True: indian market
Pred: indian market

True: indian market
Pred: indian market

True: onion
Pred: potato

True: onion
Pred: indian market

True: potato
Pred: tomato

True: potato
Pred: onion

True: tomato
Pred: onion

True: tomato
Pred: tomato

6.3.6 Save the model

[98]:
```
model.save("saved_models/1_pretrained_resnet50_trainable0_model.keras")
```

6.4 DATA AUGMENTED RESNET50

6.4.1 Load the previously trained model architecture and weights

[99]:
```
# Load the entire model (architecture + weights)
model = keras.models.load_model("saved_models/
 ↪1_pretrained_resnet50_trainable0_model.keras")
```

[100]:
```
model.summary()
```

Model: "pretrained_resnet50"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 8, 8, 2048)	23,587,712
global_average_pooling2d_6 (GlobalAveragePooling2D)	(None, 2048)	0
dense_18 (Dense)	(None, 512)	1,049,088
batch_normalization_12 (BatchNormalization)	(None, 512)	2,048
dropout_6 (Dropout)	(None, 512)	0
dense_19 (Dense)	(None, 256)	131,328
batch_normalization_13 (BatchNormalization)	(None, 256)	1,024
dense_20 (Dense)	(None, 4)	1,028

Total params: 27,138,190 (103.52 MB)

Trainable params: 1,182,980 (4.51 MB)

Non-trainable params: 23,589,248 (89.99 MB)

```
Optimizer params: 2,365,962 (9.03 MB)
```

6.4.2 Model Compilation and Training

[101]:
```
ckpt_path = "checkpoints/1_pretrained_resnet50_aug_trainable0.weights.h5"
```

[102]:
```
log_dir = 'logs/1_pretrained_resnet50_aug_trainable0'
```

[103]:
```
run_name = '1_pretrained_resnet50_aug_trainable0'
```

[104]:
```
model_fit = compile_train(model, train_aug_ds, val_aug_ds,␣
↪log_dir=log_dir,epochs=20,ckpt_path=ckpt_path)
```

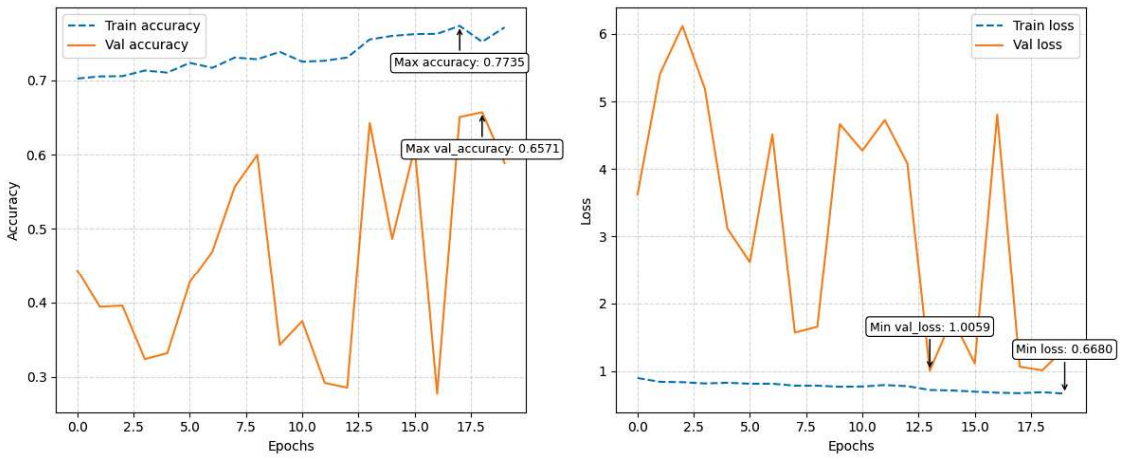
```
Epoch 1/20
79/79             138s 2s/step -
accuracy: 0.7046 - loss: 0.9642 - precision_4: 0.7288 - recall_4: 0.6683 -
val_accuracy: 0.4434 - val_loss: 3.6197 - val_precision_4: 0.4457 -
val_recall_4: 0.4386 - learning_rate: 0.0010
Epoch 2/20
79/79             129s 2s/step -
accuracy: 0.7112 - loss: 0.8268 - precision_4: 0.7432 - recall_4: 0.6651 -
val_accuracy: 0.3939 - val_loss: 5.4068 - val_precision_4: 0.3939 -
val_recall_4: 0.3939 - learning_rate: 0.0010
Epoch 3/20
79/79             128s 2s/step -
accuracy: 0.6950 - loss: 0.8709 - precision_4: 0.7396 - recall_4: 0.6384 -
val_accuracy: 0.3955 - val_loss: 6.1206 - val_precision_4: 0.3971 -
val_recall_4: 0.3939 - learning_rate: 0.0010
Epoch 4/20
79/79             133s 2s/step -
accuracy: 0.6953 - loss: 0.8382 - precision_4: 0.7365 - recall_4: 0.6324 -
val_accuracy: 0.3238 - val_loss: 5.1845 - val_precision_4: 0.3237 -
val_recall_4: 0.3222 - learning_rate: 0.0010
Epoch 5/20
79/79             131s 2s/step -
accuracy: 0.7122 - loss: 0.8262 - precision_4: 0.7551 - recall_4: 0.6668 -
val_accuracy: 0.3317 - val_loss: 3.1185 - val_precision_4: 0.3350 -
val_recall_4: 0.3270 - learning_rate: 0.0010
Epoch 6/20
79/79             131s 2s/step -
accuracy: 0.7328 - loss: 0.7921 - precision_4: 0.7695 - recall_4: 0.6700 -
val_accuracy: 0.4274 - val_loss: 2.6197 - val_precision_4: 0.4272 -
val_recall_4: 0.4163 - learning_rate: 0.0010
Epoch 7/20
79/79             131s 2s/step -
accuracy: 0.7107 - loss: 0.8195 - precision_4: 0.7416 - recall_4: 0.6646 -
val_accuracy: 0.4689 - val_loss: 4.5126 - val_precision_4: 0.4689 -
```

```
val_recall_4: 0.4689 - learning_rate: 0.0010
Epoch 8/20
79/79            130s 2s/step -
accuracy: 0.7222 - loss: 0.8058 - precision_4: 0.7617 - recall_4: 0.6734 -
val_accuracy: 0.5566 - val_loss: 1.5669 - val_precision_4: 0.5590 -
val_recall_4: 0.5439 - learning_rate: 0.0010
Epoch 9/20
79/79            128s 2s/step -
accuracy: 0.7359 - loss: 0.7748 - precision_4: 0.7727 - recall_4: 0.6799 -
val_accuracy: 0.5997 - val_loss: 1.6530 - val_precision_4: 0.6050 -
val_recall_4: 0.5837 - learning_rate: 0.0010
Epoch 10/20
79/79            165s 2s/step -
accuracy: 0.7360 - loss: 0.7911 - precision_4: 0.7706 - recall_4: 0.6818 -
val_accuracy: 0.3429 - val_loss: 4.6655 - val_precision_4: 0.3435 -
val_recall_4: 0.3429 - learning_rate: 0.0010
Epoch 11/20
79/79            185s 2s/step -
accuracy: 0.7221 - loss: 0.7943 - precision_4: 0.7686 - recall_4: 0.6722 -
val_accuracy: 0.3748 - val_loss: 4.2717 - val_precision_4: 0.3728 -
val_recall_4: 0.3668 - learning_rate: 0.0010
Epoch 12/20
79/79            183s 2s/step -
accuracy: 0.7111 - loss: 0.8148 - precision_4: 0.7534 - recall_4: 0.6591 -
val_accuracy: 0.2919 - val_loss: 4.7266 - val_precision_4: 0.2919 -
val_recall_4: 0.2919 - learning_rate: 0.0010
Epoch 13/20
79/79            188s 2s/step -
accuracy: 0.7177 - loss: 0.7980 - precision_4: 0.7624 - recall_4: 0.6699 -
val_accuracy: 0.2855 - val_loss: 4.0851 - val_precision_4: 0.2859 -
val_recall_4: 0.2855 - learning_rate: 0.0010
Epoch 14/20
79/79            187s 2s/step -
accuracy: 0.7376 - loss: 0.7565 - precision_4: 0.7827 - recall_4: 0.6839 -
val_accuracy: 0.6427 - val_loss: 1.0059 - val_precision_4: 0.6667 -
val_recall_4: 0.6061 - learning_rate: 3.0000e-04
Epoch 15/20
79/79            187s 2s/step -
accuracy: 0.7504 - loss: 0.7256 - precision_4: 0.7835 - recall_4: 0.7107 -
val_accuracy: 0.4864 - val_loss: 1.7771 - val_precision_4: 0.4934 -
val_recall_4: 0.4801 - learning_rate: 3.0000e-04
Epoch 16/20
79/79            184s 2s/step -
accuracy: 0.7562 - loss: 0.7041 - precision_4: 0.7937 - recall_4: 0.7085 -
val_accuracy: 0.6093 - val_loss: 1.1126 - val_precision_4: 0.6241 -
val_recall_4: 0.5694 - learning_rate: 3.0000e-04
Epoch 17/20
79/79            182s 2s/step -
```

```
accuracy: 0.7558 - loss: 0.7005 - precision_4: 0.7862 - recall_4: 0.7105 -
val_accuracy: 0.2775 - val_loss: 4.8054 - val_precision_4: 0.2775 -
val_recall_4: 0.2775 - learning_rate: 3.0000e-04
Epoch 18/20
79/79              183s 2s/step -
accuracy: 0.7733 - loss: 0.6748 - precision_4: 0.8140 - recall_4: 0.7374 -
val_accuracy: 0.6507 - val_loss: 1.0644 - val_precision_4: 0.6644 -
val_recall_4: 0.6252 - learning_rate: 3.0000e-04
Epoch 19/20
79/79              184s 2s/step -
accuracy: 0.7408 - loss: 0.7053 - precision_4: 0.7774 - recall_4: 0.7087 -
val_accuracy: 0.6571 - val_loss: 1.0117 - val_precision_4: 0.6785 -
val_recall_4: 0.6427 - learning_rate: 3.0000e-04
Epoch 20/20
79/79              183s 2s/step -
accuracy: 0.7528 - loss: 0.7177 - precision_4: 0.7832 - recall_4: 0.7126 -
val_accuracy: 0.5885 - val_loss: 1.3327 - val_precision_4: 0.5940 -
val_recall_4: 0.5694 - learning_rate: 9.0000e-05
```

6.4.3 Plot loss & accuracy for training and validation wrt epoch

[105]:
```
plot_loss_and_accuracy(["accuracy","loss"],model_fit)
```



6.4.4 Tensorboard

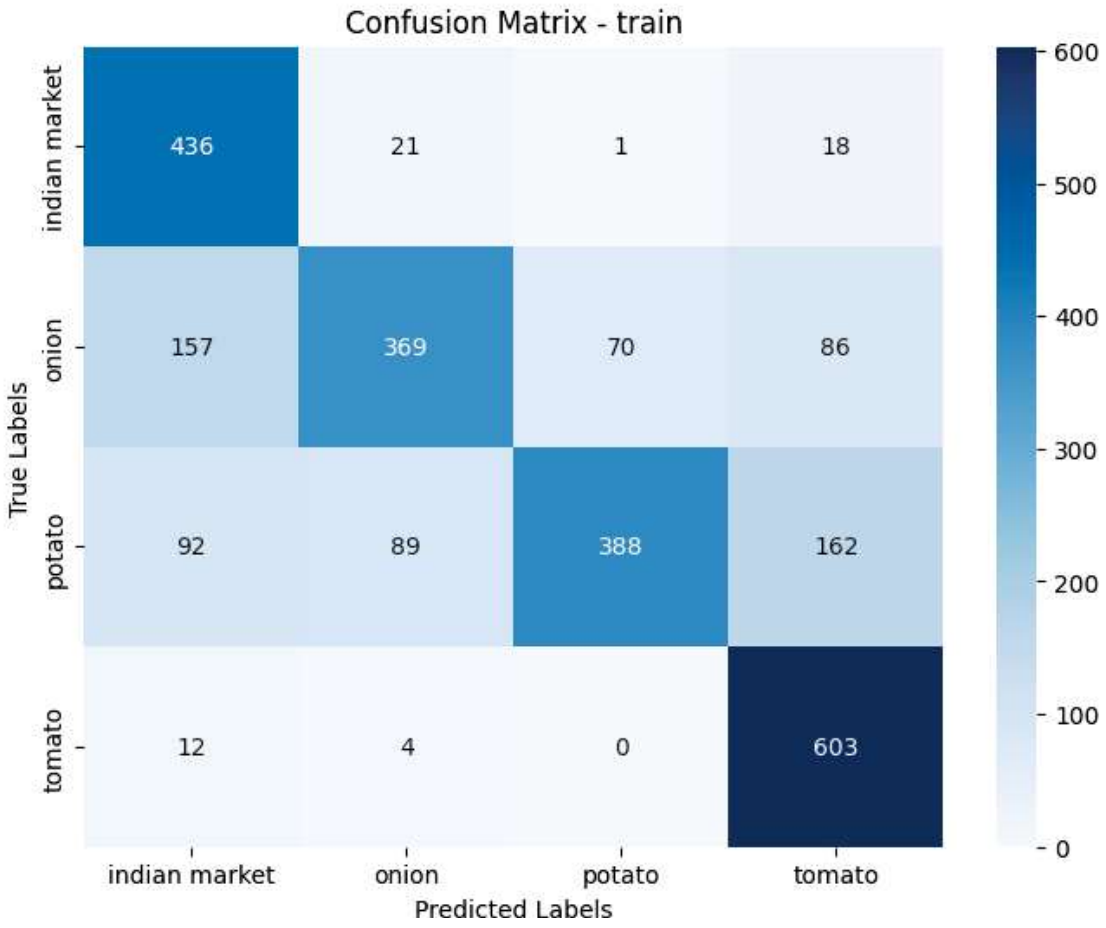
[106]:
```
%load_ext tensorboard
```

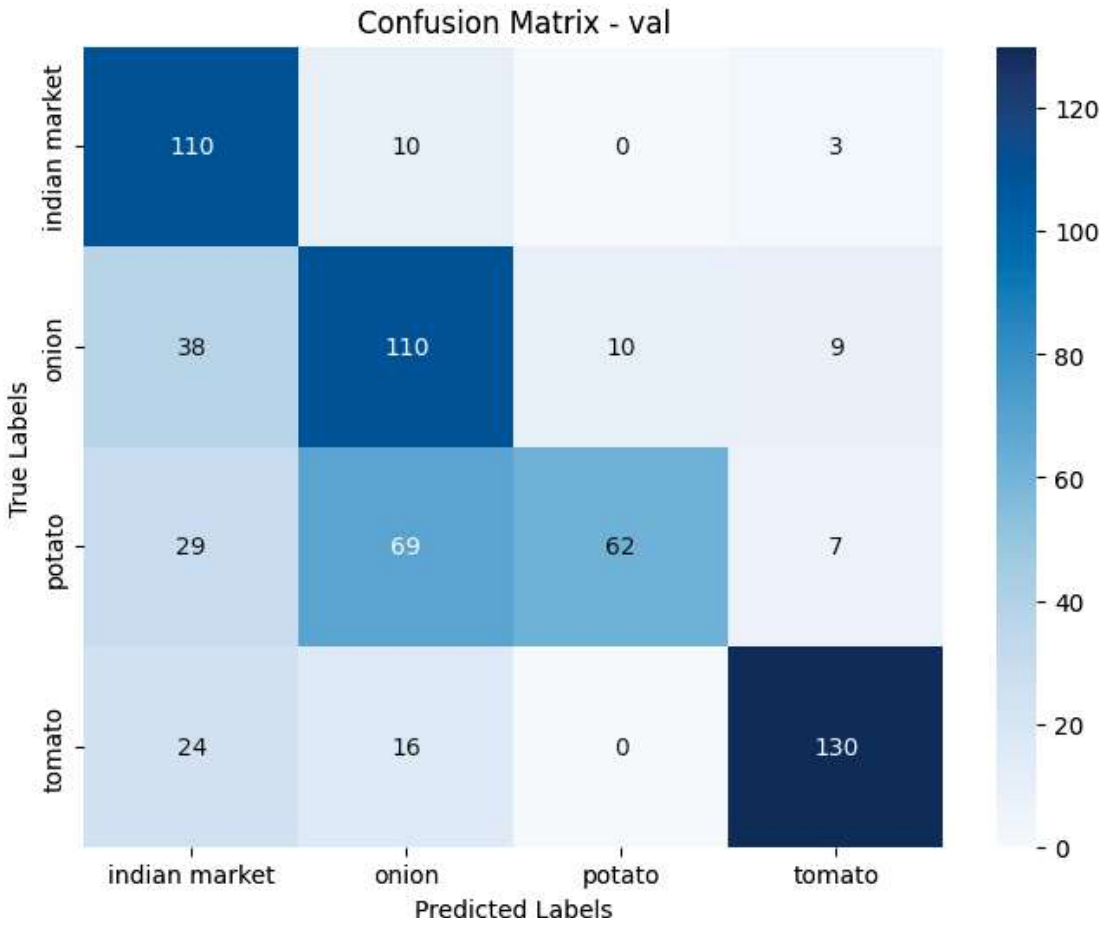
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

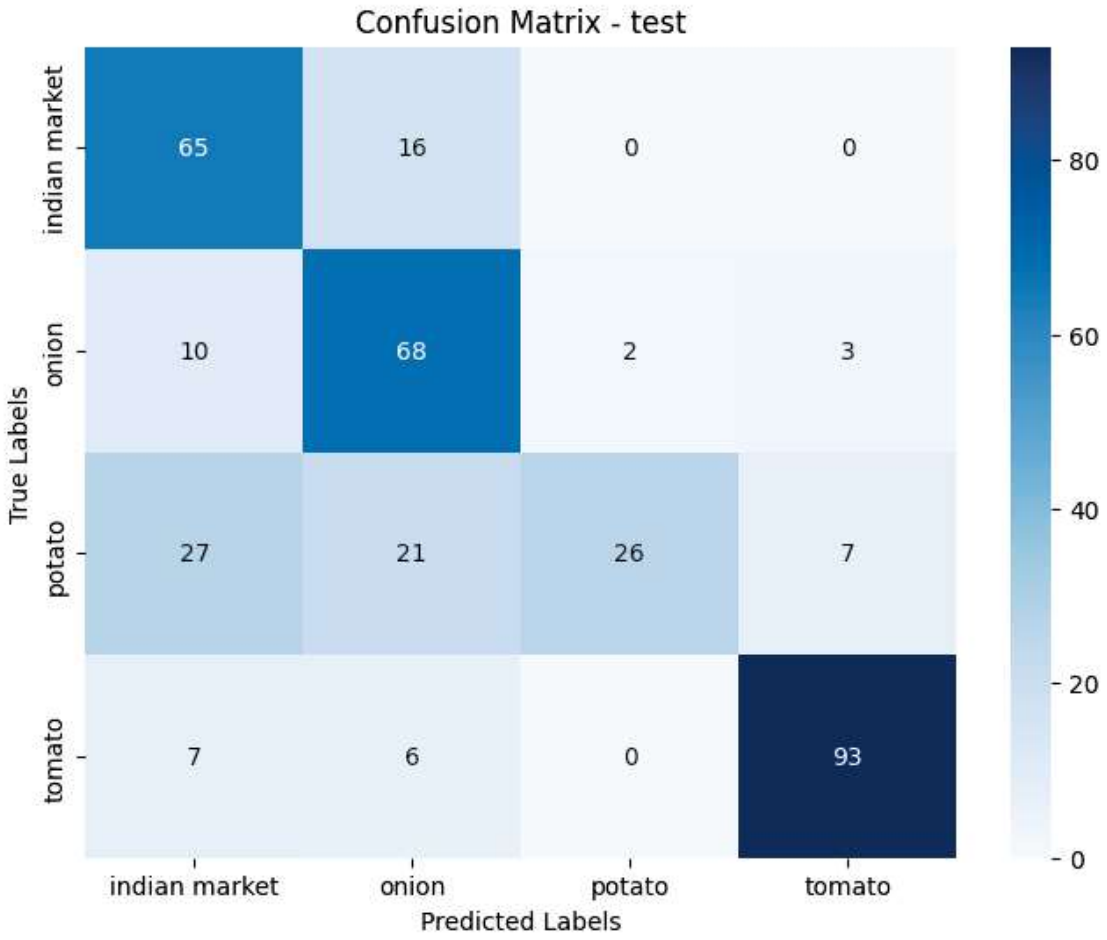
6.4.5 Model Evaluation, Mlflow Logging and Printing Metrics

[107]:
```
evaluate_model(model, train_aug_ds, val_aug_ds, test_aug_ds,␣
↪class_names,run_name=run_name,ckpt_path= ckpt_path)
```

```
2025/02/10 22:55:08 WARNING mlflow.tensorflow: You are saving a TensorFlow Core
model or Keras model without a signature. Inference with
mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation
accepts pandas DataFrames as inference inputs.
2025/02/10 22:55:34 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.


train Metrics:
 Accuracy: 71.61%
 Precision: 73.22%
 Recall: 74.05%

val Metrics:
 Accuracy: 65.71%
 Precision: 70.44%
 Recall: 67.22%

test Metrics:
 Accuracy: 71.79%
 Precision: 76.01%
 Recall: 70.50%
```


Confusion Matrix - train
True Labels
indian market
onion
potato
tomato
436
21
1
18
157
369
70
86
92
89
388
162
12
4
0
603
indian market
onion
potato
tomato
Predicted Labels
600
500
400
300
200
100
0


Confusion Matrix - val
True Labels
Predicted Labels
indian market
onion
potato
tomato
110
10
0
3
38
110
10
9
29
69
62
7
24
16
0
130
120
100
80
60
40
20
0




```
Classification Report - train
              precision    recall  f1-score   support

indian market       0.63      0.92      0.74       476
        onion       0.76      0.54      0.63       682
       potato       0.85      0.53      0.65       731
       tomato       0.69      0.97      0.81       619

     accuracy                           0.72      2508
    macro avg       0.73      0.74      0.71      2508
 weighted avg       0.74      0.72      0.70      2508


Classification Report - val
              precision    recall  f1-score   support

indian market       0.55      0.89      0.68       123
```

```
        onion       0.54      0.66      0.59       167
       potato       0.86      0.37      0.52       167
       tomato       0.87      0.76      0.82       170

     accuracy                           0.66       627
    macro avg       0.70      0.67      0.65       627
 weighted avg       0.72      0.66      0.65       627


Classification Report - test
               precision    recall  f1-score   support

indian market       0.60      0.80      0.68        81
        onion       0.61      0.82      0.70        83
       potato       0.93      0.32      0.48        81
       tomato       0.90      0.88      0.89       106

     accuracy                           0.72       351
    macro avg       0.76      0.71      0.69       351
 weighted avg       0.77      0.72      0.70       351


Class-wise Accuracy:

Train Class-wise Accuracy:
 indian market: 91.60% (436/476)
 onion: 54.11% (369/682)
 potato: 53.08% (388/731)
 tomato: 97.42% (603/619)

Val Class-wise Accuracy:
 indian market: 89.43% (110/123)
 onion: 65.87% (110/167)
 potato: 37.13% (62/167)
 tomato: 76.47% (130/170)

Test Class-wise Accuracy:
 indian market: 80.25% (65/81)
 onion: 81.93% (68/83)
 potato: 32.10% (26/81)
 tomato: 87.74% (93/106)

Random Test Predictions:
```

True: indian market
Pred: indian market

True: indian market
Pred: indian market

True: onion
Pred: onion

True: onion
Pred: onion

True: potato
Pred: indian market

True: potato
Pred: indian market

True: tomato
Pred: tomato

True: tomato
Pred: tomato

6.4.6 Save the model

[108]:
```
model.save("saved_models/1_pretrained_resnet50_aug_trainable0_model.keras")
```

6.5 INCEPTIONV3

6.5.1 Building the model

[144]:
```
def pretrained_inceptionv3(height=256, width=256, num_classes=4,␣
↪trainable_layers=0, ckpt_path=None):
    base_model = applications.InceptionV3(weights="imagenet",␣
↪include_top=False, input_shape=(height, width, 3))

    # Freeze all layers initially
    base_model.trainable = False

    # If fine-tuning, unfreeze last `trainable_layers`
    if trainable_layers > 0:
        for layer in base_model.layers[-trainable_layers:]:
            layer.trainable = True

    model = keras.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(512, activation="relu", kernel_regularizer=regularizers.
↪l2(5e-4)),
        layers.BatchNormalization(),
        layers.Dropout(0.3),

        layers.Dense(256, activation="relu", kernel_regularizer=regularizers.
↪l2(5e-4)),
        layers.BatchNormalization(),

        layers.Dense(num_classes, activation="softmax")
    ], name="pretrained_inceptionv3")

    # Load weights only if a valid checkpoint is provided
    if ckpt_path and os.path.exists(ckpt_path):
        print(f"Loading weights from {ckpt_path}")
        model.load_weights(ckpt_path)

    return model
```

[145]:
```
ckpt_path = "checkpoints/1_pretrained_inceptionv3_trainable0.weights.h5"
log_dir = "logs/1_pretrained_inceptionv3_trainable0"

# Load pretrained VGG16 model
```

```
model = pretrained_inceptionv3(height=256, width=256, num_classes=4,␣
↪trainable_layers=0,ckpt_path=ckpt_path)
```

[146]:
```
model.summary()
```

Model: "pretrained_inceptionv3"

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 6, 6, 2048)	21,802,784
global_average_pooling2d_12 (GlobalAveragePooling2D)	(None, 2048)	0
dense_36 (Dense)	(None, 512)	1,049,088
batch_normalization_110 (BatchNormalization)	(None, 512)	2,048
dropout_14 (Dropout)	(None, 512)	0
dense_37 (Dense)	(None, 256)	131,328
batch_normalization_111 (BatchNormalization)	(None, 256)	1,024
dense_38 (Dense)	(None, 4)	1,028

Total params: 22,987,300 (87.69 MB)

Trainable params: 1,182,980 (4.51 MB)

Non-trainable params: 21,804,320 (83.18 MB)

6.5.2 Model Compilation and Training

[147]:
```
model_fit = compile_train(model, train_ds, val_ds,␣
↪log_dir=log_dir,epochs=50,ckpt_path=ckpt_path)
```

```
Epoch 1/50
79/79              88s 1s/step -
accuracy: 0.8402 - loss: 1.0339 - precision_11: 0.8697 - recall_11: 0.8139 -
val_accuracy: 0.9346 - val_loss: 0.7284 - val_precision_11: 0.9463 -
```

```
val_recall_11: 0.9282 - learning_rate: 0.0010
Epoch 2/50
79/79            112s 1s/step -
accuracy: 0.9688 - loss: 0.6379 - precision_11: 0.9717 - recall_11: 0.9667 -
val_accuracy: 0.9458 - val_loss: 0.6981 - val_precision_11: 0.9485 -
val_recall_11: 0.9394 - learning_rate: 0.0010
Epoch 3/50
79/79            115s 1s/step -
accuracy: 0.9803 - loss: 0.5795 - precision_11: 0.9807 - recall_11: 0.9787 -
val_accuracy: 0.9346 - val_loss: 0.7433 - val_precision_11: 0.9374 -
val_recall_11: 0.9314 - learning_rate: 0.0010
Epoch 4/50
79/79            116s 1s/step -
accuracy: 0.9778 - loss: 0.5403 - precision_11: 0.9781 - recall_11: 0.9769 -
val_accuracy: 0.9442 - val_loss: 0.7112 - val_precision_11: 0.9456 -
val_recall_11: 0.9426 - learning_rate: 0.0010
Epoch 5/50
79/79            116s 1s/step -
accuracy: 0.9834 - loss: 0.5035 - precision_11: 0.9837 - recall_11: 0.9825 -
val_accuracy: 0.9601 - val_loss: 0.5410 - val_precision_11: 0.9601 -
val_recall_11: 0.9585 - learning_rate: 0.0010
Epoch 6/50
79/79            116s 1s/step -
accuracy: 0.9847 - loss: 0.4618 - precision_11: 0.9848 - recall_11: 0.9844 -
val_accuracy: 0.9314 - val_loss: 0.6270 - val_precision_11: 0.9313 -
val_recall_11: 0.9298 - learning_rate: 0.0010
Epoch 7/50
79/79            117s 1s/step -
accuracy: 0.9924 - loss: 0.4155 - precision_11: 0.9924 - recall_11: 0.9924 -
val_accuracy: 0.9474 - val_loss: 0.5656 - val_precision_11: 0.9489 -
val_recall_11: 0.9474 - learning_rate: 0.0010
Epoch 8/50
79/79            117s 1s/step -
accuracy: 0.9931 - loss: 0.3736 - precision_11: 0.9933 - recall_11: 0.9931 -
val_accuracy: 0.9490 - val_loss: 0.5359 - val_precision_11: 0.9519 -
val_recall_11: 0.9474 - learning_rate: 0.0010
Epoch 9/50
79/79            121s 2s/step -
accuracy: 0.9857 - loss: 0.3715 - precision_11: 0.9857 - recall_11: 0.9857 -
val_accuracy: 0.9569 - val_loss: 0.4644 - val_precision_11: 0.9615 -
val_recall_11: 0.9553 - learning_rate: 0.0010
Epoch 10/50
79/79            116s 1s/step -
accuracy: 0.9891 - loss: 0.3274 - precision_11: 0.9894 - recall_11: 0.9891 -
val_accuracy: 0.9410 - val_loss: 0.5002 - val_precision_11: 0.9410 -
val_recall_11: 0.9410 - learning_rate: 0.0010
Epoch 11/50
79/79            116s 1s/step -
```

```
accuracy: 0.9873 - loss: 0.3340 - precision_11: 0.9878 - recall_11: 0.9873 -
val_accuracy: 0.9139 - val_loss: 0.5525 - val_precision_11: 0.9152 -
val_recall_11: 0.9123 - learning_rate: 0.0010
Epoch 12/50
79/79            117s 1s/step -
accuracy: 0.9889 - loss: 0.2901 - precision_11: 0.9890 - recall_11: 0.9889 -
val_accuracy: 0.9091 - val_loss: 0.6122 - val_precision_11: 0.9098 -
val_recall_11: 0.9011 - learning_rate: 0.0010
Epoch 13/50
79/79            116s 1s/step -
accuracy: 0.9871 - loss: 0.2798 - precision_11: 0.9875 - recall_11: 0.9871 -
val_accuracy: 0.9362 - val_loss: 0.4498 - val_precision_11: 0.9377 -
val_recall_11: 0.9362 - learning_rate: 0.0010
Epoch 14/50
79/79            116s 1s/step -
accuracy: 0.9822 - loss: 0.2908 - precision_11: 0.9833 - recall_11: 0.9810 -
val_accuracy: 0.9266 - val_loss: 0.5284 - val_precision_11: 0.9265 -
val_recall_11: 0.9250 - learning_rate: 0.0010
Epoch 15/50
79/79            117s 1s/step -
accuracy: 0.9754 - loss: 0.3034 - precision_11: 0.9775 - recall_11: 0.9754 -
val_accuracy: 0.9378 - val_loss: 0.4553 - val_precision_11: 0.9378 -
val_recall_11: 0.9378 - learning_rate: 0.0010
Epoch 16/50
79/79            116s 1s/step -
accuracy: 0.9898 - loss: 0.2477 - precision_11: 0.9909 - recall_11: 0.9896 -
val_accuracy: 0.9394 - val_loss: 0.4391 - val_precision_11: 0.9408 -
val_recall_11: 0.9378 - learning_rate: 0.0010
Epoch 17/50
79/79            116s 1s/step -
accuracy: 0.9926 - loss: 0.2278 - precision_11: 0.9926 - recall_11: 0.9926 -
val_accuracy: 0.9442 - val_loss: 0.3898 - val_precision_11: 0.9472 -
val_recall_11: 0.9442 - learning_rate: 0.0010
Epoch 18/50
79/79            116s 1s/step -
accuracy: 0.9942 - loss: 0.2032 - precision_11: 0.9948 - recall_11: 0.9941 -
val_accuracy: 0.9442 - val_loss: 0.4018 - val_precision_11: 0.9456 -
val_recall_11: 0.9426 - learning_rate: 0.0010
Epoch 19/50
79/79            116s 1s/step -
accuracy: 0.9834 - loss: 0.2198 - precision_11: 0.9850 - recall_11: 0.9833 -
val_accuracy: 0.9474 - val_loss: 0.4001 - val_precision_11: 0.9489 -
val_recall_11: 0.9474 - learning_rate: 0.0010
Epoch 20/50
79/79            117s 1s/step -
accuracy: 0.9941 - loss: 0.1923 - precision_11: 0.9941 - recall_11: 0.9941 -
val_accuracy: 0.9314 - val_loss: 0.4691 - val_precision_11: 0.9314 -
val_recall_11: 0.9314 - learning_rate: 0.0010
```

```
Epoch 21/50
79/79            115s 1s/step -
accuracy: 0.9932 - loss: 0.1877 - precision_11: 0.9932 - recall_11: 0.9932 -
val_accuracy: 0.9474 - val_loss: 0.3815 - val_precision_11: 0.9487 -
val_recall_11: 0.9442 - learning_rate: 0.0010
Epoch 22/50
79/79            120s 2s/step -
accuracy: 0.9874 - loss: 0.1976 - precision_11: 0.9874 - recall_11: 0.9864 -
val_accuracy: 0.9522 - val_loss: 0.3465 - val_precision_11: 0.9552 -
val_recall_11: 0.9522 - learning_rate: 0.0010
Epoch 23/50
79/79            115s 1s/step -
accuracy: 0.9824 - loss: 0.2282 - precision_11: 0.9836 - recall_11: 0.9822 -
val_accuracy: 0.9458 - val_loss: 0.3607 - val_precision_11: 0.9473 -
val_recall_11: 0.9458 - learning_rate: 0.0010
Epoch 24/50
79/79            115s 1s/step -
accuracy: 0.9915 - loss: 0.1833 - precision_11: 0.9915 - recall_11: 0.9915 -
val_accuracy: 0.9075 - val_loss: 0.5115 - val_precision_11: 0.9075 -
val_recall_11: 0.9075 - learning_rate: 0.0010
Epoch 25/50
79/79            116s 1s/step -
accuracy: 0.9895 - loss: 0.1728 - precision_11: 0.9900 - recall_11: 0.9895 -
val_accuracy: 0.9474 - val_loss: 0.3808 - val_precision_11: 0.9489 -
val_recall_11: 0.9474 - learning_rate: 0.0010
Epoch 26/50
79/79            117s 1s/step -
accuracy: 0.9919 - loss: 0.1630 - precision_11: 0.9919 - recall_11: 0.9915 -
val_accuracy: 0.8676 - val_loss: 0.7373 - val_precision_11: 0.8712 -
val_recall_11: 0.8628 - learning_rate: 0.0010
Epoch 27/50
79/79            116s 1s/step -
accuracy: 0.9850 - loss: 0.1868 - precision_11: 0.9849 - recall_11: 0.9845 -
val_accuracy: 0.9298 - val_loss: 0.4813 - val_precision_11: 0.9297 -
val_recall_11: 0.9282 - learning_rate: 0.0010
Epoch 28/50
79/79            117s 1s/step -
accuracy: 0.9925 - loss: 0.1657 - precision_11: 0.9930 - recall_11: 0.9924 -
val_accuracy: 0.9633 - val_loss: 0.3095 - val_precision_11: 0.9633 -
val_recall_11: 0.9633 - learning_rate: 3.0000e-04
Epoch 29/50
79/79            116s 1s/step -
accuracy: 0.9971 - loss: 0.1463 - precision_11: 0.9971 - recall_11: 0.9971 -
val_accuracy: 0.9633 - val_loss: 0.2921 - val_precision_11: 0.9633 -
val_recall_11: 0.9633 - learning_rate: 3.0000e-04
Epoch 30/50
79/79            117s 1s/step -
accuracy: 0.9988 - loss: 0.1394 - precision_11: 0.9988 - recall_11: 0.9988 -
```

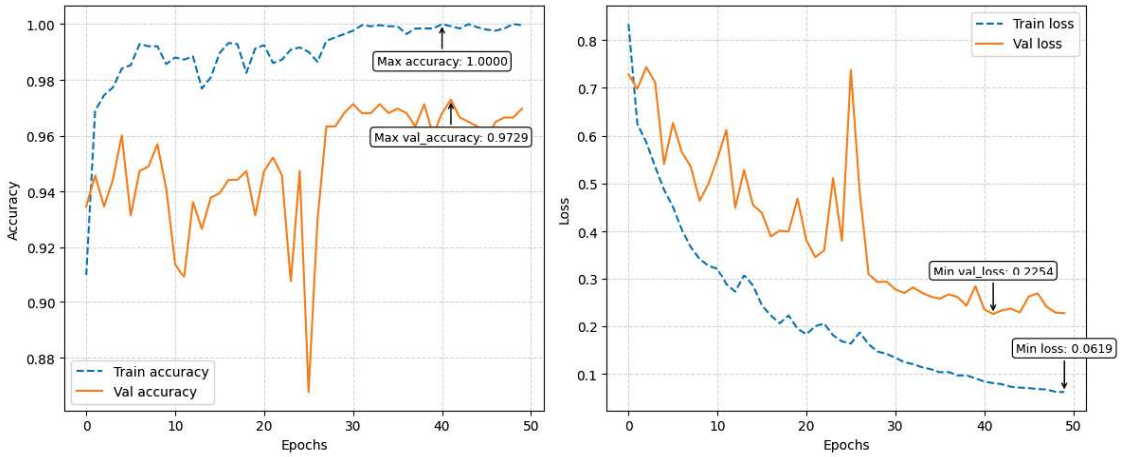
```
val_accuracy: 0.9681 - val_loss: 0.2934 - val_precision_11: 0.9681 -
val_recall_11: 0.9681 - learning_rate: 3.0000e-04
Epoch 31/50
79/79            117s 1s/step -
accuracy: 0.9972 - loss: 0.1369 - precision_11: 0.9972 - recall_11: 0.9972 -
val_accuracy: 0.9713 - val_loss: 0.2771 - val_precision_11: 0.9712 -
val_recall_11: 0.9697 - learning_rate: 3.0000e-04
Epoch 32/50
79/79            116s 1s/step -
accuracy: 0.9996 - loss: 0.1257 - precision_11: 0.9996 - recall_11: 0.9996 -
val_accuracy: 0.9681 - val_loss: 0.2694 - val_precision_11: 0.9696 -
val_recall_11: 0.9681 - learning_rate: 3.0000e-04
Epoch 33/50
79/79            116s 1s/step -
accuracy: 0.9990 - loss: 0.1243 - precision_11: 0.9990 - recall_11: 0.9990 -
val_accuracy: 0.9681 - val_loss: 0.2812 - val_precision_11: 0.9681 -
val_recall_11: 0.9681 - learning_rate: 3.0000e-04
Epoch 34/50
79/79            145s 1s/step -
accuracy: 1.0000 - loss: 0.1145 - precision_11: 1.0000 - recall_11: 1.0000 -
val_accuracy: 0.9713 - val_loss: 0.2697 - val_precision_11: 0.9713 -
val_recall_11: 0.9713 - learning_rate: 3.0000e-04
Epoch 35/50
79/79            116s 1s/step -
accuracy: 0.9988 - loss: 0.1111 - precision_11: 0.9988 - recall_11: 0.9988 -
val_accuracy: 0.9681 - val_loss: 0.2618 - val_precision_11: 0.9681 -
val_recall_11: 0.9665 - learning_rate: 3.0000e-04
Epoch 36/50
79/79            116s 1s/step -
accuracy: 0.9994 - loss: 0.1040 - precision_11: 0.9994 - recall_11: 0.9994 -
val_accuracy: 0.9697 - val_loss: 0.2575 - val_precision_11: 0.9697 -
val_recall_11: 0.9697 - learning_rate: 3.0000e-04
Epoch 37/50
79/79            116s 1s/step -
accuracy: 0.9953 - loss: 0.1060 - precision_11: 0.9959 - recall_11: 0.9953 -
val_accuracy: 0.9681 - val_loss: 0.2666 - val_precision_11: 0.9681 -
val_recall_11: 0.9681 - learning_rate: 3.0000e-04
Epoch 38/50
79/79            116s 1s/step -
accuracy: 0.9996 - loss: 0.0960 - precision_11: 0.9996 - recall_11: 0.9996 -
val_accuracy: 0.9633 - val_loss: 0.2610 - val_precision_11: 0.9633 -
val_recall_11: 0.9633 - learning_rate: 3.0000e-04
Epoch 39/50
79/79            116s 1s/step -
accuracy: 0.9985 - loss: 0.0978 - precision_11: 0.9985 - recall_11: 0.9985 -
val_accuracy: 0.9713 - val_loss: 0.2429 - val_precision_11: 0.9713 -
val_recall_11: 0.9713 - learning_rate: 3.0000e-04
Epoch 40/50
```

```
79/79            116s 1s/step -
accuracy: 0.9996 - loss: 0.0893 - precision_11: 0.9996 - recall_11: 0.9996 -
val_accuracy: 0.9601 - val_loss: 0.2837 - val_precision_11: 0.9601 -
val_recall_11: 0.9601 - learning_rate: 3.0000e-04
Epoch 41/50
79/79            115s 1s/step -
accuracy: 1.0000 - loss: 0.0850 - precision_11: 1.0000 - recall_11: 1.0000 -
val_accuracy: 0.9681 - val_loss: 0.2350 - val_precision_11: 0.9681 -
val_recall_11: 0.9665 - learning_rate: 3.0000e-04
Epoch 42/50
79/79            116s 1s/step -
accuracy: 0.9996 - loss: 0.0812 - precision_11: 0.9996 - recall_11: 0.9996 -
val_accuracy: 0.9729 - val_loss: 0.2254 - val_precision_11: 0.9729 -
val_recall_11: 0.9729 - learning_rate: 3.0000e-04
Epoch 43/50
79/79            117s 1s/step -
accuracy: 0.9990 - loss: 0.0780 - precision_11: 0.9990 - recall_11: 0.9990 -
val_accuracy: 0.9665 - val_loss: 0.2333 - val_precision_11: 0.9665 -
val_recall_11: 0.9665 - learning_rate: 3.0000e-04
Epoch 44/50
79/79            116s 1s/step -
accuracy: 1.0000 - loss: 0.0740 - precision_11: 1.0000 - recall_11: 1.0000 -
val_accuracy: 0.9649 - val_loss: 0.2367 - val_precision_11: 0.9649 -
val_recall_11: 0.9649 - learning_rate: 3.0000e-04
Epoch 45/50
79/79            116s 1s/step -
accuracy: 0.9984 - loss: 0.0720 - precision_11: 0.9984 - recall_11: 0.9984 -
val_accuracy: 0.9633 - val_loss: 0.2286 - val_precision_11: 0.9633 -
val_recall_11: 0.9633 - learning_rate: 3.0000e-04
Epoch 46/50
79/79            116s 1s/step -
accuracy: 0.9977 - loss: 0.0710 - precision_11: 0.9977 - recall_11: 0.9977 -
val_accuracy: 0.9585 - val_loss: 0.2621 - val_precision_11: 0.9601 -
val_recall_11: 0.9585 - learning_rate: 3.0000e-04
Epoch 47/50
79/79            116s 1s/step -
accuracy: 0.9978 - loss: 0.0684 - precision_11: 0.9991 - recall_11: 0.9978 -
val_accuracy: 0.9649 - val_loss: 0.2686 - val_precision_11: 0.9649 -
val_recall_11: 0.9649 - learning_rate: 3.0000e-04
Epoch 48/50
79/79            116s 1s/step -
accuracy: 0.9987 - loss: 0.0671 - precision_11: 0.9988 - recall_11: 0.9987 -
val_accuracy: 0.9665 - val_loss: 0.2407 - val_precision_11: 0.9665 -
val_recall_11: 0.9665 - learning_rate: 9.0000e-05
Epoch 49/50
79/79            116s 1s/step -
accuracy: 1.0000 - loss: 0.0629 - precision_11: 1.0000 - recall_11: 1.0000 -
val_accuracy: 0.9665 - val_loss: 0.2282 - val_precision_11: 0.9681 -
```

```
val_recall_11: 0.9665 - learning_rate: 9.0000e-05
Epoch 50/50
79/79              116s 1s/step -
accuracy: 0.9999 - loss: 0.0614 - precision_11: 0.9999 - recall_11: 0.9999 -
val_accuracy: 0.9697 - val_loss: 0.2271 - val_precision_11: 0.9697 -
val_recall_11: 0.9697 - learning_rate: 9.0000e-05
```

6.5.3 Plot loss & accuracy for training and validation wrt epoch

[148]:
```
plot_loss_and_accuracy(["accuracy","loss"],model_fit)
```



6.5.4 Tensorboard

[149]:
```
%load_ext tensorboard
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

6.5.5 Model Evaluation, Mlflow Logging and Printing Metrics

[150]:
```
ckpt_path
```

[150]: 'checkpoints/1_pretrained_inceptionv3_trainable0.weights.h5'

[151]:
```
log_dir
```

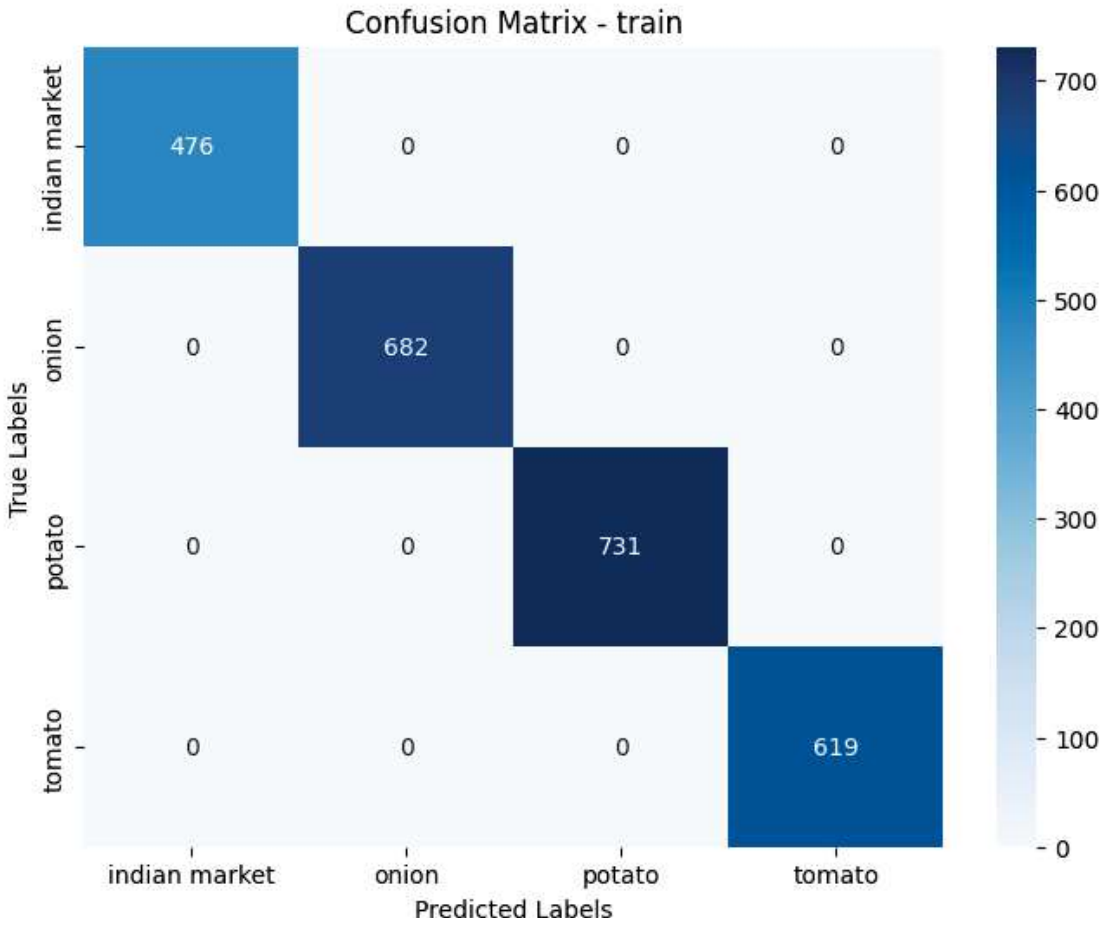
[151]: 'logs/1_pretrained_inceptionv3_trainable0'

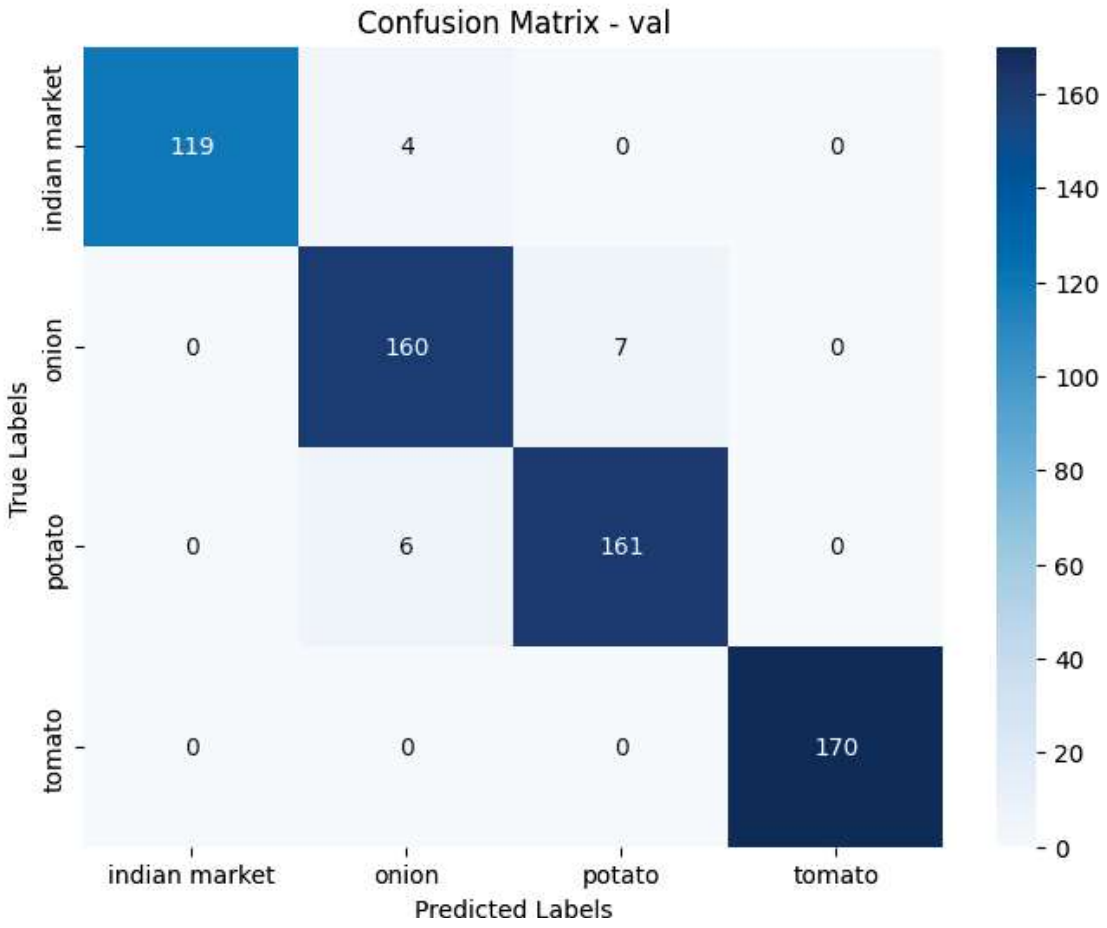
[152]:
```
run_name = "1_pretrained_inceptionv3_trainable0"
run_name
```

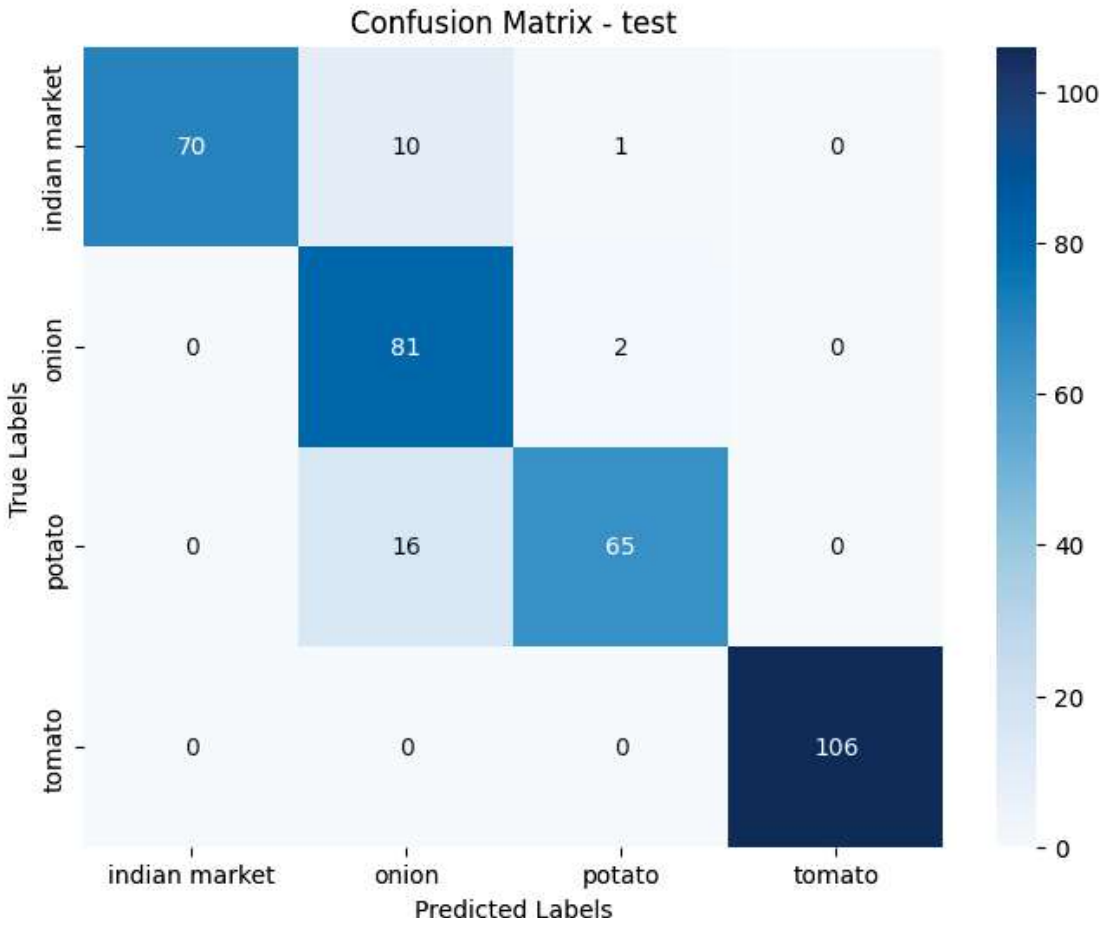
[152]: '1_pretrained_inceptionv3_trainable0'

[153]:
```
evaluate_model(model, train_ds, val_ds, test_ds,␣
↪class_names,run_name=run_name,ckpt_path= ckpt_path)
```

```
2025/02/11 02:03:20 WARNING mlflow.tensorflow: You are saving a TensorFlow Core
model or Keras model without a signature. Inference with
mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation
accepts pandas DataFrames as inference inputs.
2025/02/11 02:03:50 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.


train Metrics:
 Accuracy: 100.00%
 Precision: 100.00%
 Recall: 100.00%

val Metrics:
 Accuracy: 97.29%
 Precision: 97.49%
 Recall: 97.24%

test Metrics:
 Accuracy: 91.74%
 Precision: 92.82%
 Recall: 91.06%
```


Confusion Matrix - train
True Labels
Predicted Labels
indian market
onion
potato
tomato
476
0
0
0
0
682
0
0
0
0
731
0
0
0
0
619
700
600
500
400
300
200
100
0


Confusion Matrix - val
indian market
onion
potato
tomato
True Labels
119
4
0
0
0
160
7
0
0
6
161
0
0
0
0
170
indian market
onion
potato
tomato
Predicted Labels
160
140
120
100
80
60
40
20
0




```
Classification Report - train
               precision    recall  f1-score   support

indian market       1.00      1.00      1.00       476
        onion       1.00      1.00      1.00       682
       potato       1.00      1.00      1.00       731
       tomato       1.00      1.00      1.00       619

     accuracy                           1.00      2508
    macro avg       1.00      1.00      1.00      2508
 weighted avg       1.00      1.00      1.00      2508


Classification Report - val
               precision    recall  f1-score   support

indian market       1.00      0.97      0.98       123
```

```
        onion       0.94      0.96      0.95       167
       potato       0.96      0.96      0.96       167
       tomato       1.00      1.00      1.00       170

     accuracy                           0.97       627
    macro avg       0.97      0.97      0.97       627
 weighted avg       0.97      0.97      0.97       627


Classification Report - test
               precision    recall  f1-score   support

indian market       1.00      0.86      0.93        81
        onion       0.76      0.98      0.85        83
       potato       0.96      0.80      0.87        81
       tomato       1.00      1.00      1.00       106

     accuracy                           0.92       351
    macro avg       0.93      0.91      0.91       351
 weighted avg       0.93      0.92      0.92       351


Class-wise Accuracy:

Train Class-wise Accuracy:
 indian market: 100.00% (476/476)
 onion: 100.00% (682/682)
 potato: 100.00% (731/731)
 tomato: 100.00% (619/619)

Val Class-wise Accuracy:
 indian market: 96.75% (119/123)
 onion: 95.81% (160/167)
 potato: 96.41% (161/167)
 tomato: 100.00% (170/170)

Test Class-wise Accuracy:
 indian market: 86.42% (70/81)
 onion: 97.59% (81/83)
 potato: 80.25% (65/81)
 tomato: 100.00% (106/106)

Random Test Predictions:
```

True: indian market
Pred: indian market

True: indian market
Pred: indian market

True: onion
Pred: onion

True: onion
Pred: onion

True: potato
Pred: potato

True: potato
Pred: potato

True: tomato
Pred: tomato

True: tomato
Pred: tomato

6.5.6 Save the model

[154]:
```
model.save("saved_models/1_pretrained_inceptionv3_trainable0_model.keras")
```

6.6 DATA AUGMENTED VGG16

6.6.1 Load the previously trained model architecture and weights

[155]:
```
# Load the entire model (architecture + weights)
model = keras.models.load_model("saved_models/
 ↪1_pretrained_inceptionv3_trainable0_model.keras")
```

[156]:
```
model.summary()
```

Model: "pretrained_inceptionv3"

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 6, 6, 2048)	21,802,784
global_average_pooling2d_12 (GlobalAveragePooling2D)	(None, 2048)	0
dense_36 (Dense)	(None, 512)	1,049,088
batch_normalization_110 (BatchNormalization)	(None, 512)	2,048
dropout_14 (Dropout)	(None, 512)	0
dense_37 (Dense)	(None, 256)	131,328
batch_normalization_111 (BatchNormalization)	(None, 256)	1,024
dense_38 (Dense)	(None, 4)	1,028

Total params: 25,353,262 (96.72 MB)

Trainable params: 1,182,980 (4.51 MB)

Non-trainable params: 21,804,320 (83.18 MB)

```
Optimizer params: 2,365,962 (9.03 MB)
```

6.6.2 Model Compilation and Training

[157]:
```
ckpt_path = "checkpoints/1_pretrained_inceptionv3_aug_trainable0.weights.h5"
```

[158]:
```
log_dir = 'logs/1_pretrained_inceptionv3_aug_trainable0'
```

[159]:
```
run_name = '1_pretrained_inceptionv3_aug_trainable0'
```

[160]:
```
model_fit = compile_train(model, train_aug_ds, val_aug_ds,␣
↪log_dir=log_dir,epochs=20,ckpt_path=ckpt_path)
```

```
Epoch 1/20
79/79             89s 1s/step -
accuracy: 0.9471 - loss: 0.3011 - precision_12: 0.9478 - recall_12: 0.9452 -
val_accuracy: 0.8644 - val_loss: 0.8308 - val_precision_12: 0.8640 -
val_recall_12: 0.8612 - learning_rate: 0.0010
Epoch 2/20
79/79             79s 995ms/step -
accuracy: 0.9529 - loss: 0.2465 - precision_12: 0.9550 - recall_12: 0.9512 -
val_accuracy: 0.9617 - val_loss: 0.2177 - val_precision_12: 0.9633 -
val_recall_12: 0.9617 - learning_rate: 0.0010
Epoch 3/20
79/79             78s 986ms/step -
accuracy: 0.9590 - loss: 0.2061 - precision_12: 0.9612 - recall_12: 0.9577 -
val_accuracy: 0.9410 - val_loss: 0.3192 - val_precision_12: 0.9440 -
val_recall_12: 0.9410 - learning_rate: 0.0010
Epoch 4/20
79/79             78s 983ms/step -
accuracy: 0.9573 - loss: 0.2169 - precision_12: 0.9617 - recall_12: 0.9570 -
val_accuracy: 0.9537 - val_loss: 0.2454 - val_precision_12: 0.9553 -
val_recall_12: 0.9537 - learning_rate: 0.0010
Epoch 5/20
79/79             79s 990ms/step -
accuracy: 0.9627 - loss: 0.2095 - precision_12: 0.9644 - recall_12: 0.9618 -
val_accuracy: 0.9537 - val_loss: 0.2779 - val_precision_12: 0.9537 -
val_recall_12: 0.9522 - learning_rate: 0.0010
Epoch 6/20
79/79             79s 986ms/step -
accuracy: 0.9704 - loss: 0.1912 - precision_12: 0.9707 - recall_12: 0.9694 -
val_accuracy: 0.9314 - val_loss: 0.3298 - val_precision_12: 0.9313 -
val_recall_12: 0.9298 - learning_rate: 0.0010
Epoch 7/20
79/79             79s 994ms/step -
accuracy: 0.9788 - loss: 0.1709 - precision_12: 0.9788 - recall_12: 0.9773 -
val_accuracy: 0.9298 - val_loss: 0.3051 - val_precision_12: 0.9328 -
```

```
val_recall_12: 0.9298 - learning_rate: 0.0010
Epoch 8/20
79/79            78s 977ms/step -
accuracy: 0.9781 - loss: 0.1804 - precision_12: 0.9782 - recall_12: 0.9774 -
val_accuracy: 0.9569 - val_loss: 0.2174 - val_precision_12: 0.9569 -
val_recall_12: 0.9569 - learning_rate: 3.0000e-04
Epoch 9/20
79/79            78s 985ms/step -
accuracy: 0.9833 - loss: 0.1555 - precision_12: 0.9842 - recall_12: 0.9833 -
val_accuracy: 0.9633 - val_loss: 0.2243 - val_precision_12: 0.9649 -
val_recall_12: 0.9633 - learning_rate: 3.0000e-04
Epoch 10/20
79/79            78s 980ms/step -
accuracy: 0.9783 - loss: 0.1539 - precision_12: 0.9794 - recall_12: 0.9783 -
val_accuracy: 0.9649 - val_loss: 0.2119 - val_precision_12: 0.9649 -
val_recall_12: 0.9649 - learning_rate: 3.0000e-04
Epoch 11/20
79/79            78s 975ms/step -
accuracy: 0.9815 - loss: 0.1580 - precision_12: 0.9815 - recall_12: 0.9805 -
val_accuracy: 0.9649 - val_loss: 0.2113 - val_precision_12: 0.9649 -
val_recall_12: 0.9633 - learning_rate: 3.0000e-04
Epoch 12/20
79/79            78s 977ms/step -
accuracy: 0.9799 - loss: 0.1476 - precision_12: 0.9830 - recall_12: 0.9787 -
val_accuracy: 0.9697 - val_loss: 0.2016 - val_precision_12: 0.9712 -
val_recall_12: 0.9697 - learning_rate: 3.0000e-04
Epoch 13/20
79/79            77s 969ms/step -
accuracy: 0.9820 - loss: 0.1451 - precision_12: 0.9835 - recall_12: 0.9811 -
val_accuracy: 0.9665 - val_loss: 0.1888 - val_precision_12: 0.9681 -
val_recall_12: 0.9665 - learning_rate: 3.0000e-04
Epoch 14/20
79/79            77s 971ms/step -
accuracy: 0.9866 - loss: 0.1333 - precision_12: 0.9867 - recall_12: 0.9861 -
val_accuracy: 0.9633 - val_loss: 0.2071 - val_precision_12: 0.9633 -
val_recall_12: 0.9633 - learning_rate: 3.0000e-04
Epoch 15/20
79/79            78s 982ms/step -
accuracy: 0.9790 - loss: 0.1380 - precision_12: 0.9790 - recall_12: 0.9790 -
val_accuracy: 0.9729 - val_loss: 0.1797 - val_precision_12: 0.9728 -
val_recall_12: 0.9681 - learning_rate: 3.0000e-04
Epoch 16/20
79/79            78s 984ms/step -
accuracy: 0.9850 - loss: 0.1290 - precision_12: 0.9853 - recall_12: 0.9850 -
val_accuracy: 0.9617 - val_loss: 0.1920 - val_precision_12: 0.9617 -
val_recall_12: 0.9601 - learning_rate: 3.0000e-04
Epoch 17/20
79/79            78s 982ms/step -
```

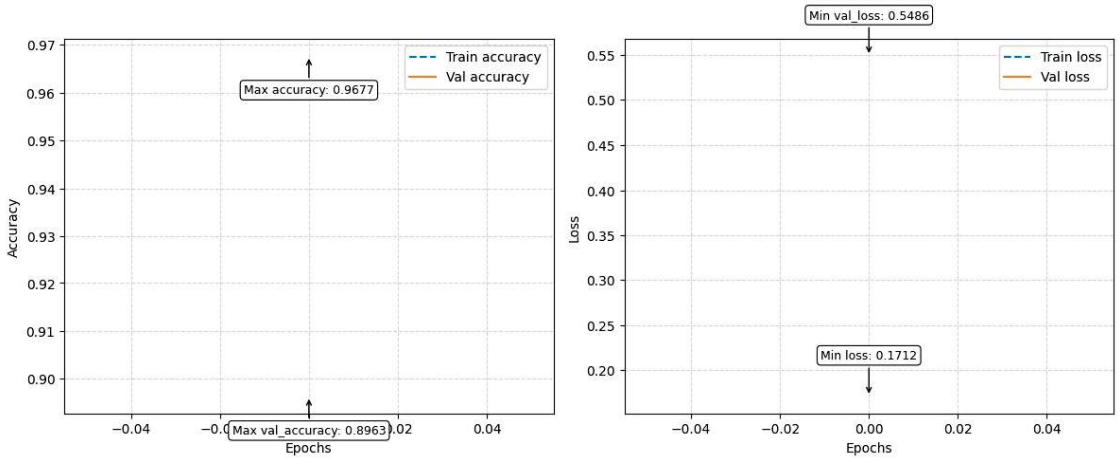
```
accuracy: 0.9878 - loss: 0.1263 - precision_12: 0.9885 - recall_12: 0.9874 -
val_accuracy: 0.9681 - val_loss: 0.1694 - val_precision_12: 0.9681 -
val_recall_12: 0.9681 - learning_rate: 3.0000e-04
Epoch 18/20
79/79              77s 972ms/step -
accuracy: 0.9875 - loss: 0.1335 - precision_12: 0.9879 - recall_12: 0.9875 -
val_accuracy: 0.9729 - val_loss: 0.1745 - val_precision_12: 0.9729 -
val_recall_12: 0.9729 - learning_rate: 3.0000e-04
Epoch 19/20
79/79              78s 977ms/step -
accuracy: 0.9884 - loss: 0.1175 - precision_12: 0.9885 - recall_12: 0.9884 -
val_accuracy: 0.9617 - val_loss: 0.2057 - val_precision_12: 0.9617 -
val_recall_12: 0.9617 - learning_rate: 3.0000e-04
Epoch 20/20
79/79              78s 977ms/step -
accuracy: 0.9832 - loss: 0.1290 - precision_12: 0.9837 - recall_12: 0.9832 -
val_accuracy: 0.9649 - val_loss: 0.2000 - val_precision_12: 0.9649 -
val_recall_12: 0.9649 - learning_rate: 3.0000e-04
```

[161]:
```
model_fit = compile_train(model, train_aug_ds, val_aug_ds,␣
↪log_dir=log_dir,epochs=1,ckpt_path=ckpt_path)
```

```
79/79              89s 1s/step -
accuracy: 0.9696 - loss: 0.1551 - precision_13: 0.9719 - recall_13: 0.9691 -
val_accuracy: 0.8963 - val_loss: 0.5486 - val_precision_13: 0.9035 -
val_recall_13: 0.8963 - learning_rate: 0.0010
```

6.6.3 Plot loss & accuracy for training and validation wrt epoch

[162]:
```
plot_loss_and_accuracy(["accuracy","loss"],model_fit)
```



6.6.4 Tensorboard

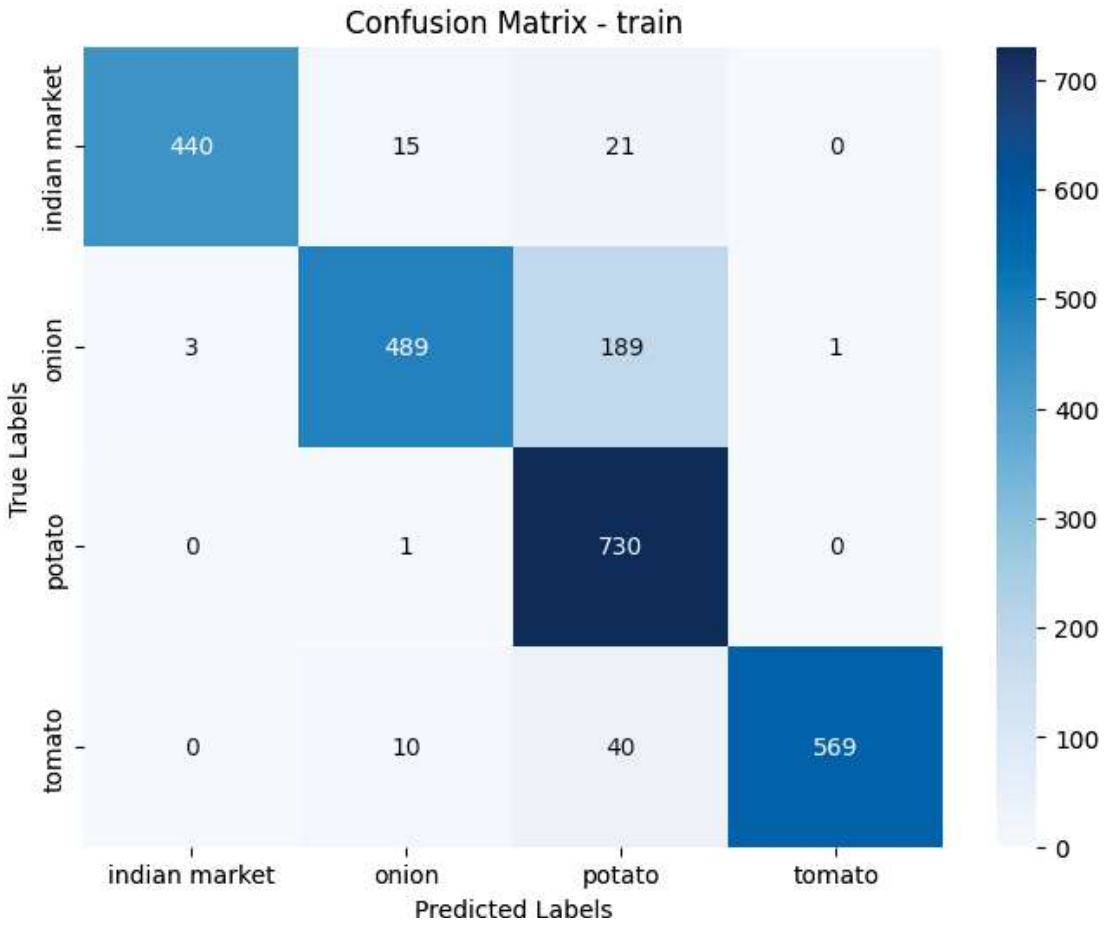
[163]:
```
%load_ext tensorboard
```

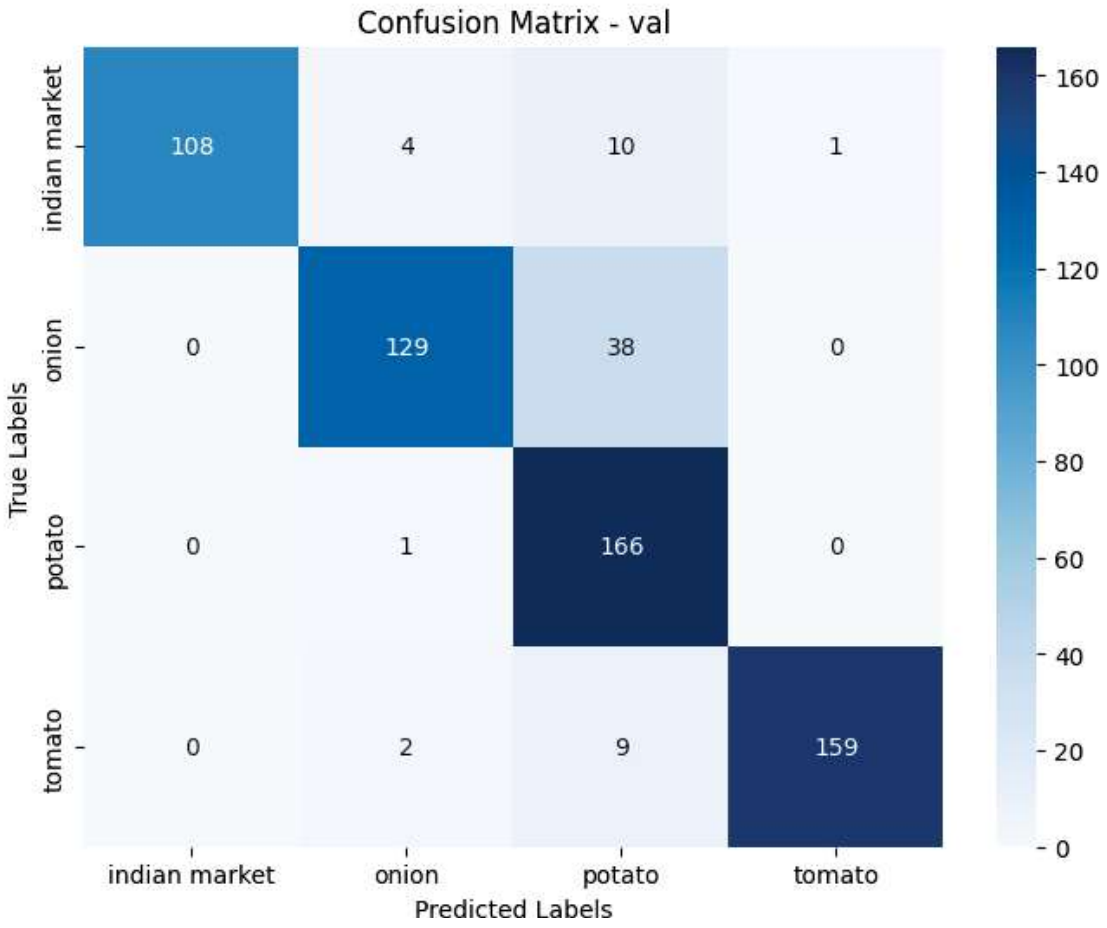
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

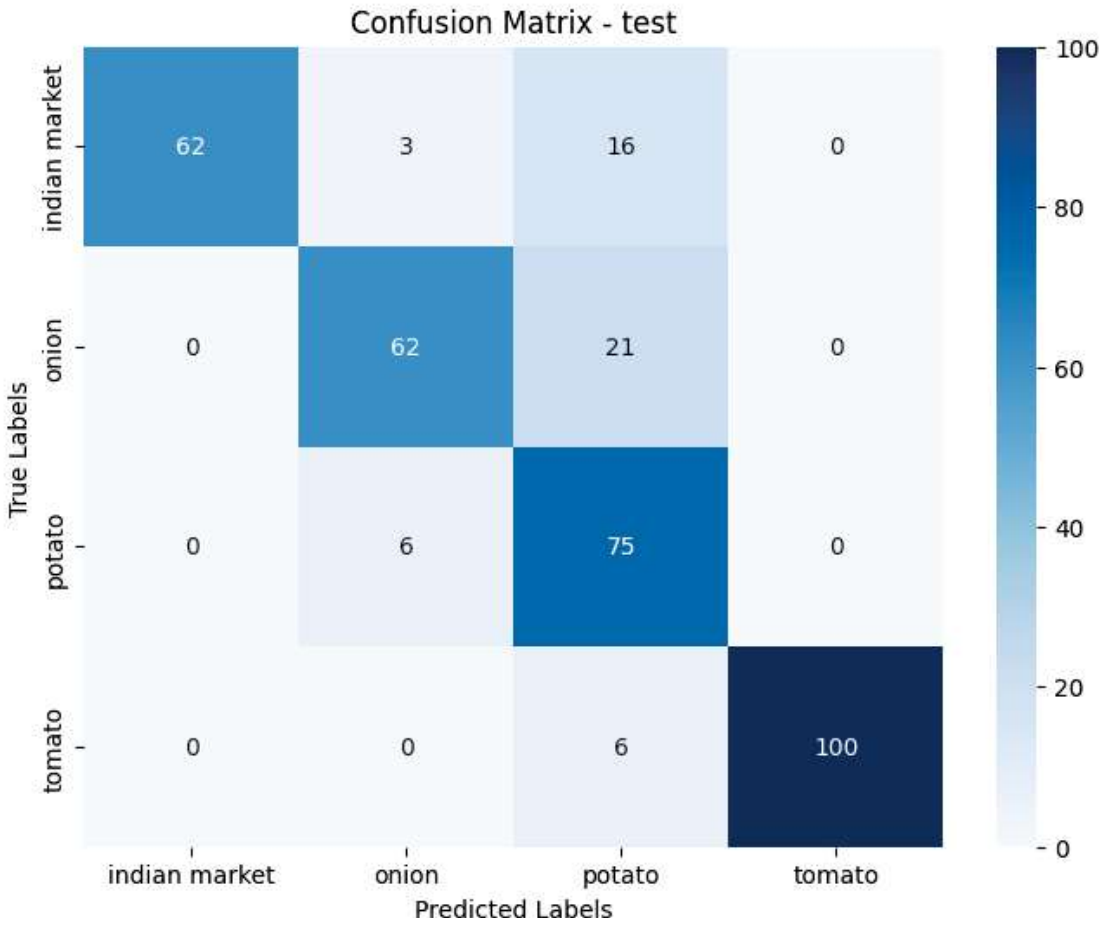
6.6.5 Model Evaluation, Mlflow Logging and Printing Metrics

[164]:
```
evaluate_model(model, train_aug_ds, val_aug_ds, test_aug_ds,␣
 ↪class_names,run_name=run_name,ckpt_path= ckpt_path)
```

```
2025/02/11 02:34:37 WARNING mlflow.tensorflow: You are saving a TensorFlow Core
model or Keras model without a signature. Inference with
mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation
accepts pandas DataFrames as inference inputs.
2025/02/11 02:34:50 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.


train Metrics:
 Accuracy: 88.84%
 Precision: 92.15%
 Recall: 88.98%

val Metrics:
 Accuracy: 89.63%
 Precision: 92.17%
 Recall: 89.50%

test Metrics:
 Accuracy: 85.19%
 Precision: 87.72%
 Recall: 84.54%
```


Confusion Matrix - train
indian market
onion
potato
tomato
True Labels
440
15
21
0
3
489
189
1
0
1
730
0
0
10
40
569
indian market
onion
potato
tomato
Predicted Labels
700
600
500
400
300
200
100
0


Confusion Matrix - val
indian market
onion
potato
tomato
True Labels
108
4
10
1
0
129
38
0
0
1
166
0
0
2
9
159
indian market
onion
potato
tomato
Predicted Labels
160
140
120
100
80
60
40
20
0




```
Classification Report - train
              precision    recall  f1-score   support

indian market       0.99      0.92      0.96       476
        onion       0.95      0.72      0.82       682
       potato       0.74      1.00      0.85       731
       tomato       1.00      0.92      0.96       619

     accuracy                           0.89      2508
    macro avg       0.92      0.89      0.90      2508
 weighted avg       0.91      0.89      0.89      2508


Classification Report - val
              precision    recall  f1-score   support

indian market       1.00      0.88      0.94       123
```

```
        onion       0.95      0.77      0.85       167
       potato       0.74      0.99      0.85       167
       tomato       0.99      0.94      0.96       170

     accuracy                           0.90       627
    macro avg       0.92      0.89      0.90       627
 weighted avg       0.92      0.90      0.90       627


Classification Report - test
               precision    recall  f1-score   support

indian market       1.00      0.77      0.87        81
        onion       0.87      0.75      0.81        83
       potato       0.64      0.93      0.75        81
       tomato       1.00      0.94      0.97       106

     accuracy                           0.85       351
    macro avg       0.88      0.85      0.85       351
 weighted avg       0.89      0.85      0.86       351


Class-wise Accuracy:

Train Class-wise Accuracy:
 indian market: 92.44% (440/476)
 onion: 71.70% (489/682)
 potato: 99.86% (730/731)
 tomato: 91.92% (569/619)

Val Class-wise Accuracy:
 indian market: 87.80% (108/123)
 onion: 77.25% (129/167)
 potato: 99.40% (166/167)
 tomato: 93.53% (159/170)

Test Class-wise Accuracy:
 indian market: 76.54% (62/81)
 onion: 74.70% (62/83)
 potato: 92.59% (75/81)
 tomato: 94.34% (100/106)

Random Test Predictions:
```

True: indian market
Pred: indian market

True: indian market
Pred: indian market

True: onion
Pred: potato

True: onion
Pred: onion

True: potato
Pred: onion

True: potato
Pred: potato

True: tomato
Pred: tomato

True: tomato
Pred: tomato

6.6.6 Save the model

[165]:
```
model.save("saved_models/1_pretrained_inceptionv3_aug_trainable0_model.keras")
```

6.7 RESNET50

6.7.1 Building the model

[166]:
```
def pretrained_mobilenetv2(height=256, width=256, num_classes=4,␣
↪trainable_layers=0, ckpt_path=None):
    base_model = applications.MobileNetV2(weights="imagenet",␣
↪include_top=False, input_shape=(height, width, 3))

    # Freeze all layers initially
    base_model.trainable = False

    # If fine-tuning, unfreeze last `trainable_layers`
    if trainable_layers > 0:
        for layer in base_model.layers[-trainable_layers:]:
            layer.trainable = True

    model = keras.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(512, activation="relu", kernel_regularizer=regularizers.
↪l2(5e-4)),
        layers.BatchNormalization(),
        layers.Dropout(0.3),

        layers.Dense(256, activation="relu", kernel_regularizer=regularizers.
↪l2(5e-4)),
        layers.BatchNormalization(),

        layers.Dense(num_classes, activation="softmax")
    ], name="pretrained_mobilenetv2")

    # Load weights only if a valid checkpoint is provided
    if ckpt_path and os.path.exists(ckpt_path):
        print(f"Loading weights from {ckpt_path}")
        model.load_weights(ckpt_path)

    return model
```

[167]:
```
ckpt_path = "checkpoints/1_pretrained_mobilenetv2_trainable0.weights.h5"
log_dir = "logs/1_pretrained_mobilenetv2_trainable0"

# Load pretrained mobilenetv2 model
```

```
model = pretrained_mobilenetv2(height=256, width=256, num_classes=4,␣
 ↪trainable_layers=0,ckpt_path=ckpt_path)
```

```
C:\Users\saina\AppData\Local\Temp\ipykernel_11564\411930353.py:2: UserWarning:
`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160,
192, 224]. Weights for input shape (224, 224) will be loaded as the default.
  base_model = applications.MobileNetV2(weights="imagenet", include_top=False,
input_shape=(height, width, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicatio
ns/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h
5
9406464/9406464             2s
0us/step
```

[168]:
```
model.summary()
```

Model: "pretrained_mobilenetv2"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 8, 8, 1280)	2,257,984
global_average_pooling2d_13 (GlobalAveragePooling2D)	(None, 1280)	0
dense_39 (Dense)	(None, 512)	655,872
batch_normalization_112 (BatchNormalization)	(None, 512)	2,048
dropout_15 (Dropout)	(None, 512)	0
dense_40 (Dense)	(None, 256)	131,328
batch_normalization_113 (BatchNormalization)	(None, 256)	1,024
dense_41 (Dense)	(None, 4)	1,028

Total params: 3,049,284 (11.63 MB)

Trainable params: 789,764 (3.01 MB)

```
 Non-trainable params: 2,259,520 (8.62 MB)
```

6.7.2 Model Compilation and Training

[169]:
```
model_fit = compile_train(model, train_ds, val_ds,␣
 ↪log_dir=log_dir,epochs=50,ckpt_path=ckpt_path)
```

```
Epoch 1/50
79/79             57s 667ms/step -
accuracy: 0.8482 - loss: 0.9569 - precision_14: 0.8636 - recall_14: 0.8352 -
val_accuracy: 0.9633 - val_loss: 0.6440 - val_precision_14: 0.9633 -
val_recall_14: 0.9617 - learning_rate: 0.0010
Epoch 2/50
79/79             50s 633ms/step -
accuracy: 0.9781 - loss: 0.5865 - precision_14: 0.9791 - recall_14: 0.9764 -
val_accuracy: 0.9745 - val_loss: 0.5820 - val_precision_14: 0.9760 -
val_recall_14: 0.9729 - learning_rate: 0.0010
Epoch 3/50
79/79             50s 632ms/step -
accuracy: 0.9878 - loss: 0.5278 - precision_14: 0.9878 - recall_14: 0.9875 -
val_accuracy: 0.9697 - val_loss: 0.6175 - val_precision_14: 0.9696 -
val_recall_14: 0.9665 - learning_rate: 0.0010
Epoch 4/50
79/79             50s 637ms/step -
accuracy: 0.9903 - loss: 0.4948 - precision_14: 0.9911 - recall_14: 0.9896 -
val_accuracy: 0.9649 - val_loss: 0.6096 - val_precision_14: 0.9649 -
val_recall_14: 0.9649 - learning_rate: 0.0010
Epoch 5/50
79/79             52s 657ms/step -
accuracy: 0.9914 - loss: 0.4669 - precision_14: 0.9914 - recall_14: 0.9914 -
val_accuracy: 0.9809 - val_loss: 0.4762 - val_precision_14: 0.9808 -
val_recall_14: 0.9793 - learning_rate: 0.0010
Epoch 6/50
79/79             51s 645ms/step -
accuracy: 0.9901 - loss: 0.4354 - precision_14: 0.9901 - recall_14: 0.9900 -
val_accuracy: 0.9729 - val_loss: 0.5023 - val_precision_14: 0.9744 -
val_recall_14: 0.9697 - learning_rate: 0.0010
Epoch 7/50
79/79             51s 647ms/step -
accuracy: 0.9946 - loss: 0.3968 - precision_14: 0.9946 - recall_14: 0.9926 -
val_accuracy: 0.9856 - val_loss: 0.4153 - val_precision_14: 0.9856 -
val_recall_14: 0.9856 - learning_rate: 0.0010
Epoch 8/50
79/79             51s 648ms/step -
accuracy: 0.9931 - loss: 0.3662 - precision_14: 0.9939 - recall_14: 0.9931 -
val_accuracy: 0.9777 - val_loss: 0.4443 - val_precision_14: 0.9777 -
val_recall_14: 0.9777 - learning_rate: 0.0010
```

```
Epoch 9/50
79/79            51s 645ms/step -
accuracy: 0.9947 - loss: 0.3435 - precision_14: 0.9947 - recall_14: 0.9947 -
val_accuracy: 0.9777 - val_loss: 0.3905 - val_precision_14: 0.9777 -
val_recall_14: 0.9777 - learning_rate: 0.0010
Epoch 10/50
79/79            51s 640ms/step -
accuracy: 0.9870 - loss: 0.3349 - precision_14: 0.9871 - recall_14: 0.9870 -
val_accuracy: 0.9713 - val_loss: 0.4037 - val_precision_14: 0.9713 -
val_recall_14: 0.9713 - learning_rate: 0.0010
Epoch 11/50
79/79            51s 639ms/step -
accuracy: 0.9928 - loss: 0.3069 - precision_14: 0.9929 - recall_14: 0.9926 -
val_accuracy: 0.9585 - val_loss: 0.4291 - val_precision_14: 0.9601 -
val_recall_14: 0.9585 - learning_rate: 0.0010
Epoch 12/50
79/79            51s 641ms/step -
accuracy: 0.9929 - loss: 0.2990 - precision_14: 0.9929 - recall_14: 0.9926 -
val_accuracy: 0.9809 - val_loss: 0.3325 - val_precision_14: 0.9809 -
val_recall_14: 0.9809 - learning_rate: 0.0010
Epoch 13/50
79/79            51s 641ms/step -
accuracy: 0.9927 - loss: 0.2729 - precision_14: 0.9927 - recall_14: 0.9927 -
val_accuracy: 0.9809 - val_loss: 0.3055 - val_precision_14: 0.9809 -
val_recall_14: 0.9809 - learning_rate: 0.0010
Epoch 14/50
79/79            50s 635ms/step -
accuracy: 0.9901 - loss: 0.2665 - precision_14: 0.9904 - recall_14: 0.9901 -
val_accuracy: 0.9601 - val_loss: 0.3600 - val_precision_14: 0.9632 -
val_recall_14: 0.9601 - learning_rate: 0.0010
Epoch 15/50
79/79            50s 636ms/step -
accuracy: 0.9860 - loss: 0.2667 - precision_14: 0.9860 - recall_14: 0.9860 -
val_accuracy: 0.9793 - val_loss: 0.3043 - val_precision_14: 0.9808 -
val_recall_14: 0.9793 - learning_rate: 0.0010
Epoch 16/50
79/79            51s 638ms/step -
accuracy: 0.9889 - loss: 0.2464 - precision_14: 0.9893 - recall_14: 0.9889 -
val_accuracy: 0.9825 - val_loss: 0.2916 - val_precision_14: 0.9825 -
val_recall_14: 0.9825 - learning_rate: 0.0010
Epoch 17/50
79/79            50s 636ms/step -
accuracy: 0.9960 - loss: 0.2146 - precision_14: 0.9963 - recall_14: 0.9954 -
val_accuracy: 0.9793 - val_loss: 0.2851 - val_precision_14: 0.9792 -
val_recall_14: 0.9777 - learning_rate: 0.0010
Epoch 18/50
79/79            50s 637ms/step -
accuracy: 0.9957 - loss: 0.2020 - precision_14: 0.9957 - recall_14: 0.9957 -
```

```
val_accuracy: 0.9745 - val_loss: 0.2747 - val_precision_14: 0.9760 -
val_recall_14: 0.9745 - learning_rate: 0.0010
Epoch 19/50
79/79            51s 641ms/step -
accuracy: 0.9902 - loss: 0.2076 - precision_14: 0.9909 - recall_14: 0.9899 -
val_accuracy: 0.9777 - val_loss: 0.2714 - val_precision_14: 0.9777 -
val_recall_14: 0.9777 - learning_rate: 0.0010
Epoch 20/50
79/79            51s 647ms/step -
accuracy: 0.9895 - loss: 0.2001 - precision_14: 0.9895 - recall_14: 0.9885 -
val_accuracy: 0.9888 - val_loss: 0.2130 - val_precision_14: 0.9888 -
val_recall_14: 0.9872 - learning_rate: 0.0010
Epoch 21/50
79/79            50s 638ms/step -
accuracy: 0.9948 - loss: 0.1813 - precision_14: 0.9948 - recall_14: 0.9948 -
val_accuracy: 0.9809 - val_loss: 0.2593 - val_precision_14: 0.9809 -
val_recall_14: 0.9809 - learning_rate: 0.0010
Epoch 22/50
79/79            51s 638ms/step -
accuracy: 0.9971 - loss: 0.1693 - precision_14: 0.9975 - recall_14: 0.9971 -
val_accuracy: 0.9569 - val_loss: 0.3318 - val_precision_14: 0.9569 -
val_recall_14: 0.9553 - learning_rate: 0.0010
Epoch 23/50
79/79            52s 658ms/step -
accuracy: 0.9915 - loss: 0.1844 - precision_14: 0.9917 - recall_14: 0.9915 -
val_accuracy: 0.9793 - val_loss: 0.2264 - val_precision_14: 0.9793 -
val_recall_14: 0.9793 - learning_rate: 0.0010
Epoch 24/50
79/79            51s 638ms/step -
accuracy: 0.9907 - loss: 0.1773 - precision_14: 0.9910 - recall_14: 0.9907 -
val_accuracy: 0.9713 - val_loss: 0.2572 - val_precision_14: 0.9713 -
val_recall_14: 0.9713 - learning_rate: 0.0010
Epoch 25/50
79/79            51s 646ms/step -
accuracy: 0.9937 - loss: 0.1741 - precision_14: 0.9947 - recall_14: 0.9937 -
val_accuracy: 0.9697 - val_loss: 0.2787 - val_precision_14: 0.9697 -
val_recall_14: 0.9697 - learning_rate: 0.0010
Epoch 26/50
79/79            51s 639ms/step -
accuracy: 0.9929 - loss: 0.1683 - precision_14: 0.9929 - recall_14: 0.9929 -
val_accuracy: 0.9793 - val_loss: 0.2126 - val_precision_14: 0.9793 -
val_recall_14: 0.9793 - learning_rate: 3.0000e-04
Epoch 27/50
79/79            51s 640ms/step -
accuracy: 0.9969 - loss: 0.1469 - precision_14: 0.9969 - recall_14: 0.9969 -
val_accuracy: 0.9841 - val_loss: 0.2013 - val_precision_14: 0.9841 -
val_recall_14: 0.9841 - learning_rate: 3.0000e-04
Epoch 28/50
```

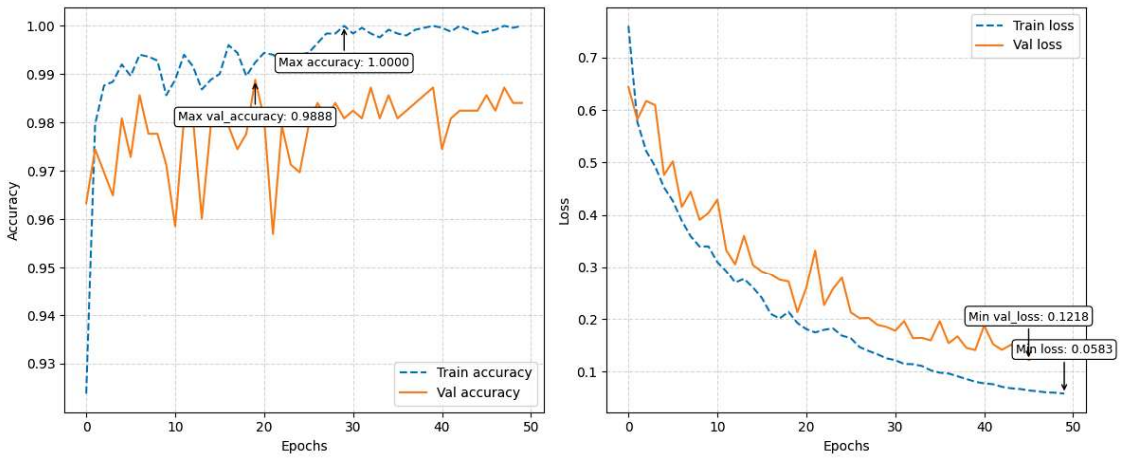
```
79/79            51s 642ms/step -
accuracy: 0.9983 - loss: 0.1398 - precision_14: 0.9983 - recall_14: 0.9983 -
val_accuracy: 0.9809 - val_loss: 0.2020 - val_precision_14: 0.9809 -
val_recall_14: 0.9809 - learning_rate: 3.0000e-04
Epoch 29/50
79/79            51s 640ms/step -
accuracy: 0.9981 - loss: 0.1348 - precision_14: 0.9981 - recall_14: 0.9981 -
val_accuracy: 0.9841 - val_loss: 0.1888 - val_precision_14: 0.9841 -
val_recall_14: 0.9841 - learning_rate: 3.0000e-04
Epoch 30/50
79/79            51s 640ms/step -
accuracy: 1.0000 - loss: 0.1264 - precision_14: 1.0000 - recall_14: 1.0000 -
val_accuracy: 0.9809 - val_loss: 0.1849 - val_precision_14: 0.9809 -
val_recall_14: 0.9809 - learning_rate: 3.0000e-04
Epoch 31/50
79/79            51s 644ms/step -
accuracy: 0.9989 - loss: 0.1224 - precision_14: 0.9989 - recall_14: 0.9989 -
val_accuracy: 0.9825 - val_loss: 0.1777 - val_precision_14: 0.9825 -
val_recall_14: 0.9825 - learning_rate: 3.0000e-04
Epoch 32/50
79/79            51s 651ms/step -
accuracy: 0.9988 - loss: 0.1168 - precision_14: 0.9988 - recall_14: 0.9988 -
val_accuracy: 0.9809 - val_loss: 0.1959 - val_precision_14: 0.9809 -
val_recall_14: 0.9809 - learning_rate: 3.0000e-04
Epoch 33/50
79/79            51s 643ms/step -
accuracy: 0.9987 - loss: 0.1135 - precision_14: 0.9987 - recall_14: 0.9987 -
val_accuracy: 0.9872 - val_loss: 0.1634 - val_precision_14: 0.9872 -
val_recall_14: 0.9872 - learning_rate: 3.0000e-04
Epoch 34/50
79/79            51s 639ms/step -
accuracy: 0.9975 - loss: 0.1110 - precision_14: 0.9975 - recall_14: 0.9975 -
val_accuracy: 0.9809 - val_loss: 0.1641 - val_precision_14: 0.9809 -
val_recall_14: 0.9809 - learning_rate: 3.0000e-04
Epoch 35/50
79/79            51s 642ms/step -
accuracy: 0.9998 - loss: 0.1040 - precision_14: 0.9998 - recall_14: 0.9998 -
val_accuracy: 0.9856 - val_loss: 0.1592 - val_precision_14: 0.9856 -
val_recall_14: 0.9856 - learning_rate: 3.0000e-04
Epoch 36/50
79/79            50s 637ms/step -
accuracy: 0.9986 - loss: 0.0990 - precision_14: 0.9986 - recall_14: 0.9986 -
val_accuracy: 0.9809 - val_loss: 0.1957 - val_precision_14: 0.9809 -
val_recall_14: 0.9809 - learning_rate: 3.0000e-04
Epoch 37/50
79/79            51s 643ms/step -
accuracy: 0.9978 - loss: 0.0990 - precision_14: 0.9978 - recall_14: 0.9976 -
val_accuracy: 0.9825 - val_loss: 0.1539 - val_precision_14: 0.9824 -
```

```
val_recall_14: 0.9809 - learning_rate: 3.0000e-04
Epoch 38/50
79/79            51s 639ms/step -
accuracy: 0.9991 - loss: 0.0921 - precision_14: 0.9991 - recall_14: 0.9991 -
val_accuracy: 0.9841 - val_loss: 0.1669 - val_precision_14: 0.9841 -
val_recall_14: 0.9841 - learning_rate: 3.0000e-04
Epoch 39/50
79/79            51s 642ms/step -
accuracy: 0.9997 - loss: 0.0866 - precision_14: 0.9997 - recall_14: 0.9997 -
val_accuracy: 0.9856 - val_loss: 0.1452 - val_precision_14: 0.9856 -
val_recall_14: 0.9856 - learning_rate: 3.0000e-04
Epoch 40/50
79/79            51s 642ms/step -
accuracy: 1.0000 - loss: 0.0819 - precision_14: 1.0000 - recall_14: 1.0000 -
val_accuracy: 0.9872 - val_loss: 0.1410 - val_precision_14: 0.9872 -
val_recall_14: 0.9872 - learning_rate: 3.0000e-04
Epoch 41/50
79/79            51s 648ms/step -
accuracy: 0.9999 - loss: 0.0780 - precision_14: 0.9999 - recall_14: 0.9999 -
val_accuracy: 0.9745 - val_loss: 0.1876 - val_precision_14: 0.9745 -
val_recall_14: 0.9745 - learning_rate: 3.0000e-04
Epoch 42/50
79/79            51s 647ms/step -
accuracy: 0.9990 - loss: 0.0779 - precision_14: 0.9991 - recall_14: 0.9990 -
val_accuracy: 0.9809 - val_loss: 0.1521 - val_precision_14: 0.9809 -
val_recall_14: 0.9809 - learning_rate: 3.0000e-04
Epoch 43/50
79/79            51s 640ms/step -
accuracy: 1.0000 - loss: 0.0724 - precision_14: 1.0000 - recall_14: 1.0000 -
val_accuracy: 0.9825 - val_loss: 0.1414 - val_precision_14: 0.9825 -
val_recall_14: 0.9825 - learning_rate: 3.0000e-04
Epoch 44/50
79/79            51s 639ms/step -
accuracy: 0.9993 - loss: 0.0688 - precision_14: 0.9993 - recall_14: 0.9993 -
val_accuracy: 0.9825 - val_loss: 0.1509 - val_precision_14: 0.9825 -
val_recall_14: 0.9825 - learning_rate: 3.0000e-04
Epoch 45/50
79/79            51s 638ms/step -
accuracy: 0.9989 - loss: 0.0663 - precision_14: 0.9989 - recall_14: 0.9989 -
val_accuracy: 0.9825 - val_loss: 0.1417 - val_precision_14: 0.9825 -
val_recall_14: 0.9825 - learning_rate: 3.0000e-04
Epoch 46/50
79/79            51s 643ms/step -
accuracy: 0.9986 - loss: 0.0646 - precision_14: 0.9986 - recall_14: 0.9986 -
val_accuracy: 0.9856 - val_loss: 0.1218 - val_precision_14: 0.9856 -
val_recall_14: 0.9856 - learning_rate: 9.0000e-05
Epoch 47/50
79/79            51s 639ms/step -
```

```
accuracy: 0.9995 - loss: 0.0631 - precision_14: 0.9995 - recall_14: 0.9995 -
val_accuracy: 0.9825 - val_loss: 0.1301 - val_precision_14: 0.9825 -
val_recall_14: 0.9825 - learning_rate: 9.0000e-05
Epoch 48/50
79/79              51s 646ms/step -
accuracy: 1.0000 - loss: 0.0611 - precision_14: 1.0000 - recall_14: 1.0000 -
val_accuracy: 0.9872 - val_loss: 0.1284 - val_precision_14: 0.9872 -
val_recall_14: 0.9872 - learning_rate: 9.0000e-05
Epoch 49/50
79/79              52s 658ms/step -
accuracy: 0.9996 - loss: 0.0604 - precision_14: 1.0000 - recall_14: 0.9996 -
val_accuracy: 0.9841 - val_loss: 0.1274 - val_precision_14: 0.9841 -
val_recall_14: 0.9841 - learning_rate: 9.0000e-05
Epoch 50/50
79/79              51s 641ms/step -
accuracy: 1.0000 - loss: 0.0584 - precision_14: 1.0000 - recall_14: 1.0000 -
val_accuracy: 0.9841 - val_loss: 0.1238 - val_precision_14: 0.9841 -
val_recall_14: 0.9841 - learning_rate: 9.0000e-05
```

6.7.3 Plot loss & accuracy for training and validation wrt epoch

[170]:
```
plot_loss_and_accuracy(["accuracy","loss"],model_fit)
```



6.7.4 Tensorboard

[171]:
```
%load_ext tensorboard
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

6.7.5 Model Evaluation, Mlflow Logging and Printing Metrics

[172]:
```
ckpt_path
```

[172]:
```
'checkpoints/1_pretrained_mobilenetv2_trainable0.weights.h5'
```

[173]:
```
log_dir
```

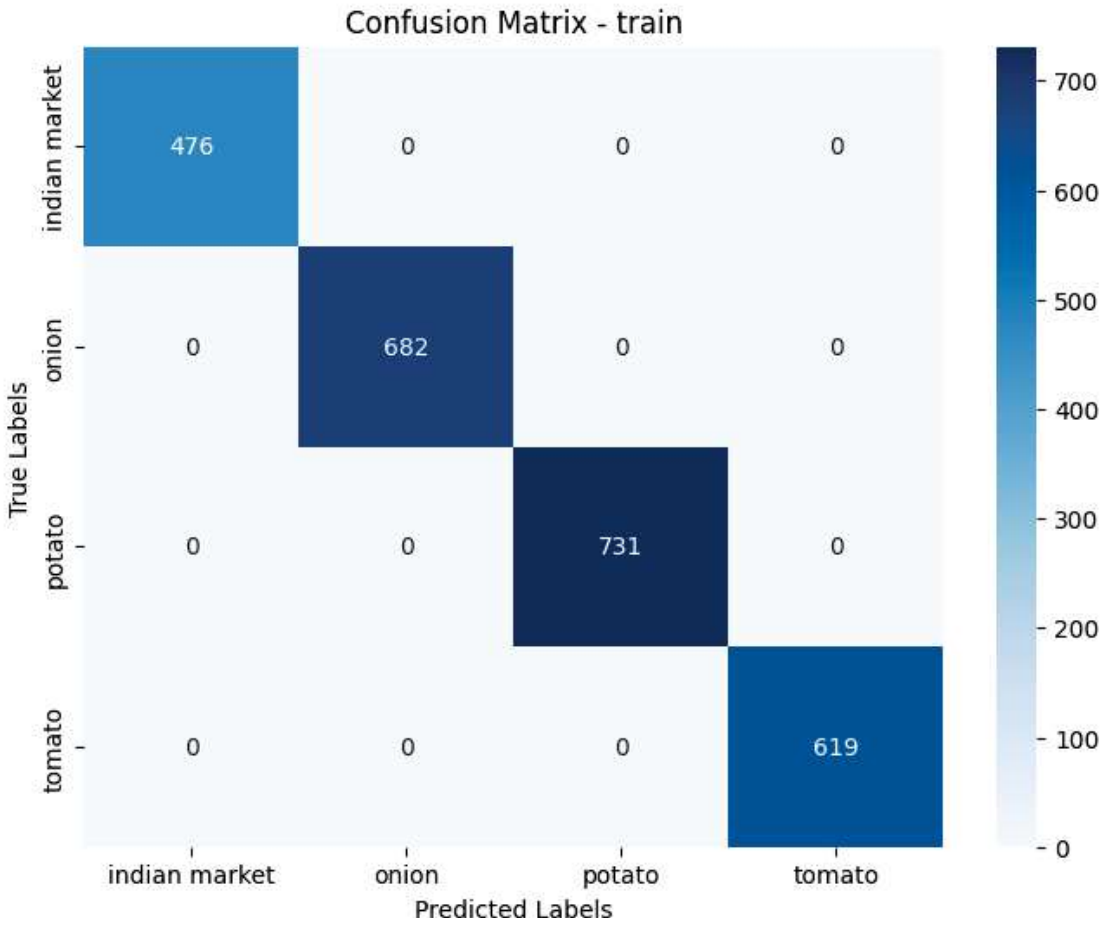
[173]:
```
'logs/1_pretrained_mobilenetv2_trainable0'
```

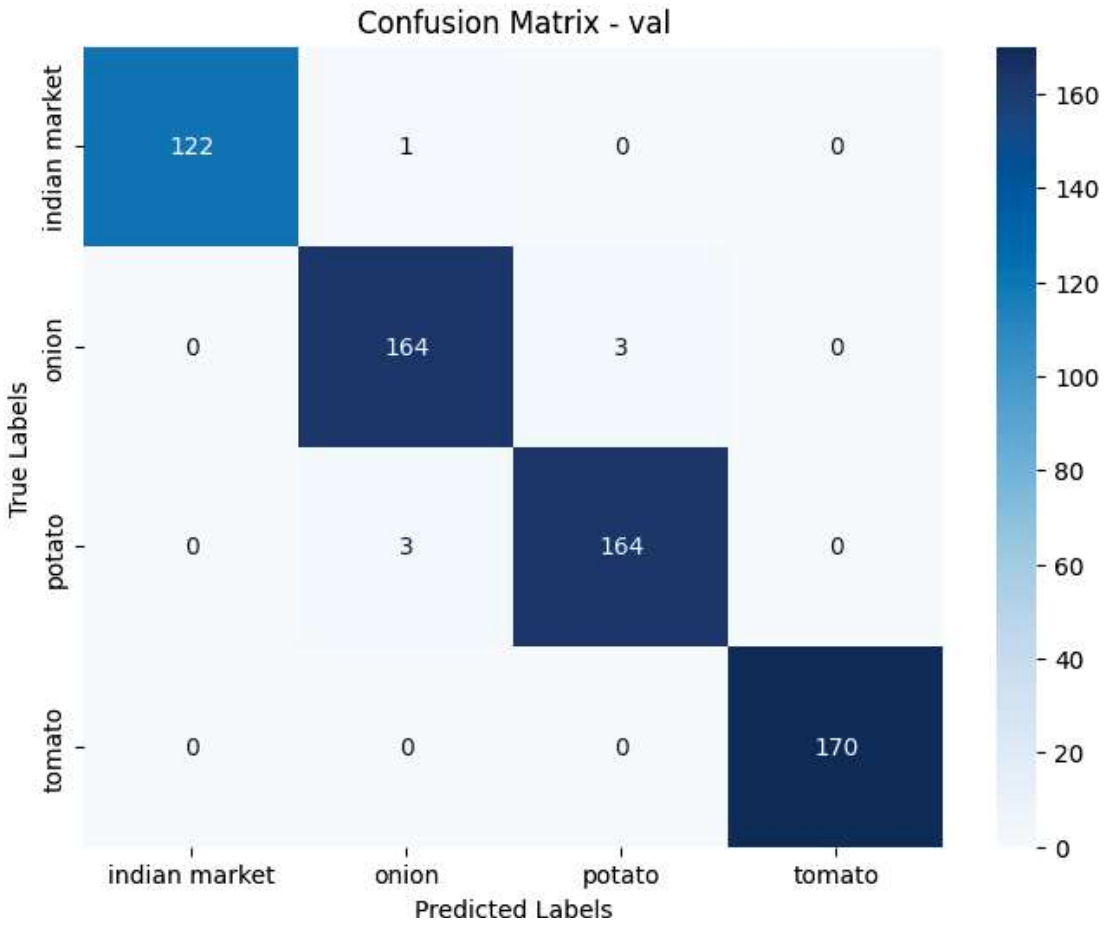
[174]:
```
run_name = "1_pretrained_mobilenetv2_trainable0"
run_name
```

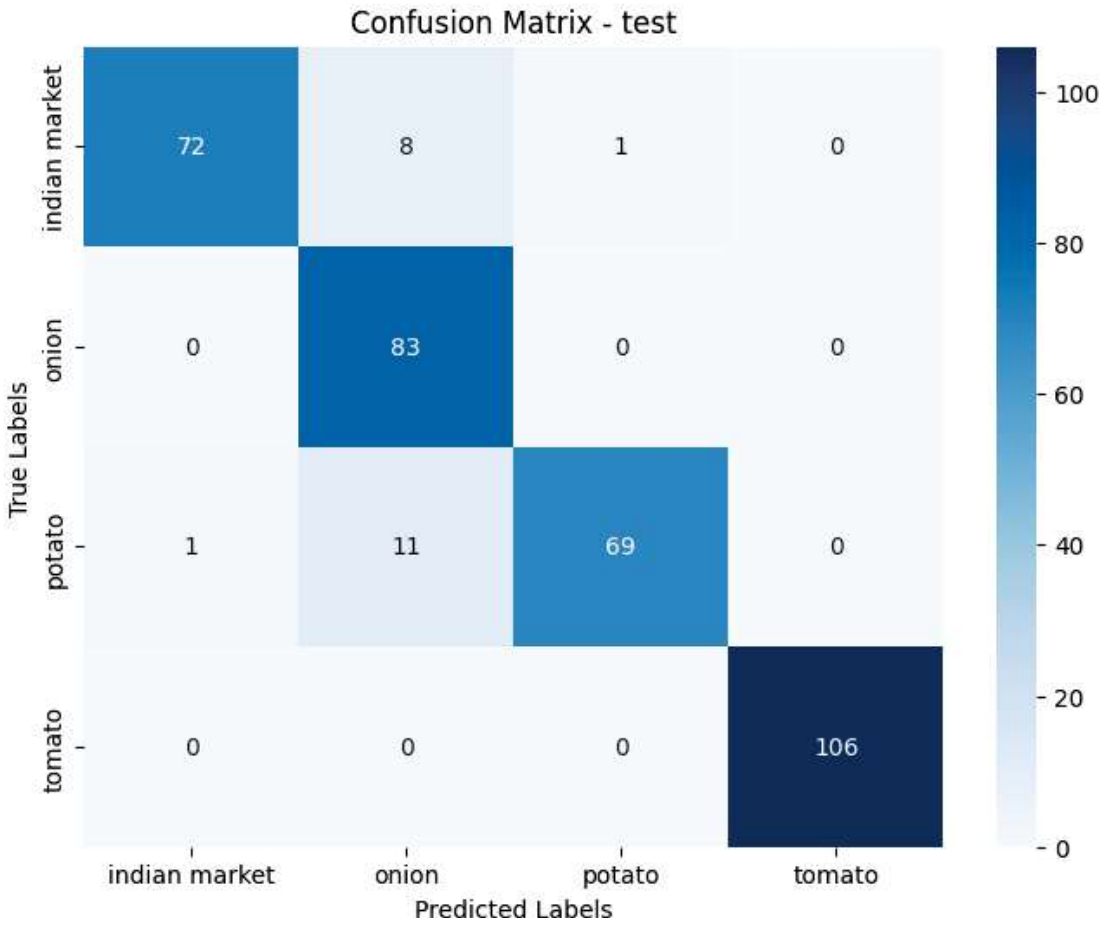
[174]:
```
'1_pretrained_mobilenetv2_trainable0'
```

[175]:
```
evaluate_model(model, train_ds, val_ds, test_ds,␣
↪class_names,run_name=run_name,ckpt_path= ckpt_path)
```

```
2025/02/11 03:19:39 WARNING mlflow.tensorflow: You are saving a TensorFlow Core
model or Keras model without a signature. Inference with
mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation
accepts pandas DataFrames as inference inputs.
2025/02/11 03:19:50 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.


train Metrics:
 Accuracy: 100.00%
 Precision: 100.00%
 Recall: 100.00%

val Metrics:
 Accuracy: 98.88%
 Precision: 98.96%
 Recall: 98.90%

test Metrics:
 Accuracy: 94.02%
 Precision: 94.64%
 Recall: 93.52%
```


Confusion Matrix - train
True Labels
Predicted Labels
indian market
onion
potato
tomato
476
0
0
0
0
682
0
0
0
0
731
0
0
0
0
619
700
600
500
400
300
200
100
0


Confusion Matrix - val
True Labels
Predicted Labels
indian market
onion
potato
tomato
122
1
0
0
0
164
3
0
0
3
164
0
0
0
0
170
160
140
120
100
80
60
40
20
0




```
Classification Report - train
               precision    recall  f1-score   support

indian market       1.00      1.00      1.00       476
        onion       1.00      1.00      1.00       682
       potato       1.00      1.00      1.00       731
       tomato       1.00      1.00      1.00       619

     accuracy                           1.00      2508
    macro avg       1.00      1.00      1.00      2508
 weighted avg       1.00      1.00      1.00      2508


Classification Report - val
               precision    recall  f1-score   support

indian market       1.00      0.99      1.00       123
```

```
        onion       0.98      0.98      0.98       167
       potato       0.98      0.98      0.98       167
       tomato       1.00      1.00      1.00       170

     accuracy                           0.99       627
    macro avg       0.99      0.99      0.99       627
 weighted avg       0.99      0.99      0.99       627


Classification Report - test
               precision    recall  f1-score   support

indian market       0.99      0.89      0.94        81
        onion       0.81      1.00      0.90        83
       potato       0.99      0.85      0.91        81
       tomato       1.00      1.00      1.00       106

     accuracy                           0.94       351
    macro avg       0.95      0.94      0.94       351
 weighted avg       0.95      0.94      0.94       351


Class-wise Accuracy:

Train Class-wise Accuracy:
 indian market: 100.00% (476/476)
 onion: 100.00% (682/682)
 potato: 100.00% (731/731)
 tomato: 100.00% (619/619)

Val Class-wise Accuracy:
 indian market: 99.19% (122/123)
 onion: 98.20% (164/167)
 potato: 98.20% (164/167)
 tomato: 100.00% (170/170)

Test Class-wise Accuracy:
 indian market: 88.89% (72/81)
 onion: 100.00% (83/83)
 potato: 85.19% (69/81)
 tomato: 100.00% (106/106)

Random Test Predictions:
```

True: indian market
Pred: indian market

True: indian market
Pred: indian market

True: onion
Pred: onion

True: onion
Pred: onion

True: potato
Pred: potato

True: potato
Pred: onion

True: tomato
Pred: tomato

True: tomato
Pred: tomato

6.7.6 Save the model

[176]:
```
model.save("saved_models/1_pretrained_mobilenetv2_trainable0_model.keras")
```

6.8 DATA AUGMENTED RESNET50

6.8.1 Load the previously trained model architecture and weights

[177]:
```
# Load the entire model (architecture + weights)
model = keras.models.load_model("saved_models/
 ↪1_pretrained_mobilenetv2_trainable0_model.keras")
```

[178]:
```
model.summary()
```

Model: "pretrained_mobilenetv2"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 8, 8, 1280)	2,257,984
global_average_pooling2d_13 (GlobalAveragePooling2D)	(None, 1280)	0
dense_39 (Dense)	(None, 512)	655,872
batch_normalization_112 (BatchNormalization)	(None, 512)	2,048
dropout_15 (Dropout)	(None, 512)	0
dense_40 (Dense)	(None, 256)	131,328
batch_normalization_113 (BatchNormalization)	(None, 256)	1,024
dense_41 (Dense)	(None, 4)	1,028

Total params: 4,628,814 (17.66 MB)

Trainable params: 789,764 (3.01 MB)

Non-trainable params: 2,259,520 (8.62 MB)

```
Optimizer params: 1,579,530 (6.03 MB)
```

6.8.2 Model Compilation and Training

[179]:
```
ckpt_path = "checkpoints/1_pretrained_mobilenetv2_aug_trainable0.weights.h5"
```

[180]:
```
log_dir = 'logs/1_pretrained_mobilenetv2_aug_trainable0'
```

[181]:
```
run_name = '1_pretrained_mobilenetv2_aug_trainable0'
```

[182]:
```
model_fit = compile_train(model, train_aug_ds, val_aug_ds,␣
↪log_dir=log_dir,epochs=20,ckpt_path=ckpt_path)
```

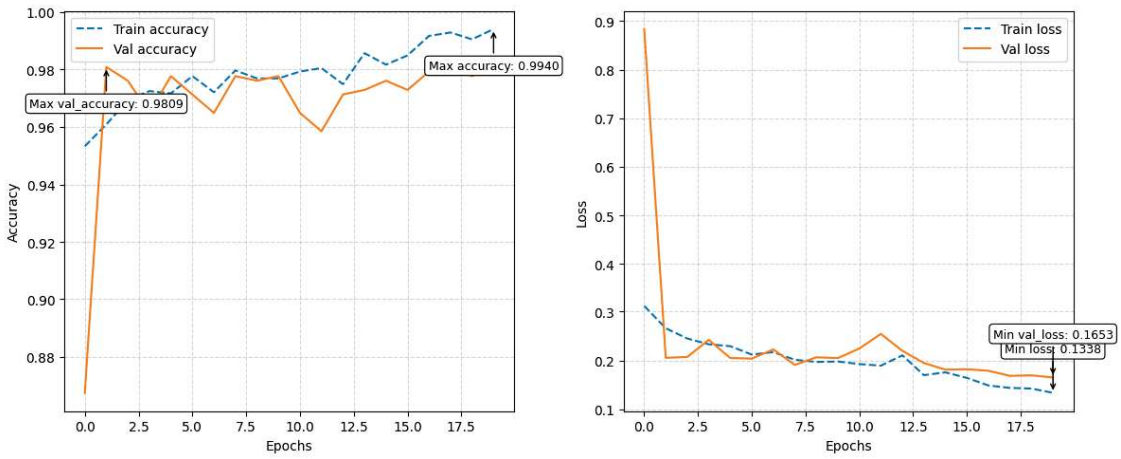
```
Epoch 1/20
79/79             59s 680ms/step -
accuracy: 0.9526 - loss: 0.3458 - precision_15: 0.9547 - recall_15: 0.9526 -
val_accuracy: 0.8676 - val_loss: 0.8836 - val_precision_15: 0.8676 -
val_recall_15: 0.8676 - learning_rate: 0.0010
Epoch 2/20
79/79             54s 680ms/step -
accuracy: 0.9604 - loss: 0.2680 - precision_15: 0.9631 - recall_15: 0.9577 -
val_accuracy: 0.9809 - val_loss: 0.2056 - val_precision_15: 0.9824 -
val_recall_15: 0.9809 - learning_rate: 0.0010
Epoch 3/20
79/79             53s 666ms/step -
accuracy: 0.9720 - loss: 0.2435 - precision_15: 0.9728 - recall_15: 0.9696 -
val_accuracy: 0.9761 - val_loss: 0.2073 - val_precision_15: 0.9760 -
val_recall_15: 0.9745 - learning_rate: 0.0010
Epoch 4/20
79/79             53s 665ms/step -
accuracy: 0.9750 - loss: 0.2250 - precision_15: 0.9750 - recall_15: 0.9748 -
val_accuracy: 0.9649 - val_loss: 0.2423 - val_precision_15: 0.9664 -
val_recall_15: 0.9633 - learning_rate: 0.0010
Epoch 5/20
79/79             53s 667ms/step -
accuracy: 0.9698 - loss: 0.2389 - precision_15: 0.9710 - recall_15: 0.9689 -
val_accuracy: 0.9777 - val_loss: 0.2052 - val_precision_15: 0.9792 -
val_recall_15: 0.9777 - learning_rate: 0.0010
Epoch 6/20
79/79             53s 659ms/step -
accuracy: 0.9786 - loss: 0.2179 - precision_15: 0.9785 - recall_15: 0.9776 -
val_accuracy: 0.9713 - val_loss: 0.2039 - val_precision_15: 0.9712 -
val_recall_15: 0.9697 - learning_rate: 0.0010
Epoch 7/20
79/79             53s 660ms/step -
accuracy: 0.9744 - loss: 0.2138 - precision_15: 0.9749 - recall_15: 0.9743 -
val_accuracy: 0.9649 - val_loss: 0.2226 - val_precision_15: 0.9665 -
```

```
val_recall_15: 0.9649 - learning_rate: 0.0010
Epoch 8/20
79/79            53s 668ms/step -
accuracy: 0.9752 - loss: 0.2198 - precision_15: 0.9773 - recall_15: 0.9752 -
val_accuracy: 0.9777 - val_loss: 0.1910 - val_precision_15: 0.9792 -
val_recall_15: 0.9777 - learning_rate: 0.0010
Epoch 9/20
79/79            53s 659ms/step -
accuracy: 0.9756 - loss: 0.1932 - precision_15: 0.9780 - recall_15: 0.9754 -
val_accuracy: 0.9761 - val_loss: 0.2066 - val_precision_15: 0.9761 -
val_recall_15: 0.9761 - learning_rate: 0.0010
Epoch 10/20
79/79            53s 661ms/step -
accuracy: 0.9800 - loss: 0.1960 - precision_15: 0.9807 - recall_15: 0.9792 -
val_accuracy: 0.9777 - val_loss: 0.2050 - val_precision_15: 0.9777 -
val_recall_15: 0.9777 - learning_rate: 0.0010
Epoch 11/20
79/79            53s 665ms/step -
accuracy: 0.9805 - loss: 0.1929 - precision_15: 0.9810 - recall_15: 0.9805 -
val_accuracy: 0.9649 - val_loss: 0.2246 - val_precision_15: 0.9665 -
val_recall_15: 0.9649 - learning_rate: 0.0010
Epoch 12/20
79/79            53s 658ms/step -
accuracy: 0.9849 - loss: 0.1721 - precision_15: 0.9861 - recall_15: 0.9837 -
val_accuracy: 0.9585 - val_loss: 0.2545 - val_precision_15: 0.9600 -
val_recall_15: 0.9569 - learning_rate: 0.0010
Epoch 13/20
79/79            53s 659ms/step -
accuracy: 0.9700 - loss: 0.2231 - precision_15: 0.9708 - recall_15: 0.9674 -
val_accuracy: 0.9713 - val_loss: 0.2199 - val_precision_15: 0.9713 -
val_recall_15: 0.9713 - learning_rate: 0.0010
Epoch 14/20
79/79            53s 659ms/step -
accuracy: 0.9876 - loss: 0.1686 - precision_15: 0.9880 - recall_15: 0.9873 -
val_accuracy: 0.9729 - val_loss: 0.1949 - val_precision_15: 0.9729 -
val_recall_15: 0.9729 - learning_rate: 3.0000e-04
Epoch 15/20
79/79            54s 673ms/step -
accuracy: 0.9792 - loss: 0.1803 - precision_15: 0.9812 - recall_15: 0.9788 -
val_accuracy: 0.9761 - val_loss: 0.1813 - val_precision_15: 0.9792 -
val_recall_15: 0.9761 - learning_rate: 3.0000e-04
Epoch 16/20
79/79            53s 666ms/step -
accuracy: 0.9835 - loss: 0.1714 - precision_15: 0.9837 - recall_15: 0.9835 -
val_accuracy: 0.9729 - val_loss: 0.1821 - val_precision_15: 0.9729 -
val_recall_15: 0.9729 - learning_rate: 3.0000e-04
Epoch 17/20
79/79            53s 660ms/step -
```

```
accuracy: 0.9891 - loss: 0.1504 - precision_15: 0.9893 - recall_15: 0.9888 -
val_accuracy: 0.9793 - val_loss: 0.1790 - val_precision_15: 0.9793 -
val_recall_15: 0.9793 - learning_rate: 3.0000e-04
Epoch 18/20
79/79              53s 660ms/step -
accuracy: 0.9923 - loss: 0.1467 - precision_15: 0.9923 - recall_15: 0.9923 -
val_accuracy: 0.9809 - val_loss: 0.1686 - val_precision_15: 0.9809 -
val_recall_15: 0.9809 - learning_rate: 3.0000e-04
Epoch 19/20
79/79              53s 658ms/step -
accuracy: 0.9881 - loss: 0.1483 - precision_15: 0.9882 - recall_15: 0.9881 -
val_accuracy: 0.9777 - val_loss: 0.1697 - val_precision_15: 0.9792 -
val_recall_15: 0.9777 - learning_rate: 3.0000e-04
Epoch 20/20
79/79              53s 660ms/step -
accuracy: 0.9945 - loss: 0.1344 - precision_15: 0.9944 - recall_15: 0.9942 -
val_accuracy: 0.9809 - val_loss: 0.1653 - val_precision_15: 0.9808 -
val_recall_15: 0.9793 - learning_rate: 3.0000e-04
```

6.8.3 Plot loss & accuracy for training and validation wrt epoch

[183]:
```
plot_loss_and_accuracy(["accuracy","loss"],model_fit)
```



6.8.4 Tensorboard

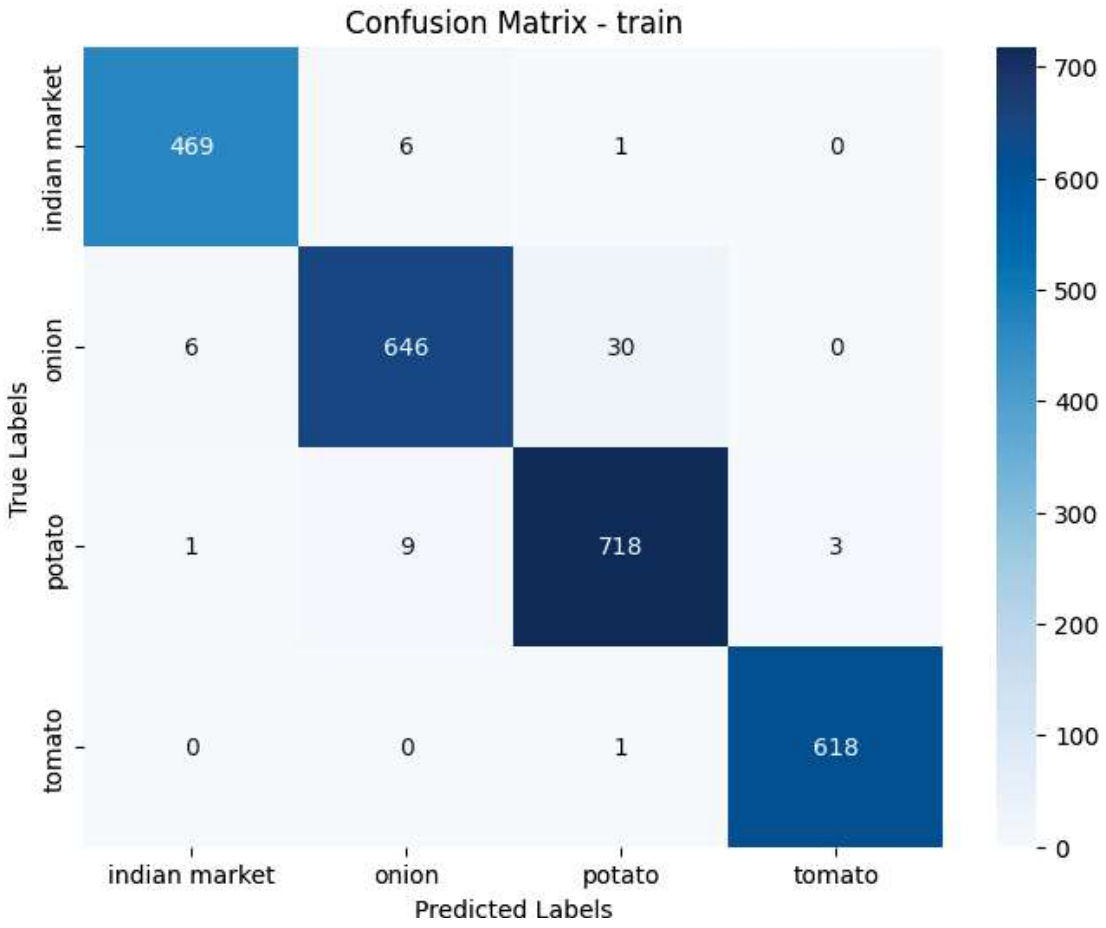
[184]:
```
%load_ext tensorboard
```

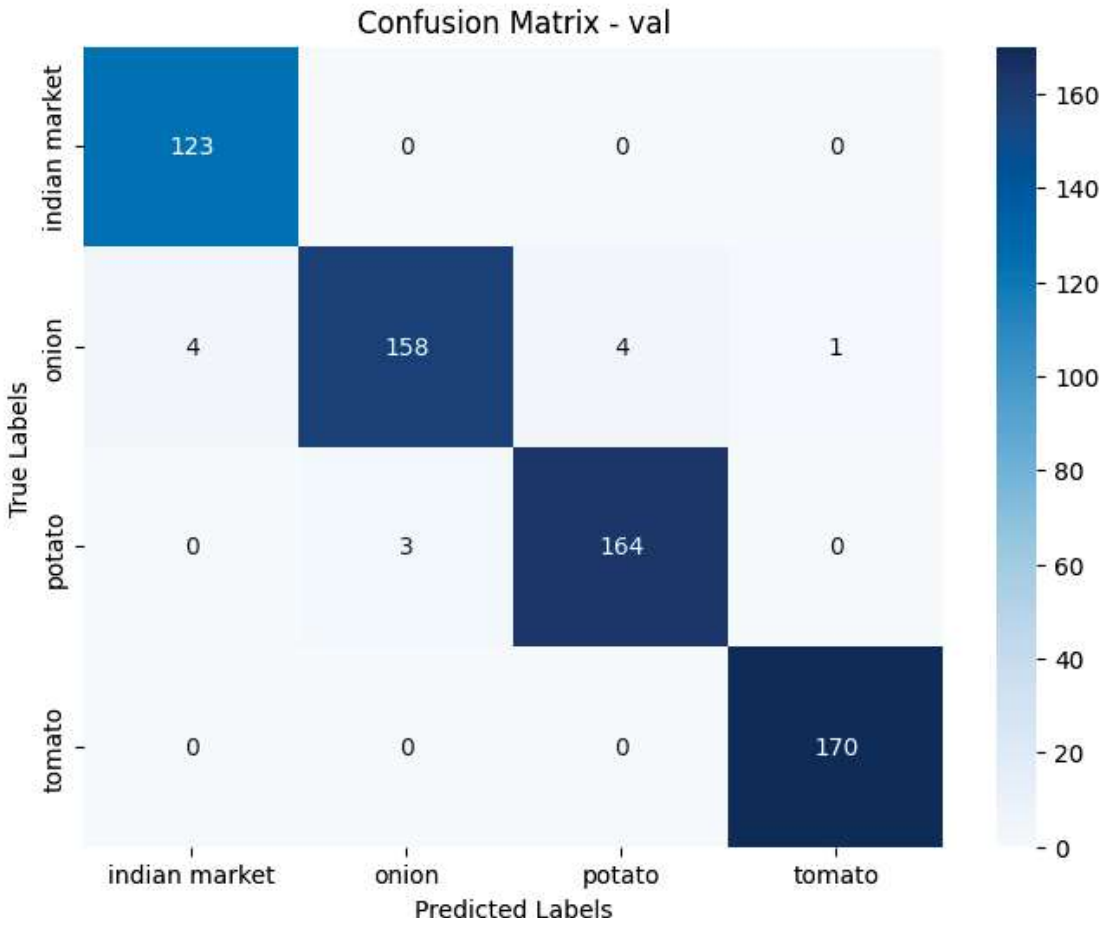
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

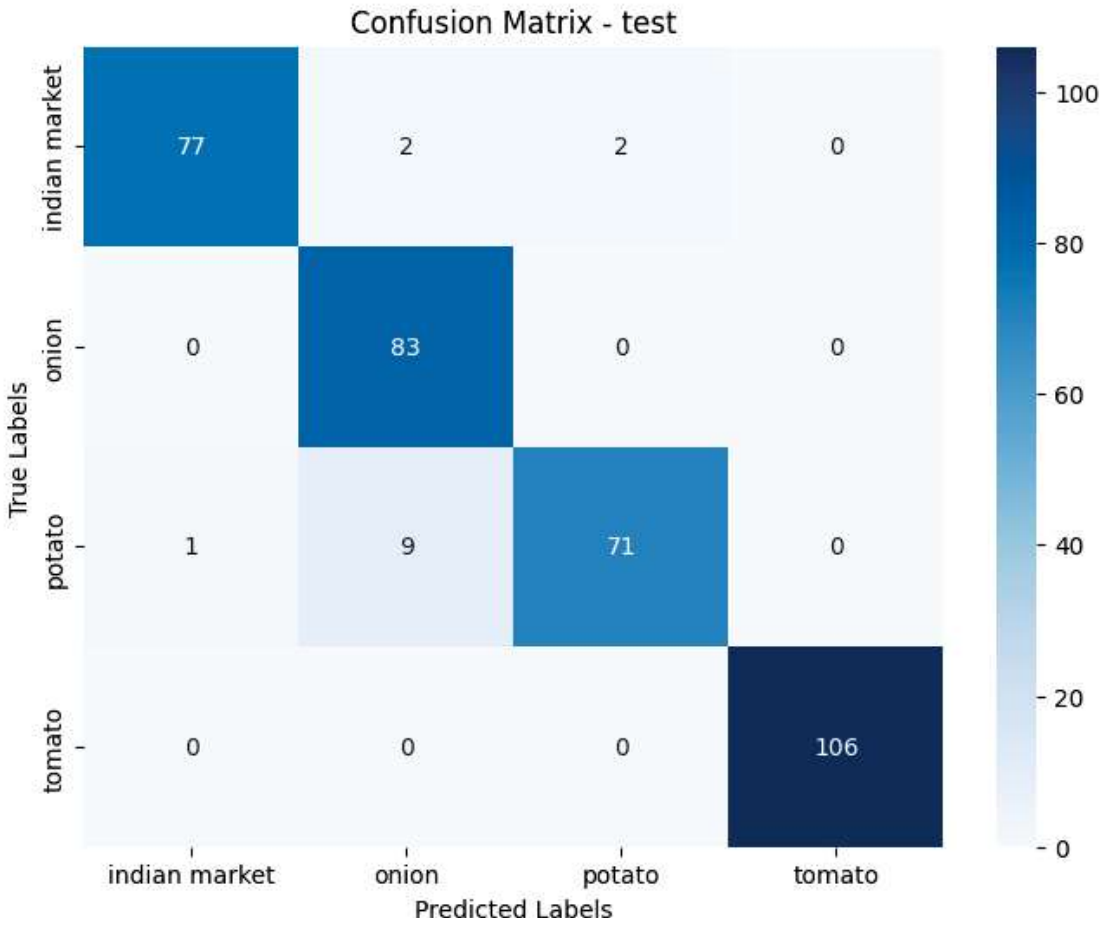
6.8.5 Model Evaluation, Mlflow Logging and Printing Metrics

[185]:
```
evaluate_model(model, train_aug_ds, val_aug_ds, test_aug_ds,␣
↪class_names,run_name=run_name,ckpt_path= ckpt_path)
```

```
2025/02/11 03:39:03 WARNING mlflow.tensorflow: You are saving a TensorFlow Core
model or Keras model without a signature. Inference with
mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation
accepts pandas DataFrames as inference inputs.
2025/02/11 03:39:14 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.


train Metrics:
 Accuracy: 97.73%
 Precision: 97.88%
 Recall: 97.83%

val Metrics:
 Accuracy: 98.09%
 Precision: 98.01%
 Recall: 98.20%

test Metrics:
 Accuracy: 96.01%
 Precision: 96.07%
 Recall: 95.68%
```


Confusion Matrix - train
True Labels
Predicted Labels
indian market
onion
potato
tomato
469
6
1
0
6
646
30
0
1
9
718
3
0
0
1
618
700
600
500
400
300
200
100
0


Confusion Matrix - val
indian market
onion
potato
tomato
True Labels
123
0
0
0
4
158
4
1
0
3
164
0
0
0
0
170
indian market
onion
potato
tomato
Predicted Labels
160
140
120
100
80
60
40
20
0




```
Classification Report - train
              precision    recall  f1-score   support

indian market       0.99      0.99      0.99       476
        onion       0.98      0.95      0.96       682
       potato       0.96      0.98      0.97       731
       tomato       1.00      1.00      1.00       619

     accuracy                           0.98      2508
    macro avg       0.98      0.98      0.98      2508
 weighted avg       0.98      0.98      0.98      2508


Classification Report - val
              precision    recall  f1-score   support

indian market       0.97      1.00      0.98       123
```

```
        onion       0.98      0.95      0.96       167
       potato       0.98      0.98      0.98       167
       tomato       0.99      1.00      1.00       170

     accuracy                           0.98       627
    macro avg       0.98      0.98      0.98       627
 weighted avg       0.98      0.98      0.98       627


Classification Report - test
               precision    recall  f1-score   support

indian market       0.99      0.95      0.97        81
        onion       0.88      1.00      0.94        83
       potato       0.97      0.88      0.92        81
       tomato       1.00      1.00      1.00       106

     accuracy                           0.96       351
    macro avg       0.96      0.96      0.96       351
 weighted avg       0.96      0.96      0.96       351


Class-wise Accuracy:

Train Class-wise Accuracy:
 indian market: 98.53% (469/476)
 onion: 94.72% (646/682)
 potato: 98.22% (718/731)
 tomato: 99.84% (618/619)

Val Class-wise Accuracy:
 indian market: 100.00% (123/123)
 onion: 94.61% (158/167)
 potato: 98.20% (164/167)
 tomato: 100.00% (170/170)

Test Class-wise Accuracy:
 indian market: 95.06% (77/81)
 onion: 100.00% (83/83)
 potato: 87.65% (71/81)
 tomato: 100.00% (106/106)

Random Test Predictions:
```

True: indian market
Pred: indian market

True: indian market
Pred: indian market

True: onion
Pred: onion

True: onion
Pred: onion

True: potato
Pred: potato

True: potato
Pred: potato

True: tomato
Pred: tomato

True: tomato
Pred: tomato

6.8.6 Save the model

[186]:
```
model.save("saved_models/1_pretrained_mobilenetv2_aug_trainable0_model.keras")
```

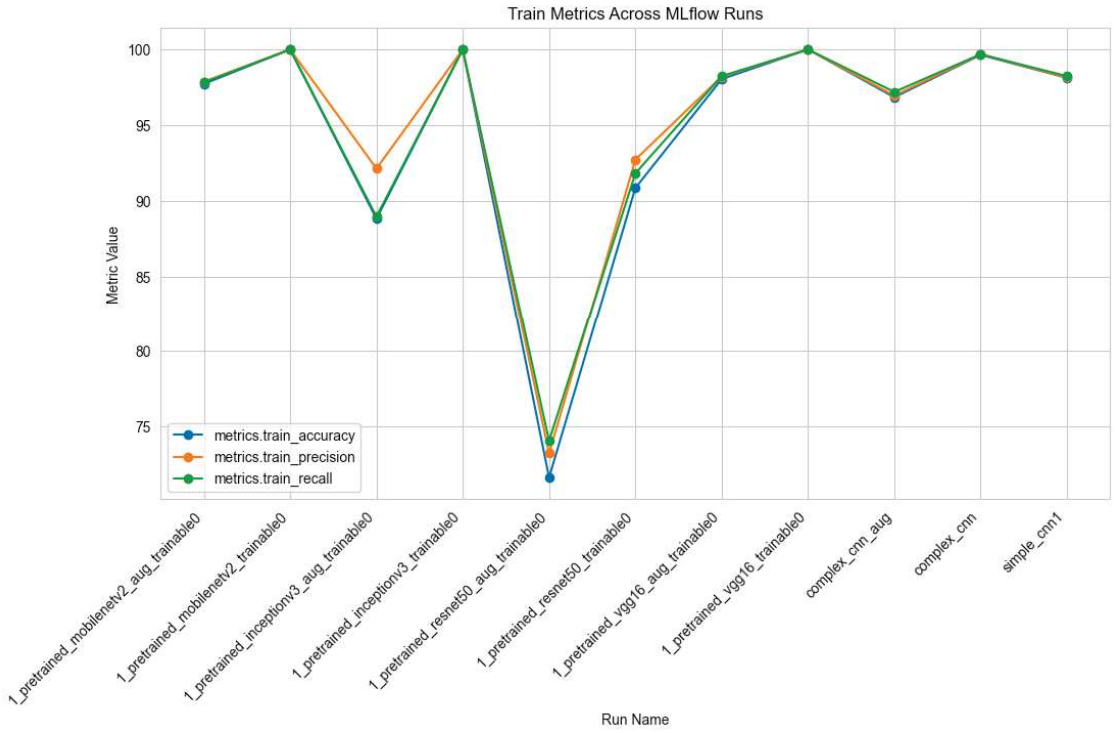
7 MODEL COMPARISONS

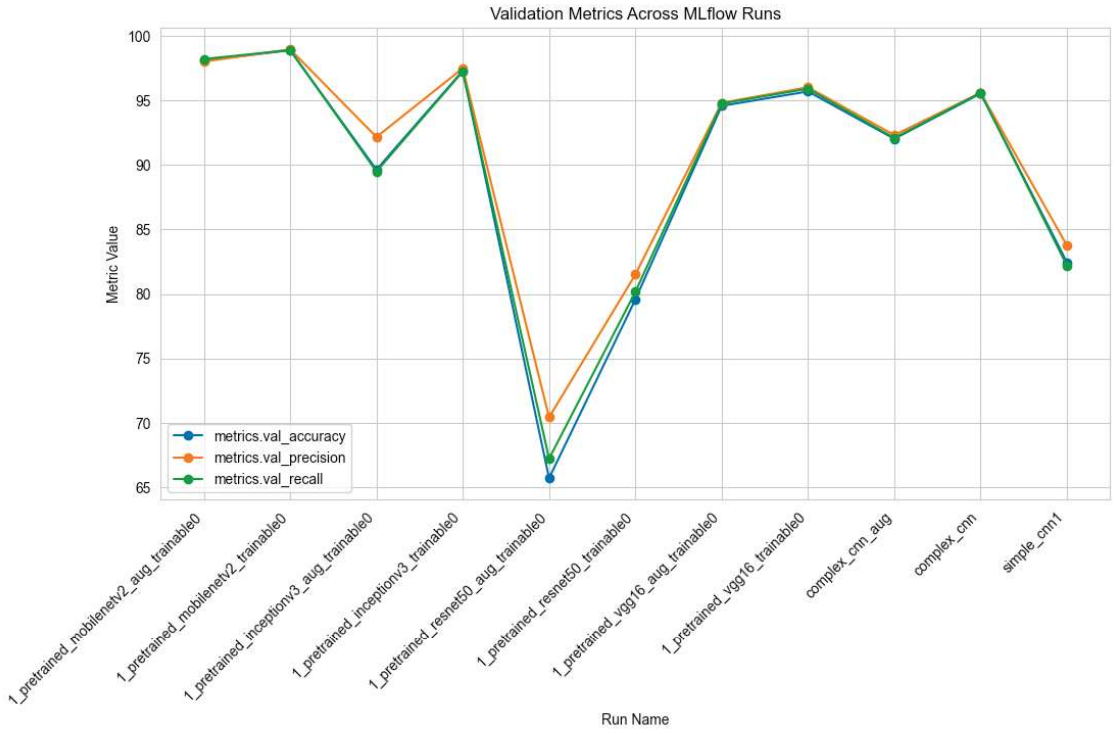
[188]:
```
# Define experiment name
experiment_name = "Vegetable_image_classification"
experiment = mlflow.get_experiment_by_name(experiment_name)

# Get all runs
runs_df = mlflow.search_runs(experiment_ids=[experiment.experiment_id])

# Print all available columns to check metric names
print("Available Columns:", runs_df.columns.tolist())

# Identify metric categories
train_metrics = [col for col in runs_df.columns if "train" in col.lower()]
val_metrics = [col for col in runs_df.columns if "val" in col.lower()]
test_metrics = [col for col in runs_df.columns if "test" in col.lower()]

# Print detected metrics
print("Train Metrics:", train_metrics)
print("Validation Metrics:", val_metrics)
print("Test Metrics:", test_metrics)

# Check if "run_name" exists, else use "run_id"
run_identifier = "tags.mlflow.runName" if "tags.mlflow.runName" in runs_df.
 ↪columns else "run_id"

# Extract relevant metrics with proper indexing to avoid SettingWithCopyWarning
train_df = runs_df[[run_identifier] + train_metrics].copy()
val_df = runs_df[[run_identifier] + val_metrics].copy()
test_df = runs_df[[run_identifier] + test_metrics].copy()

# Rename run column for clarity
train_df.rename(columns={run_identifier: "Run"}, inplace=True)
val_df.rename(columns={run_identifier: "Run"}, inplace=True)
test_df.rename(columns={run_identifier: "Run"}, inplace=True)

# Convert to numeric (MLflow sometimes stores metrics as strings)
for df, metric_list in zip([train_df, val_df, test_df], [train_metrics,␣
 ↪val_metrics, test_metrics]):
    df.loc[:, metric_list] = df.loc[:, metric_list].apply(pd.to_numeric,␣
 ↪errors='coerce')

# Function to plot metrics
```

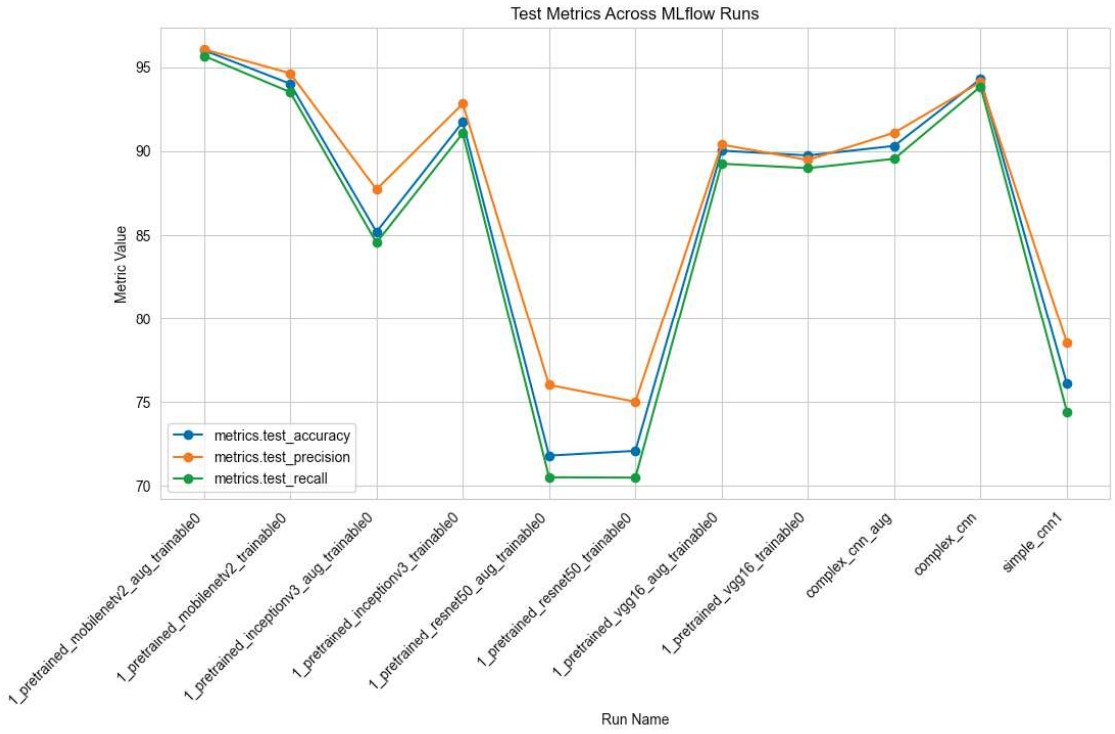
```
def plot_metrics(df, metric_list, title):
    sns.set_style("whitegrid")
    plt.figure(figsize=(12, 6))

    for metric in metric_list:
        plt.plot(df["Run"], df[metric], marker='o', label=metric)

    plt.xticks(rotation=45, ha="right")
    plt.xlabel("Run Name")
    plt.ylabel("Metric Value")
    plt.title(title)
    plt.legend()
    plt.show()

# Plot Train, Validation, and Test metrics separately
plot_metrics(train_df, ["metrics.train_accuracy", "metrics.train_precision",␣
 ↪"metrics.train_recall"], "Train Metrics Across MLflow Runs")
plot_metrics(val_df, ["metrics.val_accuracy", "metrics.val_precision", "metrics.
 ↪val_recall"], "Validation Metrics Across MLflow Runs")
plot_metrics(test_df, ["metrics.test_accuracy", "metrics.test_precision",␣
 ↪"metrics.test_recall"], "Test Metrics Across MLflow Runs")
```

```
Available Columns: ['run_id', 'experiment_id', 'status', 'artifact_uri',
'start_time', 'end_time', 'metrics.test_recall', 'metrics.train_recall',
'metrics.train_class_accuracy_indian market',
'metrics.train_class_accuracy_potato', 'metrics.train_class_accuracy_onion',
'metrics.val_accuracy', 'metrics.train_precision', 'metrics.test_precision',
'metrics.test_accuracy', 'metrics.test_class_accuracy_potato',
'metrics.val_class_accuracy_tomato', 'metrics.train_class_accuracy_tomato',
'metrics.test_class_accuracy_onion', 'metrics.test_class_accuracy_tomato',
'metrics.val_recall', 'metrics.val_precision',
'metrics.val_class_accuracy_indian market', 'metrics.test_class_accuracy_indian
market', 'metrics.train_accuracy', 'metrics.val_class_accuracy_potato',
'metrics.val_class_accuracy_onion', 'params.checkpoint_path',
'tags.mlflow.runName', 'tags.mlflow.source.type', 'tags.mlflow.log-
model.history', 'tags.mlflow.source.name', 'tags.mlflow.user']
Train Metrics: ['metrics.train_recall', 'metrics.train_class_accuracy_indian
market', 'metrics.train_class_accuracy_potato',
'metrics.train_class_accuracy_onion', 'metrics.train_precision',
'metrics.train_class_accuracy_tomato', 'metrics.train_accuracy']
Validation Metrics: ['metrics.val_accuracy',
'metrics.val_class_accuracy_tomato', 'metrics.val_recall',
'metrics.val_precision', 'metrics.val_class_accuracy_indian market',
'metrics.val_class_accuracy_potato', 'metrics.val_class_accuracy_onion']
Test Metrics: ['metrics.test_recall', 'metrics.test_precision',
'metrics.test_accuracy', 'metrics.test_class_accuracy_potato',
'metrics.test_class_accuracy_onion', 'metrics.test_class_accuracy_tomato',
```

```
'metrics.test_class_accuracy_indian market']
```






Test Metrics Across MLflow Runs
Metric Value
95
90
85
80
75
70
metrics.test_accuracy
metrics.test_precision
metrics.test_recall
1_pretrained_mobilenetv2_aug_trainable0
1_pretrained_mobilenetv2_trainable0
1_pretrained_inceptionv3_aug_trainable0
1_pretrained_inceptionv3_trainable0
1_pretrained_resnet50_aug_trainable0
1_pretrained_resnet50_trainable0
1_pretrained_vgg16_aug_trainable0
1_pretrained_vgg16_trainable0
complex_cnn_aug
complex_cnn
simple_cnn1
Run Name