```
[notebook]$
[notebook]$ ls
[notebook]$ cd ~/
[notebook]$ ls -ltr
total 1016
drwxr-xr-x 2 root root 4096 Jan 15 21:47 kaggle_cache_dir
drwxr-xr-x 2 root root 4096 Jan 15 21:47 .ipynb_checkpoints
drwxr-xr-x 2 root root 4096 Jan 15 21:47 .jupyter
drwxr-xr-x 2 root root 4096 Jan 15 21:47 .local
drwxr-xr-x 2 root root 4096 Jan 15 21:47 .ssh
[notebook]$

# [cell 1]
%pwd
%cd /kaggle/working/
!git clone https://github.com/Tencent/AutoCodebenchmark.git
!ls -ltr /kaggle/working/AutoCodebenchmark

# [cell 2]
%cd /kaggle/working/AutoCodebenchmark
!ls -ltr
!ls -ltr data/
!ls -ltr data/AutoCodeBenchmark-Lite-1001/
!head -n 1 data/AutoCodeBenchmark-Lite-1001/data.jsonl
!wc -l data/AutoCodeBenchmark-Lite-1001/data.jsonl

# [cell 3]
import os, sys, glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tqdm
import tqdm.notebook as tqdm_nb
import pyarrow, pyarrow.parquet
import sklearn, sklearn.metrics, sklearn.model_selection
import tqdm.auto, rich, rich.console, rich.table
import jsonlines, orjson, ujson
import tiktoken, tiktoken.load, tiktoken.core
import transformers, tokenizers, huggingface_hub
import datasets, datasets.load, datasets.arrow_dataset
import torch, torch.nn, torch.nn.functional, torch.utils.data
import numpy.lib.npyio, numpy.lib.format, numpy.lib.arraypad
import numpy.ma.core, numpy.ma.extras, numpy.ma.mrecords
import numpy.linalg.linalg, numpy.linalg.lapack_lite
import numpy.random.mtrand, numpy.random.bit_generator
import numpy.fft.pocketfft, numpy.fft.helper
import numpy.polynomial.polynomial, numpy.polynomial.chebyshev
import numpy.core.multiarray, numpy.core.umath, numpy.core.shape_base
import numpy.core.fromnumeric, numpy.core.numeric, numpy.core.arrayprint
import numpy.core.defchararray, numpy.core.records, numpy.core._ufunc_config
import numpy.core._dtype, numpy.core._dtype_ctypes, numpy.core._exceptions
import numpy.core._multiarray_umath, numpy.core._multiarray_tests
import numpy.core._rational_tests, numpy.core._pickle_dispatch
import numpy.core._umath_tests, numpy.core._operand_flag_tests
import numpy.core._ufunc_reduceat_tests, numpy.core._rational_tests.test_rational
import numpy.core._struct_ufunc_tests, numpy.core._simd, numpy.core._simd.targets
import numpy.core._simd.baseline, numpy.core._simd.sse, numpy.core._simd.sse2
import numpy.core._simd.sse3, numpy.core._simd.ssse3, numpy.core._simd.sse41
import numpy.core._simd.sse42, numpy.core._simd.avx, numpy.core._simd.avx2
import numpy.core._simd.avx512, numpy.core._simd.avx512.knl, numpy.core._simd.avx512.skx
import numpy.core._simd.avx512.cnl, numpy.core._simd.avx512.icl, numpy.core._simd.avx512.clx
import numpy.core._simd.avx512.cpx, numpy.core._simd.avx512.cpx.bf16, numpy.core._simd.avx512.cpx.vbmi
import numpy.core._simd.avx512.cpx.vbmi2, numpy.core._simd.avx512.cpx.vbmi2.bf16, numpy.core._simd.avx512.cpx.vbmi2.fp16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd, numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw, numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma, numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16.avx512mf16_fp16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16.avx512mf16_fp16.avx512vl_fp16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16.avx512mf16_fp16.avx512vl_fp16.avx512vl_bf16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16.avx512mf16_fp16.avx512vl_fp16.avx512vl_bf16.avx512vl_mf16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16.avx512mf16_fp16.avx512vl_fp16.avx512vl_bf16.avx512vl_mf16.avx512vl_mf16_fp16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16.avx512mf16_fp16.avx512vl_fp16.avx512vl_bf16.avx512vl_mf16.avx512vl_mf16_fp16.avx512vl_bf16_fp16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16.avx512mf16_fp16.avx512vl_fp16.avx512vl_bf16.avx512vl_mf16.avx512vl_mf16_fp16.avx512vl_bf16_fp16.avx512vl_mf16_fp16_fp16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16.avx512mf16_fp16.avx512vl_fp16.avx512vl_bf16.avx512vl_mf16.avx512vl_mf16_fp16.avx512vl_bf16_fp16.avx512vl_mf16_fp16_fp16.avx512vl_bf16_fp16_fp16
import numpy.core._simd.avx512.cpx.vbmi2.fp16.avx512cd.avx512vl.avx512bw.avx512dq.avx512ifma.avx512vbmi.avx512vbmi2.avx512vnni.avx512bitalg.avx512vpopcntdq.avx512vp2intersect.avx512fp16.avx512bf16.avx512mf16.avx512mf16_fp16.avx512vl_fp16.avx512vl_bf16.avx512vl_mf16.avx512vl_mf16_fp16.avx512vl_bf16_fp16.avx512vl_mf16_fp16_fp16.avx512vl_bf16_fp16_fp16.avx512vl_mf16_fp
```

```
# Model 2: DenseNet
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedGroupKFold
from PyTorchTools.ImageLoading.image_dataset import ImageDataset
from sklearn.metrics import *
from PyTorchTools.Models.model_factory import *
from PyTorchTools.Training.trainer_classification import *
from PyTorchTools.Utils.metrics.base_metrics import MetricsPlotter, MetricsPrinter, ConfusionMatrix
from PyTorchTools.Utils.metrics.classification_metrics import ClassificationMetrics, ClassificationMetricsPlotter

# Will Schedule Without Error
torch.set_float32_matmul_precision('high')

image_root = Path(r'Images\224_224\_v1.1mm\*image_root*')

# Load processed dataset
test_df = pd.read_csv(r'Splits\_v1.1mm_by_patient\*dataset*\test_dataset.csv')
test_dataset = ImageDataset(test_df, image_root, 'image_path')

# Create dataloader
test_loader = DataLoader(test_dataset, batch_size=32)
# * Open Device CPU
device = torch.device('cpu')

# Load model
model = torch.load(r'Models\_v1.1mm_by_patient\*dataset*\DenseNet.pth', weights_only=False)

# Create trainer
trainer = ClassificationTrainer(model=model, device=device)

# Get predictions
test_preds = trainer.predict(test_loader)

# Create metrics
metrics = ClassificationMetrics(test_df['cancer'], test_preds[:, 1], threshold=0.5)

# Print metrics
metrics.print()

# Create and save plots
plotter = ClassificationMetricsPlotter(metrics)
plotter.save_plots('Results\_v1.1mm_by_patient\*dataset*\DenseNet')

# Save predictions to CSV
test_df['predicted_cancer'] = (test_preds[:, 1] > 0.5).astype(int)
test_df['prob_cancer'] = test_preds[:, 1]
test_df.to_csv('Results\_v1.1mm_by_patient\*dataset*\DenseNet_predictions.csv', index=False)
```

```
Model: "DenseNet_2"
┏━━━
```

```
class MultiLayerPerceptron(object):
    ''' Multi-layer perceptron class.
        activation: activation function
        derivative: derivative of activation function
        layers: list of integers
        weight_range: tuple of floats
        bias_range: tuple of floats
        learning_rate: float
        momentum: float
        weights: list of numpy arrays
        biases: list of numpy arrays
        deltas: list of numpy arrays
        delta_names: list of strings
        activations: list of numpy arrays
        activation_names: list of strings
        weight_updates: list of numpy arrays
        bias_updates: list of numpy arrays
        error_history: list of floats
        validation_error_history: list of floats
        epoch: int
        stopped_epochs: int
        stopped_error: float
        stopped_weights: list of numpy arrays
        stopped_biases: list of numpy arrays
        training_time: float
        termination_condition: str
        verbose: int
        rng: numpy.random.mtrand.RandomState instance
        validation_fraction: float
        validation_mask: numpy.ndarray
        validation_scores: list of tuples
        ann_score: float
        ann_score_on_error: float
        ann_stopped_score: float
        ann_stopped_score_on_error: float
        ann_training_time: float
        ann_termination_condition: str
        ann_error_history: list of floats
        ann_validation_error_history: list of floats
        ann_epoch: int
        ann_stopped_epochs: int
        ann_stopped_error: float
        ann_stopped_weights: list of numpy arrays
        ann_stopped_biases: list of numpy arrays
        ann_training_time: float
        ann_termination_condition: str
        ann_verbose: int
        ann_rng: numpy.random.mtrand.RandomState instance
        ann_validation_fraction: float
        ann_validation_mask: numpy.ndarray
        ann_validation_scores: list of tuples
        '''
    def __init__(self, layers, weight_range=
                 activation=activations.sigmoid,
                 derivative=derivatives.sigmoid,
                 learning_rate=0.1,
                 momentum=0.0,
                 verbose=0,
                 rng=None,
                 validation_fraction=0.0,
                 ann_score=None,
                 ann_score_on_error=None,
                 ann_stopped_score=None,
                 ann_stopped_score_on_error=None,
                 ann_training_time=None,
                 ann_termination_condition=None,
                 ann_error_history=[],
                 ann_validation_error_history=[],
                 ann_epoch=0,
                 ann_stopped_epochs=0,
                 ann_stopped_error=None,
                 ann_stopped_weights=[],
                 ann_stopped_biases=[],
                 ann_training_time=0.0,
                 ann_termination_condition=None,
                 ann_verbose=0,
                 ann_rng=None,
                 ann_validation_fraction=0.0,
                 ann_validation_mask=None,
                 ann_validation_scores=[]):
        ''' Initialize the network.
            layers: list of integers
            weight_range: tuple of floats
            activation: function
            derivative: function
            learning_rate: float
            momentum: float
            verbose: int
            rng: numpy.random.mtrand.RandomState instance
            validation_fraction: float
            ann_score: float
            ann_score_on_error: float
            ann_stopped_score: float
            ann_stopped_score_on_error: float
            ann_training_time: float
            ann_termination_condition: str
            ann_error_history: list of floats
            ann_validation_error_history: list of floats
            ann_epoch: int
            ann_stopped_epochs: int
            ann_stopped_error: float
            ann_stopped_weights: list of numpy arrays
            ann_stopped_biases: list of numpy arrays
            ann_training_time: float
            ann_termination_condition: str
            ann_verbose: int
            ann_rng: numpy.random.mtrand.RandomState instance
            ann_validation_fraction: float
            ann_validation_mask: numpy.ndarray
            ann_validation_scores: list of tuples
            '''
        self.layers = layers
        self.weight_range = weight_range
        self.activation = activation
        self.derivative = derivative
        self.learning_rate = learning_rate
        self.momentum = momentum
        self.verbose = verbose
        self.rng = rng
        self.validation_fraction = validation_fraction
        self.ann_score = ann_score
        self.ann_score_on_error = ann_score_on_error
        self.ann_stopped_score = ann_stopped_score
        self.ann_stopped_score_on_error = ann_stopped_score_on_error
        self.ann_training_time = ann_training_time
        self.ann_termination_condition = ann_termination_condition
        self.ann_error_history = ann_error_history
        self.ann_validation_error_history = ann_validation_error_history
        self.ann_epoch = ann_epoch
        self.ann_stopped_epochs = ann_stopped_epochs
        self.ann_stopped_error = ann_stopped_error
        self.ann_stopped_weights = ann_stopped_weights
        self.ann_stopped_biases = ann_stopped_biases
        self.ann_training_time = ann_training_time
        self.ann_termination_condition = ann_termination_condition
        self.ann_verbose = ann_verbose
        self.ann_rng = ann_rng
        self.ann_validation_fraction = ann_validation_fraction
        self.ann_validation_mask = ann_validation_mask
        self.ann_validation_scores = ann_validation_scores
        self.weights = []
        self.biases = []
        self.deltas = []
        self.delta_names = []
        self.activations = []
        self.activation_names = []
        self.weight_updates = []
        self.bias_updates = []
        self.error_history = []
        self.validation_error_history = []
        self.epoch = 0
        self.stopped_epochs = 0
        self.stopped_error = None
        self.stopped_weights = []
        self.stopped_biases = []
        self.training_time = 0.0
        self.termination_condition = None
        self.verbose = verbose
        self.rng = rng
        self.validation_fraction = validation_fraction
        self.validation_mask = None
        self.validation_scores = []
        self.ann_score = None
        self.ann_score_on_error = None
        self.ann_stopped_score = None
        self.ann_stopped_score_on_error = None
        self.ann_training_time = None
        self.ann_termination_condition = None
        self.ann_error_history = []
        self.ann_validation_error_history = []
        self.ann_epoch = 0
        self.ann_stopped_epochs = 0
        self.ann_stopped_error = None
        self.ann_stopped_weights = []
        self.ann_stopped_biases = []
        self.ann_training_time = 0.0
        self.ann_termination_condition = None
```

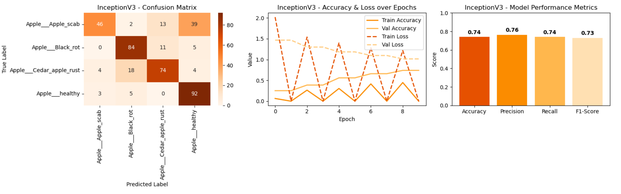
concatenate_2 (Concatenate)	(None, 5, 5, 768)	0	activation_82[0]… activation_83[0]…
activation_84 (Activation)	(None, 5, 5, 192)	0	batch_normalizat…
mixed9 (Concatenate)	(None, 5, 5, 2048)	0	activation_76[0]… mixed9_0[0][0], concatenate_2[0]… activation_84[0]…
conv2d_89 (Conv2D)	(None, 5, 5, 448)	917,504	mixed9[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 448)	1,344	conv2d_89[0][0]
activation_89 (Activation)	(None, 5, 5, 448)	0	batch_normalizat…
conv2d_86 (Conv2D)	(None, 5, 5, 384)	786,432	mixed9[0][0]
conv2d_90 (Conv2D)	(None, 5, 5, 384)	1,548,288	activation_89[0]…
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 384)	1,152	conv2d_86[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 384)	1,152	conv2d_90[0][0]
activation_86 (Activation)	(None, 5, 5, 384)	0	batch_normalizat…
activation_90 (Activation)	(None, 5, 5, 384)	0	batch_normalizat…
conv2d_87 (Conv2D)	(None, 5, 5, 384)	442,368	activation_86[0]…
conv2d_88 (Conv2D)	(None, 5, 5, 384)	442,368	activation_86[0]…
conv2d_91 (Conv2D)	(None, 5, 5, 384)	442,368	activation_90[0]…
conv2d_92 (Conv2D)	(None, 5, 5, 384)	442,368	activation_90[0]…
average_pooling2d_8 (AveragePooling2D)	(None, 5, 5, 2048)	0	mixed9[0][0]
conv2d_85 (Conv2D)	(None, 5, 5, 320)	655,360	mixed9[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 384)	1,152	conv2d_87[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 384)	1,152	conv2d_88[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 384)	1,152	conv2d_91[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 384)	1,152	conv2d_92[0][0]
conv2d_93 (Conv2D)	(None, 5, 5, 192)	393,216	average_pooling2…
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 320)	960	conv2d_85[0][0]
activation_87 (Activation)	(None, 5, 5, 384)	0	batch_normalizat…
activation_88 (Activation)	(None, 5, 5, 384)	0	batch_normalizat…
activation_91 (Activation)	(None, 5, 5, 384)	0	batch_normalizat…
activation_92 (Activation)	(None, 5, 5, 384)	0	batch_normalizat…
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 192)	576	conv2d_93[0][0]
activation_85 (Activation)	(None, 5, 5, 320)	0	batch_normalizat…
mixed9_1 (Concatenate)	(None, 5, 5, 768)	0	activation_87[0]… activation_88[0]…
concatenate_3 (Concatenate)	(None, 5, 5, 768)	0	activation_91[0]… activation_92[0]…
activation_93 (Activation)	(None, 5, 5, 192)	0	batch_normalizat…
mixed10 (Concatenate)	(None, 5, 5, 2048)	0	activation_85[0]… mixed9_1[0][0], concatenate_3[0]… activation_93[0]…
global_average_pooo… (GlobalAveragePool…	(None, 2048)	0	mixed10[0][0]
weather_input (InputLayer)	(None, 7)	0	–
dense_10 (Dense)	(None, 128)	262,272	global_average_p…
dense_11 (Dense)	(None, 32)	256	weather_input[0]…
dropout_4 (Dropout)	(None, 128)	0	dense_10[0][0]
dense_12 (Dense)	(None, 16)	528	dense_11[0][0]
concatenate_4 (Concatenate)	(None, 144)	0	dropout_4[0][0], dense_12[0][0]
dense_13 (Dense)	(None, 64)	9,280	concatenate_4[0]…
dropout_5 (Dropout)	(None, 64)	0	dense_13[0][0]
dense_14 (Dense)	(None, 4)	260	dropout_5[0][0]

Total params: 22,075,380 (84.21 MB)
Trainable params: 272,596 (1.04 MB)
Non-trainable params: 21,802,784 (83.17 MB)

```
Epoch 1/10
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 878ms/step - accuracy: 0.0429 - loss: 2.1564 - val_precision: 0.2149 - val_recall: 0.2525 - val_f1: 0.1740
13/13 ━━━━━━━━━━━━━━━━━━━━ 54s 3s/step - accuracy: 0.0650 - loss: 2.0186 - val_accuracy: 0.2525 - val_loss: 1.4653 - learning_rate: 1.0000e-04 - val_precision: 0.2149 -
val_recall: 0.2525 - val_f1: 0.1740
Epoch 2/10
- val_precision: 0.2149 - val_recall: 0.2525 - val_f1: 0.1740
13/13 ━━━━━━━━━━━━━━━━━━━━ 24s 2s/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.2525 - val_loss: 1.4653 - learning_rate: 1.0000e-04 - val_precision:
0.2149 - val_recall: 0.2525 - val_f1: 0.1740
Epoch 3/10
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 846ms/step - accuracy: 0.3045 - loss: 1.4653 - val_precision: 0.3814 - val_recall: 0.3875 - val_f1: 0.3419
13/13 ━━━━━━━━━━━━━━━━━━━━ 35s 3s/step - accuracy: 0.2658 - loss: 1.5393 - val_accuracy: 0.3875 - val_loss: 1.2993 - learning_rate: 1.0000e-04 - val_precision: 0.3814 -
val_recall: 0.3875 - val_f1: 0.3419
Epoch 4/10
- val_precision: 0.3814 - val_recall: 0.3875 - val_f1: 0.3419
13/13 ━━━━━━━━━━━━━━━━━━━━ 23s 2s/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.3875 - val_loss: 1.2993 - learning_rate: 1.0000e-04 - val_precision:
0.3814 - val_recall: 0.3875 - val_f1: 0.3419
Epoch 5/10
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 869ms/step - accuracy: 0.3049 - loss: 1.3943 - val_precision: 0.5613 - val_recall: 0.5600 - val_f1: 0.5439
13/13 ━━━━━━━━━━━━━━━━━━━━ 35s 3s/step - accuracy: 0.3050 - loss: 1.3982 - val_accuracy: 0.5600 - val_loss: 1.1836 - learning_rate: 1.0000e-04 - val_precision: 0.5613 -
val_recall: 0.5600 - val_f1: 0.5439
Epoch 6/10
- val_precision: 0.5613 - val_recall: 0.5600 - val_f1: 0.5439
13/13 ━━━━━━━━━━━━━━━━━━━━ 23s 2s/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.5600 - val_loss: 1.1836 - learning_rate: 1.0000e-04 - val_precision:
0.5613 - val_recall: 0.5600 - val_f1: 0.5439
Epoch 7/10
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 861ms/step - accuracy: 0.3331 - loss: 1.3657 - val_precision: 0.7035 - val_recall: 0.6600 - val_f1: 0.6391
13/13 ━━━━━━━━━━━━━━━━━━━━ 36s 3s/step - accuracy: 0.4150 - loss: 1.2796 - val_accuracy: 0.6600 - val_loss: 1.0970 - learning_rate: 1.0000e-04 - val_precision: 0.7035 -
val_recall: 0.6600 - val_f1: 0.6391
Epoch 8/10
- val_precision: 0.7035 - val_recall: 0.6600 - val_f1: 0.6391
13/13 ━━━━━━━━━━━━━━━━━━━━ 23s 2s/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.6600 - val_loss: 1.0970 - learning_rate: 1.0000e-04 - val_precision:
0.7035 - val_recall: 0.6600 - val_f1: 0.6391
Epoch 9/10
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 865ms/step - accuracy: 0.3577 - loss: 1.3023 - val_precision: 0.7627 - val_recall: 0.7400 - val_f1: 0.7298
13/13 ━━━━━━━━━━━━━━━━━━━━ 36s 3s/step - accuracy: 0.4475 - loss: 1.2162 - val_accuracy: 0.7400 - val_loss: 1.0151 - learning_rate: 1.0000e-04 - val_precision: 0.7627 -
val_recall: 0.7400 - val_f1: 0.7298
Epoch 10/10
- val_precision: 0.7627 - val_recall: 0.7400 - val_f1: 0.7298
13/13 ━━━━━━━━━━━━━━━━━━━━ 24s 2s/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.7400 - val_loss: 1.0151 - learning_rate: 1.0000e-04 - val_precision:
0.7627 - val_recall: 0.7400 - val_f1: 0.7298

Test Accuracy: 0.7400
Test Precision: 0.7627
Test Recall: 0.7400
Test F1-Score: 0.7298

Classification Report:
                          precision    recall  f1-score   support

      Apple___Apple_scab       0.87      0.46      0.60       100
   Apple___Black_rot       0.77      0.84      0.80       100
Apple___Cedar_apple_rust       0.76      0.74      0.75       100
    Apple___healthy       0.66      0.92      0.77       100

             accuracy                           0.74       400
            macro avg       0.76      0.74      0.73       400
         weighted avg       0.76      0.74      0.73       400
```

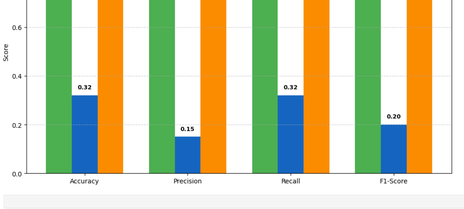
In [15]:

```
# Visualizations for InceptionV3 Multimodal Model

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
import numpy as np

# 1⃣ Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
class_names = list(test_gen.class_indices.keys())

plt.figure(figsize=(16,5))

# ==== Confusion Matrix ====
plt.subplot(1,3,1)
sns.heatmap(cm, annot=True, fmt="d", cmap="Oranges",
            xticklabels=class_names, yticklabels=class_names)
plt.title("InceptionV3 - Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")

# 2⃣ Accuracy & Loss Curves
plt.subplot(1,3,2)
plt.plot(history.history["accuracy"], label="Train Accuracy", linewidth=2, color="#FBBC00")
plt.plot(history.history["val_accuracy"], label="Val Accuracy", linewidth=2, color="#FFB74D")
plt.plot(history.history["loss"], label="Train Loss", linestyle="--", linewidth=2, color="#E65100")
plt.plot(history.history["val_loss"], label="Val Loss", linestyle="--", linewidth=2, color="#FFCC80")
plt.legend()
plt.title("InceptionV3 - Accuracy & Loss over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Value")

# 3⃣ Performance Metrics Bar Chart
metrics = ["Accuracy", "Precision", "Recall", "F1-Score"]
values = [acc, prec, rec, f1]

plt.subplot(1,3,3)
bars = plt.bar(metrics, values, color=["#E65100", "#FBBC00", "#FFB74D", "#FFE082"])
plt.ylim(0, 1)
plt.title("InceptionV3 - Model Performance Metrics")
plt.ylabel("Score")

for bar, val in zip(bars, values):
    plt.text(bar.get_x() + bar.get_width()/2, val + 0.02, f"{val:.2f}",
             ha='center', va='bottom', fontsize=10, fontweight='bold')

plt.tight_layout()
plt.show()
```



In [17]:

```
# Final Comparison Between All Models

import matplotlib.pyplot as plt
import numpy as np

# Insert your final metrics for each model

mobilenet_metrics = [0.7825, 0.7858, 0.7825, 0.7805]
resnet_metrics    = [0.3200, 0.1502, 0.3200, 0.2008]
inception_metrics = [0.7400, 0.7627, 0.7400, 0.7298]

metrics_names = ["Accuracy", "Precision", "Recall", "F1-Score"]

# Create grouped bar chart
x = np.arange(len(metrics_names))
width = 0.25

plt.figure(figsize=(10,6))
plt.bar(x - width, mobilenet_metrics, width, label="MobileNetV2", color="#4CAF50")
plt.bar(x, resnet_metrics, width, label="ResNet50", color="#1565C0")
plt.bar(x + width, inception_metrics, width, label="InceptionV3", color="#FBBC00")

# Customize chart
plt.xticks(x, metrics_names)
plt.ylim(0, 1)
plt.ylabel("Score")
plt.title("📊 Model Performance Comparison (Apple Disease Detection)")
plt.legend()
plt.grid(axis="y", linestyle="--", alpha=0.6)

# Annotate bars with values
for i, values in enumerate([mobilenet_metrics, resnet_metrics, inception_metrics]):
    for j, val in enumerate(values):
        plt.text(j + (i-1)*width, val + 0.02, f"{val:.2f}",
                 ha='center', va='bottom', fontsize=9, fontweight='bold')

plt.tight_layout()
plt.show()
```



In [ ]: