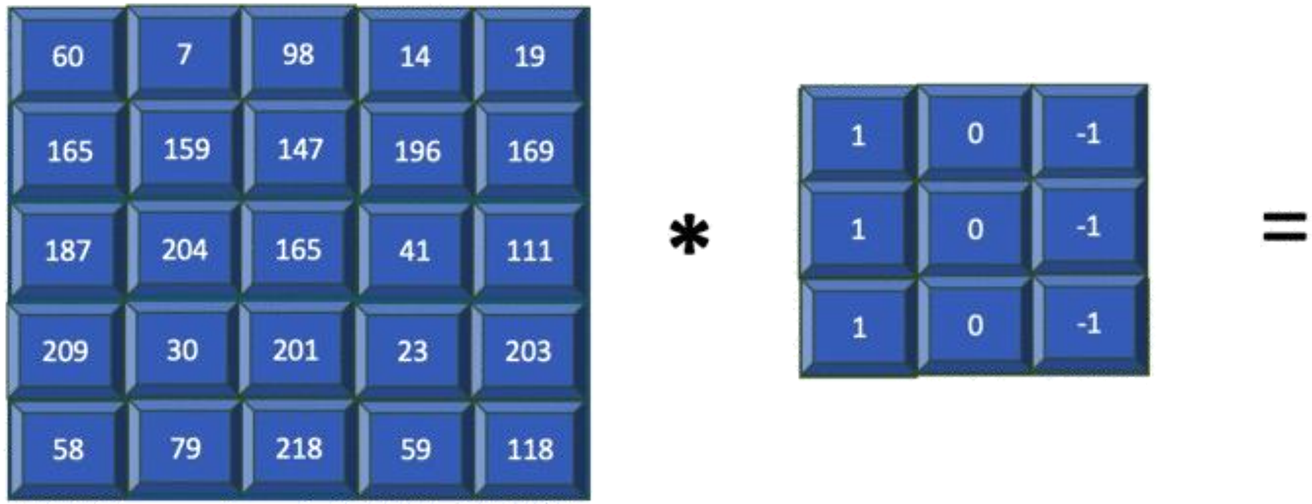
Example: 3 filters → 3 feature maps → Output depth = 3.



Deep learning models for image classification



Deep learning models for image classification



Deep learning models for image classification

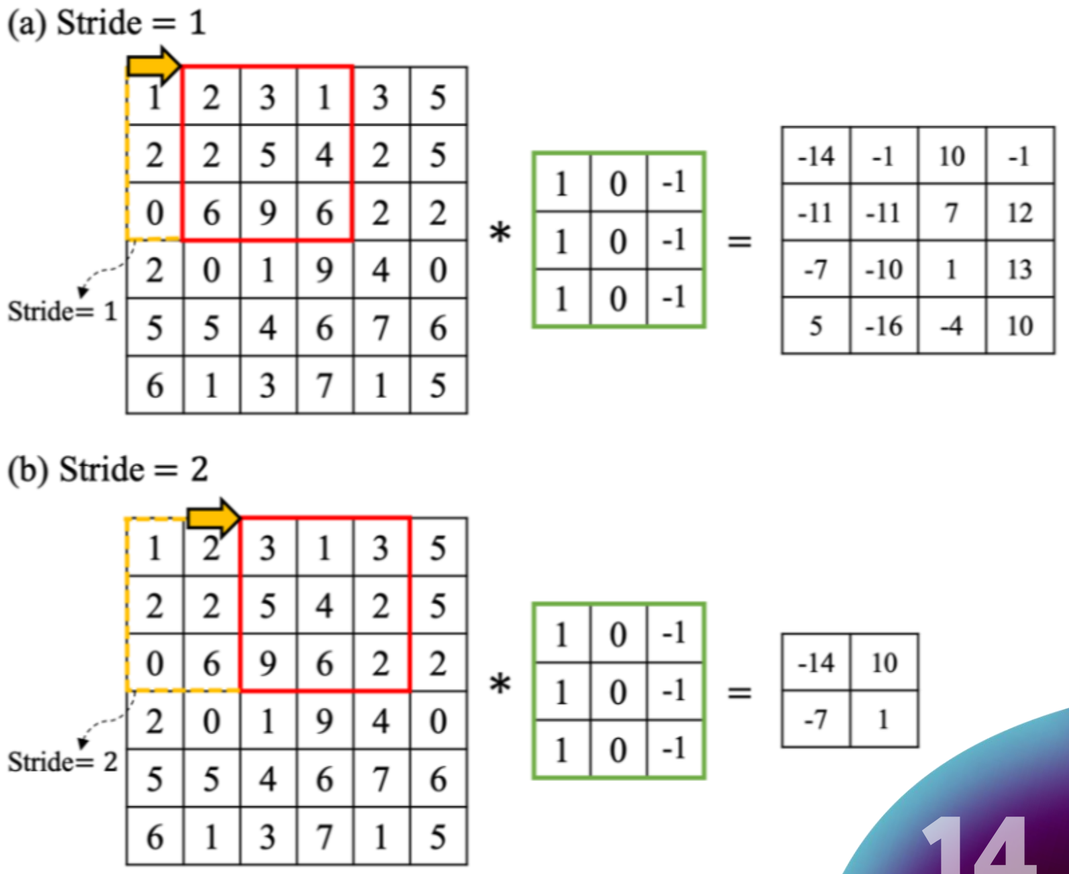
1. Convolution Layer

2. Stride & Output Size

Definition: Steps the kernel takes across the input (e.g., stride=1 or 2).

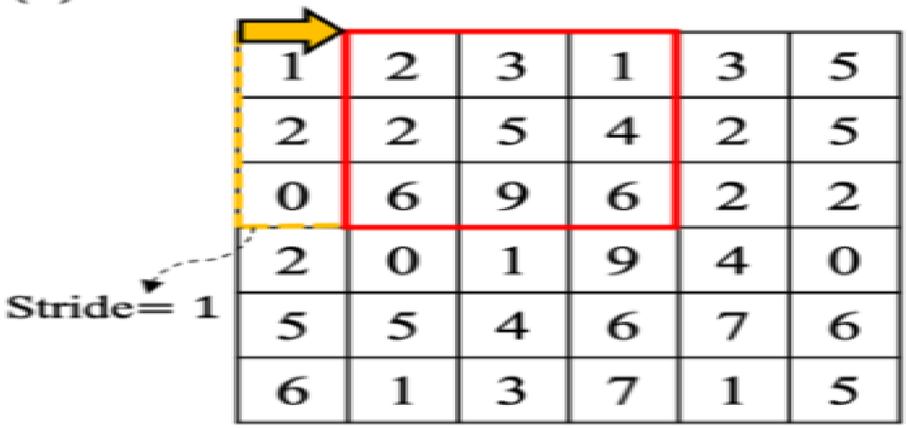
Impact: Larger stride → Smaller output size.

Example: Stride=2 halves output dimensions.



Deep learning models for image classification

(a) Stride = 1



1	2	3	1	3	5
2	2	5	4	2	5
0	6	9	6	2	2
2	0	1	9	4	0
5	5	4	6	7	6
6	1	3	7	1	5

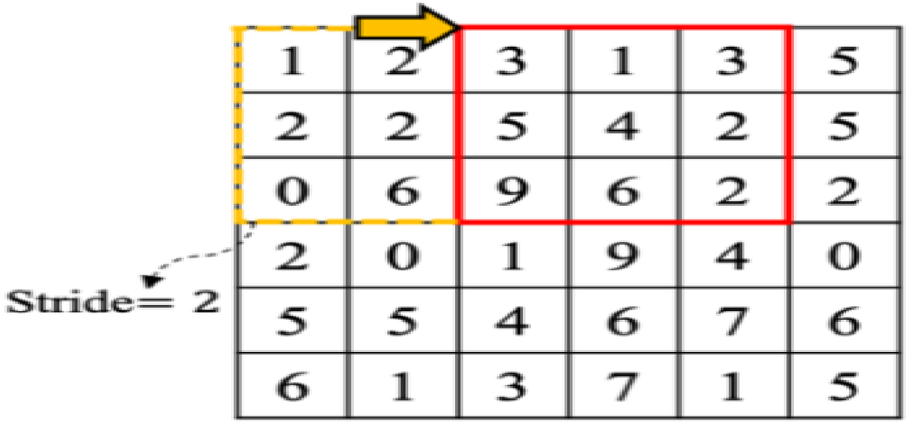
*

1	0	-1
1	0	-1
1	0	-1

=

-14	-1	10	-1
-11	-11	7	12
-7	-10	1	13
5	-16	-4	10

(b) Stride = 2



1	2	3	1	3	5
2	2	5	4	2	5
0	6	9	6	2	2
2	0	1	9	4	0
5	5	4	6	7	6
6	1	3	7	1	5

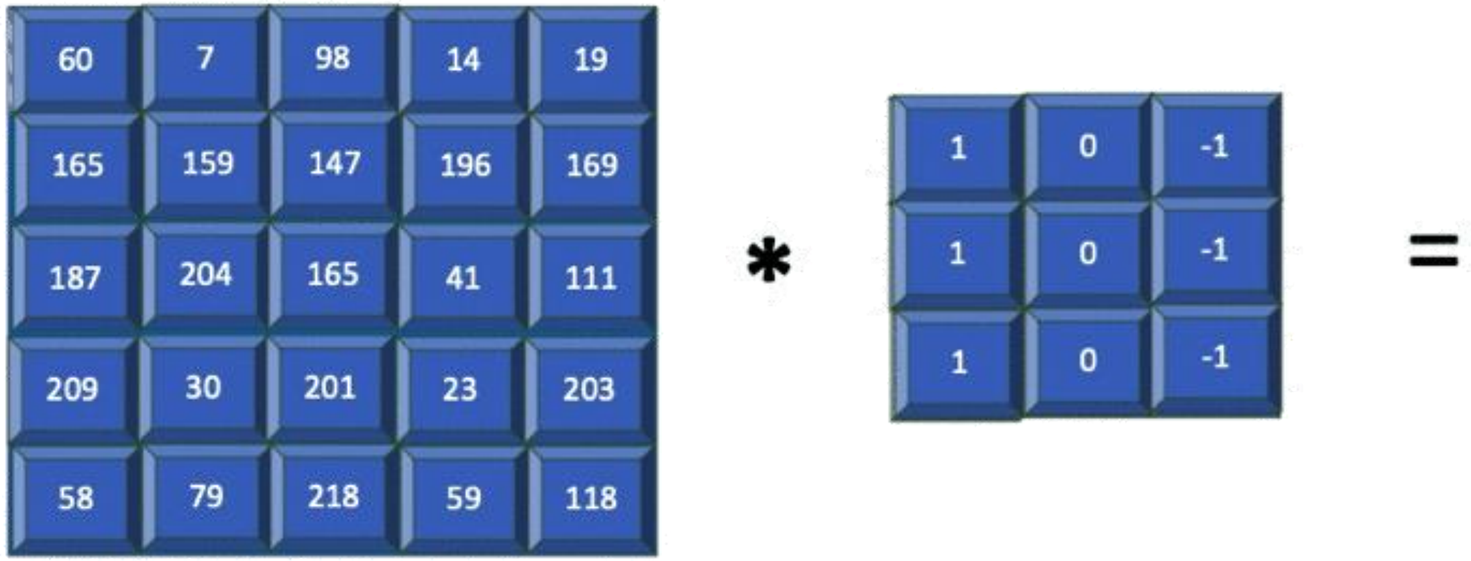
*

1	0	-1
1	0	-1
1	0	-1

=

-14	10
-7	1

Deep learning models for image classification



Deep learning models for image classification

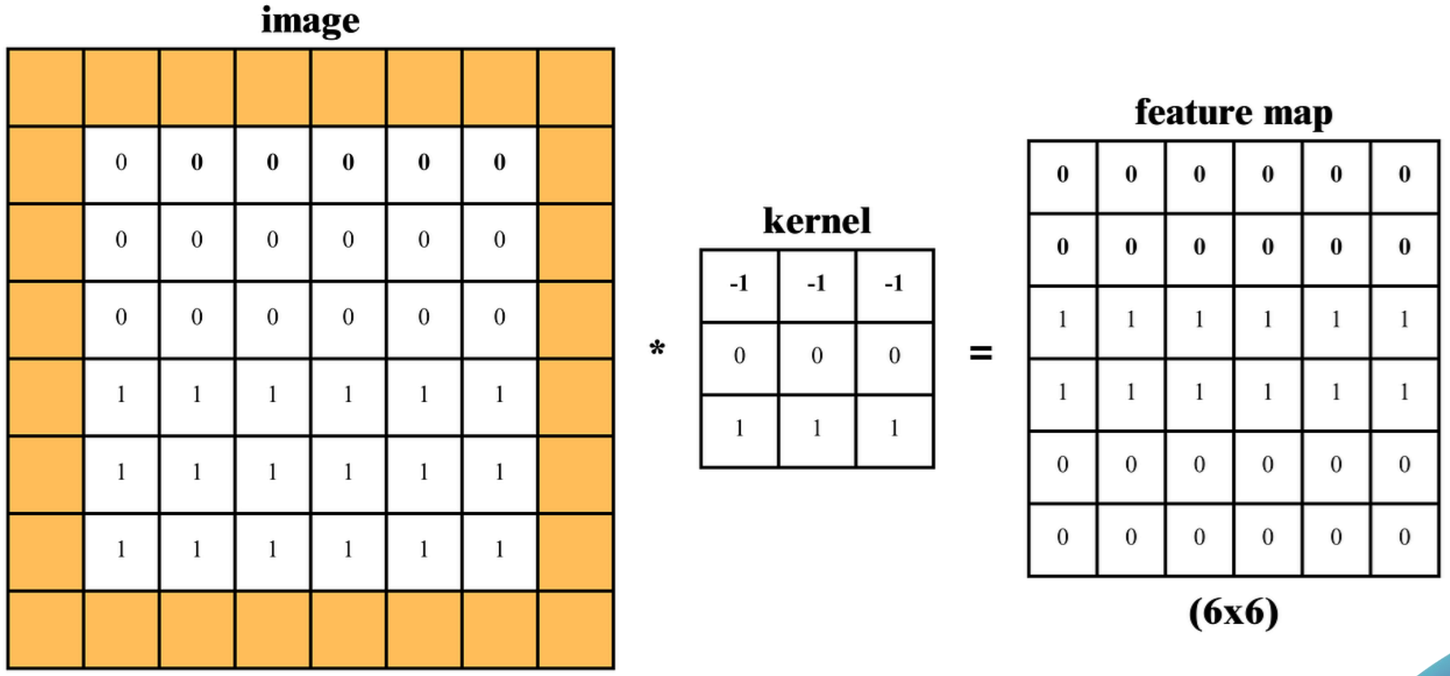
1. Convolution Layer

3.Padding Types:

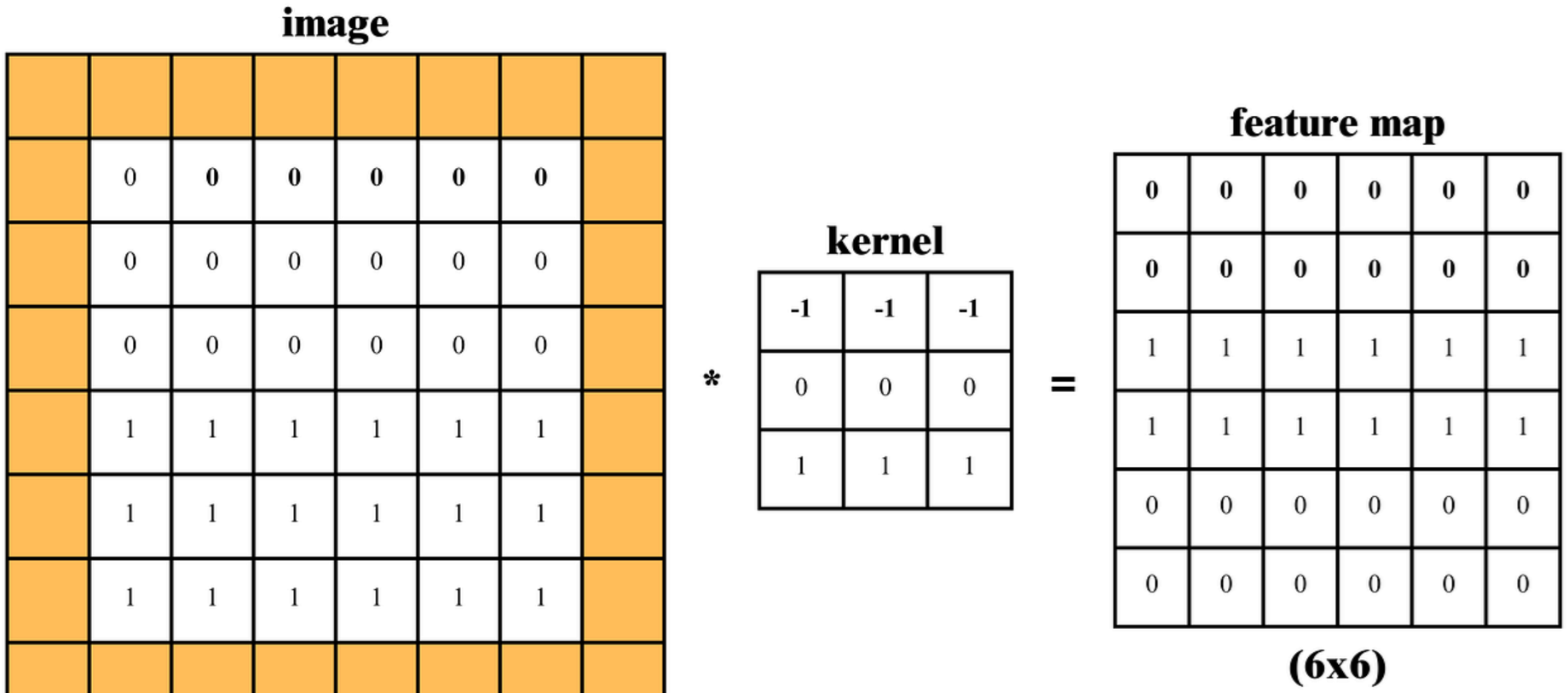
Valid: No padding → Output ≤ input.

Same: Output = input (adds zeros).

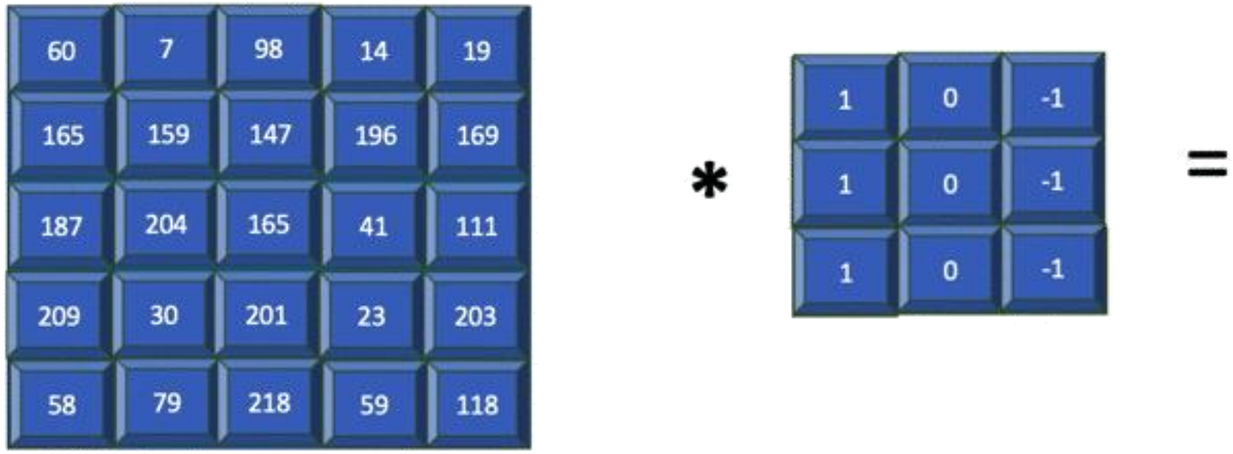
Full: Output > input (maximal zero padding).



Deep learning models for image classification



Deep learning models for image classification



Deep learning models for image classification

1. Convolution Layer

Output size in a Convolutional Neural Network (CNN)

$$\text{Output Size} = \left\lfloor \frac{W - F + 2P}{S} \right\rfloor + 1$$

Variables:

- W: Input width/height (e.g., 32x32 image → W = 32).
- F: Filter/kernel size (e.g., 3x3 filter → F = 3).
- P: Padding (number of zeros added around the input).
- S: Stride (steps the filter moves).
- $[\cdot]$ = Floor function (rounds down to the nearest integer)

Deep learning models for image classification

1. Convolution Layer

Output size in a Convolutional Neural Network (CNN)

Example Calculations

1. Input: 28x28 (**W = 28**). Filter: 3x3 (**F = 3**). Stride: 1 (**S = 1**). Padding: 1 (**P = 1**):

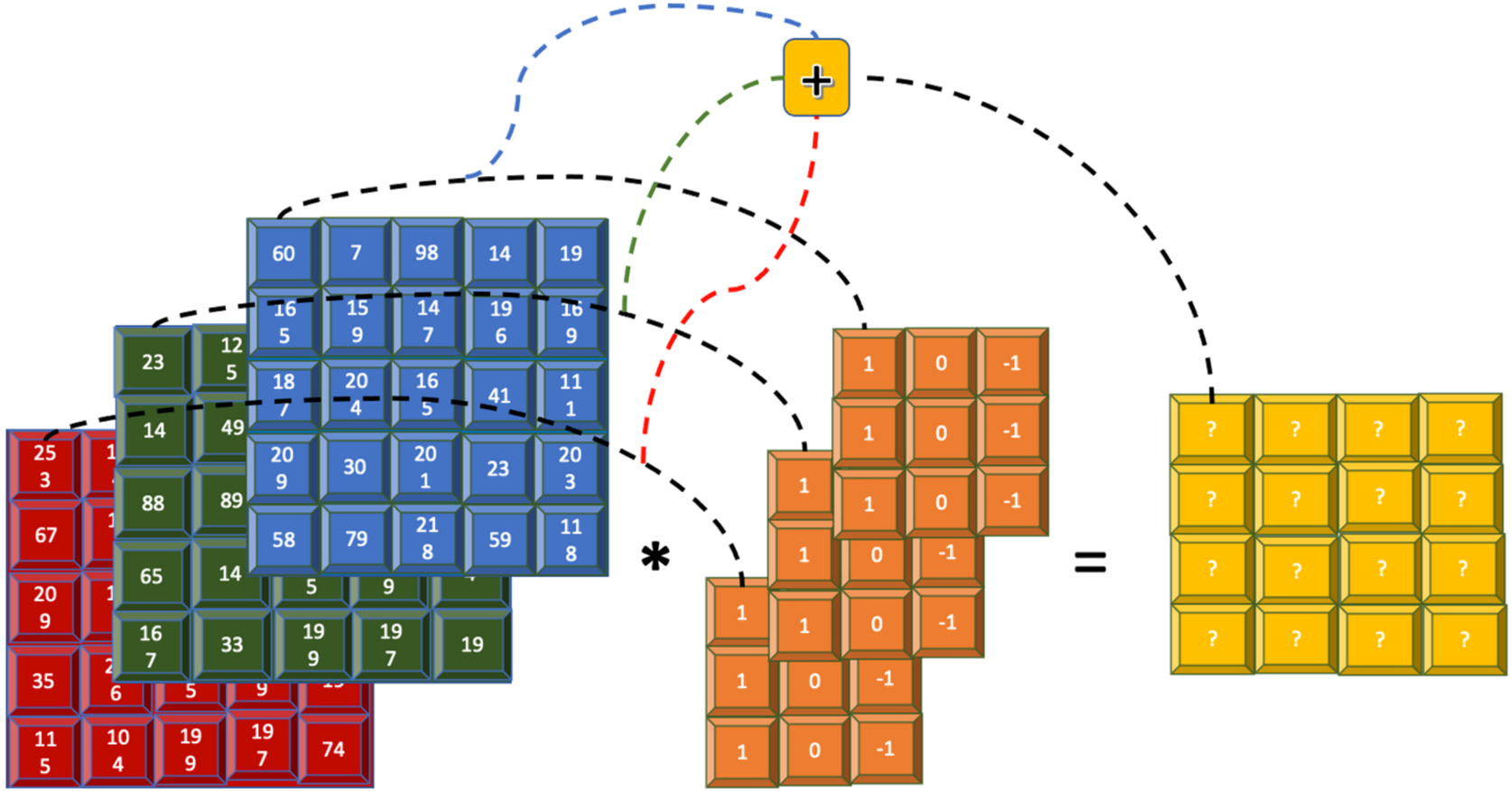
$$\frac{28 - 3 + 2(1)}{1} + 1 = 28 \quad (\textbf{Same padding}!)$$

2. Input: 32x32 (**W = 32**). Filter: 5x5 (**F = 5**). Stride: 2 (**S = 2**). Padding: 0 (**P = 0**):

$$\frac{32 - 5 + 2(0)}{2} + 1 = 14.5$$
$$\rightarrow Floor\ to\ 14 \quad (\textbf{Valid padding}!)$$

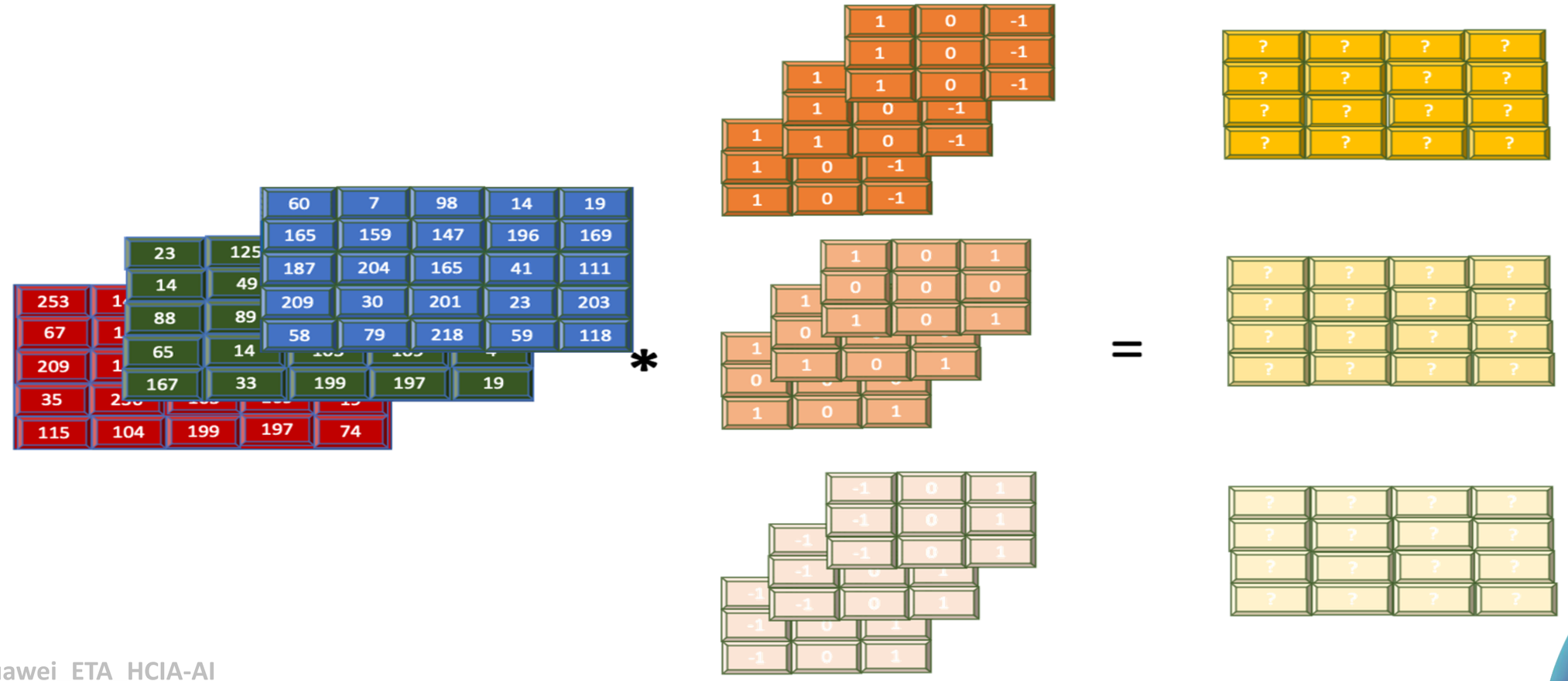
Deep learning models for image classification

- *1. Convolutions applied over the RGB channels*



Deep learning models for image classification

- *1. Convolutions applied over the RGB channels*



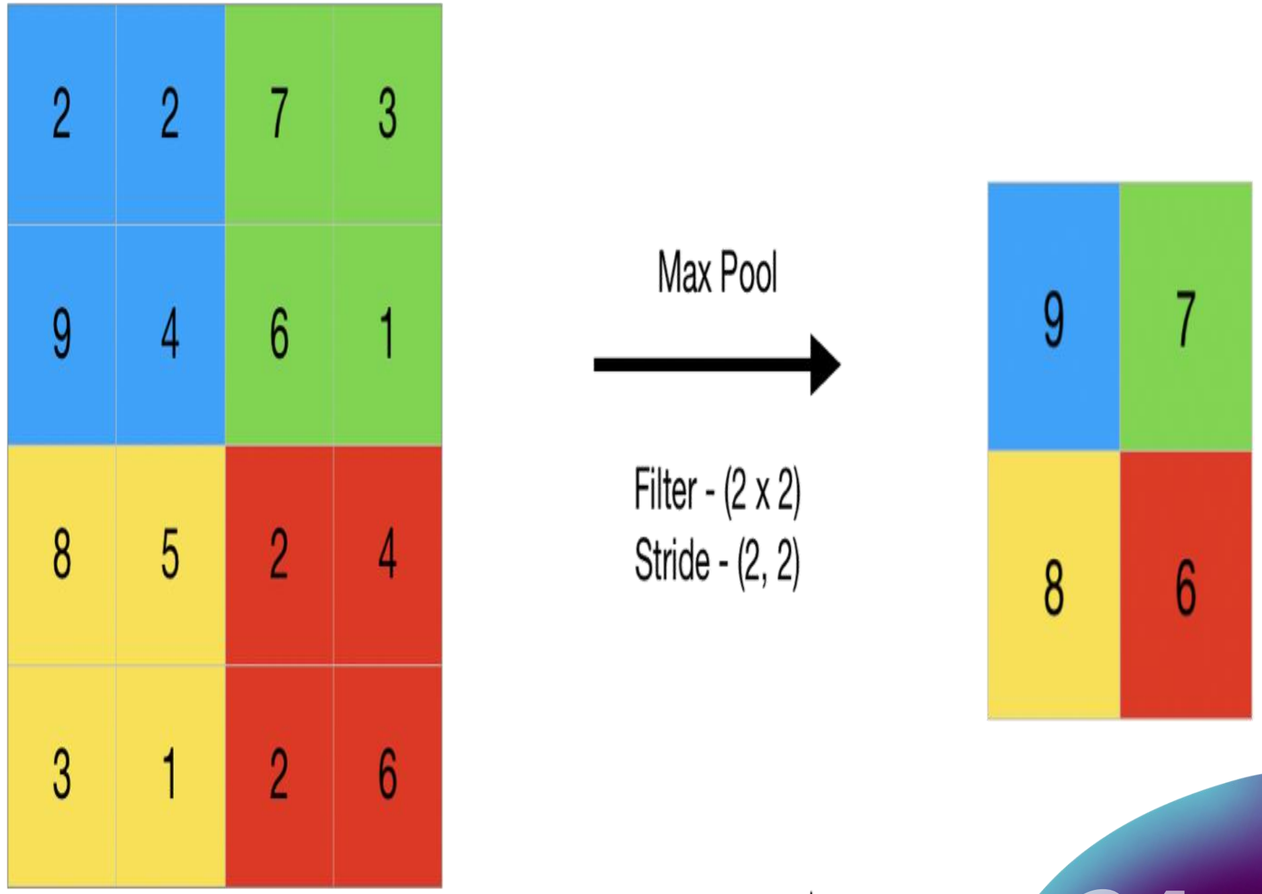
Deep learning models for image classification

2. Max Pooling Layer

How it works

Mechanism:

Slides a window (e.g., 2×2) over feature maps → selects maximum value in each region.



Deep learning models for image classification

2. Max Pooling Layer

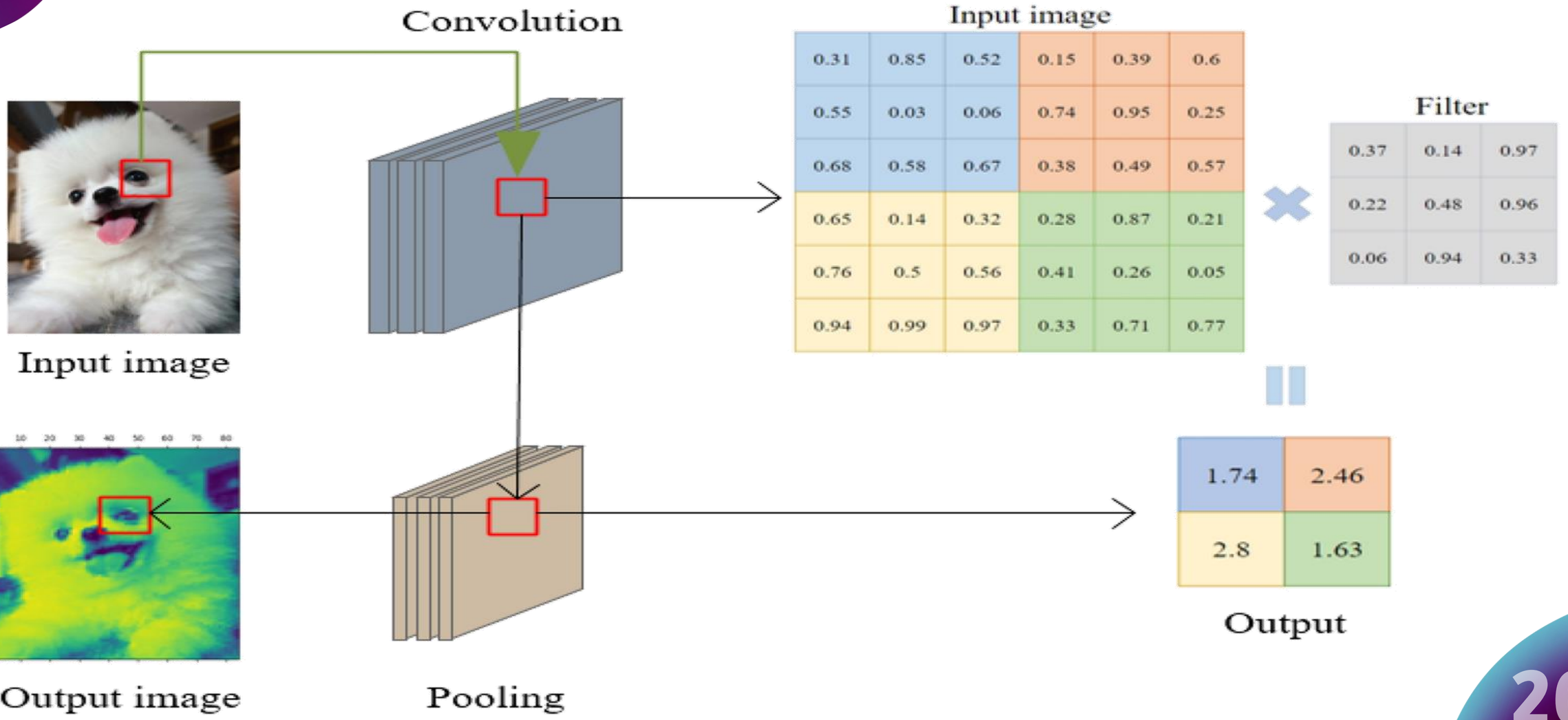
Purpose

Down sampling: Reduces spatial dimensions (e.g., 4x4 → 2x2).

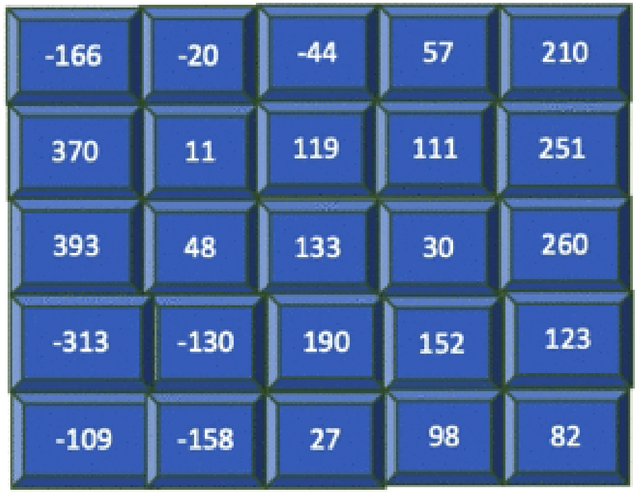
Efficiency: Lowers computational load.

Translation Invariance: Focuses on strongest features (e.g., edges, patterns), regardless of exact location.

Deep learning models for image classification



Deep learning models for image classification



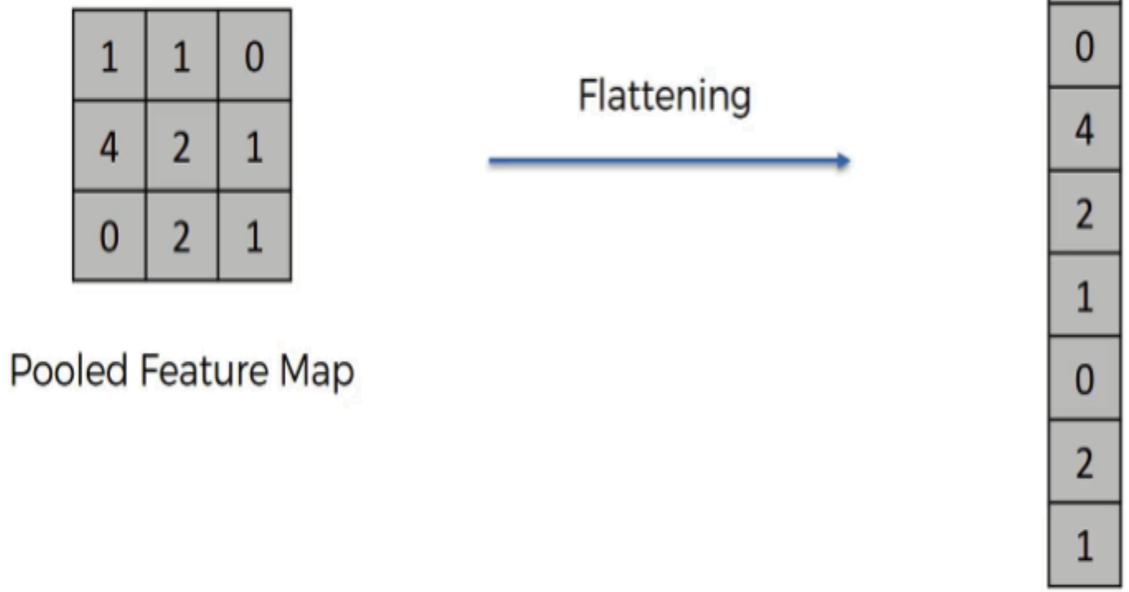
=

Deep learning models for image classification

3. Flatten Layer

How it works

The flatten layer converts the 2D feature maps (after convolution and pooling) into a 1D vector.

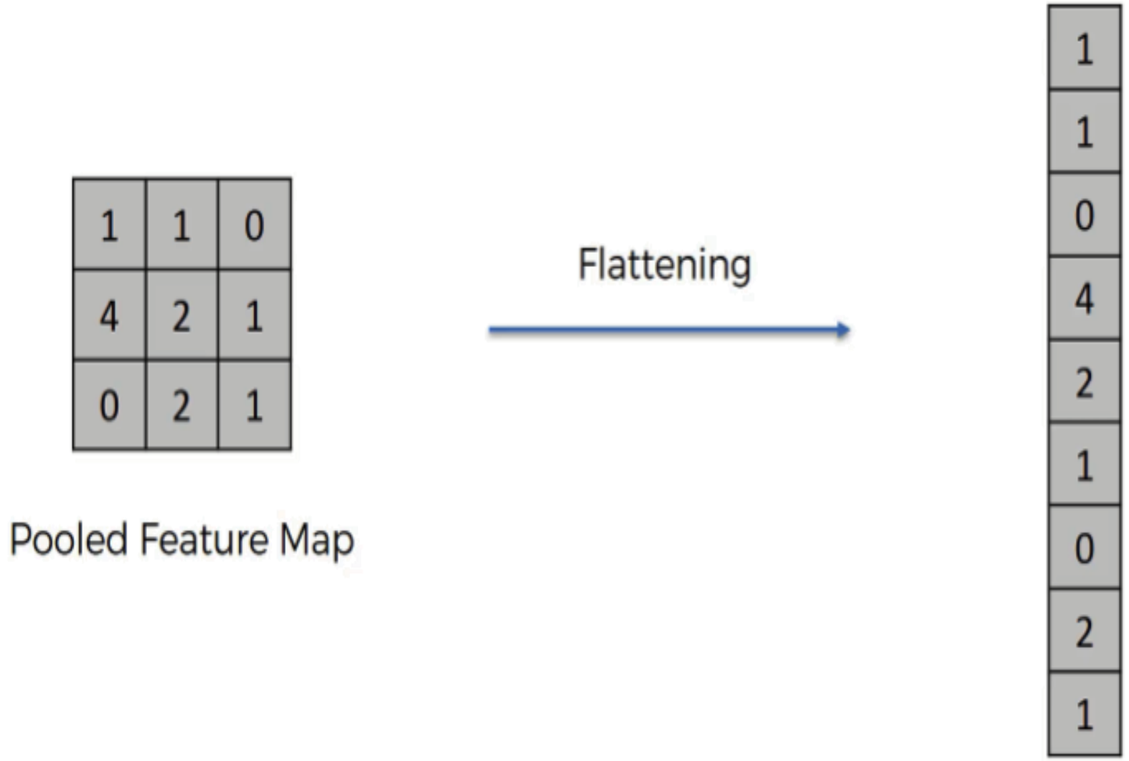


Deep learning models for image classification

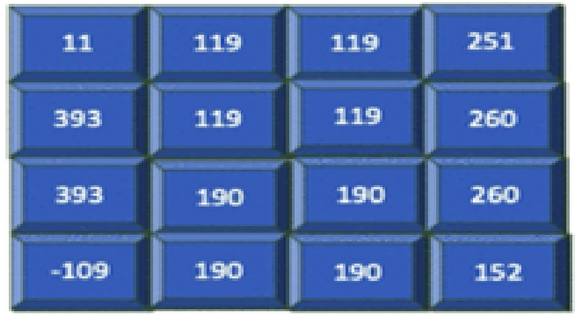
3. Flatten Layer

Purpose

Prepares the data for the fully connected layers (which act like a traditional neural network classifier).

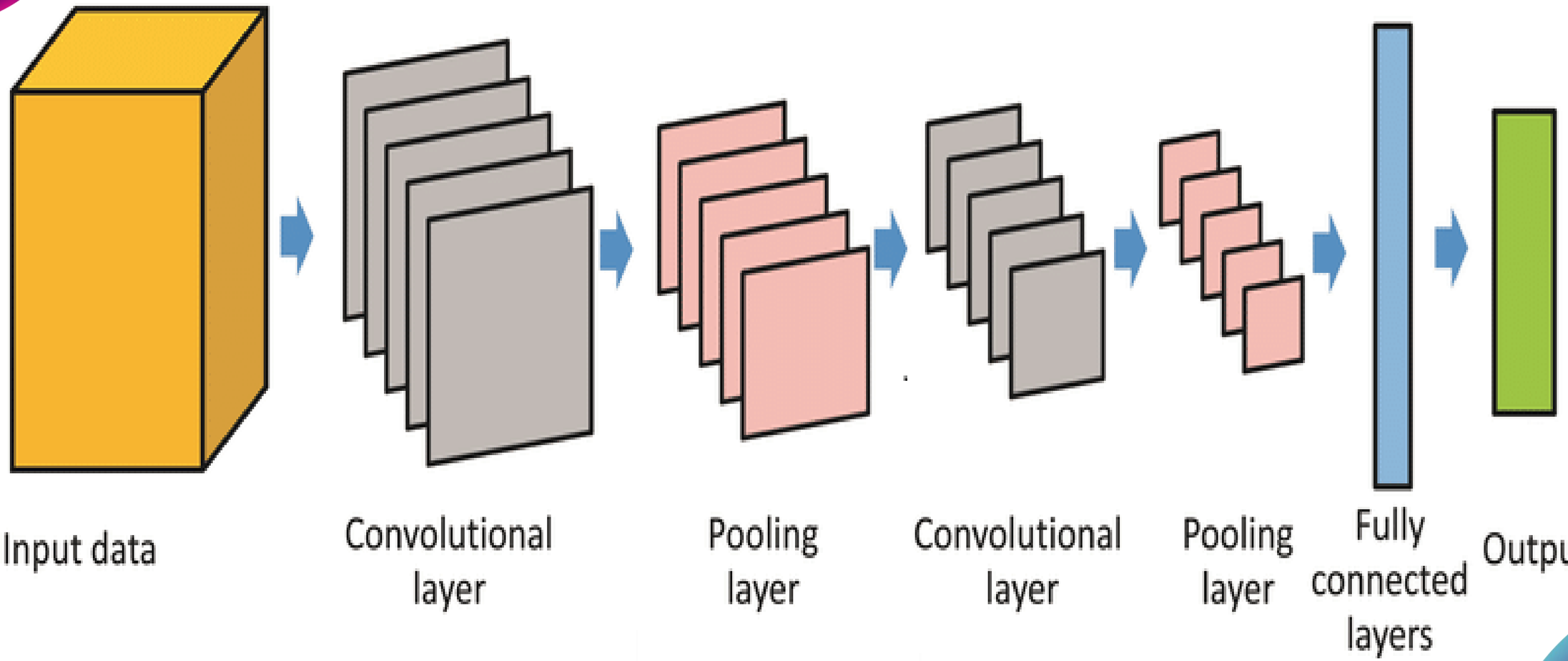


Deep learning models for image classification

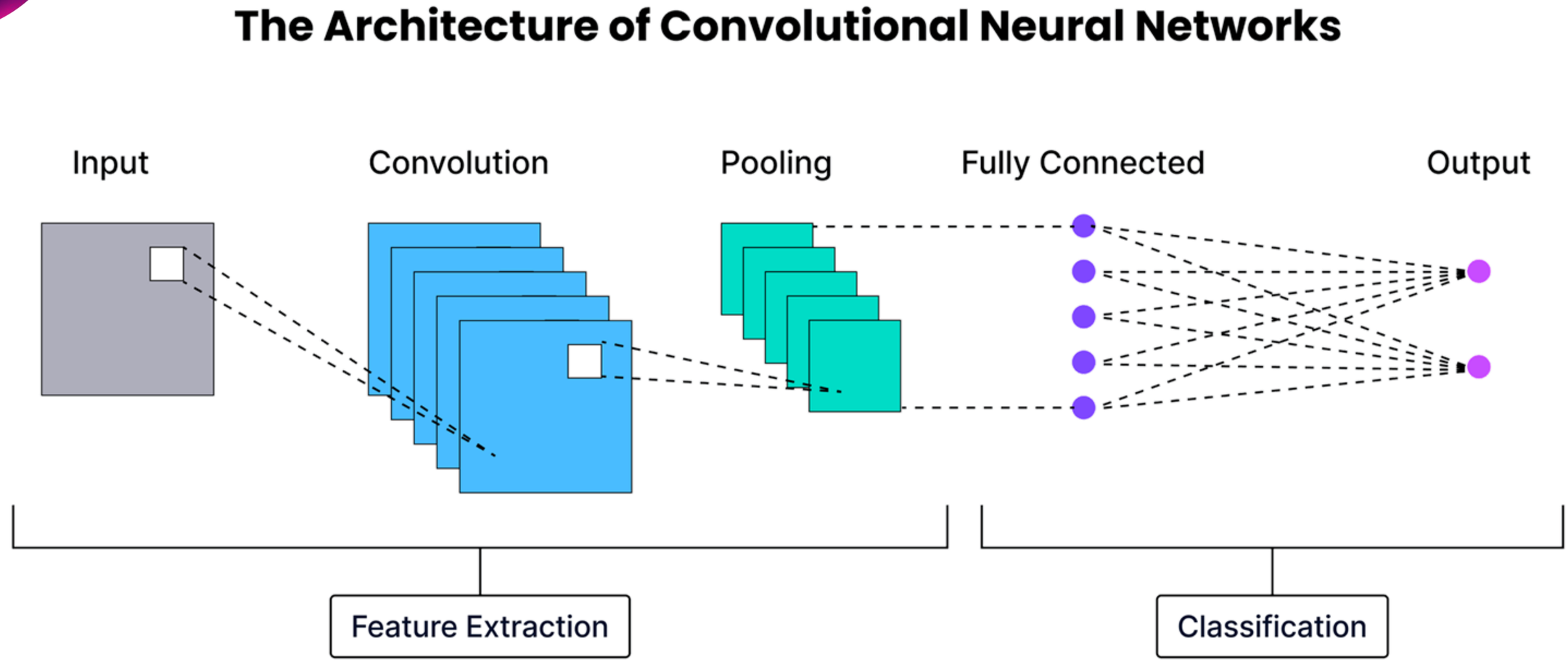


=

Deep learning models for image classification



Deep learning models for image classification



INCEPTIONRESNET V2 ARCHITECTURE

Deep learning models for image classification

InceptionResNetV2 Architecture

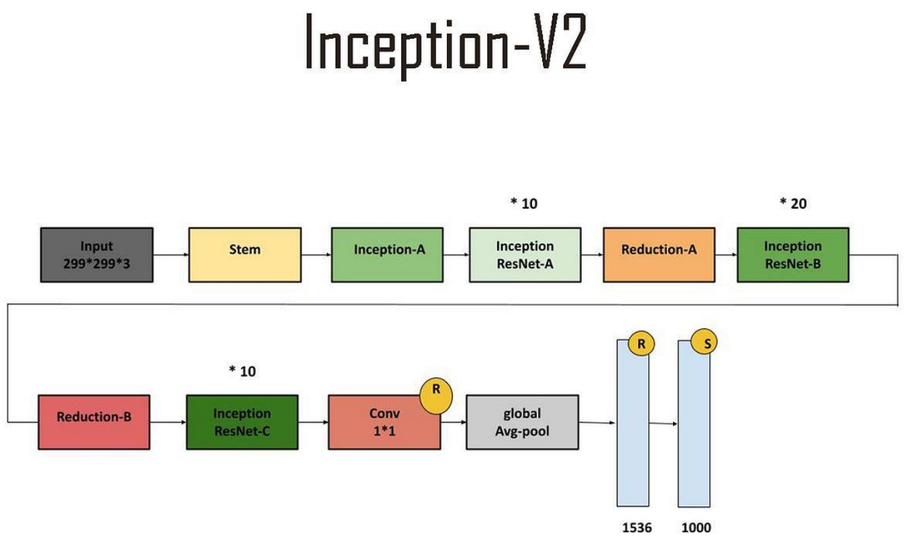
Description

InceptionResNetV2 is a deep convolutional neural network that combines:

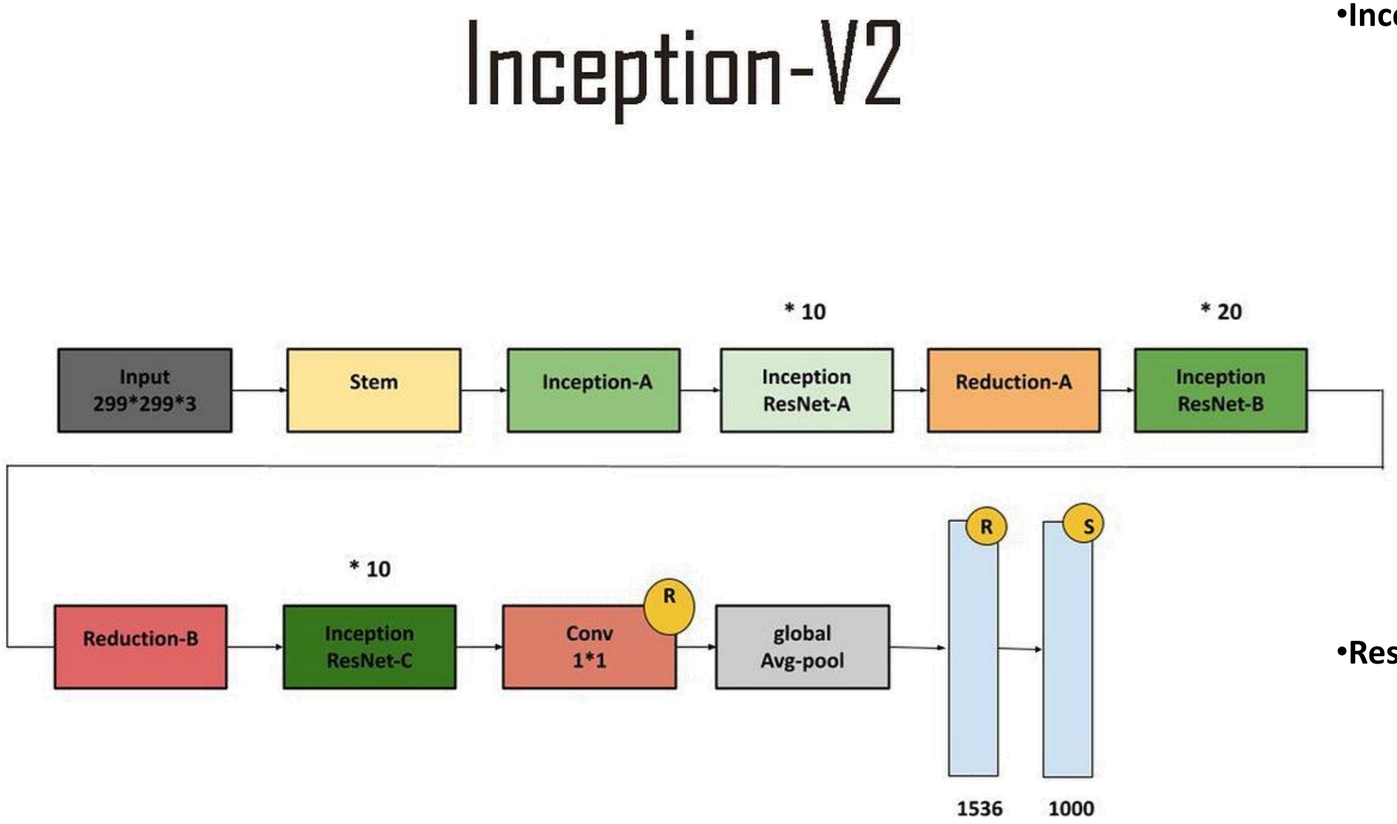
- **Inception modules**
- **Residual connections**

Number of Layers: Approximately **572 layers**, making it one of the **deepest** CNNs available.

Input Size: Typically **299 × 299 pixels**.



Deep learning models for image classification



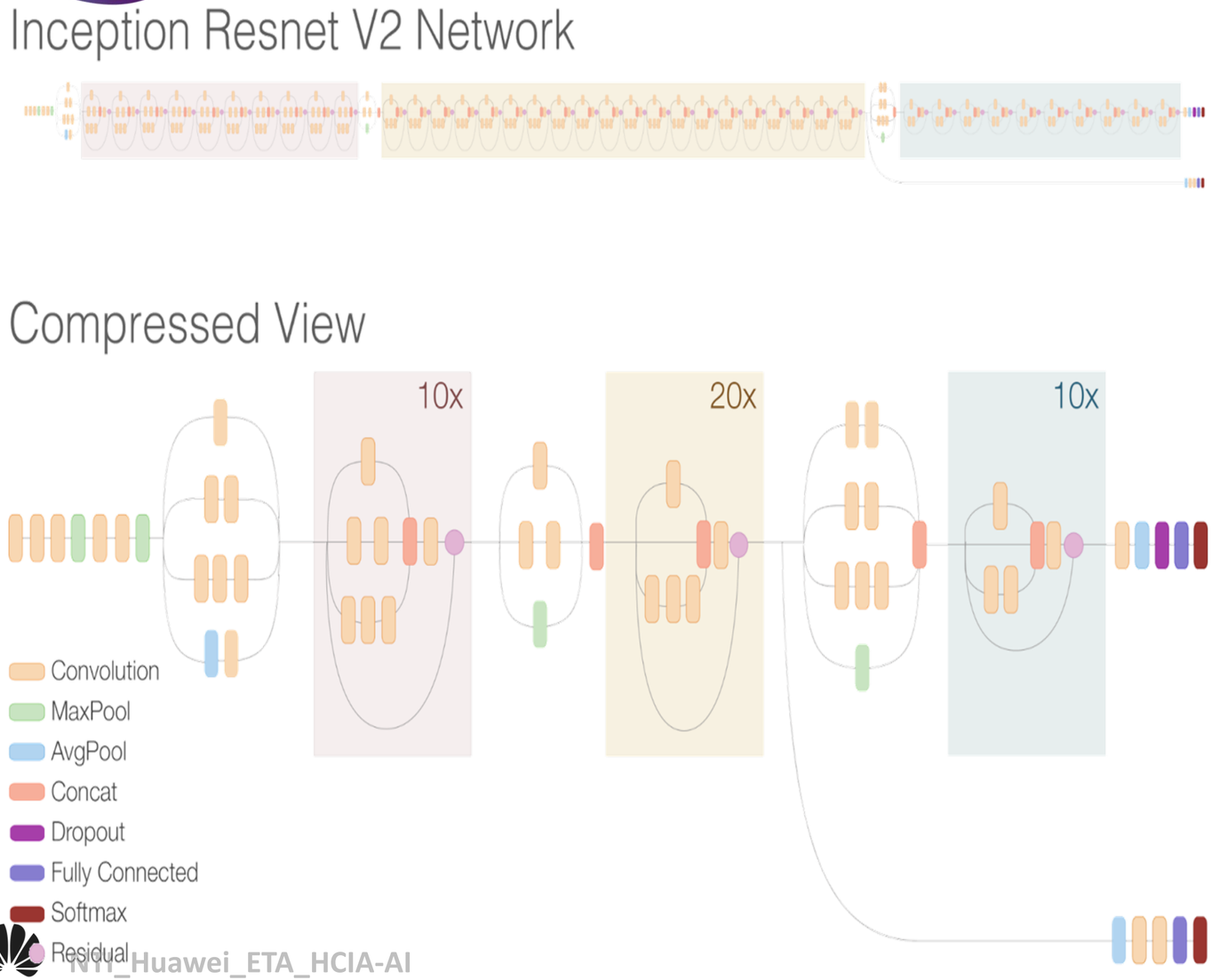
- **Inception Modules**:
  - Use **parallel convolutional branches** with different filter sizes (e.g., 1×1, 3×3, 5×5) to capture features at multiple scales.
  - Example: A single Inception block might combine:
    - **1×1 convolutions** (for **dimensionality reduction**),
    - **3×3 convolutions** (**extract local information**),
    - **5×5 convolutions** (**extract broader information**),
    - **3×3 max pooling** (to retain spatial information).
  - **Goal**: **Efficiently learn both local** and **global** patterns.

$$\text{InceptionBlock}(x) \cdot \alpha + x = \text{Output}$$

- **Residual Connections (Shortcuts)**:
  - Borrowed **from ResNet** architectures.
  - Add the input of a block to its output:

$$\text{Output} = F(x) + x$$

Deep learning models for image classification



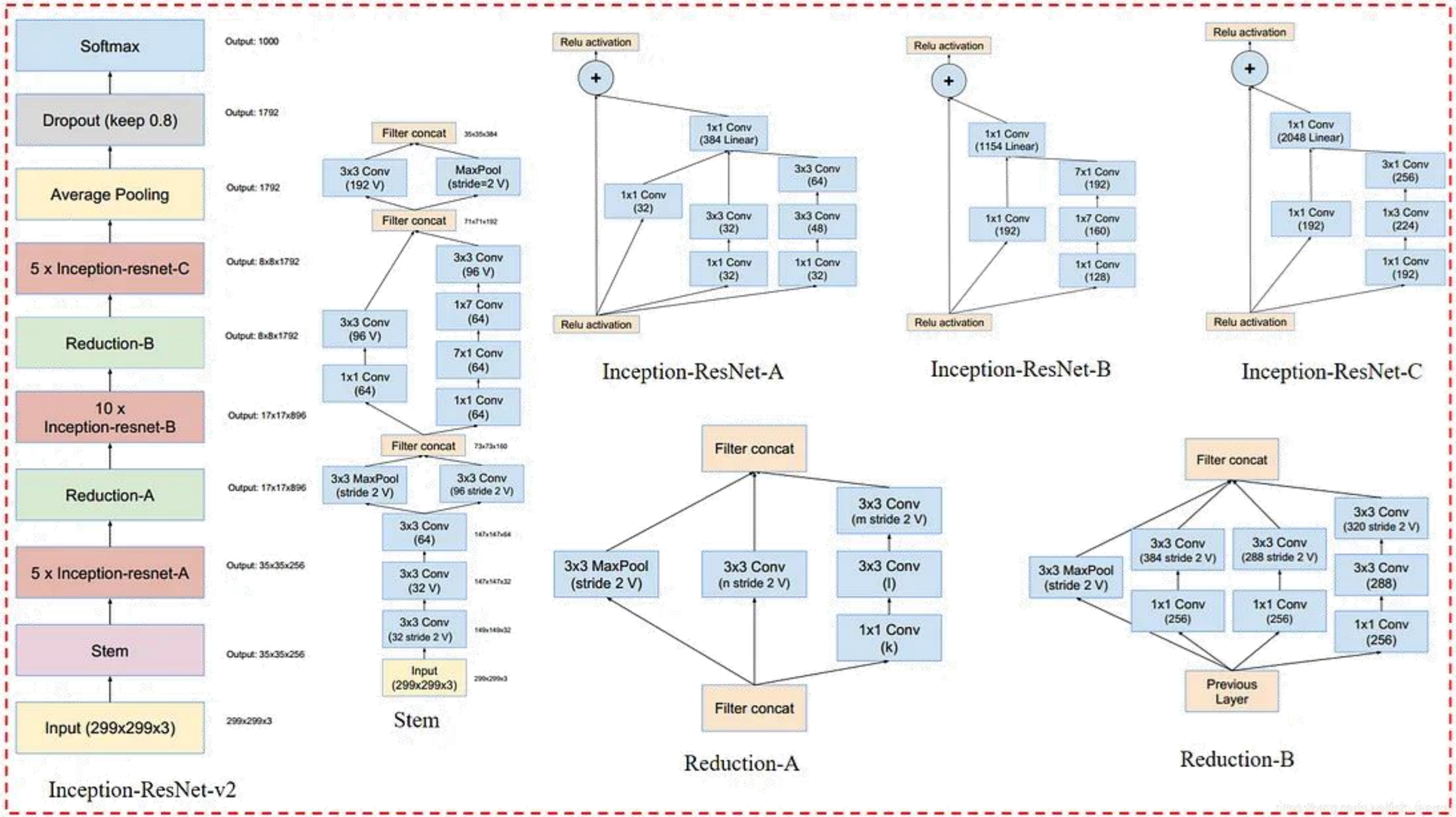
Layer Count:

•**Total Layers**: ~**572 layers** (including convolutional, pooling, batch norm, and fully connected layers).

•**Breakdown**:

- 164 convolutional layers (with varying filter sizes),
- 12 Inception-ResNet modules (complex multi-branch blocks),
- 2 stem blocks (initial feature extractors),
- Auxiliary classifiers (for training stabilization).

Deep learning models for image classification



Deep learning models for image classification

InceptionResNetV2 Architecture

Strengths

Learns both **fine-grained** **and** **high-level features**.

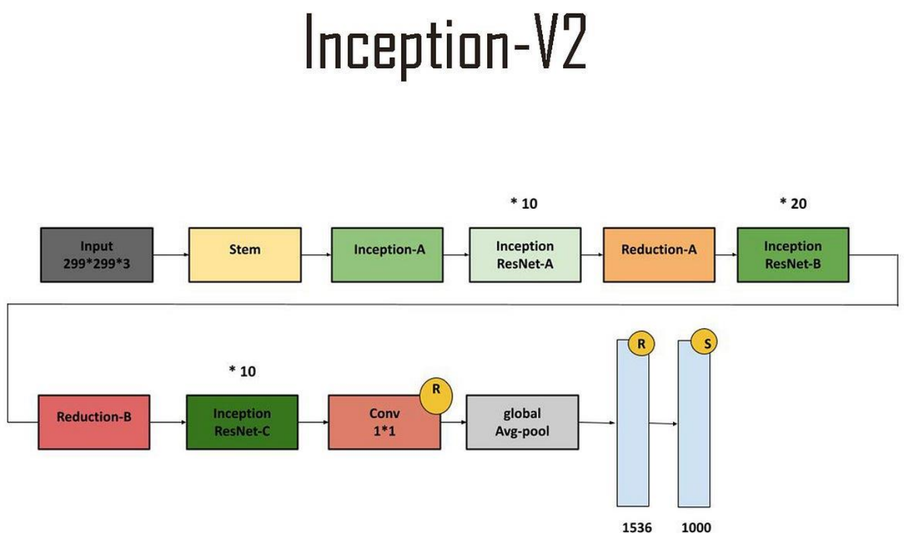
Faster convergence during training **due to residual** paths.

High accuracy in complex classification tasks.

Limitations

High computational cost and **memory usage**.

Requires large datasets and **powerful GPUs** for effective training.

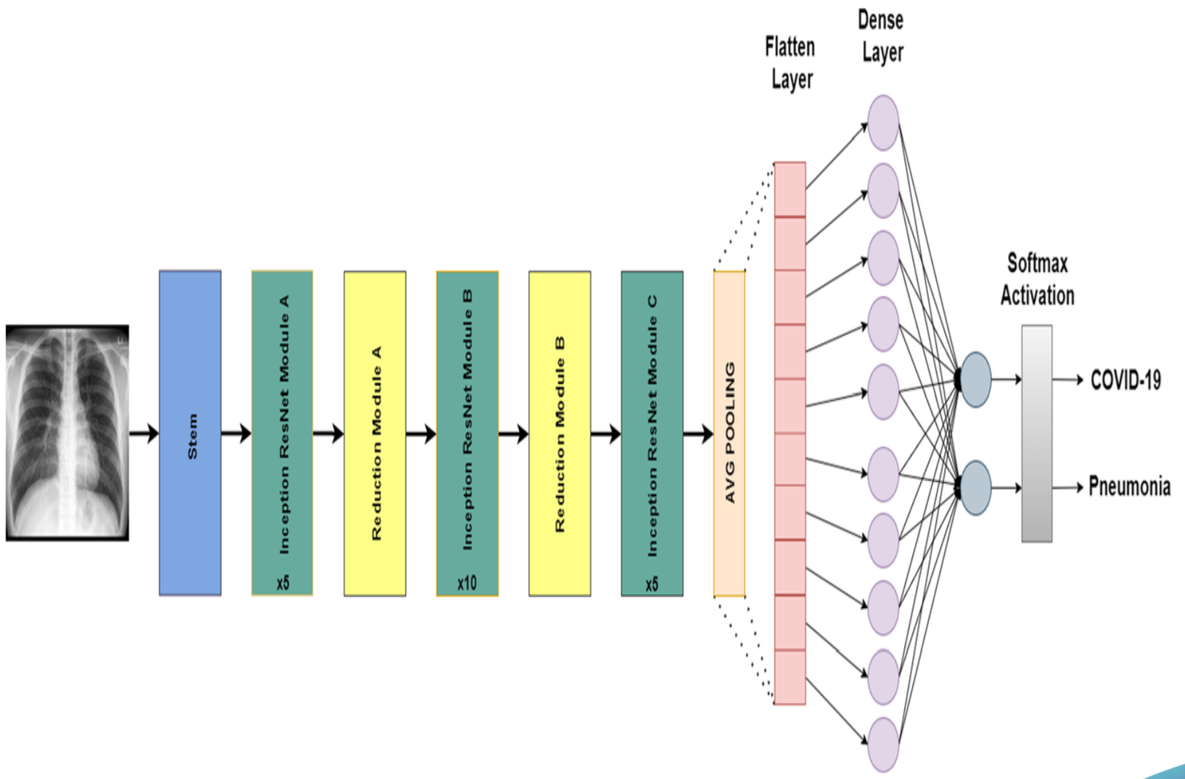


Deep learning models for image classification

InceptionResNetV2 Architecture

Common Use Cases

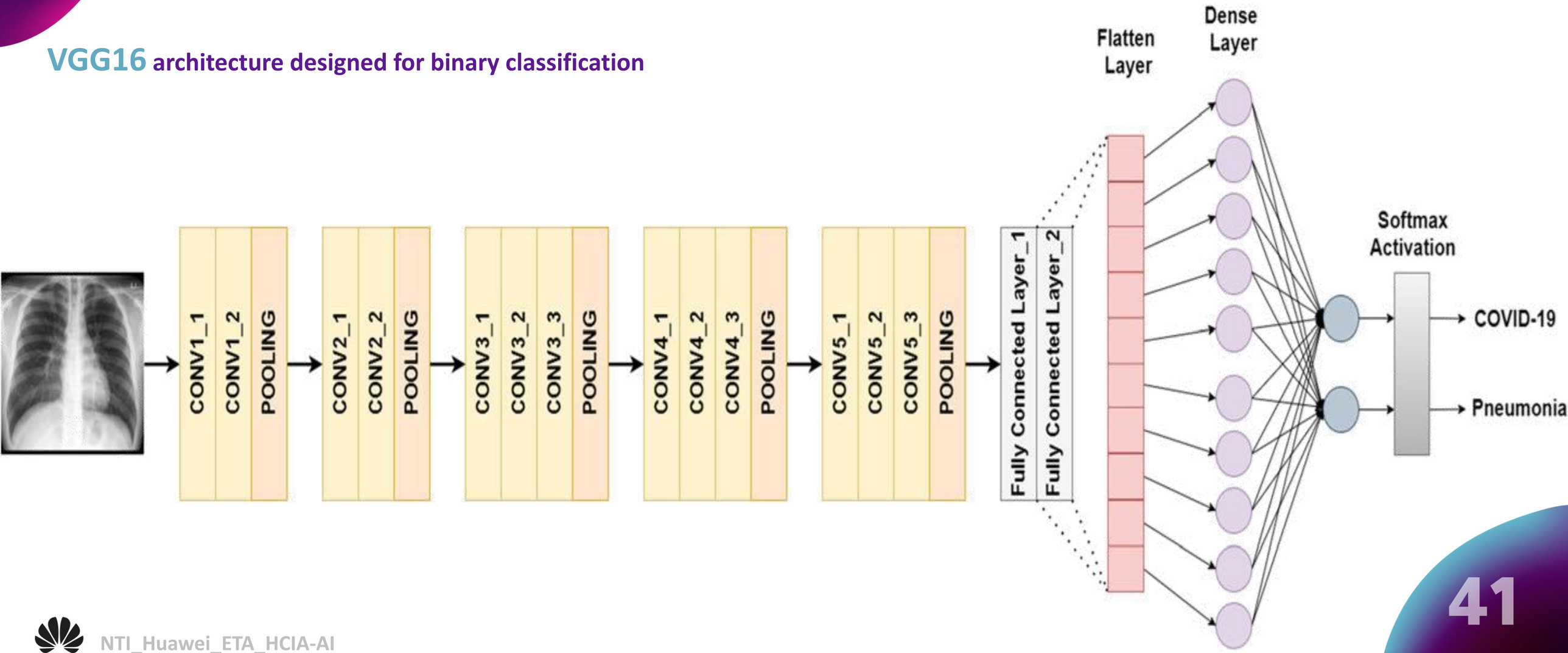
- **Medical image** analysis (e.g., **COVID-19 detection**)
- **Fine-grained classification** tasks (e.g., **plant disease detection**)
- **Industrial inspection** and **defect detection**.



A QUICK COMPARISON OF THE MODELS THAT WERE USED

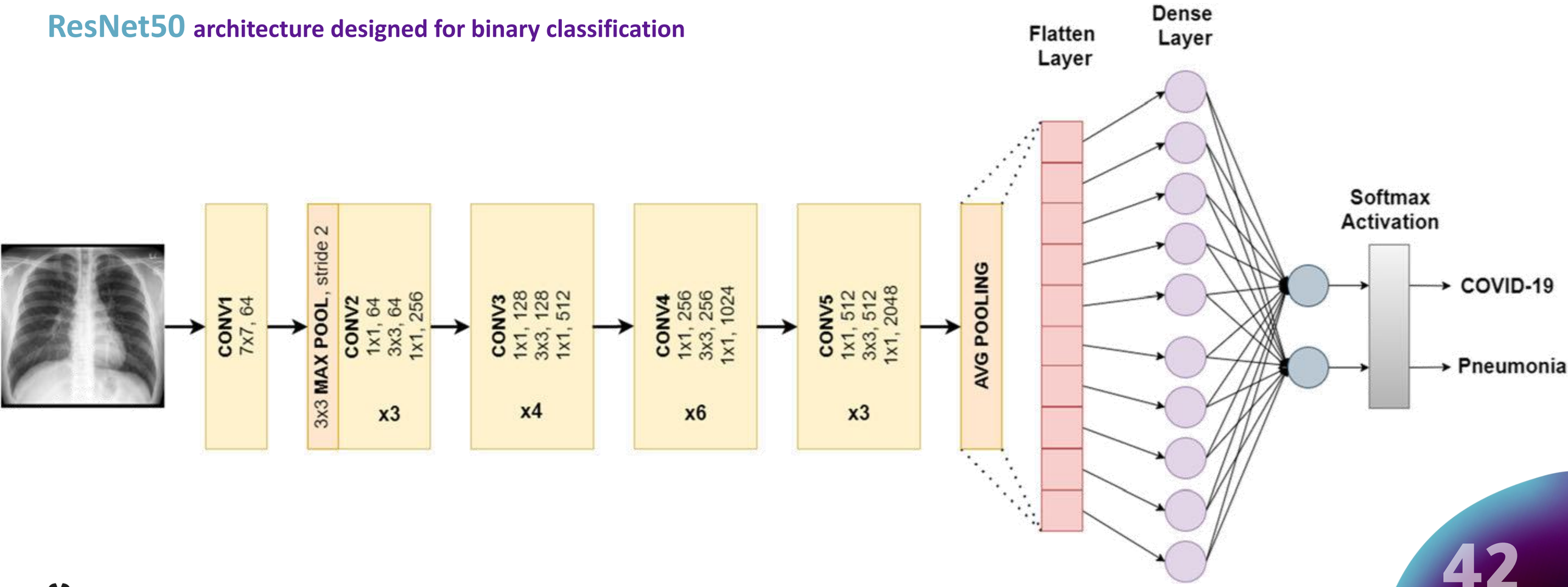
A quick comparison of the models that were used

VGG16 architecture designed for binary classification



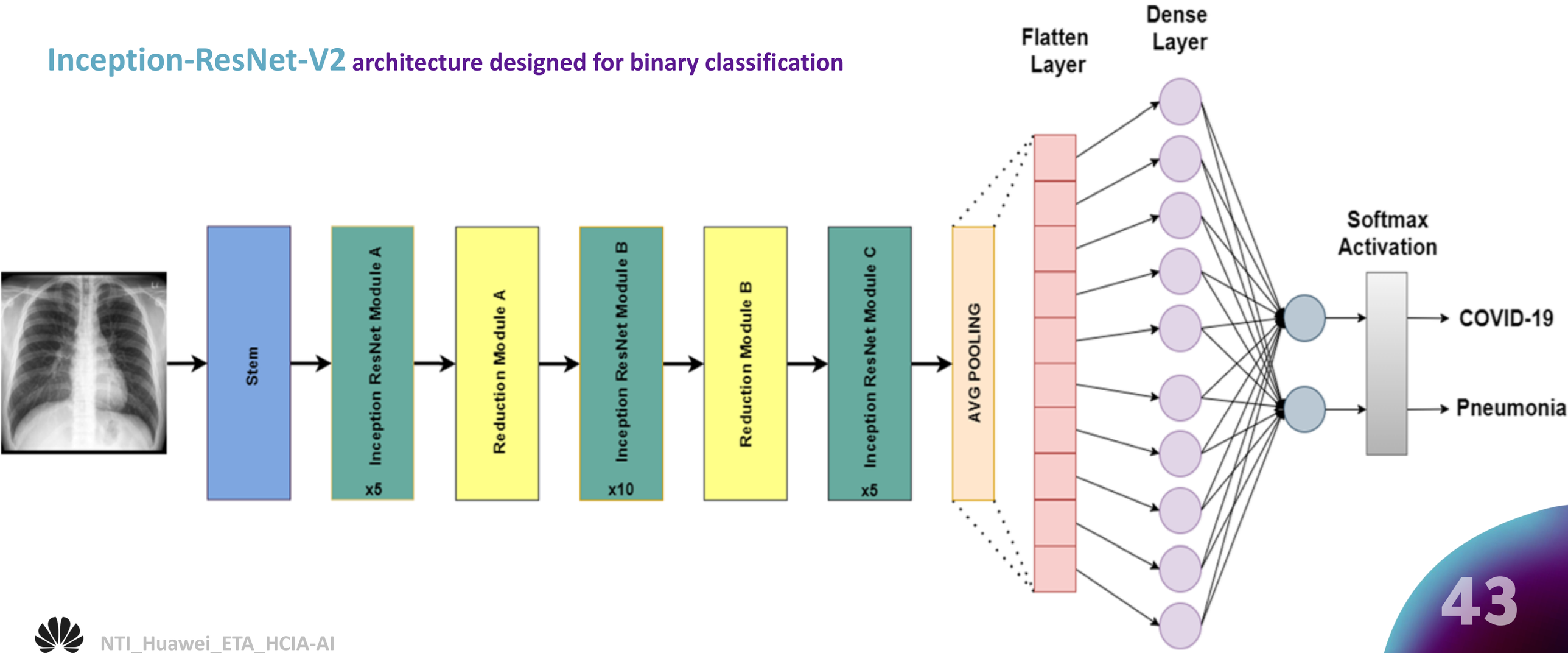
A quick comparison of the models that were used

ResNet50 architecture designed for binary classification



A quick comparison of the models that were used

Inception-ResNet-V2 architecture designed for binary classification



A quick comparison of the models that were used

Model	Number of Layers	Strengths	Limitations	Common Use Cases
InceptionResNetV2	~572	- Captures multi -scale and abstract features- Residual connections improve training stability- High classification accuracy	- Very high computational and memory cost- Requires large datasets and powerful GPUs	Medical image analysis, plant disease detection
ResNet50	50	- Efficient depth with residual connections- Good performance on moderate hardware	- May miss very fine-grained features- Not ideal for highly detailed textures	General image classification, diagnostics
VGG16	16	- Easy to understand and implement- Solid baseline for model comparison	- Large number of parameters (~138M)- Prone to overfitting- Slow training	Academic demos, basic visual recognition tasks
Custom CNN	5–15 (typical)	- Lightweight and fast- Fully adaptable to specific tasks and limited hardware	- Lower accuracy in complex domains- Limited generalization capabilities	Mobile apps, embedded systems, real-time detection

PRACTICAL APPLICATION

Practical application

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import ResNet50, InceptionResNetV2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

```
# ----------------------------------------
# 2. Basic Settings and Data Paths
# ----------------------------------------
dataset_path = "/kaggle/input/new-plant-diseases-dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)"
train_dir = os.path.join(dataset_path, "train")
test_dir = os.path.join(dataset_path, "valid")
batch_size = 32
img_size = 224
epochs = 3  # Increased for ResNet
```


https://github.com/mostafa-khalid208/Revolutionizing-Disease-Detection

Practical application

```
# ------------------------------------------------
# 3. Data Preparation and Generation
# ------------------------------------------------
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=45,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.3,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='nearest')

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical')

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)
```

Output

```
Found 70295 images belonging to 38 classes.
Found 17572 images belonging to 38 classes.
```

https://www.kaggle.com/code/mostafakhalid208/plant-disease-discovery

Practical application

```
# ----------------------------------------
# 4. Build CNN Model
# ----------------------------------------
def build_cnn_model():
    model = models.Sequential([
        layers.Conv2D(32, (3,3), activation='relu', input_shape=(img_size, img_size, 3)),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3,3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(128, (3,3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(train_generator.num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

Practical application

```
# ------------------------------------------------
# 5. Build ResNet Model with Improvements
# ------------------------------------------------
def build_resnet_model():
    base_model = ResNet50(
        weights='imagenet',
        include_top=False,
        input_shape=(img_size, img_size, 3)
    )

    # Fine-tune more layers (last 50 layers)
    for layer in base_model.layers[-50:]:
        layer.trainable = True
    for layer in base_model.layers[:-50]:
        layer.trainable = False

    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(1024, activation='relu'),  # Increased units
        layers.BatchNormalization(),
        layers.Dropout(0.4),  # Adjusted dropout
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.3),
        layers.Dense(train_generator.num_classes, activation='softmax')
    ])

    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

Practical application

```
# ----------------------------------------
# 6. Build Inception-ResNet-V2 Model
# ----------------------------------------
def build_inception_model():
    base_model = InceptionResNetV2(
        weights='imagenet',
        include_top=False,
        input_shape=(img_size, img_size, 3)
    )

    for layer in base_model.layers[-30:]:
        layer.trainable = True
    for layer in base_model.layers[:-30]:
        layer.trainable = False

    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(train_generator.num_classes, activation='softmax')
    ])

    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

Practical application

```
# ----------------------------------------
# 7. Train Models
# ----------------------------------------
# Callbacks for ResNet
early_stopping = EarlyStopping(monitor='val_loss', patience=2, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, min_lr=1e-6)

# Train Inception-ResNet-V2
inception_model = build_inception_model()
print("\n--- Training Inception-ResNet-V2 Model ---")
history_inception = inception_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=1,  # Keep 1 epoch for Inception
    validation_data=test_generator,
    validation_steps=test_generator.samples // batch_size)
```

```
--- Training Inception-ResNet-V2 Model ---
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 1191s 524ms/step - accuracy: 0.6655 - loss: 1.2271 - val_accuracy: 0.9364 - val_loss: 0.1946
```

Practical application

```
# Train CNN
cnn_model = build_cnn_model()
print("\n--- Training CNN Model ---")
history_cnn = cnn_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=1,  # Keep 1 epoch for CNN
    validation_data=test_generator,
    validation_steps=test_generator.samples // batch_size)
```

```
--- Training CNN Model ---
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 895s 405ms/step - accuracy: 0.3093 - loss: 2.4712 - val_accuracy: 0.6935 - val_loss: 0.9804
```

Practical application

```
# Train ResNet
resnet_model = build_resnet_model()
print("\n--- Training ResNet Model ---")
history_resnet = resnet_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=1,
    validation_data=test_generator,
    validation_steps=test_generator.samples // batch_size,
    callbacks=[early_stopping, reduce_lr])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step

--- Training ResNet Model ---
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 977s 428ms/step - accuracy: 0.1700 - loss: 3.3417 - val_accuracy: 0.0448 - val_loss: 31.1074 - learning_rate: 1.0000e-04
```

Practical application

```
# ------------------------------------------
# 8. Evaluation Functions
# ------------------------------------------
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix

# List to store evaluation results
results = []

def evaluate_model(model, history, model_name, test_generator):
    # Evaluate the model
    test_loss, test_acc = model.evaluate(test_generator)
    print(f'\nModel {model_name} Test Accuracy: {test_acc:.4f}, Test Loss: {test_loss:.4f}')

    # Store results
    results.append({
        'Model': model_name,
        'Test Accuracy': test_acc,
        'Test Loss': test_loss
    })

    # Predictions and classification report
    y_pred = model.predict(test_generator)
    y_pred_classes = np.argmax(y_pred, axis=1)

    print(f"\nDetailed Classification Report for {model_name}:")
    print(classification_report(test_generator.classes, y_pred_classes,
                                target_names=list(test_generator.class_indices.keys())))

    # Confusion matrix
    plt.figure(figsize=(10, 8))
    sns.heatmap(confusion_matrix(test_generator.classes, y_pred_classes),
                annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix - {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.savefig(f'{model_name}_confusion_matrix.png')
    plt.show()  # Display the plot
    plt.close()

    return history, test_acc, test_loss
```

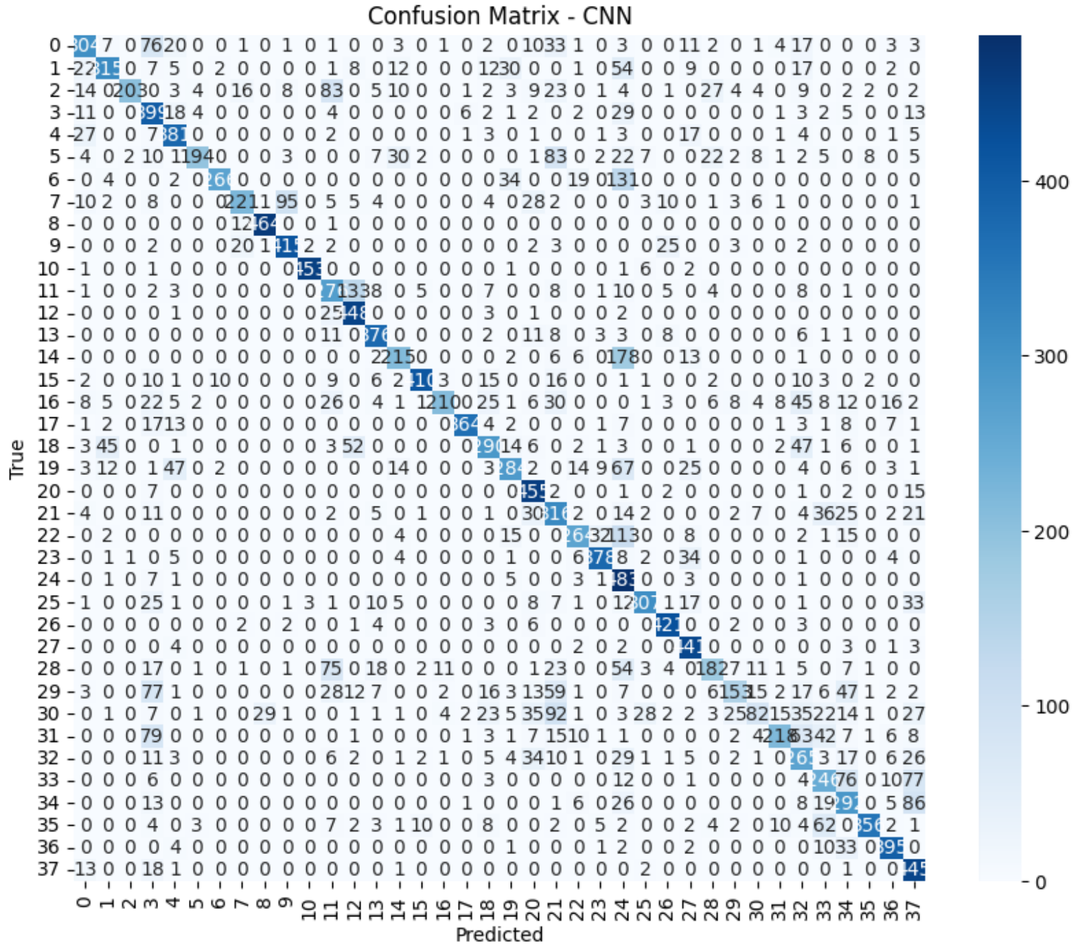
```
def plot_comparative_metrics(histories, model_names):
    plt.figure(figsize=(15, 5))

    # Training and validation accuracy
    plt.subplot(1, 2, 1)
    for history, name in zip(histories, model_names):
        plt.plot(history.history['accuracy'], marker='o', label=f'{name} Training')
        plt.plot(history.history['val_accuracy'], marker='x', linestyle='--', label=f'{name} Validation')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.grid(True)

    # Training and validation loss
    plt.subplot(1, 2, 2)
    for history, name in zip(histories, model_names):
        plt.plot(history.history['loss'], marker='o', label=f'{name} Training')
        plt.plot(history.history['val_loss'], marker='x', linestyle='--', label=f'{name} Validation')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    plt.savefig('comparative_metrics.png')
    plt.show()  # Display the plot
    plt.close()

def save_comparison_table(results):
    df = pd.DataFrame(results)
    df.to_csv('model_comparison.csv', index=False)
    print("\nModel Comparison Table:")
    print(df)
    print("Comparison table saved as 'model_comparison.csv'")

# ------------------------------------------
# 9. Execute Evaluation
# ------------------------------------------
print("\n=== Evaluating Models ===")

# Lists to store histories
histories = []
model_names = ["CNN", "ResNet", "InceptionResNetV2"]
```

Practical application

```
# ----------------------------------------
# 9. Execute Evaluation
# ----------------------------------------
print("\n=== Evaluating Models ===")

# Lists to store histories
histories = []
model_names = ["CNN", "ResNet", "InceptionResNetV2"]

# Evaluate CNN
print("\nEvaluating CNN Model...")
history_cnn, cnn_acc, cnn_loss = evaluate_model(cnn_model, history_cnn, "CNN", test_generator)
histories.append(history_cnn)

# Evaluate ResNet
print("\nEvaluating ResNet Model...")
history_resnet, resnet_acc, resnet_loss = evaluate_model(resnet_model, history_resnet, "ResNet", test_generator)
histories.append(history_resnet)

# Evaluate Inception-ResNet-V2
print("\nEvaluating Inception-ResNet-V2 Model...")
history_inception, inception_acc, inception_loss = evaluate_model(inception_model, history_inception, "InceptionResNetV2", test_generator)
histories.append(history_inception)

# Plot comparative metrics
plot_comparative_metrics(histories, model_names)

# Save comparison table
save_comparison_table(results)
```

```
=== Evaluating Models ===

Evaluating CNN Model...
550/550 ━━━━━━━━━━━━━━━━━━━━ 34s 61ms/step - accuracy: 0.6855 - loss: 1.0096

Model CNN Test Accuracy: 0.6935, Test Loss: 0.9802
550/550 ━━━━━━━━━━━━━━━━━━━━ 35s 63ms/step

Detailed Classification Report for CNN:
                                                 precision    recall  f1-score   support

                                Apple___Apple_scab       0.70      0.60      0.65       504
                                 Apple___Black_rot       0.79      0.63      0.70       497
                          Apple___Cedar_apple_rust       0.99      0.46      0.63       440
                                 Apple___healthy       0.47      0.79      0.59       502
                            Blueberry___healthy       0.73      0.84      0.78       454
          Cherry_(including_sour)___Powdery_mildew       0.93      0.46      0.62       421
                 Cherry_(including_sour)___healthy       0.95      0.58      0.72       456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot       0.81      0.54      0.65       410
                       Corn_(maize)___Common_rust_       0.94      0.97      0.95       477
              Corn_(maize)___Northern_Leaf_Blight       0.79      0.87      0.83       477
                        Corn_(maize)___healthy       0.99      0.97      0.98       465
                             Grape___Black_rot       0.49      0.58      0.53       472
                  Grape___Esca_(Black_Measles)       0.67      0.93      0.78       480
        Grape___Leaf_blight_(Isariopsis_Leaf_Spot)       0.82      0.87      0.84       430
                             Grape___healthy       0.71      0.51      0.59       423
          Orange___Haunglongbing_(Citrus_greening)       0.95      0.82      0.88       503
                        Peach___Bacterial_spot       0.91      0.46      0.61       459
                               Peach___healthy       0.97      0.84      0.90       432
                 Pepper,_bell___Bacterial_spot       0.67      0.61      0.63       478
                        Pepper,_bell___healthy       0.70      0.57      0.63       497
                         Potato___Early_blight       0.68      0.94      0.79       485
                          Potato___Late_blight       0.43      0.65      0.52       485
                              Potato___healthy       0.77      0.58      0.66       456
                           Raspberry___healthy       0.86      0.85      0.86       445
                            Soybean___healthy       0.38      0.96      0.54       505
                      Squash___Powdery_mildew       0.85      0.71      0.77       434
                     Strawberry___Leaf_scorch       0.87      0.95      0.91       444
                         Strawberry___healthy       0.74      0.97      0.84       456
                      Tomato___Bacterial_spot       0.70      0.43      0.53       425
                        Tomato___Early_blight       0.71      0.32      0.44       480
                         Tomato___Late_blight       0.57      0.18      0.27       463
                           Tomato___Leaf_Mold       0.82      0.46      0.59       470
                  Tomato___Septoria_leaf_spot       0.45      0.61      0.52       436
Tomato___Spider_mites Two-spotted_spider_mite       0.53      0.57      0.54       435
                         Tomato___Target_Spot       0.50      0.64      0.56       457
       Tomato___Tomato_Yellow_Leaf_Curl_Virus       0.96      0.73      0.83       490
                 Tomato___Tomato_mosaic_virus       0.85      0.88      0.87       448
                             Tomato___healthy       0.57      0.93      0.71       481

                                     accuracy                           0.69     17572
                                    macro avg       0.74      0.69      0.69     17572
                                 weighted avg       0.74      0.69      0.69     17572
```

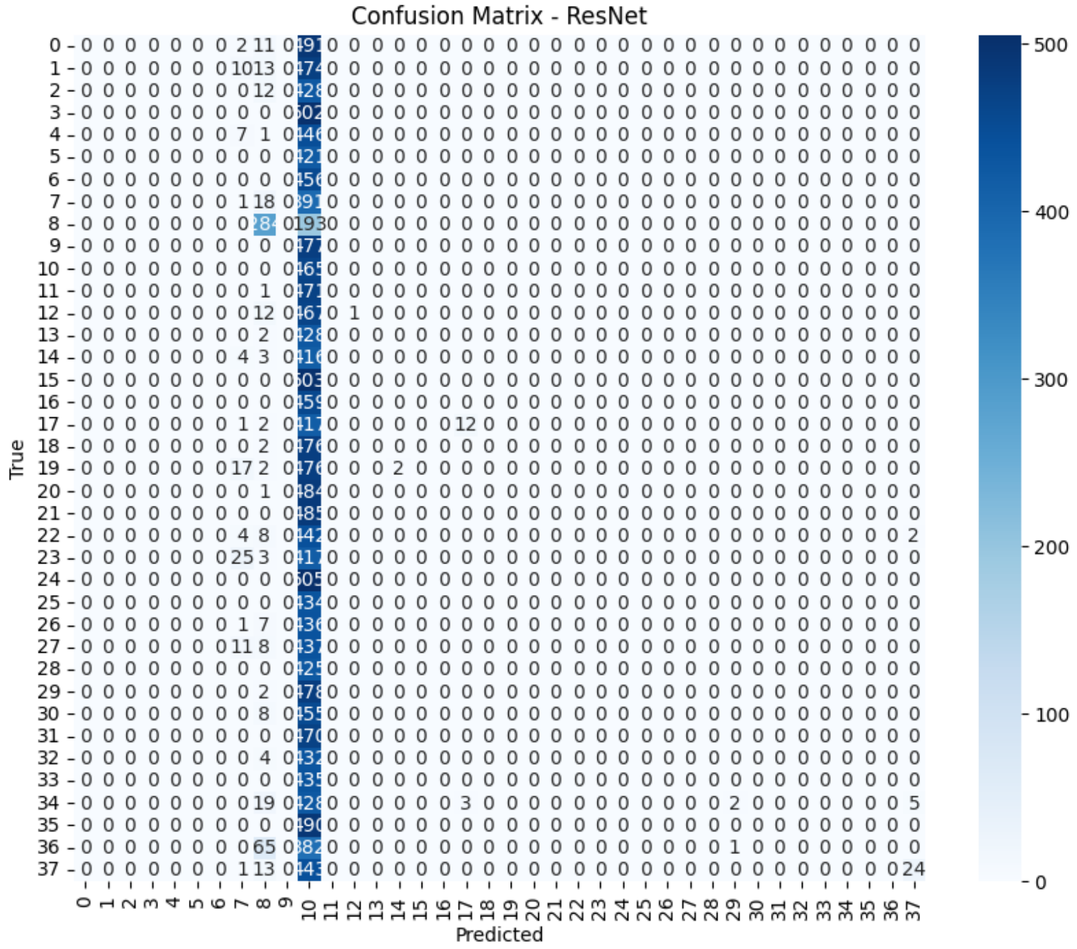
Practical application




```
Evaluating ResNet Model...
550/550 ━━━━━━━━━━━━━━━━━━━━ 54s 97ms/step - accuracy: 0.0589 - loss: 29.1654

Model ResNet Test Accuracy: 0.0448, Test Loss: 31.1037
550/550 ━━━━━━━━━━━━━━━━━━━━ 51s 93ms/step

Detailed Classification Report for ResNet:
```

	precision	recall	f1-score	support
Apple___Apple_scab	0.00	0.00	0.00	504
Apple___Black_rot	0.00	0.00	0.00	497
Apple___Cedar_apple_rust	0.00	0.00	0.00	440
Apple___healthy	0.00	0.00	0.00	502
Blueberry___healthy	0.00	0.00	0.00	454
Cherry_(including_sour)___Powdery_mildew	0.00	0.00	0.00	421
Cherry_(including_sour)___healthy	0.00	0.00	0.00	456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.01	0.00	0.00	410
Corn_(maize)___Common_rust_	0.57	0.60	0.58	477
Corn_(maize)___Northern_Leaf_Blight	0.00	0.00	0.00	477
Corn_(maize)___healthy	0.03	1.00	0.05	465
Grape___Black_rot	0.00	0.00	0.00	472
Grape___Esca_(Black_Measles)	1.00	0.00	0.00	480
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0.00	0.00	0.00	430
Grape___healthy	0.00	0.00	0.00	423
Orange___Haunglongbing_(Citrus_greening)	0.00	0.00	0.00	503
Peach___Bacterial_spot	0.00	0.00	0.00	459
Peach___healthy	0.80	0.03	0.05	432
Pepper,_bell___Bacterial_spot	0.00	0.00	0.00	478
Pepper,_bell___healthy	0.00	0.00	0.00	497
Potato___Early_blight	0.00	0.00	0.00	485
Potato___Late_blight	0.00	0.00	0.00	485
Potato___healthy	0.00	0.00	0.00	456
Raspberry___healthy	0.00	0.00	0.00	445
Soybean___healthy	0.00	0.00	0.00	505
Squash___Powdery_mildew	0.00	0.00	0.00	434
Strawberry___Leaf_scorch	0.00	0.00	0.00	444
Strawberry___healthy	0.00	0.00	0.00	456
Tomato___Bacterial_spot	0.00	0.00	0.00	425
Tomato___Early_blight	0.00	0.00	0.00	480
Tomato___Late_blight	0.00	0.00	0.00	463
Tomato___Leaf_Mold	0.00	0.00	0.00	470
Tomato___Septoria_leaf_spot	0.00	0.00	0.00	436
Tomato___Spider_mites Two-spotted_spider_mite	0.00	0.00	0.00	435
Tomato___Target_Spot	0.00	0.00	0.00	457
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.00	0.00	0.00	490
Tomato___Tomato_mosaic_virus	0.00	0.00	0.00	448
Tomato___healthy	0.77	0.05	0.09	481
accuracy			0.04	17572
macro avg	0.08	0.04	0.02	17572
weighted avg	0.08	0.04	0.02	17572

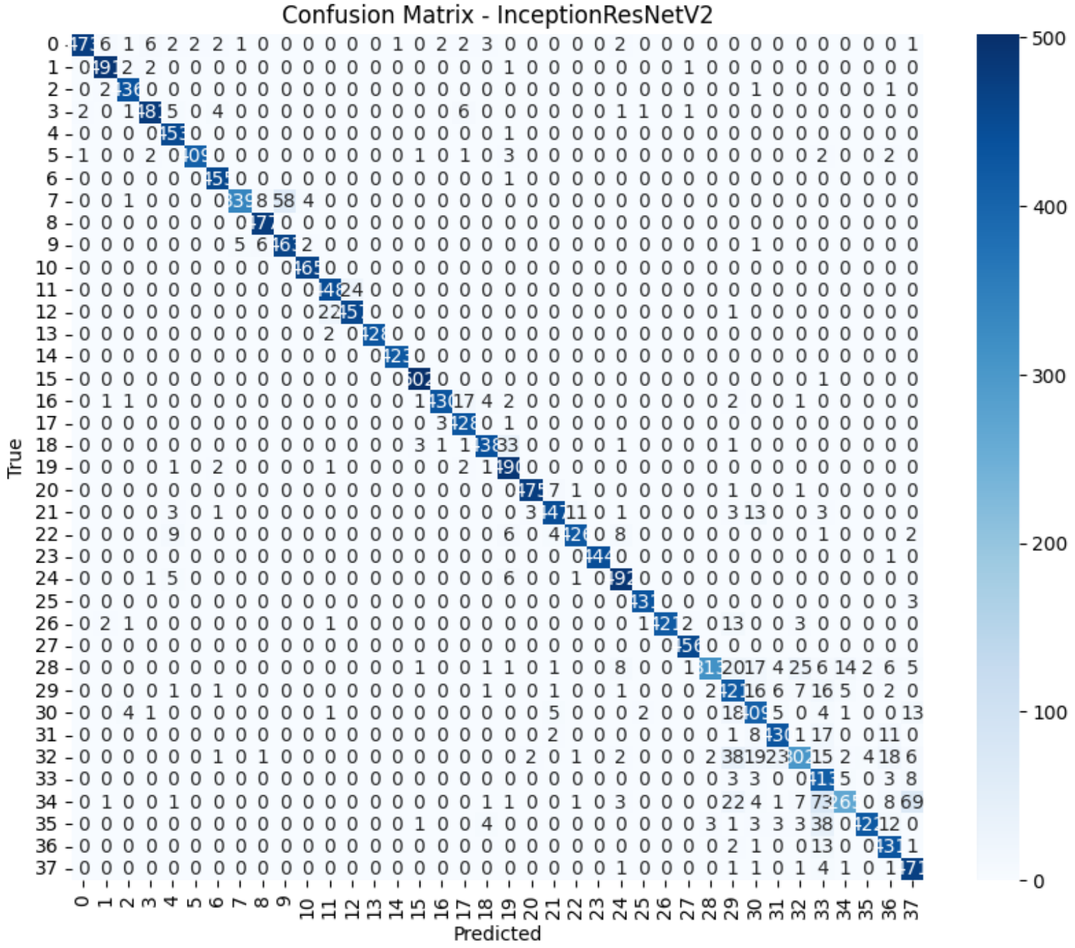
Practical application

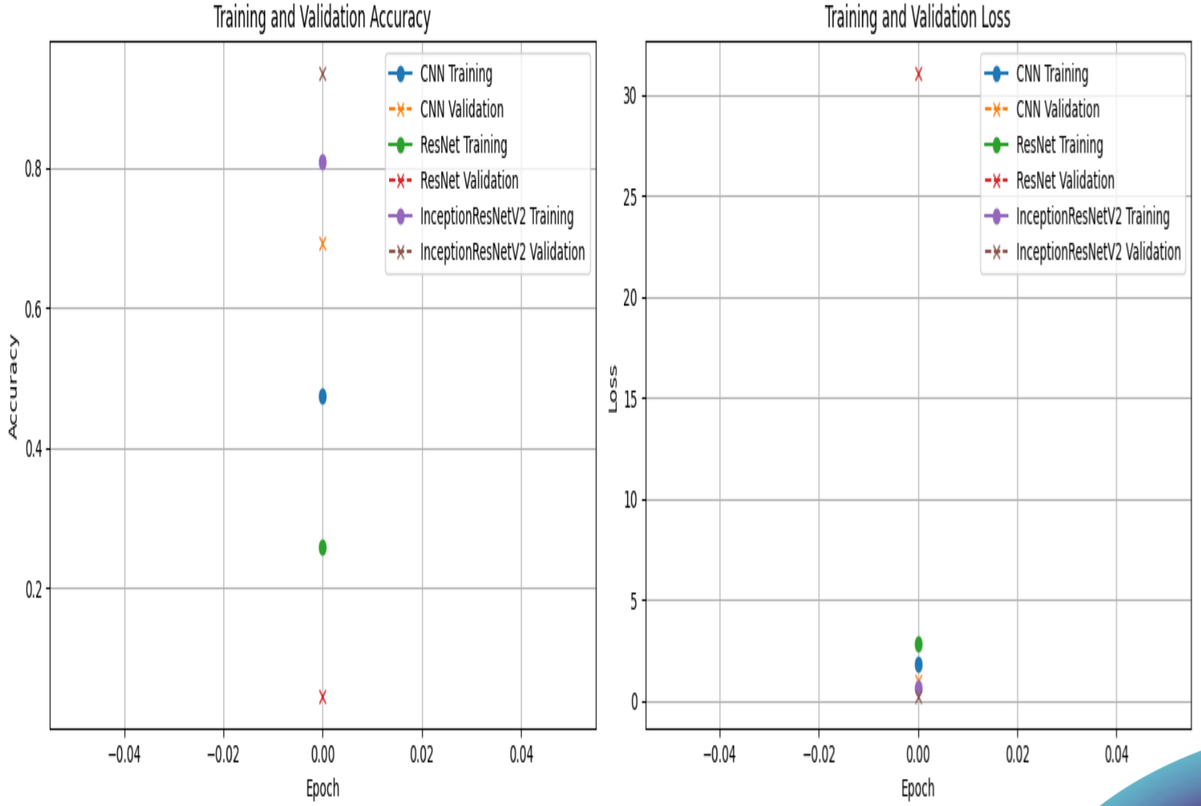



```
Evaluating Inception-ResNet-V2 Model...
550/550 ━━━━━━━━━━━━━━━━━━━━ 87s 157ms/step - accuracy: 0.9615 - loss: 0.1216

Model InceptionResNetV2 Test Accuracy: 0.9364, Test Loss: 0.1946
550/550 ━━━━━━━━━━━━━━━━━━━━ 86s 157ms/step

Detailed Classification Report for InceptionResNetV2:
```

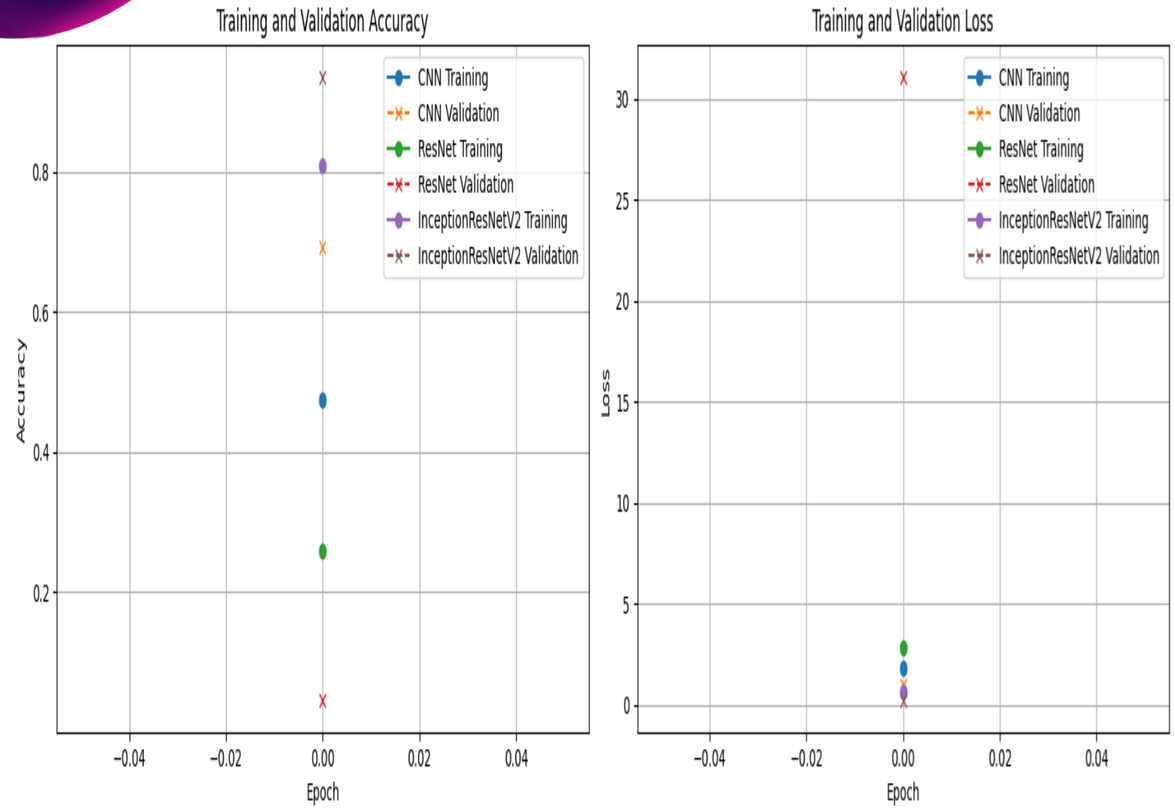
	precision	recall	f1-score	support
Apple___Apple_scab	0.99	0.94	0.97	504
Apple___Black_rot	0.98	0.99	0.98	497
Apple___Cedar_apple_rust	0.98	0.99	0.98	440
Apple___healthy	0.98	0.96	0.97	502
Blueberry___healthy	0.94	1.00	0.97	454
Cherry_(including_sour)___Powdery_mildew	1.00	0.97	0.98	421
Cherry_(including_sour)___healthy	0.98	1.00	0.99	456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.98	0.83	0.90	410
Corn_(maize)___Common_rust_	0.97	1.00	0.98	477
Corn_(maize)___Northern_Leaf_Blight	0.89	0.97	0.93	477
Corn_(maize)___healthy	0.99	1.00	0.99	465
Grape___Black_rot	0.94	0.95	0.95	472
Grape___Esca_(Black_Measles)	0.95	0.95	0.95	480
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	1.00	1.00	430
Grape___healthy	1.00	1.00	1.00	423
Orange___Haunglongbing_(Citrus_greening)	0.99	1.00	0.99	503
Peach___Bacterial_spot	0.99	0.94	0.96	459
Peach___healthy	0.94	0.99	0.96	432
Pepper,_bell___Bacterial_spot	0.97	0.92	0.94	478
Pepper,_bell___healthy	0.90	0.99	0.94	497
Potato___Early_blight	0.99	0.98	0.98	485
Potato___Late_blight	0.96	0.92	0.94	485
Potato___healthy	0.97	0.93	0.95	456
Raspberry___healthy	1.00	1.00	1.00	445
Soybean___healthy	0.95	0.97	0.96	505
Squash___Powdery_mildew	0.99	0.99	0.99	434
Strawberry___Leaf_scorch	1.00	0.95	0.97	444
Strawberry___healthy	0.99	1.00	0.99	456
Tomato___Bacterial_spot	0.98	0.74	0.84	425
Tomato___Early_blight	0.77	0.88	0.82	480
Tomato___Late_blight	0.82	0.88	0.85	463
Tomato___Leaf_Mold	0.91	0.91	0.91	470
Tomato___Septoria_leaf_spot	0.86	0.69	0.77	436
Tomato___Spider_mites Two-spotted_spider_mite	0.68	0.95	0.79	435
Tomato___Target_Spot	0.90	0.58	0.71	457
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.99	0.86	0.92	490
Tomato___Tomato_mosaic_virus	0.87	0.96	0.91	448
Tomato___healthy	0.81	0.98	0.89	481
accuracy			0.94	17572
macro avg	0.94	0.94	0.94	17572
weighted avg	0.94	0.94	0.94	17572

Practical application



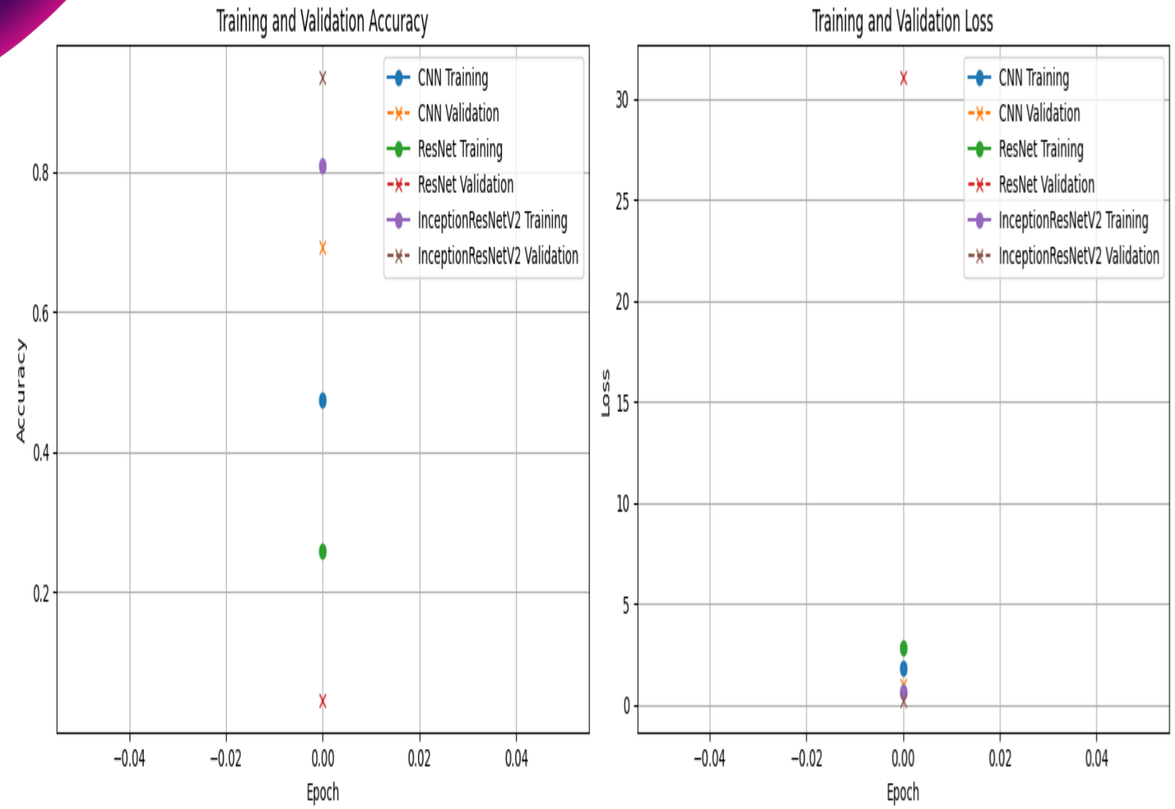


Practical application




```
Model Comparison Table:
                 Model  Test Accuracy  Test Loss
0                  CNN       0.693547   0.980185
1               ResNet       0.044787  31.103748
2  InceptionResNetV2       0.936433   0.194551
Comparison table saved as 'model_comparison.csv'
```

Practical application




```
Model Comparison Table:
                 Model  Test Accuracy  Test Loss
0                  CNN       0.693547   0.980185
1               ResNet       0.044787  31.103748
2  InceptionResNetV2       0.936433   0.194551
Comparison table saved as 'model_comparison.csv'
```

Practical application

Inception-ResNet-V2 (Epoch=3)

```
# ----------------------------------------
# 4. Build Inception-ResNet-V2 Model
# ----------------------------------------
def build_inception_model():
    base_model = InceptionResNetV2(
        weights='imagenet',
        include_top=False,
        input_shape=(img_size, img_size, 3)
    )

    for layer in base_model.layers[-30:]:
        layer.trainable = True
    for layer in base_model.layers[:-30]:
        layer.trainable = False

    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(train_generator.num_classes, activation='softmax')
    ])

    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

```
# Train Inception-ResNet-V2
inception_model = build_inception_model()
print("\n--- Training Inception-ResNet-V2 Model ---")
history_inception = inception_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=3,
    validation_data=test_generator,
    validation_steps=test_generator.samples // batch_size)
```

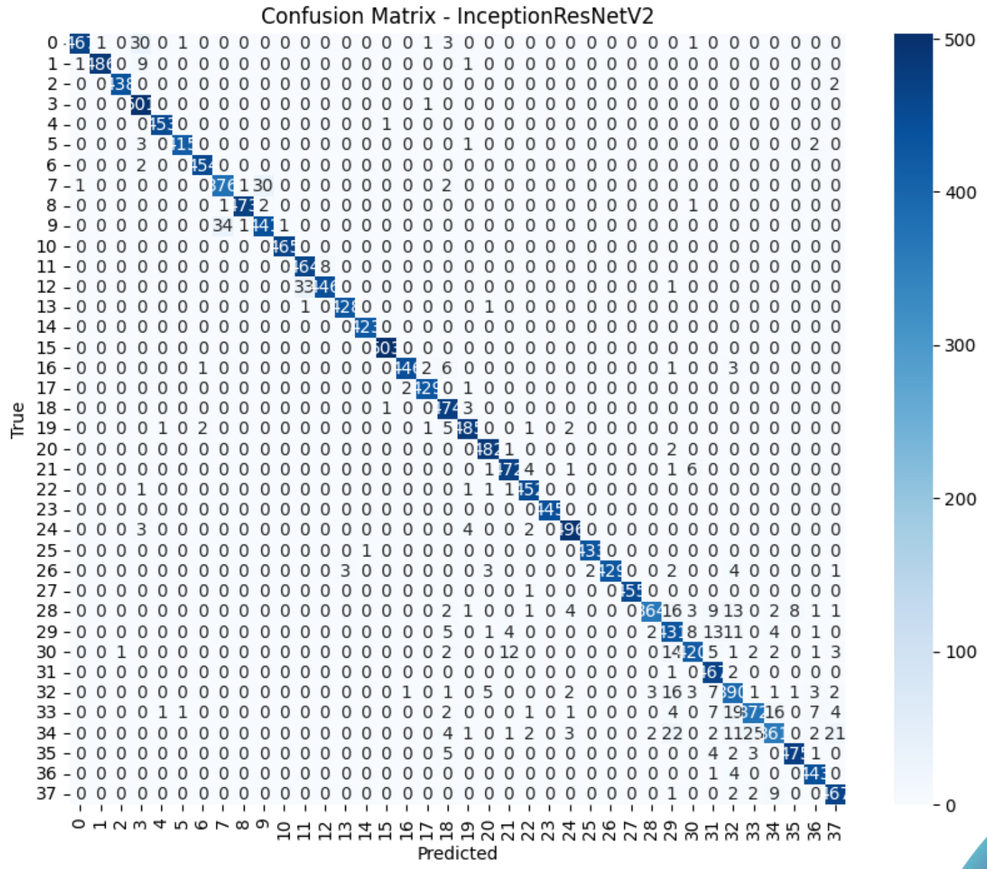
```
--- Training Inception-ResNet-V2 Model ---
Epoch 1/3
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 1040s 455ms/step - accuracy: 0.6668 - loss: 1.2330 - val_accuracy: 0.9445 - val_loss: 0.1658
Epoch 2/3
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 79s 36ms/step - accuracy: 0.8750 - loss: 0.2283 - val_accuracy: 0.9443 - val_loss: 0.1650
Epoch 3/3
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 1016s 463ms/step - accuracy: 0.9251 - loss: 0.2385 - val_accuracy: 0.9629 - val_loss: 0.1092
```

https://www.kaggle.com/code/mostafakhalid208/plant-disease-discovery-inceptionresnetv2

Practical application

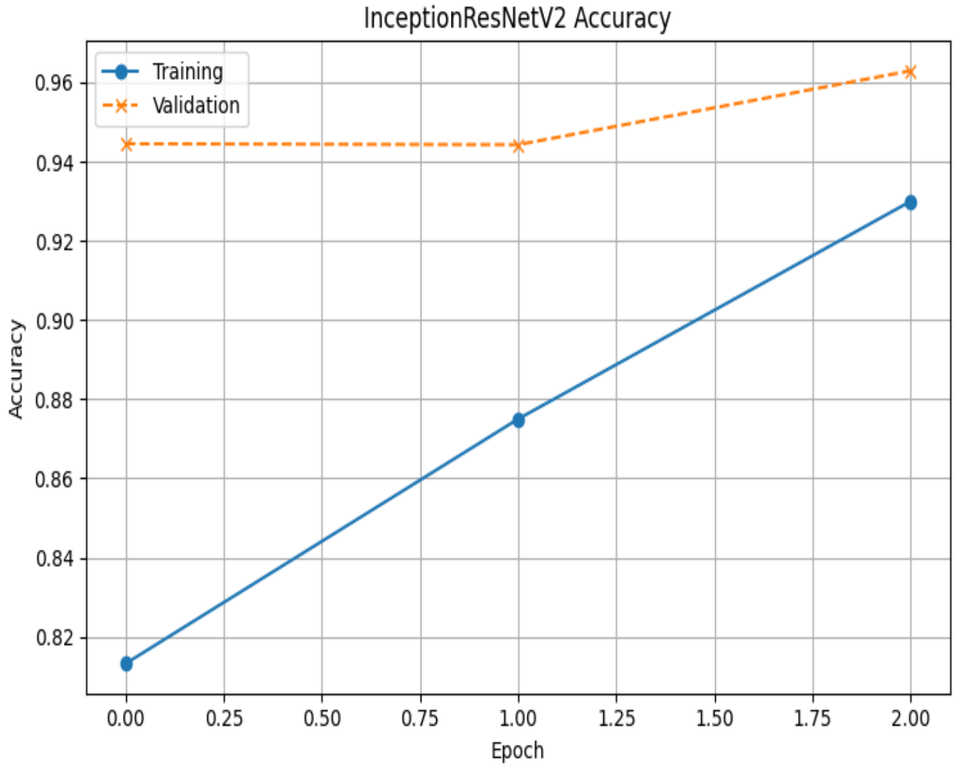
Inception-ResNet-V2 (Epoch=3)

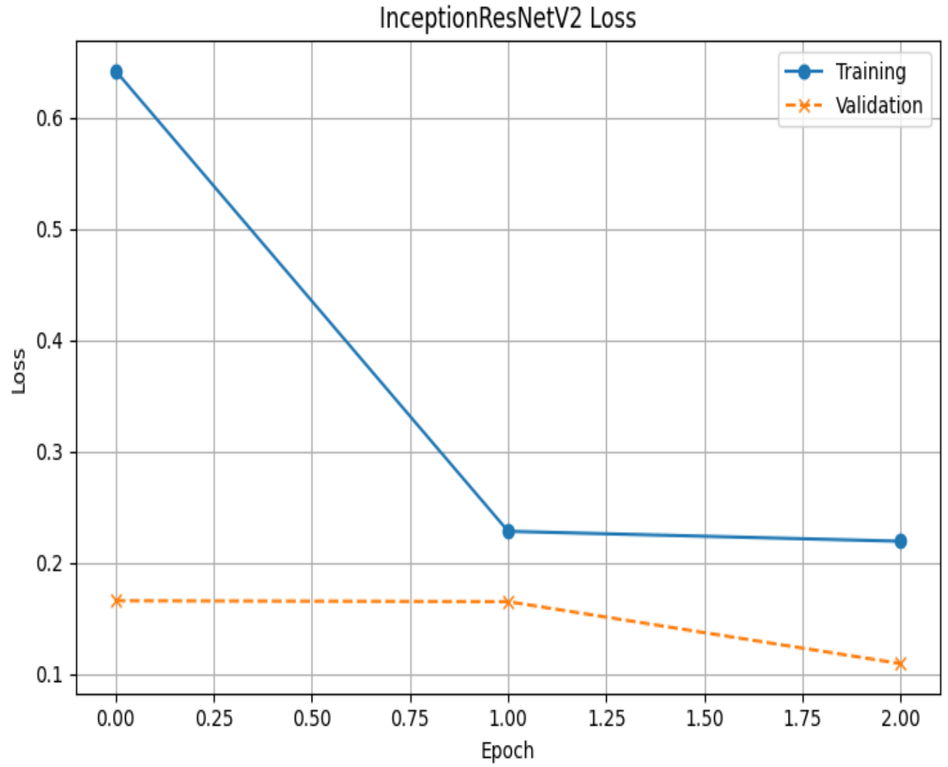
```
=== Evaluating InceptionResNetV2 Model ===

Evaluating Inception-ResNet-V2 Model...
550/550 ━━━━━━━━━━━━━━━━━━━━ 80s 146ms/step - accuracy: 0.9718 - loss: 0.0841

Model InceptionResNetV2 Test Accuracy: 0.9630, Test Loss: 0.1092
550/550 ━━━━━━━━━━━━━━━━━━━━ 104s 162ms/step

Detailed Classification Report for InceptionResNetV2:
                                              precision    recall  f1-score   support

                                Apple___Apple_scab       1.00      0.93      0.96       504
                                 Apple___Black_rot       1.00      0.98      0.99       497
                          Apple___Cedar_apple_rust       1.00      1.00      1.00       440
                                 Apple___healthy         0.91      1.00      0.95       502
                            Blueberry___healthy          1.00      1.00      1.00       454
          Cherry_(including_sour)___Powdery_mildew       1.00      0.99      0.99       421
                 Cherry_(including_sour)___healthy       0.99      1.00      0.99       456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot       0.91      0.92      0.92       410
                       Corn_(maize)___Common_rust_       1.00      0.99      0.99       477
              Corn_(maize)___Northern_Leaf_Blight       0.93      0.92      0.93       477
                       Corn_(maize)___healthy          1.00      1.00      1.00       465
                            Grape___Black_rot          0.93      0.98      0.96       472
                 Grape___Esca_(Black_Measles)          0.98      0.93      0.96       480
      Grape___Leaf_blight_(Isariopsis_Leaf_Spot)       0.99      1.00      0.99       430
                            Grape___healthy          1.00      1.00      1.00       423
        Orange___Haunglongbing_(Citrus_greening)       1.00      1.00      1.00       503
                          Peach___Bacterial_spot       0.99      0.97      0.98       459
                                 Peach___healthy       0.99      0.99      0.99       432
                Pepper,_bell___Bacterial_spot          0.93      0.99      0.96       478
                       Pepper,_bell___healthy          0.97      0.98      0.97       497
                        Potato___Early_blight          0.98      0.99      0.98       485
                         Potato___Late_blight          0.96      0.97      0.97       485
                            Potato___healthy           0.97      0.99      0.98       456
                         Raspberry___healthy           1.00      1.00      1.00       445
                          Soybean___healthy            0.97      0.98      0.98       505
                       Squash___Powdery_mildew         1.00      1.00      1.00       434
                      Strawberry___Leaf_scorch         1.00      0.97      0.98       444
                          Strawberry___healthy         1.00      1.00      1.00       456
                       Tomato___Bacterial_spot         0.98      0.86      0.91       425
                         Tomato___Early_blight         0.84      0.90      0.87       480
                          Tomato___Late_blight         0.95      0.91      0.93       463
                            Tomato___Leaf_Mold         0.91      0.99      0.95       470
                     Tomato___Septoria_leaf_spot       0.84      0.89      0.87       436
   Tomato___Spider_mites Two-spotted_spider_mite       0.92      0.86      0.89       435
                          Tomato___Target_Spot       0.91      0.79      0.85       457
        Tomato___Tomato_Yellow_Leaf_Curl_Virus       0.98      0.97      0.98       490
                  Tomato___Tomato_mosaic_virus       0.96      0.99      0.97       448
                            Tomato___healthy         0.93      0.97      0.95       481

                                        accuracy                         0.96     17572
                                       macro avg       0.96      0.96      0.96     17572
                                    weighted avg       0.96      0.96      0.96     17572
```



Practical application

Inception-ResNet-V2 (Epoch=3)






```
Model Results:
                Model  Test Accuracy  Test Loss
0  InceptionResNetV2       0.962952   0.109196
Results saved as 'model_results.csv'
```

Practical application

Inception-ResNet-V2 (Epoch=5)

```python
# ----------------------------------------
# 4. Build Inception-ResNet-V2 Model
# ----------------------------------------
def build_inception_model():
    base_model = InceptionResNetV2(
        weights='imagenet',
        include_top=False,
        input_shape=(img_size, img_size, 3)
    )

    for layer in base_model.layers[-30:]:
        layer.trainable = True
    for layer in base_model.layers[:-30]:
        layer.trainable = False

    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(train_generator.num_classes, activation='softmax')
    ])

    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

```python
# Train Inception-ResNet-V2
inception_model = build_inception_model()
print("\n--- Training Inception-ResNet-V2 Model ---")
history_inception = inception_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=5,
    validation_data=test_generator,
    validation_steps=test_generator.samples // batch_size)
```

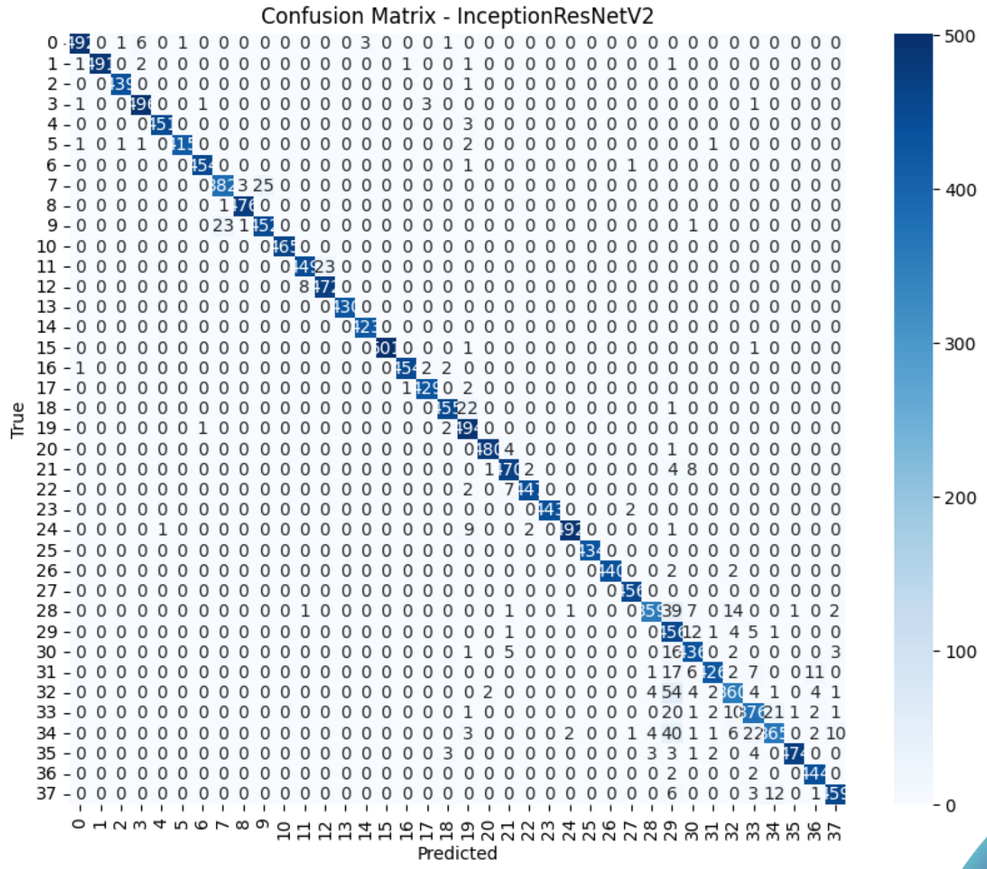
```
--- Training Inception-ResNet-V2 Model ---
Epoch 1/5
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 949s 414ms/step - accuracy: 0.6541 - loss: 1.2864 - val_accuracy: 0.9345 - val_loss: 0.1977
Epoch 2/5
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 82s 37ms/step - accuracy: 0.9688 - loss: 0.2091 - val_accuracy: 0.9343 - val_loss: 0.1970
Epoch 3/5
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 901s 410ms/step - accuracy: 0.9226 - loss: 0.2442 - val_accuracy: 0.9603 - val_loss: 0.1192
Epoch 4/5
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 82s 37ms/step - accuracy: 0.9375 - loss: 0.1194 - val_accuracy: 0.9603 - val_loss: 0.1195
Epoch 5/5
2196/2196 ━━━━━━━━━━━━━━━━━━━━ 919s 419ms/step - accuracy: 0.9423 - loss: 0.1805 - val_accuracy: 0.9639 - val_loss: 0.1033
```


https://www.kaggle.com/code/mostafakhalid208/plant-disease-discovery-inceptionresnetv2-5epochs

Practical application

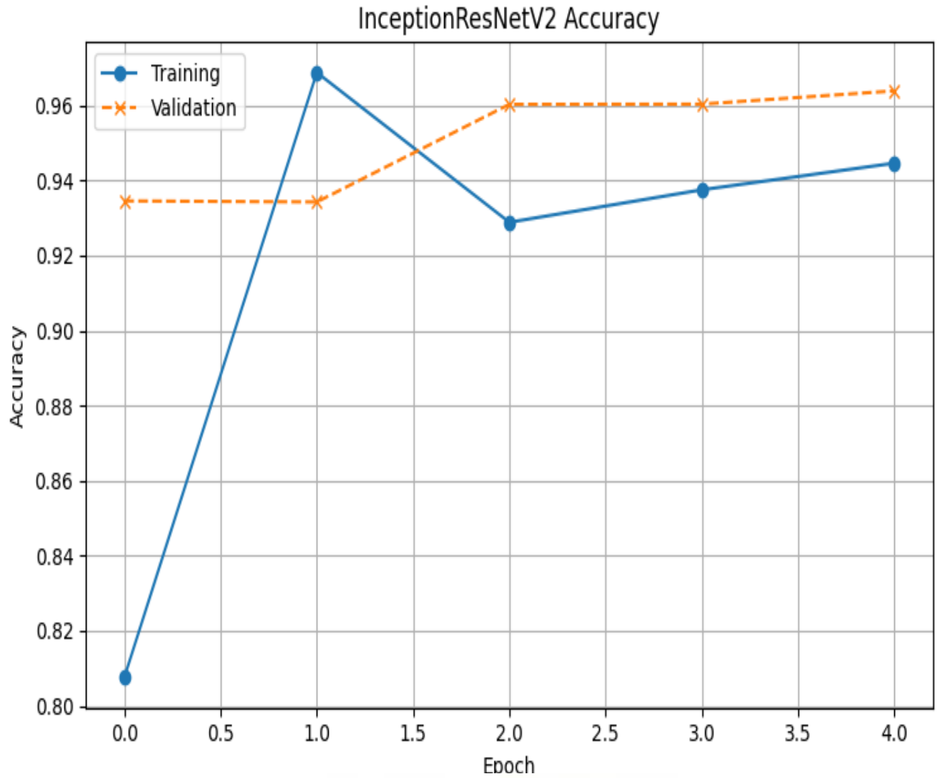
Inception-ResNet-V2 (Epoch=5)

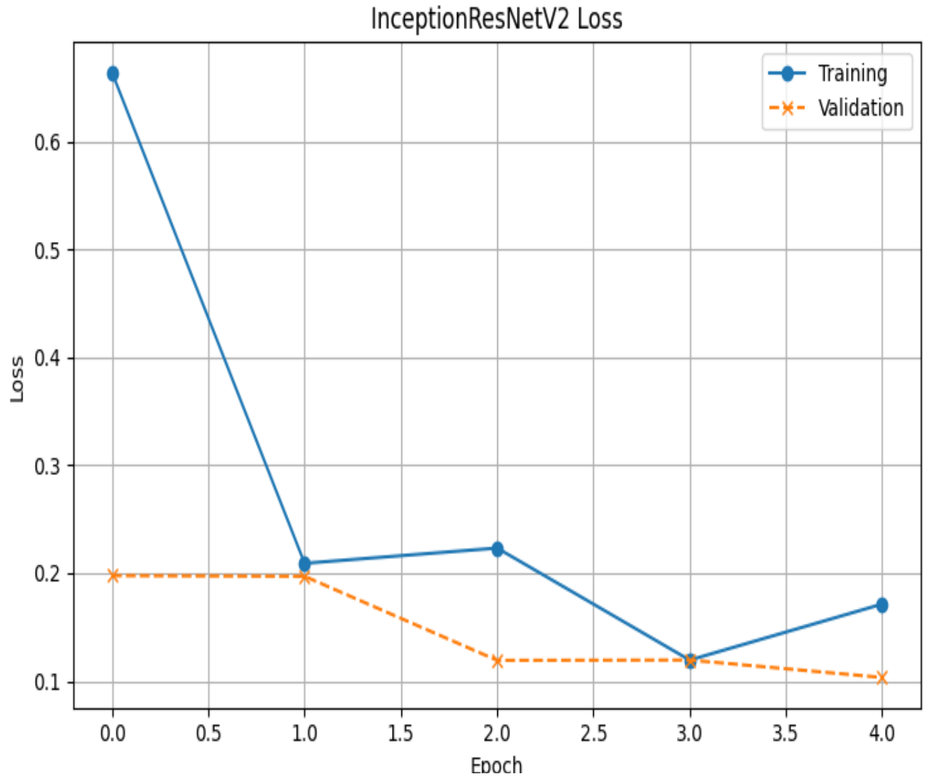
```
=== Evaluating InceptionResNetV2 Model ===

Evaluating Inception-ResNet-V2 Model...
550/550 ━━━━━━━━━━━━━━━━━━━━ 88s 161ms/step - accuracy: 0.9804 - loss: 0.0583

Model InceptionResNetV2 Test Accuracy: 0.9639, Test Loss: 0.1033
550/550 ━━━━━━━━━━━━━━━━━━━━ 108s 173ms/step

Detailed Classification Report for InceptionResNetV2:
                                                    precision    recall  f1-score   support

                                Apple___Apple_scab       0.99      0.98      0.98       504
                                 Apple___Black_rot       1.00      0.99      0.99       497
                          Apple___Cedar_apple_rust       1.00      1.00      1.00       440
                                   Apple___healthy       0.98      0.99      0.99       502
                               Blueberry___healthy       1.00      0.99      1.00       454
          Cherry_(including_sour)___Powdery_mildew       1.00      0.99      0.99       421
                 Cherry_(including_sour)___healthy       1.00      1.00      1.00       456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot       0.94      0.93      0.94       410
                       Corn_(maize)___Common_rust_       0.99      1.00      0.99       477
               Corn_(maize)___Northern_Leaf_Blight       0.95      0.95      0.95       477
                         Corn_(maize)___healthy       1.00      1.00      1.00       465
                             Grape___Black_rot       0.98      0.95      0.97       472
                  Grape___Esca_(Black_Measles)       0.95      0.98      0.97       480
       Grape___Leaf_blight_(Isariopsis_Leaf_Spot)       1.00      1.00      1.00       430
                               Grape___healthy       0.99      1.00      1.00       423
         Orange___Haunglongbing_(citrus_greening)       1.00      1.00      1.00       503
                        Peach___Bacterial_spot       1.00      0.99      0.99       459
                               Peach___healthy       0.99      0.99      0.99       432
                  Pepper,_bell___Bacterial_spot       0.98      0.95      0.97       478
                         Pepper,_bell___healthy       0.91      0.99      0.95       497
                          Potato___Early_blight       0.99      0.99      0.99       485
                           Potato___Late_blight       0.96      0.97      0.97       485
                               Potato___healthy       0.99      0.98      0.99       456
                            Raspberry___healthy       1.00      1.00      1.00       445
                              Soybean___healthy       0.99      0.97      0.98       505
                        Squash___Powdery_mildew       1.00      1.00      1.00       434
                       Strawberry___Leaf_scorch       1.00      0.99      1.00       444
                           Strawberry___healthy       0.99      1.00      1.00       456
                        Tomato___Bacterial_spot       0.97      0.84      0.90       425
                          Tomato___Early_blight       0.69      0.95      0.80       480
                           Tomato___Late_blight       0.91      0.94      0.93       463
                             Tomato___Leaf_Mold       0.98      0.91      0.94       470
                    Tomato___Septoria_leaf_spot       0.90      0.83      0.86       436
  Tomato___Spider_mites Two-spotted_spider_mite       0.88      0.86      0.87       435
                           Tomato___Target_Spot       0.91      0.80      0.85       457
         Tomato___Tomato_Yellow_Leaf_Curl_Virus       1.00      0.97      0.98       490
                   Tomato___Tomato_mosaic_virus       0.96      0.99      0.97       448
                               Tomato___healthy       0.96      0.95      0.96       481

                                       accuracy                           0.96     17572
                                      macro avg       0.97      0.96      0.96     17572
```



Practical application

Inception-ResNet-V2 (Epoch=5)






```
Model Results:
                 Model  Test Accuracy  Test Loss
0  InceptionResNetV2       0.963863   0.103289
Results saved as 'model_results.csv'
```

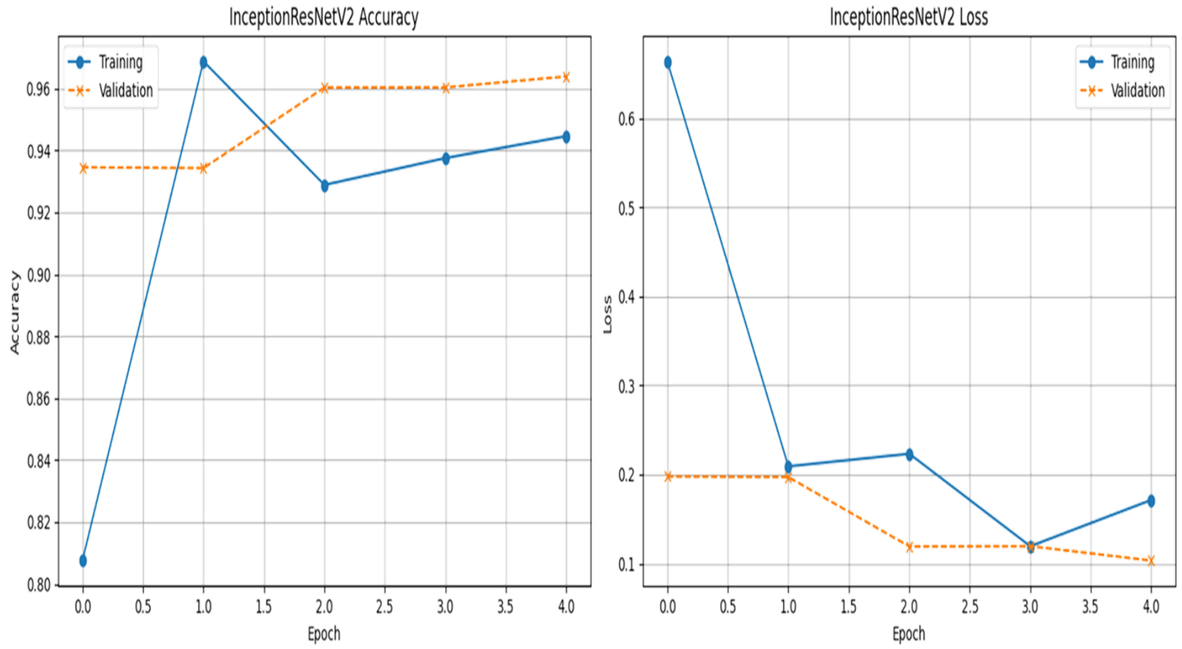
CONCLUSION

Conclusion

- "This study **showed deep learning (InceptionResNetV2)** effectively **detects plant diseases** from leaf images, achieving **high accuracy** rapidly using a **diverse dataset** and **data augmentation**."

Key Points Retained:

- **Disease detection via inceptionresnetv2**.
- Critical methods (**dataset diversity** + **augmentation**).
- **Efficiency high accuracy with minimal training**.



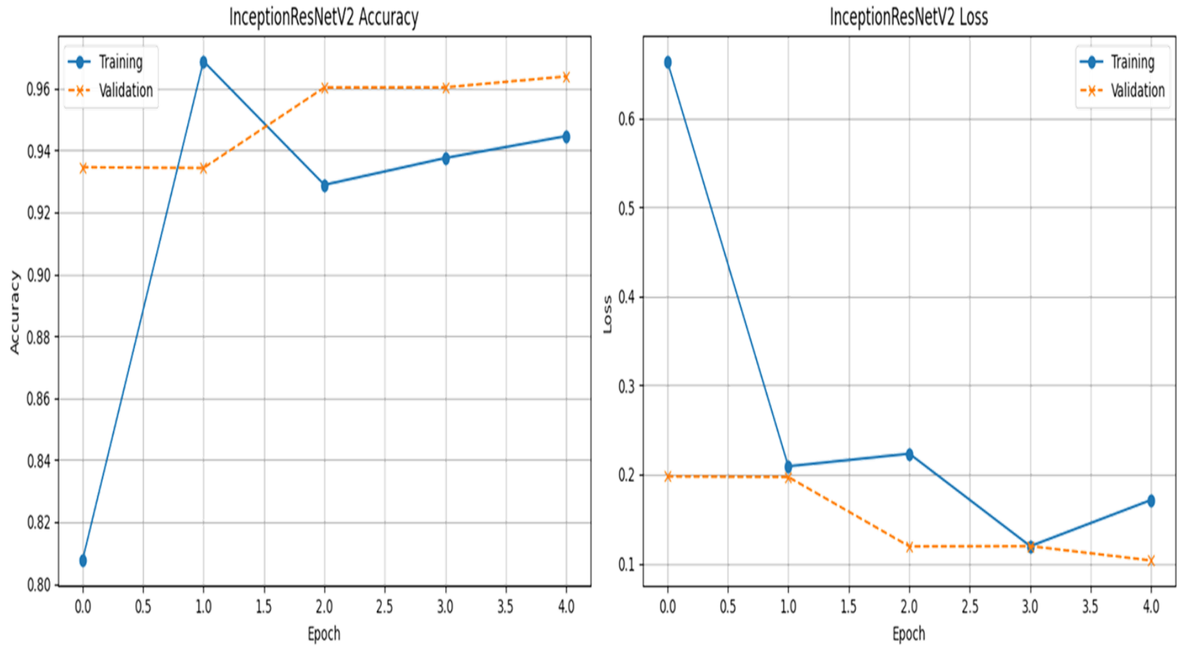

```
Model Results:
                Model  Test Accuracy  Test Loss
0  InceptionResNetV2       0.963863   0.103289
Results saved as 'model_results.csv'
```

Conclusion

Importance

The integration of **AI** into agriculture opens new doors for **early disease detection**, leading to:

- Increased **crop productivity**
- Reduced **economic losses**
- Improved **food security**



Deep learning models like InceptionResNetV2 have the potential to **revolutionize plant health monitoring**, making agriculture more **intelligent**, **scalable**, and **sustainable**.

```
Model Results:
                Model  Test Accuracy  Test Loss
0  InceptionResNetV2       0.963863   0.103289
Results saved as 'model_results.csv'
```

Thank You