

# Training

August 4, 2024

```
[ ]: import os
import shutil
from pathlib import Path

# Define the base directory
base_dir = 'app/GroceryStoreDataset/dataset/test'

# Define the directories for each class
fruit_dir = os.path.join(base_dir, 'Fruit')
packages_dir = os.path.join(base_dir, 'Packages')
vegetables_dir = os.path.join(base_dir, 'Vegetables')

# List of source directories
source_dirs = [fruit_dir, packages_dir, vegetables_dir]

# Define the combined directory
combined_dir = 'app/GroceryStoreDataset/dataset/test-1'
Path(combined_dir).mkdir(parents=True, exist_ok=True) # Create the destination_
↳directory if it doesn't exist

def combine_directories(source_dirs, dest_dir):
    for src_dir in source_dirs:
        for root, dirs, files in os.walk(src_dir):
            for file in files:
                # Construct the full file path
                file_path = os.path.join(root, file)
                # Get the class name (subdirectory name)
                class_name = os.path.basename(os.path.dirname(file_path))
                # Create the class directory in the destination if it doesn't_
↳exist

                dest_class_dir = os.path.join(dest_dir, class_name)
                Path(dest_class_dir).mkdir(parents=True, exist_ok=True)
                # Copy file to the destination class directory
                shutil.copy(file_path, dest_class_dir)
                print(f"Copied {file_path} to {dest_class_dir}")

# Combine directories
```



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]



[illegible]



[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]



[illegible]



[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]





[illegible]

[illegible]



[illegible]

[illegible]



[illegible]

[illegible]



[illegible]

[illegible]



[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



```
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_009.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_018.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_031.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_035.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_023.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_024.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_036.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_021.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_017.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Union/Yellow-
Onion/Yellow-Onion_010.jpg to app/GroceryStoreDataset/dataset/test-1/Yellow-
Onion
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]





[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

```
to app/GroceryStoreDataset/dataset/test-1/Cabbage
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Cabbage/Cabbage_002.jpg
to app/GroceryStoreDataset/dataset/test-1/Cabbage
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Cabbage/Cabbage_006.jpg
to app/GroceryStoreDataset/dataset/test-1/Cabbage
Copied app/GroceryStoreDataset/dataset/test/Vegetables/Cabbage/Cabbage_017.jpg
to app/GroceryStoreDataset/dataset/test-1/Cabbage
```

```
[ ]: import tensorflow as tf
import os

base_dir = '/app/GroceryStoreDataset/dataset'
data_dir = os.path.join(base_dir, 'train-1')

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)

train_generator = datagen.flow_from_directory(
    data_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    data_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='categorical',
    subset='validation'
)
```

Found 2142 images belonging to 81 classes.

Found 498 images belonging to 81 classes.

```
[ ]: from tensorflow.keras import layers, models, regularizers

model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Dropout(0.2), # Dropout layer to prevent overfitting

    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.
↪001)),

    layers.Dense(81, activation='softmax')
])

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

model.summary()
```

Model: "sequential\_7"

Layer (type)	Output Shape	Param #
conv2d_28 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_27 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_29 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_28 (MaxPooling2D)	(None, 36, 36, 64)	0
dropout_6 (Dropout)	(None, 36, 36, 64)	0

conv2d_30 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_29 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_31 (Conv2D)	(None, 15, 15, 128)	147,584
max_pooling2d_30 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten_7 (Flatten)	(None, 6272)	0
dense_14 (Dense)	(None, 512)	3,211,776
dense_15 (Dense)	(None, 81)	41,553

Total params: 3,494,161 (13.33 MB)

Trainable params: 3,494,161 (13.33 MB)

Non-trainable params: 0 (0.00 B)

```
[ ]: from tensorflow.keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint('/app/best_model.keras', monitor='val_accuracy',
                             ↪save_best_only=True, mode='max')

history = model.fit(
    train_generator,
    steps_per_epoch=100,
    epochs=45,
    validation_data=validation_generator,
    validation_steps=50,
    callbacks=[checkpoint])
```

Epoch 1/45

100/100 21s 205ms/step -

accuracy: 0.6102 - loss: 1.4122 - val\_accuracy: 0.5703 - val\_loss: 1.5578

Epoch 2/45

100/100 4s 41ms/step -

accuracy: 0.5441 - loss: 1.5215 - val\_accuracy: 0.4880 - val\_loss: 1.7648

Epoch 3/45

100/100 30s 297ms/step -

accuracy: 0.5987 - loss: 1.4462 - val\_accuracy: 0.5241 - val\_loss: 1.6346

Epoch 4/45  
100/100            6s 61ms/step -  
accuracy: 0.5415 - loss: 1.6140 - val\_accuracy: 0.6024 - val\_loss: 1.4650

Epoch 5/45  
100/100            35s 346ms/step -  
accuracy: 0.6229 - loss: 1.3033 - val\_accuracy: 0.6064 - val\_loss: 1.4320

Epoch 6/45  
100/100            6s 55ms/step -  
accuracy: 0.6534 - loss: 1.2212 - val\_accuracy: 0.5924 - val\_loss: 1.3758

Epoch 7/45  
100/100            35s 343ms/step -  
accuracy: 0.5936 - loss: 1.4168 - val\_accuracy: 0.6205 - val\_loss: 1.3556

Epoch 8/45  
100/100            6s 60ms/step -  
accuracy: 0.6489 - loss: 1.3454 - val\_accuracy: 0.6084 - val\_loss: 1.4716

Epoch 9/45  
100/100            38s 369ms/step -  
accuracy: 0.6050 - loss: 1.4303 - val\_accuracy: 0.6285 - val\_loss: 1.3304

Epoch 10/45  
100/100            8s 84ms/step -  
accuracy: 0.6398 - loss: 1.3418 - val\_accuracy: 0.6325 - val\_loss: 1.3841

Epoch 11/45  
100/100            36s 352ms/step -  
accuracy: 0.6590 - loss: 1.1970 - val\_accuracy: 0.6024 - val\_loss: 1.3691

Epoch 12/45  
100/100            6s 57ms/step -  
accuracy: 0.6652 - loss: 1.1598 - val\_accuracy: 0.6064 - val\_loss: 1.4429

Epoch 13/45  
100/100            44s 430ms/step -  
accuracy: 0.6849 - loss: 1.1366 - val\_accuracy: 0.6365 - val\_loss: 1.2679

Epoch 14/45  
100/100            6s 63ms/step -  
accuracy: 0.7435 - loss: 1.1367 - val\_accuracy: 0.6446 - val\_loss: 1.2836

Epoch 15/45  
100/100            37s 362ms/step -  
accuracy: 0.6574 - loss: 1.2256 - val\_accuracy: 0.5944 - val\_loss: 1.4612

Epoch 16/45  
100/100            6s 61ms/step -  
accuracy: 0.6827 - loss: 1.1419 - val\_accuracy: 0.6647 - val\_loss: 1.2747

Epoch 17/45  
100/100            38s 369ms/step -  
accuracy: 0.6683 - loss: 1.1807 - val\_accuracy: 0.6627 - val\_loss: 1.3001

Epoch 18/45  
100/100            6s 63ms/step -  
accuracy: 0.6889 - loss: 1.1407 - val\_accuracy: 0.6747 - val\_loss: 1.2217

Epoch 19/45  
100/100            37s 359ms/step -  
accuracy: 0.7307 - loss: 0.9643 - val\_accuracy: 0.6506 - val\_loss: 1.3233

Epoch 20/45  
100/100 6s 58ms/step -  
accuracy: 0.6845 - loss: 1.1059 - val\_accuracy: 0.6566 - val\_loss: 1.3465

Epoch 21/45  
100/100 72s 717ms/step -  
accuracy: 0.6867 - loss: 1.1032 - val\_accuracy: 0.7048 - val\_loss: 1.1269

Epoch 22/45  
100/100 4s 41ms/step -  
accuracy: 0.7249 - loss: 0.9983 - val\_accuracy: 0.7008 - val\_loss: 1.1912

Epoch 23/45  
100/100 21s 204ms/step -  
accuracy: 0.7301 - loss: 0.9934 - val\_accuracy: 0.6386 - val\_loss: 1.3453

Epoch 24/45  
100/100 4s 43ms/step -  
accuracy: 0.6626 - loss: 1.1325 - val\_accuracy: 0.6928 - val\_loss: 1.1811

Epoch 25/45  
100/100 27s 263ms/step -  
accuracy: 0.7019 - loss: 1.0836 - val\_accuracy: 0.7108 - val\_loss: 1.2085

Epoch 26/45  
100/100 5s 48ms/step -  
accuracy: 0.7238 - loss: 1.0376 - val\_accuracy: 0.6205 - val\_loss: 1.3218

Epoch 27/45  
100/100 43s 426ms/step -  
accuracy: 0.7181 - loss: 1.0430 - val\_accuracy: 0.6847 - val\_loss: 1.2522

Epoch 28/45  
100/100 7s 66ms/step -  
accuracy: 0.7194 - loss: 0.9971 - val\_accuracy: 0.6847 - val\_loss: 1.2191

Epoch 29/45  
100/100 38s 373ms/step -  
accuracy: 0.7533 - loss: 0.9752 - val\_accuracy: 0.7088 - val\_loss: 1.1139

Epoch 30/45  
100/100 6s 63ms/step -  
accuracy: 0.7348 - loss: 1.0064 - val\_accuracy: 0.7269 - val\_loss: 1.1080

Epoch 31/45  
100/100 37s 360ms/step -  
accuracy: 0.7548 - loss: 0.9380 - val\_accuracy: 0.7289 - val\_loss: 1.1261

Epoch 32/45  
100/100 6s 59ms/step -  
accuracy: 0.7162 - loss: 0.9320 - val\_accuracy: 0.7149 - val\_loss: 1.1157

Epoch 33/45  
100/100 37s 365ms/step -  
accuracy: 0.7475 - loss: 0.9434 - val\_accuracy: 0.6767 - val\_loss: 1.2327

Epoch 34/45  
100/100 6s 63ms/step -  
accuracy: 0.7562 - loss: 0.9519 - val\_accuracy: 0.6145 - val\_loss: 1.4354

Epoch 35/45  
100/100 39s 387ms/step -  
accuracy: 0.7394 - loss: 0.9539 - val\_accuracy: 0.7149 - val\_loss: 1.1225



```

Epoch 36/45
100/100          6s 61ms/step -
accuracy: 0.8053 - loss: 0.8085 - val_accuracy: 0.7108 - val_loss: 1.1247
Epoch 37/45
100/100          38s 375ms/step -
accuracy: 0.7675 - loss: 0.8752 - val_accuracy: 0.6988 - val_loss: 1.2112
Epoch 38/45
100/100          7s 66ms/step -
accuracy: 0.7609 - loss: 0.9592 - val_accuracy: 0.7129 - val_loss: 1.1490
Epoch 39/45
100/100          35s 342ms/step -
accuracy: 0.7543 - loss: 0.9527 - val_accuracy: 0.7209 - val_loss: 1.1347
Epoch 40/45
100/100          6s 56ms/step -
accuracy: 0.7586 - loss: 0.9887 - val_accuracy: 0.6968 - val_loss: 1.1398
Epoch 41/45
100/100          35s 339ms/step -
accuracy: 0.7590 - loss: 0.9442 - val_accuracy: 0.6807 - val_loss: 1.2618
Epoch 42/45
100/100          5s 54ms/step -
accuracy: 0.7548 - loss: 0.9268 - val_accuracy: 0.7289 - val_loss: 1.0683
Epoch 43/45
100/100          34s 331ms/step -
accuracy: 0.7863 - loss: 0.8414 - val_accuracy: 0.7169 - val_loss: 1.1889
Epoch 44/45
100/100          6s 58ms/step -
accuracy: 0.7818 - loss: 0.8821 - val_accuracy: 0.7149 - val_loss: 1.1620
Epoch 45/45
100/100          34s 337ms/step -
accuracy: 0.7892 - loss: 0.8597 - val_accuracy: 0.6847 - val_loss: 1.2406

```

```

[ ]: from tensorflow.keras.models import load_model
    from tensorflow.keras.preprocessing.image import ImageDataGenerator

best_model = load_model('best_model.keras')
test_datagen = ImageDataGenerator(rescale=1.0/255)
base_dir = '/app/GroceryStoreDataset/dataset'
data_dir = os.path.join(base_dir, 'test-1')

test_generator = test_datagen.flow_from_directory(
    data_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='categorical',
    shuffle=False # Important to not shuffle the data for evaluation
)
# Get the ground truth labels

```

```

true_labels = test_generator.classes

# Get the class indices
class_indices = test_generator.class_indices

# Get the list of class names
class_names = list(class_indices.keys())

# Predict on the test data
predictions = model.predict(test_generator, steps=len(test_generator))
predicted_labels = predictions.argmax(axis=-1)

```

Found 2485 images belonging to 81 classes.  
125/125                      10s 78ms/step

```

[ ]: from sklearn.metrics import accuracy_score, f1_score, classification_report

# Calculate accuracy
accuracy = accuracy_score(true_labels, predicted_labels)

# Calculate F1 score (macro, micro, or weighted)
f1 = f1_score(true_labels, predicted_labels, average='weighted')

# Print classification report
report = classification_report(true_labels, predicted_labels,
    ↪target_names=class_names)

print(f'Accuracy: {accuracy}')
print(f'F1 Score: {f1}')
print('Classification Report:')
print(report)

```

Accuracy: 0.4261569416498994  
F1 Score: 0.4084897943061138  
Classification Report:

	precision	recall	f1-score	support
Alpro-Blueberry-Soyghurt	0.71	0.18	0.29	28
Alpro-Fresh-Soy-Milk	1.00	0.21	0.35	28
Alpro-Shelf-Soy-Milk	0.52	0.77	0.62	30
Alpro-Vanilla-Soyghurt	0.00	0.00	0.00	19
Anjou	0.02	0.03	0.02	35
Arla-Ecological-Medium-Fat-Milk	0.45	0.52	0.48	29
Arla-Ecological-Sour-Cream	0.63	0.52	0.57	23
Arla-Lactose-Medium-Fat-Milk	1.00	0.92	0.96	25
Arla-Medium-Fat-Milk	0.82	0.41	0.55	34
Arla-Mild-Vanilla-Yoghurt	0.11	0.07	0.09	27
Arla-Natural-Mild-Low-Fat-Yoghurt	0.67	0.43	0.53	23

Arla-Natural-Yoghurt	0.78	0.98	0.87	41
Arla-Sour-Cream	0.58	0.39	0.47	18
Arla-Sour-Milk	0.72	0.95	0.82	19
Arla-Standard-Milk	0.88	0.93	0.90	30
Asparagus	0.00	0.00	0.00	14
Aubergine	0.94	0.68	0.79	22
Avocado	0.22	0.20	0.21	40
Banana	0.62	0.36	0.46	44
Beef-Tomato	0.28	0.70	0.40	10
Bravo-Apple-Juice	0.96	0.96	0.96	23
Bravo-Orange-Juice	0.76	0.61	0.68	31
Brown-Cap-Mushroom	0.12	0.08	0.10	39
Cabbage	0.50	0.11	0.17	19
Cantaloupe	0.43	0.59	0.49	39
Carrots	0.57	0.62	0.59	42
Conference	0.14	0.32	0.20	44
Cucumber	0.81	0.63	0.71	27
Floury-Potato	0.17	0.06	0.09	16
Galia-Melon	0.47	0.62	0.53	32
Garant-Ecological-Medium-Fat-Milk	0.74	0.89	0.81	35
Garant-Ecological-Standard-Milk	0.28	1.00	0.43	11
Garlic	0.59	0.68	0.63	25
Ginger	0.00	0.00	0.00	15
God-Morgon-Apple-Juice	0.55	0.58	0.56	19
God-Morgon-Orange-Juice	0.49	0.91	0.63	22
God-Morgon-Orange-Red-Grapefruit-Juice	0.45	0.53	0.49	19
God-Morgon-Red-Grapefruit-Juice	0.47	0.50	0.48	14
Golden-Delicious	0.28	0.31	0.29	45
Granny-Smith	0.33	0.09	0.14	58
Green-Bell-Pepper	0.92	0.48	0.63	25
Honeydew-Melon	0.28	0.53	0.37	36
Kaiser	0.00	0.00	0.00	29
Kiwi	0.17	0.04	0.07	45
Leek	0.52	0.81	0.63	21
Lemon	0.03	0.02	0.03	41
Lime	0.62	0.43	0.51	30
Mango	0.40	0.06	0.11	31
Nectarine	0.54	0.54	0.54	35
Oatly-Natural-Oatghurt	0.68	0.43	0.53	30
Oatly-Oat-Milk	0.62	0.58	0.60	31
Orange	0.25	0.45	0.32	56
Orange-Bell-Pepper	0.57	0.81	0.67	26
Papaya	0.04	0.05	0.04	21
Passion-Fruit	0.17	0.41	0.24	27
Peach	0.25	0.11	0.15	36
Pineapple	0.27	0.24	0.26	25
Pink-Lady	0.51	0.68	0.58	59
Plum	0.65	0.68	0.67	22

Pomegranate	0.21	0.16	0.18	25
Red-Beet	0.23	0.41	0.30	17
Red-Bell-Pepper	0.50	0.18	0.27	33
Red-Delicious	0.80	0.24	0.37	50
Red-Grapefruit	0.00	0.00	0.00	34
Regular-Tomato	0.71	0.89	0.79	47
Royal-Gala	0.14	0.12	0.13	64
Satsumas	0.14	0.12	0.13	68
Solid-Potato	0.32	0.63	0.42	27
Sweet-Potato	0.44	0.56	0.49	27
Tropicana-Apple-Juice	0.82	0.50	0.62	28
Tropicana-Golden-Grapefruit	0.85	0.89	0.87	19
Tropicana-Juice-Smooth	0.82	0.75	0.78	24
Tropicana-Mandarin-Morning	0.59	0.65	0.62	20
Valio-Vanilla-Yoghurt	0.49	0.77	0.60	31
Vine-Tomato	0.44	0.53	0.48	43
Watermelon	0.46	0.13	0.20	46
Yellow-Bell-Pepper	0.37	0.27	0.31	26
Yellow-Onion	0.32	0.49	0.38	37
Yoggi-Strawberry-Yoghurt	0.51	0.66	0.58	32
Yoggi-Vanilla-Yoghurt	0.00	0.00	0.00	18
Zucchini	0.35	0.59	0.44	29
accuracy			0.43	2485
macro avg	0.46	0.45	0.42	2485
weighted avg	0.45	0.43	0.41	2485